Докторска дисертација

Перформанси и безбедност кај пресметување во облак

Сашко Ристов

Универзитет Св. Кирил и Методиј
Факултет за информатички науки и компјутерско инженерство
Скопје, Македонија

Ментор: Проф. Марјан Гушев

Перформанси и безбедност кај пресметување во облак

Сашко Ристов

29 август 2012 г.

Истражувањето за оваа докторска дисертација е спроведено на Универзитетот „Свети Кирил и Методиј“, Факултетот за информатички науки и компјутерско инженерство во Скопје, Република Македонија. Почнувајќи како продолжение на истражувањето спроведено во мојот магистерски труд со наслов „Анализа на безбедноста на веб сервиси и нејзино влијание врз перформансите на веб серверите“ во насока на загубата на перформанси и безбедносни предизвици кај пресметување во облак, истражувањето се втурна во длабока и детална анализа на архитектурата на повеќе процесорските системи и нивното искористување во пресметување со високи перформанси кај пресметување во облак.

Критичен момент за ова интензивно истражување од нецела година беше следењето на курсот за докторанти „Податочни структури и пресметување со високи перформанси (Data Structures and High Performance Computing)“ каде што еден од предавачите беше и мојот ментор професор Марјан Гушев. Продолжувајќи го истражувањето во областа на пресметување со високи перформанси добивме суперлинеарно забрзување за множење матрици на мултипроцесор со споделена меморија. Од овој значаен момент го интензивирав истражувањето и работевме заедно со активно вклучување на мојот ментор во следните неколку месеци.

Вовед

Во минатото беше тешко да се најде потребната информација. Многу повеќе енергија се трошеше за да се пренесе информацијата отколку нејзината обработка. Со појавата на Интернет и подобрувањето на податочните комуникации информацијата стана подостапна со што се зголеми и потребата за поголеми ресурси за обработка на информациите, особено за сложени пресметки.

Употребата на паралелизацијата и пресметување со високи перформанси нагло се зголеми кога моќните компјутерски ресурси почнаа да пресметуваат во разумно време. Точната временска прогноза за следните неколку дена може да се пресмета за еден ден. Суперкомпјутерот на ИБМ Дипблу го победи тогашниот светски шампион во шах Каспаров играјќи по правилата за турнир. Многу софтверски апликации ги предвидуваат движењата на берзите. Сите овие барања мораат да се извршат точно и прецизно за кратко време.

Постојат многу механизми за да се забрза извршувањето. На пример, подобрување на одреден алгоритам намалувајќи ги бројот на пресметки и програмски чекори на ист компјутерски систем; или извршувањето на ист алгоритам со ист број на операции за помалку време на побрзи компјутери. Воведувањето на различни начини на паралелно извршување и пресметување со високи перформанси се модерни механизми денеска за побрзо извршување на одредена програма.

Постојат два различни пристапи за ХПЦ: грид пресметувања и суперкомпјутери. И двата пристапи користат огромен број на мултипроцесори но различно организирани. Кај архитектурата грид тие се дистрибуирани и „лабаво поврзани“ (loosely coupled), т.е. како кластер од кластери, додека кај суперкомпјутерите се „цврсто поврзани“ (tightly coupled), т.е. тие се блиску еден до друг.

Сепак и двата пристапи повеќе се достапни за универзитетите и научните организации отколку за бизнис компаниите. Нивната работна околина е различна од случај до случај и затоа проблемите мораат да се редизајнираат според соодветната спецификација за секој посебно.

Воведувањето на концептот пресметување во облак ги направи ресурсите подостапни и поблиску до корисниците. Исто така тој нуди проширливи, флексибилни и неограничени ресурси за пресметување како и грид и суперкомпјутерите. Но тој нуди и многу повеќе. Корисниците можат да изнајмат произволен број на виртуелни машни и секоја од нив со произволни ресурси и платформи според нивните потреби. Уште повеќе, корисниците можат да си креираат свои виртуелни машини каде што нивните апликации ќе работат перфектно и ќе ги пренесат да се извршуваат во облак.

Работејќи со големи податоци и нивна миграција од корисниците кон облакот и обратно, покрај постоечките безбедносни предизвици произлегуваат и нови. Поради фактот што апликациите и нивните податоци излегуваат од обезбеденатa зона на корисникот, најважен дел е да се обезбеди информациска сигурност. Постојат неколку отворени прашања како несогласувања со законската регулатива, довербата и приватноста на податоците. Повеќето од добавувачите на услуги во облак (Cloud

Service Providers - CSP) најмногу се фокусирани на безбедноста на нивните корисници и речиси сите имаат сертификати за безбедност за нивните јавни облаци [16]. Уште повеќе, тие им нудат на своите корисници и алатки за проверка на безбедноста дали нивните сервиси кои се поставени во облакот на CSP се во согласност со одреден безбедносен стандард.

Опис на проблемот и целите

Целта на оваа докторска дисертација е да спроведе детална анализа на перформансите и безбедноста кај пресметување во облак. За да се постигне оваа цел дисертацијата има четири области на истражување: 1) Основни концепти за облак и ХПЦ, 2) анализа на перформанси на кеш интензивни алгоритми и веб сервиси поставени во традиционална околина, 3) анализа на перформанси кога се поставени во традиционална околина и во облак со различни слоеви во облакот и различни стандарди за безбедност на веб сервиси, и на крај 4) евалуации на безбедност и безбедносни предизвици, проверка на ризици и одржување на континуитет на бизнисот кај деловните информациски системи.

Целта на првата област на истражување за основните концепти на облак и ХПЦ е да не запознае со концептот пресметување во облак, перформансите и користените алгоритми и нивните имплементации во оваа дисертација. По приложувањето на основните концепти, целта на следните две области е да ги анализира намалувањето на перформансите што ги предизвикува облакот и воведувањето безбедност. Целта на последната област е да презентира неколку методологии за евалуација на безбедноста за различни слоеви од облакот и решенијата за облак со отворен код.

Основните прашања за истражување на оваа дисертација се:

- Дали дополнителниот слој од виртуелизацијата ги намалува перформансите?
- Кои се намалувањата на перформансите на различни сервиси и апликации при нивна миграција во облак?
- Дали постојат проблеми за пресметување кои се извршуваат подобро во облак отколку во традиционална околина на страна на корисникот?
- Дали постојат некои региони каде што се постигнува суперлинеарно забрзување во облак?

- Кои безбедносни предизвици постојат за деловните информациски системи доколку мигрираат во облак?
- Кои слоеви во облакот имаат поголема важност отколку традиционалните околини за контролните цели за безбедност
- Кое решение за облак со отворен код е најсоодветно за развој и одржување на систем за управување со информациска сигурност?

Опсег на дисертацијата

Опсегот на оваа дисертација е определување на влијанието на различните слоеви во облакот IaaS, PaaS и SaaS врз перформансите и безбедноста на кеш интензивните алгоритми и веб сервиси. Оваа студија се концентрира на избор на најдобрата распределба на ресурсите и најсоодветната платформа за да се добијат најдобри перформанси за иста цена.

Оваа дисертација го моделира влијанието на инфраструктурата во облакот врз перформансите на различните типови, број и големина на влезни пораки кои ја оптеретуваат истата. Исто така дефинира и квантитативни индикатори за определување на ризикот од мигрирање на сервисите во облак за различна големина на пораките и различен број на конкурентни пораки.

Методи

За да се обработат прашањата за истражување од областа на перформансите поставени се неколку различни инфраструктури со три различни платформи: традиционална (домаќин), виртуелна (гостин) и облак со отворен код. Исто така, различни платформи и нфраструктури се искористени на комерцијалниот облак Windows Azure.

Како тестни податоци се изработени различни веб сервиси кои се поставени во облак и на традиционална платформа, како и неколку различни референтни (benchmark) апликации за секвенцијална и паралелна имплемнетација на пресметковно, мемориско и кеш интензивниот алгоритам за множење матрици.

Најпрвин е извршена детална теоретска анализа за да се определат номиналните перформанси во традиционална околина. Потоа се реализирани огромен број на експерименти за различни влезни параметри и различно оптеретување за да се измерат перформансите на алгоритми-

те. На крај е реализирана анализа и класификација на теоретските и експерименталните резултати.

За истражувањето од областа на безбедност извршен е преглед на главните меѓународни и индустриски стандарди што ја обработуваат безбедноста, како и анализа на нивната применливост кај пресметување во облак. Изработени се неколку евалуации за безбедност кај пресметување во облак користејќи ја како основа серијата на стандарди ИСО 27000.

Содржина на дисертацијата

Оваа дисертација се состои од 6 делови: 1) Основни концепти за перформансите, множење матрици и пресметување во облак, 2) Подобрувања на алгоритамот за множење матрици, 3) Теоретски и практичен доказ за суперлинеарно забрзување на мултипроцесор и графички карти, 4) Анализа на перформансите на кеш интензивен алгоритам кај пресметување во облак, 5) Анализа на перформансите на веб сервисите кај пресметување во облак и 6) Безбедносни предизвици и евалуации кај пресметување во облак.

Во Дел I се прикажани основните концепти и дефиниции за потребите на оваа дисертација. Тој се состои од пет глави. Глава 1 ги опишува основните концепти и архитектурата на облакот, моделите за поставување и нивоата на услуги. Основните поими за перформанси и ограничувањата се дефинирани и елаборирани во Глава 2. Посебен осврт на мемориската хиерархија и воведувањето на кеш меморијата во модерните мултипроцесори е даден во Глава 3 бидејќи пристапот до меморија е тесно грло на сите пресметувања и влијае на перформансите. Глава 4 ги презентира основните секвенцијални и паралелни имплементации на алгоритамот за густо множење на матрици на мултипроцесор и графички карти како пресметковно, мемориско и кеш интензивен алгоритам. На крај, Глава 5 врши краток преглед на веб сервисите, факторите за нивните перформанси и неколку предизвици.

Делот II во две глави го анализира и предлага подобрувања на алгоритамот за множење на матрици и презентира теоретска анализа и експериментален доказ за падот на перформансите кој се појавува поради употреба на сет асоцијативен кеш за кеш интензивни алгоритми. Глава 6 дефинита просечен вкупен број на циклуси по инструкција и анализира која политика за замена во кеш е најоптимална за алгоритамот за множење матрици. Во Глава 7 го презентираме симулаторот ММКешСим кој ги предвидува перформансите на алгоритамот за множење матрици на одреден постоечки или непостоечки повеќепроцесорски систем. Глава

8 презентира две верзии на нов хибриден 2Д/1Д алгоритам за множење матрици со декомпозиција кој што го подобрува основниот алгоритам за множење матрици со 2Д декомпозиција. Глава 9 го анализира и моделира падот на перформансите за одредена големина на влезните податоци поради сет асоцијативен кеш. Исто така ги презентира и големите падови забележани при експериментите за забрзувањето при паралелно извршување и брзината при секвенцијално и паралелно извршување.

Делот III се состои од две глави и го обработува добиеното суперлинеарно забрзување кај множење матрици. Глава 10 ги презентира теоретските анализи со експериментален доказ за тоа зошто, како и кога може да се добие суперлинеарно забрзување при множење матрици на мултипроцесор. Во Глава 11 анализираме како да се добие суперлинеарно забрзување при множење матрици на НВИДИА (NVIDIA) графички картички со конфигурирачки кеш и ги прикажуваме резултатите од експериментите за брзината и забрзувањето во зависност од побарувањето за кеш меморија и број на СМ-и (SMs).

Следниот Дел IV е наменет за перформансите на алгоритамот за множење на густи матрици во различни околини во облак. Се состои од 5 глави. Глава 12 ја побива хипотезата дека дополнителниот слој од виртуелизацијата придонесува за полоши перформанси, т.е. презентира дека постои регион за одредена големина на влезните податоци за множење матрици каде што програмата работи побрзо во виртуелна околина, и при секвенцијална и при паралелна имплементација. Главите 13 и 14 прикажуваат како различните слоеви на сервиси PaaS и SaaS влијаат на перформансите на алгоритамот за множење матрици соодветно. Меѓусебното влијание на станарите во облакот врз перформансите на алгоритамот за множење матрици е презентирано во Глава 15. Конечно, постоењето на суперлинеарно забрзување при паралелно извршување на алгоритамот за множење матрици е дадено во Глава 16.

Делот V ги покрива разликите во перформансите за различни веб сервиси кога се поставени во традиционална околина и во облак за оптеретувања на серверот со различни влезни параметри. Се состои од три глави. Глава 17 ги прикажува резултатите од експериментите за падот на перформансите кога веб сервисите се поставени во облак за различно оптеретување на серверот. Следните две глави 18 и 19 предлагаат две стратегии за подобрување на перформансите на веб сервисите кога се поставени во облак.

Последниот Дел VI ги покрива безбедносните предизвици и кај пресметување во облак и врши неколку евалуации на безбедноста во четири глави. Во Глава 20 го презентираме падот на перформансите поради имплементација на стандарди за безбедност на веб сервисите за да ја по-

добриме нивната безбедност во нивната нова околина - во облакот. Глава 21 врши преглед на основните меѓународни и индустриски стандарди кои ја опфаќаат информацискатa сигурност и ја анализираме нивната применливост со предизвиците за безбедност кај пресметување во облак. Глава 22 ги адресира ризиците од безбедносните предизвици во облакот со пристап од високо ниво базиран на ризици за да го подобри континуитетот на бизнисот на клиентот доколку ги мигрира своите податоци во облак. Глава 23 предлага нови методологии за безбедносна евалуација доколку сервисите на корисникот се поставени во неговите простории или во облак или во различни слоеви на сервиси IaaS, PaaS или SaaS со ИСО 27001:2005 контролните цели како основа за евалуацијата. Глава 24 ги презентира безбедносните евалуации за решението за облак со отворен код ОпенСтек (OpenStack) и три други најраспространети решенија за облак со отворен код.

Главни резултати

Многу резултати од истражувањата за оваа докторска дисертација беа објавени во рецензирани конференции и журнали. Најважните резултати се во областите на перформанси и безбедност. Исто така постојат многу детални анализи, критики на објавени статии, теоретски анализи на различни алгоритми во насока на хардверската инфраструктура и платформа, практични експерименти и евалуации.

Сите дадени хипотези се докажани или побиени и теоретски и експериментално. Многу неочекувани резултати беа разгледани и анализирани за време на експериментите и сите се детално објаснети и елаборирани со длабоки анализи.

Главните резултати се елаборирани во следните секции.

Оптимална политика за замена во кеш за алгоритамот за множење матрици

Бидејќи алгоритамот за множење матрици е кеш интензивен, политиката за замена во кешот е следниот важен параметар кој влијае на перформансите по произлегувањето на проблемот со капацитетот на кешот. Бидејќи бројот на операции не зависи од политиката за замена во кешот, ние дефинираме и предлагаме методологија за определување на просечниот борј на мемориски циклуси по инструкција кои ги изведува алгоритамот, затоа што има најголемо влијание врз перформансите.

Резултатите од експериментите кои се објавени од авторите во [1] како дел од истражувањето за оваа докторска дисертација покажуваат оптимална политика за замена во кеш за секвенцијална и паралелни имплементации на алгоритамот за множење на густи матрици.

Ние утврдивме дека и двете политики за замена во кеш, LRU и FIFO, обезбедуваат слична брзина и просечен број на циклуси по инструкција $CPI_T(N)$ за секвенцијално и паралелно извршување. Сепак, резултатите покажуваат дека LRU политиката за замена во кеш обезбедува подобро $CPI_T(N)$ за секвенцијално извршување кај кеш меморија дедицирана по јадро. Паралелното извршување обезбедува најдобро $CPI_T(N)$ кај процесори со споделена меморија и LRU политика на замена, односно LRU произведува поголемо забрзување од FIFO и е посоодветна отколку FIFO политика за замена на кеш за алгоритамот за множење густи матрици.

Високо конфигурлив симулатор ММКешСим

Како дел од истражувањето за оваа докторска дисертација авторот објави во [29, 24] нови методологии и подобрувања на хардверските курсеви користејќи визуелни симулатори, вежби со постепено зголемување на тежината и воведување на работење на хардверски компоненти со што се добија значајни подобрувања во распределбата на оценките и зголемување на интересот на студентите на компјутерски науки за архитектурата на хардверот.

Како понатамошна работа, авторите развија и објавија во [2] високо конфигурлив симулатор - ММКешСим (MMCacheSim). Главната придобивка од ММКешСим е што овозможува симулација на различните процесорски архитектури и да определи која е најдобрата внатрешната архитектура на кешот во процесорот за определена секвенцијална или паралелна имплементација на даден алгоритам. Со негова употреба се олеснува и подобрува учењето и предавањето на компјутерските архитектури и пресметувањето со високи перформанси. ММКешСим овозможува да се конфигурира:

- Хиерархијата меѓу нивоата на кеш меморија, дали да бидат споделени или дедицирани;
- Инклузивноста помеѓу различните нивоа на кеш меморија;
- Големината на кеш меморијата, асоцијативноста, големината на кеш блокот; и
- Политиката на замена, со можност за различни политики за замена за различни нивоа во кешот.

Подобрување на алгоритмот за множење матрици

Алгоритамот со 2Д декомпозиција на матрици кои можат да се сместат во Л1 кешот на процесорот ги намалува промашувањата во кешот бидејќи операциите ќе пристапуваат до податоци сместени само во Л1 кешот. Сепак, како што елабориравме во оваа дисертација, големината на кешот не е единствениот важен параметар кој влијае на перформансите.

Како дел од истражувањето за оваа дисертација предложивме во [9] нов хибриден 2Д/1Д алгоритам со декомпозиција кој го намалува бројот на операции и пристапи до меморија во однос на оригиналниот алгоритам со 2Д декомпозиција објавен неодамна во литературата како оптимизација на кешот. Идеата е да се користат правоаголни наместо квадратни блок матрици со цел намалување на операциите.

Користејќи теориска анализа за да ги искористиме предностите на другите параметри на кешот кои можат да влијаат врз перформансите, предлагаме модифициран хибриден 2Д/1Д алгоритам со декомпозиција кој дури и го подобрува основниот хибриден 2Д/1Д алгоритам со декомпозиција. Уште повеќе, модифицираниот алгоритам е поотпорен на малата асоцијативност на кешот кај процесорите од марка АМД отколку оригиналниот алгоритам со 2Д декомпозиција. Експериментите ги докажуваат теоретските анализи, т.е. и двата наши предложени алгоритми ги надминуваат перформансите на оригиналниот алгоритам со 2Д декомпозиција за големи матрици на АМД Феном (AMD Phenom) процесор.

Анализа на перформансите кај мултипроцесор со сет асоцијативен кеш

Пад на перформансите за одредена големина на влезните параметри се коментирани низ литературата без детално објаснување. Ние ги анализиравме и моделиравме падовите на перформансите за одредена големина на влезните параметри за секвенцијално и паралелно извршување на кеш интензивен алгоритам и резултатите од ова истражување ги објавивме во [11, 6, 13].

Врз основа на теоретската анализа и експериментално истражување заклучивме дека n асоцијативниот кеш може сериозно да влијае врз перформансите. Ги анализиравме и теоретски ги најдовме точките каде што асоцијативноста предизвикува пад на перформансите и предложивме организација на алгоритамот за множење матрици со што би се одбегнале ситуации каде што мапирањето на n сет асоцијативниот кеш ќе користи само мал дел од кешот наместо целиот капацитет на кешот. Реализираното експериментално истражување ги потврди резултатите покажувајќи

реални случаи за пад на перформансите и при секвенцијално и при паралелно извршување.

Презентиран е теоретски доказ за појавување на екстреми на времето на извршување за одреден случај кога се користи асоцијативен кеш за кеш интензивни алгоритми како што е алгоритамот за множење матрици. Големи падови на перформансите се добиени и анализирани за брзината при секвенцијално извршување. Падот на брзината е по изразен за поголеми матрици отколку за помали во критичните точки.

Голем пад на перформансите се добиени и анализирани и за брзината и за забрзувањето при паралелно извршување. Паралелно извршување на алгоритамот резултира со пад на брзината поголем од секвенцијалното извршување. Падот на брзината и забрзувањето при паралелно извршување е незначителен за критичната големина на матрици од L_2 регионот на кеш дедициран по јадро. Значајни падови на на брзината и забрзувањето при паралелно извршување се забележани во L_3 и L_4 регионите, особено при извршување за повеќе јадра. Тие се изразени за експериментите при извршување на $P = 4$ процесори каде брзината $V(4)$ е помала од брзината $V(2)$. За $N = 512$ брзината $V(4)$ е слична како брзината $V(1)$ во разгледаните точки од дадената област. Падовите на брзината за експериментите извршени на $P = 2$ процесори се исто така изразити. Брзината $V(2)$ е слична или дури помала од брзината $V(1)$ во другите точки од дадената област.

Ние заклучивме дека падовите на перформансите се појавуваат поради зголемен број на генерирани промашувања на кешот од последното ниво.

Суперлинеарно забрзување при множење матрици

И покрај Густафсоновиот закон дека максималното скалирано забрзување е p и ограничувањето на Ши (Shi) дека суперлинеарно забрзување е можно само за алгоритми *без постојана структура*, ние добивме суперлинеарен регион за паралелно извршување и реализиравме длабока и опсежна анализа. За потребите на истражувањето за оваа дисертација ги објавивме теоретската анализа и експериментални резултати во [14, 8, 3].

Суперлинеарно забрзување на мултипроцесор

Исто така пробавме да одиме над границите специфицирани во Густафсоновиот закон и тоа не само да најдеме примери за **суперлинеарно забрзување** за множење матрици, туку и да извршиме теоретска анализа како да го добиеме во реален систем со споделена меморија. По-

кажуваме и обезбедуваме докази за постоењето во реален мултипроцесорски систем кој користи кеш меморија. Суперлинеарно забрзување е можно во случаи каде што при секвенцијалното извршување се иницира повеќе промашувања во кешот отколку при паралелното извршување. Ова се случува на пример, во мултипроцесор со споделена меморија со дедицирани кешови по процесор (јадро).

Презентиравме теоретска поткрепа за постоење на суперлинеарно забрзување и воведовме методологија како, кога и каде да се добие. Исто така дефиниравме и методологија за тестирање и реализиравме бројни експерименти за да обезбедимие доказ за суперлинеарно забрзување и неочекуваните високи перформанси.

Експериментите исто така ги потврдија теоретските резултати во [8]. Добиените резултати со високи перформанси и суперлинеарно забрзување се демонстрирани на пример од алгоритамот за множење густи матрици. Можно е во рамките на одреден ранг на големина на матриците, дури и кога се извршува во виртуелна околината.

Суперлинеарно забрзување кај графички картички

Во [3] ние ги презентиравме теоретската анализа и експерименталните резултати за добиеното суперлинеарно забрзување на графички картички, кои исто така се карактеризирани како СИМД. Имплементиравме алгоритам со постојана структура кој ефикасно го искористува споделената кеш меморија и ги избегнува промашувањата во кешот што е можно повеќе.

Експериментите ја потврдија теоретската анализа и суперлинеарните региони од влезните параметри за множењето матрици на графички картички, каде што нормализираните перформанси по процесирачки елемент за паралелно извршување се подобри од оние при секвенцијално извршување.

Понатаму презентиравме и доказ базиран на експериментите за постоење на суперлинеарно забрзување за процесори со СИМД архитектура со дедициран кеш, т.е. графички картички. Сепак, добивме суперлинеарно забрзување само во суперлинеарниот регион каде што побарувањата за кеш меморија можат да се сместат во Л2 кешот за паралелно извршување, а се генерираат промашувања во Л2 кешот при секвенцијално извршување.

Влијание на виртуелизацијата врз перформансите на кеш интензивните алгоритми

Одредена програма би требало да се изврши побавно во виртуелна околина во облак споредено со традиционален сервер поради дополнителниот слој за виртуелизација. Сепак, и покрај оваа хипотеза, експерименталните резултати потврдуваат дека перформансите во виртуелна средина се подобри од традиционалната за дистрибуирана меморија отколку во споделена меморија каде што има голем пад на перформансите.

Реализиравме детални експерименти објавени во [4] за потребите на истражувањето од оваа дисертација. Заклучивме големи неправилности во перформансите за кеш интензивни алгоритми во виртуелна средина. Постои регион за одредена големина на матриците при нивно множење каде што програмата работи побрзо во виртуелна средина.

Виртуелизацијата има речиси исти перформанси од 98% како традиционалната средина при секвенцијално извршување на алгоритамот за множење матрици. При паралелизација се добиваат уште подобри брзини од речиси 15% за одреден број на процесори (јадра) во L_1 и L_2 регионите (дистрибуирани кешови по јадро).

Сепак, виртуелизацијата ја губи битката во L_4 регионот (споделена меморија) при појавување на голем број скапи промашувања во кешот. Нејзините перформанси се за 33.42% полоши од традиционалната средина во овој регион.

Влијанието на PaaS врз перформансите на кеш интензивни алгоритми

Извршувајќи иста програма во различни работни околини обезбедува различни перформанси. Спротивно на хипотезата дека Линукс базираните работни околини обезбедуваат подобри перформанси, резултатите од експериментите објавени во [7] покажуваат дека Windows базираните работни околини во Windows Azure облакот работат подобро од Линукс, особено кога големината на влезните податоци може да се смести во кеш и не се генерираат голем број на промашувања во кешот.

Измерените брзини за истиот алгоритам на Windows се поголеми отколку на Линукс постигнувајќи до 2.5 пати подобри перформанси особено во регионите L_1-L_3. Однесувањето во L_4 регионот е споредливо, но сепак Windows платформата постигнува подобри перформанси.

Влијанието на IaaS врз перформансите на кеш интензивни алгоритми

Цената за изнајмување на ресурси има линеарна зависност од потрошените ресурси, но не секогаш сите понудени ресурси на инстанците од виртуелните машини се најпогодни за клиентите. Истата количина на ресурси понудени од страна на облакот може да се изнајмуваат и да се користат различно за да се забрзаар пресметувањата. Еден начин е да се користат техники за паралелизација на инстанци со повеќе ресурси. Другиот начин е да се раздели работата помеѓу неколку инстанци од виртуелните машини со помалку ресурси. Како дел од истражувањето за оваа дисертација во [21] се анализира кој е најдобриот начин да се скалираат ресурсите за да се забрзуваат пресметувањата и да се добијат најдобри перформанси за истата сума на пари потребни за изнајмување на тие ресурси во Windows Azure облакот.

Се добија очекувани резултати од експериментите за секвенцијално извршување, односно Extra Large виртуелната машина постигнува максимална брзина пред Large, Medium и Small во L_2 регионот. Сепак, Small виртуелната машина постигнува слична брзина како Extra Large виртуелната машина и тие водат пред Large и Medium виртуелната машина во L_4 регионот.

При паралелно извршување се добија почудни резултати. Алгоритамот за множење на густи матрици постигнува максимална брзина при паралелно извршување на 8 x Small инстанци, пред 4 x Medium, 2 x Large, и 1 x Extra Large во L_2 и L_3 регионите и речиси целиот набљудуван L_4 регион. Ова значи дека најдобри перформанси може да се постигнат ако алгоритамот за множење на густи матрици се гранулира на 8 делови и секој дел треба да се изврши на 8 конкурентни процеси со една нишка во Small Windows Azure виртуелна машина. Истата средина постигнува максимално забрзување во L_2 регионот. Во L_3 и L_4 регионот максимално забрзување се постигнува ако алгоритамот за множење на густи матрици е гранулиран на 4 делови и секој дел треба да се изврши на 4 конкурентни процеси со две нишки во Medium Windows Azure виртуелна машина.

Влијанието на повеќестанарството (multitenancy) врз перформансите на кеш интензивни алгоритми

Поради фактот што модерните повеќе јадрени мултипроцесори исто така го споделуваат последното ниво кеш меѓу сите јадра на еден чип, целта е да се овозможи оптимална алокација на ресурси, преку избегнување на промашувања на кешот колку што е можно повеќе, бидејќи тоа ќе

доведе до зголемување на перформансите. Реализираните експерименти објавени во [5] за потребите на истражувањето од оваа дисертација покажуваат дека и алокацијата на сите сингл-станар и мулти-станар во облак обезбедуваат подобри перформанси отколку во просториите на клиентот за одреден обем на работа.

Извршените експерименти адресираат неколку виртуелни машини во облак користејќи различен број на процесори (под претпоставка дека сите јадра се користат). Секој експеримент користи исти ресурси, но различно групирани низ виртуелните машини. Резултатите можат да бидат сумирани како:

- Експериментите потврдија дека постои регион (L_2 регион) каде што облакот постигнува подобри перформанси отколку традиционалната и виртуелната средина, и за паралелно и за секвенцијално извршување, и
- Експериментите потврдија дека пресметување во облак обезбедува подобри перформанси во мулти-виртуелна околина, наместо доделување на сите ресурси на само една виртуелна машина.

Најдобра алокација на ресурси за традиционална средина за кеш интензивни алгоритми е користење на повеќе процеси со една нишка. Повеќе виртуелни машини со една нишка е најдобра алокација на ресурси за во облак. Споредувајќи ги околините, пресметувањето во облак дава најдобри перформанси.

Суперлинеарно забрзување во облак

Резултатите објавени од авторите во [20] како дел од истражувањето за оваа дисертација покажуваат дека облакот исто така може да се постигне суперлинеарно забрзување при извршување на кеш интензивни алгоритми кога се користи пресметување со високи перформанси во виртуелни машини доделени со повеќе од еден процесор (јадро). Забрзување почнува да се зголемува за оние матрици A и B кои не се вклопуваат во Л2 кешот од последното ниво за секвенцијално извршување, т.е. половина од L2 кешот, но во исто време се вклопуваат во целиот Л2 кеш кој е достапен за паралелно извршување со две или четири нишки на две или четири јадра соодветно. Забрзувањето се зголемува до $N = 628$ утврдено теоретски за традиционална средина кога почнуваат да се генерираат промашувања во Л2 кешот. Исто така постојат и точки на свртување на забрзувањето за виртуелна средина и облак поголема од теоретската вредност бидејќи виртуелизацијата обезбедува подобри перформанси

за паралелно извршување отколку секвенцијално во споделена меморија како што објавувивме во [4].

Виртуелна средина и облак постигнуваат подобро забрзување за дедициран кеш и најдобри перформанси се постигнуваат во облак. По регионот L_2, каде се генерираат промашувања во Л2 кешот, традиционалната средина го извршува подобро алгоритамот во споредба со облакот. Виртуелната средина постигнува најлошо забрзување кога алгоритамот бара многу пристапи кон споделената главна меморија.

Експериментите покажуваат различни опсег на забрзувањето. Најширокиот опсег на суперлинеарно забрзување се доби за традиционална средина, додека најтенок за виртуелната средина. Облакот и виртуелната средина имаат поширок опсег на суперлинеарно забрзување за паралелно извршување со два наместо четири нишки бидејќи последното ниво кеш е дедицирано по јадро што е случај каде што виртуелизација обезбедува подобри перформанси отколку споделена меморија како што објавиме во [4] за потребите на истражувањето за оваа дисертација. Опсегот е скратен до 3 пати од десната страна (за големи вредности на N) во споредба со левиот регион (помали вредности за N). Опсегот во традиционална средина се намалува за 80 од левата страна и 120 од десната страна на опсег. Во виртуелната средина се скратува за 56 и 184, а во облакот 88 и 144 од левата и десната страна соодветно. Суперлинеарниот регион е најмногу скратен во виртуелната средина додека најмалку во традиционална средина.

Друг важен резултат е добивањето на суперлинеарно забрзување во виртуелна околина со три различни хипервизори: Microsoft Hyper-V, KVM и VMware ESX.

Перформанси на веб сервиси во облак

Иако најдовме региони каде што облакот е подобра средина за кеш интензивна алгоритми, па дури и со суперлинеарно забрзување при паралелно извршување, тоа не е случај за веб сервисите поставени во облак. Дополнителниот слој поради виртуелизацијата во облакот ги намалува перформансите на веб сервисите. Серија на експерименти се реализирани и објавени во [28] за потребите на истражувањето за оваа дисертација за да се анализираат перформансите на веб сервисите и да се спореди нивото на деградација ако веб сервисите се мигрираат од просториите кај клиентот во облак со користење на истите хардверски ресурси.

Резултатите од експериментите покажуваат дека перформансите директно зависат од големината на влезната порака особено за веб сервис кој побарува големи пресметувања и количество меморија, без оглед на

платформата. Ова не е нагласено за веб сервис кој побарува само меморија.

Исто така дефиниравме квантитативни индикатори за да се утврди ризикот од миграција на сервисите во облакот за различна големина на пораки и број на конкурентни пораки. Заклучокот е дека перформансите се намалени на 71.10 % од традиционалната околина за веб сервисите што побаруваат меморија и 73.86 % за веб сервисите кои побаруваат големи пресметувања и количество меморија ако мигрираат во облак. Облакот најмалку ги намалува перформансите за поголеми пораки без оглед на бројот на конкурентни пораки за веб сервис што побарува меморија. Меѓутоа, најмало намалување на перформансите за веб сервис кој побарува големи пресметувања и количество меморија при мигрирање во облак има за помал број на конкурентни пораки и за поголеми пораки.

Стратегија со посредник за подобрување на перформансите на веб сервисите во облак

Ова решение што динамички се справува со екстремни пресметковни оптоварувања за веб сервисите поставени во облак е предложен во [23] за потребите на истражувањето за оваа дисертација. Тоа воведува нов слој - посредник помеѓу клиентите и серверот кој што динамички ќе покрене дополнителни виртуелни машини по барање штом оптеретувањето ќе го достигне минималното ниво на перформанси што е дефинирано и ќе го проследува пораките низ виртуелните машини. Дополнителните виртуелни машини ќе се исклучат кога оптовареноста на сервисите ќе се врати на номиналната вредност.

И покрај латентноста на едноставни веб сервиси, експериментите покажуваат дека посредникот ги подобрува перформансите на пресметковно интензивни веб сервиси (каде голем дел од времето на извршување се троши на пресметувања).

Трансформација на пораки за подобри перформанси во облак

Иако стратегијата со посредник се справува со екстреми кои се појавуваат како резултат на зголемениот број на конкурентни барања, тоа не се справува добро со екстреми со зголемување на оптоварувањето поради огромна големина на пораките.

Предложивме друга стратегија наречена стратегија со трансформација на пораките во [22] за потребите на истражувањето за оваа дисертација. Ги употребивме заклучоците од [10] и од оваа дисертација дека

Microsoft Windows обезбедува подобри перформанси од Linux Ubuntu за големи пораки, и Linux Ubuntu OS обезбедува подобри перформанси од Microsoft Windows за голем број на конкурентни пораки и за мали пораки.

Ако времето за одговор се зголемува над прагот, тогаш посредникот ќе ги подели влезните параметри во помали парчиња кои Линукс оперативниот систем може да ги процесира побрзо отколку целата порака. Ако екстремот е уште поголем, тогаш посредникот ќе подигне дополнителни инстанци инсталирани со Windows Server оперативен систем и ќе ги пренасочува клиентските барања меѓу двата веб сервиси, целата порака на Windows Server оперативен систем и пораките поделени во помали парчиња на Линукс сервер базиран оперативен систем.

Перформанси на безбедносните мерки во облак

Преместувањето на податоците и апликациите надвор од безбедниот периметар на компанијата наметнува имплементирање на стандарди за безбедност за да се постигне соодветно ниво на крај-до-крај безбедност со информациските системи во просториите на корисникот. Бидејќи веб сервисите се најчесто користена техника, ние ги анализиравме перформансите на воведување на најчестите стандарди за безбедност на веб сервис XML Signature и XML Encryption објавени во [25, 27, 26] за потребите на истражувањето за оваа дисертација.

Резултатите покажуваат дека зголемувајќи ја големината на пораките и бројот на конкурентни пораки ги намалуваат перфромансите на веб сервисите. Различни платформи се исто така анализирани. Linux оперативниот систем се справува подобро со бројот на конкурентни пораки а Windows обратно, т.е. тој се справува подобро со поголеми пораки.

Ги анализиравме и максималниот проток со имплементација само на XML Signature и имплементација и на XML Signature и на XML Encryption. Ги споредивме двете платформи и определивме дека Linux има подобар проток од Windows за мал број на пораки со и без имплементација на безбедност. Со имплементација на безбедност Linux исто така обезбедува подобри перформанси за големи пораки. Windows има подобри перформанси за пораки со средна големина.

Подобрувања на стандардите за безбедност во облак

Направивме преглед на главните меѓународни и индустриски стандарди наменети за безбедност на информациите и ја анализиравме нивната сообразност со безбедносните предизвици во облак. Исто така направивме

и преглед на напорите направени кон стандардизација на безбедност во облак. Резултатите се објавени во [16, 15, 17, 19] за потребите на истражувањето за оваа дисертација.

ИСО 27000 серијата (27001:2005, 27002:2005, and 27005:2011) од стандарди се дефинирани како генерални и ги покриваат не само техничките решенија на технички идентификуваните опасности и слабости, туку ги земаат во предвид и операционите, организациските и менаџмент слабостите исто така. Поради нивната општост, како и многуте отворени предизвици за безбедност во облак, ИСО 27001:2005 не е целосно применлив за систем за информациска безбедност во облак. Затоа ние предлагаме нова контролна цел во барањата на ИСО 27001:2005, *управување со виртуелизацијата*, со две контроли кои ќе покриваат *виртуелизација* и *управување на виртуелните машини*.

Со пристап базиран на ризици од високо ниво ги адресиравме ризиците од безбедносните предизвици во облакот со цел да го подобриме континуитетот на бизнисот на компанијата доколку ги мигрира своите сервиси во облак.

Не најдовме ниеден труд кој ги покрива аспектите на континуитетот на бизнисот во детали за облак и не предизвика да ги адресираме недостатоците на континуитетот на бизнисот за клиентот на облакот: приватност на податоци и нивна заштита, усогласеност со законската регулатива и стандардите, губењето на управување, локација на податоците, хетерогеност, сложеност и интероперабилност, околина со повеќе станари, и опоравување по катастрофа - усогласеност со RPO и RTO.

Воведовме предлози кои го минимизираат влијанието на континуитет на бизнисот и веројатноста за случување инцидент за секој недостаток. Овие главни ризици може да се оценат соодветно и да се ублажат до прифатливо ниво со примена на препораките од овие предлози според матрицата за нивото на ризик како функција од влијанието врз бизнисот и веројатност за случување на инцидент.

Ги адресиравме придобивките од облакот кои го подобруваат континуитетот на бизнисот: елиминирање прекини, подобро управување со безбедност на мрежа и информациите, опоравување по катастрофа со управување ма резервна копија на повеќе места. Тоа исто така го избегнува или елиминира нарушување на работењето, се зголемува достапноста на услугата и ги намалува DoS нападите.

Нови методологии за евалуација на безбедност на страна на корисникот спрема облак

Како дел од оваа дисертација дефиниравме и објавивме во [17, 16] две методологии за проценка на безбедноста на страна кај корисникот или во облак и различните слоеви за услуги во облакот. ИСО 27001:2005 контролните цели се земени како основа за оценувањето.

Првата методологија ги квантифицира барањата на ИСО 27001:2005 групирани во контролни цели, споредувајќи ги страната на корисникот и облакот. Евалуацијата и анализата на ИСО 27001:2005 стандардот резултира во пренесување на влијанието од корисникот кон ЦСП. Истовремено корисникот мора да обезбеди голем напор за да ги имплементира сите контролни цели со намалена важност во SLA договорот со неговиот ЦСП.

Втората методологија ги квантифицира барањата на ИСО 27001:2005 групирани во контролни цели за страната на корисникот и различните слоеви на услуги во облакот. Евалуиравме дека при миграција во облак, 12 од 39 контролни цели се за управување и не зависат дали сервисите се на страна на корисникот или во облак. Факторот на важност не се менува во просек за контролните цели, се намалува на 18, а се зголемува на само две. Според тоа, мигрирајќи во облак, корисниците ја пренесуваат важноста на безбедноста на својот ЦСП и очекуваат дека нивните податоци и апликации ќе бидат безбедни. Затоа, поради новите безбедносни предизвици кои ги продуцира облакот, корисниците мораат да го реевалуираат својот план за континуитет на бизнисот.

Нови методологии за евалуација на безбедност за решенија за облак со отворен код

Сите клучни комерцијални добавувачи на облак поседуваат сертификати за безбедност како компанија. Дополнително сите тие нудат некои услуги за безбедност на своите корисници [16]. Решенијата со отворен код нудат мал број на услуги за безбедност кон своите корисници или во општ случај не нудат.

Ниту безбедносна проценка ниту компаративна анализа на безбедноста на облак не се извршени досега во литературата. Ние предложивме две методологии за евалуација на безбедноста на решенијата со отворен код објавени во [18, 12] како дел од истражувањето за оваа дисертација.

Ги анализиравме работите околу безбедноста која Опенстек решението за облак со отворен код и останатите безбедносни алатки можат да се интегрираат за да се подобри неговата безбедност. Евалуацијата на

безбедноста беше реализирана анализирајќи ги соодветните контролни цели дефинирани во ИСО 27001:2005 и споредувајќи ги со останатите решенија за облак со отворен код.

Резултатите од нашата проценка покажуваат дека Еукалиптус и CloudStack имаат интегрирано максимално ниво на безбедност пред OpenNebula. OpenStack има интегрирано најмалку безбедност во споредба со другите решенија.

Општ заклучок од евалуацијата е дека сите решенија за облак со отворен код се грижат до некое ниво на безбедност. Резултатите од евалуацијата покажа дека CloudStack е најдобар избор од сите решенија за облак со отворен код за да се мигрираат сервисите и има интегрирано максимално ниво на безбедност во својата архитектура. Тоа е во согласност со сите ИСО 27001:2005 11 контролни цели кои зависат од решението за облак. Еукалиптус и OpenNebula, исто така, достигнаа далеку во безбедноста. OpenStack е најлошото решение да се мигрираат сервисите од аспект на безбедност.

Иако решенијата за облак со отворен код внимаваат на безбедноста, компанијата сè уште има и други 28 технички барања, организациски и менаџмент барања кои треба да се усогласат со стандардот. Исто така, ИСО 27001:2005 дефинира општи барања, т.е. одговорност на менаџментот и воспоставување, управување, преглед и подобрување на систем за управување со информациска сигурност.

Применливост на резултатите

Покрај огромниот број на важни резултати кои ги споменавме претходно, вреди да се спомене и нивната применливост.

ХПЦ подобрувања

Реализацијата на модерни процесори се базира на повеќе јадрена архитектура со зголемување на бројот на јадра по процесор што се всушност организирани како мултипроцесор со споделена меморија со заеднички Л2 кеш и дистрибуирани L1 кеш. Затоа, овие резултати ќе имаат влијание врз иднината на развој на софтвер и експлоатација на паралелел хардвер.

Резултатите и методологијата можат да се користат во пресметки на масовни податоци со голема рата на промашувања во кешот. Поделбата на податоците во помали парчиња со оптимална големина пресметани со нашата методологија ги намалува промашувањата во кешто при па-

ралелно извршување во систем со дедициран кеш по јадро. Иако ние ги поедноставивме нашите пресметки, нашата методологија може да се користи во други слични пресметки со високи перформанси.

Како заклучок, математичките релации покажаа можност за суперлинеарно забрзување и голем број експериментални истражувања ги потврдија резултатите покажувајќи реални случаи на неочекувано зголемени перформанси во мултипроцесорски или мултијадрени (или двете) системи, со дедициран кеш по јадро и на тој начин предлагаме употреба на таквите системи во паралелно процесирање.

Избор на платформа за облак

Главниот придонес на оваа дисертација се базира врз експериментални докази и препораки за користење на Windows платформата при користење на Windows Azure облак за кеш интензивни проблеми, и покрај хипотезата дека Линукс оперативниот систем има подобри перформанси од Windows. Со други зборови, хипервизорите имаат огромно влијание на вкупните перформанси, а понекогаш дури и ги подобруваат перформансите.

Виртуелизацијата во облак може дури да ги подобри перформансите

Ние откривме дека перформансите во виртуелна средина е подобра отколку традиционалната средина за дистрибуирана меморија (дедициран кеш по јадро). За споделена меморија постои огромен пад во перформансите во виртуелната средина (L_4 регион). Затоа, ние предлагаме процесори со дедициран кеш по јадро при градење на облаците.

Извршување на ХПЦ апликации во облак

Ние ја дефинираме најдобрата алокација на ресурси меѓу виртуелните машини во облакот за да се постигнат максимални перформанси. Најдобрата алокација на ресурси за традиционална средина за кеш интензивни алгоритми е користење на повеќе процеси со една нишка. Повеќе виртуелни машини со една нишка е најдобра алокација на ресурси во облак.

Најдобри перформанси можат да се добијат ако алгоритамот за множење матрици се гранулира на 8 делови и секој дел да се изврши на 8 конкурентни процеси со по една нишка one во Small Windows Azure виртуелна машина. Истата околина добива најдобро забрзување во L_2 регионот. Во L_3 и L_4 регионите максимално забрзување се постигнува

ако алгоритамот за множење матрици се гранулира на 4 делови и секој дел да се изврши на 4 конкурентни процеси секој со по две нишки во Medium Windows Azure виртуелни машини.

Евалуација на безбедност во облак

Дефиниравме 4 методологии за евалуација на безбедноста ако компанијата ги мигрира сервисите од своите простории во облак, потоа во кој слој од сервиси во облак, кое решение за облак со отворен код и миграција од едно на друго решение за облак со отворен код.

Скопје, *Сашко Ристов*
Август 2012

Литература

1. Anchev, N., Gusev, M., Ristov, S., Atanasovski, B.: Optimal cache replacement policy for matrix multiplication. In: to be published in ICT Innovations 2012. Springer Berlin / Heidelberg (2012)

2. Atanasovski, B., Ristov, S., Gusev, M., Anchev, N.: Mmcachesim: A highly configurable matrix multiplication cache simulator. In: to be published in ICT Innovations 2012, Web Proceedings, Skopje, Macedonia (2012)

3. Djinevski, L., Ristov, S., Gusev, M.: Superlinear speedup in gpu devices. In: to be published in ICT Innovations 2012. Springer Berlin / Heidelberg (2012)

4. Gusev, M., Ristov, S.: Matrix multiplication performance analysis in virtualized shared memory multiprocessor. In: MIPRO, 2012 Proceedings of the 35th International Convention, IEEE Conference Publications. pp. 264–269 (2012)

5. Gusev, M., Ristov, S.: The optimal resource allocation among virtual machines in cloud computing. In: Proceedings of The 3rd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012). pp. 36–42 (2012)

6. Gusev, M., Ristov, S.: Performance gains and drawbacks using set associative cache. Journal of Next Generation Information Technology (JNIT) 3(3) (31 Aug 2012)

7. Gusev, M., Ristov, S.: Superlinear speedup in windows azure cloud. Tech. Rep. IIT:06-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jul 2012)

8. Gusev, M., Ristov, S.: A superlinear speedup region for matrix multiplication. Tech. Rep. IIT:02-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jan 2012)

9. Gusev, M., Ristov, S., Velkoski, G.: Hybrid 2d/1d blocking as optimal matrix-matrix multiplication. In: ICT Innovations 2012. Springer Berlin / Heidelberg (2012)

10. Ristov, S.: Analysis of Web Service Security and its Impact on Web Server Performance. Master's thesis, University Sts Cyril and Methodius, Faculty of Electrical Engineering and Information Technologies, Macedonia (May 2011)

11. Ristov, S., Gusev, M.: Achieving maximum performance for matrix multiplication using set associative cache. In: The 8th Int. Conf. on,Computing Technology and Information Management (ICCM2012), IEEE Conference Publications. ICNIT '12, vol. 2, pp. 542–547 (2012)

12. Ristov, S., Gusev, M.: Open source cloud security audit. Tech. Rep. IIT:08-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jul 2012)

13. Ristov, S., Gusev, M.: Performance gains and drawbacks in multiprocessor using set associative cache. Tech. Rep. IIT:10-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jul 2012)

14. Ristov, S., Gusev, M.: Superlinear speedup for matrix multiplication. In: Information Technology Interfaces, Proceedings of the ITI 2012 34th International Conference on. pp. 499–504 (2012)

15. Ristov, S., Gusev, M., Kostoska, M.: Information security management system for cloud computing. In: ICT Innovations 2011, Web Proceedings, Skopje, Macedonia (2011)

16. Ristov, S., Gusev, M., Kostoska, M.: Cloud computing security in business information systems. International Journal of Network Security & Its Applications (IJNSA) 4(2), 75–93 (2012)

17. Ristov, S., Gusev, M., Kostoska, M.: A new methodology for security evaluation in cloud computing. In: MIPRO, 2012 Proc. of the 35th Int. Convention, IEEE Conference Publications. pp. 1808–1813 (2012)

18. Ristov, S., Gusev, M., Kostoska, M.: Security assessment of openstack open source cloud solution. In: Proceedings of the 7th South East European Doctoral Student Conference (DSC2012) (2012)

19. Ristov, S., Gusev, M., Kostoska, M., Kiroski, K.: Business continuity challenges in cloud computing. In: ICT Innovations 2011, Web Proceedings, Skopje, Macedonia (2011)

20. Ristov, S., Gusev, M., Kostoska, M., Kjiroski, K.: Virtualized environments in cloud can have superlinear speedup. In: ACM Proceedings of 5th Balkan Conference of Informatics (BCI2012) (2012)

21. Ristov, S., Gusev, M., Osmanovic, S., Rahmani, K.: Optimal resource scaling for hpc in windows azure. In: to be published in ICT Innovations 2012, Web Proceedings, Skopje, Macedonia (2012)

22. Ristov, S., Gusev, M., Velkoski, G.: Message transformation to gain maximum web server performance in cloud computing. In: Proceedings of the Conference of Informatics and Information Technology CiiT (2012)

23. Ristov, S., Gusev, M., Velkoski, G.: A middleware strategy to survive peak loads in cloud. In: Proceedings of the Conference of Informatics and Information Technology CiiT (2012)

24. Ristov, S., Stolikj, M., Ackovska, N.: Awakening curiosity - hardware education for computer science students. In: MIPRO, 2011 Proceedings of the 34th International Convention, IEEE Conference Publications. pp. 1275 –1280 (may 2011)

25. Ristov, S., Tentov, A.: Performance comparison of web service security on windows platform – message size vs concurrent users. In: X International Conference ETAI 2011 (2011)

26. Ristov, S., Tentov, A.: Security based performance issues in agent-based web services integrating legacy information systems. In: CEUR Workshop Proceedings. WASA 2011, vol. 752, pp. 45–51 (2011)

27. Ristov, S., Tentov, A.: Performance impact correlation of message size vs. concurrent users implementing web service security on linux platform. In: ICT Innovations 2011. Advances in Intelligent and Soft Computing, vol. 150, pp. 367–377. Springer Berlin / Heidelberg (2012)

28. Ristov, S., Velkoski, G., Gusev, M., Kjiroski, K.: Compute and memory intensive web service performance in the cloud. In: to be published in ICT Innovations 2012. Springer Berlin / Heidelberg (2012)

29. Stolikj, M., Ristov, S., Ackovska, N.: Challenging students software skills to learn hardware based courses. In: Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on. pp. 339 –344 (june 2011)

Sasko Ristov

Performance and Security in Cloud Computing

PhD Thesis

August 29, 2012

Dedicated to my father Kiro who is
unfortunately not among us long time ago,
but would be very proud with this thesis.

Preface

The PhD research was realized at the Ss. Cyril and Methodius University, Faculty of Information Sciences and Computer Engineering in Skopje, Macedonia. The motivation was based on my M.Sc. thesis research titled "Analysis of Web Service Security and its Impact on Web Server Performance" with goal to continue in the world of performance penalties and security challenges in cloud computing. The research was extended into in-depth and comprehensive analysis of multiprocessor architectures and their exploitation in high performance computing and cloud computing environment.

A critical point for this intensive research of almost 9 months was learning the PhD Course "Data Structures and High Performance Computing" where one of the teachers was my supervisor professor Marjan Gusev. Continuing the research in the field of HPC, we have achieved a superlinear speedup for matrix multiplication algorithm executions on a shared memory multiprocessor. From this crucial moment I changed several gears forward and was working together with active involvement of my supervisor in the next period.

Background

Some time ago it was difficult to find the information that we need. More efforts were performed to transfer the information rather than their computation. Internet and improved data communications enabled more information available online and thus increase the computing resources requirements.

Usage of parallelization or high performance computing increased enormous when the powerful computing resources begin to compute in reasonable time. Consistent weather forecast for the next few days can be performed in a day. First computer program on IBM Deep Blue supercomputer defeated the chess world champion Kasparov in a match organized by tournament regulations [66]. Many software application predict the stock exchange rates.All these requirements must be accurately and precisely executed in proper time.

There are several mechanisms to speedup the execution. For example, improving particular algorithm to reduce the computation and program steps on the same computer system; or executing the same algorithm with the same number of operations for less time on faster computers. Introducing different versions of parallelism and high performance computing are todays' modern mechanisms for faster program execution.

There are two different approaches in HPC: 1) grid computing and 2) supercomputers. Both approaches use massive number of multiprocessors orchestrated differently. While grid computing organizes the multiprocessors distributed and loosely coupled, i.e. as cluster of clusters, the multiprocessors in supercomputer are "tightly" coupled, i.e. close to each other.

However, both HPC solutions, the grid computing and supercomputers are mainly available for universities and scientific organizations, rather than for business and companies. Their runtime environment is unique and the problems should be redesigned according to the specifications.

Introducing the cloud computing paradigm made the resources more available and more closely to the consumers. It also offers scalable, flexible and infinite available computing resources as grid and supercomputers. It offers even more features. The customers can rent an arbitrary number of virtual machines each with arbitrary resources as they need with different platforms. Even more, the customers can create their own virtual machines where their application work perfectly and upload in the cloud for execution.

New security challenges arise when working with huge amount of data and migrating in the cloud, beside the existing ones. Since the applications and data are moving outside of the customer security perimeter, the most important part is to ensure certain information security. Several open issues exist like regulatory violation, trust and data privacy. Most common cloud service providers (CSPs) are mostly focused on customer security and almost all have security certificates or compliances for their public cloud infrastructure [126]. Even more, they offer security features to their customers to assess whether their services hosted in the CSP cloud are compliant to particular security standard.

Problem Description and Objectives

The goal of this thesis is to conduct a comprehensive analysis of performance and security in cloud computing. To achieve this objective the thesis follows four research areas: 1) Basic cloud computing and HPC performance concepts, 2) Performance analysis of cache intensive algorithms and web services on-premise, 3) Performance analysis when hosted on-premise and in the cloud with different cloud service layers and different web service security standards, and finally 4) High-level cloud computing security challenges and evaluations, risk assessment and business continuity for business information systems.

The purpose of first research area for basic cloud computing and HPC performance concepts is to introduce the concept of cloud computing, performance and used algorithms and their implementations in this thesis. Once the basic concepts are finished, the purpose of the next two research areas is to analyze the amount of performance penalties that cloud and security provide. The purpose of the final area is to present several methodologies for security evaluation of different cloud computing service layers and open source cloud solutions.

The main research questions of this thesis are:

- Is the additional cloud virtualization layer providing performance discrepancy and drawbacks?
- What are the penalties when different services and application migrate from on-premise to the cloud?
- Are there any computing problems that run better in the cloud than on-premise?
- Is there a region where a superlinear speedup can be achieved in the Cloud?
- What are the security challenges for business information systems if they migrate in the cloud?
- Which cloud service layer provides better importance for security control objectives than on-premise?
- What open source cloud solution is the most appropriate to develop and maintain information security management system?

Scope

The thesis scope is to determine the impact of different cloud service layers IaaS, PaaS and SaaS to the performance and security of cache intensive algorithms and web services. This study concentrates on selecting the best resource allocation and the most appropriate platform to achieve the best performance for the same cost.

This thesis models the impact of the cloud infrastructure to the performance for different types, number and size of input messages that load the cloud infrastructure. It also defines quantitative performance indicators to determine the risk of migrating the services in the cloud for various message size and number of concurrent messages.

Methods

To address the research questions from the performance area, several different infrastructures are deployed with three different platforms: Traditional on-premise (host), Virtual (guest) and Open source cloud platforms. Also, different platforms and infrastructures are used in Windows Azure Cloud.

As test data, different web services are deployed in the cloud and on-premise, as well as several different benchmark application for sequential and parallel im-

plementations of compute, memory and cache intensive dense matrix multiplication algorithm.

Detailed theoretical analysis is performed in order to analyze the on-premise performance. Next, huge number of experiments are realized for different problem size and different load to measure the algorithm performance. Detailed analysis and classification of the theory and the results of the experiments are realized.

We overview main international and industry standards towards security and analyze their conformity to cloud computing. Several security evaluations are performed using ISO 27000 series of standards as a baseline.

Content of the Thesis

This thesis consists of six parts: 1) Basic concepts of performance, matrix multiplication and cloud computing, 2) Matrix multiplication algorithm improvements, 3) Theoretical and practical proof of superlinear speedup in CPUs and GPUs, 4) Performance analysis of cache intensive algorithm in cloud computing, 5) Web service performance analysis in cloud computing, and 6) Cloud computing security challenges and evaluation.

Part I presents the basic concepts and definitions for the purpose of this thesis. It consists of five chapters. Chapter 1 describes the basic concepts and architecture of cloud computing, its deployment models and service layers. Basic performance fundamentals and limits are defined and elaborated in Chapter 2. The memory hierarchy and introducing cache memory in modern multiprocessors is described in Chapter 3 since memory access is the bottleneck of all computations and impacts to the performance. Chapter 4 presents basic sequential and several parallel implementations of dense matrix multiplication algorithm on CPU and GPU as compute, data and cache intensive algorithm. Finally, Chapter 5 briefly overviews the web services, their performance factors and several challenges.

Part II in four chapters analyzes and proposes matrix multiplication algorithm improvements and presents the theoretical analysis and experimental proof for performance drawbacks that appear when using set associative cache for cache intensive algorithm. Chapter 6 defines the Average Total Cycles Per Instruction and analyze which replacement policy is most appropriate for matrix multiplication algorithm. In Chapter 7 we present the MMCacheSim simulator which predicts matrix multiplication performance on particular existing or non-existing multiprocessor. Chapter 8 presents two versions of new hybrid 2D/1D partitioning that improves 2D blocking matrix multiplication algorithm. Chapter 9 analyzes and models the performance drawbacks for particular problem size due to set associative cache. It also presents huge performance drawbacks that are observed experimentally for speedup in parallel execution and speed in sequential and parallel execution.

Part III consists of two chapters and deals with superlinear speedup in matrix multiplication. Chapter 10 presents the theoretical analysis with experimental proof about why, how and when superlinear speedup can be achieved in multiprocessor. In

Chapter 11 we analyze how to achieve superlinear speedup for matrix multiplication on NVIDIA GPU and show the results of the experiment for speed and speedup vs cache requirements and number of SMs.

The next Part IV is devoted for the performance of dense matrix multiplication algorithm in the different cloud environments. It consists of five chapters. Chapter 12 denies the hypothesis that additional virtualization layer provides worse performance, i.e. it presents that there is a region with particular problem size for matrix multiplication where the program runs faster in virtual environment, both for sequential and parallel implementation. Chapters 13 and 14 present how different PaaS and SaaS cloud service layers impact to the matrix multiplication algorithm performance correspondingly. Cloud multitenant environment impact to the matrix multiplication algorithm performance is presented in Chapter 15. Finally the existence of superlinear speedup for parallel execution of matrix multiplication algorithm is given in Chapter 16.

Part V covers the performance discrepancy of different web services when hosted on-premise and in the cloud for different input parameters server load. It also presents the performance drawbacks due to security implementation to web services. It consists of three chapters. Chapter 17 presents the results of the experiments for performance drawbacks when web services are hosted in the cloud for varying the server load. The next two chapters 18 and 19 propose two strategies for better web service performance in the cloud.

Finally, Part VI covers the cloud computing security challenges and performs several security evaluations in four chapters. In Chapter 20 we present the performance drawbacks due to security standards implementation to web services in order to improve their security in the new cloud environment. Chapter 21 overviews the main international and industrial standards targeting information security and analyzes their conformity with cloud computing security challenges. Chapter 22 addresses the risks of the security challenges in the cloud with high-level risk-based approach this chapter in order to improve the client company business continuity if migrates its services into cloud. Chapter 23 proposes new methodologies for security evaluation of the security on-premise or in the cloud and cloud service layers with ISO 27001:2005 control objectives as a baseline for the evaluation. Chapter 24 presents the cloud security challenges evaluation of OpenStack open source cloud solution and three other open source clouds.

Main Results

A lot of results were published within this PhD thesis research on reviewed conferences and journals. The most important results are in the areas of performance and security. There are also plenty of state-of-the-art overviews, literature reviews, theoretical analyses of different algorithms towards hardware infrastructure and platform, practical experiments and evaluations.

All given hypotheses are proved or denied both theoretically and experimentally. Many unexpected results were observed and analyzed during the experiments and all of them are explained with comprehensive elaboration and in-depth analysis.

The main results are elaborated in the following sections.

Optimal Replacement Policy for Matrix Multiplication Algorithm

Since matrix multiplication algorithm is cache intensive, cache replacement policy is the next important parameter that impacts its performance after arising cache capacity problem. As the number of operations does not depend on cache replacement policy, we define and propose a methodology to determine the average memory cycles per instruction that the algorithm performs, since it mostly affects the performance. The results of the experiments published by the authors in [9] for the purpose of this thesis research show the optimal cache replacement policy for sequential and parallel dense matrix multiplication algorithm implementations.

We determined that both cache replacement policies, LRU and FIFO, provide similar speed and average cycles per instruction $CPI_T(N)$ for sequential and parallel execution. However, the results show that LRU replacement policy provides better $CPI_T(N)$ for sequential execution in dedicated per core cache memory. Parallel execution provides the best $CPI_T(N)$ in shared memory LRU CPU, i.e. LRU produces greater speedup than FIFO and is more appropriate rather than FIFO cache replacement policy for dense matrix multiplication algorithm.

Highly Configurable MMCacheSim Simulator

For the purpose of this thesis research the autors published in [145, 134] a new teaching methodologies and hardware courses improvements using visual simulators, incrementally weighted exercises, and finally working on real hardware controllers achieved significant improvements in grade distribution and computer science student interest in hardware.

As a further work, the authors develop and published in [11] Highly Configurable MMCacheSim Simulator. The main contribution of MMCacheSim is to allow simulation of various CPU architectures and to find out which possible internal CPU cache architecture is the most appropriate for particular sequential or parallel algorithm execution. The simulator makes the computer architecture and HPC teaching and learning process most appropriate. MMCacheSim allows to be configured:

- The hierarchy among cache levels, to be shared between cores or dedicated;
- The inclusivity between different cache levels;
- The size of the cache memory, the associativity, cache line sizes; and
- Replacement policy, with ability to have different cache replacement policies per different cache levels.

Matrix Multiplication Algorithm Improvements

2D block decomposition of matrices that can be placed in L1 CPU cache decreases the cache misses since the operations will access data only stored in L1 cache. However, as we elaborate in this thesis, cache size is not the only one important cache parameter that impacts to the performance.

As part of this thesis research we propose in [53] a new hybrid 2D/1D partitioning that reduces the number of operations and memory accesses than the original 2D blocking algorithm reported recently in the literature as cache optimization. The idea is to use rectangles instead of squared blocks in order to minimize the operations.

Using theoretical analysis to exploit the advantages of other cache parameters that can impact the algorithm performance we propose modified hybrid 2D/1D partitioning algorithm that even improved the basic 2D / 1D hybrid algorithm. Even more, the modified algorithm is prone to small cache set associativity on AMD CPU caches rather than the original 2D blocking algorithm. The experiments prove also the theoretical analysis, i.e. both our proposed algorithms outperform the 2D blocking algorithm for huge matrices on AMD Phenom CPU.

Performance Analysis of Multiprocessor with Set Associative Cache

Performance drawbacks in particular problem sizes are commented in literature without detailed explanation. We analyzed and modeled the performance drawbacks for particular problem sizes for sequential and parallel execution of cache intensive algorithm and within this research we have published the results in [120, 50, 123].

Based on theoretical analysis and experimental research we have concluded how n-way associative cache can seriously affect performance. We have analyzed and found theoretically the points where the associativity causes performance drawbacks and suggest organization of the matrix multiplication algorithm avoiding situations where mapping onto n-way set associative cache will use only a small part of the cache instead of whole cache capacity. The performed experimental research approved the results showing real cases of performance drawbacks in both sequential and parallel executions.

A theoretical proof of execution time peaks is presented for a case when set associative cache for cache intensive algorithms is used for execution of the matrix multiplication algorithm. Huge performance drawbacks are observed and analyzed for speed in sequential execution. The speed drawback is more expressive for greater matrix size than smaller in the critical points.

Huge performance drawbacks are also observed and analyzed both for speed and speedup in parallel execution. Parallel algorithm executions result with speed drawbacks more than sequential. Parallel speed and speedup drawbacks are inconsiderable for critical matrix sizes in L2 region dedicated per core. Significant speed and speedup drawbacks are found in L3 and L4 regions, especially for parallel execution on more cores. They are expressive for the experiments executing on $P = 4$ processors where the speed $V(4)$ is smaller than the speed $V(2)$. For $N = 512$ speed $V(4)$ is

similar as speed $V(1)$ in the observed points of the particular area. The speed drawbacks for the experiments executing on $P = 2$ processors are also expressive. Speed $V(2)$ is similar or even smaller than speed $V(1)$ in other points of the particular area.

We conclude that performance drawbacks appear due to increased number of generated cache misses in last level cache.

Superlinear Speedup in Matrix Multiplication

Despite the Gustafson's Law that maximum scaled speedup is p and Shi's limitation that superlinear speedup is only possible for *Non Structure Persistent* algorithms, we found superlinear region for parallel execution and realized deep and comprehensive analysis. For the purpose of this thesis research we published the theoretical analysis and experimental results in [124, 52, 33].

Superlinear Speedup on Multiprocessor

We tried to go beyond the limits specified in Gustafson's law not just finding examples of **superlinear speedup** for matrix multiplication but also to provide theoretical analysis how to achieve it in a real shared memory system. We show and provide a proof of its existence in a real multiprocessor system that uses caches. The superlinear speedup is possible in cases where sequential execution initiates more cache misses than for parallel execution. This happens for example, in a shared memory multiprocessor with dedicated caches.

We have presented the theoretical background of superlinear speedup existence and also introduced a methodology how to achieve it, when and where it can be achieved. We have also defined a testing methodology and realized a series of experiments to provide evidence of superlinear speedup and unexpected high performance.

The experiments also confirmed our theoretical results in [52]. The achieved high performance results and superlinear speedup is demonstrated on the example of dense matrix multiplication algorithm. It is possible within a range of values of matrix sizes, even if the environment is virtualized.

Superlinear Speedup on GPU

In [33] we present the theoretical analysis and experimental results of achieved superlinear speedup for GPU devices, which are also categorized as SIMD. We implement a structure persistent algorithm which efficiently exploits the shared cache memory and avoids cache misses as much as possible.

The experiments have confirmed the theoretical analysis about existence of superlinear regions of the problem size for matrix multiplication using GPU devices,

where the normalized performance per processing element for parallel execution is better than in sequential execution.

Based on the experiments, we have presented further proof that there is existence of superlinear speedup for SIMD architecture processors with dedicated caches, more particular GPU devices. However, we have only obtained superlinear speedup in the superlinear region where the cache memory requirements of the problem fit in L2 for parallel execution and generate L2 cache misses for sequential execution.

Virtualization Impact on Cache Intensive Algorithm Performance

The program should be executed slower in cloud virtual environment compared to traditional server due to additional virtualization layer. Despite this hypothesis, the experimental results prove that virtualization performance is even better than traditional for distributed memory, rather than shared where there is a huge performance drawback.

We performed detailed experiments published in [48] for the purpose of this thesis research. We concluded a huge performance discrepancy for cache intensive algorithm in virtualized environment. There is a region with particular problem size for matrix multiplication where the program runs faster in virtual environment.

Virtualization provides almost equal performance of 98% as traditional for sequential execution on matrix multiplication algorithm. Using parallelization it provides even greater speed of almost 15% for particular number of processing elements in L1 and L2 regions (distributed per core).

However, the virtualization loses the battle in L4 region when a lot of costly cache misses appear. Its performance is 33.42% worse than traditional in this region.

PaaS Impact on Cache Intensive Algorithm Performance

Executing the same program in different runtime environments provides different performance. Opposite to the hypothesis that Linux based runtime environment provides better performance, the results of the experiments published in [51] show that Windows based runtime environment in Windows Azure Cloud runs better than Linux, especially for problem size that can be placed in cache and will not generate a lot of cache misses.

The measured speeds for the same algorithm on Windows are greater than on Linux achieving up to 2.5 times better performance especially in L1-L3 regions. The behavior in L4 region is comparable, but still Windows platform achieves better performance.

IaaS Impact on Cache Intensive Algorithm Performance

The price for renting the resources has linear dependency on consumed resources, but not always all offered resources of virtual machine instances are most suitable for the customers. The same amount of resources offered by the cloud can be rented and utilized differently to speedup the computation. One way is to use techniques for parallelization on instances with more resources. Other way is to spread the job among several instances of virtual machine with less resources. As part of this thesis research in [131] we analyze which is the best way to scale the resources to speedup the calculations and obtain best performance for the same amount of money needed to rent those resources in Windows Azure cloud.

The results of the experiments are as expected for sequential execution, i.e. Extra Large VM achieves maximum speed in front of Large, Medium and Small in L2 region. However, Small VM achieves similar speed as Extra Large VM and they lead in front of Large and Medium VMs in L4 region.

Parallel execution provides more strange results. Dense matrix multiplication algorithm achieves maximum speed when executed parallel on 8 x Small instances, in front of 4 x Medium, 2 x Large, and 1 x Extra Large in L2 and L3 regions, and almost all observed L4 region. This means that the best performance can be achieved if dense matrix multiplication algorithm is granulated on 8 chunks and each chunk to be executed on 8 concurrent processes with one thread in Small Windows Azure VM. The same environment achieves maximum speedup in L2 region. In L3 and L4 region maximum speedup is achieved if dense matrix multiplication algorithm is granulated on 4 chunks and each chunk to be executed on 4 concurrent processes with two threads in Medium Windows Azure VM.

Multitenancy Impact on Cache Intensive Algorithm Performance

Since modern multi-core multiprocessors also share the last level cache among all cores on one chip, the goal is to enable an optimal resource allocation by avoiding cache misses as much as possible, since this will lead to performance increase. The realized experiments published in [49] for the purpose of this thesis research show that both single-tenant and multi-tenant resource allocation in the cloud provide better performance than on-premise for certain workload.

The performed experiments address several virtual machine instances in a cloud system using different number of CPUs (assuming all cores are utilized). Each experiment uses the same resources but orchestrated differently. The results can be summarized as:

- The experiments prove that there is a region (L2 region) where cloud environment achieves better performance than traditional and virtual environment, both for parallel and sequential process execution, and
- The experiments prove that cloud computing provides better performance in a multi-VM environment, rather than allocating all the resources to only one VM.

The best resource allocation for traditional environment for cache intensive algorithms is the usage of multiple processes with single threads. Multiple VMs with single threads is the best resource allocation for cloud environment. Comparing the environments, cloud computing provides the best performance.

Superlinear Speedup in Cloud Virtual Environment

The results published by the authors in [130] as part of this thesis research show that cloud environment can also achieve superlinear speedup for execution of cache intensive algorithms when high performance computing is used in virtual machines allocated with more than one processor (core). The speedup begins to increase for those matrices A and B that do not fit in available last level L2 cache for sequential execution, i.e. half of L2 total cache, but in the same time fit in the whole L2 cache which is available for parallel execution with two or four threads on two or four cores, correspondingly. The speedup increases until $N = 628$ determined theoretically for traditional environment when L2 cache misses begin to appear. There are also a speedup turnover points for virtual and cloud environment greater than theoretical value since virtualization provides better performance for parallel execution rather than sequential in shared memory as we published in [48].

Virtual and cloud environments achieve better speedup for dedicated cache and the best performance is achieved by the cloud environment. After the L2 region, where the L2 cache misses are generated, the traditional environment performs better in comparison to the cloud environment. Virtual environment achieves the worst speedup when the algorithm requires a lot of accesses to shared main memory.

The experiments show different speedup range. The widest superlinear speedup range is present at traditional environment, while the thinest is found at the virtual one. Cloud and virtual environments have wider superlinear speedup range for parallel execution with two rather than four threads because the last level cache is dedicated per core which is the case where virtualization provides better performance than shared memory as we published in [48] for the purpose of this thesis research. The range is shortened up to 3 times from the right side (for great values of N) compared to the left region (smaller values of N). The range in traditional environment shortens 80 from the left side and 120 from the right side of the range. In virtual environment it shortens 56 and 184, and in cloud environment 88 and 144 for left and right side correspondingly. Virtual environment's superlinear range is the most shortened while the superlinear speedup region in traditional environment shortens the least.

Another important result was obtaining superlinear speedup in virtualized environment with three different hypervisors: Microsoft Hyper-V, KVM and VMware ESX.

Web Service Performance in the Cloud

Although we found regions where cloud is better environment for cache intensive algorithms, even with superlinear speedup for parallel execution, it is not the case for web services hosted in the cloud. Additional layer that virtualization adds in the cloud decreases the performance of the web services. Series of experiments are realized and published in [138] for the purpose of this thesis research to analyze the web services performance and compare what is the level of degradation if the web services are migrating from on-premises to cloud using the same hardware resources.

The results of the experiments show that the performance directly depends on input message size especially for both memory demand and compute intensive web service regardless of the platform. This is not emphasized for memory only demand web service.

We also defined quantitative performance indicators to determine the risk of migrating the services in the cloud for various message size and number of concurrent messages. The conclusion is that the performance is decreased to 71.10% of on-premise for memory demand and to 73.86% for both memory demand and compute intensive web service if it is migrated on the cloud. The cloud provides the smallest penalties for greater message sizes regardless of number of concurrent messages for memory demand web service. However, the smallest penalties for both memory demand and compute intensive web service migrated in the cloud are provided for smaller number of concurrent messages and for greater message sizes.

A Middleware Strategy to Improve Cloud WS Performance

This solution that handles the compute peak loads dynamically for web services hosted in cloud is proposed in [133] for the purpose of this thesis research. It introduces a middleware layer between clients and server which will instantiate additional VMs dynamically on demand as service load reaches defined minimum performance level and will forward the messages across VMs. The additional VMs will be shut down when service load returns to defined nominal value.

Despite the latency for simple web services, the experiments prove that the middleware improves the performance of compute intensive web services (where huge part of the response time is spent for service calculations).

Message Transformation for Better Cloud WS Performance

Although a middleware strategy handles peaks that appear due to increased number of concurrent requests, it does not handles well peaks with increased load due to huge message size.

We proposed another strategy called message transformation strategy in [132] for the purpose of this thesis research. We used the conclusion from [119] that Mi-

crosoft Windows OS provides better performance than Linux Ubuntu OS for huge messages, and Linux Ubuntu OS provides better performance than Microsoft Windows OS for huge number of concurrent messages and for small messages.

If the response time increases beyond the threshold, then the middleware strategy will split the input parameters into smaller chunks that Linux OS can process faster rather than the whole message. If the peak is even bigger, then the middleware will start additional instance installed with Windows Server based OS and forwards the client requests among two endpoint web services, the whole messages to Windows Server based OS and the messages divided into smaller chunks on Linux Server based OS.

Performance of Security Measures in the Cloud

Moving the data and applications outside the company security perimeter enforces implementing security standards to achieve proper end-to-end security level with on-premise information systems. Since web services are the most used technique, we analyzed the performance of introducing most common web service security standards XML Signature and XML Encryption and published the results in [135, 137, 136] as part of this thesis research.

The results show that increasing message size and number of concurrent messages degrade the web service performance. Platform environment is also analyzed. Linux OS handles better the number of concurrent messages and Windows the opposite, i.e. it handles better greater messages.

We analyzed the maximum throughput via web services implementing XML Signature and both XML Signature and XML Encryption. We compared both platform and determine that Linux OS provides better throughput than Windows OS for small number of messages with and without security implementation. When implementing security Linux also provides better performance for huge messages. Windows provide better performance for middle sized messages.

Cloud Security Standardization Improvements

We overview the main international and industrial standards targeting information security and analyzes their conformity with cloud computing security challenges. We also overview the efforts done towards cloud security standardization. The results are published in [126, 125, 127, 129] as part of this thesis research.

ISO 27000 series (27001:2005, 27002:2005, and 27005:2011) of standards are defined as generic and they cover not only the technical solutions to technically identified threats and vulnerabilities, but take into account the operational, organizational and management vulnerability, as well. Due to its generality, as well as many open cloud security challenges, ISO 27001:2005 is not fully conformal with cloud information security system. Therefore, we propose a new control objective in ISO

27001:2005 requirements, *virtualization management*, with two controls covering *virtualization* and *virtual machines control*.

With high-level risk-based approach we addressed the risks of the security challenges in the cloud in order to improve the client company business continuity if migrates its services into cloud.

No paper so far has presented business continuity aspects in detail of cloud computing and it challenged us to address the cloud computing model security detriments that depreciate the cloud customer business continuity: data privacy and protection, regulatory and standards compliance, loss of control, data location, heterogeneity, complexity, and interoperability, multi-tenant environment, and disaster recovery - RPO and RTO compliance and effectiveness.

We introduced proposals which minimize the impact to business continuity and the probability of incident scenario for each detriment. These main risks can be assessed appropriately and mitigated to the acceptable level by applying recommendations in these proposals according to matrix for risk level as a function of the business impact and probability of incident scenario.

We address cloud computing model security beneficial that improves the business continuity: eliminating downtime, better network and information security management, disaster recovery with both backup management and geographic redundancy. It also avoids or eliminates disruption of operations, increases service availability and decreases DoS attacks.

New Methodologies for On-premise vs Cloud Security Evaluation

As a part of this thesis we defined and published in [127, 126] two methodologies for security evaluation of the security on-premise or in the cloud and cloud service layers. ISO 27001:2005 control objectives are taken as a baseline for the evaluation.

The first methodology quantifies the ISO 27001:2005 requirements grouped in control objectives, comparing on-premise and cloud environments. The evaluation and analysis of ISO 27001:2005 standard result in the importance transfer from cloud customer to CSP. Simultaneously cloud customer must provide a huge effort to implement all control objectives with decreased importance in SLA with its CSP.

The second methodology quantifies the ISO 27001:2005 requirements that are grouped in control objectives, for on-premise and different cloud service layers. We evaluate that moving into cloud, 12 of 39 control objectives are for management, and are not affected if the services are on-premise or in cloud. Importance factor doesn't change on average seven Control Objectives, depreciates on 18, and increases on only two of them. Thus, moving into cloud, cloud customers (SMEs) transfer the importance of the security to its CSP, and expect that their data and applications are to be secured. Therefore, due to emergent security challenges that cloud computing produces, cloud customers must re-evaluate their BCPs.

New Methodologies for Open Source Cloud Security Evaluation

All key commercial cloud providers possess some security certificate as a company. Additionally all of them offer some security services to their customers [126]. Open source solutions provide a small number of security services to the clients or generally do not provide any.

Neither security assessment nor comparative security analysis of the cloud were not performed in the literature so far. We proposed two new methodologies for Open Source Cloud Security Evaluation and published in [128, 122] as a part of this thesis research.

We have analyzed the security issues that OpenStack cloud software possess and what other security tools can be integrated to improve its overall security. The evaluation of the security was realized by assessing the relevant control objectives defined by ISO 27001:2005 and comparing it to the other open source cloud computing solutions.

The results of our assessment show that Eucalyptus and CloudStack have integrated the maximum security level in front of OpenNebula. OpenStack has integrated the least security compared to others solutions.

General conclusion of the evaluation is that all open source clouds take care about some level of security. The results of the evaluations show that CloudStack is the best choice of all open source clouds to migrate the services and integrated the maximum security level in its architecture. It conforms with all ISO 27001:2005 11 control objectives that depends of the cloud solution. Eucalyptus and OpenNebula has also reached far in security. OpenStack is worst solution to migrate the services in the manner of security.

Although open source clouds heed the security, the company still have other 28 technical requirements, organizational and management requirements that should be conformed. Also, ISO 27001:2005 defines general requirements, i.e.management responsibility and establishing, managing, reviewing and improving the information security management system.

Applicability of the Results

Besides the huge number of important results that we mentioned previously, it is worth to mention their applicability.

HPC Improvements

The realization of modern processors is based on a multicore architecture with increasing number of cores per processor which is actually organized as a shared memory multiprocessor with shared L2 cache and distributed L1 cache. Therefore

these results will have impact on future software development and exploitation of parallel hardware.

The results and methodology can be used in the massive data computations with high cache miss ratio. Dividing data into smaller chunks with optimal size calculated with our methodology, reduces cache misses in parallel execution in the dedicated cache per core system. Although we simplified our calculations, our methodology can be used in other similar high performance numeric computations.

As a conclusion mathematical relations showed a possibility of superlinear speedup and extensive experimental research approved the results showing real cases of increased unexpected performance in a multiprocessor or multicore system (or both), with dedicated cache per core and thus propose using such systems in parallel computing.

Cloud Platform Selection

The main contribution of this thesis is based on experimental proof and recommendation to use the Windows platform while using Azure cloud for cache intensive problems, despite the hypothesis that Linux Operating System has better performance than Windows. That is, hypervisors have a huge impact to the overall performance, sometimes they even improve the performance.

Cloud Virtualization Can Even Improves Performance

We found that virtualization performance is better than traditional for distributed memory (dedicated caches per core). For shared memory there is a huge performance drawback in virtual environment (L4 region). Therefore, we propose CPUs with dedicated cache per core when building the clouds.

HPC Application Execution in Cloud

We defined the best resource allocation among virtual machines in the cloud to achieve maximum performance. The best resource allocation for traditional environment for cache intensive algorithms is the usage of multiple processes with single threads. Multiple VMs with single threads is the best resource allocation for cloud environment.

The best performance can be achieved if matrix multiplication algorithm is granulated on 8 chunks and each chunk to be executed on 8 concurrent processes with one thread in Small Windows Azure VMs. The same environment achieves maximum speedup in L2 region. In L3 and L4 region maximum speedup is achieved if multiplication algorithm is granulated on 4 chunks and each chunk to be executed on 4 concurrent processes with two threads in Medium Windows Azure VMs.

Cloud Security Evaluation

We defined 4 methodologies for security evaluation if the company migrates from on-premise in the cloud, then what cloud service layer, which open source cloud solution and migrating from one to another open source cloud solution.

Skopje, *Sasko Ristov*
August 2012

Acknowledgements

First and foremost, I wish to thank my supervisor, professor Marjan Gusev for not just given greatest support, but active involvement in research and writing our published papers for this thesis research without saving the efforts even during weekends and holidays. He gave unnecessary contribution in improving and publishing papers. I am very happy for choosing him as a supervisor. I hope and I'll be very glad to continue and amplify our common research in the future.

I would also like to thank my wife Monika and my children Kirjana and Teo for their given support and understanding. Many thanks also to my mother Slavica and my brother Blaze for their encouragement.

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
B2B	Business to Business
BCP	Business Continuity Plan
CaaS	Communication-as-a-Service
CAPEX	Capital expenditure
cbs	Cache block (line) Size
CC	Cluster Controller (in Eucalyptus Cloud)
CLC	Cloud Controller (in Eucalyptus Cloud)
CPU	Central Processing Unit
CSA	Cloud Security Alliance
CSP	Cloud Service Provider
DaaS	Data Storage-as-a-Service
EBS	Amazon's Elastic Block Store
EC2	Amazon's Amazon Elastic Compute Cloud
FIFO	First-In-First-Out Cache Replacement Policy
GPU	Graphics Processing Unit
HPC	High Performance Computing
IaaS	Infrastructure-as-a-Service
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IDSaaS	Intrusion Detection System-as-a-Service
IPS	Intrusion Prevention System
ISMS	Information Security Management System
ISRM	Information Security Risk Management
IT	Information Technology
Java EE	Java Enterprise Edition
KPI	Key Performance Indicators
KVM	Kernel-based Virtual Machine
l	Number of elements that fit in a cache block (line)
LAN	Local Area Network
LIP	LRU Insertion Policy

LM	Cache Level Misses
LRU	Least-Recently-Used Cache Replacement Policy
ME	Memory Element Size
MPI	Message Passing Interface
NC	Node Controller (in Eucalyptus Cloud)
OpenMP	Open Multiprocessing
OPEX	Operational Expenditure
OS	Operating System
PaaS	Platform-as-a-Service
RBAC	Role Based Access Control
RPC	Remote Procedure Call
RPO	Recovery Point Objective
RTO	Recovery Time Objective
SaaS	Software-as-a-Service
SC	Storage Controller (in Eucalyptus Cloud)
SIMD	Single Instruction Multiple Data
SIMT	Single Instruction Multiple Thread
SLA	Service Level Agreement
SM	Streaming Multiprocessor
SME	Small and Medium Enterprise
SP	Scalar Processor
SECaaS	SECurity-as-a-Service
VLAN	Virtual Local Area Network
VM	Virtual Machine
VPN	Virtual Private Network
W3C	World Wide Web Community
WSDL	Web Service Definition Language
XaaS	Everything-as-a-Service
XML	Extensible Markup Language

Part I
Basic Concepts

Chapter 1
Cloud Computing

Abstract Cloud computing is a new concept of resource allocation and utilization. It offers scalable, flexible and on-demand resources to host the companies applications and data. The on-demand concepts rent whenever you need and pay when you rent offer the customers to invest the money into their business rather to invest in advance for underutilized ICT equipment. For service providers it reduces CAPEX, such as better resource utilization due to virtualization and multi-tenancy, hardware and licenses costs, and reduces OPEX, such as human resources and equipment maintenance. For customers it offers massive scalability, elasticity, and self provisioning of user resources.

1.1 Global Concepts

1.1.1 What is Cloud Computing

Cloud computing is the fifth generation of computing after Mainframe, Personal Computer, Client-Server Computing, and Web [116]. It is a model that enables convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or cloud provider interaction [98]. It enhances collaboration, agility, scaling, and availability, and provides the potential for cost reduction through optimized and efficient computing [29].

From customer perspective, cloud computing means unlimited on-demand, flexible and scaled computing and storage resources. The customers do not know where the data and application are hosted and if they are distributed or shared. From CSP perspective, it is a multi-tenant shared environment with a usage-based billing model.

1.1.2 Virtualization

Virtualization is a technique that offers creation of more dynamic, flexible and scalable datacenters. Figure 1.1 presents the architecture differences of traditional and virtual environment.

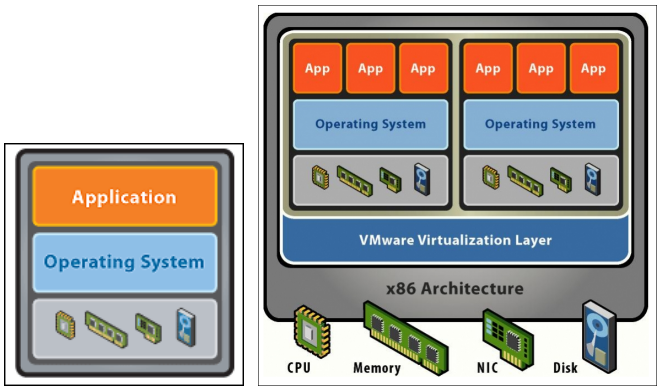

Fig. 1.1 Traditional (left) [156] and Virtual (right) environment comparison [157]

It is impossible to build a modern enterprise without virtualization. Cost reduction for power consumption and cooling, easy administration, licenses regulation, backup, migration, and security isolation are benefits and reasons why virtualization techniques usage rapidly grows.

Virtualization is the baseline for cloud computing IaaS and PaaS service layers [48]. Using virtualization the cloud resources are shared among multiple tenants in one or more instances of virtual machines according to their needs. Therefore isolating tenants in separated and secured platform environment is vital. However, the tenants still share the hardware resources and this makes an impact to the overall performances.

Despite all the advantages, current virtualization solutions do not produce performance isolation among virtual machines (VMs). The introduction of a new layer produces overall performance discrepancy of the application in a guest operating systems. Running the same virtual machine on the same hardware at different times among the other active VMs will not achieve the same performance [85].

Many different virtualization techniques exist, such as operating system virtualization, platform virtualization, storage virtualization, network virtualization, and application virtualization explained in details in [13].

1.2 Cloud Deployment Models and Service Layers

Different companies posses different data types and application and they want from the service providers to offer the services appropriate as their need. It is cheaper to migrate the data and application in the cloud reducing IT administration and costs for IT equipment and service management. However, not all the data and application can be outsourced to the CSPs.

Three main cloud deployments exist:

- *Public Cloud* - The resources are dynamically shared to the customers by third party CSP. The same services are offered to the customers (tenants). The resources are hosted in one or many data centers and the customers do not know where their data and application are hosted;
- *Private Cloud* - The resources are used only by the company that builds the cloud. All the costs, management and administration remain within the company. However, the private cloud offer increased security level than the public cloud since the applications and data are not moving outside of the customer security perimeter; and
- *Hybrid cloud* - Uses the benefits of the other two deployments. The company can outsource its public data and non-core applications to the public cloud and build its own private cloud to host its confidential data and core applications according to its information security policy. The companies can use multiple public clouds or hybrid clouds to increase the service availability. There are several multiple cloud management strategies described in [64].

The main goal of the cloud is to offer Everything-as-a-Service (XaaS). This section shows the three main cloud service layers IaaS, PaaS and SaaS, as well as others XaaS that can be found in the literature.

1.2.1 Main Cloud Service Layers

Cloud services are delivered in three main cloud service layers:

- *Infrastructure-as-a-Service* (IaaS) - CSPs provide the entire hardware infrastructure for the customers to host their data and run the applications. The customer is provided with processing, storage, networks and other computing resources, and he is able to deploy and run arbitrary software [84]. IaaS offers scalable resources in near real-time if the customer applications performance decrease or the storage place becomes insufficient. The resources are charged in pay-as-you-go model;
- *Platform-as-a-Service* (PaaS) - The customer is provided with programming languages and tools, so as to be able to create and use various applications [84]. PaaS gives the illusion of infinite resources to the customers since they can scale the resources without requiring additional actions on its site; and

- *Software-as-a-Service* (SaaS) - the customer uses CSPs' applications running on a cloud infrastructure. It represents the cloud computing from the end-users point of view, used in everyday work [84].

Figure 1.2 compares the differences among three service layers of cloud computing versus traditional on-premises computing through deducing which resources or services are executed by the customer and CSP. The resources and services in responsibility of the CSP are shown in green boxes, while those in responsibility of the cloud customer are shown in red.

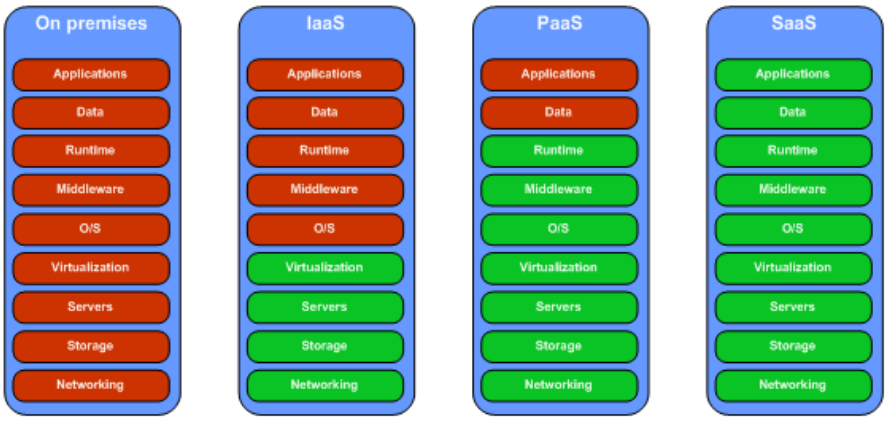

Fig. 1.2 Comparison between On-premises computing, IaaS, PaaS and SaaS [84]

Figure 1.2 shows that SaaS solution may be used from anywhere and at any time, provided a client (web browser) and Internet connection. These features make SaaS software most attractive for Small and Medium Enterprise (SMEs), and require no additional expensive and complex resources and hardware on customer's part. The responsibilities for all parts of the IT services hosted on-premises are on the resource owner, the customer in our case. Going from IaaS, through Paas to SaaS service layers in cloud computing, more and more responsibilities are transferred from the cloud customer to CSP [126].

1.2.2 Other Cloud Service Layers

Many other *Something-as-a-Service* items are proposed for different purposes. *Security-as-a-Service* (SECaaS) and *Data protection and privacy-as-a-Service* will speed up cloud market growth, both for the providers offers and clients, as well as cloud trustworthiness [126]. Cloud Security Alliance (CSA) offers 10 candidate domains for SECaaS [28].

Data Storage-as-a-Service (DaaS) allows cloud customers to store their data on CSPs' servers located in remote locations [64]. *Communication as a Service* (CaaS) provides reliable, schedulable, configurable, and (if necessary) encrypted communication [64]. *Intrusion Detection System-as-a-Service* (IDSaaS) [4] allows cloud customers to define a virtual private area within the public cloud space for their applications that can be secured with application-specific policies.

1.3 Which Cloud to Migrate the Services on?

In this section we present the analysis for current most common cloud computing offers on the market performed for this thesis by the authors in [122]. The main question at the beginning of cloud computing was if the cloud concept can become basic ICT choice for the companies. Nowadays the main question is when the cloud will become basic ICT choice for the companies. A list of 10 critical obstacles and opportunities for growth of cloud computing is given in [10]. However, not all companies will migrate their services in the public cloud. Some will keep the services in their own IT resources, such as business critical software, and others will change the use of IT resources into private clouds [114].

Many CSPs offer various public cloud solutions on the market, such as Amazon's AWS [6], Salesforce's Sales Cloud [139], Google's App Engine [45] and Cloud Storage [46], Microsoft Azure [92] and Live [91], VMware's vCloud [160] etc. Table 1.1 presents the main cloud platform offers for different cloud service layers.

CSP	IaaS	PaaS	SaaS
Amazon	EC2, S3, Simple Queue Service, SimpleDB		
Salesforce		Force.com, Heroku, Database.com	Sales cloud, Service Cloud
Microsoft		Azure (Windows, SQL, .NET)	Live, Hotmail, Office Web App
Google		Google App Engine	Gmail, Google Docs
IBM	SmartCloud	CloudBurst Appliance	Lotus Live, Blueworks Live

Table 1.1 Cloud platform offers for different cloud service layers [126]

Despite the commercial clouds and their services, there are many open source cloud solutions that allow the customers to develop their own private cloud, especially IaaS cloud service layer, such as well known OpenStack [106], Eucalyptus [38], OpenNebula [104], and CloudStack [23]. The authors in [162] devise a set of criteria to evaluate and compare most common open source IaaS cloud solutions. Almost all open source cloud solutions provide interfaces to commercial cloud services Amazon's EC2 and S3, and Google's App Engine.

1.4 Open Source Cloud Architectures

This section gives an overview and comparison of four most common Open Source cloud components, architecture and features in more details that was performed by the authors in [122] for the purpose of this thesis research.

1.4.1 The OpenStack Cloud Architecture

Openstack cloud consists of three main components: Compute, Object Storage, and Image Service. *Compute Infrastructure (Nova)* is the core part of the cloud that manages instances of virtual machines (VMs) and networking. *Object Storage* is the subsystem that stores the objects in a massively scalable large capacity system. It back ups and archives data, stores secondary or tertiary static data, stores data when predicting storage capacity is difficult, and creates the elasticity and flexibility of cloud-based storage for customer web applications. *Image Service* is lookup and retrieval subsystem for VM images.

OpenStack can be deployed and runs on Linux Ubuntu, CentOS and RedHat. It supports Kernel-based Virtual Machine (KVM), Xen, UML, Microsoft Hyper-V and QEMU hypervisors. Nova services can be deployed either on the same physical server or they can be installed on separate servers.

OpenStack network consists of two networks, public and private as depicted in Figure 1.3. The IP addresses from the public network are associated with instances of VMs to be accessed from the public internet. The private network is used for internal web service communication.

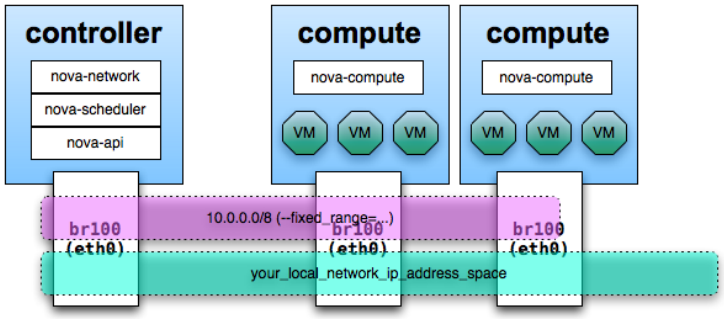

Fig. 1.3 OpenStack networking example [107]

1.4.2 OpenNebula Cloud Architecture

OpenNebula is an open source cloud software that builds both public and private clouds. It supports all most common hypervisors and operating systems. The OpenNebula architecture is depicted in Fig 1.4. Front-end executes the OpenNebula services. Hypervisor-enabled hosts provide the resources that instances of VMs need. Datastores hold the base images of the VMs. Service Network is used to support interconnection of the storage servers and OpenNebula control operations. VM Networks are physical networks that support VLANs for the VMs.

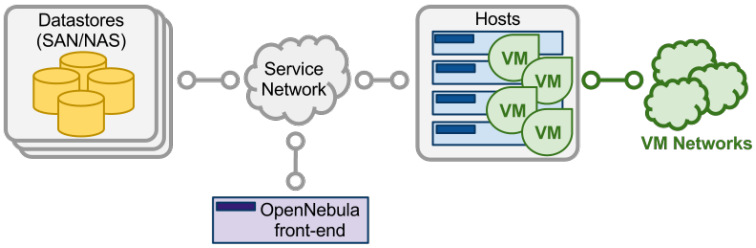

Fig. 1.4 The OpenNebula Cloud Software Architecture [104]

1.4.3 CloudStack Architecture

CloudStack is an open source cloud software that builds both public and private clouds. It supports all most common hypervisors and operating systems. The CloudStack architecture is depicted in Fig 1.5. Management Server is a single point of configuration that provides web user interface and APIs, manages the assignment of VM instances to particular hosts, public and private IP addresses to particular accounts, and the allocation of storage. Cloud infrastructure is organized as Zone (equivalent to a single datacenter) with one or more Pods (typically one rack) such that each Pod with one or more clusters.

1.4.4 Eucalyptus Architecture

Eucalyptus is an open source cloud software that builds both public and private clouds. It supports all most common hypervisors and Linux operating systems. The Eucalyptus architecture is depicted in Fig 1.6. It consists of six components: Cloud Controller (CLC), Walrus, Cluster Controller (CC), Storage Controller (SC), NodeController (NC). Each component is a stand-alone web service. CLC makes

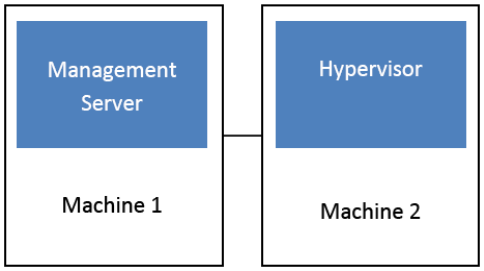

Fig. 1.5 The CloudStack Cloud Software Architecture [23]

high-level resources scheduling decisions and makes requests to the CCs and is responsible for exposing and managing the underlying virtualized resources (servers, network, and storage). Walrus stores and accesses VM images and user data. CC schedules VM execution on specific nodes and manages the VM instances networks. SC has a function as Amazon's EBS [5]. NC controls activities of VM instances and manages the virtual network endpoint.

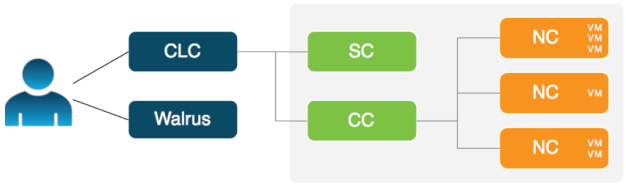

Fig. 1.6 The Eucalyptus Cloud Software Architecture [38]

1.5 Summary

This chapter presents the basic concepts of cloud computing. It relies on virtualization technique. The cloud services are grouped into three main service layers: IaaS, PaaS and SaaS and can be deployed in private, public or hybrid cloud. We also present the main CSPs and open source cloud solutions for private and hybrid clouds.

Chapter 2
Performance

Abstract Todays huge computing requirements make a desire for faster program execution. Many mechanisms exist to speedup the execution. Examples start from improving algorithm and executing less operations and program steps on the same computer; and move towards usage of faster computers executing the same number of operations for less time. Implementation of parallelism and high performance computing is final modern mechanism for faster program execution. It is necessary to define a way to compare two or more different algorithms, computers or approaches and determine which is faster.

2.1 Performance Fundamentals

The performance can be defined differently depending on the problem and the environment. A faster software program is the one that finishes in less time. Faster bank cashier is the one that serves more customers in the bank in one working day. In both cases the time spent is very important.

2.1.1 Basic Definition of Performance

Since smaller execution time is needed for better performance, the performance of a system A is defined in (2.1) [63].

$$Performance_A = 1/ExecutionTime_A \tag{2.1}$$

Therefore, system A has better performance than system B, i.e. $Performance_A > Performance_B$ if $ExecutionTime_A < ExecutionTime_A$

2.1.2 Speed

Todays vendors race which computer will execute more operations in second. This is the most important performance factor, i.e. computer system speed V which is defined in (2.2) as a ratio between number of operations o and time required t.

$$V = o/t \tag{2.2}$$

The *Speed* is equal parameter to performance and reciprocal value of execution time.

In addition, two important parameters determine the CPU performance for a given software program. The first one is *MIPS* (Millions of Instructions Per Second) defined in (2.3).

$$MIPS = \frac{InstructionCount}{CPUExecutionTime \cdot 10^6} \tag{2.3}$$

More important parameter is GFLOPS (Giga FLOating point instructions Per Second) defined in (2.4).

$$GFLOPS = \frac{FPInstructionCount}{CPUExecutionTime \cdot 10^9} \tag{2.4}$$

Todays modern supercomputers produce speed of more than 20.000 TFlops [141].

2.1.3 Speedup Factor

If some problem can be divided on several smaller chunks then it can be executed concurrent on several processors. *Speedup* $S(p)$ that can be achieved is defined in (2.9) as a ratio of execution time using one processor t_s and execution time using p processors t_p.

$$Speedup = \frac{t_s}{t_p} \tag{2.5}$$

The best sequential algorithm should be executed and underlying algorithm for parallel implementation might be different for maximum speedup.

2.1.4 Efficiency

Most of modern clusters, grids or clouds are heterogeneous environments. Therefore, neither all processors work with the same speed nor all processing elements finish the same job in total execution time. $Efficiency\ E$ defines the fraction of time

that the processors are being used. It is formally defined as a ratio of execution time using one processor t_s and the product of execution time using p processors t_p with the number of processors p (2.6).

$$E = \frac{t_s}{t_p \cdot p} = \frac{S(p)}{p} \tag{2.6}$$

2.1.5 Cost

The *Cost* is another parameter for parallel execution. It defines the total time that p processors are used or utilized for program execution. Relation (2.7) gives a formal definition of cost as a product of execution time and the number of used processors p. It also show several derived variants with other parameters.

$$C(p) = t_p \cdot p = \frac{t_s \cdot p}{S(p)} = \frac{t_s}{E} \tag{2.7}$$

2.2 Performance Limits

This section presents the performance limits analysis published by the authors in [124] for the purpose of this thesis research.

Amdahl has shown that multiprocessor execution performance is not proportional to the number of processors [7]. Gustafson has found a way to show that there are algorithms which can have almost linear speedup [54].

2.2.1 Speedup Analysis

Both Amdahl's law [7] and Gustafson's scaled speedup [54] use a single equation (2.8) and bring conclusions according to the value of a single parameter s, i.e. the sequential portion of a parallel algorithm which characterizes the algorithm.

$$Speedup = \frac{1}{s + p/P} \tag{2.8}$$

The parallel portion of the program is p and can be executed by P processors in parallel. Normalized time where $s + p = 1$ is used in (2.8). Amdahl assumes that the number of processors $P \to \infty$ and in (2.9) concludes limited speedup independent of the number of processors.

$$Speedup \leq \frac{1}{s} \tag{2.9}$$

Gustafson walks on the surface of small sequential fractions assuming $s \to 0$ and in (2.10) concludes scaled speedup bounded by the number of processors P.

$$Scaled\ speedup = P + (1-P)s' \tag{2.10}$$

Figure 2.1 from [54] shows both fixed size and fixed time (scaled) speedups.

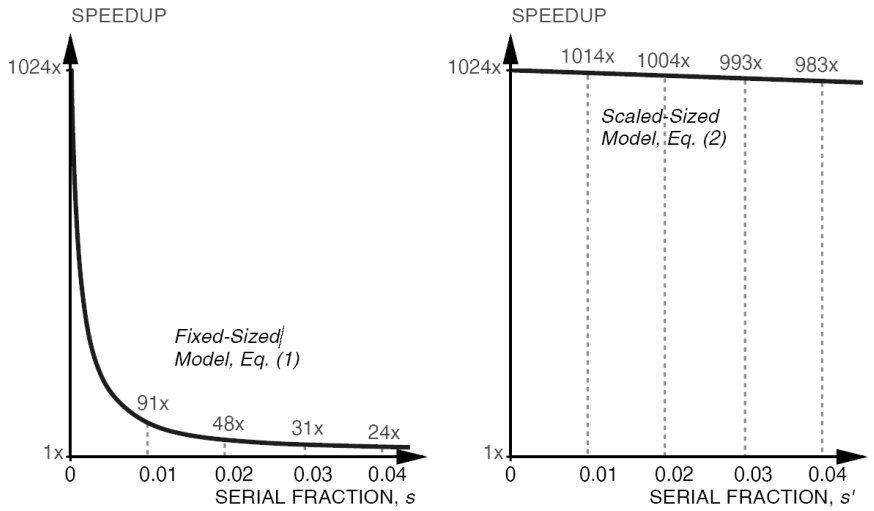

Fig. 2.1 Speedup given by Amdahl's law and by problem scaling [54]

A good understanding of these relations is given in [140] with equivalence of both approaches along with examples of possible misuse of given formulas.

Sun and Ni in [147] consider uneven workload allocation and communication overhead to evaluate the speedup. They use W to be the amount of work of an application, $T_i(W)$ the time required to complete W amount of work on i processors, W_i be the amount of work executed with degree of parallelism i and m the maximum degree of parallelism. Thus, $W = \sum W_i,\ i = 1,\ldots,m$. By $Q_P(W)$ they express the communication overhead when P processors are used and derive the speedup as

$$S_P = \frac{\sum_{i=1}^{m} W_i}{\sum_{i=1}^{m} \frac{W_i}{i} \lceil \frac{i}{P} \rceil}$$

The simplified version without communication overhead of the speedup is

$$S_P = \frac{W_1 + G(P)W_P}{W_1 + \frac{G(P)}{P} W_P + Q_P(W*)}$$

where $G(P) = W_P^*/W_P$ is defined to represent the ratio of work increment. It equals to Amdahl's Law for fixed-size speedup and to Gustafson's scaled speedup for fixed-time speedup, as special cases of memory bound speedup.

2.2.2 Speedup Limits

In [146] if both A and B are restricted to square matrices with dimension N, then the computation requirement of matrix multiplication is equal to $W_P = 2N^3$ and the memory requirement is $M = 3N^2$. Therefore the authors compute $W_P = (\frac{2M}{3})^{3/2}$ and according to [146] the function $G(P)$ will be $G(P) = \sqrt{(\frac{3P}{P+2})^3}$, which is less than P, for $P > 1$, and is bounded by $3^{\frac{3}{2}}$. Note that due to data replication, the memory capacity requirement increases faster than the computation requirement does.

The conclusion in [54] is that linear speedup is maximum speedup in a parallel system and presents the domain of computing performance pattern in the log scaled Figure 2.2. The conclusion is that fixed-size speedup (Amdahl's law) bounds speedup to the amount dependent of sequential fraction of the algorithm; the fixed-time speedup is less than scaled speedup bounded to the linear speedup and the author specifies the domain of insufficient memory.

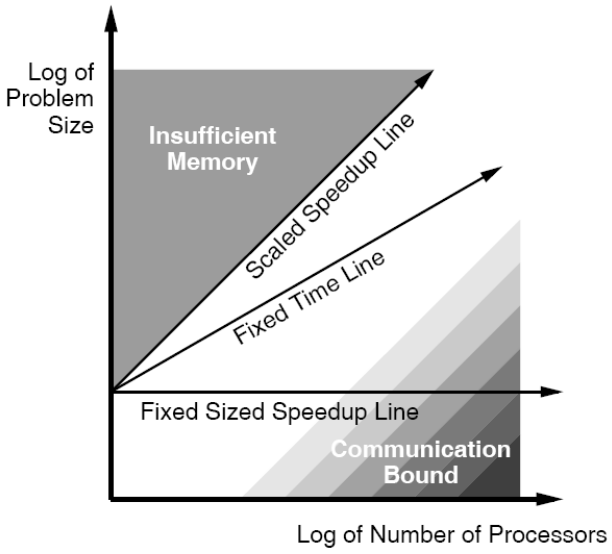

Fig. 2.2 Ensemble computing performance pattern [54]

Authors in [39] conclude the impossibility of superlinear speedup, based on implicitly on the assumption there is no savings from diminished loop overhead.

Shi in [140] refers to a prerequisite to apply Amdahl's or Gustafson's formulation clarifying that the sequential and parallel programs should take the same number of total calculation steps for the same input and avoid cheating to break the law. Shi points to $O(n^2)$ comparison-based sort algorithm as an example to "break" the law, since the $O(n^2)$ sort algorithm cannot retain its structure when crafting a parallel algorithm from it. In other words, partitioning such a sequential algorithm can improved its efficiency. Therefore the introduction of *structure persistent* algorithm is imperative.

Gustafson has referred to memory limitations in [56] and makes further analysis. Historical ensemble models hold uniprocessor performance flat as problem size varies, even beyond physical memory size. However, Gustafson defines tiered memory as a system which can make performance increase instead of decrease as problem size per processor shrinks, and workload can shift to routines with higher speed as the problem is scaled. Superlinear speedup results in such cases and is far from being an anomaly, it becomes a common place when the performance model makes realistic assumptions about memory speed and problem scaling.

Gustafson refers to flat memory approximation in [57] comparing it to assumption that the Earth is flat. For many everyday activities, like estimating short distances, a planar view of the world simplifies life at little cost of accuracy. If we scale up the problem to distances of thousands of miles or down to a few inches, the flat Earth assumption gets us into trouble.

A typical curve for fixed size speedup (Amdahl's Law) is presented in the log scale Figure 2.3 bounded by the superlinear speedup.

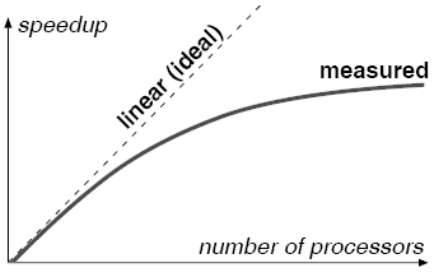

Fig. 2.3 Typical Speedup Curve [57]

As a final example of a surprising result of fixed time performance evaluation, the fixed time model creates a new source of superlinear speedup. In this context, "speed" means operations per second for some type of operation; whatever the measure, we assume speed varies on each processor as a function of time. You can think of the speed as a function of the subtask. If there are two subtasks, each growing with problem size N, and each possible to run in parallel, then Figure 2.4 shows how the changing speed "profile" can increase performance superlinearly [56].

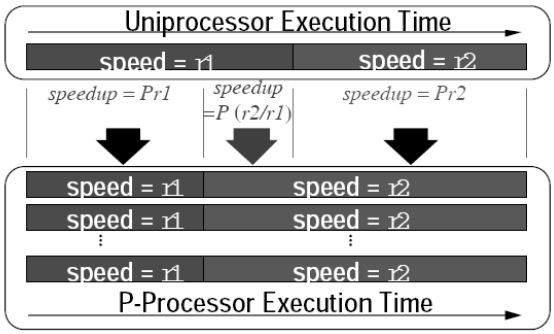

Fig. 2.4 Fixed Time Superlinear Speedup according to [57]

2.3 Intensive Algorithms

Each processor's operation consists of several parts: load some data from the memory, executes the operation upon the data and then store the result in the memory.

2.3.1 Compute and Data Intensive Algorithms

The total execution time (and the performance) mostly depends of what type of operations (computations) particular algorithm performs and the total ammount of data. The authors in [42] define two types of intensive algorithms presented as definitions 2.1 and def:DataIntensiveAlgorithm.

Definition 2.1 (Compute Intensive Algorithm).
The *Compute Intensive Algorithm* is the algorithm which spends the most of its time on computations. These algorithms usually perform the computations on small amount of data and I/O operations. However, increasing the problem size also increases the computation complexity. Parallelization of compute intensive algorithms achieves almost linear speedup because usually small amount of data is used without any data dependency.

Definition 2.2 (Data Intensive Algorithm). The *Data Intensive Algorithm* (or Memory Demanding) is the algorithm that requires the computations on huge amount of data and I/O operations. It spends the most of its time on loading and storing the data into memory. These algorithms usually achieve smaller speedup since usually there is data dependency and memory access performance bottleneck.

Both *Shared memory* and *Message passing* concurrent systems usually have a part of the program that need to be executed sequentially due to synchronization, data consistency and coherency.

2.3.2 Cache Intensive Algorithms

However, dividing parallel processing as either compute-intensive or data-intensive does not satisfies todays computing demands. Many algorithms exist that have another feature beside compute or data intensive. It is reuse of the same data. We introduce Definition 2.3 that is published in [130] for the purpose of this thesis research, which formally defines those algorithms as *cache intensive algorithm*.

Definition 2.3 (Cache intensive algorithm). *Cache intensive algorithm* with complexity k is the algorithm where the average number of accesses per element is $k > 1$.

The notation $O(k)$ is used for k-cache intensive algorithm. For example, the dense matrix multiplication algorithm is $O(N)$ cache intensive since each element is accessed N times for different computations. Nevertheless, it is compute intensive $O(N^3)$ and memory demanding $O(N^2)$.

2.4 Summary

To compare two algorithms or programs we need to define and measure their performances. This chapter defines the performance factors and its limits in parallel execution. Besides compute and data intensive, we introduce new additional *cache intensive algorithm* for better analysis and faster execution of algorithms.

Chapter 3
Memory Hierarchy

Abstract Memory access is the main bottleneck in all kinds of program execution, regardless it is compute, data or cache intensive algorithm. Memory access time is 100 to 1000 times slower than executing one basic processor operation [62], such as addition or multiplication. Introducing memory hierarchy, i.e. on-chip or off-chip cache memory, increases memory access performance, especially for the algorithms that repeatedly use the same data. Only intelligent program transformation approach based on cache size analysis and algorithm organization can efficiently exploit better performance [120]. Therefore it is important to understand the memory hierarchical architecture organization and its different parameters that impact to the overall algorithm performance.

3.1 Memory is the Bottleneck

Basic computer system is built on the Von Neumann concept (or Eckert-Mauchly as recently recognized) with central processing unit (CPU), main memory and bus. The problem of matching the speed of the instruction execution with the speed of fetching and storing the data / instruction degrades the overall performance of the computer system. Modern multiprocessors use multilayer cache memory system [63] to balance the gap between CPU and main memory and to speedup data access.

CPU runs a particular program by accessing data from the memory, executing basic operations addition or multiplication and storing the results in the memory. The main bottleneck in the process is the data access in memory which is approximately up to 1000 times slower than floating point operation execution [63]. Introducing memory hierarchy based on caches in CPU speeds up the execution of programs that reuse the same data, i.e. cache intensive algorithms.

3.1.1 Multilevel Cache to speedup the Memory

Nowadays, most of modern multiprocessors use three layer cache memory to speedup main memory access. The cache size grows but the access time and miss penalty rise going from the lowest L1 to L3 cache, main memory and disk. Figure 3.1 depicts the memory hierarchy.

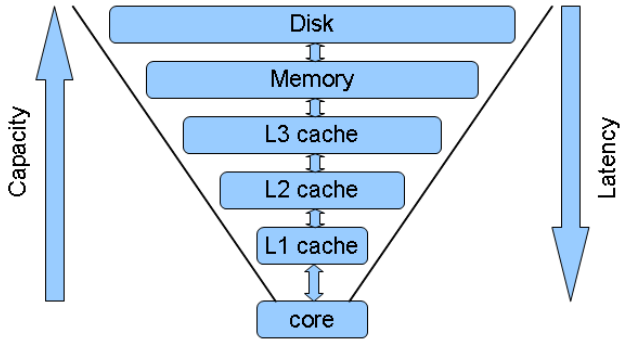

Fig. 3.1 CPU cache structure [141]

The effect of exploiting last level shared cache affinity is considerable, due its sharing among multiple threads and high reloading cost [112]. Intel introduces Intel Smart Cache into their newest CPUs to improve their performance [69].

3.1.2 Cache Regions

The same application behaves discrepant for different problem size [48]. Therefore we introduce new important application parameter *Cache storage requirements CSR* defined in Definition 3.1. It concerns the data re-use and does not refer to cache misses generating for the first usage. Once the data is stored in the cache they should be re-used for the whole algorithm execution.

Definition 3.1 (Cache Storage Requirements). Cache Storage Requirements is the size of the cache memory that algorithm requires to fit the whole data without generating cache misses for data reuse. It is determined by relation (3.1) as a sum of products of the total number of data elements N_i and the memory element size ME_i for each group i of total groups g of memory elements with different size.

$$CSR = \sum_{i=1}^{i=g} N_i \cdot ME_i \tag{3.1}$$

Particular Cache storage requirement determines the algorithm behavior when executed on particular cache. The following definitions 3.2, 3.3 and 3.4 define the particular cache regions L_1, L_2, L_3 and L_4 correspondingly for todays shared memory multiprocessors with memory hierarchy as depicted in Figure 3.1.

Definition 3.2 (First Level Cache Region (L_1 Region)). The algorithm works in cache region L_1 if relation (3.2) is satisfied.

$$CSR \leq CacheSize(\text{L1}) \tag{3.2}$$

Definition 3.3 (Medium Level Cache Region (L_2 and L_3 Region)). The algorithm works in medium level cache regions L_2 or L_2 if relation (3.3) or (3.3) is satisfied correspondingly.

$$CacheSize(\text{L1}) < CSR \leq CacheSize(\text{L2}) \tag{3.3}$$

$$CacheSize(\text{L2}) < CSR \leq CacheSize(\text{L3}) \tag{3.4}$$

Definition 3.4 (Last Level Cache Region (L_4 Region)). The algorithm works in cache region L_4 if relation (3.5) is satisfied.

$$CSR > CacheSize(\text{L3}) \tag{3.5}$$

Lets explain the definitions in more details. The region L_1 is determined as memory size of the L1 cache and its capability to store complete memory requirements. It is assumed that in this case no cache miss will be generated in L1 cache since all relevant data will be stored in the cache. Increasing the problem size N will produce cache misses, which causes to enter in the L_2 region which enables storage of all memory requirements in L2 cache and avoid generation of cache misses on L2 level. The same definitions continue to explain L_3 and L_4 regions.

3.2 Cache Parameters

Previous Section 3.1 presents that introducing multilevel cache memory speedups the memory access. This section analyzes in details all the cache parameters that impacts on the overall performance of the algorithms, particular on cache intensive algorithms.

3.2.1 Cache Size - Capacity problem

Cache memory speeds up the execution only when the data fit in the cache. Only in this case faster cache data access will be exploited rather than much slower main

memory access for the previously accessed data that are already placed in the cache. When the problem size exceeds a particular cache size then the cache misses start generating and the performance decreases.

In Definition 3.5 we introduce a new parameter *Capacity Problem* that impacts the algorithm performance.

Definition 3.5 (Capacity Problem). Capacity problem arises when the condition in relation (3.6) is satisfied, i.e. the cache storage requirement exceeds the size of particular cache level i.

$$CSR > CacheSize(\mathrm{Li}) \tag{3.6}$$

One can propose why don't we use cache with bigger size, but that is not an appropriate solution since that kind of cache will be much slower than the one with the smaller one.

3.2.2 Cache Line (block)

Cache line speeds up the time locality, i.e. if sometimes a particular memory location is referenced, then it is likely that near or even the same location will be referenced again in the near future.

When the processor initiates an access to some data X from the memory this action will result with transfer of the whole cache line (block) in the cache. This activity will transfer all memory elements from the block to be loaded and stored into the same cache line.

Lets denote with cbs the size of cache line in bytes, with ME the size of a memory element. Then the number of elements l that can fit in a cache line is determined with relation (3.7).

$$l = \lceil \frac{cbs}{ME} \rceil \tag{3.7}$$

For example, since todays modern multiprocessor cache line is $64B$, if the algorithm uses double precision data ($ME = 8Bytes$ each), then $l = 8$ elements can fit in a cache line and will be loaded from the memory reading particular element of them.

Cache line is very important for algorithms with arrays or matrices. However, it is more important how they are stored in the memory. For example, C++ stores the data row major, but Fortran column major. Reading the matrix column by column in C++ will be much slower than row by row and opposite for Fortran.

3.2.3 Cache Replacement Policy

If capacity problem arises and a new data is needed, then some cache line in the cache will be replaced with the new data from the lower cache level or main memory. Which cache line will be replaced in this case depends on the *Cache Replacement Policy*. Thus cache replacing policy also impacts the algorithm performance. Three basic cache replacement policies are suggested: Random, Least-Recently-Used (LRU) and First-In-First-Out (FIFO) [63].

Many proposals for cache replacement policies can be found in the literature. Several improvements are proposed for LRU. LRU Insertion Policy (LIP) places the incoming line in the LRU position instead of the MRU [115]. The authors in [35] propose even better replacement policy, i.e. a Score-Based Memory Cache Replacement Policy. Adaptive Subset Based Replacement Policy for High Performance Caching is proposed in [61], i.e. to divide one cache set into multiple subsets and victims should be always taken from one active subset when cache miss occurs. Map-based adaptive insertion policy estimates the data reuse possibility on the basis of data reuse history [72]. The authors in [79] propose Dueling CLOCK cache replacement policy that has low overhead, captures recency information in memory accesses and exploits the frequency pattern of memory accesses compared to LRU. A new replacement algorithm PBR_L1 is proposed for merge sort which is better than FIFO and LRU [47]. The authors in [87] propose LRU-PEA replacement policy that enables more intelligent replacement decisions due to the fact that some types of data are less commonly accessed depending on which bank they reside in. The authors in [78] propose cache replacement using re-reference interval prediction to outperform LRU in many real world game, server, and multimedia applications. However, improving replacing policies requires either additional hardware or modification of existing. PAC-PLRU replacing policy utilizes the prediction results generated by the existing stride prefetcher and prevents these predicted cache blocks from being replaced in the near future [171].

3.2.4 Cache Associativity

This section presents a part of the analysis published by the authors in [120] for the purpose of this thesis research.

Most of modern processors use n-way associative cache where a block can be placed in a restricted set of places in the cache [62]. There are S sets in the cache memory and each set is a group of n blocks in the cache. A block is first mapped onto a set and then the block can be placed anywhere within that set. Equation (3.8) gives the relation for cache associativity with other cache parameters.

$$CacheSize = S \cdot n \cdot cbs \tag{3.8}$$

The cache associativity is another important performance parameter in addition to the cache size, especially for particular problem size. In Definition 3.6 we introduce *Cache Associativity Problem* that arises due to cache associativity.

Definition 3.6 (Cache Associativity Problem). Cache Associativity Problem arises in storage of matrix columns and inefficient usage of cache where the matrix will always map onto a small group of same cache sets and initiate a significant number of cache misses. In this case it looks like the processor is using only a small group of cache sets instead of complete number of sets in associative memory where maximum performance can be achieved. Thus, Capacity problem arises much earlier than relation (3.6) is satisfied.

The worst case appears in the strongest relation (3.9).

$$N > n \tag{3.9}$$

Chapter 9 will present the deeper analysis for dense matrix multiplication algorithm performance gains and drawbacks in n-way associative cache.

3.2.5 Inclusive / Exclusive Cache

Multi level caches has another feature that can differently impact to the performance, inclusive or exclusive multi level caches.

Inclusive caches are the ones that all data of L1 cache is also placed in L2 cache. Since this cache loses the L1 cache size area in L2 cache, the L1 cache should be much smaller than L2. This is the case of Intel CPU. The data in L1 and L2 caches are disjunct in *Exclusive caches*. AMD CPU mostly posses exclusive cache.

3.2.6 Intel Advanced Smart Cache

This section briefly presents the main features of Intel Advanced Smart Cache [69]. Since todays CPUs are multi-cores instead of obsolete single-cores, Intel shares the last level cache between the cores. This sharing stores the data in one place accessible to all cores. Thus the cores dynamically can use up to 100 percent of available last level cache and threads dynamically can use the required cache capacity.

For example, if CPU has 4 cores and 12MB shared cache, if three cores are inactive then the only active core will access to the full 12MB cache.

3.3 Summary

As memory accesses are more demanding operation than computations, we analyze the multilevel memory hierarchy in the modern multiprocessors. In this chapter we define L_1 to L_4 cache regions that directly depend of the size of different cache level. Cache intensive algorithms behave similar for different problem sizes in the same cache region. However, we analyze and the other important cache parameters beside cache size that impact to the algorithm performance: cache line, replacement policy, cache associativity, inclusive / exclusive cache and smart last level cache.

Chapter 4
Matrix Multiplication Algorithm Implementations

Abstract Matrix multiplication is the most common representative of many linear algebra algorithms which performance directly depends of the cache. It is one of the most utilized tasks in many different numerical computations and is an excellent algorithm for parallel computing. This chapter presents on-premise sequential and several parallel implementation that are used as a test algorithm in the experiments in this Part II, and parts III and IV.

4.1 Dense Matrix Multiplication Algorithm

To simplify, squared matrices with dimension N are used. The result product matrix $C_{N\cdot N} = [c_{ij}]$ for all $i, j = 0, 1, \ldots N-1$ is defined in (4.1) by multiplying the multiplier matrix $A_{N\cdot N} = [a_{ij}]$ and the multiplicand matrix $B_{N\cdot N} = [b_{ij}]$ for $i, j = 0, 1, \ldots N-1$.

$$C_{N\cdot N} = A_{N\cdot N} \cdot B_{N\cdot N}, \quad c_{ij} = \sum_{k=0}^{N-1} a_{ik} \cdot b_{kj} \tag{4.1}$$

Each element c_{ij} in (4.1) is calculated as an inner product of row i from matrix A and column j from matrix B.

4.2 CPU Parallelization Requirements

Parallel architecture and tool for parallel execution of parallel implementation of some algorithm are required. This section presents possible architectures and tools that achieve scaled speedup, and even superlinear speedup for dense matrix multiplication algorithm.

4.2.1 CPU Parallel Architectures

Several modern processors use shared L3 cache, besides L1 and L2. In most cases, L1 and L2 are dedicated per core, and L3 is shared per chip. In the following we assume that s is number of chips and c is number of cores per chip in a modern processor.

In this thesis we'll use all 3 possible CPU shared memory environments shown in Table 4.1. These environments differ in number of chips and cores per chip, i.e. values of s and c. In case 1, both $s, c > 1$; in case 2, $s = 1$ and $c > 1$; in case 3, $s > 1$ and $c = 1$. The case where $s = c = 1$ is equal to sequential execution.

Case	Acronym	Environment
1	Multi Chip Multi Core	- Multiple s processors each with multiple c cores. Each core has its own dedicated L1 and L2 cache, and each chip has its own last level shared L3 cache.
2	Single Chip Multi Core	- Single processor with multiple c cores. Each core has its own dedicated L1 and L2 cache, and the processor poses last level shared L3 cache.
3	Multi Chip Single Core	- Multiple s processors each with single core ($c = 1$). Each processor (core) has its own dedicated L1 and L2 cache. In this test case L2 is last level cache since it is in the same time dedicated and shared per core.

Table 4.1 Different Shared Memory Multiprocessors with L3 cache [52]

4.2.2 Runtime Environments for Parallelization

We present four most common runtime environments for parallelization:

- *C++ / Fortran with OpenMP on Linux Server for CPU*. This is the most common used runtime environment for shared memory multi-processors, such as multi-core and multi-chip multiprocessors. The program is written in C++ (or Fortran), and compiled with gcc compiler with option -fopen to use OpenMP [103] API for parallelization and then executed in Linux operating system. We use in the most of the cases this runtime environment;
- *C++ with MPI on Linux Server for CPU*. This is the most common used runtime environment for distributed memory multiprocessors where the intra-processor communication latency is very low, such as supercomputers. The program can be written in C++, and compiled and then executed in Linux operating system. We don't use this runtime environment in this thesis since we are working on shared memory multiprocessors;
- *C# with threading on Windows Server for CPU* This runtime environment can be used for distributed memory multiprocessors where the intra-processor commu-

nication latency is very low, such as supercomputers. The program is written in C# with .NET Framework and compiled and then executed in Windows operating system. We use this runtime environment for Windows Azure environment; and

- *CUDA programming model [101] for GPU* facilitates by tapping into the available computational resources. CUDA programs are accelerated by data-parallel computations of millions of threads, which in this context means instance of a kernel, where krenel is the program running on the GPU device.

4.3 Parallel Implementations on CPU

In the experiment of this thesis 3 different parallel implementations of dense matrix multiplication on CPU are used in order to have maximum efficiency and lower cost.

4.3.1 1D Partitioning Matrix A in Rows

This section presents the algorithm that the authors used in [48].

Matrix A is partitioned on N rows and each row C_i of matrix C is calculated as defined in (4.2) and depicted in Figure 4.1,

$$C_i = A_i \cdot B \tag{4.2}$$

where each row matrix A_i (for $i = 0, 1, \cdots, N-1$) consists of

$$A_i = \begin{bmatrix} a_{i0} & a_{i1} & \cdots & a_{i\,N-1} \end{bmatrix}_{1 \cdot N}$$

and each row matrix C_i (for $i = 0, 1, \cdots, N-1$) consists of

$$C_i = \begin{bmatrix} c_{i0} & c_{i1} & \cdots & c_{i\,N-1} \end{bmatrix}_{1 \cdot N}.$$

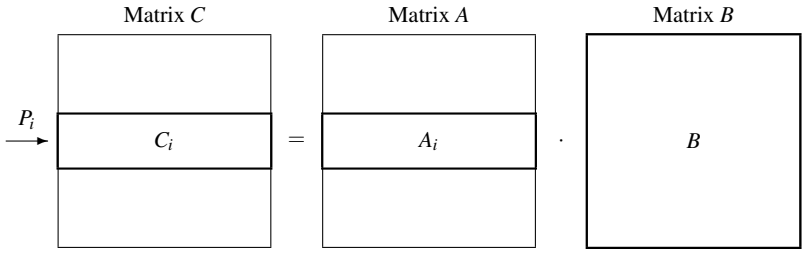

Fig. 4.1 Parallel Matrix Multiplication [48]

To exploit maximum performance for parallel execution on P processors rows A_i are grouped in P sets, such that for all $i = 0, 1, \cdots, N-1$ and $s = 0, 1, \cdots, P-1$ row A_i is placed in Set_s if equation (4.3) is satisfied. The algorithm assures that each row A_i will be placed in exactly one set.

$$s = i \ mod \ P \tag{4.3}$$

Each set Set_s is sent to execution on the processor P_s.

4.3.2 1D Partitioning Matrix A in Blocks

This section presents the algorithm that the authors used in [52].

The idea for this parallel matrix multiplication in a shared memory multiprocessor assumes matrix size N such that P is divisor of N to achieve maximum efficiency and minimum cost. The basic idea is to partition the matrices A and C into smaller P sub matrices (block rows) A_i and C_i respectively, as presented in Figure 4.1.

Let the matrix size $N = q \cdot P$, where P is the number of processing elements, and $q \geq 1$ is an integer. The basic idea is to partition the matrix A into smaller P sub matrices (block rows) A_i

$$A_{N \cdot N} = \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{P-1} \end{bmatrix}$$

where each matrix A_i (for $i = 0, 1, ..., P-1$) consists of

$$A_i = \begin{bmatrix} a_{(q \cdot i)0} & \cdots & a_{(q \cdot i)(N-1)} \\ a_{(q \cdot i+1)0} & \cdots & a_{(q \cdot i+1)(N-1)} \\ \vdots & \vdots & \vdots \\ a_{(q \cdot (i+1)-1)0} & \cdots & a_{(q \cdot (i+1)-1)(N-1)} \end{bmatrix}_{q \cdot N}.$$

Matrix C from (4.1) can be constructed as

$$C_{N \cdot N} = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_{P-1} \end{bmatrix}$$

where

$$C_i = A_i \cdot B_{N \cdot N}, \quad i = 0, 1, ..., P-1. \tag{4.4}$$

In this algorithm each processor computes its own partition C_i of Matrix C by multiplying partition A_i with matrix B.

The P equations defined in (4.4) can be easily parallelized and scaled to P processing elements in a shared memory environment as shown in [124]. A program is developed using C++ and OpenMP that parallelize P equations defined in (4.4) such as each processing element P_i computes its own partition C_i of matrix C.

```
#pragma omp parallel
{
 int id=omp_get_thread_num();
 int num=omp_get_num_threads();
 for(i=id*N/num;i<(id+1)*N/num;++i)
   for(j=0; j<N; ++j)
   {
     double sum = 0.0;
     for (k=0; k<N; ++k)
     {
       sum += A[i*N+k] * B[k*N+j];
     }
     C[i*N+j] = sum;
   }
}
```

The algorithm takes the same number of total calculation steps for the same input, both for sequential and parallel, and avoids cheating to break the law. Thus, the algorithm is *Structure Persistent* according to [140] and therefore, maximum speedup should be limited to the number of processors P.

4.3.3 1D Partitioning Matrix B in Blocks

This section presents the algorithm that the authors used in [130].

This implementation is similar as the partitioning described in Section 4.3.2 but the partitioning is implemented on matrix B. That is, each thread multiplies the whole matrix $A_{N \cdot N}$ and column matrix $B_{N \cdot N/c}$ where c denotes the total number of parallel threads.

4.4 Sequential vs Parallel Complexity and Cache Requirements

Matrix multiplication algorithm is computationally, data and cache intensive and depends of matrix size N. The next two sections 4.4.1 and 4.4.2 present the part of the analysis that the authors published in [48] for matrix multiplication algorithm computational and memory complexity.

4.4.1 Computational Complexity

The algorithm performs N^3 sums and N^3 products, or total $2 \cdot N^3$ operations for sequential execution on one processing element.

The matrix partitioning algorithm explained previously defines different data and code for each processing element in parallel execution. Each processing element in parallel environment executes average $2 \cdot N/P \cdot N^2$ operations. In total the algorithm performs $2 \cdot N^3$ operations for parallel execution, i.e. same as sequential execution on one processing element.

4.4.2 Memory Complexity

The algorithm needs to store $3 \cdot N^2$ elements. If ME denotes the size of one matrix element in bytes, then total memory requirement is $3 \cdot N^2 \cdot ME$ for sequential execution. Cache memory does not need to store three matrices but only two input matrices A and B since the write is directly forwarded to main memory. No matter what cache memory type the processor uses it needs to store $2 \cdot N^2 \cdot ME$ bytes.

Due to shared memory parallel system, last level memory (L4 or main memory in this case) needs to store the whole matrices A, B and C, or total $3 \cdot N^2 \cdot ME$ bytes. Opposite to sequential execution the cache memory requirement directly depends on cache type and level for parallel execution.

Dedicated cache memory per processing element (core) needs to store the whole matrix B with capacity of $N^2 \cdot ME$ bytes and only chunk of matrix A. The amount of necessary space in dedicated per core cache memory is given in (4.5).

$$DedicatedCachePerCore = (1 + 1/P) \cdot N^2 \cdot ME \tag{4.5}$$

L3 cache is usually shared per chip if exists. If s denotes the number of chips the shared multiprocessor consists, then L3 needs to place the whole matrix B with $N^2 \cdot ME$ bytes and around N^2/s elements of matrix A, i.e. around $1/s \cdot N^2 \cdot ME$ bytes. The total amount of necessary space in dedicated per chip cache memory is given in (4.6).

$$DedicatedCachePerChip = (1 + 1/s) \cdot N^2 \cdot ME \tag{4.6}$$

The test environments in [48] use L1 and L2 dedicated per core caches. Each L2 cache should be able to store $DedicatedCachePerCore$ bytes defined in (4.5). The multiprocessor uses $s = 2$ chips with dedicated per chip L3 cache and thus each L3 cache needs to store $DedicatedCachePerChip$ bytes defined in (4.6).

4.4.3 Cache Requirements

In this section we present the analysis that the authors published in [124] for the purpose of this thesis research.

Each element c_{ij} in matrix C is calculated as inner product of row i from matrix A and column j from matrix B. The required number of reads from main memory for each element c_{ij} is $2 \cdot N$, i.e. one row from matrix A and one column from matrix B. This means that each element a_{ij} and b_{ij} need to be accessed N times. If the elements are not present in the cache, they need to be loaded from main memory, which is much expensive operation. Therefore, increasing the matrix size N will occupy the cache faster, increase the cache miss ratio and thus increase the total execution time. We must address that matrix C does not need to be loaded into the cache because the program writes the values c_{ij} in the memory and the program stores the elements c_{ij} only once in the memory.

Let's analyze the cache occupancy by the given algorithms. A part of the cache is occupied by the operating system (OS) for its requirements, usually a small portion. The cache will not generate cache misses if all the matrices A and B are both stored in the cache.

Note that no space is required for matrix C, since the values are computed and stored with write no allocation algorithm directly in main memory. If write allocation is used then a small space will be used in the cache but its dimension is small in comparison to the need of storage of whole matrices A and B.

Suppose that matrices are stored in L1 cache. If matrix dimension increases then there is a need for more space and cache misses generation starts provoking performance degradation. The analysis continues with storage problems in the next level of the caches L2, L3 and so on. The same situation with generation of cache misses happens when L2 (L3) cache will be occupied by both matrices. The performance degradation is presented in Fugire 4.2.

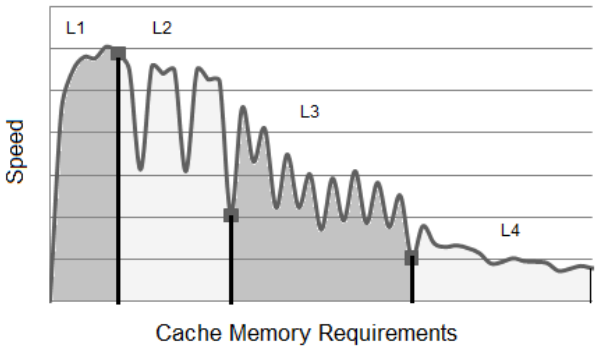

Fig. 4.2 Measured average processor speed with real cache [120]

4.5 Parallelization on GPU

In this section we present the parallel implementation that was proposed by the authors in [33] for the purpose of this thesis research in order to maximum exploit the GPU architecture with matrix multiplication algorithm.

4.5.1 NVIDIA GPU Architecture and Runtime Environment

NVIDIA GPUs have evolved into massively parallel, many-core architectures. These GPUs contain an array of Streaming Multiprocessors (SM), each containing 8 Scalar Processors (SP) for the Tesla architecture [86], 32 SPs for the Fermi architecture [43], and 192 SPs for the latest Kepler architecture [102]. However, CUDA in particular is a Single Instruction Multiple Thread (SIMT) programming model [96], where all threads execute in step the same instruction, but within one SM. On the other hand, threads in different SMs are executing instructions independently from each other, thus providing scalability.

The memory hierarchy of NVIDIA Fermi GPU device is presented in Fugire 4.3. The GPU devices have off-chip memory, so called global memory where average single fetching of data takes at least 400 cycles.

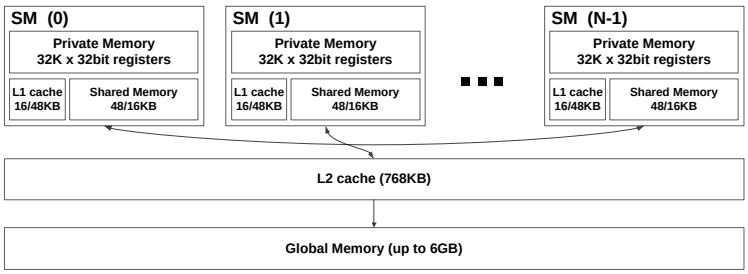

Fig. 4.3 Memory hierarchy of NVIDIA Fermi architecture [33]

The first level in the memory hierarchy is the L1 and shared memory, which is shared by a number of threads organized in thread blocks. It can be accessed almost as fast as register memory and is called private memory which is exclusive to a single thread. L2 cache is off-chip memory and can be accessible from all threads in any SM.

4.5.2 Parallel Implementation on GPU

For simplification, we multiply square matrices of same sizes $N \cdot N$. The basic dense matrix multiplication algorithm is defined by $c_{ij} = \sum_{k=0}^{N-1} a_{ik} \cdot b_{kj}$ where a_{ik}, b_{kj} and c_{ij} are correspondingly elements of matrices A, B and C, for all $i, j = 0, \ldots, N-1$.

The idea is to store greater part of B in the L2 cache and share it among all processes avoiding cache misses as much as possible. Based on the algorithm in [124], we have developed a parallel algorithm for a GPU device. Since multicore processors have larger cache memories it is easier to store the whole matrix B. In GPU the largest cache memory is L2 (736KB) and the matrix B cannot be fitted in L2 for larger problem sizes. However, we solve this, by partitioning the matrix B with the number of available processing elements. An example of two processing elements is presented in Figure 4.4, where the horizontal and the vertical striped matrices (A and B respectively) are multiplied, and the unstriped matrix which is divided in four regions is the resulting matrix C, m is the size of the partitioned submatrix, *am_ost* is the residual of problem size and the number of processing elements, bx stands for the ID of the processing element and jj together with bx indicate which submatrix has to be processed.

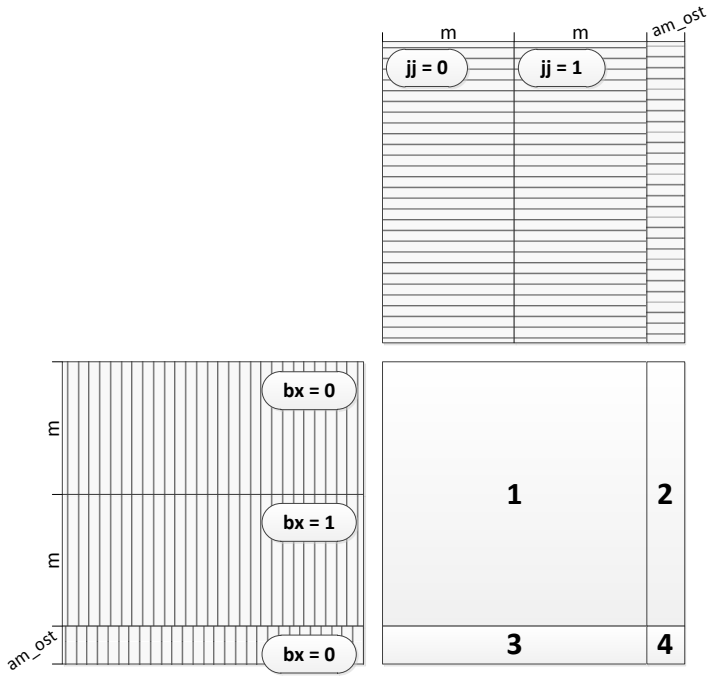

Fig. 4.4 GPU matrix multiplication algorithm [33]

The region 1 in matrix C is calculated for each problem size. However, for problem sizes which is a factor of the number of processing elements, there is no residual and there is perfect alignment with the number of divided submatrices, thus regions 2, 3 and 4 do not have data to calculate.

4.6 Summary

Dense matrix multiplication algorithm is the best representative algorithm of cache intensive algorithms. In this chapter we presents the sequential and several parallel implementations that are used in this thesis research to maximize CPU exploitation. We analyze the algorithm complexities and cache memory requirements for CPU and GPU implementations.

Chapter 5
Web Service Fundamentals

Abstract Web services are the most commonly used technology as a standardized mechanism to describe, locate and communicate with web applications. They are used for collaboration between loosely bound components. This chapter gives a brief overview of web services, their performance factors and several challenges.

5.1 Introduction

W3C defines a *Web service* as a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL) [163]. Effective and ubiquitous B2B systems are being built using web services [31]. Additionally, independence of the underlying development technology enhances web services usage due to the mitigation to development process time and effort [151]. SOAP and REST are two main approaches for interfaces between web site and web services. A high-level comparison of theses approaches is realized in [17]. RESTful web services are more convenient to be hosted on mobile devices than SOAP [93].

Web services usage increases due to their advantages over other types of distributed computing architectures and benefits they provide. Microsoft defines several key benefits for software developers in [95]. Interoperability and usability are the most important ones. Standardization of the web services allows a possibility for developers to reduce learning curve for other web services following the standards. Easy deployment forces IT managers to transfer their services to web service technology.

5.2 Web Service Models

This section presents two different web service client service models as they developed from the traditional client-server model to todays client-server model in modern virtualized enterprise data centers.

5.2.1 Traditional Client-Server Concept

The first generation of web services are *traditional web services* which rely on client - server concept depicted in Figure 5.1. One or many clients request some service from one or several web services hosted on the web server. Web service can call another web service hosted on the same or other web server or can write or retrieve some data from some database server.

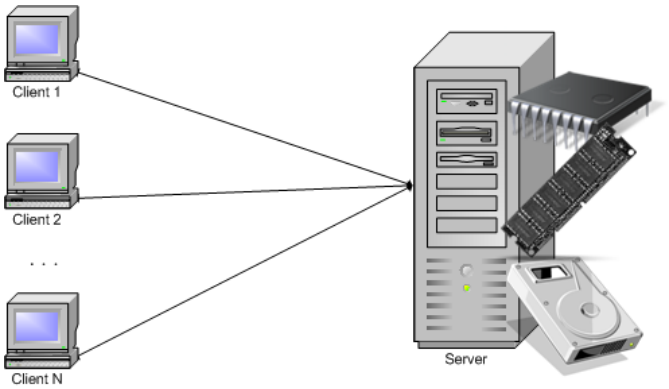

Fig. 5.1 Traditional web service client server model [133]

IT managers propose a hardware, system, network and software resources for web server according to prediction of server average load and risk management of possible peaks. A vast number of this generation servers are either underutilized for small loads and over-utilized for peak loads. The former is desired by the customers as they want the best service performance. The latter is desired by finance department as they want to cut all possible costs.

The solution for underutilization does not exist. Maybe the transfer of server's hardware resources or change the whole web server with other server will mitigate the underutilization.

5.2.2 Client-Server Concept with virtualization

The second generation of web services are *virtualized web services* which also rely on client - server concept depicted in Figure 5.2. Virtualization solves some open issues in the traditional client server model. Web service can now easily utilize more hardware resources (CPU, RAM, HDD) for peak load or release the unnecessary hardware resources when load decreases. However, there will be still some significant service downtime for maintenance, although it will be smaller than traditional client server concept.

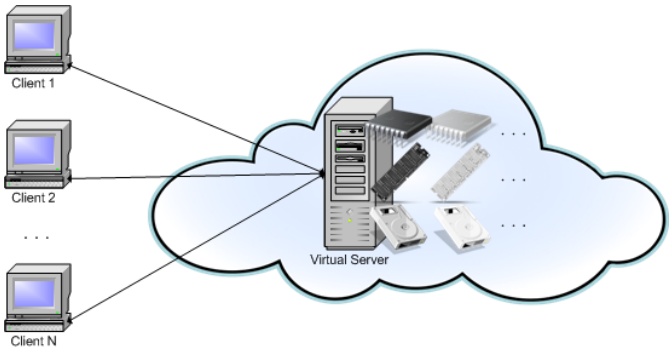

Fig. 5.2 web service client server model hosted in cloud [133]

Cloud computing concept offers on-demand dynamic and elastic resources which improve web service availability and scalability. Nevertheless, there is a service downtime for maintenance, although it will be the smallest compared to other timings for both virtual or traditional client server model.

5.3 Web Service Performance

Web service customers want fast responses for their requests. Therefore web service performance is vital to preserve and even increase web service technology usage. There are external and internal parameters that impact the web service performance. Throughput expressed by the number of concurrent messages in a second, with their size and type are the most important external parameters that depend on customer activities. Web server hardware resources are internal parameters that depend on IT and business managers. Implementing web service security standards outcomes with message overhead and requires complex cryptographic operations for each message, thus decreasing the web service performance. The authors in [137] define quantitative indicators to determine risk of introducing web service security standards for SOAP messages for various message size and number of concurrent messages.

Web service payload is unpredictable most of the time and in most cases. Adding more hardware resources can increase overall web service performance if the number and size of the requests increases. However, the additional hardware will be underutilized for small payload. The costs will be increased due to additional power consumption as well.

Research results about web service performance can be found in many papers in different domains. Web services can be simulated and tested for various performance metrics before they are deployed on Internet servers, which give results close to the real environment [152]. The authors in [148] propose a deserialization mechanism to reuse matching regions from the previously deserialized application objects from previous messages, and performs deserialization only for a new region that would not be processed before. Web service performance in wireless environments and implementing WS-Security are analyzed in [144]. Web server performance parameters response time and throughput are analyzed via web services with two main input factors message size and number of messages in [137]. This thesis extends this research to compare the web service performance with the same input factors in the cloud.

Web servers are usually underutilized since IT managers plan the strategy for hardware resources in advance for the period of several years. Servers are overutilized in peaks which can enormously increase web service response time or even make the services unavailable. Companies can benefit if they migrate their services in the cloud since it offers flexible, scalable and dynamic resources.

5.4 Web Service Improvements - Load Balancing

We found a nice approach for Load Balancing HTTP and Web Services in [88] depicted in Figure 5.3.

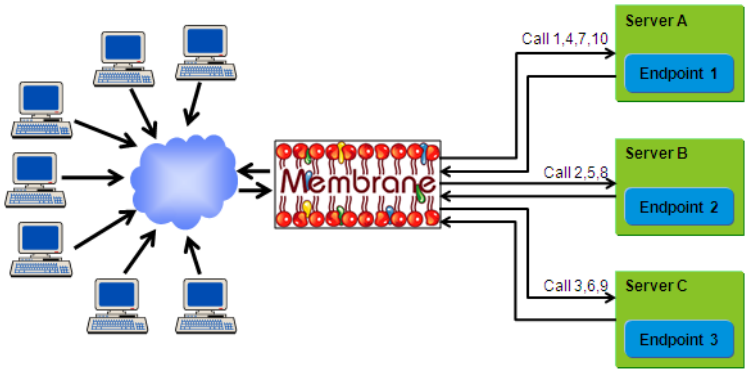

Fig. 5.3 Load Balancing HTTP and Web Services [88]

The authors also present Web Services Loadbalancing in the Amazon Cloud Using Membrane. However, the solution sets up a web services cluster in the Amazon Cloud and the cluster instances always run. Our proposed solution and strategy works with minimum necessary resources and dynamically utilize new resources in peak load.

5.5 Moving Web Services in the Cloud

Chapter 1 presents the cloud concept and the advantages compared to other resource models and concepts. In this section we focus on cloud features to host web services.

5.5.1 The Cloud Challenges

Cloud computing is a paradigm that offers scalable and high quality resources, redundancy, elasticity and multi-tenancy. The concept of cloud computing reduces customers' cost. The on-demand concepts "rent whenever you need" and "pay when you rent" offers the customers to invest the money into their business rather to invest in advance for underutilized ICT equipment. However, the overall cost is not always the key factor in business manager decisions. Cloud computing provides many benefits and detriments to business continuity. A comprehensive analysis for business information system security in cloud computing is given in [126]. Service unavailability for only several hours or even minutes can be source of costs bigger than those for IT equipment.

5.5.2 Migration Challenges

Hosting web services in public cloud can be a good solution for SMEs. However, it provides several open issues: Software Licensing; Security, Privacy and Trust; Cloud Lock-In worries and Interoperability; Application Scalability Across Multiple Clouds; Dynamic Pricing of Cloud Services; Dynamic Negotiation and SLA Management; Regulatory and Legal Issues [16]. It is not an optimal solution for many-tasks scientific computing [70]. The cloud and virtual environments are also worse than on-premise environment for cache intensive algorithms when the data exceeds the cache size [130]. EC2 is slower than a typical mid-range Linux cluster and a modern HPC system for HPC applications due to interconnection on the EC2 cloud platform which limits performance and causes significant variability [76]. However, the cloud provides better performance in distributed memory per core [49].

Implementing security often adds an overhead and outcomes with complex cryptographic operations that always degrades the service overall performance.

Faster web service response time is imperative for both the clients and the providers. A lot of proposals and solutions exist to speedup the web service response time. Algorithm transformation can highly improve the web service performance. Installing more hardware resources on web server is another solution. Cloud computing should facilitate this issue. However, both solutions add additional cost to service providers. The former costs concern additional software developer man hours. The latter costs concern additional OPEX (operating costs) for renting more instances of virtual machines or the instances with more resources and in the most of the cases additional system administrator man hours.

Our intention is to find a solution that will improve the overall performance of web services with less additional costs, or even without it if possible. In Chapter 18 we introduce a middleware layer implementation between the clients and the endpoint web service as a strategy to survive compute peak loads in cloud computing. The experiments prove that although the middleware produces additional latency to overall response time, this solution provides better web service performance for compute intensive web services. This solution reduces the costs for additional hardware only during the peaks and also reduces the system administrator man hours since it automatically starts and shut downs instances with needed resources. Chapter 19 continues the idea introducing message transformation to achieve better performance from particular OS on web server.

5.6 Summary

Web service technology is *sine qua non* in todays business information systems. We overview some web service models, their performance factors and several deployment and migration in the cloud challenges.

Part II
Matrix Multiplication Algorithm Improvements

Chapter 6
Matrix Multiplication Algorithm Analysis

Abstract Matrix multiplication performs $O(N^3)$ operations, demands storing $O(N^2)$ elements and accesses $O(N)$ times each element, where N is the matrix size. Since it is cache intensive algorithm, cache replacement policy is the next important parameter that impacts its performance after arising cache capacity problem. Several cache replacement policies are proposed to speedup different program executions. This chapter analyzes and compares two most implemented cache replacement policies FIFO and LRU. The results of the experiments published by the authors in [9] for the purpose of this thesis research show the optimal solutions for sequential and parallel dense matrix multiplication algorithm. As the number of operations does not depend on cache replacement policy, we define and determine the average memory cycles per instruction that the algorithm performs, since it mostly affects the performance.

6.1 Algorithm Analysis

In this section we analyze the dense matrix multiplication algorithm execution. For better presentation and analysis we use CPU clock cycles instead of execution time. Relation (6.1) derives the total execution clock cycles (TC) as a sum of clock cycles needed for operation execution (CC) and clock cycles needed for accessing the matrix elements (MC) [63].

$$TC = CC + MC \tag{6.1}$$

CC does not depend neither of CPU architecture nor cache size, associativity and replacement policy, but directly depends of matrix size N. CPU executes N^3 sums and N^3 multiplications or total $2 \cdot N^3$ floating points operations. MC is more interesting for analysis. It depends on matrix size N, but also on cache size, associativity and replacement policy.

More important parameters for analysis are the average values of TC, MC and CC defined in the next three definitions [9].

Definition 6.1 (Average Total Cycles Per Instruction). $CPI_T(N)$ for particular matrix size N is defined as a ratio of total number of clock cycles and total number of instructions given in (6.2).

$$CPI_T(N) = \frac{TC}{2 \cdot N^3} \tag{6.2}$$

Definition 6.2 (Average Memory Cycles Per Instruction). $CPI_M(N)$ for particular matrix size N is defined as a ratio of total number of memory cycles and total number of instructions given in (6.3).

$$CPI_M(N) = \frac{MC}{2 \cdot N^3} \tag{6.3}$$

Definition 6.3 (Average Calculation Cycles Per Instruction). $CPI_C(N)$ for particular matrix size N is defined as a ratio of total number of calculation cycles CC and total number of instructions given in (6.4).

$$CPI_C(N) = \frac{CC}{2 \cdot N^3} \tag{6.4}$$

We measure speed, TC, CC, MC for each matrix size, number of cores in defined testing environments. We calculate $CPI_T(N)$, $CPI_M(N)$ and $CPI_C(N)$ and analyze the distribution of $CPI_M(N)$ in $CPI_T(N)$. All the experiments are realized both for sequential and parallel execution.

We measure total execution time TT for each experiment with algorithm described in (4.1) and then calculate TC as defined in [63] and calculate $CPI_T(N)$ using (6.2).

To measure MC we developed another algorithm defined in (6.5) and published in [9]. This algorithm performs the same floating point operations on constant operands and writes the results in matrix C elements. The difference is that it does not read from memory or some cache the elements of matrices A and B.

$$c_{ij} = \sum_{k=0}^{N-1} a \cdot b \tag{6.5}$$

Executing this algorithm we measure its execution time CT for each experiment and then calculate the difference from TC and CT. Then we calculate MC as defined in [63] using CPU speed for particular processor and calculate $CPI_M(N)$ using (6.3).

CC and $CPI_C(N)$ are calculated as defined in (6.6) and (6.4).

$$CC = TC - MC \tag{6.6}$$

6.2 The Testing Environment

Two servers with different CPUs with different cache replacement policies are used: FIFO and LRU. Both servers are installed with Linux Ubuntu 10.10. C++ with OpenMP support is used for parallel execution.

FIFO testing hardware infrastructure consists of one Intel(R) Xeon(R) CPU X5680 @ 3.33GHz and 24GB RAM. It has 6 cores, each with 32 KB 8-way set associative L1 and 256 KB 8-way set associative L2 cache. All 6 cores share 12 MB 16-way set associative L3 cache. Each experiment is executed using different matrix size N for different number of cores from 1 to 6. Tests are performed by unit incremental steps for matrix size and number of cores.

LRU testing hardware infrastructure consists of one CPU Quad-Core AMD Phenom(tm) 9550. It has 4 cores, each with 64 KB 2-way set associative L1 and 512 KB 16-way set associative L2 cache. All 4 cores share 2 MB 32-way set associative L3 cache. Each experiment is executed using different matrix size N on different number of cores from 1 to 4. Tests are performed by unit incremental steps for matrix size and number of cores.

6.3 Results of the Experiments

This section presents the results of realized experiments to determine how different replacement policies impact to dense matrix multiplication cache intensive algorithm.

6.3.1 Results for CPU with FIFO Cache Replacement Policy

Figure 6.1 depicts the results of measured speed. $SpeedT(i)$ denotes the speed in gigaFLOPS for algorithm execution on i cores where $i = 1,2,...,6$.

$CPI_T(N)$ presents another perspective of the experiment. Figure 6.2 depicts the results for algorithm execution on $1,2,...,6$ cores for each matrix size $128 < N < 1000$. We can conclude that executing the dense matrix multiplication algorithm on more cores needs more average cycles per core for each matrix size N. Also, the speed decreases by increasing the matrix size N.

The next experiment analyzes the decomposition of the average total cycles per instruction on average calculation cycles per instructions and average memory cycles per instruction.

Figure 6.3 depicts the absolute decomposition of $CPI_T(N)$ on $CPI_M(N)$ and $CPI_C(N)$ for sequential execution.

The conclusion is that $CPI_C(N)$ is almost constant with average value of 4.93 cycles per instruction. More important is that $CPI_M(N)$ follows $CPI_T(N)$, i.e. $CPI_T(N)$ depends directly of average memory cycles per instruction.

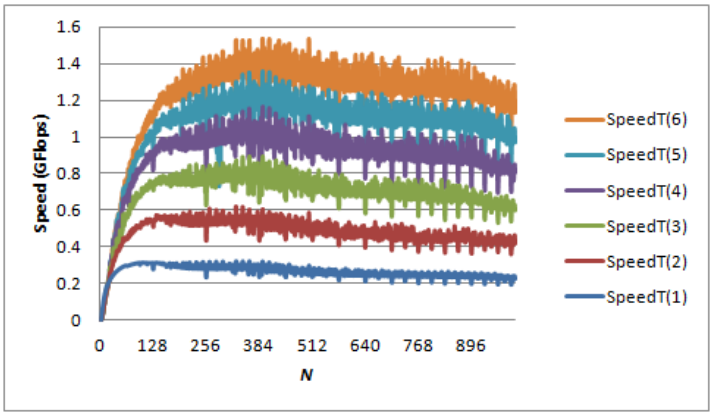

Fig. 6.1 Speed for execution on FIFO CPU [9]

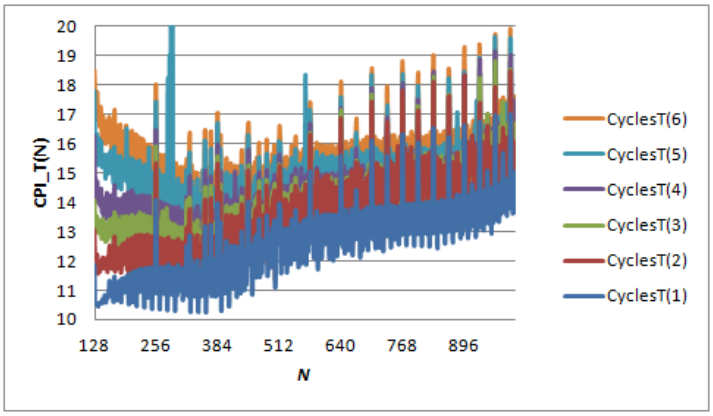

Fig. 6.2 $CPI_T(N)$ for execution on FIFO CPU [9]

Figure 6.4 depicts the relative value of $CPI_M(N)$ to $CPI_T(N)$ for sequential execution.

We can conclude that $CPI_M(N)$ has a trend to equalize with $CPI_T(N)$ as N grows, i.e. for greater matrix size N the total execution time depends directly of average memory access time, instead of time for computations.

6.3.2 Results for CPU with LRU Cache Replacement Policy

Figure 6.5 depicts the results of measured speed. $SpeedT(i)$ denotes the speed in gigaFLOPS for algorithm execution on i cores where $i = 1, 2, ..., 6$. We can conclude

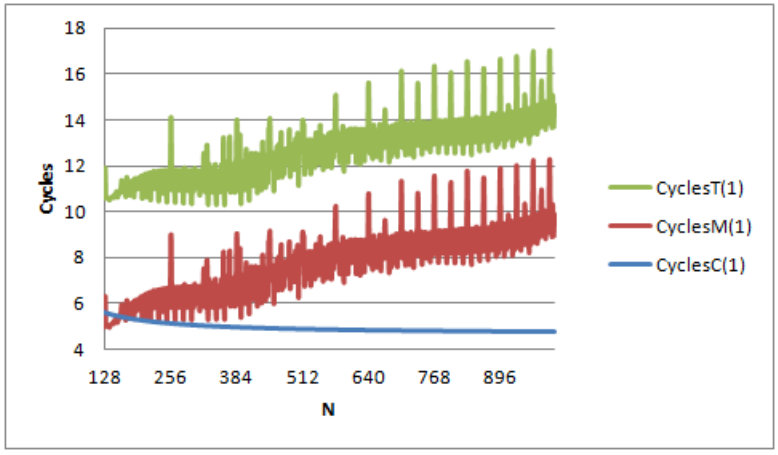

Fig. 6.3 Decomposed $CPI_T(N)$ for sequential execution on FIFO CPU [9].

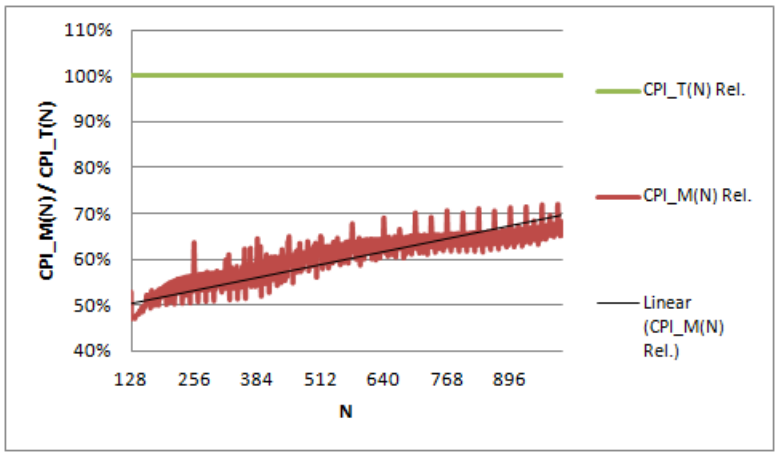

Fig. 6.4 Relative $CPI_M(N)$ to $CPI_T(N)$ for sequential execution on FIFO CPU [9].

that there is a huge performance drawback after $N > 362$ which is entrance in the L4 region, i.e. the region where elements of matrices A and B cannot be placed in L3 cache and thus producing L3 cache miss.

$CPI_T(N)$ presents better the information. Figure 6.6 depicts results for executions on $1,2,3$ and 4 cores for each matrix size $128 < N < 1000$. We can see 2 regions, Region 1 for $N < 362$ and Region 2 for $N > 362$. The former presents the L1 and L2 cache regions, i.e. dedicated per core regions where matrices can be stored completely in L1 and L2 caches correspondingly. In this region sequential execution provides the worst $CPI_T(N)$ compared to parallel execution. The latter presents L3 and L4 regions, i.e. shared memory regions where matrices can and

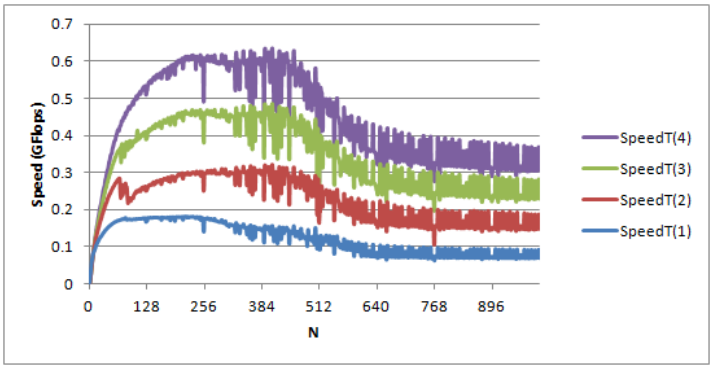

Fig. 6.5 Speed for execution on LRU CPU [9].

cannot be stored completely in L3 cache correspondingly. In this region sequential execution provides the best $CPI_T(N)$ compared to parallel execution.

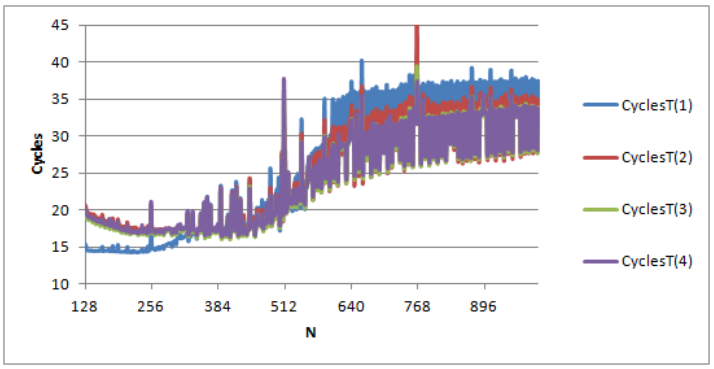

Fig. 6.6 $CPI_T(N)$ for execution on LRU CPU [9].

Figure 6.7 depicts the absolute decomposition of $CPI_T(N)$ on $CPI_M(N)$ and $CPI_C(N)$ for sequential execution.

$CPI_C(N)$ is almost constant to the average value of 7.17 cycles per instruction. More important is that $CPI_M(N)$ follows $CPI_T(N)$, i.e. $CPI_T(N)$ depends directly of average memory cycles per instruction.

Figure 6.8 depicts the relative value of $CPI_M(N)$ to $CPI_T(N)$.

As depicted, $CPI_M(N)$ has a trend to equalize with $CPI_T(N)$ as N grows for $N \cdot (N+1) < 2MB$. This is the case when matrix $B_{N \cdot N}$ and one row od matrix $A_{1 \cdot N}$ can be placed in the L3 cache. $CPI_M(N)$ relative remains constant for greater N.

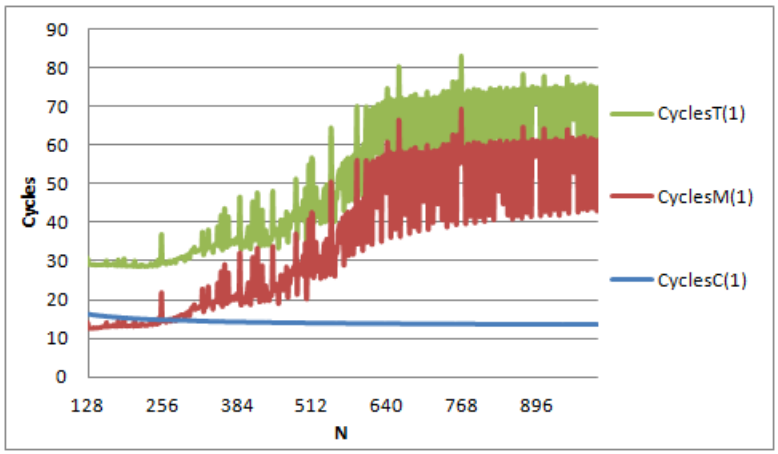

Fig. 6.7 Decomposed $CPI_T(N)$ for sequential execution on LRU CPU [9].

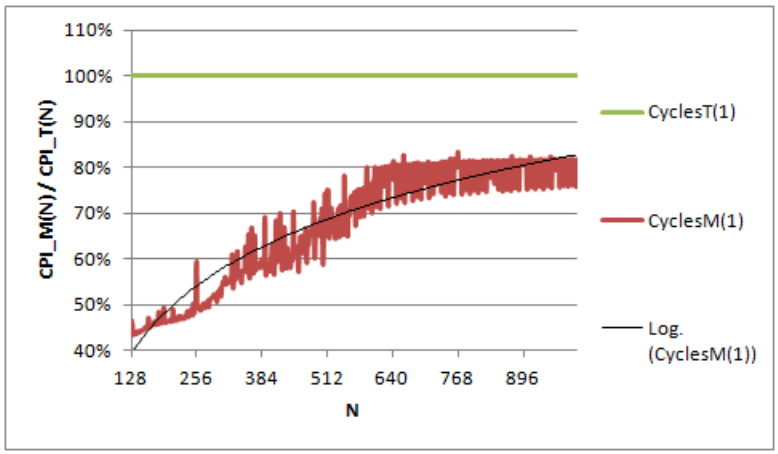

Fig. 6.8 Relative $CPI_M(N)$ to $CPI_T(N)$ for sequential execution on LRU CPU [9].

6.4 LRU and FIFO Cache Replacement Policy Comparison

In this section we compare the results and analyze the difference between performance of FIFO and LRU cache replacement policies.

6.4.1 Speed Comparison

Comapring figures 6.1 and 6.5 we can conclude that both infrastructures have a region around entrance to L3 region when the speed begins to fall down to a local maximum. The graphs show that the speed decrease is more emphasized in LRU rather than FIFO. However, it is because L3 region in LRU begins for $N > 362$ and for FIFO CPU for $N > 886$. Therefore the real comparison should be the regions $N > 362$ on LRU CPU with $N > 886$ on FIFO CPU, which are the beginning of L4 region.

6.4.2 $CPI_T(N)$ Comparison

Comparing figures 6.2 and 6.6 we can conclude that both infrastructures have similar curves for $CPI_T(N)$ for particular region. The important conclusion is that FIFO CPU needs more cycles per core for each matrix size N regardless of cache region (dedicated or shared). However, the LRU CPU has different features. Sequential execution has the best $CPI_T(N)$ in dedicated per core L1 and L2 regions and parallel execution on greater number of cores in shared L3 and L4 regions.

6.4.3 $CPI_T(N)$ Decomposition Comparison

Comparing figures 6.3 and 6.7 we can conclude that both infrastructures have similar curves for $CPI_T(N)$. The graphs show that $CPI_T(N)$ is greater in LRU than FIFO. However, the real comparison should be the regions $N > 362$ on LRU CPU with $N > 886$ on FIFO CPU as explained in the previous subsection. $CPI_M(N)$ is almost parallel compared to $CPI_T(N)$ for all matrix size N in both infrastructures. Also, the similar result is the fact that $CPI_C(N)$ is almost constant for each matrix size N for both CPUs.

6.4.4 $CPI_M(N)$ Comparison

Comparing figures 6.4 and 6.8 we can conclude that $CPI_M(N)$ is relative more closer to $CPI_T(N)$ in LRU than FIFO. However, it is because L3 region in LRU begins for $N > 362$ and for FIFO CPU for $N > 886$. Therefore the real comparison should be the regions $N > 362$ on LRU CPU with $N > 886$ on FIFO CPU, which are the beginning of L4 region and the relative values in LRU CPU are better than FIFO CPU. LRU CPU has average of 59.77% in the region of $N = 362$ and FIFO CPU has average 65.84% in the region of $N = 886$.

6.5 Summary

We determined that both cache replacement policies provide similar speed and average cycles per instruction $CPI_T(N)$ for sequential and parallel execution. However, the results show that LRU replacement policy provides best $CPI_T(N)$ for sequential execution in dedicated per core cache memory. Parallel execution provides the best $CPI_T(N)$ in shared memory LRU CPU, i.e. LRU produces greater speedup than FIFO and is more appropriate rather than FIFO cache replacement policy for dense matrix multiplication algorithm.

Chapter 7
Matrix Multiplication Algorithm Simulation

Abstract An optimal architecture to execute certain compute and memory intensive algorithm is desirable in most applications. This chapter presents the MMCacheSim simulator published in [11] for the purpose of this thesis research. MMCacheSim simulator predicts matrix multiplication performance on particular existing or non-existing multiprocessor. It simulates the execution time and number of cache misses that matrix multiplication algorithm performs with particular matrix size and element size executing on processor with different cache size, line, level associativity, and replacement policy. Parallel execution of the matrix multiplication algorithm can be simulated also on dedicated / shared cache memory per core in shared memory multiprocessor. The experiments prove MMCacheSim's accuracy especially for sequential execution.

7.1 Introduction to Simulators

All cache parameters presented in Chapter 3 impact the algorithm overall performance and it is difficult to select the cache with optimal parameters for particular algorithm. Even more, the same algorithm behaves differently for different input size data. Applications provide better performance when they are executed on flexible cache with reconfigurability [150]. Using a proper simulators to predict the algorithm performance can save time and wasted money for unnecessary hardware. They can be used to measure the performance of new proposed schemes [8]. The authors in [20] propose techniques to predict the performance impact using hybrid analytical models. The authors in [172] propose a technique to overcome inter-thread cache conflict misses on shared cache and develop a highly configurable multi-core cache contention MCCCSim simulator that reproduces parallel instruction execution. A predictive model is proposed in [169] to allow fast and accurate estimation of system performance degradation also due to shared cache contention in parallel execution. The authors in [37] propose a statistical cache model Statstack that mod-

els a fully associative cache with LRU replacement policy and compared the results with the output from a traditional cache simulator.

There are two types of simulation: trace or execution driven simulation. The former is the case when the simulator analysis a list of pre generated memory addresses by a program and then uses them to simulate the memory activities. The latter is the case when the simulator is active while the program is running and intercepts the accesses to the memory to do the simulation of the cache. Trace driven simulation is inconvenient because of the large trace files that need to be created and transferred [77].

Simulation based methods depend on accurate data like complete memory traces and produce results that are completely correct. Estimation based methods use heuristics to complete their simulations and finish the task faster, but with lower accuracy levels.

7.2 Literature Review

This section presents different purpose cache simulators that we found in the literature. Dinero IV is the cache simulator that simulates a memory hierarchy with various caches [36]. A DEW strategy [60] speeds up the simulation of multiple combinations of cache parameters. It simulates only FIFO replacement policy. The authors in [41] define a fully parameterizable models applicable to n-way associative caches, but only for LRU replacement policy. Our MMCacheSim simulates both FIFO and LRU cache replacement policies and all levels of cache hierarchy.

The authors in [77] propose a CMP$im simulator based on the Pin binary instrumentation tool. It is a better simulator offering multi core support and data gathering for all levels of the cache. However, the capturing the results is more complex than our MMCacheSim. HC-Sim is also based on Pin that generate traces during runtime and simulates multiple cache configurations in one run [20]. An on-line cache simulation using a retargetable application specific instruction set simulator is provided in [118]. CMPSched$im evaluates the interaction of operating system and chip multiprocessor architectures [94].

Simulators can be also used in the teaching process. Hardware courses in software oriented curriculum require a lot of effort, both from instructors and students [145]. The authors in [134] using visual simulators, incrementally weighted exercises, and finally working on real hardware controllers achieved significant improvements in grade distribution and computer science student interest in hardware. Visual EduMIPS64 helps teachers to better present the specific topics of computer architecture and also help students to better learn [110].

We present our MMCacheSim simulator and analyze if a successful prediction of cache performance can be achieved by simulating the execution of an algorithm and measuring the number of misses on different levels of CPU cache. We build a model that can be easily configured to represent different types of cache architectures with different replacement policies. Series of experiments were performed for

execution of dense matrix multiplication algorithm on real world implementations and simulation with same parameters for the CPU cache architecture. We have also performed simulation of the parallel execution of the same algorithms and compare with the results of real experiments with the same cache parameters.

7.3 MMCacheSim architecture

The MMCacheSim simulator is implemented as a set of Java classes, each representing a different CPU cache element:

- *Cache Line* - Represents a single cache line. It is initialized with the size of the cache line, the size of the elements saved inside it, and the address of the first element saved inside. Contains methods for writing new elements in the cache line and checking if an element is in the cache line;
- *Cache Set* - Represents a collection of cache lines available for both LRU and FIFO implementations as cache replacement policies. It is initialized with the associativity and line size. Contains methods for writing an address inside the cache and with it replacing the obsolete one according to the chosen replacing policy, checking whether an address is inside the given set;
- *L1, L2, L3 Cache Levels* - The actual cache memory, also available as LRU and FIFO implementations initialized with the size, associativity and the cache line size. Contains the cache sets, the data about misses and hits made on that particular level and methods for reading from and writing to the level;
- *Processor Core* - As a real processor core would have access to the cache. Several cores may share same cache structures. The simulated model of a core is initialized with instances of cache levels, by giving different cores the same instance of a cache level we simulate sharing. A cache core has only method to read a data element. If the element is not found in the cache levels a cache miss is recorded;

The MMCacheSim simulates execution of the simple dense matrix multiplication algorithm. The simulation does not take into account the time required for arithmetic operations and memory writes because we are looking for the effect that the cache produces when the same data is accessed multiple times and the speedup that can be gained when parallelizing the execution. The input for MMCacheSim is number of cores, cache levels, shared / dedicated cache per core, cache line, cache size, and cache replacement policy for each cache as input parameters. It returns the average clock cycles for cache hit per each cache level and cache miss for last level cache. It also measures the total clock cycles for accessing the data.

7.4 Experiment Environment

The experiments are performed on the real multiprocessors with totally different cache architectures. The first multiprocessor consists of 2 chips Intel(tm) Xeon(tm) CPU X5680 @ 3.33GHz and 24GB RAM. Each chip has 6 cores, each with 32 KB 8-way set associative L1 data cache dedicated per core and 256 KB 8-way set associative L2 cache dedicated per core. All 6 cores share 12 MB 16-way set associative L3 cache. The second server has one chip AMD Phenom(tm) 9950 Quad-Core Processor @ 2.6 GHz and 8 GB RAM. The multiprocessor has 4 cores, each with 64 KB 2-way set associative L1 data cache dedicated per core, and 512 KB 16-way set associative L2 cache dedicated per core. All 4 cores share 2 MB 32-way set associative L3 cache. The real machines are installed with Linux Ubuntu 10.10. C++ with OpenMP for parallelization are used.

The simulation is platform independent since it does not measure algorithm response time, but only simulates the number of cache misses and hits on different levels of simulated CPU cache. We simulate the same 2 multiprocessors as the experiments.

7.5 The Results of the Experiments

The first performed test is to determine the number of CPU cycles needed to access different levels of the cache in the simulated architectures. The same experimental tests are executed on both servers. Table 7.1 presents the results. These results of the practical experiments are used in the simulation.

Parameter	Xeon	Phenom
L1 hit	2.85	6.5
L2 hit	14.01	23.63
L3 hit	21.04	37.30
L3 miss	126.04	113.06

Table 7.1 Cache hit and miss cost in cycles [11]

Figure 7.1 depicts the comparison of the simulation of matrix multiplication on a cache with FIFO replacement policy and cache parameters as Intel(tm) processor.

The horizontal axis represents the matrix size. The vertical axis represents the average number of memory accesses *MA* to each element of a matrix calculated as defined in (7.1). The values for total memory access cycles from the simulator are calculated using the values from Table 7.1 as defined in [63].

$$MA = \frac{TotalMemoryAccessCycles}{N^3} \tag{7.1}$$

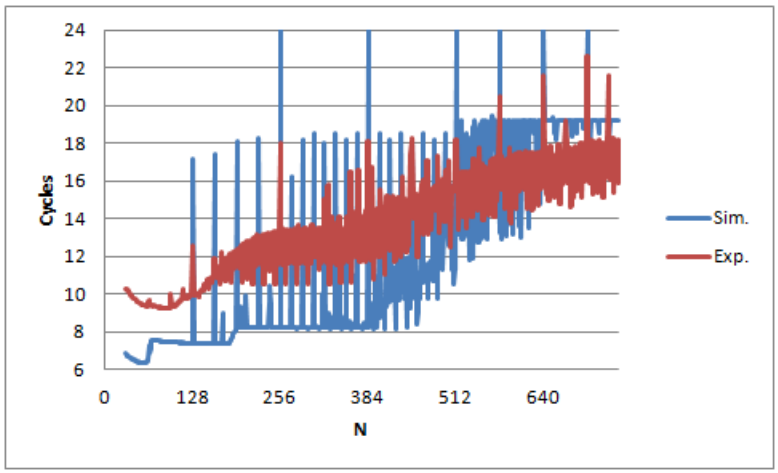

Fig. 7.1 Comparison CPU cycles for memory access for MMCacheSim simulation and sequential execution on Xeon server with FIFO replacement policy [11]

The results show that MMCacheSim gives values close to the experimental ones and successfully simulates the performance drawbacks due to cache associativity.

The final experiment was to simulate with a new replacement policy Bit-Pseudo-LRU. Each cache line is associated with a MRU bit (most recently used) in this cache replacement policy. When the line is read the MRU bit is set to 1. When all lines in a cache set have their MRU bits set to 1, they are reset to 0. If some cache line should be replaced then the cache line in a set with the largest index that has a MRU bit 0 is replaced.

Figure 7.2 depicts that the simulation is much closer to the experimental values. The simulation is still stepping away as the sizes of the matrices exceed the size of the cache memory. A possible explanation to the differences are: the authors in [25] show that the L3 cache at the Opteron processors uses some kind of pseudo-LRU cache replacement policy. The way of choosing the line inside the set seems to be different than the proposed Bit-PLRU policy. This is a logical explanation of the differences between simulated and experimental results, with the assumption that the cache replacement policy described in [25] is used in Phenom processors too. However this shows the ability to use the simulator not just to find favorable configurations for a certain algorithm but to research the configuration of a computer system when data for it are not available.

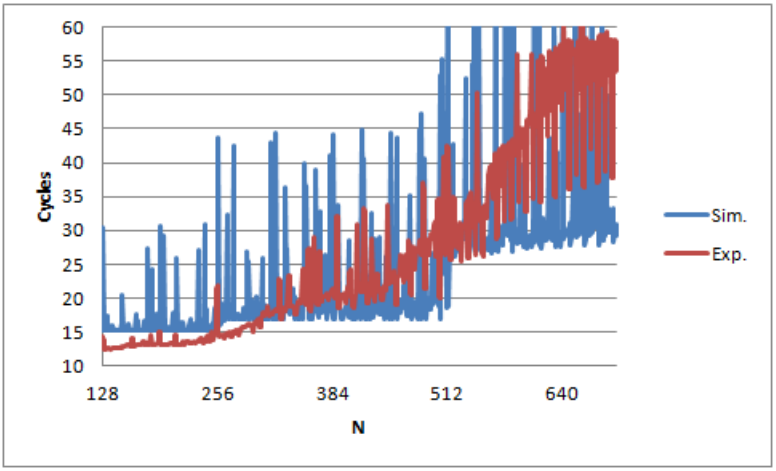

Fig. 7.2 Comparison of CPU cycles used for memory access for sequential execution on Phenom CPU and simulate with Bit-PLRU replacement policy [11]

7.6 Summary

Our MMCacheSim simulator simulates main FIFO and LRU cache replacement policies and it is easy to implement other policies. All levels of cache hierarchy can be simulated. It is platform independent since the cache parameters are input parameters in the simulator.

MMCacheSim can simulate both sequential and parallel implementation of matrix multiplication algorithm. It can be easily used to simulate any algorithm by giving a trace of memory accesses. The modularity of the implementation allows any type of cache hierarchy to be simulated. MMCacheSim allows to change:

- The hierarchy between cache levels, to be shared between cores or dedicated;
- The inclusivity between different cache levels;
- The size of the cache memory, the associativity, cache line sizes; and
- Replacement policy, with ability to have different cache replacement policies per different cache levels.

Chapter 8
Matrix Multiplication Algorithm Improvements

Abstract Multiplication of huge matrices generates more cache misses than smaller matrices. 2D block decomposition of matrices that can be placed in L1 CPU cache decreases the cache misses since the operations will access data only stored in L1 cache. However, it also requires additional reads, writes, and operations compared to 1D partitioning, since the blocks are read multiple times. This chapter presents a new hybrid 2D/1D partitioning to exploit the advantages of both approaches which the authors proposed in [53]. The idea is first to partition the matrices in 2D blocks and then to multiply each block with 1D partitioning to achieve minimum cache misses. We select also a block size to fit in L1 cache as 2D block decomposition, but we use rectangle instead of squared blocks in order to minimize the operations but also cache associativity. The experiments show that our proposed algorithm outperforms the 2D blocking algorithm for huge matrices on AMD Phenom CPU.

8.1 Matrix Multiplication Algorithm Optimizations

Matrix multiplication algorithm is a basic linear algebra operation used in almost all scientific computations. Different techniques are proposed to speedup the execution. Another very important issue is the selection of appropriate powerful hardware environment for efficient execution. Fused multiply-add (FMA) units with in modern CPU architecture execute both addition and multiplication in one clock cycle. Since matrix multiplication is compute intensive algorithm with $O(N^3)$ complexity increasing the processor speed will speedup the execution. It is also a memory demanding algorithm with $O(N^2)$ complexity. Although memory complexity is smaller than compute, it impacts more the overall performance due to average memory cycles per instruction is 50 to 80% from the average total cycles per instruction [9]. However, the most important is the fact that matrix multiplication algorithm is cache intensive algorithm with $O(N)$ complexity since each matrix element is accessed N times for different computations and the memory access per element vary

between several clocks for the lowest L1 cache memory and up to 1000 for main memory [63].

Introducing multi-chip and multi-core CPUs, and many-core GPU processors together with different APIs and libraries for parallelization can significantly speedup the execution since matrix multiplication is excellent algorithm for parallelization and thus can maximize the efficiency. It can achieve almost linear speedup. However, there are regions where superlinear speedup can be achieved for multi-chip, multi-core and GPU implementations as presented in Part III. It is also possible in multi-GPU implementation on Fermi architecture GPU, due to configurable cache [113].

Another focus is optimizing the algorithm if the matrices have some features like symmetric, zero rows or columns, sparse, squared or triangle. The authors in [34] optimized multiplication of small sized matrices. Superlinear speedup was reported for sparse symmetric matrix vector multiplication in [142]. [80] reports also superlinear speedup is also found with parallel execution of matrix multiplication algorithm using MPI and transposing one source matrix. The authors in [12] improved parallel execution based on Strassen's fast matrix multiplication minimizing communication.

Optimizing the algorithm according to CPU cache, platform and runtime environment is the most appropriate approach. The authors in [164] used padding to the first element of each submatrix to land on equidistant cache sets and to avoid cache associativity performance drawbacks [50]. The authors in [63] propose compiler optimizations with loop interchange and blocking to reduce cache miss rate. The optimal rectangular partitioning can be significantly outperformed by the optimal non-rectangular one on real-life heterogeneous HPC platforms [32] .

In this chapter we analyze blocking algorithm for matrix multiplication and propose a new hybrid 2D/1D blocking method for matrix multiplication that exploits the maximum of the L1 cache size by reducing the number of compute operations and cache associativity problems.

8.2 Existing 2D Blocking matrix multiplication algorithm

The 2D blocking matrix multiplication algorithm described in [63] reduces the number of cache misses. It works on submatrices or blocks of matrices A and B with same size b instead of the entire matrix A rows and matrix B columns. Figure 10.10 depicts the algorithm. The goal is to maximize the reuse of the data of a block before they are replaced.

The total number of operations in this algorithm is increased. Additional $N/b \cdot N/b \cdot N \cdot b = N^3/b$ are performed for summing the elements of matrix C. Including the necessary $2 \cdot N^3$ operations the total number of floating point operations is $2 \cdot N^3 + N^3/b$. The benefit of these additional operations is the decreased number of memory accesses of $N/b \cdot N/b \cdot N \cdot b \cdot 2 = 2 \cdot N^3/b$ in the worst case.

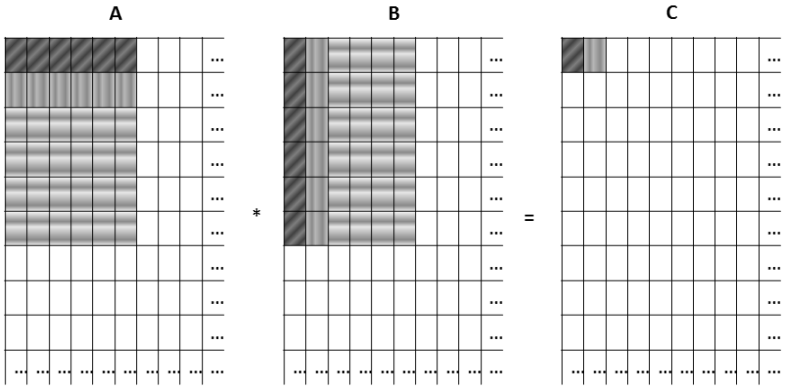

Fig. 8.1 2D Blocking matrix multiplication algorithm [53]

Maximum efficiency for parallel execution of this algorithm is realized when the product $P \cdot b^2$ is divisor of N^2.

8.3 Hybrid 2D/1D Blocking matrix multiplication algorithm

This section describes our new proposed Hybrid 2D/1D Blocking matrix multiplication algorithm, its complexity, performance, and comparison with the other algorithms. Also we analyze how different cache parameters impact the algorithm performance.

8.3.1 Decrease the Operations and Memory Accesses

The idea is to exploit the benefits of both algorithms described in Section 8.2, i.e. to use 2D blocking to minimize the memory access, but in the same time minimizing the additional operations for the elements of matrix C. Figure 8.2 depicts the algorithm for hybrid blocking matrix blocking matrix multiplication. We propose rectangles with the same area to be used instead of squares for blocks. Next, dense matrix multiplication algorithm is implemented for multiplication in each block.

Lets denote with b the squared block size of 2D blocking algorithm and with b_X and b_Y the number of rows and columns for block of matrix B correspondingly such that relations (8.1) and (8.2) are satisfied

$$b_X > b_Y \tag{8.1}$$

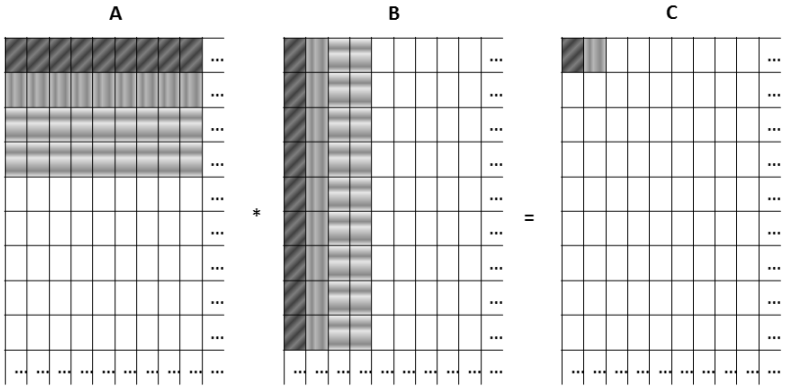

Fig. 8.2 Hybrid matrix multiplication algorithm [53]

$$b_X \cdot b_Y = b^2 \tag{8.2}$$

Lets denote with TMA the number of total memory accesses. Lemma 8.1 presents how TMA depends of matrix size N and the block sizes b_X and b_Y for new Hybrid 2D/1D Blocking matrix multiplication algorithm.

Lemma 8.1. *The total number of memory accesses TMA that new Hybrid 2D/1D Blocking matrix multiplication algorithm performs if blocks in matrix A consist of b_Y rows and b_X columns is defined in relation (8.3).*

$$TMA = \frac{N^3}{b_X} \tag{8.3}$$

Proof. The algorithm creates $N/b_Y \cdot N/b_X$ blocks per matrix and accesses each element of those blocks $N \cdot b_Y$ times. Thus $TMA = N/b_Y \cdot N/b_X \cdot N \cdot b_Y \cdot 2 = 2 \cdot N^3/b_X$.

Theorem 8.1 proves that new Hybrid 2D/1D Blocking matrix multiplication algorithm performs smaller total number of memory accesses TMA_H than the basic 2D Blocking matrix multiplication algorithm.

Theorem 8.1. *New Hybrid 2D/1D Blocking matrix multiplication algorithm performs smaller total number of memory accesses TMA_H than the basic 2D Blocking matrix multiplication algorithm TMA_{2D}, i.e. relation (8.4) is true.*

$$TMA_H < TMA_{2D} \tag{8.4}$$

Proof. According to Lemma 8.1 $TMA_H = \frac{N^3}{b_X}$ and $TMA_{2D} = \frac{N^3}{b}$. Using relation (8.1) in relation (8.2) yields $b_x > b$. Multiplying this relation with $\frac{N^3}{b_X \cdot b}$ yields relation 8.4 that proves the theorem.

Corollary 8.1. *Maximum efficiency for parallel execution on P same multiprocessors of this algorithm is realized when the product $P \cdot b_X \cdot b_Y$ is divisor of N^2.*

Proof. Let k is integer, such that $k \geq 1$ and $k = \frac{N^2}{P \cdot b_X \cdot b_Y}$. This yields then the product $P \cdot k = \frac{N^2}{b_X \cdot b_Y}$ is also integer. Since N^2 is the total number of matrix elements, and $b_X \cdot b_Y$ is the number of elements in a block, then the matrix can be divided into $P \cdot k$ blocks. These blocks can be scattered to P processors and executed in k steps with maximum efficiency.

8.3.2 The Algorithm and the Cache Parameters

Since matrix multiplication algorithm is cache intensive algorithm, lets analyze the cache impact to the new algorithm compared to 2D blocking algorithm. The previous Section 8.3.1 presents that hybrid algorithm executes smaller number of operations and memory accesses than 2D blocking. But is it enough to be a faster algorithm?

Cache has several parameters that have a huge impact to the overall performance especially for particular matrix size. Since the block area is the same for both algorithms cache size impact is also the same, i.e. the same cache misses will be generated only when the block is changed. Therefore cache replacement policy will not impact also.

What about cache line? Lets denote with l the number of matrix elements that can be placed in one cache line. Modern CPUs cache line size is 64B, i.e. it can store $l = 8$ matrix elements (double precision) which can be loaded in one operation from the memory. Our new algorithm defines that b_X should be greater as much as possible to reduce the operations and memory accesses, but still (8.2) should be satisfied. However, to exploit maximum performance the same cache line should be present in the L1 cache, and thus $b_Y < l$.

Further on we analyze *cache associativity problem* that appears in storage of matrix columns and inefficient usage of cache for particular problem size [50]. Matrix B blocks will map onto a smaller group of cache sets than the same 2D blocking matrix multiplication algorithm and initiate more cache misses. In this case our algorithm will use a smaller group of cache sets in associative memory.

Todays Intel L1 cache is 8-way set associative and AMD L1 cache is only 2-way set associative. For particular matrix sizes N it means that Intel CPU will provide better performance than AMD due to cache set associativity for the same block size $b_X \cdot b_Y$.

8.3.3 Modified Hybrid 2D / 1D matrix multiplication algorithm

To avoid cache associativity problem especially for AMD CPU we make another experiment where b_X is smaller and b_Y is maximized. Figure 8.3 depicts this algorithm. This algorithm has more operations and memory accesses than others, but will be prone to cache size associativity, i.e. L1 cache works as fully associative.

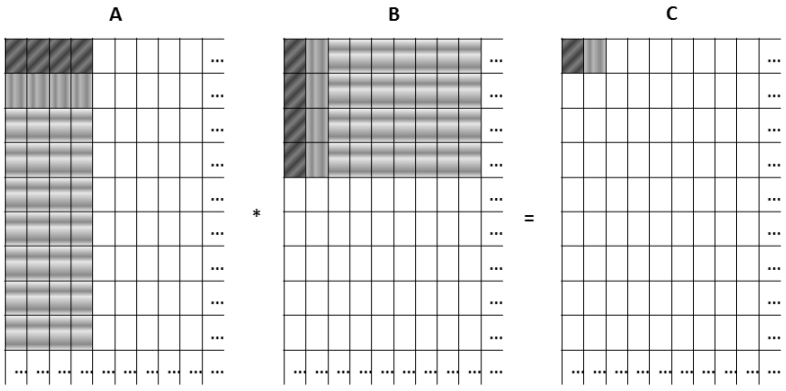

Fig. 8.3 Modified hybrid matrix multiplication algorithm [53]

8.4 The Testing Methodology

This section present used testing methodology for the experiments in order to prove that our new proposed basic and modified hybrid 2D / 1D algorithms provide better performance than basic 2D blocking algorithm.

8.4.1 Testing Environment

The experiments are performed on two multiprocessors with different cache architectures since the hardware can impact to the algorithm performance. The first multiprocessor consists of 2 chips Intel(tm) Xeon(tm) CPU X5680 @ 3.33GHz and 24GB RAM. Each chip has 6 cores, each with 32 KB 8-way set associative L1 data cache dedicated per core and 256 KB 8-way set associative L2 cache dedicated per core. All 6 cores share 12 MB 16-way set associative L3 cache. The second server has one chip AMD Phenom(tm) 9950 Quad-Core Processor @ 2.6 GHz and 8 GB RAM. The multiprocessor has 4 cores, each with 64 KB 2-way set associative L1

data cache dedicated per core, and 512 KB 16-way set associative L2 cache dedicated per core. All 4 cores share 2 MB 32-way set associative L3 cache. The servers are installed with Linux Ubuntu 10.10. C++ with OpenMP for parallelization are used without additional optimizations.

8.4.2 Test Data

We execute dense, 2D blocking, hybrid and modified hybrid matrix multiplication algorithms on each multiprocessor with different matrix sizes to test algorithm behavior in different cache regions. For 2D blocking algorithm we choose $b = 36$ for Intel CPU and $b = 48$ for AMD to satisfy that matrices can be stored in the L1 cache. For our two hybrid algorithm experiments we use $b_x = 162$ and $b_y = 8$ for hybrid matrix multiplication algorithm and $b_x = 8$ and $b_y = 162$ for modified hybrid matrix multiplication algorithm for Intel CPU. For AMD CPU $b_x = 288$ and $b_y = 8$, and $b_x = 8$ and $b_y = 288$ are used.

8.5 The Results of the Experiments

This section presents the results of the experiments realized to measure the performance of the new hybrid 2D/1D matrix multiplication algorithms compared to 2D blocking.

8.5.1 The Results on Intel CPU

Figure 8.4 depicts the execution time for dense, 2D blocking, hybrid and modified hybrid matrix multiplication algorithms on Intel CPU. For $N < 1296$ all matrix multiplication algorithms run similar. Dense algorithm increases its execution time for greater matrices more than modified hybrid 8x162. Hybrid 162x8 and blocking 2D are the best matrix multiplication algorithms.

Figure 8.5 depicts the speed on Intel CPU. Dense performs the best speed for smaller matrices as expected due to huge L3 size. However, for huge matrices all other matrix multiplication algorithms perform better speed compensating the increased number of operations with lower average access time per matrix element. We can conclude that all three other algorithms have constant speed. Our proposed hybrid matrix multiplication algorithm provides the same speed as 2D blocking, and the modified hybrid matrix multiplication algorithm is worse.

We can conclude that cache associativity problem does not impact directly on Intel CPU. However, the results do not confirm our hypothesis that our new 2D/1D will be better than 2D blocking. After deep analysis we give the following expla-

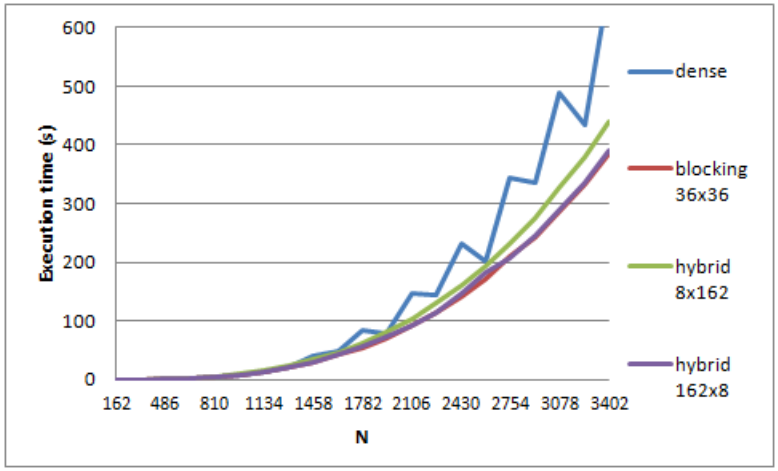

Fig. 8.4 The execution time for sequential execution on Intel CPU [53]

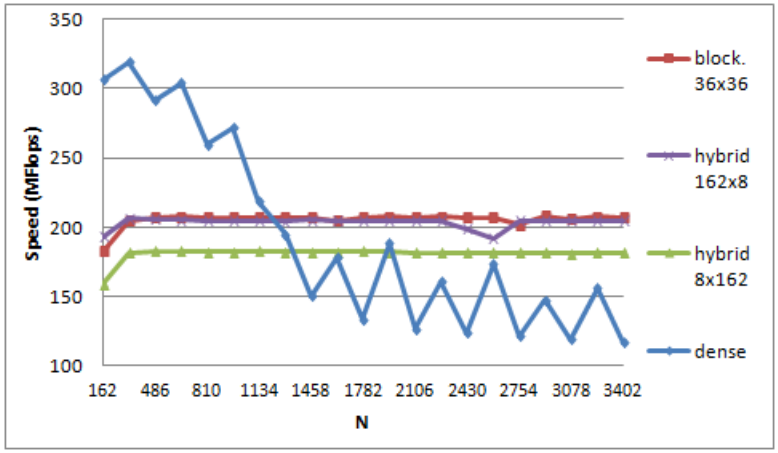

Fig. 8.5 The speed for sequential execution on Intel CPU [53]

nation. The L1 cache is occupied also from operating system. Cache misses appear from the operating system and the whole cache line is replaced.

8.5.2 The Results on AMD CPU

Figure 8.6 depicts the execution time for the same four algorithms on AMD CPU. For smaller matrices the execution time is similar for all algorithms, but for greater matrices both our algorithms outperform the 2D blocking and dense.

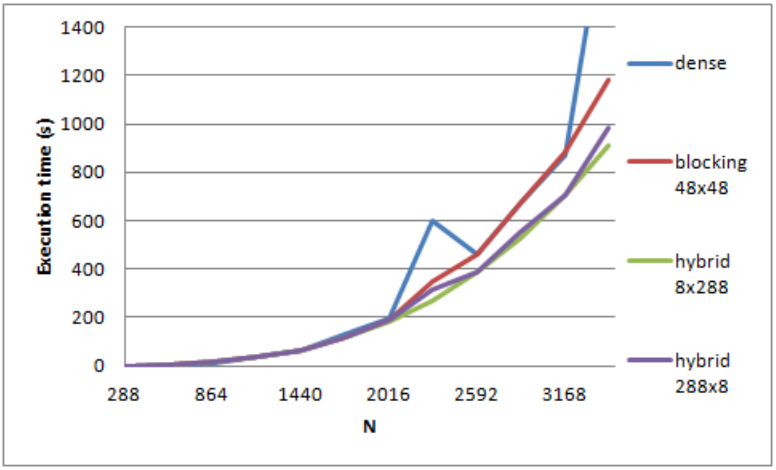

Fig. 8.6 The execution time for sequential execution on AMD CPU [53]

Figure 8.7 depicts the speed for better presentation and analysis. As depicted, Dense is the leader in front of 2D blocking, hybrid and modified hybrid for small matrices. However, increasing the matrix size N, the order is opposite: modified hybrid achieves best performance in front of hybrid, 2D blocking and Dense.

Only our modified hybrid algorithm retains the speed regardless of N. It is prone to the cache associativity problem. All other algorithms achieve drawbacks in particular N due to cache associativity problem. We see that our hybrid matrix multiplication algorithm has similar performance as modified hybrid in points where there aren't cache associativity performance drawbacks [50].

8.6 Summary

The new hybrid 2D/1D blocking matrix multiplication algorithms provide similar performance on Intel CPU as the 2D blocking matrix multiplication algorithm, and better performance on AMD Phenom CPU. Using theoretical analysis of all cache parameters that can impact the algorithm performance we modify the proposed algorithm and even improved already better performance than 2D blocking matrix mul-

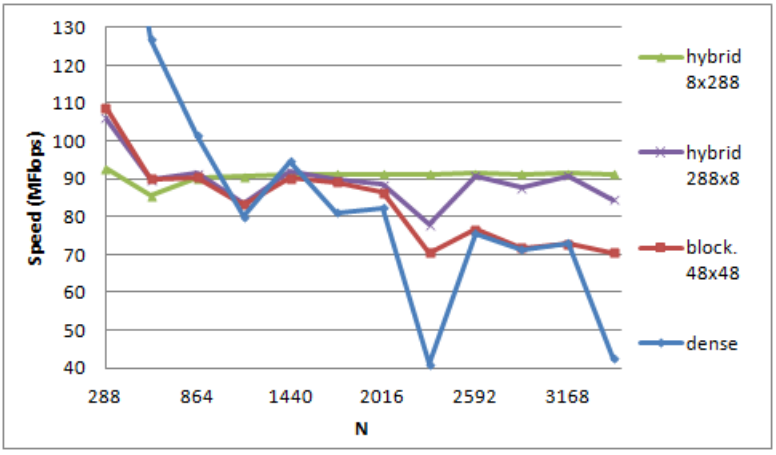

Fig. 8.7 The speed for sequential execution on AMD CPU [53]

tiplication algorithm. Even more, the modified algorithm is prone to small cache set associativity on AMD CPU caches. The experiments prove the theoretical analysis.

Chapter 9
Performance Drawbacks Using Set Associative Cache

Abstract Performance drawbacks are commented in literature without detailed explanation. Analyzing the interaction of different matrix multiplication algorithms with the memory hierarchy, the authors in [154] present performance drawback for different problem sizes. More detailed analysis of the effect of the cache and TLB is done in [58] where the authors discovered execution time peaks for particular matrix sizes and conclude that peaks occur more often for smaller cache. The authors also conclude that the peaks in CPU execution time are due to increased number of memory access conflicts. For the purpose of this thesis research the authors in [120, 50, 123] analyze and model the performance drawbacks for particular problem sizes for sequential and parallel execution. Given theoretical proof of execution time peaks when using set associative cache for cache intensive algorithms such as matrix multiplication is presented in this chapter. Huge performance drawbacks are observed and analyzed for speed in sequential execution. The authors conclude that performance drawbacks appear due to increased number of generated cache misses in last level L3 cache. For parallel execution also huge performance drawbacks are observed for speedup beside those for speed.

9.1 Storing matrix elements in set associative cache

Most of modern processors use n-way associative cache where a block can be placed in a restricted set of places in the cache [62]. There are S sets in the cache memory and each set is a group of n blocks in the cache. A block is first mapped onto a set and then the block can be placed anywhere within that set.

In this paper we analyze the organization how matrices are stored in the cache. When the processor initiates an access to the element b_{kj} from the memory this action will result with transfer of the whole cache line (block) in the cache. This activity will transfer all elements $b_{k,j+1}, b_{k,j+2} \dots b_{k,j+l-1}$ from the block to be loaded and stored into the same cache line as shown in Figure 9.1.

If *cbs* is the cache block size in bytes and *ME* is the matrix element size (usually 8 bytes for double precision real numbers) then l is number of matrix elements that can fit in a cache line expressed as an integer by (3.7).

If one element is accessed then the other $l-1$ elements from the same row will be used without generation of cache misses. If the element b_{kj} is accessed, such that $N-j<l$, then $N-j$ elements until the end of the row will be placed in the cache, together with the first $l-(N-j)$ elements from the next matrix row. This organization enables efficient usage of matrix row elements and is desired when the algorithm repeatedly uses matrix rows, for example, the matrix multiplication algorithm.

However, this is not the case for matrix column elements. When the processor initiates an access to the element b_{kj} it will not load the desired matrix column elements $b_{k+1,j}, b_{k+2,j}, \ldots$. Two problems are initiated by this organization. The first is known as matrix cache storage problem and the second is associated by the usage of n-way associative cache. The matrix cache storage problem appears due to the inefficient load of matrix elements that will not be used. In practice whenever a matrix column element is required then $l-1$ elements will be loaded in the same cache line and possibly will never be used if the cache line is replaced by some other least recently used.

This chapter analyzes the problem initiated by usage of n-way set associative cache memory. A set will be fully loaded with n cache lines. Each cache line contains consecutive matrix elements from a particular matrix row. Cache lines that map on the same set will have a specific address pattern usually expressed as modulo function.

Denote by d the smallest number such that the elements $b_{k,j}$ and $b_{k+d,j}$ map onto same set as shown in Figure 9.1. In this case d represents the address offset.

Lemma 9.1 defines the number of the blocks m necessary to store N matrix row elements.

Lemma 9.1. *The number m such that N matrix row elements will be stored in m cache blocks is defined in (9.1).*

$$m = \lceil N/l \rceil = \lceil N \cdot ME/cbs \rceil \tag{9.1}$$

Proof. Since number l is the number of matrix elements that can fit in a cache line expressed as an integer, the number of blocks m necessary to store N matrix row elements in cache block can be calculated as $m = \lceil N/l \rceil$. Using relation (3.7) proofs the relation 9.1 and this lemma.

Next issue is to calculate the number of cache blocks required to store the matrix column with N elements. We assume big sizes $N >> l$ so we do not expect that two matrix column elements will fit in the same cache block. In this case the requirement is N different cache blocks.

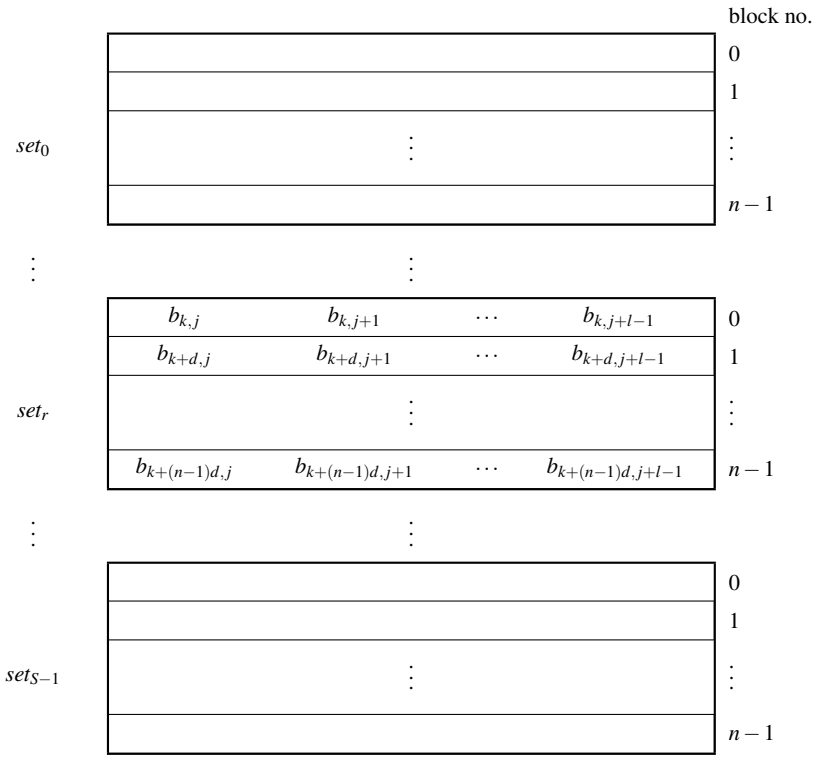

Fig. 9.1 Storing column matrix B in a n-way set associative cache [120]

9.2 Performance drawbacks in a n-way set associative cache

Matrix multiplication algorithm defined in Section (4.1) has two points of possible drawbacks. The first drawback appears if a cache block is replaced after a particular element is accessed so the next access to this element will generate a cache miss. This drawback impacts the elements of both matrices.

The second type of drawback refers to a situation where after an access to a particular element there is an access to another element from the same block but in meantime the block was replaced. This impacts only the elements of matrix B since the elements of one cache block of matrix A are accessed consecutive. This is not the case with matrix B. Once the whole matrix B column is accessed the cache blocks also contain the consecutive columns. Replacing such cache block during the loop execution will produce $l-1$ cache misses per block. In addition the first drawback repeats since each matrix element is accessed N times [124].

We are interested in a situation where matrix column elements will be mapped since there is a requirement to store N elements in at most S different sets. By analyzing the pattern where the matrix column elements will be stored we will be

able to find out the cache space available for storage of matrix column and the frequency of cache misses generation for the elements $b_{k,j+1},...,b_{k,j+l-1}$.

Theorem 9.1 [120] proves conditions for the case when the address pattern is such that maps all column elements of the matrix B onto the same group of sets. This situation will generate a lot of cache misses and it looks like we are using only a small partition of the cache instead of the whole cache.

Theorem 9.1. *Cache performance drawback in the matrix multiplication algorithm generated due to matrix storage pattern in a n-way associative memory appears if there exists an integer d such that*

$$\exists d > 0, \qquad d = \frac{S \cdot cbs}{N \cdot ME} \quad and \quad d < \frac{N}{n} \tag{9.2}$$

Proof. The relation shows how the address pattern d relates to the available S sets and matrix size N. This pattern actually shows that the address pattern is such that all the column elements will be stored in the same set.

To find the pattern how the addresses map onto cache we have to calculate d. The matrix row is stored in consecutive memory addresses in row-wise main memory and according to (9.1) each matrix row will require m cache blocks. We assume that several matrix rows will be stored in different sets. The maximum possible occupancy is valid if $m \cdot d = S$. Therefore the equation (9.2) follows from (9.1) if *cbs* is divisor of N.

The number of matrix column elements to be stored is N and the number of cache blocks in a set is n meaning that the matrix column elements will be stored in at most N/n sets, which represents the last part of the relation.

The number d has an interesting property as follows [120].

Corollary 9.1. *d in Theorem 9.1 is the number of different sets used to store a matrix column and indirectly shows the portion of cache used to store the matrix column. If $d = 1$ then all the elements of a column are mapped in the same set and if $d > 1$ then exactly d sets will be used in the mapping process.*

Proof. Normally this number is not bigger than the number of sets $d \leq S$ in the n way associative cache. The proof can be constructed based on Dirichlet principle to place objects into different places.

Corollary 9.2. *The maximum drawback for Theorem 9.1 appears if $d = 1$.*

Proof. Proof can be constructed upon conclusion of Corollary 9.1 and the smallest available cache portion to store the matrix column leading to generation of most cache misses. $d = 1$ yields $m = S$ which means that the each block of the first row of matrix B will be store in all S different sets, and based on Dirichlet principle next element of the column will be placed in the same set as the element from the first row and the same column.

Corollary 9.3. *Cache performance drawback in the matrix multiplication algorithm appears due to matrix storage pattern in a n-way associative memory for*

$$N = \frac{CS}{n \cdot d \cdot ME} \tag{9.3}$$

Proof. The proof can be constructed from the relation that the number of sets is $S = CS/(n \cdot cbs)$ and by equation for d in (9.2).

Example 9.1 (Opteron processor [165]). Matrix multiplication algorithm executed on Opteron processor using a $CS = 512KB$ 8-way L2 set associative cache with $cbs = 64B$ according to Corollary 9.3, will have performance drawback for matrix sizes

$$N = CS/(n \cdot d \cdot ME) = \frac{2^{13}}{d}.$$

The conclusion is that performance drawbacks will appear when N is power of 2. Corollary 9.2 shows the maximum drawback and worst performance for matrix size $N = 2^{13}$.

Theorem 9.2. *Cache performance drawback in the matrix multiplication algorithm generated due to matrix storage pattern in a n-way associative memory appears if there exists an integer d such that*

$$\exists d > 0, \qquad d < \sqrt{\frac{S \cdot cbs}{n \cdot ME}} \tag{9.4}$$

Proof. Proof can be constructed from (9.3) of Corollary 9.3 and the last part of the (9.2) from Theorem 9.1.

Theorem 9.3. *Cache performance drawback in the matrix multiplication algorithm appears for matrix size N such that*

$$N > \sqrt{\frac{S \cdot cbs \cdot n}{ME}} \tag{9.5}$$

Proof. Proof can be constructed directly from both relations in equation 9.2 from Theorem 9.1 $S = CS/(n \cdot cbs)$ and by equation for d in (9.2).

Corollary 9.4. *Cache performance drawback in the matrix multiplication algorithm appears if the size of matrix B is greater than CS.*

$$N^2 \cdot ME > CS \tag{9.6}$$

Proof. The proof can be constructed from the relation 9.5 of Theorem 9.3 and that the cache size is $CS = S \cdot cbs \cdot n$.

Example 9.2 (Phenom processor [166]). Quad-Core AMD Phenom(tm) Processor 9550 has 4 cores each with its own dedicated 64 KB instruction and 64 KB data L1

Cache	Type
L1	2-way associative 64 KB instruction + 64 KB data cache
L2	16-way set associative 512 KB cache
L3	32-way associative 2 MB cache

Table 9.1 Cache type [120]

Variable	Value in L1	Value in L2	Value in L3
CS	64KB	512KB	2MB
n	2	16	32
ME	8B	8B	8B
cbs	64B	64B	64B
l	8	8	8
S	512	512	1024
$N \cdot d$	4096	4096	8192

Table 9.2 Cache variables for the experiments [120]

cache, dedicated 512KB L2 cache and share 2MB shared L3 cache. Tables 9.1 and 9.2 present cache behavior and calculated values.

According to Corollary 9.3 the matrix multiplication algorithm executed on this processor using a 16-way L2 set associative cache with $cbs = 64B$ will have performance drawback for matrix sizes:

$$N = CS/(n \cdot d \cdot ME) = \frac{2^{12}}{d}.$$

Theorems 9.2 and 9.3 show that the maximum drawbacks and worst performances for L2 cache are for $d = 1, 2, 4, 8, 16$ or matrix sizes $N = 4096, 2048, 1024, 512, 256$. According to Corollary 9.2 the maximum drawback and worst performance is for $N = 2^{12}$.

The hypotheses that are confirmed experimentally in [120] and more detailed in [50] rely on theoretical results defined by theorems 9.1 to 9.3, i.e.:

- Expressive L1 cache associativity drawbacks appear for matrix sizes N = 4096, 2048, 1024, 512, 256 and 128;
- Expressive L2 cache associativity drawbacks appear for matrix sizes N = 4096, 2048, 1024, 512 and 256;
- Expressive L3 cache associativity drawbacks appear for matrix sizes N = 4096, 2048, 1024, and 512.

Nevertheless, smaller cache associativity drawbacks are possible for smaller N.

9.3 Experiments for Performance Drawbacks in Sequential Execution

The experiments are realized for problems that use matrix sizes where theoretical results show expectation of main drawbacks, and also in the area around these points, for example, for the ten points -10, -9, -8, -7, -6, -5, -4, -3, -2, and -1 bellow the expected drawbacks and for the ten po-ints +1, +2, +3, +4, +5, +6, +7, +8, +9, and +10 above the expected drawback points. Each experiment executes the sequential algorithm with and without profiler Valgrind [155] with its tool cachegrind. The execution times and speeds for these areas are analyzed. We focus on L1 and L3 (last level) data cache misses in order to prove their impact on performance drawbacks appearance. We have not tested the behavior of L2 cache since Valgrind tests measure only first and last level caches. The X-axis in each figure in this section presents the matrix size N.

9.3.1 Experiment 1 - range around $N = 64$

Experiment 1 covers the area of $N = 64$. Matrix B can be stored completely in L1 cache, but the elements of matrix A replace some of the cache blocks from matrix B, thus producing cache misses and performance drawback. The drawback is not visible enough for execution time depicted in Figure 9.2, but it is visible presenting the speed in Figure 9.3. An important result is that speed has a positive trendline.

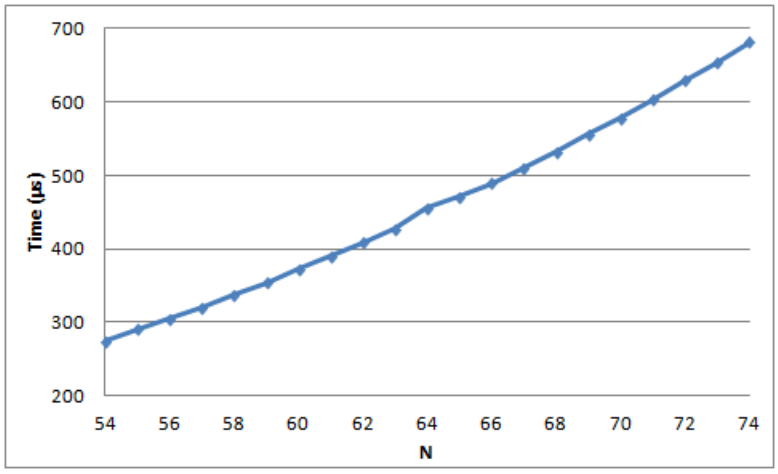

Fig. 9.2 Execution time in the area around $N = 64$ [120]

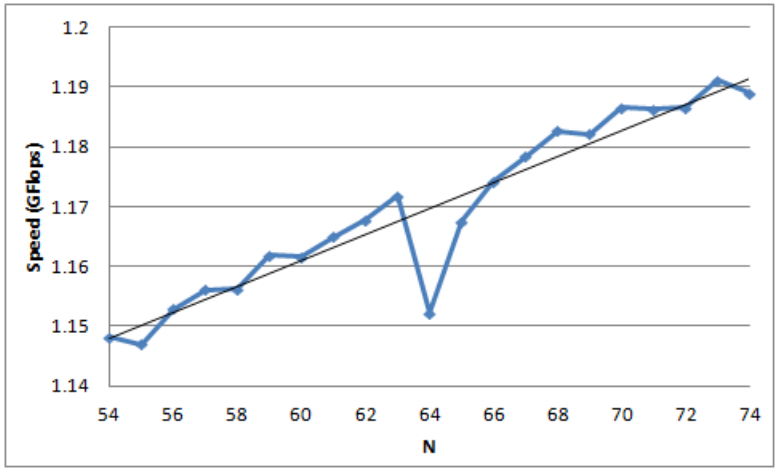

Fig. 9.3 Speed in the area around $N = 64$ [120]

Figures 9.4 and 9.5 depict the L1 and L3 data cache misses for the experiments in the range around $N = 64$. There is only a small insufficient L1 cache associativity drawback in $N = 64$. L3 associativity drawback does not appear in this area due to this is L1 region. There are no other local extremes neither for L1 nor for L3 data cache misses.

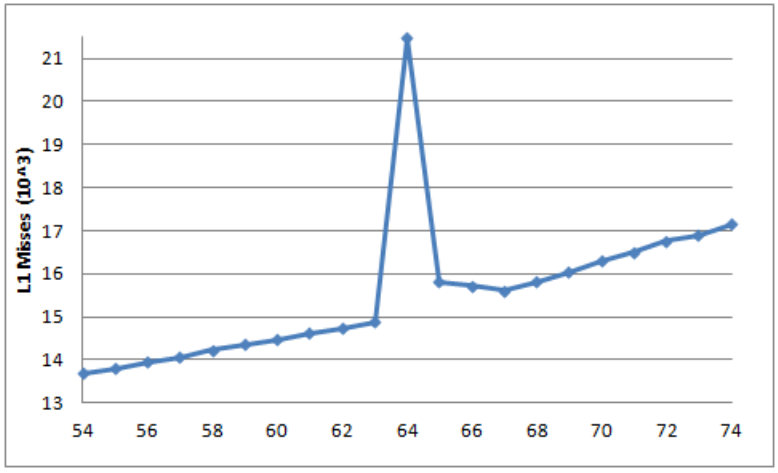

Fig. 9.4 L1 data cache misses in the area around $N = 64$ [50]

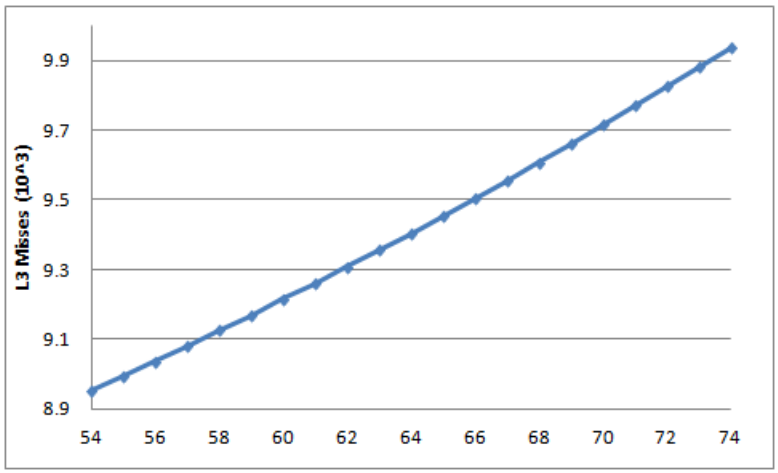

Fig. 9.5 L3 data cache misses in the area around $N = 64$ [50]

9.3.2 Experiment 2 - range around $N = 128$

Experiment 2 covers the area of $N = 128$. Matrix B cannot be stored completely in L1 cache and drawbacks appear due to insufficient L1 cache and L1 and L2 cache set associativity. The drawback is visible both for execution time and speed in figures 9.6 and 9.7 correspondingly. Local extremes are near the main drawback generated due to L1 cache set associativity. Speed in this region also has a positive trendline, but lower than Experiment 1.

Figures 9.8 and 9.9 depict the L1 and L3 data cache misses for the experiments in the range around $N = 128$. There is an expressive L1 cache associativity drawback in $N = 128$. L3 associativity drawback does not appear in this area since this is L_2 region and L3 works as full associative cache memory. There are local extremes only for L1 data cache misses.

9.3.3 Experiment 3 - range around $N = 256$

Experiment 3 covers the area of $N = 256$. Matrix B cannot be stored completely in L2 cache and drawbacks are generated due to L1, L2 and L3 cache set associativity. The drawbacks are visible both for execution time and speed in figures 9.10 and 9.11 correspondingly. Local extremes are near the main drawback generated due to L1, L2 and L3 cache set associativity. Another important result is that speed in this region has a negative trend since the transition from L_2 to L_3 region.

Figures 9.12 and 9.13 depict the L1 and L3 data cache misses for the experiments in the range around $N = 256$. There is an expressive L1 cache associativity drawback

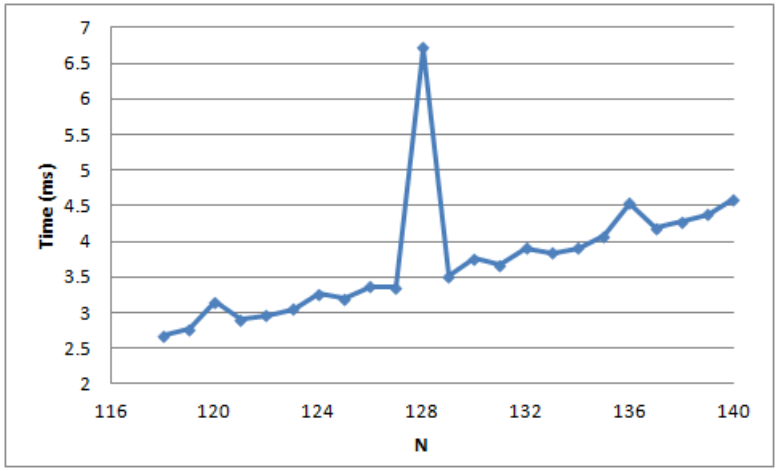

Fig. 9.6 Execution time in the area around $N = 128$ [120]

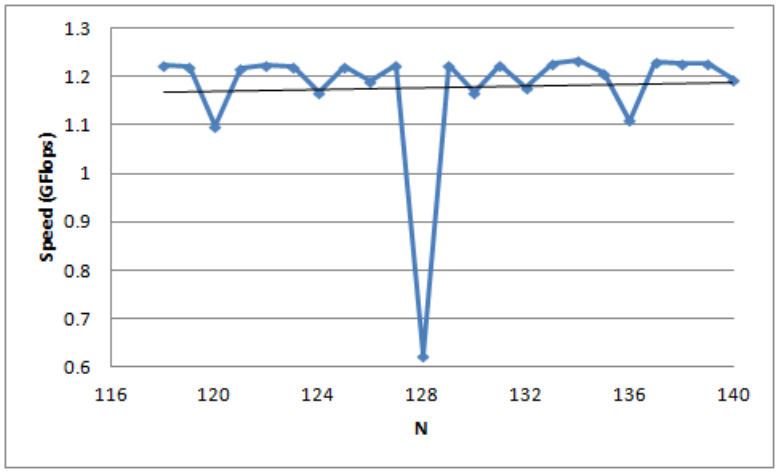

Fig. 9.7 Speed in the area around $N = 128$ [120]

in $N = 256$. Another expressive cache associativity drawback but smaller than main drawback is found for $N = 256$. L3 cache associativity drawback is also expressive. Figure 9.13 depicts the huge growth of L3 cache data misses due to entrance in L_3 region. There are local extremes for L1 data cache misses. L3 data cache misses local extremes are found only in L_2 region, i.e. for $N < 256$.

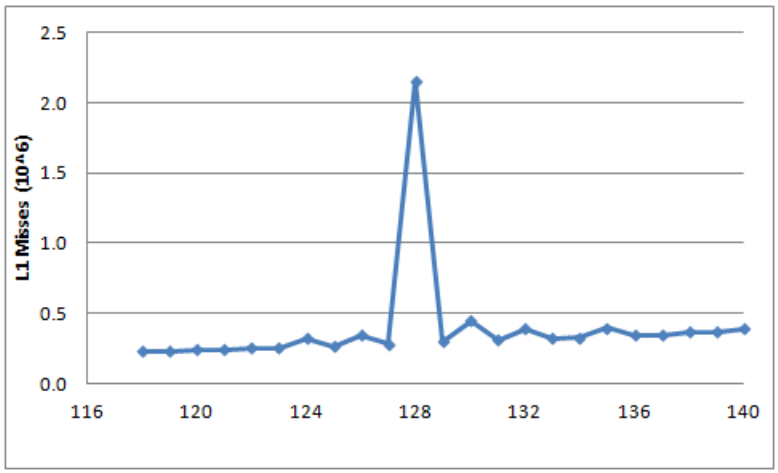

Fig. 9.8 L1 data cache misses in the area around $N = 128$ [50]

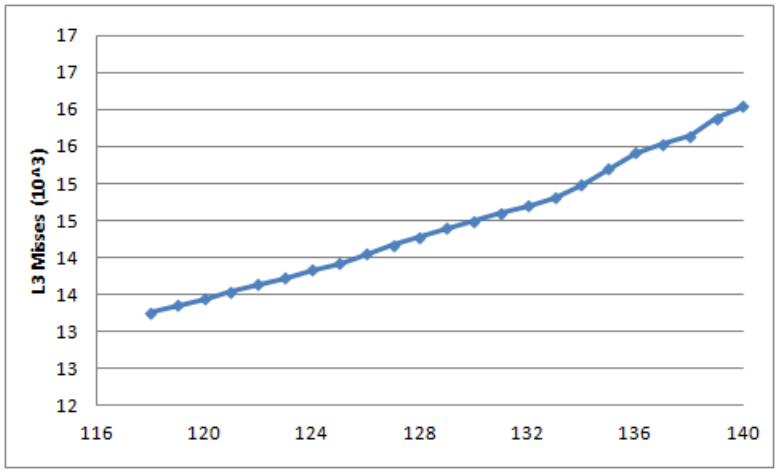

Fig. 9.9 L3 data cache misses in the area around $N = 128$ [50]

9.3.4 Experiment 4 - range around $N = 512$

Experiment 4 covers the area of $N = 512$. Matrix B cannot be stored completely neither in L2 nor L3 cache, and thus drawback is mainly due to their size and associativity. The drawbacks are visible both for execution time and speed in figures 9.14 and 9.15 correspondingly. Local extremes which are more expressive than previous experiments exist also near the main drawback. Another important result is that speed in this region has a negative trend since the transition from L_3 to L_4 region.

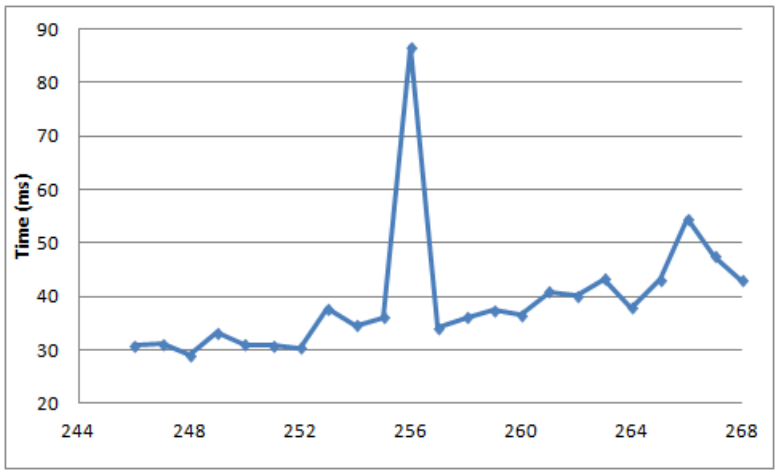

Fig. 9.10 Execution time in the area around $N = 256$ [50]

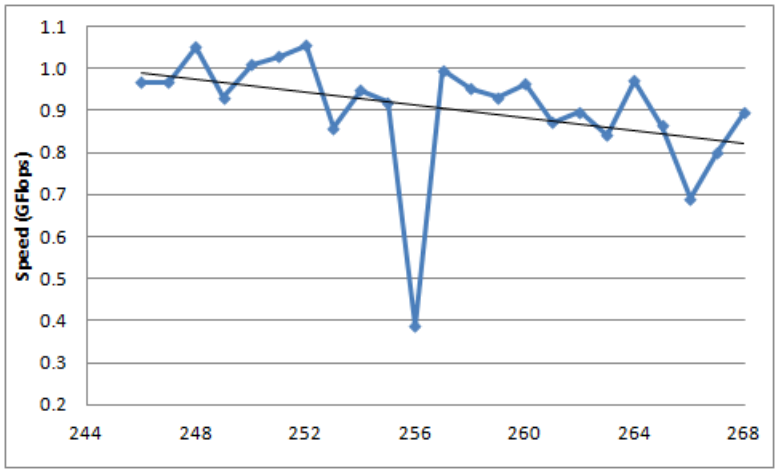

Fig. 9.11 Speed in the area around $N = 256$ [50]

Figures 9.16 and 9.17 depict the L1 and L3 data cache misses for the experiments in the range around $N = 512$. There are expressive L1 and L3 cache associativity drawbacks in $N = 512$. Absolute values for drawbacks are similar, but L3 relative drawback is greater than L1. There are other expressive local extremes only for L1 data cache misses in $N = 503, 505$ and 517.

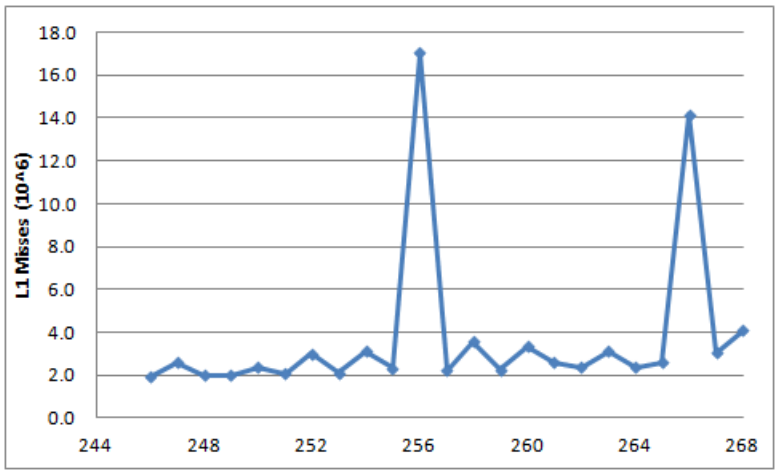

Fig. 9.12 L1 data cache misses in the area around $N = 256$ [50]

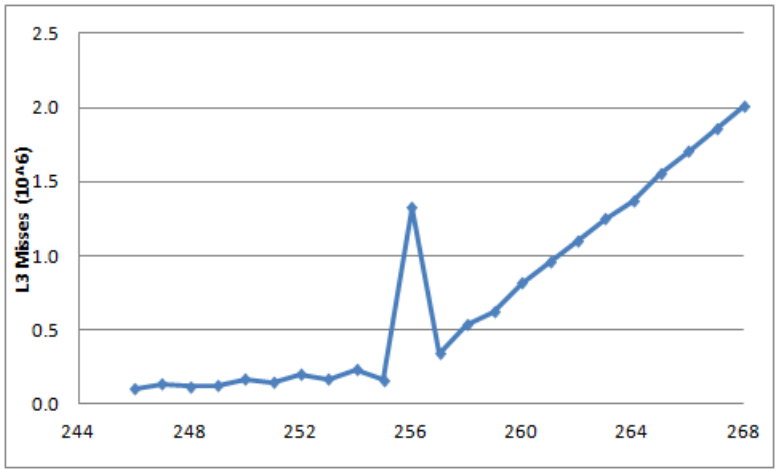

Fig. 9.13 L3 data cache misses in the area around $N = 256$ [50]

9.3.5 Experiment 5 - range around $N = 1024$

Experiment 5 covers the area of $N = 1024$. Matrix B cannot be stored completely in L3 cache and drawbacks appear mostly due to L3 cache size and associativity. The drawbacks are visible both for execution time and speed in figures 9.18 and 9.19 correspondingly. Local extremes are found near the main drawback. Another important result is that speed in this region has a positive trend.

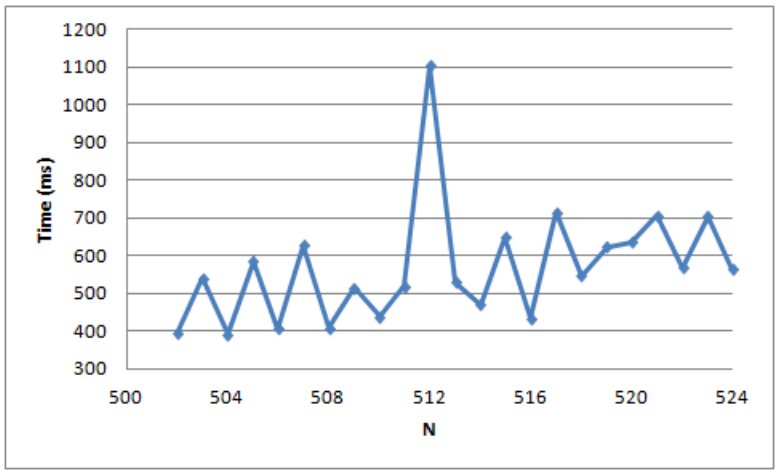

Fig. 9.14 Execution time in the area around $N = 512$ [120]

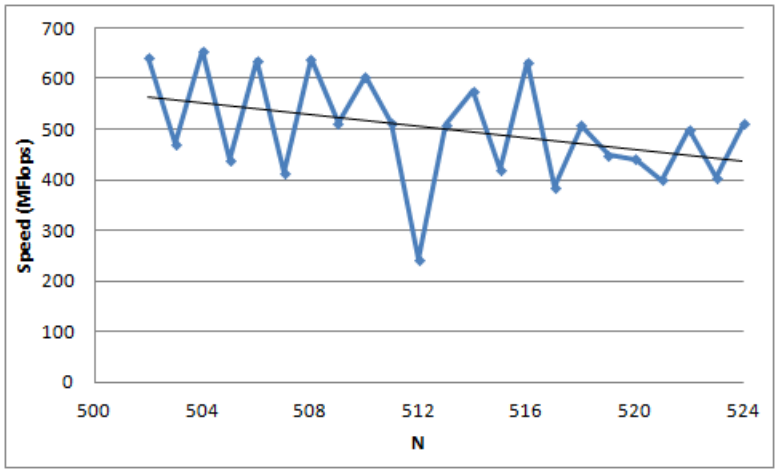

Fig. 9.15 Speed in the area around $N = 512$ [120]

Figures 9.20 and 9.21 depict the L1 and L3 data cache misses for the experiments in the range around $N = 1024$. There are expressive L1 and L3 cache associativity drawbacks in $N = 1024$. Absolute values for drawbacks are similar, but L3 relative drawback is greater than L1. There are other expressive local extremes only for L1 data cache misses in $N = 1019, 1021$ and 1030.

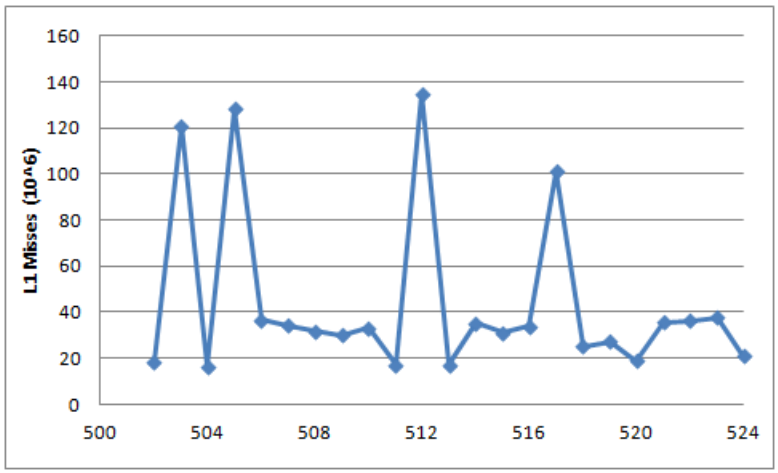

Fig. 9.16 L1 data cache misses in the area around $N = 512$ [50]

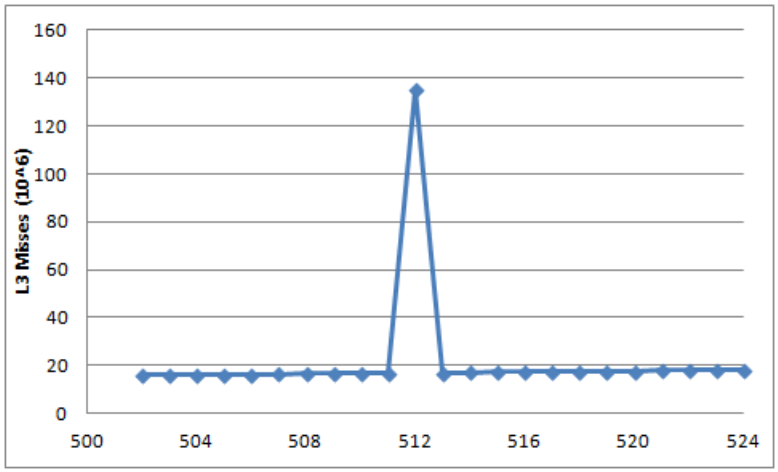

Fig. 9.17 L3 data cache misses in the area around $N = 512$ [50]

9.3.6 Experiment 6 - range around $N = 2048$

Experiment 6 covers the area of $N = 2048$. Matrix B cannot be stored completely in L3 cache. and drawbacks appear mostly due to L3 cache size and associativity. The drawbacks are visible both for execution time and speed in Figure 9.22 and 9.23 correspondingly. Local maximums and minimums are near the main drawback. Another important result is that speed in this region has a positive trend.

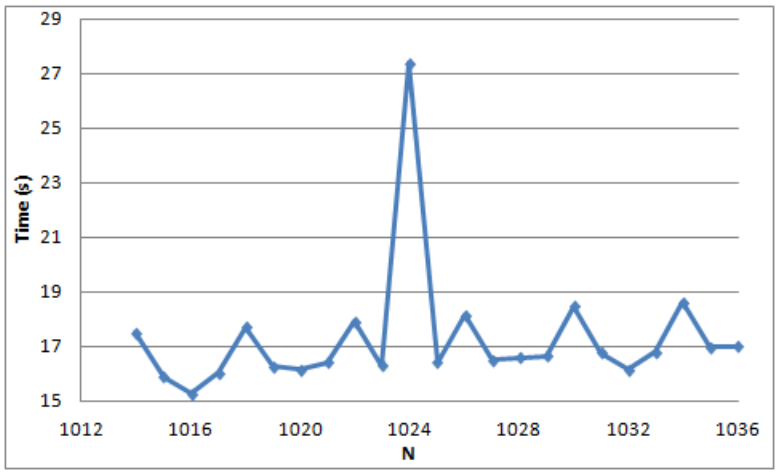

Fig. 9.18 Execution time in the area around $N = 1024$ [120]

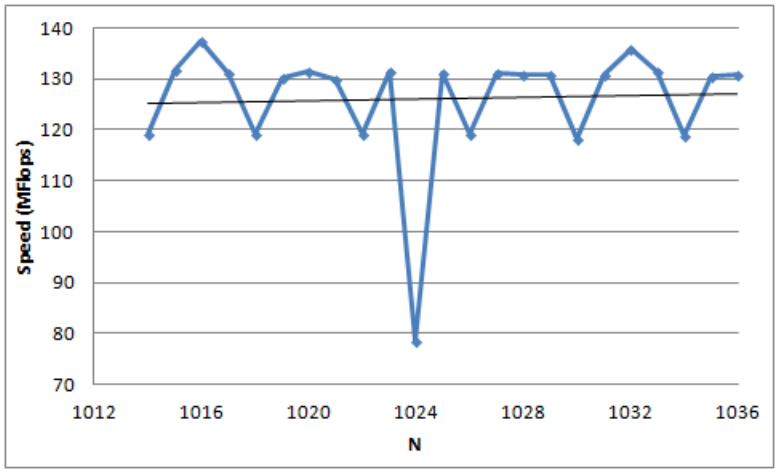

Fig. 9.19 Speed in the area around $N = 1024$ [120]

Figures 9.24 9.25 depict the L1 and L3 data cache misses for the experiments in the range around $N = 2048$. The L1 and L3 data cache misses are similar in point N=2048. There is expressive L3 cache associativity drawback in $N = 2048$. Measured L1 cache misses are even smaller in this point than other points in this area. Local extremes do not appear neither for L1 nor L3 data cache misses.

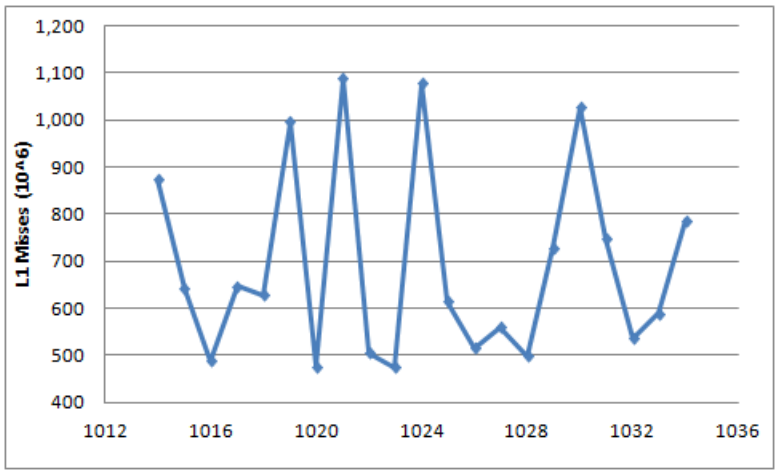

Fig. 9.20 L1 data cache misses in the area around $N = 1024$ [50]

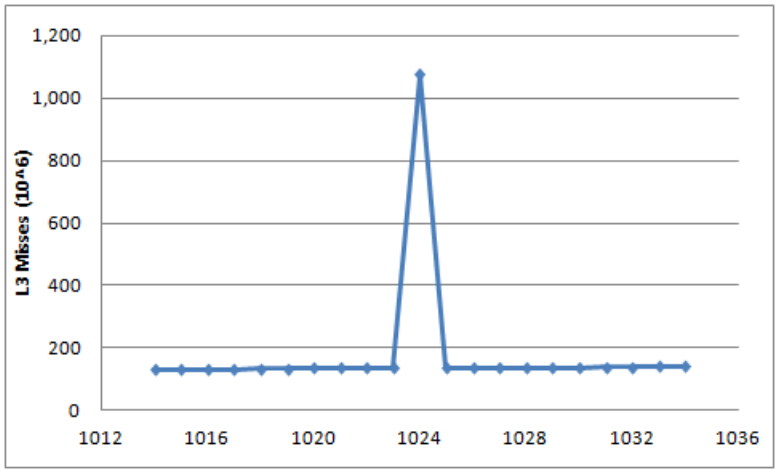

Fig. 9.21 L3 data cache misses in the area around $N = 1024$ [50]

9.4 Experiments for Performance Drawbacks in Parallel Execution

In this section we present the results of series of experiments performed on the same processor as previous Section 9.3 comparing algorithm executions with 1, 2 and 4 cores. The focus will be on the areas around the problem size for the main drawbacks points as depicted in Figure 9.26.

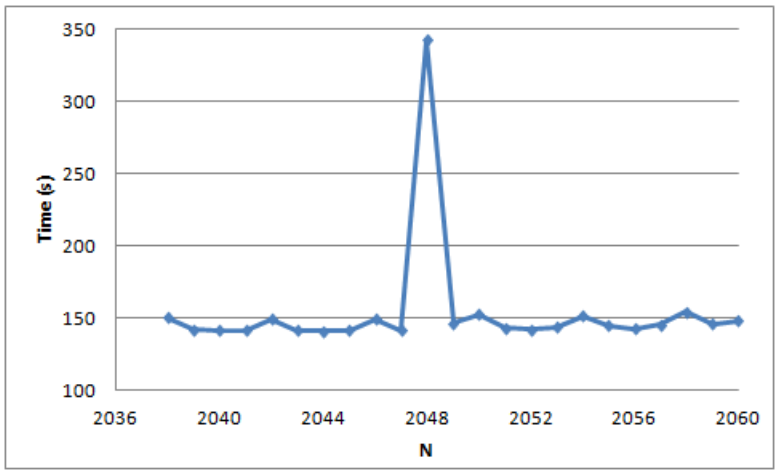

Fig. 9.22 Execution time in the area around $N = 2048$ [120]

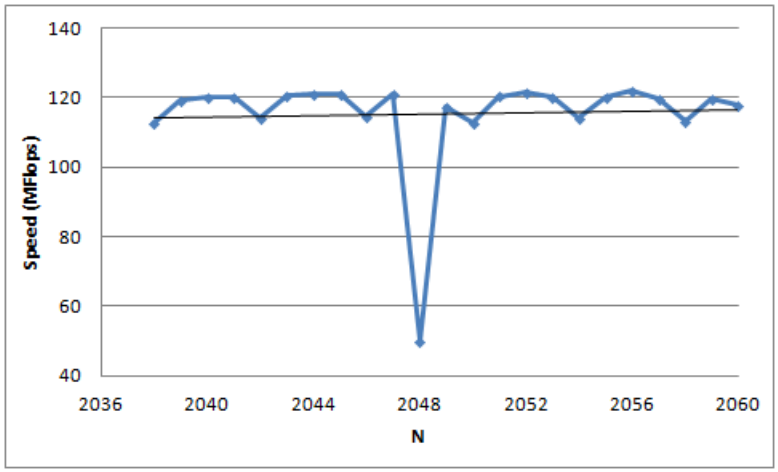

Fig. 9.23 Speed in the area around $N = 2048$ [120]

The experiments are realized for problems that use matrix sizes where theoretical results show expectation of main drawbacks, and also in the area around these points, for example, for the three points -12, -8 and -4 bellow the expected drawbacks and for the three points +4, +8 and +12 above the expected drawback points. Each experiment executes the sequential algorithm on $P = 1$ processor (core) and the parallel algorithm on 2 and 4 processors (cores).

We focus on Speed and Speedup both for sequential and parallel execution. Speeds $V(1)$, $V(2)$ and $V(4)$ are measured for execution on $P = 1,2$ and 4 processor cores correspondingly. Also speedups $S(2)$ and $S(4)$ are measured for each

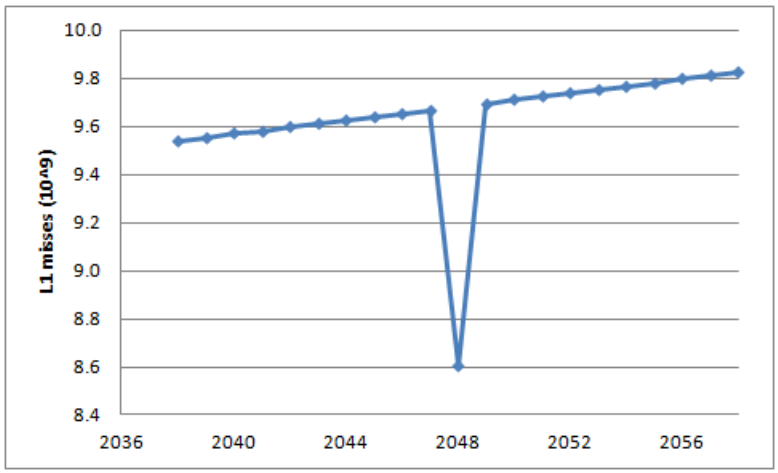

Fig. 9.24 L1 data cache misses in the area around $N = 2048$ [50]

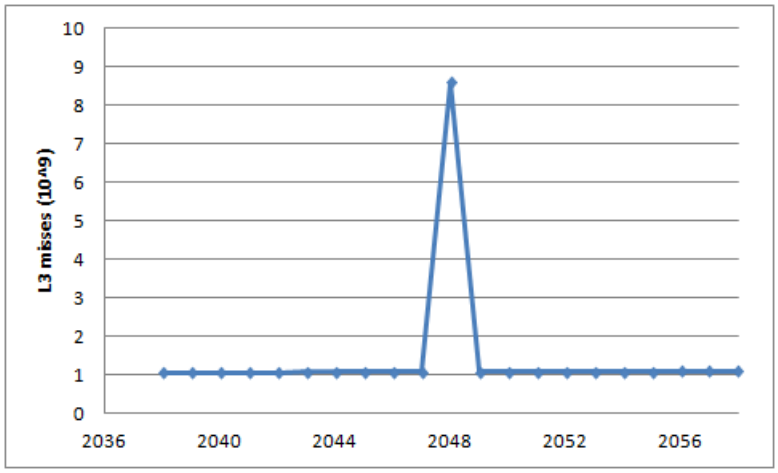

Fig. 9.25 L3 data cache misses in the area around $N = 2048$ [50]

parallel execution with 2 and 4 processor cores correspondingly. Additionally, L1 and L3 (last level) data cache misses are measured with Valgrind [155]. We have not tested the behavior of L2 cache since Valgrind tests measure only first and last level caches.

The X-axis presents the matrix size N in each figure.

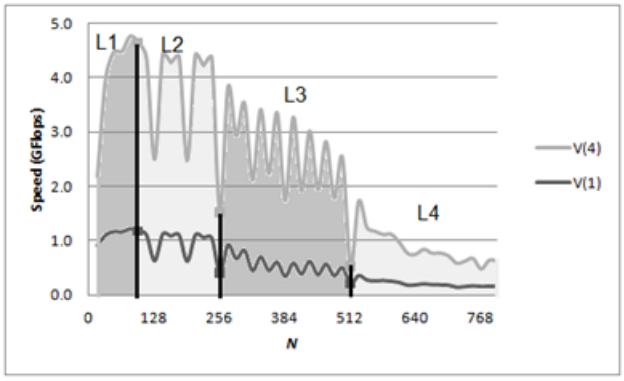

Fig. 9.26 Achieved speed with executions on 1 and 4 cores for matrix multiplication [123]

9.4.1 Experiment 1 - range around $N = 64$

Experiment 1 covers the area of $N = 64$. Matrix B can be stored completely in L1 cache, but the elements of matrix A replace some of the cache blocks from matrix B, thus producing cache misses and performance drawback. Table 9.3 presents the results for execution time (T), Speed (V) in GFlops, cache misses (LM) and Speedup (S).

N	T(1) (ms)	T(2) (ms)	T(4) (ms)	V(1)	V(2)	V(4)	L1M	L3M	S(2)	S(4)
52	0.246909	0.131249	0.0697805	1.1389	2.1426	4.0300	13,390	8,876	1.881	3.538
56	0.304647	0.164517	0.0842474	1.1529	2.1349	4.1691	13,930	9,042	1.852	3.616
60	0.371878	0.193712	0.101199	1.1617	2.2301	4.2688	14,449	9,221	1.920	3.675
64	**0.455059**	**0.238113**	**0.125037**	**1.1521**	**2.2018**	**4.1931**	**21,457**	**9,409**	**1.911**	**3.639**
68	0.531734	0.274807	0.141363	1.1827	2.2884	4.4486	15,808	9,613	1.935	3.761
72	0.629160	0.330113	0.167234	1.1865	2.2613	4.4638	16,729	9,830	1.906	3.762
76	0.735193	0.375798	0.192274	1.1942	2.3362	4.5662	18,026	10,055	1.956	3.824

Table 9.3 Results of the experiments in the area around $N = 64$ [123]

Figures 9.27 and 9.28 depict the speed and speedup for the experiments in the range around $N = 64$. The speed drawback is more expressive for $V(4)$ and $V(2)$ rather than sequential speed in point $N = 64$ as depicted in Figure 9.27. There is very small speedup drawback for $S(2)$ and $S(4)$ where $S(4)$s drawback is more expressive as depicted in Figure 9.28. An important result is that both speed and speedup have a positive trend.

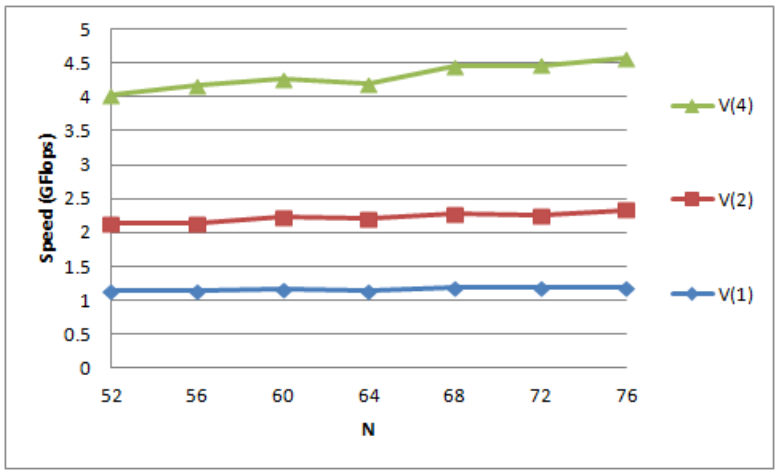

Fig. 9.27 Speeds in the area around $N = 64$ [123]

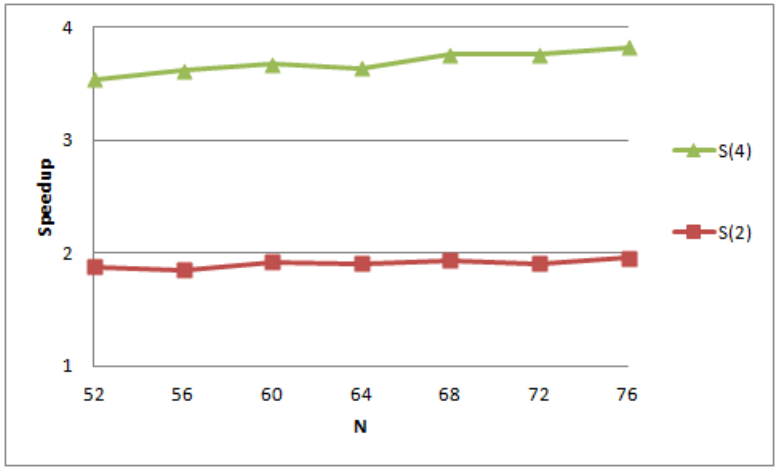

Fig. 9.28 Speedups in the area around $N = 64$ [123]

9.4.2 Experiment 2 - range around $N = 128$

Experiment 2 covers the area of $N = 128$. Matrix B cannot be stored completely in L1 cache and drawbacks appear due to insufficient L1 cache and L1 and L2 cache set associativity. Table 9.4 presents the results for execution time (T), Speed (V) in GFlops, cache misses (LM) and Speedup (S).

Figures 9.29 and 9.30 depict the speed and speedup for the experiments in the range around $N = 128$.

N	T(1) (ms)	T(2) (ms)	T(4) (ms)	V(1)	V(2)	V(4)	L1M	L3M	S(2)	S(4)
116	2.62599	1.33599	0.675012	1.1888	2.3367	4.6248	216,566	13,075	1.966	3.890
120	3.14517	1.59416	0.803838	1.0988	2.1679	4.2994	237,796	13,454	1.973	3.913
124	3.2632	1.64937	0.833448	1.1686	2.3119	4.5753	317,133	13,829	1.978	3.915
128	**6.73523**	**3.37483**	**1.691500**	**0.6227**	**1.2428**	**2.4796**	**2,152,896**	**14,282**	**1.996**	**3.982**
132	3.91838	1.98356	0.996157	1.1739	2.3190	4.6177	386,298	14,705	1.975	3.933
136	4.58864	2.31338	1.167710	1.0964	2.1747	4.3084	339,248	15,419	1.984	3.930
140	4.58651	2.32819	1.172130	1.1966	2.3572	4.6821	388,402	16,043	1.970	3.913

Table 9.4 Results of the experiments in the area around $N = 128$ [123]

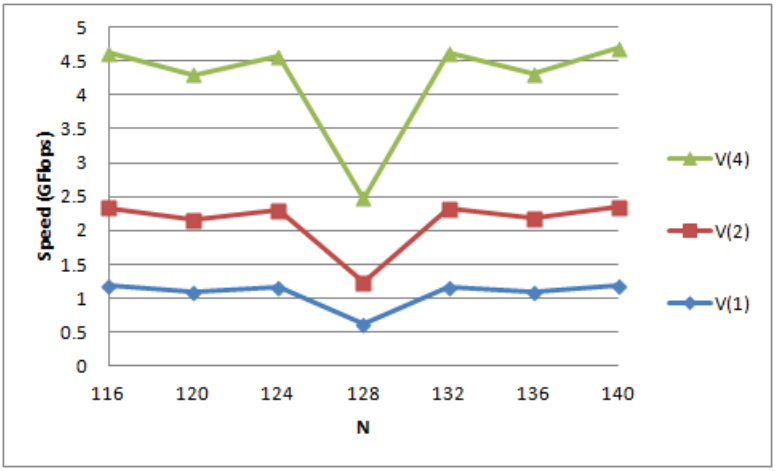

Fig. 9.29 Speeds in the area around $N = 128$ [123]

The speed drawback is expressive for each speed in point $N = 128$ as depicted in Figure 9.29. It is bigger than previous experiments and increases when P grows. Speedup drawback is not found neither for $S(2)$ nor $S(4)$ as depicted in Figure 9.30; the speedup is even better for $N = 128$ rather than other points in the area.

An important result is that all speeds have positive trend. There is a positive Speedup trendline until $N = 128$ and then it becomes negative. Local maximums and minimums are near the main drawback generated due to L1 cache set associativity. The Experiment 1 does not produce local minimums and maximums since the set associative cache works as fully associative cache for small problem size N.

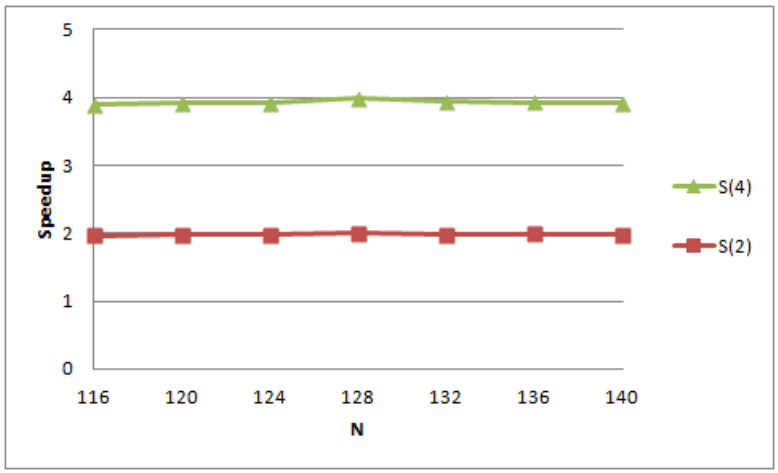

Fig. 9.30 Speedups in the area around $N = 128$ [123]

9.4.3 Experiment 3 - range around $N = 256$

Experiment 3 covers the area of $N = 256$. Matrix B cannot be stored completely in L2 cache and drawbacks are generated due to L1, L2 and L3 cache set associativity. Table 9.5 presents the results for execution time (T), Speed (V) in GFlops, cache misses (LM) and Speedup (S).

N	T(1) (ms)	T(2) (ms)	T(4) (ms)	V(1)	V(2)	V(4)	L1M	L3M	S(2)	S(4)
244	30.8041	15.1709	7.52547	0.9432	1.9151	3.8607	1,909,543	92,319	2.030	4.093
248	33.6237	17.2433	8.21905	0.9073	1.7691	3.7116	1,963,703	117,054	1.950	4.091
252	34.1971	17.2361	8.68687	0.9359	1.8569	3.6844	2,987,831	204,269	1.984	3.937
256	**104.304**	**57.2151**	**27.0405**	**0.3217**	**0.5865**	**1.2409**	**17,021,258**	**1,326,929**	**1.823**	**3.857**
260	38.8296	18.8008	9.64274	0.9053	1.8697	3.6454	3,335,444	821,196	2.065	4.027
264	41.788	20.0876	9.63306	0.8806	1.8320	3.8201	2,364,154	1,367,713	2.080	4.338
268	44.8336	21.5451	11.0572	0.8587	1.7868	3.4817	4,088,886	2,004,349	2.081	4.055

Table 9.5 Results of the experiments in the area around $N = 256$ [123]

Figures 9.31 and 9.32 depict the speed and speedup for the experiments in the range around $N = 256$.

The speed drawback is expressive for each speed in point $N = 256$ as depicted in Figure 9.31. It is bigger than previous experiments and increases when P grows. There are speedup drawbacks both for $S(2)$ and $S(4)$ where $S(4)$s drawback is more expressive as depicted in Figure 9.32. Superlinear speedup is found both for $P = 2$ and $P = 4$ processors since this area is in the superlinear region. This phenomenon

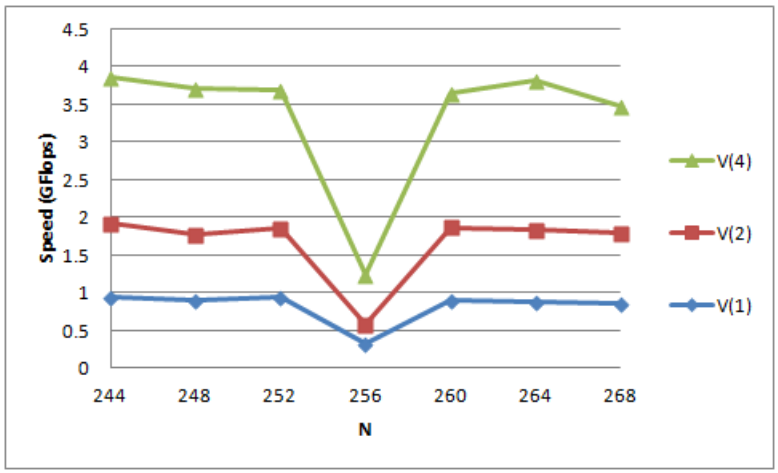

Fig. 9.31 Speeds in the area around $N = 256$ [123]

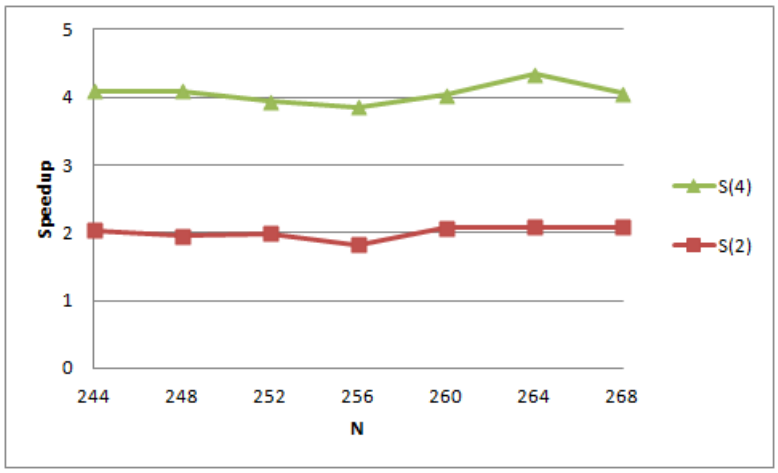

Fig. 9.32 Speedups in the area around $N = 256$ [123]

will be elaborated both theoretically and experimentally in the next Part III. Another important result is that all speeds have negative trend since the transition from L2 to L3 region.

9.4.4 Experiment 4 - range around $N = 512$

Experiment 4 covers the area of $N = 512$. Matrix B cannot be stored completely neither in L2 nor L3 cache, and thus drawback is mainly due to their size and associativity. Table 9.6 presents the results for execution time (T) in miliseconds, Speed (V) in GFlops, cache misses (LM) and Speedup (S).

N	T(1)	T(2)	T(4)	V(1)	V(2)	V(4)	L1M	L3M	S(2)	S(4)
500	465.907	257.429	116.141	0.5366	0.9711	2.1526	31,327,035	15,792,325	1.810	4.012
504	410.632	206.221	96.4347	0.6235	1.2416	2.6551	16,460,993	16,177,786	1.991	4.258
508	481.68	237.265	118.486	0.5443	1.1051	2.2129	31,937,134	16,559,469	2.030	4.065
512	**1448.94**	**893.006**	**533.299**	**0.1853**	**0.3006**	**0.5033**	**135,154,085**	**135,137,152**	**1.623**	**2.717**
516	599.456	277.324	138.58	0.4584	0.9908	1.9828	33,889,015	17,351,077	2.162	4.326
520	662.439	295.451	132.531	0.4245	0.9518	2.1219	18,887,514	17,763,756	2.242	4.998
524	643.795	322.384	163.302	0.4470	0.8926	1.7621	20,865,023	18,171,270	1.997	3.942

Table 9.6 Results of the experiments in the area around $N = 512$ [123]

Figures 9.33 and 9.34 depict the speed and speedup for the experiments in the range around $N = 512$.

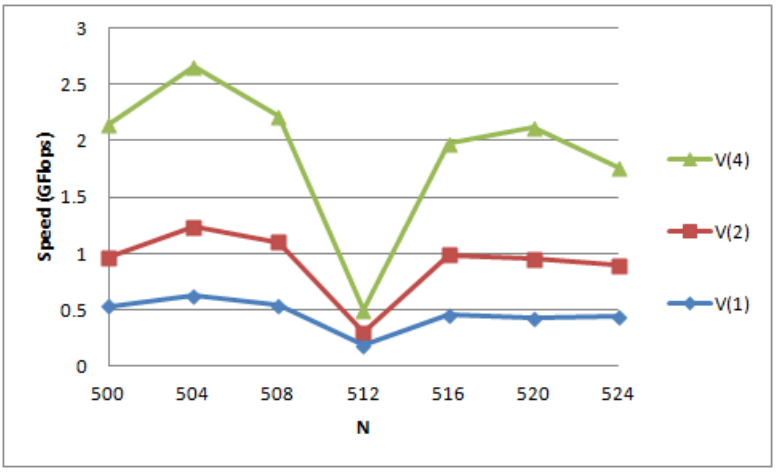

Fig. 9.33 Speeds in the area around $N = 512$ [123]

The speed drawback is expressive for each speed in point $N = 512$ as depicted in Figure 9.33. It is bigger than previous experiments and increases when P grows. The drawbacks for parallel execution are so expressive that $V(2)$ and $V(4)$ are similar as other points in this area for sequential algorithm. The speedup drawbacks are found

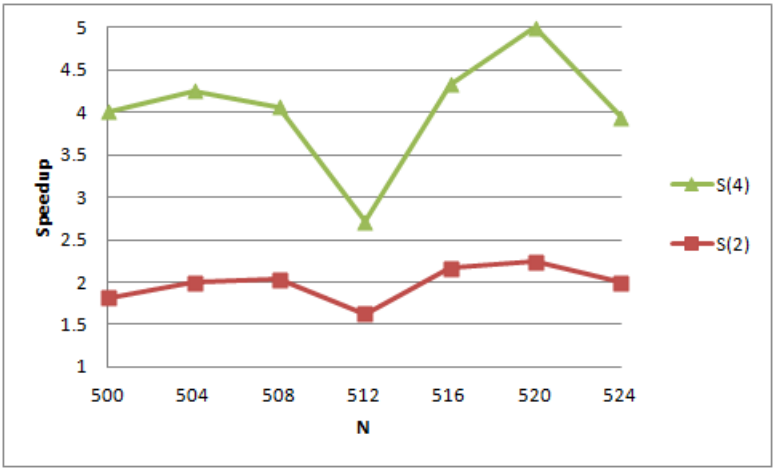

Fig. 9.34 Speedups in the area around $N = 512$ [123]

both for $S(2)$ and $S(4)$ where $S(4)$s drawback is more expressive as depicted in Figure 9.34. Superlinear speedup is also found in some points, especially expressive in point $N = 520$. Another important result is that all speeds have negative trend since the transition from L_3 to L_4 region.

9.4.5 Experiment 5 - range around $N = 1024$

Experiment 5 covers the area of $N = 1024$. Matrix B cannot be stored completely in L3 cache and drawbacks appear due to L3 cache size and associativity. Table 9.3 presents the results for execution time (T) in seconds, Speed (V) in MFlops, cache misses (LM) and Speedup (S).

N	T(1)	T(2)	T(4)	V(1)	V(2)	V(4)	L1M	L3M	S(2)	S(4)
1012	15.695	7.628	3.720	132.071	271.757	557.281	560,845,297	130,207,941	2.058	4.220
1016	15.275	7.572	3.844	137.321	277.002	545.700	489,187,519	131,755,327	2.017	3.974
1020	16.144	7.846	3.831	131.465	270.504	554.007	476,074,648	133,314,925	2.058	4.214
1024	**25.26**	**15.33**	**10.51**	**85.005**	**140.129**	**204.394**	**1,077,440,805**	**1,077,424,512**	**1.648**	**2.404**
1028	16.577	8.048	3.937	131.071	269.980	551.820	499,081,123	136,470,949	2.060	4.210
1032	16.155	8.005	4.040	136.072	274.616	544.154	537,036,472	138,067,471	2.018	3.999
1036	16.980	8.261	4.041	130.968	269.199	550.282	568,057,068	139,676,397	2.055	4.202

Table 9.7 Results of the experiments in the area around $N = 1024$ [123]

Figures 9.35 and 9.36 depict the speed and speedup for the experiments in the range around $N = 1024$.

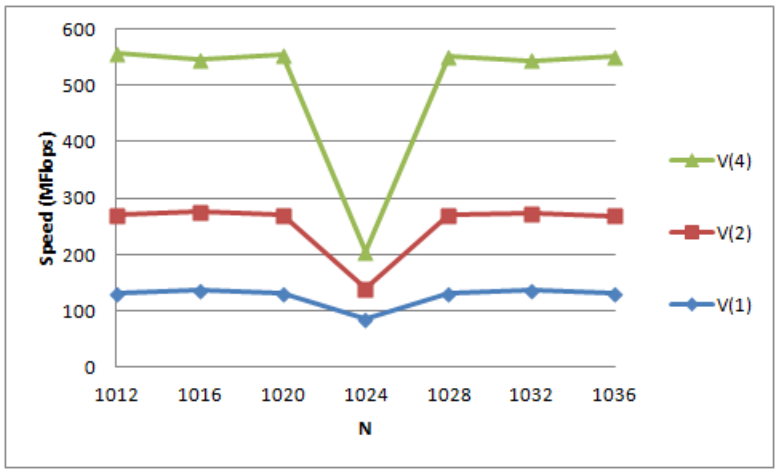

Fig. 9.35 Speeds in the area around $N = 1024$ [123]

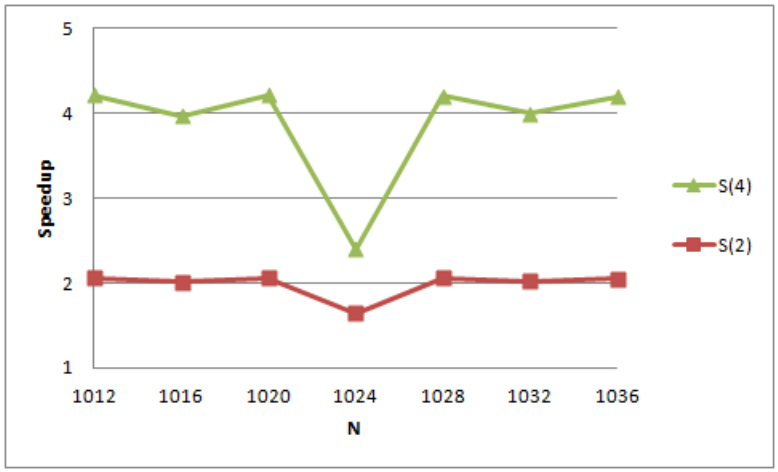

Fig. 9.36 Speedups in the area around $N = 1024$ [123]

The speed drawback is expressive for each speed in point $N = 1024$ as depicted in Figure 9.35. It increases when P grows. The speed drawbacks for parallel execution are so expressive that $V(4)$ is similar as other points in this area for parallel execution

with $P = 2$ processors and $V(2)$ is similar as other points in this area for sequential execution. The speedup drawbacks are found both for $S(2)$ and $S(4)$ where $S(4)$s drawback is so expressive that $S(4)$ is similar to other points in this area for parallel execution with $P = 2$ processors as depicted in Figure 9.36. Superlinear speedup is also found in local maximums.

9.4.6 Experiment 6 - range around $N = 2048$

Experiment 6 covers the area of $N = 2048$. Matrix B cannot be stored completely in L3 cache. Table 9.8 presents the results for execution time (T) in seconds, Speed (V) in MFlops, cache misses (LM) and Speedup (S).

N	T(1)	T(2)	T(4)	V(1)	V(2)	V(4)	L1M	L3M	S(2)	S(4)
2036	141.68	69.71	34.804	119.142	242.138	484.993	9,512,432,234	1,057,585,355	2.032	4.071
2040	141.52	70.54	36.291	119.976	240.702	467.870	9,568,573,042	1,063,825,729	2.006	3.900
2044	141.12	69.45	34.820	121.024	245.921	490.508	9,624,934,306	1,070,090,603	2.032	4.053
2048	**349.17**	**177.2**	**111.21**	**49.201**	**96.960**	**154.486**	**8,600,990,501**	**8,600,974,720**	**1.97**	**3.140**
2052	145.15	70.71	35.106	119.051	244.386	492.242	9,738,321,967	1,082,696,101	2.053	4.135
2056	143.31	71.11	36.602	121.290	244.429	474.890	9,795,347,195	1,089,034,767	2.015	3.915
2060	148.69	73.90	36.616	117.583	236.600	477.484	9,852,592,562	1,095,396,059	2.012	4.061

Table 9.8 Results of the experiments in the area around $N = 2048$ [123]

Figures 9.37 and 9.38 depict the speed and speedup for the experiments in the range around $N = 2048$.

The speed drawback is expressive for each speed in point $N = 2048$ as depicted in Figure 9.37. It increases when P grows. The speed drawbacks for parallel execution are so expressive that $V(2)$ and $V(4)$ are similar as other points in this area for sequential algorithm.

The speedup drawbacks are found both for $S(2)$ and $S(4)$ where $S(2)$s drawback is minimal while $S(4)$s drawback is more expressive as depicted in Figure 9.38. Small superlinear speedup is also found in local maximums.

9.5 Summary

Cache hierarchy and organization can seriously affect performance. This chapter proves that the cache associativity is another important performance parameter in addition to the cache size.

We have provided theoretical analysis (published in [120] for the purpose of this thesis research) why there are performance drawbacks and suggest to organize dense

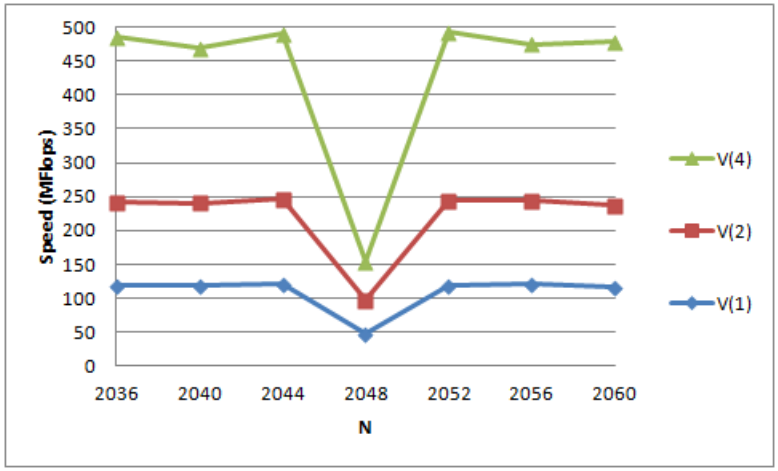

Fig. 9.37 Speeds in the area around $N = 2048$ [123]

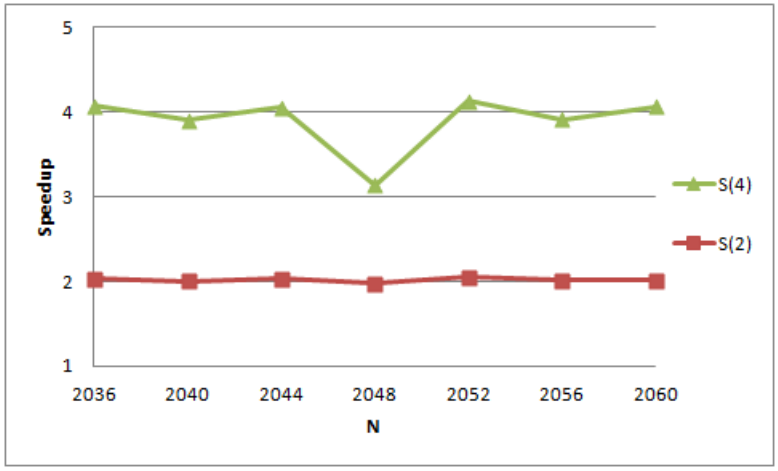

Fig. 9.38 Speedups in the area around $N = 2048$ [123]

matrix multiplication algorithms avoiding situations where mapping onto n-way set associative cache will use only a small part of the cache instead of whole cache capacity.

9.5.1 Summary for Drawbacks in Sequential Exectuion

This chapter presents the results of performed experimental research published in [50] that approve the theoretical results from Chapter 9 showing real cases of performance drawbacks. In addition to our theoretical performance analysis we have also shown trends in performance behavior.

First and last level cache misses are also profiled. The overall dense matrix multiplication algorithm performance does not change despite the local L1 cache misses maximums. The performance drawbacks appear mostly due to last level cache associativity and capacity.

L1 cache associativity drawbacks appear for $N = 64, 128, 256, 512$ and 1024 although the drawback is small for $N = 64$. L3 cache associativity drawbacks appear for $N = 256, 512, 1024$ and 2048. For $N = 512, 1024$ and 2048 the L1 and L3 cache associativity drawbacks are similar in absolute value, but L3 cache associativity drawbacks are more expressive relative, i.e. L1 drawback is due to capacity problem rather than associativity problem and each L1 cache miss generates L3 cache miss.

The speed drawback is more expressive for greater matrix size than smaller in the critical points. Inconsiderable speed drawback is found for matrix size $N = 64$. Significant speed drawback is found for matrix sizes $N = 128, 256, 512, 1024$ and 2048.

9.5.2 Summary for Drawbacks in Parallel Exectuion

Six experiments are realized for different cache regions by the authors in [123] for the purpose of this thesis research. Execution time, speed, cache misses and speedup are measured for each experiment around the critical points found as theoretical results for matrix sizes $N = 64, 128, 256, 512, 1024$ and 2048.

L1 cache associativity drawbacks appear for $N = 64, 128, 256, 512$ and 1024 although the drawback is small for $N = 64$. L3 cache associativity drawbacks appear for $N = 256, 512, 1024$ and 2048. For $N = 512, 1024$ and 2048 the L1 and L3 cache associativity drawbacks are similar in absolute value, but L3 cache associativity drawbacks are more expressive relative, i.e. L1 drawback is due to capacity problem rather than associativity problem and each L1 cache miss generates L3 cache miss.

The speed drawback is more expressive for greater matrix size than smaller in the critical points. Inconsiderable speed drawback is found for matrix size $N = 64$ both for sequential and parallel execution. Significant speed drawback is found for matrix sizes $N = 128, 256, 512, 1024$ and 2048.

Parallel algorithm executions result with speed drawbacks more than sequential. They are expressive for the experiments executing on $P = 4$ processors where the speed $V(4)$ is smaller than the speed $V(2)$. For $N = 512$ speed $V(4)$ is similar as speed $V(1)$ in the observed points of the particular area. The speed drawbacks for

the experiments executing on $P = 2$ processors are also expressive. Speed $V(2)$ is similar or even smaller than speed $V(1)$ in other points of the particular area.

Inconsiderable speedup drawback is found for matrix size $N = 64$ both for parallel execution on 2 and 4 cores. For $N = 128$ we found even greater speedup than the observed points in the area for both speedups. For each other $N = 256, 512, 1024$ and 2048 significant speedup drawback is found, especially for speedup $S(4)$. The only exception is the inconsiderable speedup drawback for parallel execution on $P = 2$ processors for matrix size $N = 2048$. Parallel speed and speedup drawbacks are inconsiderable for matrix sizes $N = 64$ and $N = 128$ since these are the points in L2 region dedicated per core. The quoted theorems are valid for parallel execution only in the dedicated cache regions. Significant speed and speedup drawbacks is found in L3 and L4 regions, especially for parallel execution on $P = 4$ cores. Further research will include performance evaluation of parallel execution including the number of processors P for shared cache.

Based on theoretical analysis and experimental research we have concluded how n-way associative cache can seriously affect performance. We have analyzed and found theoretically the points where the associativity causes performance drawbacks and suggest organization of the matrix multiplication algorithm avoiding situations where mapping onto n-way set associative cache will use only a small part of the cache instead of whole cache capacity. The performed experimental research approved the results showing real cases of performance drawbacks in both sequential and parallel executions.

Part III
Achieving Superlinear Speedup

Chapter 10
Superlinear Speedup in Matrix Multiplication on Multiprocessor

Abstract Despite the Gustafson's Law that maximum scaled speedup is p and Shi's limitation that superlinear speedup is only possible for *Non Structure Persistent* algorithms, superlinear region is found and analyzed for parallel execution. This chapter presents the analysis realized for the purpose of this thesis research by the authors in [52]. The authors show theoretically that there is a region where the superlinear speedup can be achieved. Theoretical proof of existence of a superlinear speedup is given and boundaries of the region where it can be achieved are determined. The experiments confirm our theoretical results. The realization of modern processors is based on an multicore architecture with increasing number of cores per processor which is actually organized as a shared memory multiprocessor with shared L2 cache and distributed L1 cache. Therefore these results will have impact on future software development and exploitation of parallel hardware.

10.1 Sequential vs Parallel Cache Occupancy

Let's analyze the cache occupancy by the given sequential and parallel implementations. A part of the cache is occupied by the OS for its requirements, usually a small portion. In sequential execution the cache will not generate cache misses if all the matrices A and B are both stored in the cache as presented in Figure 10.1 a).

Note that no space is required for matrix C, since the values are computed and stored with write no allocation algorithm directly in main memory. If write allocation is used then a small space will be used in the cache but its dimension is small in comparison to the need of storage of whole matrices A and B.

Suppose that matrices are stored in L1 cache. If matrix dimension increases then there is a need for more space and cache misses generation starts provoking performance degradation. The analysis continues with storage problems in the next level of the caches L2, L3 and so on. The same situation with generation of cache misses happens when L2 (L3) cache will be occupied by both matrices. The performance degradation is presented in Fig. 10.2. Note that this degradation does not happen

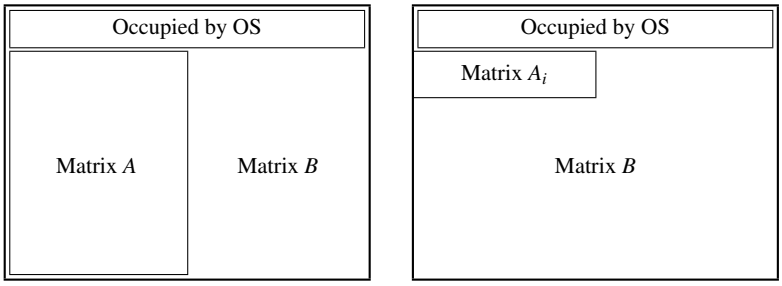

Fig. 10.1 Cache occupancy in sequential and parallel execution [52]

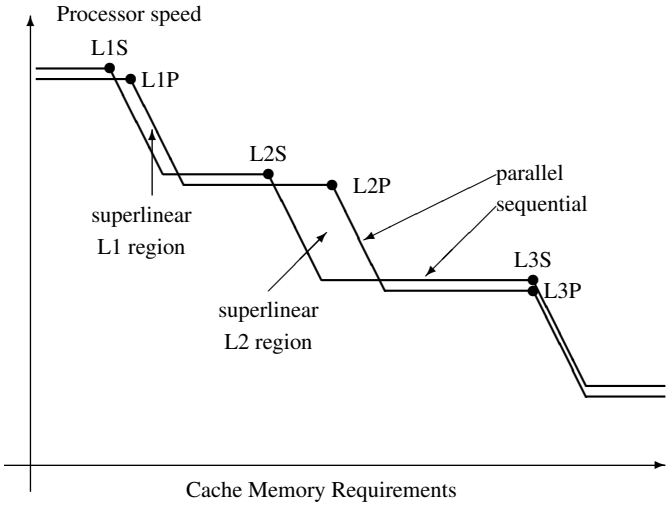

Fig. 10.2 Expected average processor speed with real cache [52]

ideally with horizontal and vertical lines, but in reality it should be a smooth curve instead of straight line, but following the presented pattern. [121] shows theoretical foundation about real processor speed with caches and present simulation and experimental diagrams of these curves.

The parallel algorithm (4.4) organizes computations by distribution to several processors and storage requirements to their distributed caches. For each processor there is a need only for a chunk A_i of matrix A to be stored in cache and whole matrix B. Therefore there is more space for matrix B allowing greater size problems to be stored without generation of cache misses, as shown in Figure 10.1 b).

The realization of this idea produces less cache misses for the same problem size in parallel execution than in sequential. This will make increased processor performance in comparison with sequential processing as presented in Fig. 10.2.

Let's define the points L1S, L2S and L3S for sequential and the points L1P, L2P and L3P for parallel execution correspondingly to mark the regions where the processor will perform the highest speed when working with L1, L2 or L3 cache [124]. Note that differences are mainly shown between sequential and parallel processing due to distributed L1 and L2 caches and not for L3 cache which is mainly shared for multicore implementations.

In these two regions there is higher average processor performance for parallel execution and these regions are correspondingly marked as L1 and L2 **superlinear regions**. L1 cache is small according to L2 and therefore we will concentrate to analysis of the L2 superlinear region.

Further on we make theoretical analysis and determine the superlinear region, i.e. the region where superlinear speedup is achieved.

10.2 Speedup Analysis with Memory Behavior

In this section we analyze the existing equations to calculate sequential and parallel execution time, and thus the speedup.

For better presentation, we'll use the abbreviations shown in Table 10.1 for variables defined in [62].

Abbreviation	Variable
T_S	$CPU\ Execution\ Time_{Sequential}$
t_p	$CPU\ Execution\ Time_{Parallel}$
CC_S	$CPU\ Clock\ Cycles_{Sequential}$
CC_P	$CPU\ Clock\ Cycles_{Parallel}$
CT	$Clock\ Cycle\ Time$
MC_S	$Memory\ Clock\ Cycles_{Sequential}$
MC_P	$Memory\ Clock\ Cycles_{Parallel}$
ME	$SizeOf(MatrixElement)$
M_S	$Memory_{Sequential}$
M_P	$Memory_{Parallel}$
CS_2	$L2\ Cache\ size\ of\ one\ core$
CS_3	$L3\ Cache\ size\ of\ one\ core$
OS_{S2}	$Size\ of\ L2\ Cache\ occupied\ by\ OS\ in\ Sequential$
OS_{P2}	$Size\ of\ L2\ Cache\ per\ core\ occupied\ by\ OS\ in\ Parallel$
OS_{S3}	$Size\ of\ L3\ Cache\ occupied\ by\ OS\ in\ Sequential$
OS_{P3}	$Size\ of\ L3\ Cache\ per\ core\ occupied\ by\ OS\ in\ Parallel$

Table 10.1 Variable abbreviations for better presentation [52]

Gustafson's law [55] assumes that parallel execution time is fixed and defines that maximum obtained speedup should be measured by scaling the problem to the

number of processors, i.e. it is **linear** equal to the number of processors P. Several other assumptions are made to prove this, including:

- The number of parallel chunks of the application is multiple of the number of processors.
- The performance of the parallel chunks never saturates.
- Creation / deletion of the parallel threads does not overhead.
- Sequential and parallel parts of the application run at the same rate.

Based on these assumptions the performance analysis should refer to both memory access and CPU execution times. The CPU execution times for sequential algorithm t_s and parallel algorithm t_p are respectively defined in (10.1) and (10.2).

$$t_s = CC_S \cdot CT \tag{10.1}$$

$$t_p = CC_P \cdot CT \tag{10.2}$$

In Theorem 10.1 we express speedup relation in a new way as a function of both the CPU and memory clock cycles.

Theorem 10.1. *The speedup for parallel execution in a multiprocessor over sequential execution is equal to*

$$Speedup = \frac{CC_S + MC_S}{CC_P + MC_P} \tag{10.3}$$

Proof. Relations (10.1) and (10.2) do not include memory access which is very important in the analysis since it can dominate the performance over the CPU execution time even in cases with intensive computation. Real CPU execution time, both for sequential and parallel, is defined in (10.4) and (10.5) according to [62].

$$t_s = (CC_S + MC_S) \cdot CT \tag{10.4}$$

$$t_p = (CC_P + MC_P) \cdot CT \tag{10.5}$$

Equation (10.3) is obtained from (2.9) by dividing (10.4) and (10.5).

Theorem 10.1 and the equation (10.3) is basics for our theoretical analysis of existence of superlinear speedup.

10.3 How to Obtain Super-Linear Speedup

In this section we introduce a new methodology how to obtain a superlinear speedup region for matrix multiplication. The idea is to determine when and how the superlinear speedup is possible and what is the matrix size and algorithm organization to obtain superlinear speedup for parallelized matrix multiplication on P processors.

10.3.1 Existence of Superlinear Speedup

Let's analyze the memory part of (10.3) in Theorem 10.1. The main idea is presented next. Is it possible that memory access time in parallel system will be reduced more than P times in comparison to memory access time when execution is sequential? It will lead to superlinear speedup. This idea is mathematically specified in Theorem 10.2.

Before we construct the proof of superlinear speedup existence, we introduce several new definitions and properties and show how they affect the performance.

Lemma 10.1. *For a given multiprocessor and its sequential and parallel executions there exists a real number* ε *such that* $0 \leq \varepsilon \leq P$ *and*

$$CC_S = CC_P \cdot (P - \varepsilon). \tag{10.6}$$

Proof. As presented in [55] the corresponding maximum speedup for P processors is given in (10.7).

$$MaxSpeedup = \frac{t_s}{t_p} \leq P \tag{10.7}$$

Further on we obtain equation (10.8) by dividing (10.1) and (10.2), and using (10.7).

$$CC_S \leq CC_P \cdot P \tag{10.8}$$

The proof can be constructed if (10.8) is rewritten in (10.6).

Lemma 10.1 is used for the next theorem.

Theorem 10.2. *Superlinear speedup in a shared memory multiprocessor with distributed cache memory per processor is possible if and only if*

$$MC_S > P \cdot MC_P + \varepsilon \cdot CC_P \tag{10.9}$$

Proof. Let's assume that (10.9) is true. Using (10.6) and (10.9), (10.3) becomes (10.10), i.e. maximum speedup will be

$$MaxSpeedup > \frac{(P - \varepsilon) \cdot CC_P + P \cdot MC_P + \varepsilon \cdot CC_P}{CC_P + MC_P} = P \tag{10.10}$$

The inverse proof is analogue.

Our main idea to achieve superlinear speedup in a shared memory multiprocessor with distributed cache specified by Theorem 10.2 will be elaborated in the next section. We aim to verify existence of (10.9) and to find if it has a solution. If there is a solution, the goal is to determine what is the value of N?

Different speeds of the processor-memory hierarchy are used for the purpose of further analysis where the processor is the fastest component and the main memory is the slowest component. (10.11) presents the relation among number of clock cycles for operation in processor, and memory cycles for L1, L2 and L3 caches and main memory.

$$CC_{CPU} < MC_{L1} < MC_{L2} < MC_{L3} < MC_{Memory} \tag{10.11}$$

For example, usual behavior is that memory clock cycles for L2 cache is at least 100 times greater than CPU clock cycles. Therefore, the focus of the analysis is not only to reduce the CPU clock cycles for parallel execution to obtain speedup, but to analyze the correlation between matrix size and required cache memory space per processor and to reduce the overall memory clock cycles as much as possible in parallel.

10.3.2 Memory and Cache Requirements

Required memory space for the three matrices A, B and C is $3 \cdot N^2$ in shared memory. But, more important is to calculate the cache space required for reading the matrices A and B from the shared memory, since matrix C usually is not stored in cache with no write allocation algorithm and only a small part will be stored if write allocation algorithm is used for the cache.

Lemma 10.2. *Cache memory storage requirements for sequential execution of the matrix multiplication algorithm (4.4) on a uniprocessor with cache is*

$$M_S = 2 \cdot N^2 \cdot ME \tag{10.12}$$

Proof. Algorithm (4.4) is equivalent to (4.1) for sequential uniprocessor implementation as a compute intensive algorithm with $O(N^3)$ computations. In both algorithms there is no need for cache memory storage requirements for 3 matrices like in the main memory, since the product matrix C will be only stored after all necessary computations will be performed in a local register and afterwards stored in the memory. Therefore the only requirement for sequential execution on a single processor is cache memory space for matrices A and B, each with $N \cdot N$ elements.

The cache memory space for parallel execution is smaller than in Lemma 10.2 and is analyzed next.

Lemma 10.3. *Cache memory storage requirements for parallel execution of matrix multiplication algorithm (4.4) on a shared memory multiprocessor with dedicated cache per processor is*

$$M_P = N^2 \cdot (1 + \frac{1}{P}) \cdot ME \tag{10.13}$$

Proof. Each processor has its own CS_2 cache. On P processors, such as $N = q \cdot P$ and according to the algorithm (4.4), the required cache memory space per core is partition A_i with $N \cdot N/P$ and also the whole matrix $B_{N \cdot N}$.

10.3.3 A Superlinear Region for Matrix Multiplication

Lemmas 10.2 and 10.3 and relations (10.12) and (10.13) define the dependencies of matrix size and cache memory storage requirements. Nowadays, processors possess L1 cache size in Kilobytes, and L2 and L3 in Megabytes. Using double precision with 8 bytes per matrix elements ($ME = 8B$), and matrix size $N > 10$, the L1 cache cannot store the whole matrix. Therefore, both the sequential and the parallel executions will need L2 cache, and our analysis will be focused there. Also, most of today's CPUs possess dedicated L2 cache per core. Thus, using P cores for parallel execution assumes usage of P times L2 cache.

Our goal is to find if there is a matrix size range, and particularly N_{min} and N_{max} sizes so that the whole M_P memory can be stored in P caches, and in the same time M_S cannot be stored in 1 core cache? If this is possible, then $MC_P << MC_S$ and (10.9) will be satisfied, and thus, maximum speedup will be superlinear as shown in Theorem 10.2.

Theorem 10.3. *The range of matrix size N for which implementation of algorithm (4.4) in a shared memory multiprocessor achieves superlinear speedup, where each processor has distributed L2 cache with size of CS_2 is given by calculating the matrix size values of N_{min} and N_{max}*

$$N \geq \sqrt{\frac{CS_2 - OS_{S2}}{2 \cdot ME}} = N_{min} \tag{10.14}$$

$$N \leq \sqrt{\frac{CS_2 - OS_{P2}}{(1 + \frac{1}{P}) \cdot ME}} = N_{max} \tag{10.15}$$

Proof. Proof can be easily constructed from the equations (10.12) and (10.13) by implementing the idea to find a range where cache memory storage requirements will generate cache misses for sequential execution and not for parallel execution - meaning that the cache space is small to store complete matrix B and complete matrix A, but it can store complete matrix B and a chunk of matrix A.

The next theorem defines existence of a superlinear speedup.

Theorem 10.4. *A superlinear speedup exists in a shared memory multiprocessor with dedicated L2 cache per processor for execution of algorithm (4.4).*

Proof. To prove this we use the results of Theorem 10.3. The condition $N_{max} \geq N_{min}$ is always valid, showing the existence of N if $N_{max} > N_{min}$. The processor uses small part of its L2 cache for instructions, i.e. $OS_{S2} << CS_2$ and $OS_{P2} << CS_2$. Due to the fact that $OS_{P2} \approx OS_{S2}$ one can conclude that $CS_2 - OS_{P2} = CS_2 - OS_{S2}$. Using this in (10.14) and (10.15) we obtain (10.16)

$$2 \geq 1 + \frac{1}{P} \tag{10.16}$$

It is always true, due to $P \geq 1$.

Let's explain in more details the condition in (10.16) for various values P. If $P = 1$, then it is a sequential algorithm which stores whole matrices A and B on one processor. If $P \to \infty$, then it is a parallel solution where small chunks of A and the whole matrix B are stored on each processor cache.

Calculation of N_{min} and N_{max} in (10.14) and (10.15) correspondingly defines our new methodology to determine a matrix size N for which the algorithm will obtain maximum performance such as superlinear speedup.

However, using a shared memory L2 cache will not lead to a superlinear speedup, as defined by the following theorem.

Theorem 10.5. *The maximum speedup in a shared memory multiprocessor with shared L2 cache among processors is always smaller than P.*

Proof. Since there is no dedicated L2 cache per processor, then L2 is last level shared cache and whole matrices A and B should be stored in it, both for sequential and parallel execution. Therefore, MC_S and MC_P have same values and

$$MC_S < P \cdot MC_P. \tag{10.17}$$

Using (10.7) and (10.17) in (10.3) it follows that

$$MaxSpeedup < \frac{P \cdot CC_P + P \cdot MC_P}{CC_P + MC_P} = P \tag{10.18}$$

10.4 Determination of Superlinear Speedup Regions

A new idea to obtain superlinear speedup assuming that dedicated L2 is the processor's last level cache was elaborated in previous section. Next we continue with details to determine matrix size N range, such that the maximum speedup can be obtained in a real shared memory system.

We'll analyze all 3 possible shared memory environments shown in Table 4.1. In each case, different chunk of matrix A and whole matrix B is stored into L2 cache per core, and the whole matrices A and B will be stored into L3 cache per chip. In a sequential execution, the whole matrices A and B are stored in the cache to the only one core.

We define a methodology to calculate best ranges for matrix size N in order to obtain superlinear speedup in the next 3 sections, 10.4.1, 10.4.2 and 10.4.3.

10.4.1 Multi Chip-Multi Core

In this shared memory environment, matrix A is partitioned to $P = s \cdot c$ partitions A_i with size $N \cdot N/P$. Each partition, together with matrix B is sent to a particular core for calculations.

Theorem 10.6. *The range of matrix size N for which implementation of algorithm (4.4) in a shared memory multiprocessor using multiple chips with shared L3 cache of size CS_3 and multiple cores per chip, where each core has distributed L2 cache with size of CS_2 is given by calculating the values of N_{min} and N_{max} for both L2 and L3 caches.*

$$N \geq \sqrt{\frac{CS_2 - OS_{S2}}{2 \cdot ME}} = N_{min}(L2) \tag{10.19}$$

$$N \geq \sqrt{\frac{CS_3 - OS_{S3}}{2 \cdot ME}} = N_{min}(L3) \tag{10.20}$$

$$N \leq \sqrt{\frac{CS_2 - OS_{P2}}{(1 + \frac{1}{c \cdot s}) \cdot ME}} = N_{max}(L2) \tag{10.21}$$

$$N \leq \sqrt{\frac{CS_3 - OS_{P3}}{(1 + \frac{1}{c}) \cdot ME}} = N_{max}(L3) \tag{10.22}$$

Proof. The 4 relations for matrix size N are derived from (10.14) and (10.15).

Dedicated L2 cache should be able to store one partition of matrix A and whole matrix B and this leads to (10.21). Shared L3 cache should be able to store c partitions of matrix A and whole matrix B leading to (10.22).

For sequential execution, L2 and L3 caches should not store the whole matrices A and B as in (10.19) and (10.20), respectly, both obtained from (10.14).

10.4.2 Single Chip-Multi Core

This shared memory environment has the same characteristics for sequential execution as a Multi Chip-Multi Core. For parallel execution, matrix A is partitioned to c partitions A_i with size $N \cdot N/c$. Each such partition, together with matrix B is sent to a particular core for calculations.

Theorem 10.7. *The range of matrix size N for which implementation of algorithm (4.4) in a shared memory multiprocessor using single chip with L3 cache and multiple cores per chip, where each core has distributed L2 cache with size of CS_2 is given by calculating the values of N_{min} in (10.19) and the following N_{max}*

$$N \leq \sqrt{\frac{CS_2 - OS_{P2}}{(1 + \frac{1}{c}) \cdot ME}} = N_{max}(L2) \tag{10.23}$$

Proof. For L2 cache, this is a special case of Theorem 10.6 and proof can be constructed by setting $s = 1$. Dedicated L2 cache should be able to store one partition of matrix A and whole matrix B leading to (10.23). Shared L3 cache should be able to store the whole matrices A and B which is the same as sequential execution.

We must note that $N_{Max}(L3)$ does not exist if multiprocessor is single chip-multi core. L3 cache is shared in this case and thus sub-linear speedup is achieved as Theorem 10.5 by applying L3 instead of L2 cache.

10.4.3 Multi Chip-Single Core

This shared memory environment has the same characteristics for sequential execution as previous two cases. For parallel execution, matrix A is partitioned to s partitions A_i with size $N \cdot N/s$. Each partition, together with matrix B is sent to a particular chip (core) for calculations. In this case, L3 cache is irrelevant.

Theorem 10.8. *The range of matrix size N for which implementation of algorithm (4.4) in a shared memory multiprocessor using multiple chip with single core per chip and L2 cache per core (chip) with size of CS_2 is given by calculating the values of N_{min} in (10.19) and N_{max} in*

$$N \leq \sqrt{\frac{CS_2 - OS_{P2}}{(1+\frac{1}{s}) \cdot ME}} = N_{max}(L2) \tag{10.24}$$

Proof. For L2 cache, this is a special case of Theorem 10.6 and proof can be constructed by setting $c = 1$. Dedicated L2 cache should be able to store one partition of matrix A and whole matrix B leading to (10.24).

10.5 Testing Methodology and Theoretical Results

In this section we present the theoretical results and hypotheses for different multi-core / multiprocessor shared memory systems shown in Table 10.2.

Case	Cpu	Sock.	Cores	L2 Type	L2 Size	L3 Size
1	Opteron(tm) Quad	4	16	Ded.	512K	2MB
2	Phenom(tm) Quad	1	4	Ded.	512K	2MB
3	Xeon(TM) Virt.	2	2	Ded.	1M	-
4	Athlon X4 Quad	1	4	Ded.	512K	-
5	Core2Duo Mobile	1	2	Shared	3M	-
6	Xeon(TM) Quad	1	4	Shared	12M	-

Table 10.2 Test Case Environments [52]

A set of experiments will be performed measuring performance dependence on the matrix size.

For each test case in Table 10.2 we calculate N_{max} and N_{min} according to (10.14) and (10.15) with assumption that P is maximum number of cores for particular test case and $N = q \cdot P$, where $q \geq 1$ is integer. Table 10.3 presents these calculated values for matrix size assuming that matrix element to be double precision floating point number, i.e. size is $ME = 8B$.

Case	Cores	N_{min}	N_{max}	Expected Speedup
1	16	362	458	Super-Linear
2	4	181	228.97	Super-Linear
3	2	256	295.6	Super-Linear
4	4	181	228.97	Super-Linear
5	2	362	362	sub-Linear
6	4	886.81	886.81	sub-Linear

Table 10.3 Calculated Matrix Size for Each Test Case [52]

Table 10.3 shows calculated values for N where a *Super-Linear speedup* can be obtained for dedicated L2 cache shared memory system. For comparison purposes we present in Table 10.3 also results for shared L2 cache. According to the Theorem 10.5 when L2 cache is last level shared cache then there is limited maximum speedup.

The test cases of Table 10.3 where superlinear speedup is calculated are analyzed in details in the following sections.

10.5.1 Results for Multichip - Multicore systems

Using (10.19) - (10.22) and the values for test case 1 from Table 10.2, we can calculate theoretical border values for matrix size N for maximum speedup shown in Table 10.4.

Cache	N_{min}	N_{max}
L2	181	248
L3	362	458

Table 10.4 Theoretical values for maximum speedup for Case 1 [52]

Let's explain the results. For each $N < 181$, both in sequential and parallel execution, L2 and L3 caches will be enough to store the matrices. For each $181 <= N <$ 248, speedup will rise, due to L2 cache will be small to store the necessary matrices in sequential execution, but enough for parallel execution. L3 cache in this range is

also enough for both sequential and parallel. For each $248 <= N < 362$, speedup will rise slower, but still rise, due to L2 cache will not be enough, but L3 will be enough for parallel execution. For $N > 362$ the speedup decreases, especially when $N > 458$.

10.5.2 Results for Singlechip - Multicore systems

To achieve superlinear speedup we use (10.23) and the values for test case 2 from Table 10.2. The obtained theoretical border values for matrix size N for maximum speedup are shown in Table 10.5.

Cache	N_{min}	N_{max}
L2	181	229

Table 10.5 Theoretical values for maximum speedup for Case 2 [52]

As a conclusion L3 cache is shared and does not affect the performance. Only the range of matrix size N between 181 and 229 can benefit with superlinear speedup due to dedicated L2 cache.

10.5.3 Results for Multichip - Singlecore systems

The theoretical border values for matrix size N for maximum speedup is calculated based on (10.24) and the values for case 3 from Table 10.2. The results are shown in Table 10.6.

Cache	N_{min}	N_{max}
L2	256	296

Table 10.6 Theoretical values for maximum speedup for Case 3 [52]

The meaning is that superlinear speedup can be achieved for N within the given range of 256 and 296.

10.5.4 Results for Singlechip - Multicore dedicated cache systems

To achieve superlinear speedup we use (10.23) and the values for test case 4 from Table 10.2. The obtained theoretical border values for matrix size N for maximum speedup are shown in Table 10.7.

Cache	N_{min}	N_{max}
L2	181	229

Table 10.7 Theoretical values for maximum speedup for Case 4 [52]

As a conclusion only the range of matrix size N between 181 and 229 can benefit with superlinear speedup due to dedicated L2 cache.

10.6 Experimental Results

This section presents the results of series of experiments performed on different multicore / multiprocessor shared memory systems shown in Table 10.2.

The tests include experiments where the performance dependence is tested upon different matrix sizes for the maximum usage of P processing elements. The results for obtained maximum speedup are presented for all test cases where we theoretically calculated both superlinear and sublinear speedup.

10.6.1 Case 1: Multi Chip-Multi Core

The results of the theoretical analysis for possible superlinear speedup are presented in Table 10.4. The practical experiments for case 1 are shown in Figure 10.3 and they confirm our theoretical results.

There is a superlinear speedup in range $288 <= N <= 480$, and maximum speedup is more than 26 with 16 processors for $N = 400$.

10.6.2 Case 2: Single Chip-Multi Core

The results of the experiments for case 2 are depicted in Figure 10.4 and also confirm our theory.

Superlinear speedup is obtained for range $192 <= N <= 736$, and maximum speedup is more than 5.6 with 4 processors for $N = 400$.

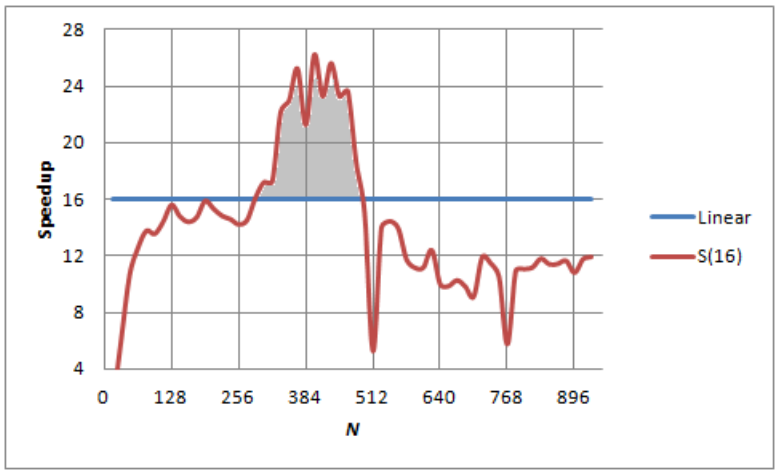

Fig. 10.3 Experimental Speedup for Test Case 1 [52]

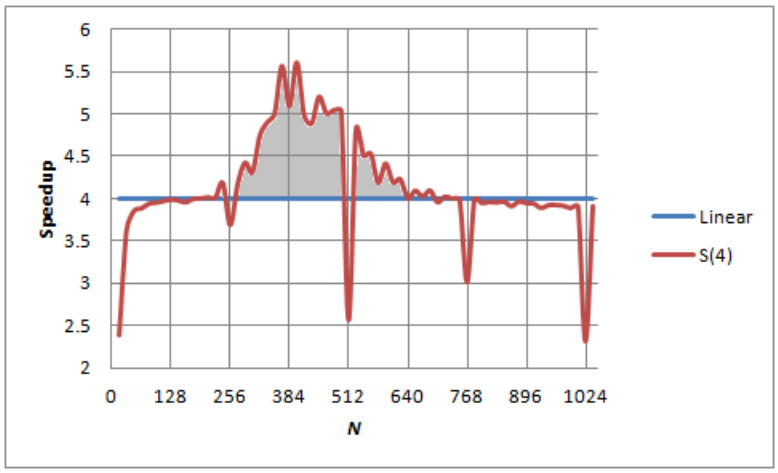

Fig. 10.4 Experimental Speedup for Case 2 [52]

10.6.3 Case 3: Multiple Chip-Single Core

The theoretical border values for matrix size N for maximum speedup as calculated by introduced methodology are presented in Table 10.6. The results of the experimental research for case 3 are depicted in Figure 10.5 and also confirm our theory with a little offset (a situation that appears due to the existence of operating system code per each processor in the cache, so not all size is available to store matrix and executed the algorithm).

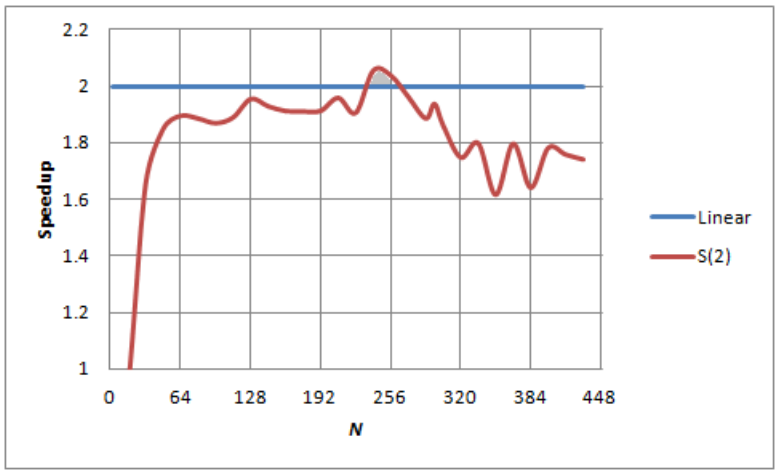

Fig. 10.5 Experimental Speedup for Case 3 [52]

Superlinear speedup is achieved only in range $240 <= N <= 256$. The maximum speedup value is just a little above the linear with 2 processors for $N = 240$. Note that these results differ from theoretical since the testing environment was virtualized using VMware ESXi 4 [158] and there were only 2 processors without usage of L3 cache.

10.6.4 Case 4: Single Chip-Multi Core - Dedicated L2 Without L3

These experiments were performed on virtualized environment using Microsoft Hyper-V Server 2008 R2 [90]. The results of the experiments for case 4 are depicted in Figure 10.6 and also confirm our theory.

Superlinear speedup is obtained for range $120 <= N <= 224$. The maximum speedup is more than 5.56 with 4 processors for $N = 156$.

10.6.5 Case 5: Single Chip-Multi Core - Shared Cache in Core2Duo

The theoretical analysis for sublinear speedup in shared L2 cache is proved with the experimental results depicted in Figure 10.7. Both the Theorem 10.5 and Figure 10.7 prove that the speedup in this case is always smaller than P.

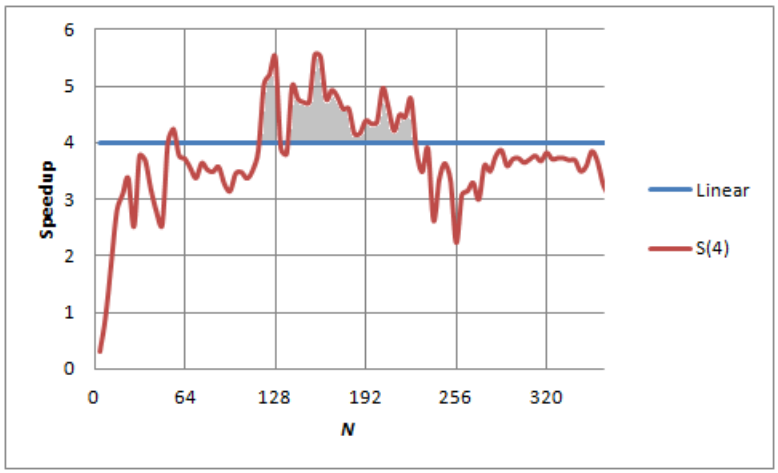

Fig. 10.6 Experimental Speedup for Case 4 [52]

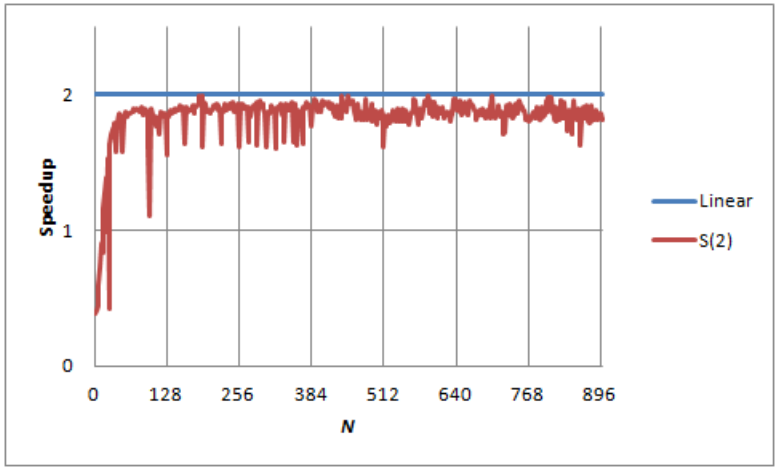

Fig. 10.7 Experimental Speedup for Test Case 5 [52]

10.6.6 Case 6: Single Chip-Multi Core - Shared Cache in CoreQuad

The theoretical analysis for sublinear speedup in shared L2 cache is proved with the experimental results depicted in Figure 10.8. Both the Theorem 10.5 and Figure 10.8 prove that the speedup in this case is always smaller than P.

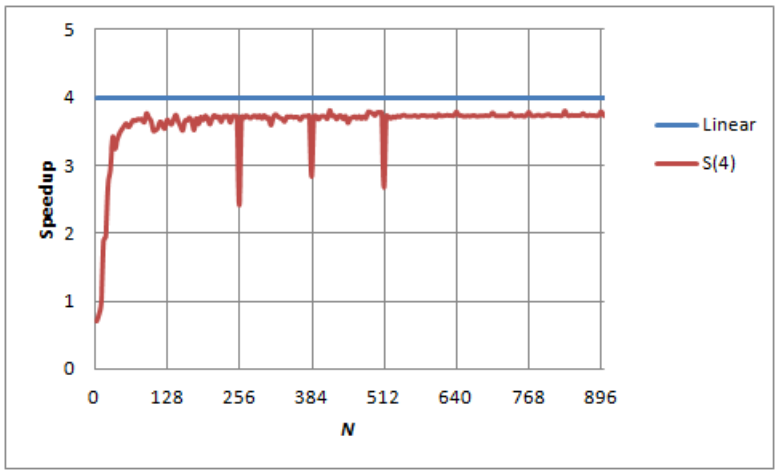

Fig. 10.8 Experimental Speedup for Test Case 6 [52]

10.7 Discussion

Superlinear speedup is achieved for test cases 1, 2, 3 and 4 of Table 10.2 due to conditions of Theorems 10.4 and 10.2, i.e. the existance of distributed L2 cache memory (and dedicated L3 cache in case 1) per processor and the cache memory speed in sequential execution is huge enough compared to memory and processor clocks in parallel execution. Furthermore, conditions of Theorem 10.2 are not satisfied for test cases 5 and 6 since they are subject of Theorem 10.5 which proves sub-linear speedup for these cases.

Although superlinear regions are achieved for expected test cases, they are not completely in the regions as calculated theoretical results. There are many factors with impact to speedup elaborated in the following sections.

10.7.1 Wider Superlinear Region

Although Theorems 10.4 and 10.2 prove the existence of a superlinear speedup, it's range determined experimentally is wider than the superlinear region calculated theoretically, i.e. for N greater than L2P and L3P. Using dedicated cache per core will generate smaller number of cache misses in corresponding cache rather than sequential for the same problem size N.

10.7.2 Shared Memory Competition

Increasing the number of processors increases the competition for the same memory resources, which provides greater memory latency than sequential. Test case 1 executes on 16 cores in parallel, placed in 4 chips. All 16 cores access the same data from main memory generating bottleneck for huge N after point L3P (Last level cache) in Figure 10.2. This is the reason why test case 1 has smaller superlinear region than test case 2.

As a prove we perform another test as a subtest of test case 1, i.e. executing parallel on the same platform for test case 1 but on 4 processors instead of 16, depicted in Figure 10.9.

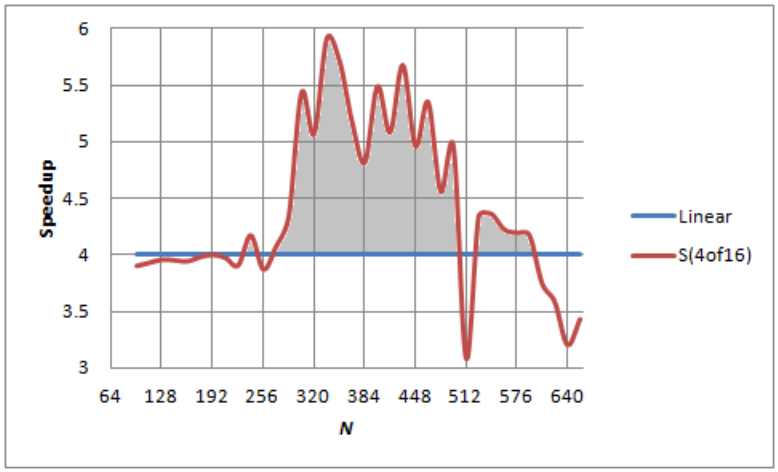

Fig. 10.9 Experimental Speedup for Test Case 1 for $P = 4$ of 16 [52]

The superlinear range is similar as test case 2 which should be because the cache sizes are same.

10.7.3 Cache Occupancy due to OS and Virtualization

Theoretical ranges for superlinear regions are calculated as the whole processor cache is available for the algorithm. However, some part of cache memory is occupied by OS as depicted in Figure 10.1. Even more the cache is occupied by the host OS in virtualized environment such as test case 3 where experimental N_{Min} and N_{Max} are smaller than theoretical.

10.7.4 What about 2D blocking Matrix Multiplication

This section describes the results of the experiments with more efficient matrix multiplication algorithm with 2D matrix blocking. The blocking algorithm is one of the ten advanced optimization of CPU cache performance presented in [63]. Our hypothesis is that existing superlinear regions for parallel execution presented in previous Chapter 10 do not appear due to bad cache utilization of the basic matrix multiplication algorithm described in Section 4.1, but due to distribution of workload to multiple processors and therefore smaller cache memory requirements for distributed L1 cache using the shared L2 cache presented in Section 10.3.

We realize matrix multiplication with 2D blocking both sequential execution using 1 core and parallel execution using 4 cores in test case environment 2 shown in Table 10.2. We use block submatrices of size 48x48 elements that can be placed in L1 cache. The experiments range is from matrix size $N = 96$ to $N = 2536$ with step $P = 4$.

Figure 10.10 depicts the experimental speedup of parallel execution. It proves also existing of superlinear speedup starting from $N = 1524$ in some points, and in all points for $N > 1976$. We must note that the speedup curve rises and falls down because the block size is constant and is not a divisor of matrix size, thus reducing the efficiency for parallel execution.

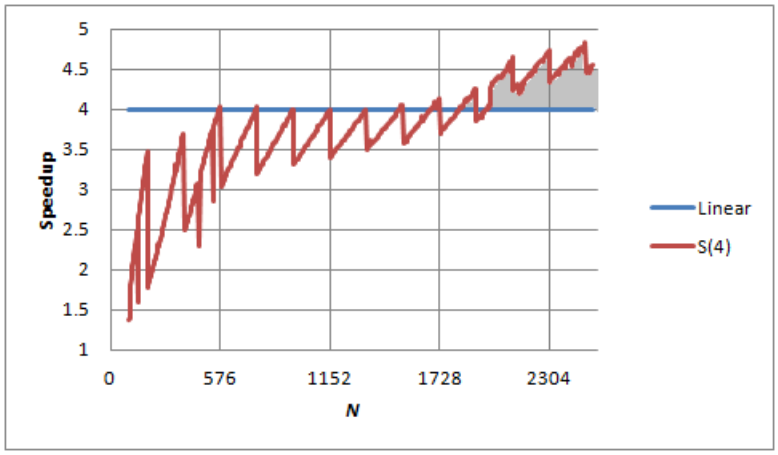

Fig. 10.10 Experimental Speedup $S(4)$ for blocking matrices [52]

10.8 Summary

The theoretical analisys in this chapter for superlinear speedup in dense matrix multiplication algorithm as an example was realized by the authors in [52] for the purpose of this thesis research. We tried to go beyond the limits specified in Gustafson's law not just finding examples of **superlinear speedup** for matrix multiplication but also to provide theoretical analysis how to achieve it in a real shared memory system. We show and provide a proof of its existence in a real multiprocessor system that uses caches. The superlinear speedup is possible in cases where sequential execution initiates more cache misses than for parallel execution. This happens for example, in a shared memory multiprocessor with dedicated caches.

We have presented the theoretical background of superlinear existence and also introduced a methodology how to achieve it, when and where it can be achieved. We have also defined a testing methodology and realized a series of experiments to provide evidence of superlinear speedup and unexpected high performance.

As a conclusion mathematical relations showed a possibility of superlinear speedup and extensive experimental research approved the results showing real cases of increased unexpected performance in a multiprocessor or multicore system (or both), with dedicated cache per core and thus propose using such systems in parallel computing.

The results and methodology can be used in the massive data computations with high cache miss ratio. Dividing data into a smaller chunks with optimal size calculated with our methodology, reduces cache misses in parallel execution in the dedicated cache per core system. Although we simplified our calculations, our methodology can be used in other similar high performance numeric computations.

The achieved high performance results and superlinear speedup is demonstrated on the example of dense matrix multiplication algorithm. It is possible within a range of values of matrix sizes, even if the environment is virtualized.

The results in this paper show superlinear speedup for some range of matrix size N and prove the theoretical analysis. Another important result was obtaining superlinear speedup in virtualized environment with two different hypervisors: Microsoft Hyper-V and VMware ESX.

Chapter 11
Superlinear Speedup in Matrix Multiplication on GPU

Abstract In previous chapters 10 and **??** we present teoretically and experimentally how to achieve superlinear speedup for matrix multiplication algorithm on specific multiprocessors. In this chapter we present the analysis and the experiments of the research realized by the authors in [33] for the purpose of this thesis. The authors theoretically analyzed and experimentally achieved superlinear speedup for GPU devices, which are also categorized as SIMD. We implement a structure persistent algorithm which efficiently exploits the shared cache memory and avoids cache misses as much as possible.

11.1 How to Achieve Superlinear Speedup in GPU

In this section we present the theoretical analysis that was performed by the authors to determine how to achieve superlinear speedup in GPU for matrix multiplication algorithm.

11.1.1 Superlinear regions

Cache memory occupancy for sequential execution is greater than the cache memory occupancy for parallel execution since, smaller chunks of data are stored, thus allowing larger size problems to be stored without generation of cache misses. As described in Section 4.5 if the cache memory requirements fit in L1 cache, then we expect the highest processor speed and call this region L1 region. In L2 region data does not fit in L1, but fits in the L2 cache generating cache misses in L1, but not for L2.

The theoretical expectations for performance are presented in Figure 11.1. Cache occupancy for the sequential execution is defined by the points L1S and L2S and for the parallel execution by the points L1P and L2P. Due to different dedicated L1

cache occupancy in dedicated L1 cache there is difference between L1 regions for sequential and parallel execution. We expect better performance for parallel execution in case when the sequential execution generates cache misses for L1 and L2 still does not generate cache misses, and in case of parallel execution L1 does not generate cache misses. This region is called *superlinear region* since it leads to a possible superlinear speedup [124].

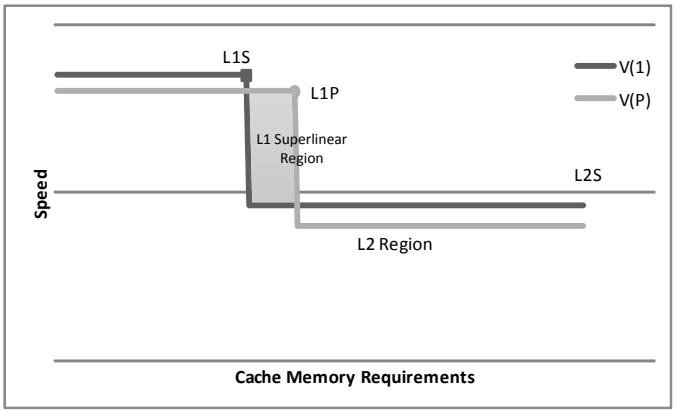

Fig. 11.1 Expected speed with real cache [33]

11.1.2 Analysis of Memory Utilization

Based on the SIMD architecture as described in Section 4.5, the L1 cache memory is shared among all 32 threads that run per single SM. However, we are interested only in scenarios where each core has its own dedicated cache memory, in order to test our assumption presented in Section 11.1.1. Hence, L1 cache is dedicated only per SM. On one hand we can control how many threads to run per threadblock, but on the other hand, there is no specified way to control how many threadblocks can be run per SM. Nevertheless, there are limitations regarding active threadblocks per SM, so by allocating maximum shared memory we can ensure that only one threadblock is running per SM. Thus, if the threadblock is defined by one thread, we ensure that one thread is running per SM.

A sequential implementation of our algorithm that runs one thread per only one active SM, occupies the L1 cache memory as presented in Fig. 11.2, where the accessed memory blocks are denoted with gray color.

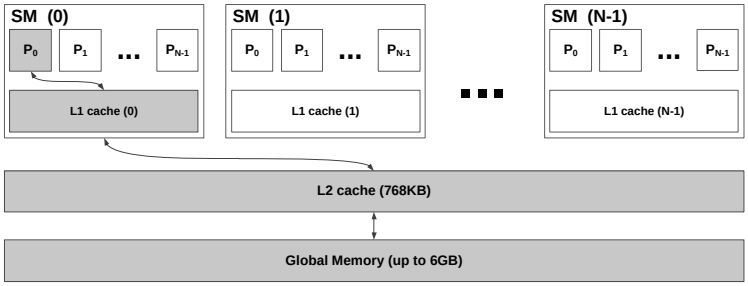

Fig. 11.2 Memory utilization of the sequential implementation [33]

The cache memory occupancy for the parallel implementation is depicted in Fig. 11.3. In this implementation several SMs are used such that each SM uses only one thread and full L1 cache. L2 cache in this implementation is shared among all SMs.

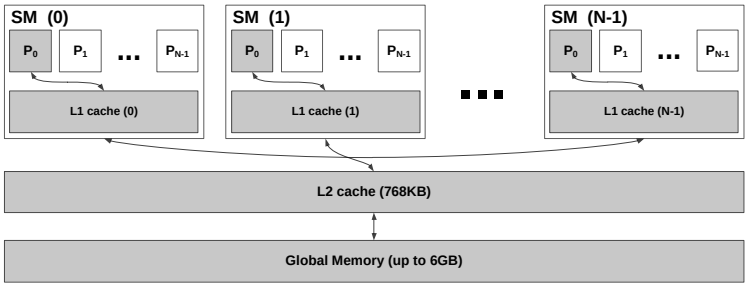

Fig. 11.3 Memory utilization of the parallel implementation [33]

11.2 Testing Methodology

The testing methodology is based on 2 experiments which show speedup dependence on problem size (cache memory requirements) and on the number of processing elements.

11.2.1 Testing data

Since, the L1 cache of the GPU device is configurable, and additionally we can configure the cache line sizes at compile time, we can obtain significant performance discrepancy. There are many aspects about performance discrepancy [161] for different combinations presented in Table 11.1. Hence, to obtain the most optimized implementation, all combinations need to be tested upon in order to decide the optimal L1 cache memory size and the caching vs. non-caching load operations.

L1 size	Cache lines
16KB	128bit
16KB	32bit
48KB	128bit
48KB	32bit

Table 11.1 Combinations of L1 cache memory sizes and cacheline sizes [33]

11.2.2 Testing Environment

We have performed our tests with the GPU device specified in Table 11.2. All experiments were performed on the same hardware infrastructure (Intel i7 920 CPU at 2.67GHz, with 12GB RAM at 1333MHz) and Linux OS Ubuntu 10.04 LTS. The implementations were compiled with the NVIDIA compiler nvcc from the CUDA 4.2 toolkit.

GPU Feature	Value
Cores	448
Number of SPs per SM	32
Number of SMs	14
L1 cache / Shared Memory (KB)	16/48
L2 cache Memory (KB)	736
DRAM (GB)	6

Table 11.2 GPU Device Specifications of Tesla C2070 [33]

11.3 Results

11.3.1 Speed and Speedup vs Cache Requirements

The results of our first experiment for the matrix multiplication is presented in figures 11.4 and 11.5. Figure 11.4 depicts the speed, where V(1) stands for the speed of the sequential execution. In order to better depict the Figure 11.4 for parallel execution on 14 cores, we have normalized the speed per core, therefore V(14) presents the average speed per processing element. Accordingly, the speedup S(14) is denoted for the parallel execution on 14 cores for the same experiment in Figure 11.6. Thus, it is easy to notice that there is an existence of a superlinear region, and an appropriate superlinear speedup.

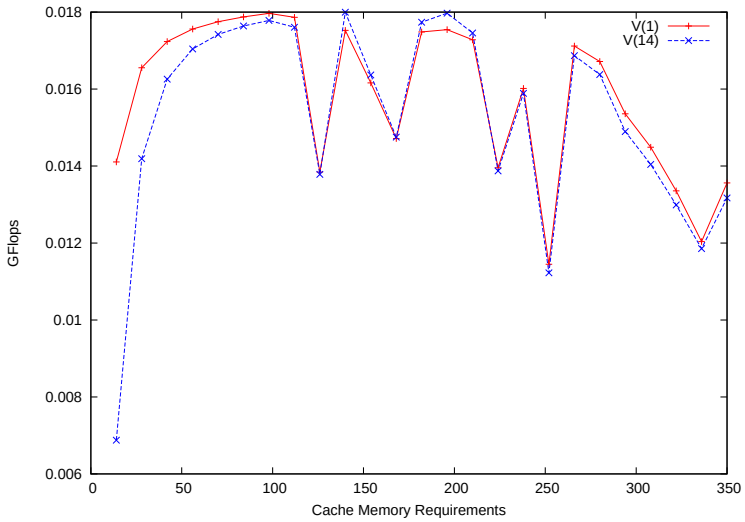

Fig. 11.4 GPU speed for sequential execution (1 active SM) and the average normalized speed per core (14 active SMs) in parallel execution [33]

11.3.2 Speedup vs SMs

In our second experiment we are testing the dependency of the speedup and the number of processing elements. For this experiment, we select 2 problem sizes from the superlinear L1 region and L2 as presented in Figure 11.6. We have performed matrix multiplication on different number of processing elements (in our case SMs).

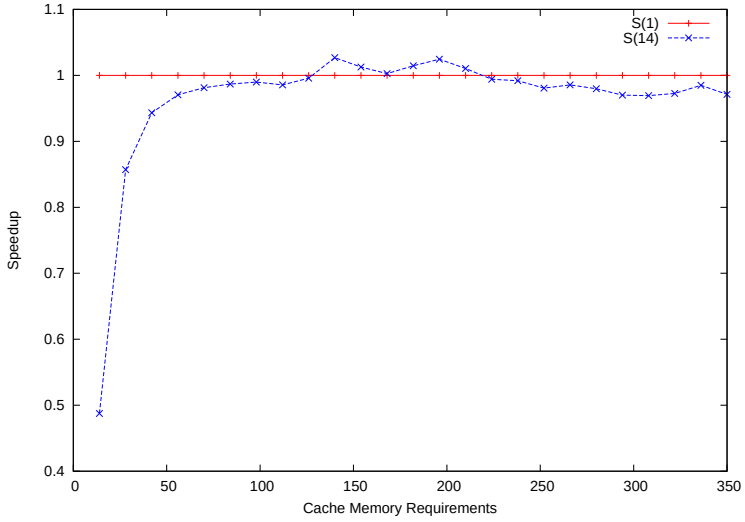

Fig. 11.5 GPU speedup for the parallel execution (14 active SMs) [33]

The results depicted in Figure 11.6 confirm the existence of superlinear speedup for each parallel execution with 2, 4, 6, 8, 10, 12 and 14 SM in the superlinear region.

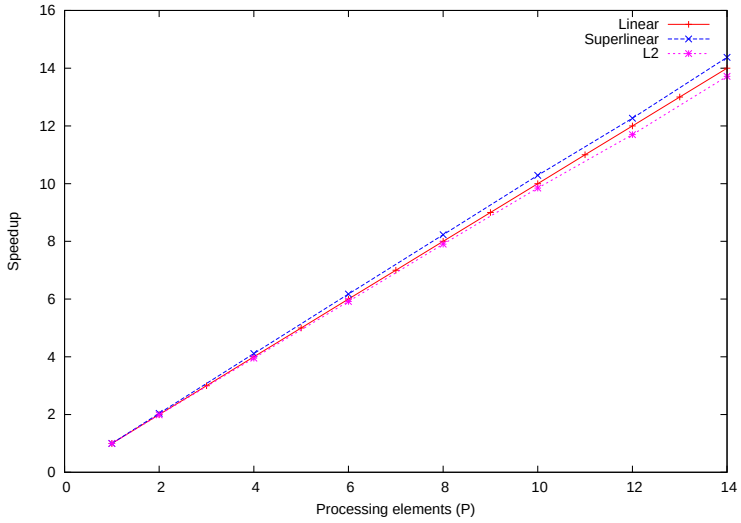

Fig. 11.6 GPU speedup for the second experiment [33]

11.4 Summary

This chapter presents the results for achieved superlinear speedup in GPU devices realized by the authors in [33]. The experiments have confirmed the theoretical analysis about existence of superlinear regions of the problem size for matrix multiplication using GPU devices, where the normalized performance per processing element for parallel execution is better than in sequential execution. Thus, we have obtained a superlinear speedup beyond the limits specified in Gustafson's law. However, we have only obtained superlinear speedup in the superlinear region where the cache memory requirements of the problem fit in L2 for parallel execution and generate L2 cache misses for sequential execution.

Based on the experiments, we have presented further proof that there is existence of superlinear speedup for SIMD architecture processors with dedicated caches, more particular GPU devices.

The speedup performance is directly dependent on cache performance and generation of cache misses. The experiments prove that superlinear speedup is possible for particular number of SMs in for particular problem size in superlinear speedup region.

Part IV
Performance Analysis of Cache Intensive Algorithms in Cloud Computing

Chapter 12
Virtualization Impact on Cache Intensive Algorithm Performance

Abstract Cache intensive algorithms performance depends mostly of cache parameters. The program should be executed slower in cloud virtual environment compared to traditional server due to additional virtualization layer. There is also a hypothesis that the speedup in cloud virtual environment for parallel execution is smaller than in sequential. However, there is a region with particular problem size for matrix multiplication where the program runs faster in virtual environment. In this chapter we will describe the results of the experiments published by the authors in [48] for the purpose of this thesis research.

12.1 Testing Environment

The experiments are performed on two same HP Proliant DL380 G7 servers, each with 2 x Intel(R) Xeon(R) CPU X5680 @ 3.33GHz GHz [168] and 24GB RAM. The processor has 6 cores, each with 32 KB 8-way set associative L1 cache dedicated per core, and 256 KB 8-way set associative L2 cache dedicated per core. All 6 cores share 12 MB 16-way set associative L3 cache.

The first server is installed with Linux Ubuntu 10.10 as traditional host OS on one real hardware machine. The second server uses Linux Ubuntu 10.10 installed as guest OS with all resources dedicated to the virtual machine using VMware ESX Server 5i [159].

Execution time is measured for the matrix multiplication algorithm on both traditional and virtual servers during the test experiments.

The test experiments have two goals. The first goal is to model the multiprocessor's behavior, i.e. speed and speedup in computationally and memory intensive matrix multiplication algorithm on both environments as a baseline. The second goal is to model the performance drawbacks due to virtualization implementation on the second server compared to traditional server.

Matrix multiplication algorithm described in Section 4.1 and its parallel implementation from Section 4.3.1 are used as test data.

All test cases are executed using different matrix size N for different number of processing elements from 1 to 12. Tests are performed by unit incremental steps for matrix size and number of processors.

12.2 Parallel Matrix Multiplication in Traditional Environment

This Section presents the result of the baseline experiments in traditional environment. Figure 12.1 presents the speed expressed in gigaflops versus matrix size N for each test case with different number of processing elements $P = 1, 2, \cdots, 12$. Figure 12.1 presents huge performance degradation when $N > 512$. This is the entrance in L4 region, i.e. the region where the matrix elements do not fit in last level L3 cache generating cache misses in L3 cache. In this region the matrix elements should be loaded from the main memory, instead of L3 cache, which is time consuming operation.

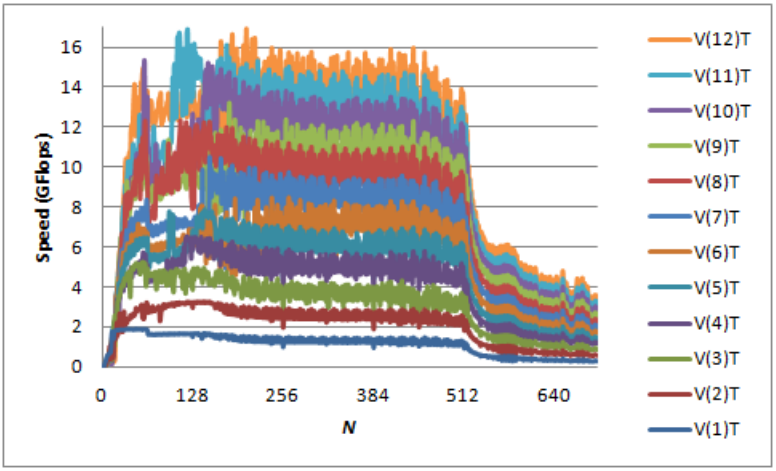

Fig. 12.1 Speed in Traditional (T) Operating System [48]

Figure 12.2 depicts the speedup as a function of matrix size N for each test case with different number of processing elements $P = 1, 2, \cdots, 12$. The speedup curves satisfy Gustafson's Law [55] and saturate below maximum speedup P besides the speed degradation in L4 region.

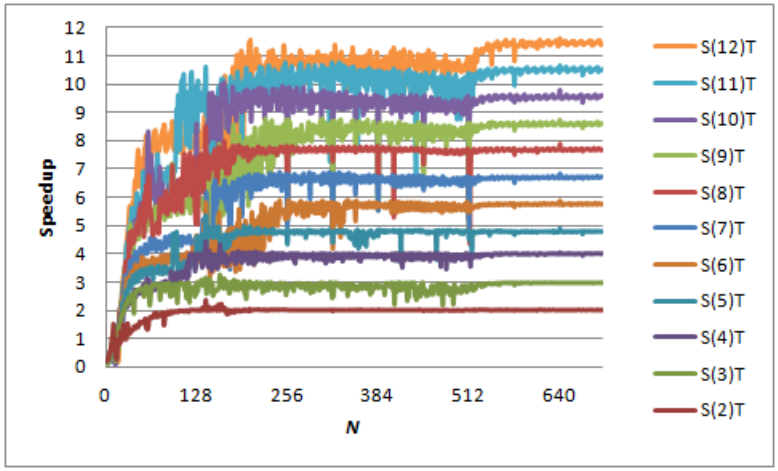

Fig. 12.2 Speedup in Traditional (T) Operating System [48]

12.3 Parallel Matrix Multiplication in Virtual Environment

This Section presents the result of the baseline experiments in virtual environment. The speed graph expressed in gigaflops versus matrix size N is shown in Figure 12.3 for $P = 1, 2, \cdots, 12$ processing elements. Figure 12.3 also presents huge performance degradation when $N > 512$ similar to the traditional environment.

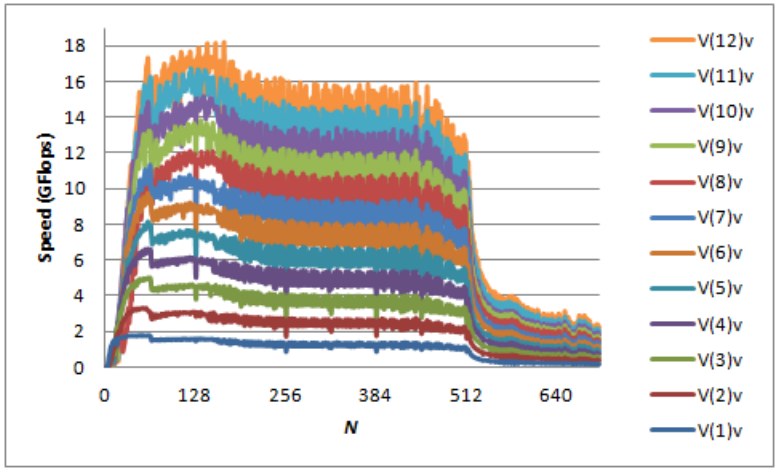

Fig. 12.3 Speed in Virtualized (v) Operating System [48]

Figure 12.4 depicts the speedup as a function of matrix size N for $P = 1, 2, \cdots, 12$ processing elements. The difference in this environment is the existence of a superlinear L4 region, which is emphasized for $P > 6$. This happens due to existence of a superlinear region when using the second L3 cache compared to the experiments performed only on one chip with one L3 cache.

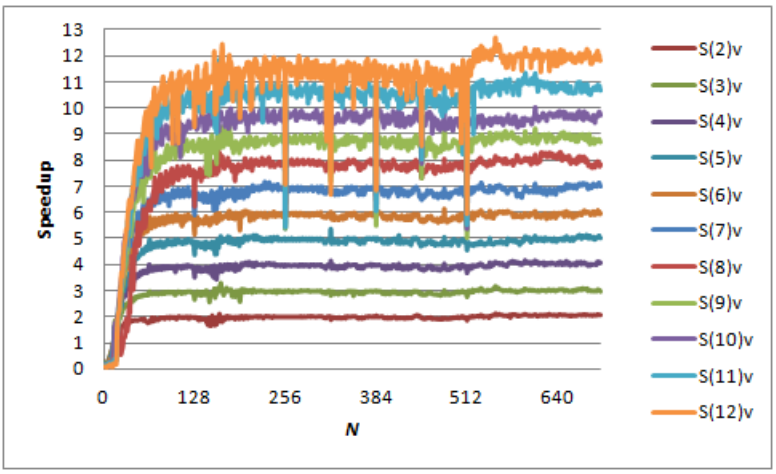

Fig. 12.4 Speedup in Virtualized (v) Operating System [48]

12.4 Traditional vs Virtual Environment

This Section compares the performance of traditional and virtual environments using the performance baseline presented in Sections 12.2 and 12.3.

12.4.1 Speed Comparison

Comparing Figures 12.1 and 12.3 one can conclude that the curves are nearly identical. All L1, L2, L3 and L4 regions are expressed in the same range of N for particular P. But, there is a difference for speed values in different regions. Figures 12.5 and 12.6 correspondingly present the speed difference between traditional and virtual environment for odd and even number of processing elements P.

The figures present that the virtual environment performs with discrepancies. On contrary to our hypothesis, virtual environment provides better performance in L1 and L2 regions compared to traditional as number of processors P increases. In

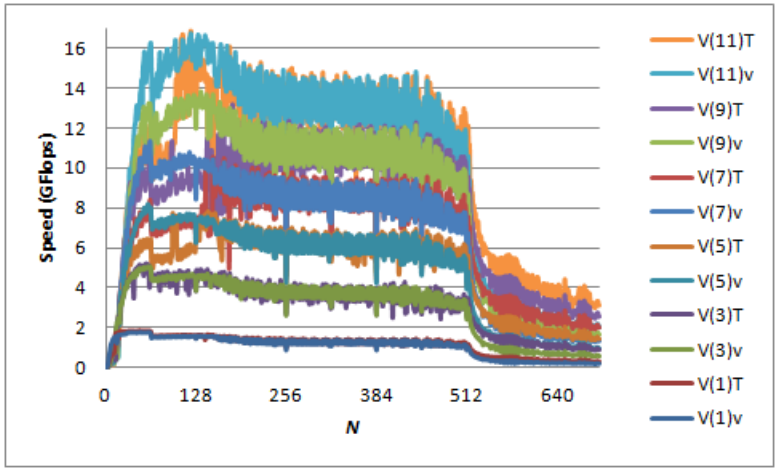

Fig. 12.5 Speed comparison for odd processing elements [48]

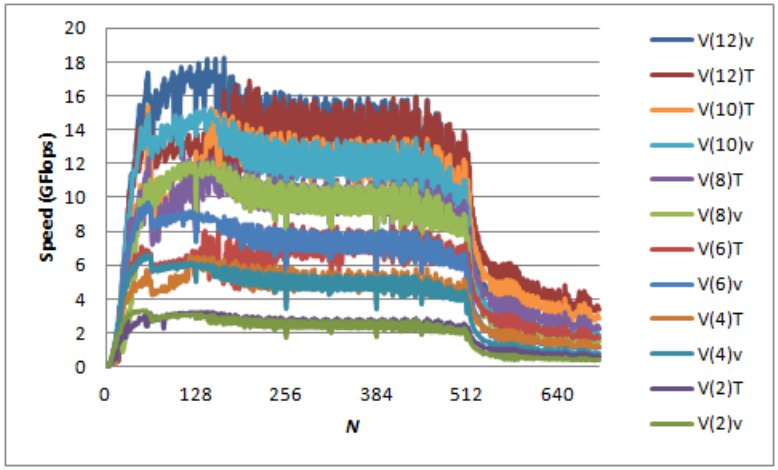

Fig. 12.6 Speed comparison for even processing elements [48]

L3 region both environments provide similar performance. But virtual environment provides huge performance drawback in L4 region compared to traditional depicted in Figure 12.7.

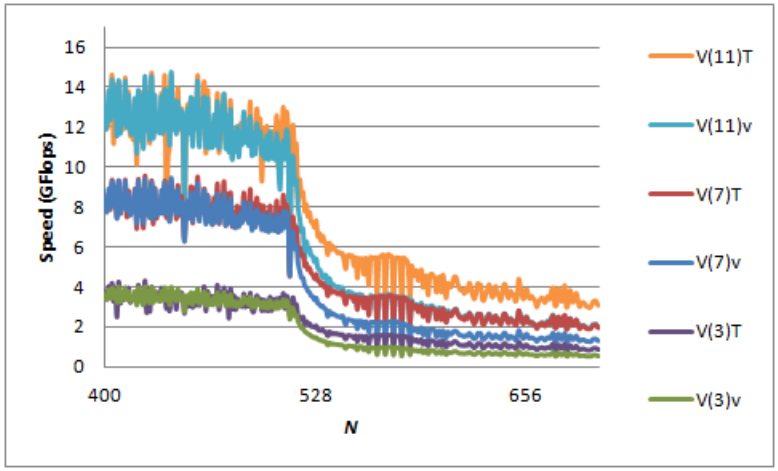

Fig. 12.7 Speed comparison in L4 region for better presentation [48]

12.4.2 Speedup Comparison

Speedup comparison is depicted in Figures 12.8 and 12.9. On contrary to our hypothesis, one can conclude from both figures that speedup in virtual environment is greater than speedup in traditional environment.

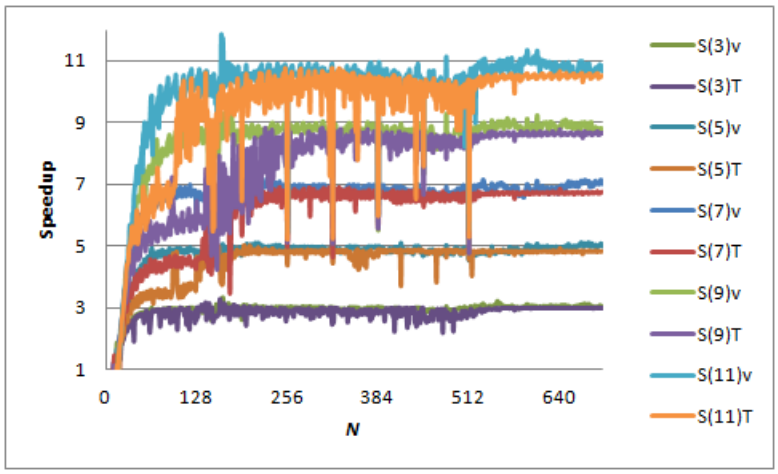

Fig. 12.8 Speedup comparison for odd processing elements [48]

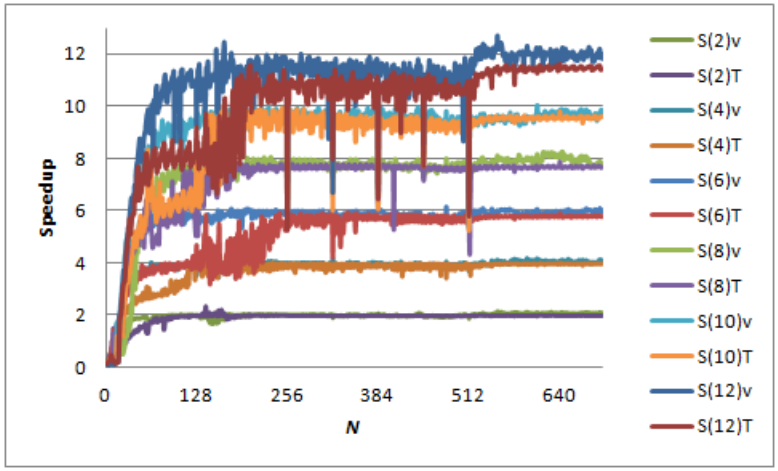

Fig. 12.9 Speedup comparison for even processing elements [48]

The performance drawback in virtual environment compared to traditional is exposed in the L4 region. Figures 12.10 and 12.11 present the relative performance expressed as percentage via ratio of achieved speeds in virtual and traditional environments for odd and even number of processing elements P. It is assumed that the speed in traditional environment is 100%.

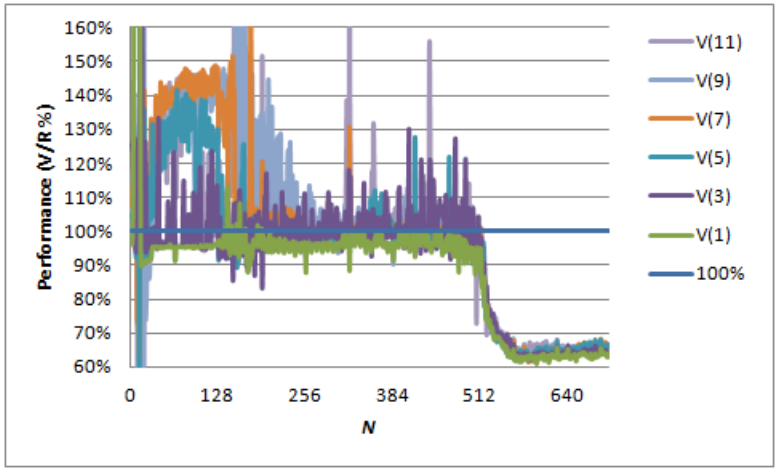

Fig. 12.10 Relative Performance for odd processing elements [48]

Using any processors P from 3 to 12, the virtual environment provides better performance than traditional in L1, L2 and L3 regions. For L4 region, there is a

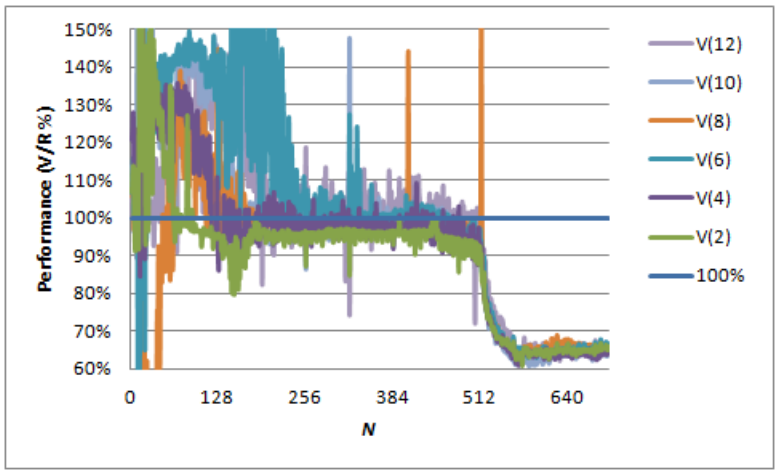

Fig. 12.11 Relative Performance for even processing elements [48]

huge performance drawback in virtual environment compared to traditional, despite the performance drawback in both solutions in this region, compared to regions L1, L2 and L3.

12.4.3 Performance Comparison in Cache Regions

Table 12.1 presents the average performance ratio between virtual and traditional environment for regions L1, L2 and L3 as a group where virtual environment provides better performance, as well as for L4 region where virtual environment provides huge performance drawback compared to traditional. All performance percentages are given for each processing element P from 1 to 12.

The maximum performance as shown in Table 12.1 for virtual environment is 14.62% better than traditional in L1-L3 region. The results show average 5.65% better performance than traditional before L4 region, but only 66.58% of performance that the traditional environment provides in L4 region. It also means that in this region the performance is 33.42% worse than traditional.

12.5 Discussion

The experimental results seem very discrepant. Our hypothesis that programs will run slower in virtual environment compared to traditional server failed in many test cases. The similar situation is also shown for speedup when executing in parallel.

P	L1-L3 Regions	L4 Region
1	98.02%	64.88%
2	99.72%	66.51%
3	101.54%	66.27%
4	104.29%	65.79%
5	105.62%	66.42%
6	114.62%	67.07%
7	110.00%	66.79%
8	98.26%	67.82%
9	111.42%	66.75%
10	109.30%	65.28%
11	105.80%	67.14%
12	109.23%	68.26%
Avg	**105.65%**	**66.58%**

Table 12.1 virtual vs traditional average speed performance [48]

Figures 12.10 and 12.11 present that for each number i of processing elements ($i = 2,3,...,P-1$) virtualization provides better performance than traditional in L1 and L2 regions. One can conclude that virtualization handles better in distributed environment rather than shared memory.

As additional confirmation to conclusion, L3 cache is "semi-shared semi-distributed", i.e. it is shared but not for all data. In L3 region virtualization and traditional environment provide similar performance.

The huge performance drawback in L4 region that virtual environment provides can be explained due to sharing the last level cache - shared memory, despite both test environment create evenly number of cache misses.

12.6 Summary

We performed detailed experiments and concluded a huge performance discrepancy in virtualized environment. Virtualization provides almost equal performance of 98% as traditional for sequential execution on compute, memory and cache intensive matrix multiplication algorithm. Using parallelization it provides even greater speed of almost 15% for particular number of processing elements in L1 and L2 regions.

The virtualization loses the battle in L4 region when a lot of costly cache misses appear. Its performance is 33.42% worse than traditional in this region.

Despite the hypothesis, the experimental results prove that virtualization performance is even better than traditional for distributed memory, rather than shared where there is a huge performance drawback.

Chapter 13
PaaS Impact on Cache Intensive Algorithm Performance

Abstract Different runtime environment also impacts the performance. Opposite to the hypothesis that Linux based runtime environment provides better performance, the results of the experiments show that Windows based runtime environment runs up to 2.5 times better than Linux, especially for problem size that can be placed in cache and will not generate a lot of cache misses. Both environments are hosted in Windows Azure Cloud. In this chapter we will describe the results of the experiments realized by the authors in [51] for the purpose of this thesis research.

13.1 Testing Methodology

This section describes the testing methodology based on 2 different platforms and 4 test cases for each platform.

13.1.1 Testing Algorithm

Matrix multiplication algorithm described in Section 4.1 and its parallel implementation from Section 4.3.2 are used as test data.

13.1.2 Testing Environments

The experiments are realized in Microsoft Windows Azure Cloud. Details about Windows Azure Platform, its components and architecture are given in [109]. Two different platforms are analyzed as testing environments using the same runtime environment hosted in the same hardware infrastructure.

Hardware Infrastructure is the same for each test case. It consists of Windows Azure Extra Large virtual machine instance with 2 processors AMD Opteron 4171 HE and total 8 CPU cores. The processor has 6 cores, but only 4 cores are dedicated per virtual machine instance. Each core possesses 64 KB L1 data and instruction caches dedicated per core, 512KB L2 dedicated per core. L3 cache with total 5 MB is shared per chip.

13.1.3 Test Platforms

Two different platforms are used:

- *Windows Server 2008* is used on the instance. C# is used with .NET framework 4 using threads for parallelism.
- *Linux Ubuntu Server* 12.04 is used for the instance. C++ uses OpenMP as API for parallelism.

13.1.4 Test Cases

Four groups of test cases are realized:

- sequential execution on only one core
- parallel execution with two threads on two cores
- parallel execution with four threads on four cores
- parallel execution with eight threads on eight cores

We realize the experiments in each test case by varying the matrix size to analyze performance behavior upon different platforms, overload and variable cache storage requirements.

Speed V and *Speedup S* are measured in each test case. *Relative speed R* of Windows compared to the Linux platform is measured by (13.1), where indexes W and L denote the speed correspondingly Windows and Linux platforms.

$$R = V_W / V_L \tag{13.1}$$

13.2 Experimental Results

This section presents the results of the experiments performed on both platforms to determine speed and speedup in particular platform.

13.2.1 Speed

Fig. 13.1 depicts the results of the experiments on Linux platform. V(1) identifies the curve for speed of sequential execution and V(2), V(4) and V(8) identify speed of parallel execution on Linux platform with 2, 4 and 8 threads correspondingly. As depicted, greater number of threads provides better performance.

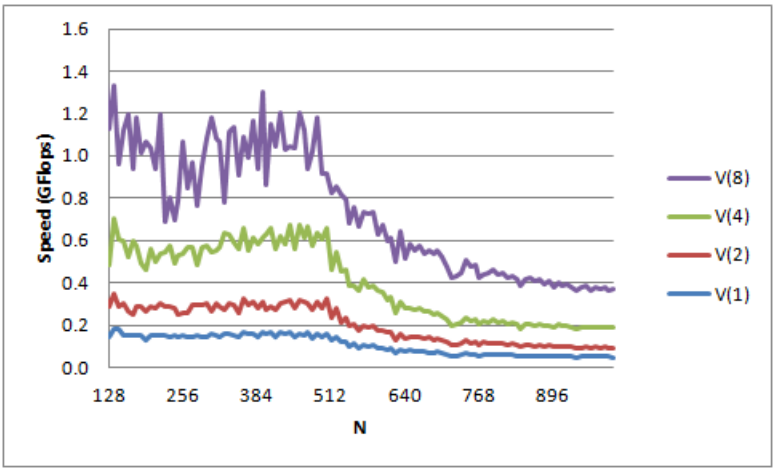

Fig. 13.1 Speed achieved on Linux platform [51]

Fig. 13.2 depicts the results of the experiments on Windows platform. The curves are identified similar as the speedup curves of Linux platform. As depicted, usage of greater number of threads provides better performance. The speed curves are similar as Linux's.

Greater problem size fulfills the caches faster, and therefore generates more cache misses and degrades the performance in both the Linux and Windows platforms.

13.2.2 Speedup

Speedup defined in relation (2.9) is calculated for each test case.

Fig. 13.3 depicts the results of the experiments on Linux platform. S(2), S(4) and S(8) identify speedup of parallel execution on Linux platform with 2, 4 and 8 threads correspondingly. As depicted, speedup satisfies Gustafson's law for scaled speedup.

Fig. 13.4 depicts the results of the experiments on Windows platform. The curves are identified similar as the speedup curves of Linux platform. As depicted, there are

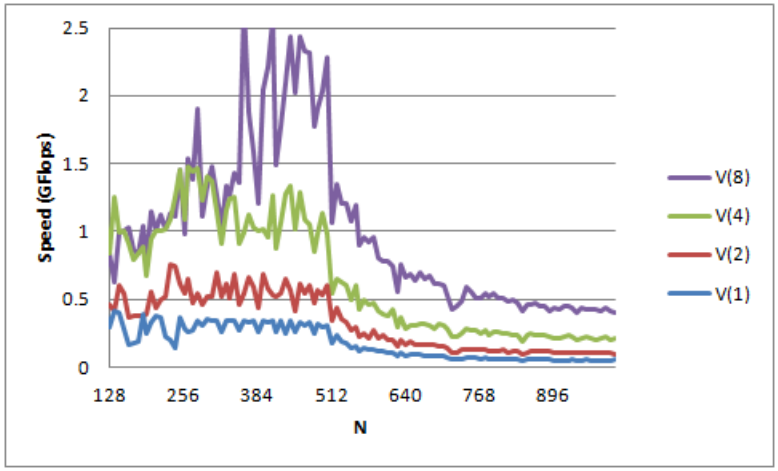

Fig. 13.2 Speed achieved on Windows platform [51]

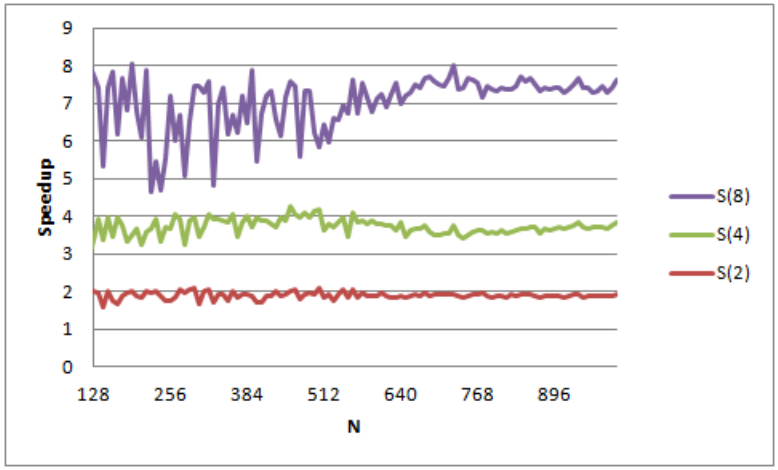

Fig. 13.3 Speedup achieved on Linux platform [51]

superlinear speedup regions for each number of threads. The superlinearity is more emphasis for greater number of threads.

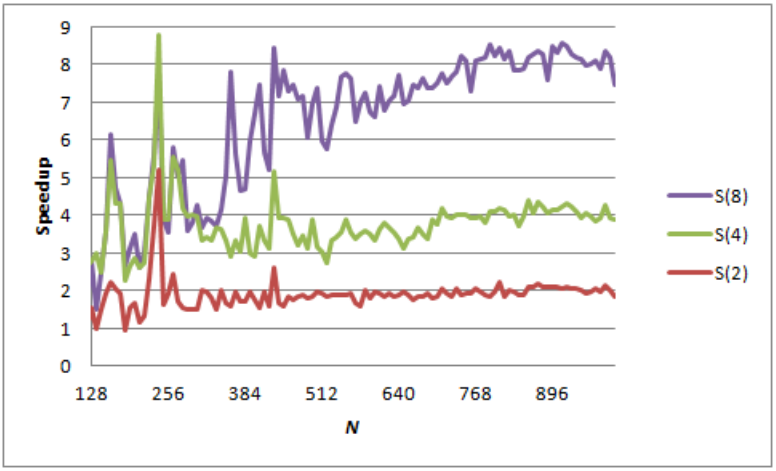

Fig. 13.4 Speedup achieved on Windows platform [51]

13.3 Performance Comparison of Linux and Windows platforms

This section presents the results of the experiments performed on both platforms to determine which platform provides better performance with particular number of threads.

13.3.1 Speed

Fig. 13.5 compares the speeds of both platform for particular number of threads. V(1)L and V(1)W identify the speed curves for sequential execution on Linux and Windows platform correspondingly. Analogue, V(2)L, V(4)L and V(8)L identify the speed curves of parallel execution on Linux platform with 2, 4 and 8 threads correspondingly. V(2)W, V(4)W and V(8)W identify the speed curves of parallel execution on Windows platform with 2, 4 and 8 threads correspondingly. As depicted, Windows platform provides much better performance than Linux for each number of threads and for each problem size N.

Fig. 13.6 depicts the relative performance ratio of Windows compared to Linux platform, calculated according to (13.1). R(1), R(2), R(4) and R(8) identify the ratio of speeds (relative speedup) on Windows and Linux platforms when executing with 1, 2, 4 and 8 threads correspondingly. As depicted, Windows provides better performance than Linux platform, especially in L1-L3 regions, i.e. the regions where the data can be placed in the caches which are dedicated per core. In L4 region, i.e. where data does not fit in cache memory, but can be placed in RAM memory, Windows also provides better performance, but comparable to Linux platform.

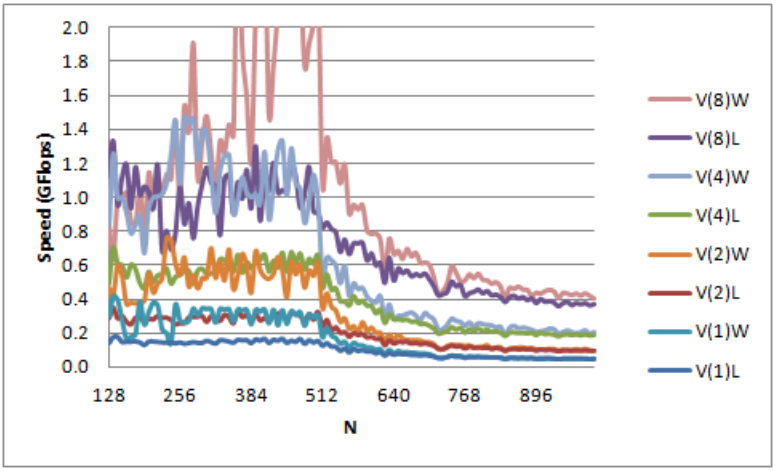

Fig. 13.5 Speed comparison of Linux and Windows platforms [51]

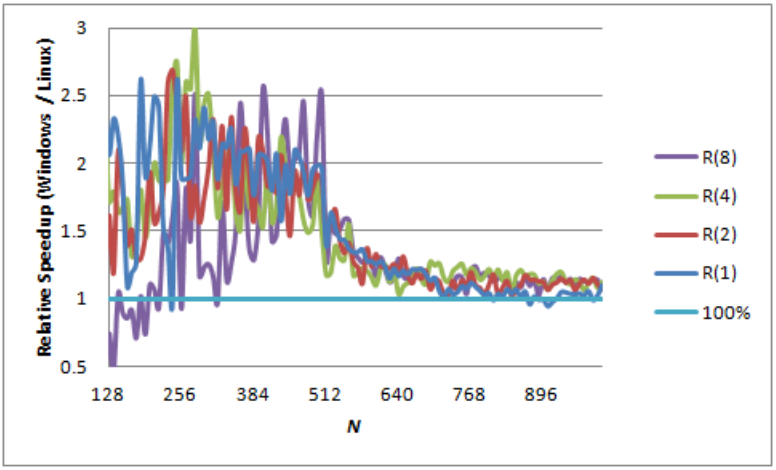

Fig. 13.6 Relative Speed comparison of Windows and Linux platforms [51]

The results of the experiments are discrepant and contrary to the hypothesis show that the Windows platform with its .NET framework 4 and C# runtime environment for parallelization provides better performance than Linux platform with C++ and OpenMP for parallelization.

13.3.2 Speedup

Speedup is determined by (2.9) and calculated for each test case.

Fig. 13.7 compares the speeds of both platform for particular number of threads. S(2)L, S(4)L and S(8)L identify the speedup curves of parallel execution on Linux platform with 2, 4 and 8 threads correspondingly. S(2)W, S(4)W and S(8)W identify the speedup curves of parallel execution on Windows platform with 2, 4 and 8 threads correspondingly. Comparing the curves with the same number of threads for both platforms, we can conclude that Linux provides better speedup for L1-L3 regions. However, Windows provides better speedup (even superlinear) in the L4 region for each number of threads. It means that sequential processing using the Linux platform is slower beyond the expectations.

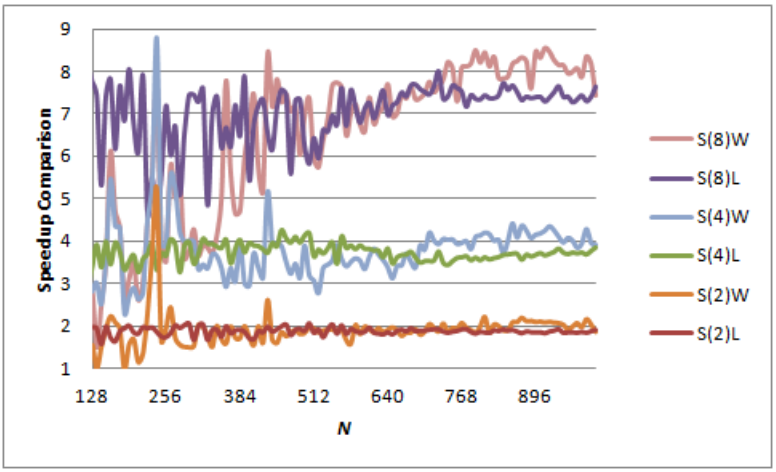

Fig. 13.7 Speedup comparison of Linux and Windows platforms [51]

13.4 Summary

The experimental research in this chapter by the authors in [51] for the purpose of this thesis research shows several conclusions testing execution compute intensive and cache memory demanding algorithm on different platforms in Azure cloud.

Better performance is achieved using Windows than Linux platform. The measured speeds for the same algorithm on Windows is greater than on Linux achieving up to 2.5 times better performance especially in L1-L3 regions. The behavior in L4 region is comparable, but still Windows platform achieves better performance.

The main contribution of this paper is based on experimental proof and recommendation to use the Windows platform while using Azure cloud for cache intensive problems, like dense matrix multiplication algorithms.

Chapter 14
IaaS Performance Impact on Cache Intensive Algorithm

Abstract CSPs offer scalable resources to their customers. The price for renting the resources is linear, i.e. the customer pays exactly double price for double resources. However, not always all offered resources of virtual machine instances are most suitable for the customers. Some problems are memory demanding, others are compute intensive or even cache intensive. The same amount of resources offered by the cloud can be rented and utilized differently to speedup the computation. One way is to use techniques for parallelization on instances with more resources. Other way is to spread the job among several instances of virtual machine with less resources. This chapter describes the results of the experiments realized by the authors in [131] for the purpose of this thesis research. The authors analyze which is the best way to scale the resources to speedup the calculations and obtain best performance for the same amount of money needed to rent those resources in the cloud.

14.1 Testing Methodology

This section describes the testing methodology based on 4 different infrastructures with same platform.

14.1.1 Testing Algorithm

Matrix multiplication algorithm described in Section 4.1 and its parallel implementation from Section 4.3.2 are used as test data.

14.1.2 Testing Environments

Testing environment is hosted in Windows Azure. The authors in [109] presents Windows Azure Platform, its components and architecture in details. We use the same platform environment in each test case with different resource allocation. Windows 2008 Server is used as operating system in each VM instances. Runtime environment consists of C# with .NET framework 4 and threads for parallelization.

We use the same total hardware resources organized in different Windows Azure VMs:

- 1 x Extra Large VM with total 8 CPU cores;
- 2 x Large VM with total 4 CPU cores per VM;
- 4 x Medium VM with total 2 CPU cores per VM;
- 8 x Small VM with total 1 CPU core per VM.

AMD Opteron 4171 HE processor(s) is used in each VM. It has 6 cores, but maximum 4 of 6 cores are dedicated per VM instance. Each core possesses 64 KB L1 data and instruction caches dedicated per core, 512KB L2 dedicated per core. L3 cache with total 5 MB is shared per chip.

14.1.3 Test Cases

By four test cases are performed for sequential and parallel execution of matrix multiplication algorithm. We realize the experiments in each test case by varying the matrix size to analyze performance behavior upon different VM resources, overload and variable cache storage requirements. The following test cases are realized for sequential and parallel execution:

- **Test Case 1: 1 VM with 1 process with 8 (max) threads per process on total 8 cores:** In this test case one Windows Azure Extra Large VM is activated allocated with 8 cores as depicted in Figure 14.1 a). One process in VM executes matrix multiplication with 8 parallel threads. Each thread runs on one core multiplying a row block of matrix $A_{N \cdot N/8}$ and the whole matrix $B_{N \cdot N}$.
- **Test Case 2: 2 concurrent VMs with 1 process per VM with 4 threads per process on total 8 cores**: In this test case two concurrent Windows Azure Large VMs are activated allocated with 4 cores per VM as depicted in Figure 14.1 b). One process in each VM executes matrix multiplication concurrently with 4 parallel threads per process (VM). Each process (in separate VM) multiplies the half of matrix $A_{N \cdot N}$ divided horizontally, i.e. a row matrix $A_{N \cdot N/2}$ and the whole matrix $B_{N \cdot N}$. Each thread multiplies a quarter of half matrix $A_{N \cdot N/2}$, i.e. $A_{N \cdot N/8}$ and the whole matrix $B_{N \cdot N}$.
- **Test Case 3: 4 concurrent VMs with 1 process per VM with 2 threads per process on total 8 cores**: In this test case four concurrent Windows Azure

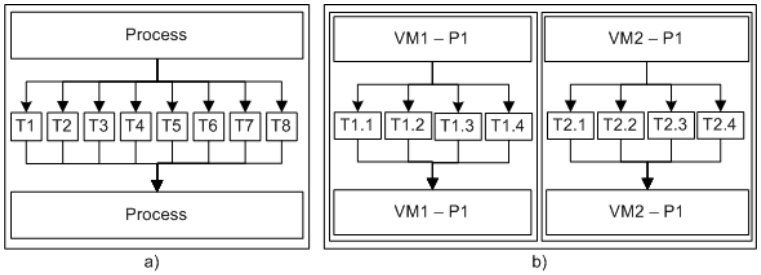

Fig. 14.1 Test Cases 1 (a) and 2 (b) [131]

Medium VMs are activated allocated with 2 cores per VM as depicted in Figure 14.2 a). One process in each VM executes matrix multiplication concurrently with 2 parallel threads per process (VM). Each process (in separate VM) multiplies the quarter of matrix $A_{N \cdot N/4}$ divided horizontally and the whole matrix $B_{N \cdot N}$. Each thread multiplies a half of quarter of matrix $A_{N \cdot N/4}$, i.e. $A_{N \cdot N/8}$ and the whole matrix $B_{N \cdot N}$.

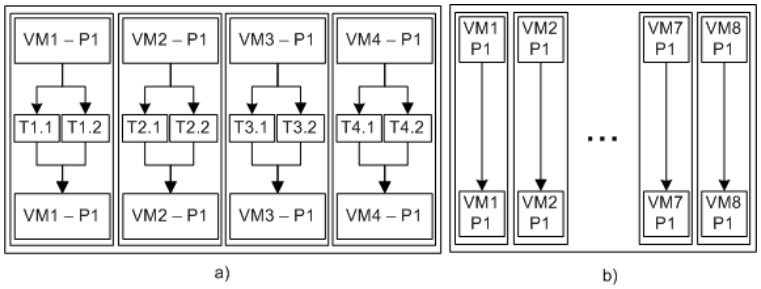

Fig. 14.2 Test Cases 3 (a) and 4 (b) [131]

- **Test Case 4: 8 concurrent VMs with 1 process per VM with 1 thread per process on total 8 cores**: In this test case eight concurrent Windows Azure Small VMs are activated allocated with 1 core per VM as depicted in Figure 14.2 b). One process in each VM executes matrix multiplication concurrently with 2 parallel threads per process (VM). Each process (in separate VM) multiplies the quarter of matrix $A_{N \cdot N/4}$ divided horizontally and the whole matrix $B_{N \cdot N}$. Each thread multiplies a half of quarter of matrix $A_{N \cdot N/4}$, i.e. $A_{N \cdot N/8}$ and the whole matrix $B_{N \cdot N}$.
- **Test Cases 5-8: sequential execution on only one core**: Test cases 5-8 execute matrix multiplication sequentially on the testing environments as test cases 1-4 correspondingly. Only one core is used in each of these test cases and all other seven cores are unused and free. The process runs on one core multiplying the whole matrix $A_{N \cdot N}$ with the whole matrix $B_{N \cdot N}$.

14.1.4 Test Data

Speed V and *Speedup S* are measured for each test case. Average execution time of all processes per test case is measured.

Additionally we measure *relative speed* R_i for sequential and parallel test cases. The relation (14.1) defines the relative speedup of sequential execution in smaller VMs compared to the Extra Large, i.e. test cases 6, 7 and 8 compared to test case 5. Analogue, the relation (14.2) defines the relative speedup of parallel execution in smaller VMs compared to the Extra Large, i.e. test cases 2, 3 and 4 compared to test case 1. The index i denotes the corresponding test case.

$$R_{iSeq} = V_i/V_5 \tag{14.1}$$

$$R_{iPar} = V_i/V_1 \tag{14.2}$$

14.1.5 Tests Goal

The test experiments have two goals:

- To determine the speedup that particular infrastructure provides; and
- To determine which hardware resource allocation among tenants and threads provides best performance for HPC application in Windows Azure.

Different sets of experiments are performed by varying the matrix size changing the processor workload and cache occupancy in the matrix multiplication algorithm.

14.2 The Results of the Experiments

This section presents the results of the experiments that run test cases. We measure the average speed and speedup for each test case and analyze their dependencies of different hardware resource allocation, that is, we compare the results of test cases 1 and 5, 2 and 6, 3 and 7, and 4 and 8 as described in Section 16.1.3.

14.2.1 Test Cases 1 and 5

Figure 14.3 depicts the speed of test cases 1 and 5 for different matrix size N. We determine two main regions with different performance. For $N < 572$ (L_3 region) the whole matrices can be placed in L3 cache and performance are much better than for $N > 572$ (L_4 region) where L3 cache misses are generated.

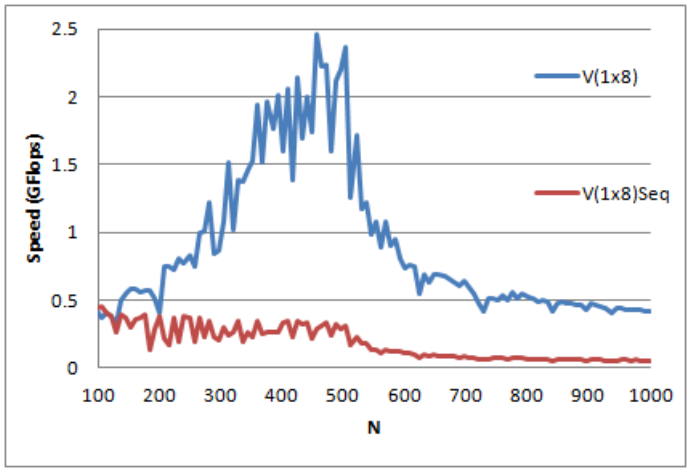

Fig. 14.3 Speed for test cases 1 and 5 [131]

The same regions are depicted for speedup depicted in Figure 14.4. A superlinear speedup is determined in some points of L_4 region due to doubled size L3 cache for parallel execution than sequential.

14.2.2 Test Cases 2 and 6

Figure 14.5 depicts the speed of test cases 2 and 6 for different matrix size N. The same regions with different performance are found also.

Figure 14.6 depicts the speedup of test cases 2 and 6 for different matrix size N. We found that the whole L_4 region is superlinear region since only half matrix A is stored in L3 cache for parallel execution rather than the whole matrix A for sequential execution.

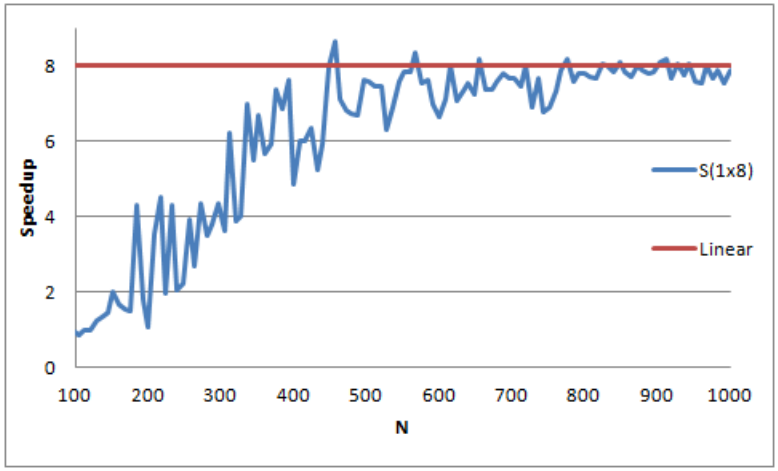

Fig. 14.4 Speedup for test cases 1 and 5 [131]

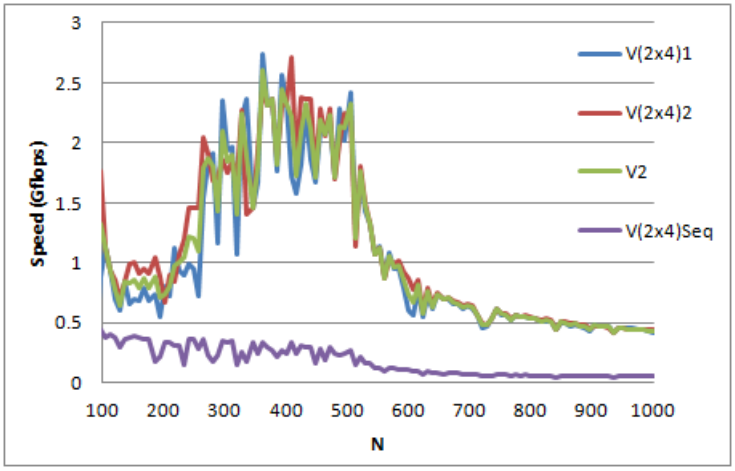

Fig. 14.5 Speed for test cases 2 and 6 [131]

14.2.3 Test Cases 3 and 7

Figure 14.7 depicts the speed of test cases 3 and 7 for different matrix size N. As we can see, there is a huge performance discrepancy in L_3 region among the processes and the average speed.

Figure 14.8 depicts the speedup of test cases 3 and 7 for different matrix size N. We also found that the whole L_4 region is superlinear region. This infrastructure provides even greater speedup than test case 2.

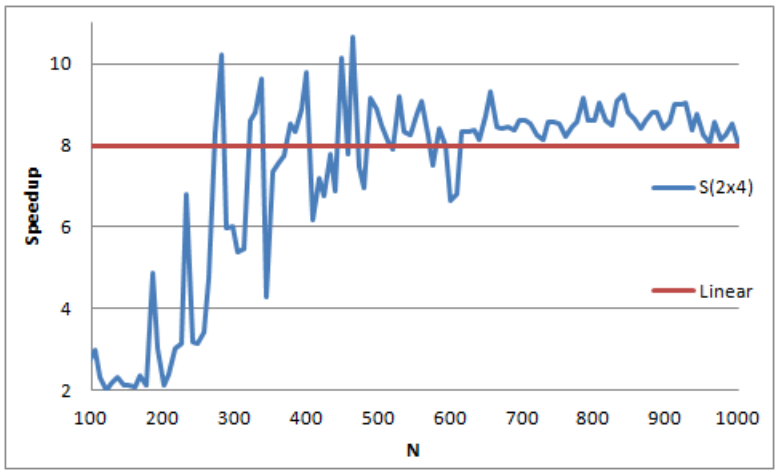

Fig. 14.6 Speedup for test cases 2 and 6 [131]

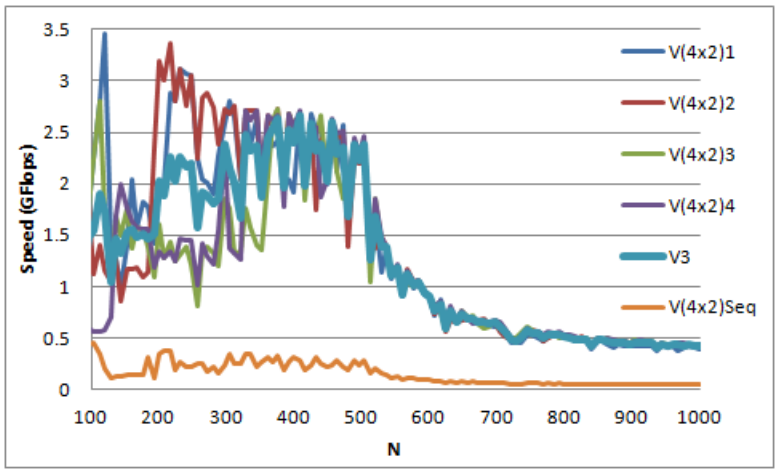

Fig. 14.7 Speed for test cases 3 and 7 [131]

14.2.4 Test Cases 4 and 8

Figure 14.9 depicts the speed of test cases 4 and 8 for different matrix size N. We also found a performance discrepancy but more emphasized in L_1 and L_2 regions which are dedicated per process and thread in test case 4 since each process has only one thread, each VM has only 1 core and L1 and L2 caches are dedicated per that core.

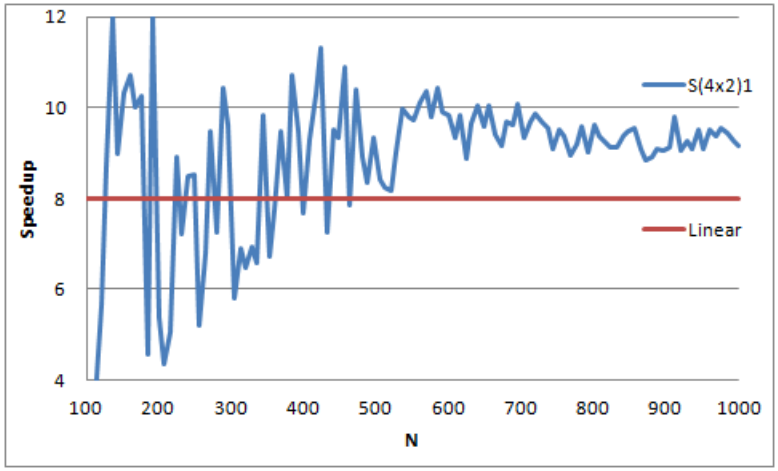

Fig. 14.8 Speedup for test cases 3 and 7 [131]

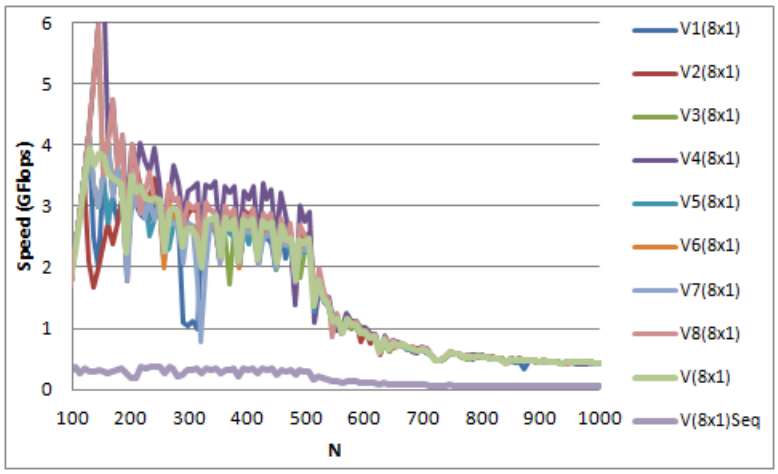

Fig. 14.9 Speed for test cases 4 and 8 [131]

Figure 14.10 depicts the speedup of test cases 4 and 8 for different matrix size N. The important result here is the superlinear speedup in L_2 region since each VM has only one core which has dedicated L1 and L2 cache per core and in this case per VM. Entering the L_3 and L_4 regions multi-tenancy provides more cache misses replacing the blocks that other VMs need from shared L3 cache. Therefore speedup in L_4 region is almost linear although we found superlinear speedup for some N.

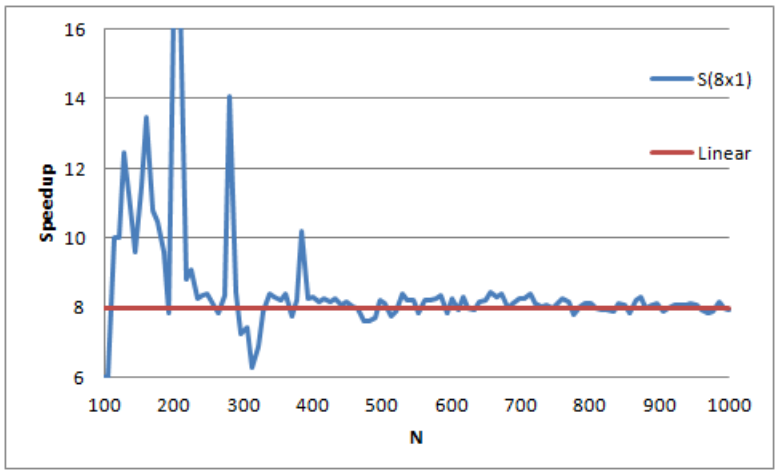

Fig. 14.10 Speedup for test cases 4 and 8 [131]

14.3 Which Hardware Infrastructure Orchestration is Optimal for HPC

This section describes the results of testing the performance impact of hardware infrastructure orchestration. We analyze the results to understand if single-tenant with multi-threading, or multi-tenant with multi-threading or multi-tenant with single-threading the optimal environment to achieve maximum performance for matrix multiplication algorithm, expressed as faster execution and greater speedup.

14.3.1 Hardware Infrastructure impact on Sequential Execution

The performance of sequential execution is measured in test cases 5 to 8. Figure 14.11 depicts the results of test cases 5 to 8 in absolute difference for different matrix size N. We clearly observe three regions with different performance: L_2 with maximum speed in front of L_3 and L_4 region. Speed retains the value with in particular region in each test case.

Figure 14.12 depicts the relative difference of test cases 6 to 8 with test case 5 for different matrix size N.

Relative difference comparison also presents interesting conclusion. We can conclude that relative speeds are stable in L_2 and L_4 regions rather than L_3 region. matrix multiplication algorithm algorithm best runs sequentially on Extra Large in front of Large, Medium and Small in L_2 region. However, Extra Large and Small VMs lead in front of Large and Medium in L_4 region.

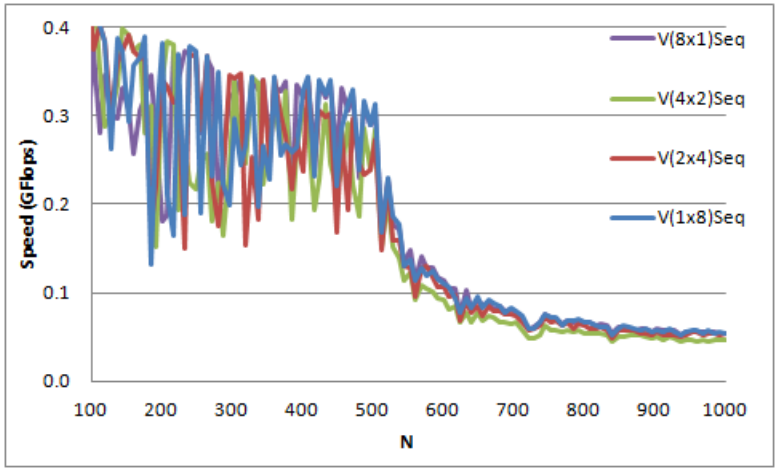

Fig. 14.11 Speed V for sequential execution [131]

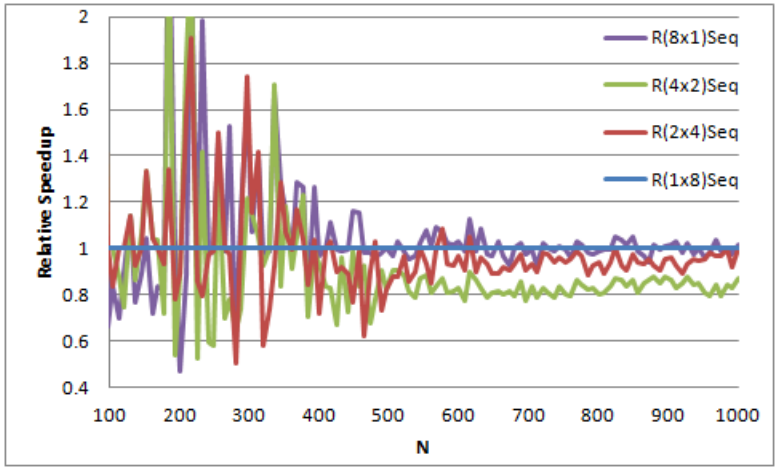

Fig. 14.12 Relative speedup R for sequential execution [131]

14.3.2 Hardware Infrastructure impact on Parallel Execution

The performance of parallel execution is measured in test cases 1 to 4. Figure 14.13 depicts the absolute difference for different matrix size N.

We also observe the same three regions L_2, L_3 and L_4 but with different results. The speed increases in L_2 region for the test cases with multi-threading, i.e. test cases 1, 2 and 3. The speed saturates in L_3 region and also in L_4 region with decreased value for all test cases.

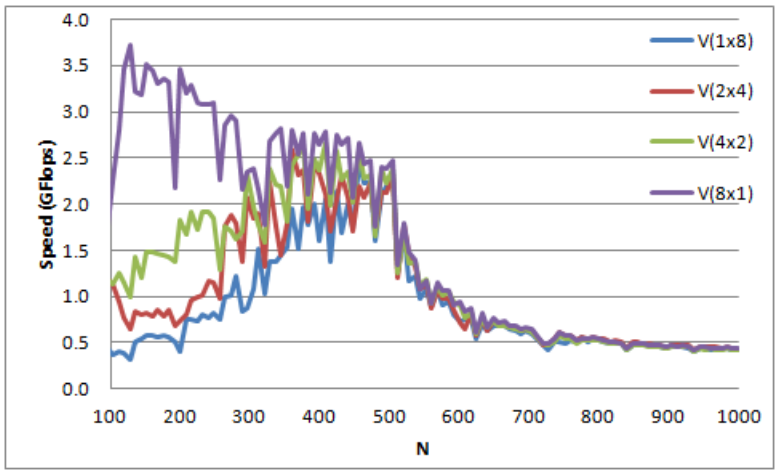

Fig. 14.13 Speed V for parallel execution [131]

Figure 14.14 depicts the relative difference of test cases 2 to 4 with test case 1 for different matrix size N.

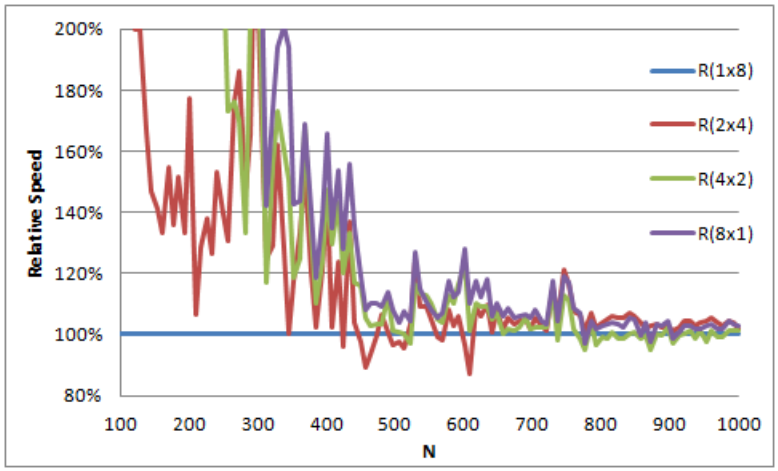

Fig. 14.14 Relative speed R for parallel execution [131]

We can conclude that matrix multiplication algorithm best runs parallel on 8 x Small instances in front of 4 x Medium, 2 x Large, and 1 x Extra Large in L_2 and L_3 regions. The order is retained in L_4 region where for huge matrices all environments provide similar performance and other algorithms should be used.

The speedup achieved for matrix multiplication algorithm is presented in Figure 14.15 for different matrix size N executing on one, two, four and eight VM using total 8 threads on all 8 cores.

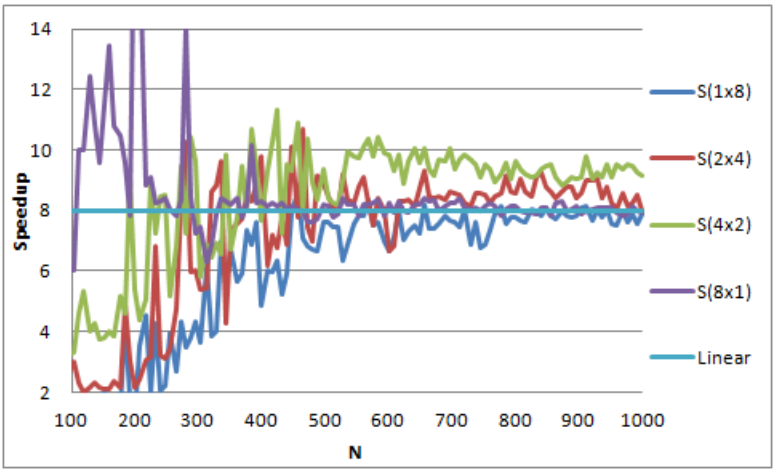

Fig. 14.15 Speedup comparison for test cases 1 to 4 [131]

Analyzing the performance behavior we can conclude that the environment defined by test case 4 is the leader in the speedup race in front of the test cases 3, 2 and 1 in L2 region, and the environment for test case 3 is the leader for the speedup race in front of the test cases 2, 4 and 1 in regions L_3 and L_4.

14.4 Summary

This chapter analyzes the performance of dense matrix multiplication algorithm in Windows Azure Cloud using the same hardware resources but differently spreaded among virtual machines. The experiments are realized by the autors in [131] for the purpose of this thesis research.

The results of the experiments are as expected for sequential execution. As expected, Extra Large VM achieves maximum speed in front of Large, Medium and Small in L2 region. However, Small VM achieves similar speed as Extra Large VM and they lead in front of Large and Medium VMs in L4 region.

Parallel execution provides even more strange results. Dense matrix multiplication algorithm achieves maximum speed when executed parallel on 8 x Small instances, in front of 4 x Medium, 2 x Large, and 1 x Extra Large in L2 and L3 regions, and almost all observed L4 region. This means that the best performance can be achieved if dense matrix multiplication algorithm is granulated on 8 chunks and

each chunk to be executed on 8 concurrent processes with one thread in Small Windows Azure VM. The same environment achieves maximum speedup in L2 region. In L3 and L4 region maximum speedup is achieved if dense matrix multiplication algorithm is granulated on 4 chunks and each chunk to be executed on 4 concurrent processes with two threads in Medium Windows Azure VM.

Chapter 15
Multitenancy Impact on Cache Intensive Algorithm Performance

Abstract Multi-tenant cloud computing enables isolation of tenants in one or more instances of virtual machines and sharing the hardware resources. Since modern multi-core multiprocessors also share the last level cache among all cores on one chip, the goal is to enable an optimal resource allocation by avoiding cache misses as much as possible, since this will lead to performance increase. In this chapter we will describe the results of the experiments published by the authors in [49] for the purpose of this thesis research, i.e. the performance of single and multi-tenant environments in cloud environment installed on a single chip multi core multiprocessor with different resource allocation to the tenants. Although one might think that virtualization and clouds include software overhead, the results show how and when cloud computing can achieve even better performance than traditional environment, both in a single-tenant and multi-tenant resource allocation for certain workload.

15.1 The Workload Environments

This section describes the testing methodology and defines the workload environments for experiments. Matrix multiplication algorithm described in Section 4.1 and its parallel implementation from Section 4.3.3 are used as test data. For all different environments, we use the same hardware and operating system. The only difference is inclusion of VMs and enabling cloud environment.

15.1.1 Traditional On-premise Environment

This environment consists of Linux Ubuntu Server 11.04 installed on Dell Optiplex 760 with 4GB DDR2 RAM and Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz [167]. The multiprocessor has 4 cores, each with 32 KB 8-way set associative L1

cache dedicated per core and 8-way set associative L2 cache with total 6 MB shared by 3MB per two cores.

Fig. 15.1 depicts the three different parallel executions that are defined as test cases 1.1, 1.2 and 1.3 in this environment:

- **Case 1.1: 1 process with 4 (max) threads on total 4 cores**. In this test case the matrix multiplication is executed by one process using 4 parallel threads as presented in Fig. 15.1 a). Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.
- **Case 1.2: 2 different processes with 2 threads per process on total 4 cores**. In this test case two concurrent processes execute matrix multiplication. Each process uses two parallel threads as shown in Fig. 15.1 b). Each process multiplies the whole matrix $A_{N \cdot N}$ and a half of matrix $B_{N \cdot N/2}$ divided vertically. Each thread multiplies matrix $A_{N \cdot N}$ and half of $B_{N \cdot N/2}$, i.e., $B_{N \cdot N/4}$.
- **Case 1.3: 4 different processes with 1 thread per process (sequentially) on total 4 cores**. In this test case 4 concurrent processes execute matrix multiplication as depicted in Fig. 15.1 c). Each process multiplies the whole matrix $A_{N \cdot N}$ and a quarter of matrix $B_{N \cdot N/4}$ divided vertically.

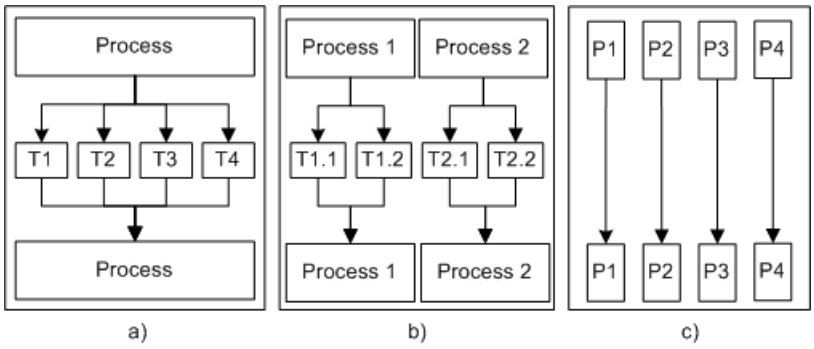

Fig. 15.1 Test Cases in Traditional Environment [49]

15.1.2 Virtual Environment

This environment consists of the same hardware and operating system as described in Section 15.1.1. Additionally new VM is installed with same Linux Ubuntu Server 11.04 using VirtualBox and Kernel-based Virtual Machine virtualization standard (KVM). All available resources (4 cores) are allocated to the only one VM for parallel execution and only one core for sequential execution.

Case 2.1: 1 VM with 1 process with 4 (max) threads on total 4 cores. In this test case one process executes matrix multiplication by 4 parallel threads, all in the VM. Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

15.1.3 Cloud Virtual Environment

Cloud virtual environment is developed using OpenStack Compute project [106] deployed in dual node as depicted in Fig. 15.2. KVM virtualization standard is also used for VMs. One Controller Node and one Compute Node are used.

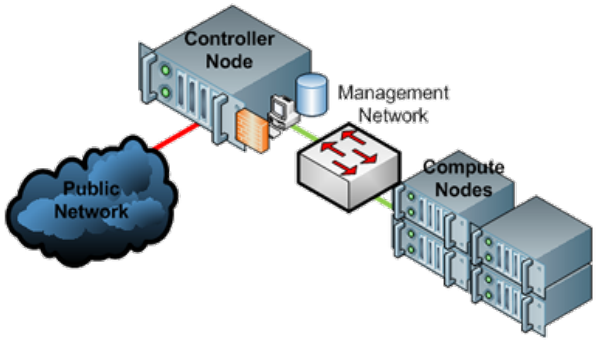

Fig. 15.2 OpenStack dual node deployment [105]

This cloud virtual environment consists of the same hardware and operating system as described in Section 15.1.1 for Compute Node server. Virtual Machine described in Section 15.1.2 is instantiated in one or more instances for the four test cases that are performed in this environment.

Figure 15.3 depicts the three test cases 3.1, 3.2 and 3.3 that are performed as parallel executions in this environment:

- **Case 3.1: 1 instance of VM with 1 process with 4 (max) threads per process on total 4 cores**. This case is similar as cases 1.1 and 2.1, i.e., one instance of VM is activated in the Cloud allocated with all 4 cores as depicted in Fig. 15.3 a). One process in VM executes matrix multiplication with 4 paralllel threads. Each thread runs on one core multiplying the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.
- **Case 3.2: 2 concurrent instances of VM with 1 process per VM with 2 threads per process on total 4 cores**. In this test case two concurrent instances of same VM are activated in the Cloud allocated with 2 cores per instance as depicted in Fig. 15.3 b). One process in each VM executes matrix multiplication

concurrently with 2 parallel threads per process (VM). Each process (in separate VM) multiplies the whole matrix $A_{N \cdot N}$ and a half of matrix $B_{N \cdot N/2}$ divided vertically. Each thread multiplies matrix $A_{N \cdot N}$ and half of $B_{N \cdot N/2}$, i.e., $B_{N \cdot N/4}$.

- **Case 3.3: 4 concurrent instances of VM with 1 process per VM with 1 thread per process (sequentially) on total 4 cores**. In this test case, 4 concurrent instances of same VM are activated in the Cloud allocated with 1 core per instance as depicted in Fig. 15.3 c). Each process (in separate VM) multiplies the whole matrix $A_{N \cdot N}$ and a column block of matrix $B_{N \cdot N/4}$.

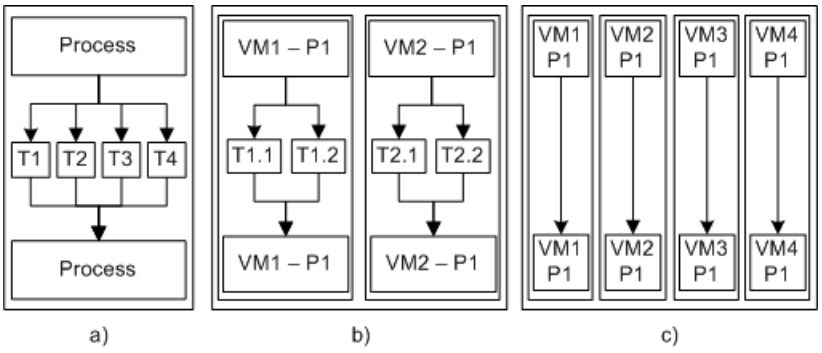

Fig. 15.3 Test Cases in Cloud Virtual Environment [49]

15.1.4 Test Goals

The test experiments have two goals:

- The first goal is to determine if the additional virtualization layer in cloud drawbacks the performances compared to traditional or virtualized operating system when all the resources are dedicated to only one tenant and multi-threading is used.
- The second goal is to determine which resource allocation among tenants and threads provides best performance in the traditional environment and in the cloud.

Different sets of experiments are performed by varying the matrix size changing the processor workload and cache occupancy in the matrix multiplication algorithm.

15.2 Environment Performance Comparison with all Resources Allocated

This Section presents the results of the experiments performed on three workload environments when all the resources (CPU cores) are rented to one tenant, i.e., test cases 1.1, 2.1 and 3.1 as described in Section 15.1.

Fig. 15.4 depicts the speed in gigaflops that matrix multiplication achieves for different matrix size N when executing one process concurrently using 4 threads on 4 cores on three same hardware resources, but different system environments as described in Section 15.1. The curves are identified by V(4)T for traditional environment, V(4)V for environment with virtual and V(4)C with cloud environment. Fig. 15.5 shows only the differences of achieved speeds in Fig. 15.4 using relative presentation of the ratio to the default speed value obtained by traditional environment.

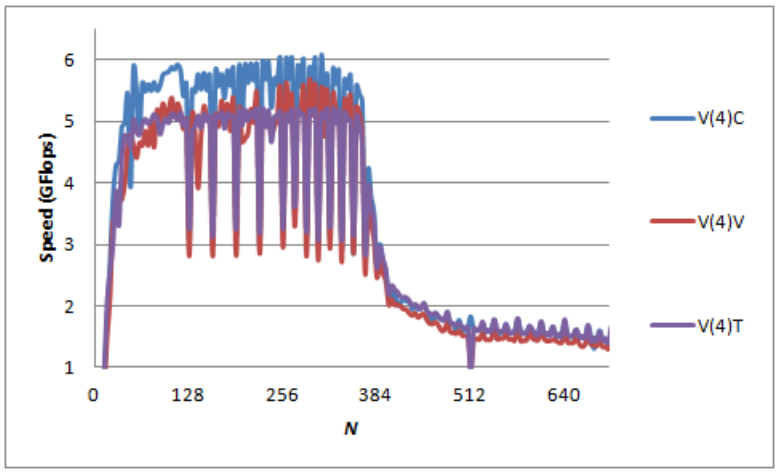

Fig. 15.4 Speed comparison for traditional / virtual machine allocated with all hardware resources (4 threads) [49]

Two regions with different performance for all three test cases are clearly depicted in Fig. 15.4; the left one with higher speed and the right one with lower speed. The first region is the L2 region as defined in [48] (the region for such matrix size N that will enable storage of all memory requirements in L2 cache and avoid generation of cache misses for reusing the same data on L2 level). The second region is the region where the matrices can not be stored completely in the L2 cache and many L2 cache misses will be generated due to re-using of data, but memory requirements will fit in the L3 cache (if it exists). This region is called the L_3 region. We must note that those matrices that fit in L_1 region are too small to produce higher speed.

Analyzing the performance by comparing the three curves in figures 15.4 and 15.5, we can conclude that cloud virtualization performs the algorithm better than other two environments in the L_2 region. Virtualization also performs better than traditional environment in the same L_2 region, but produces worse performance in points where performance drawbacks appear due to cache set associativity described in [120]. Cloud and traditional environments provide similar performance in L_3 region, i.e., shared main memory, much better than virtual environment. The conclusion is that in this region virtualization provides the worst performance and cloud environment achieves the best performance.

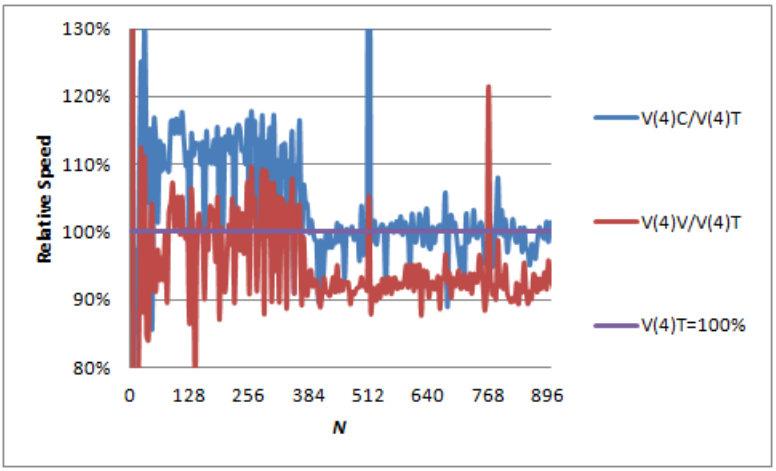

Fig. 15.5 Relative speed comparison for Fig. 15.4 [49]

Another important conclusion is the fact that the speed increases in the L2 region where the cache memory is dedicated per core (group of 2 cores) for virtual and cloud environments. However, the speed decreases in the shared memory L3 region when matrix size N increases demanding more memory requirements, generating higher cache miss penalty and increasing the overall memory access time.

Based on results of these experiments, we can conclude that cloud virtual environment achieves better performance compared to traditional environment for cache intensive algorithms in the L2 region using dedicated L2 cache per core and shared L3 cache and main memory. The authors in [49] describes the causes for this phenomenon.

15.3 Multiprocess, Multithread and Multitenant Performance

This section presents the results of the experiments that run test cases 3.1, 3.2 and 3.3 described in Section 15.1 with different resource allocation per tenant in cloud virtual environment.

The speed achieved for the matrix multiplication algorithm is presented in Fig. 15.6 for different matrix size N of the matrix multiplication executing on one, two and four VM using total 4 threads on all 4 cores on the same cloud virtual environment. The curves are identified by V(4)C for test case 3.1, V(2x2)C for test case 3.2 and V(4x1)C for test case 3.3. The relative differences to the default speed V(4)C are presented in Fig. 15.7.

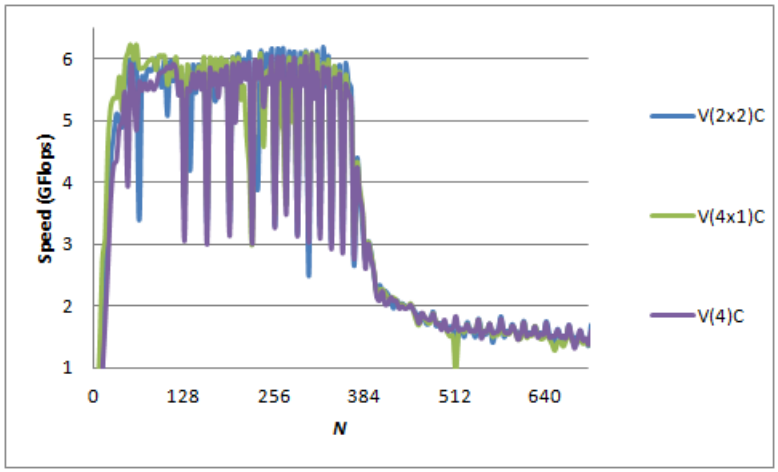

Fig. 15.6 Speed comparison for virtual machine(s) in cloud allocated with different resources per machine and per thread [49]

Fig. 15.6 presents that the same two regions L2 and L3 can be identified by different performance for all 3 test cases.

Analyzing the performance behavior presented in figures 15.6 and 15.7 we can conclude that the environment defined by test case 3.3 is the leader in the speed race in front of the test cases 3.2 and 3.1 for the left part of the L_2 region, and the environment for test case 3.2 is the leader for the speed race in front of the test cases 3.3 and 3.1 in the right part of the L2 region. All test cases provide similar performance in the L3 region with test 3.1 as a leader.

We can also conclude that the speed increases in the L_2 region where cache memory is dedicated per core (group of 2 cores) for all three test cases. However, the speed decreases for all test cases in the shared memory L_3 region when the matrix size N is increased enough and higher cache miss penalty is generated increasing the overall memory access time.

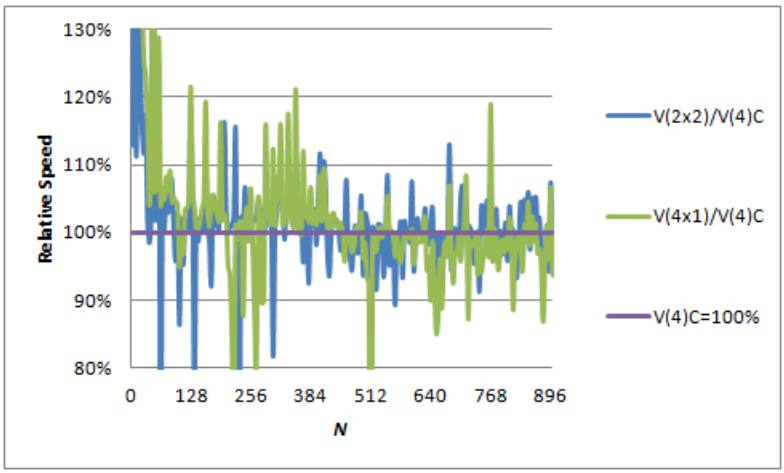

Fig. 15.7 Relative speed comparison for Fig. 15.6 [49]

Dividing the problem in separate concurrent VMs is the best solution for cache intensive algorithms in the L_2 region for dedicated L2 caches. The best solution for the L_3 region with shared main memory is to allocate all the resources to one process (VM) to be executed concurrently with maximum threads as number of cores.

15.4 Summary

Several experiments including parallel executions are performed with different resource allocation in traditional, virtual and cloud environments on the same multiprocessor. The testing methodology addresses each environment with full utilization to all CPU cores with different techniques: mono-process with multi-threading, multi-processes with multi-threading and multi-processes with single threads.

The performed experiments address several virtual machine instances in a cloud system using different number of CPUs (assuming all cores are utilized). Each experiment orchestrates the CPU cores differently. The contribution of the paper can be summarized as:

- The experiments prove that there is a region (L_2 region) where cloud environment achieves better performance than traditional and virtual environment, both for parallel and sequential process execution, and
- The experiments prove that cloud computing provides better performance in a multi-VM environment, rather than allocating all the resources to only one VM.

The best resource allocation for traditional environment for cache intensive algorithms is the usage of multiple processes with single threads. Multiple VMs with

single threads is the best resource allocation for cloud environment. Comparing the environments, cloud computing provides the best performance.

Chapter 16
Superlinear Speedup in Cloud Virtual Environment

Abstract CPU cache is used to speedup the execution of memory intensive algorithms. Usage of greater cache memory sizes reduces the cache misses and overall execution time. Different cache occupancy for sequential and parallel execution can lead to superlinear speedup. In this chapter we will describe the results of the experiments published by the authors in [130] for the purpose of this thesis research. The testing methodology and experiments for this research are applied also to cloud environment. The results show that cloud environment can also achieve superlinear speedup for execution of cache intensive algorithms when high performance computing is used in virtual machines allocated with more than one processor (core).

16.1 Testing Methodology

This section describes the testing methodology based on 3 different environments and 3 test cases for each environment.

16.1.1 Testing Algorithm

Matrix multiplication algorithm described in Section 4.1 and its parallel implementation from Section 4.3.3 are used as test data.

16.1.2 Testing Environments

Three different platforms are analyzed as testing environments using the same runtime environment hosted in the same hardware infrastructure as described in Section 15.1.

16.1.3 Test Cases

Three groups of test cases are realized with different resource allocation:

- sequential execution with one thread on one core
- parallel execution with two threads on two cores
- parallel execution with four threads on four cores

We realize a series of experiments in each test case by varying the matrix size to analyze performance behavior upon different overload and variable cache storage requirements.

16.1.4 Test Goals

Our plan is to provide answers to the following hypotheses:

- is there a superlinear region for virtual and cloud environment, and
- is there a region where cloud environment achieves better speedup than traditional and virtual environment

16.2 Experimental Results

This section presents the results of the experiments performed on three environments to determine if superlinear speedup can be achieved for parallel execution with two and four threads compared to sequential execution.

The results prove superlinear speedup region existence in all three platforms for parallel execution with 2 and 4 threads.

16.2.1 Speedup Analysis in Traditional Environment

Figure 16.1 depicts the speedup for parallel execution with two and four threads in traditional environment. The curves are identified by S(4)T for parallel execution with four threads and S(2)T for parallel execution with two threads. Note that there are performance drawbacks due to usage of set associative cache memory as explained in [120].

The obtained results prove our hypothesis of superlinear existence in the regions $424 \leq N \leq 992$ for two threads and $504 \leq N \leq 872$ for four threads. The experiments also prove that the speedup increases until $N = 628$ for both parallel execution and then starts to drop down.

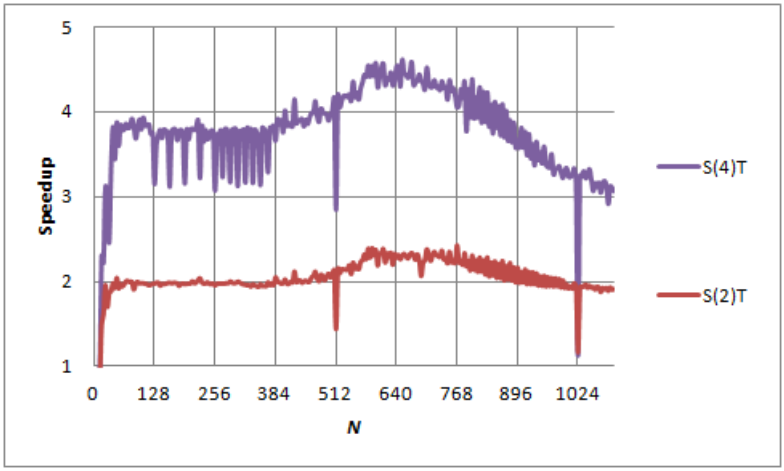

Fig. 16.1 Speedup in traditional environment [130]

16.2.2 Speedup Analysis in Virtual Environment

Figure 16.2 depicts the speedup for parallel execution with two and four threads in virtual environment. The curves are identified by S(4)V for parallel execution with four threads and S(2)V for parallel execution with two threads.

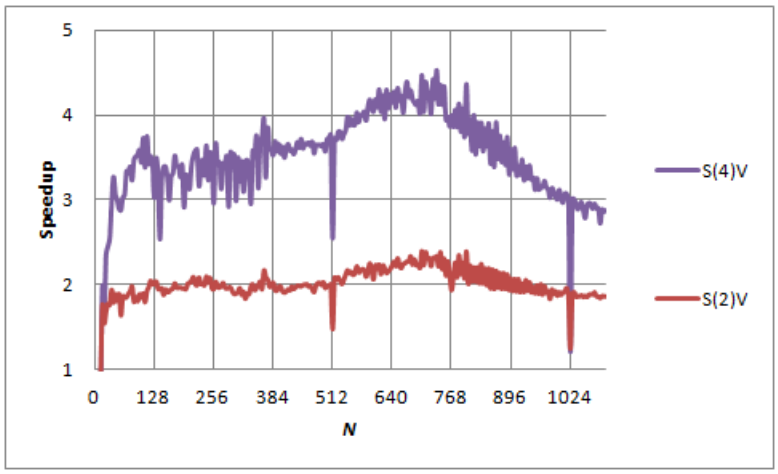

Fig. 16.2 Speedup in virtual environment [130]

The obtained results also prove our hypothesis of superlinear existence in the regions $508 \leq N \leq 936$ for two threads and $564 \leq N \leq 752$ for four threads. The speedup increases until $N = 704$ for both parallel execution and then starts to drop down.

16.2.3 Speedup Analysis in Cloud Environment

Figure 16.3 depicts the speedup for parallel execution with two and four threads in cloud environment. The curves are identified by S(4)V for parallel execution with four threads and S(2)V for parallel execution with two threads.

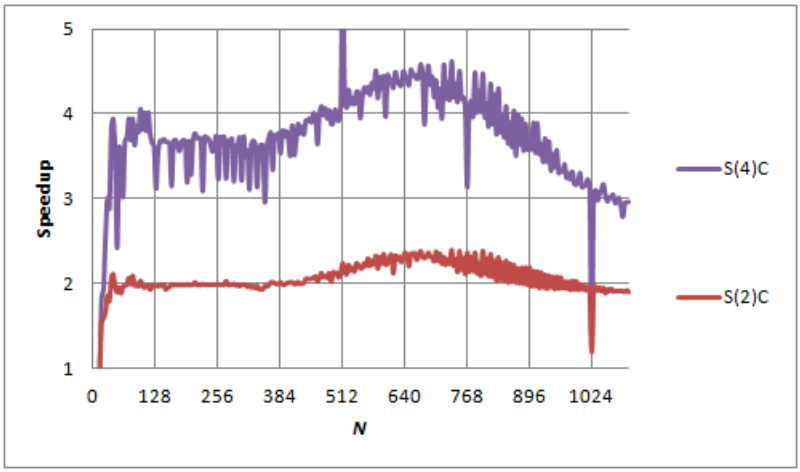

Fig. 16.3 Speedup in cloud environment [130]

The results also prove the existence of superlinear region $420 \leq N \leq 992$ for two threads and $508 \leq N \leq 848$ for four threads. The speedup increases until $N = 672$ for both parallel execution and then starts to drop down.

16.2.4 Speedup Comparison for two threads

Figure 16.4 depicts the speedup for parallel execution with two threads in all three environments. The curves are identified by S(2)T for speedup in traditional environment, S(2)V for speedup in virtual environment and S(2)C for speedup in cloud environment for parallel execution with two threads. Figure 16.4 presents that su-

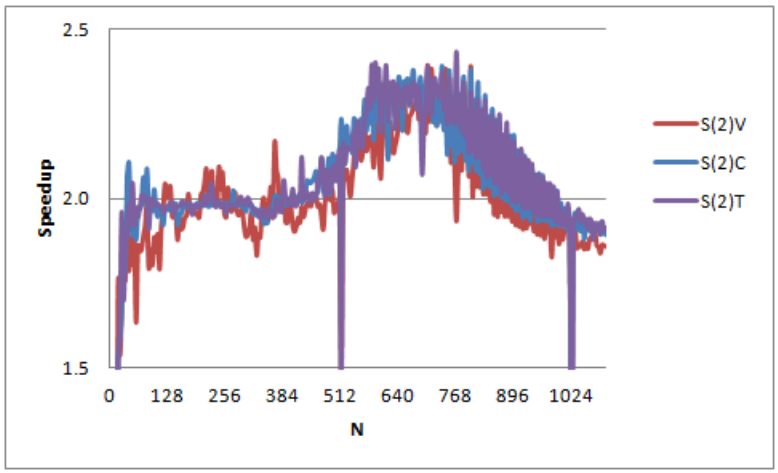

Fig. 16.4 Speedup comparison for two threads [130]

perlinear speedup is achieved in each environment for parallel execution with two threads.

Figure 16.5 shows the relative speedup expressed as percentage via ratio of achieved speedups in virtual and cloud environments to traditional environment for parallel execution with two threads. It is assumed that the speedup in traditional environment is 100%.

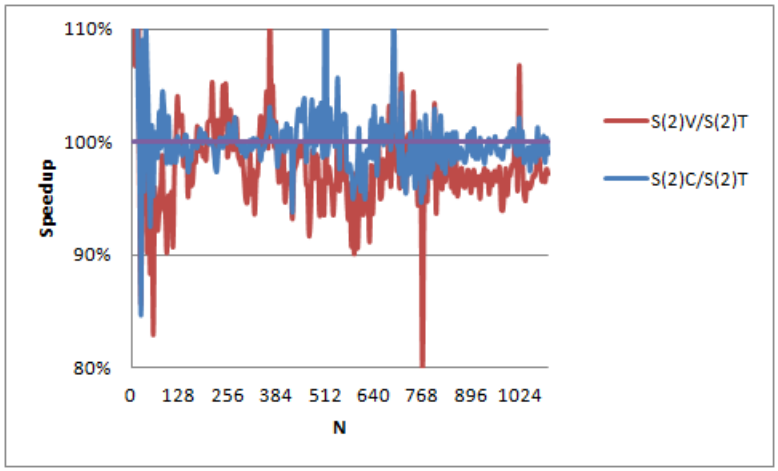

Fig. 16.5 Relative speedup comparison for Figure 16.4 [130]

Comparing the curves in Figure 16.5 we can conclude that cloud virtualization performs better than traditional environment until L2 starts to generate cache misses when the traditional environment achieves better speedup. Virtual environment achieves the best speedup for small N but the performance is worse than other two environments when L2 cache misses start to generate.

Table 16.1 presents the analytical comparison for characteristic environment values. We can conclude that traditional and virtual environment have wider super-

Parameter	Tradit.	Virtual	Cloud
Superlinear region	[424, 992]	[508, 936]	[420, 992]
Max. speedup S_{max}	2.34	2.39	2.37
N_{max} (S_{max} point)	628	704	672

Table 16.1 Environment Comparison for two threads [130]

linear speedup region than virtual, and virtual environment achieves the maximum speedup $S(2)_{max}$.

16.2.5 Speedup Comparison for four threads

Figure 16.6 presents the speedup for parallel execution with four threads in all three environments. The curves are identified by S(4)T for speedup in traditional environment, S(4)V for speedup in virtual environment and S(4)C for speedup in cloud environment for parallel execution with four threads.

Superlinear speedup is achieved in each environment for parallel execution with four threads, as shown in Figure 16.6.

Figure 16.7 presents the relative speedup expressed as percentage via ratio of achieved speedups in virtual and cloud environments to traditional environment for parallel execution with four threads. It is assumed that the speedup in traditional environment is 100%.

Analyzing the curves in Figure 16.7 we can conclude that the virtual environment achieves smaller speedup than other environments. Cloud virtualization achieves the best speedup until L2 cache misses start to generate when the traditional environment performs better.

Table 16.2 presents the analytical comparison for characteristic environment values. We can conclude that the widest superlinear speedup region is achieved for traditional environment. Cloud environment achieves the maximum speedup value $S(2)_{max}$.

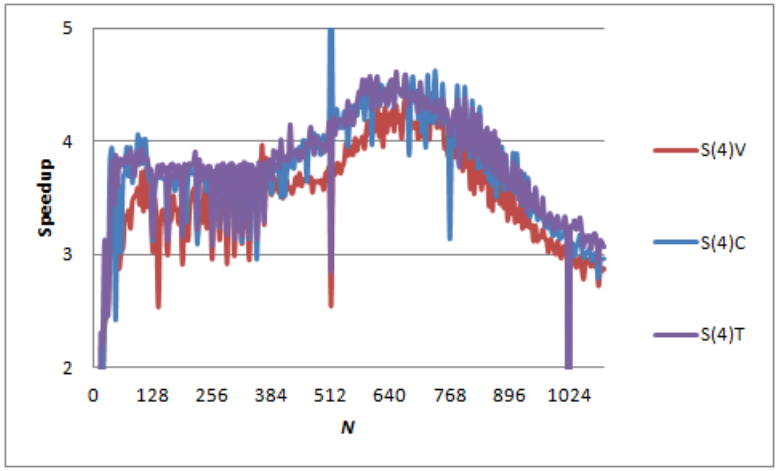

Fig. 16.6 Speedup comparison for 4 threads [130]

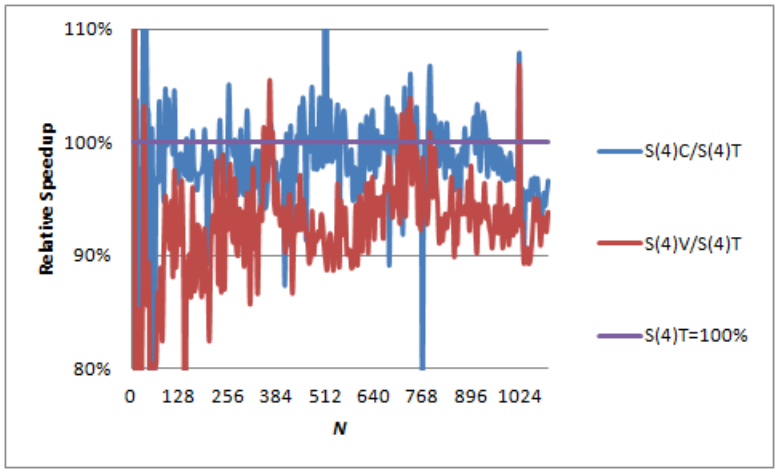

Fig. 16.7 Relative comparison for Figure 16.6 [130]

16.3 Summary

The experiments presented in this chapter are published by the authors in [130] for the purpose of this thais research prove our hypotheses. The speedup begins to increase for those matrices A and B that do not fit in available L2 cache for sequential execution, i.e. half of L2 total cache, but in the same time fit in the whole L2 cache which is available for parallel execution with two or four threads on two or four cores, correspondingly. The speedup increases until $N = 628$ determined theoretically for traditional environment when L2 cache misses begin to appear. There are also a speedup turnover points for virtual and cloud environment greater

Parameter	Tradit.	Virtual	Cloud
Superlinear region	[504, 872]	[564, 752]	[508, 848]
Max. speedup S_{max}	4.62	4.54	4.63
N_{max} (S_{max} point)	628	704	672

Table 16.2 Environment Comparison for four threads [130]

than theoretical value since virtualization provides better performance for parallel execution rather than sequential in shared memory [48].

Our second hypothesis is also proved, i.e. virtual and cloud environments achieve better speedup for dedicated cache and the best performance is achieved by the cloud environment. After the L2 region, where the L2 cache misses are generated, the traditional environment performs better in comparison to the cloud environment. Virtual environment achieves the worst speedup when the algorithm requires a lot of accesses to shared main memory.

We prove that **superlinear speedup is possible for cache intensive algorithm** even in the virtualized and cloud environment.

The experiments show different speedup range. The widest superlinear speedup range is present at traditional environment, while the thinest is found at the virtual one. Cloud and virtual environments have wider superlinear speedup range for parallel execution with two rather than four threads because the last level cache is dedicated per core which is the case where virtualization provides better performance then shared memory [48]. The range is shortened up to 3 times from the right size (for great values of N) compared to the left region (smaller values of N). The range in traditional environment shortens 80 from the left side and 120 from the right side of the range. In virtual environment it shortens 56 and 184, and in cloud environment 88 and 144 for left and right side correspondingly. Virtual environment's superlinear range is the most shortened while the superlinear speedup region in traditional environment shortens the least.

Part V
Performance Analysis of Web Services in Cloud Computing

Chapter 17
Web Service Performance in the Cloud

Abstract Additional layer that virtualization adds in the cloud decreases the performance of the web services. The goal is to test the performance of compute and memory intensive web services on both on-premises and cloud environments. In this chapter we will describe the results of the experiments realized by the authors in [138] for the purpose of this thesis research. Series of experiments are realized to analyze the web services performance and compare what is the level of degradation if the web services are migrating from on-premises to cloud using the same hardware resources. The results show that there is a performance degradation on cloud for each test performed varying the server load by changing the message size and the number of concurrent messages. The cloud decreases the performance to 71.10% of on-premise for memory demand and to 73.86% for both memory demand and compute intensive web services. The cloud achieves smaller performance degradation for greater message sizes using the memory demand web service, and also for greater message sizes and smaller number of concurrent messages for both memory demand and compute intensive web services.

17.1 The Testing Methodology

This section describes testing methodology including identification of environment, infrastructure and platform, test plan and design implementation details. Several steps were performed to create efficient and effective tests and results.

17.1.1 Test Environment Identification

The experiments are realized on traditional client-server architecture on the same hardware infrastructure but different platform. Two same web servers are used as hardware infrastructure with Intel(R) Xeon(R) CPU X5647 @ 2.93GHz with 4 cores

and 8GB RAM. The other server with the same hardware infrastructure is used as a client. Linux Ubuntu 64 bit Server 11.04 is installed on the machines on both the server and the client side. Apache Tomcat 6.0 is used as web server where RPC style web services are being deployed. SOAPUI [143] is used to create various server load tests. Client and server are in the same LAN segment to exclude the network impact shown in [82].

Two different platforms are deployed. *On-premise* platform environment consists of traditional Linux operating system installed as host. *Cloud environment* is developed using OpenStack Compute project deployed in dual node [105]. We use one Controller Node and one Compute Node. KVM virtualization standard is used for instancing virtual machine. The cloud consists of the same hardware and operating system as previously described.

17.1.2 Performance Criteria Identification

We measure response time for various experiments with different number and sizes of concurrent requests for both platforms. Client is on the same VLAN as the web server, with network response time smaller than 1 *ms*, and none of the packets are lost during the test. This means that we can assume that the response time measured with SOAPUI is the same as the server response time.

17.1.3 Test Data

The basic goal is to measure the performance drawbacks caused by migration of web services in the cloud.

Test data consists of Concat and Sort web services. The *Concat web service* accepts two string parameters and returns a string which is concatenation of the input. This is a memory demand web service that depends on the input parameter size M with complexity $O(M)$. The *Sort web service* also accepts two string parameters and returns a string that is concatenation of the two input strings which is then alphabetically sorted using sort function in [14]. This is also a memory demand service that depends on the input parameter size M. In addition it is a computational intensive web service with complexity $O(M \cdot log_2 M)$.

Experiments are repeated for parameter sizes M that change values from $256B$, $768B$, $1280B$, $1792B$, $2304B$ to $2816B$. The generated SOAP messages have the following sizes $786B$, $1810B$, $2834B$, $3858B$, $4882B$ and $5908B$ correspondingly. The server is loaded with various number of messages (requests) N in order to retain server normal workload mode, that is, 500, 1000, 1500 and 2000 requests per second for each message size.

17.1.4 Test Plan

The first part of the experiment consists of series of test cases that examine the impact of increasing the message size to the server response time. The second part of the experiment consists of series of test cases that examine the impact of increasing the number of concurrent messages to the server response time. All test cases are performed on: 1) web services hosted on-premise; and 2) web services hosted in the cloud.

Each test case runs for 60 seconds, N messages are sent with M bytes each, with variance 0.5. The accent is put on server response time in regular mode, and neither burst nor overload mode.

We expect that response time will be increased while increasing the number of messages and their size. We would like to determine which parameter impacts the server performance most? Is it the number of concurrent messages or message sizes and has the platform any influence if it is on-premise or in the cloud?

Monitors are checked before each test. All server performance parameters are examined if their status is returned to nominal state after execution of each test. If not, the server is restarted and returned into it's nominal state. Network latency is measured to ensure proper response time results during the tests.

17.2 The Results and Analysis

This section describes the results of testing the performance impact of cloud virtualization layer. We also analyze the results to understand the performance impact of different message sizes and number of concurrent messages on both web services described in 17.1.3.

17.2.1 Web Service Performance Hosted On-premise

The performance of web services is measured while hosted on-premise with different payload: 1) different message size for constant number of concurrent messages and 2) different number of concurrent messages for a constant message size.

Figure 17.1 depicts the response time of Concat web service hosted on-premise. We can conclude that both input factors are important for Concat web service performance, i.e. response time increases when message size or number of concurrent messages increase.

Response time of Sort web service hosted on-premise is presented in Figure 17.2. We can conclude that only input factor message size is important for Sort web service performance. That is, response time increases only if message size increases regardless of number of concurrent messages.

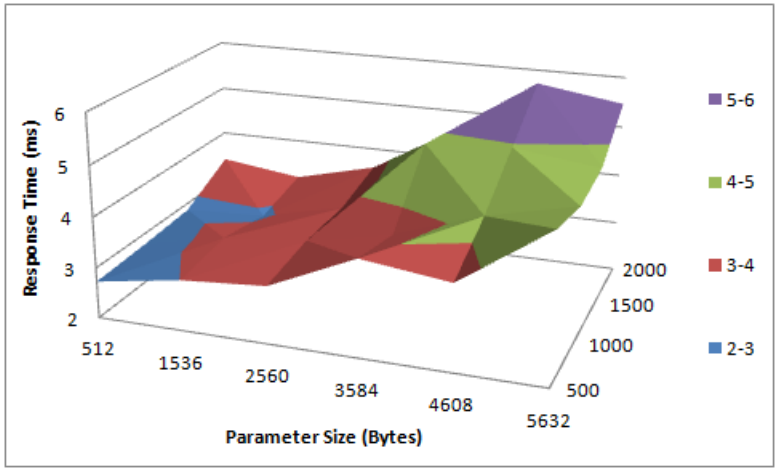

Fig. 17.1 Concat web service response time while hosted on-premise [138]

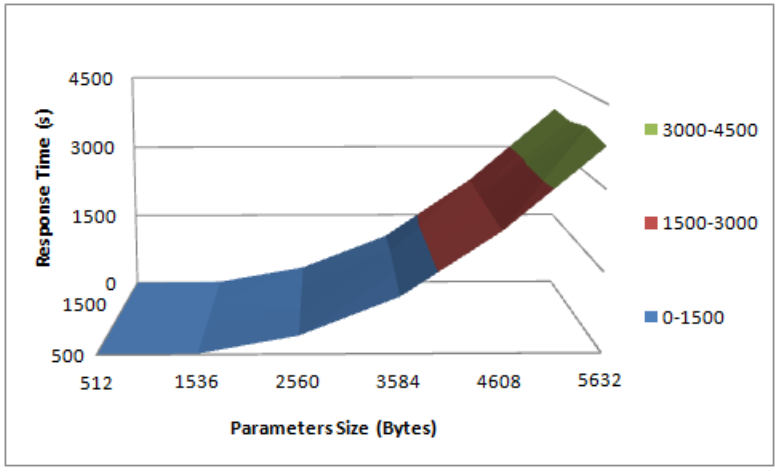

Fig. 17.2 Sort web service response time while hosted on-premise [138]

17.2.2 Web Service Performance Hosted in the Cloud

We measure the performance of web services hosted in the cloud with different payload: 1) different message size for constant number of concurrent messages and 2) different number of concurrent messages for constant message size.

The results for response time of Concat web service hosted in the cloud is shown in Figure 17.3. Both input factors are important for Concat web service performance, i.e. response time increases for greater message sizes or number of con-

current messages. However, there are performance drawbacks due to additional virtualization layer and cloud software, and small response time in ms comparable to network latency which will be the subject in our further research.

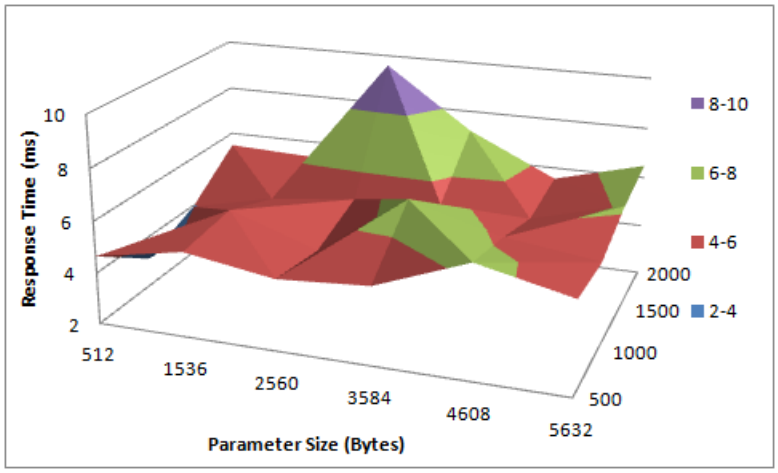

Fig. 17.3 Concat web service response time while hosted in the cloud [138]

Figure 17.4 presents the response time of Sort web service hosted in the cloud. Only the message size impacts its performance (the response time increases as message size increases regardless of the number of concurrent messages).

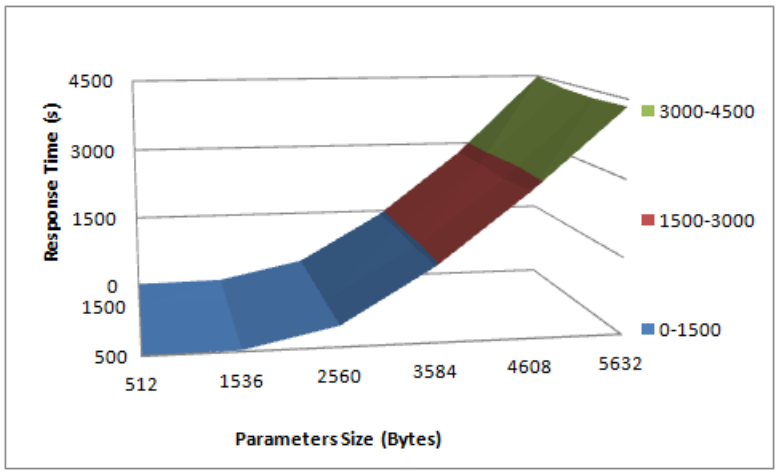

Fig. 17.4 Sort web service response time while hosted in the cloud [138]

17.2.3 On-Premise vs Cloud Performance Comparison

The performance of both web services (hosted on-premise and in the cloud) are compared with different payload depending on different message sizes and different number of concurrent messages.

Figure 17.5 depicts the Cloud vs on-premise relative response time comparison for Concat web service. The results show that cloud environment provides worse response time than traditional on-premise environment for each message size and for each number of concurrent messages. An interesting conclusion is that the cloud provides smaller penalties for greater messages. However, we found a local extreme. We believe that it appears due to communication time impact for small response time and the effect of the virtualization and cloud software which is part our further research.

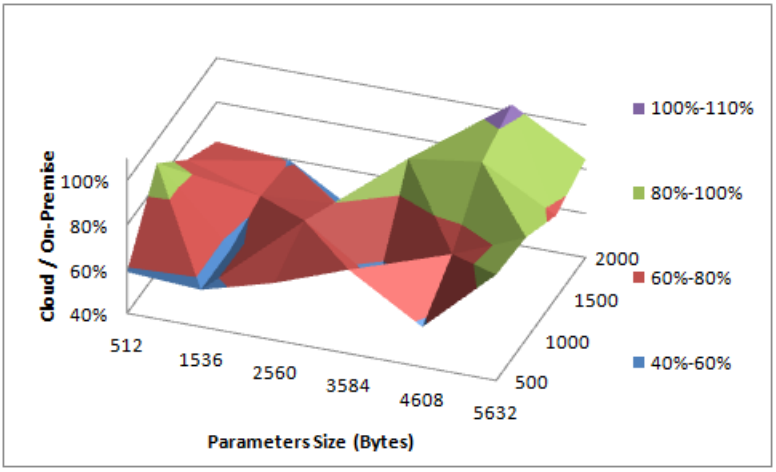

Fig. 17.5 Cloud vs on-premise relative response time for Concat web service [138]

Table 17.1 presents the relative performance for each test case and average for Concat web service. As in previous conclusions, the cloud performance average penalties depend only on the message size. The cloud provides smaller performance penalties for greater message sizes achieving an average performance of 82.42% and 83.99% correspondingly for parameters sizes of 4608 and 5632 bytes.

The relative response time comparison for Sort web service for cloud vs on-premise is shown in Figure 17.6. The results also show that cloud provides worse response time than on-premise for each message size and for each number of concurrent messages. The cloud provides smaller penalties for greater number of messages regardless of number of concurrent messages. The number of concurrent messages impacts the cloud performance for smaller messages.

Number / Size	512C	1536C	2560C	3584C	4608C	5632C	**AVG**
500	58.71%	56.71%	65.89%	77.96%	58.14%	87.91%	**67.55%**
1000	89.39%	56.58%	75.57%	53.44%	72.56%	86.19%	**72.29%**
1500	72.41%	62.16%	64.84%	91.17%	95.75%	77.45%	**77.30%**
2000	62.16%	59.55%	37.40%	56.88%	103.24%	84.41%	**67.27%**
AVG	**70.67%**	**58.75%**	**60.93%**	**69.86%**	**82.42%**	**83.99%**	**71.10%**

Table 17.1 Cloud relative performance compared to on-premise for Concat web service [138]

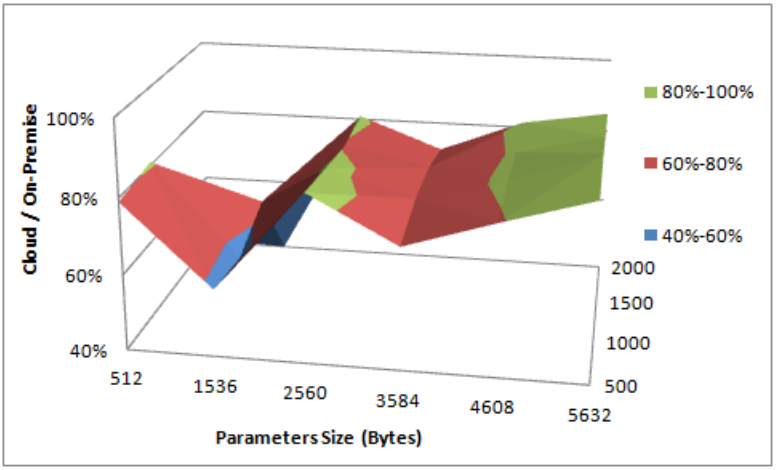

Fig. 17.6 Cloud vs on-premise relative response time for Sort web service [138]

The relative performance of cloud vs. on-premises for Sort web service is shown numerically in Table 17.2. The worst performance the cloud provides for smaller parameters size, i.e. total 512*B* and 1536*B* for average 66.58% and 53.92% from on-premise performance. The cloud provides smaller performance penalties for greater messages for average 86.67% and for smaller number of concurrent messages, i.e. for 500 messages/sec. It provides on average 76.41% from on-premise performance. For huge number of concurrent messages it provides greater performance penalties, i.e. for 2000 messages/sec. it provides average 66.61% from on-premise performance.

17.3 Summary

Two web services were tested for different loads varying the main input factors: the message size and the number of messages. The experiments are realized on the same web services hosted on-premise and in the cloud on the same hardware and runtime environment.

Number / Size	512C	1536C	2560C	3584C	4608C	5632C	AVG
500	78.90%	57.80%	84.07%	72.00%	79.40%	86.29%	**76.41%**
1000	80.93%	56.14%	81.88%	75.21%	81.29%	89.39%	**77.47%**
1500	65.26%	62.87%	79.70%	75.21%	80.66%	86.01%	**74.95%**
2000	41.22%	38.88%	80.87%	72.37%	81.35%	84.99%	**66.61%**
AVG	**66.58%**	**53.92%**	**81.63%**	**73.70%**	**80.68%**	**86.67%**	**73.86%**

Table 17.2 Cloud relative performance compared to on-premise for Sort web service [138]

The results of the experiments show that the performance directly depends on input message size especially for both memory demand and compute intensive web service regardless of the platform as depicted in figures 17.7 and 17.7. This is not emphasized for memory only demand web service.

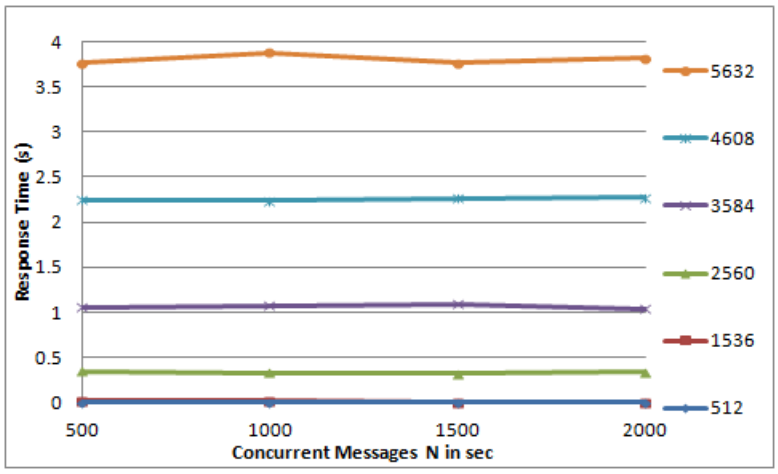

Fig. 17.7 Response time for constant message size but different number of concurrent messages for Sort web service hosted on-premise [138]

We also defined quantitative performance indicators to determine the risk of migrating the services in the cloud for various message size and number of concurrent messages. The conclusion is that the performance is decreased to 71.10% of on-premise for memory demand and to 73.86% for both memory demand and compute intensive web service if it is migrated on the cloud. The cloud provides the smallest penalties for greater message sizes regardless of number of concurrent messages for memory demand web service. However, the smallest penalties for both memory demand and compute intensive web service migrated in the cloud are provided for smaller number of concurrent messages and for greater message sizes.

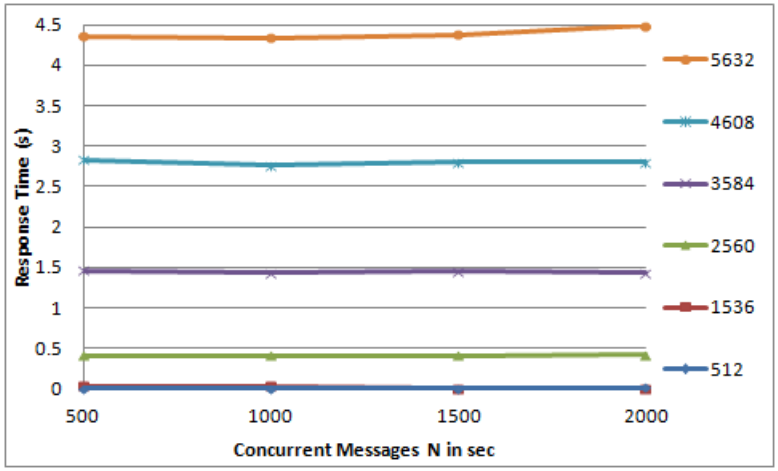

Fig. 17.8 Response time for constant message size but different number of concurrent messages for Sort web service hosted in the cloud [138]

Chapter 18
A Middleware Strategy to Improve Web Service Performance in the Cloud

Abstract CSPs offer infinite scalable hardware resources to their customers. However, instances of VMs are often manually initiated by cloud clients producing degradation of service availability and service latency in peak loads. In this chapter we will present a solution that handles the compute peak loads dynamically for web services hosted in cloud proposed by the authors in [133] for the purpose of this thesis research. The solution introduces a middleware layer between clients and server. The middleware layer will instantiate additional VMs dynamically on demand as service load reaches defined minimum performance level and will forward the messages across VMs. The additional VMs will be shut down when service load returns to defined nominal value.

18.1 Middleware Architecture

This Section describes the web services that are used in the experiments. They are deployed in a virtual machine image and on the Controller Node in the testing environment described in Section 20.1.1.

The architecture using middleware layer is depicted in Figure 18.1. Standard endpoint web service is deployed in one active instance of VM on the Compute Node. Additional middleware web service is deployed in the same instance logically between the endpoint web service(s) and clients. New infrastructure web service is deployed in the Controller Node to instantiate / shut down the additional instance(s) of virtual machine. The additional instances can be instantiated with arbitary CPU, RAM and HDD resources.

The following sections describe closely all web services deployed in the cloud workload environment.

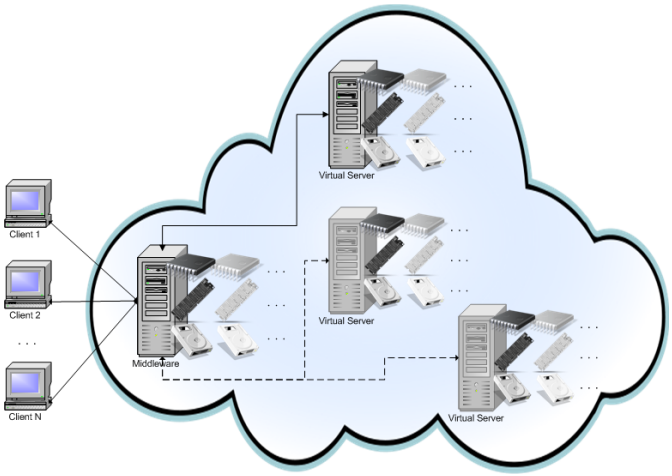

Fig. 18.1 Middleware web service client server model in cloud [133]

18.1.1 The Endpoint Web Service

The endpoint web service is developed in Java EE and communicates with clients using SOAP messages. It receives two input strings and returns to the clients the concatenated string in SOAP message.

The endpoint web service is deployed in the VM. One active instance of the VM is activated with deployed endpoint web service. The same endpoint web service is deployed also in the additional instance of the VM that is instantiated automatically by the middleware when peak load occurs.

18.1.2 The Middleware Web Service

The Middleware is also a web service developed in Java EE and deployed in the same active instance of virtual machine as the endpoint web service. It intercepts the requests from the clients and forwards to the endpoint web service if it is in the normal mode. If the middleware works in the peak mode, then it forwards the particular request randomly either to the endpoint web service hosted on the same active instance or to the endpoint web service hosted on additional instance of virtual machine. The transitions between the two modes are described in details in Section 18.2.

After the middleware layer gets the response from some of the endpoint web services, it forwards the concatenated string to the client. The response time of the middleware layer is measured for each client's request.

18.1.3 The Infrastructure Web Service

This is also a web service in Java EE and deployed in the Controller Node. It's job is to start or shut down the additional instance of the virtual machine when it is invoked by the middleware web service.

18.2 Middleware Strategy

18.2.1 Traditional Scenario

This scenario is the standard client server concept where the clients directly invoke the endpoint service. The response time is measured for each client's request for different test cases.

18.2.2 Middleware Scenario

This scenario is our new middleware strategy and has two sub scenarios: normal and peak. The flag is set for peak mode and unset for normal mode. The clients in this scenario always invoke the middleware web service instead of endpoint web service. The middleware then checks the flag to determine the mode in which it should work.

In normal mode it forwards the particular request to the endpoint web service on the same machine measuring the response time of the endpoint service. If the response time of the endpoint web service exceeds the threshold, then the middleware sets the flag that the peak mode is reached. In the same time it invokes the infrastructure web service to instantiate the additional virtual machine. The middleware forwards back the response to the client after it gets from the endpoint.

If the middleware layer realizes that a peak mode is reached, then it sends dummy request to the additional endpoint web service. It sets the flag for the peak mode when the additional endpoint web service is started, i.e. the additional instance of the virtual machine is started. Otherwise the middleware layer still forwards the requests to the active endpoint web service deployed on the same machine.

If a peak mode is activated then the middleware web service uses random function to determine where to forward the requests, i.e. on endpoint web service on the same machine or to the additional for which it concludes that is already active.

18.2.3 Configuration Parameters

The threshold time in this scenario is set to 50*ms* and the flag is set in the next five minutes and after that automatically unset. The random function is set to evenly balance the forwarding to the particular endpoint service. All these configuration can be configured differently according to cloud platform environment and IT and Quality managers' decisions.

18.3 The Experiments and the Results

This section presents the results of the experiments realized to determine the performance impact of the middleware introduction in the cloud environment.

18.3.1 Middleware Additional Latency

At the beginning we measure the nominal performance of traditional scenario and the additional latency that middleware produces without instantiating additional instance. The response time and their average for particular parameter size of 14.5KB in normal web service load is depicted in Figure 18.2.

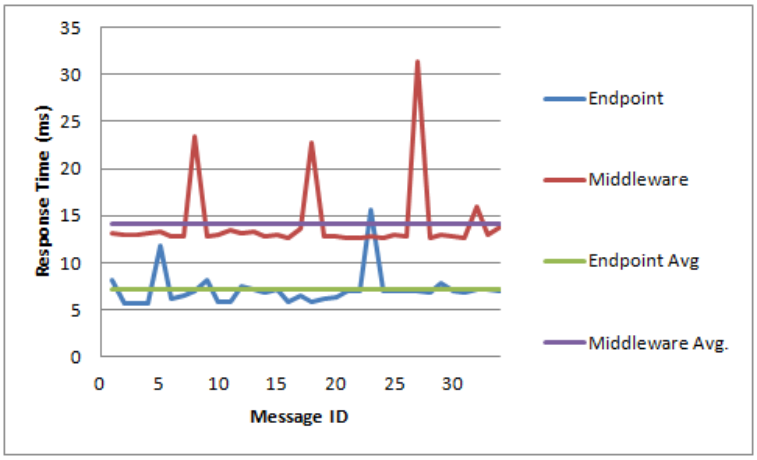

Fig. 18.2 Latency for simple web service after introduction of middleware [133]

We repeat the experiments for different parameter sizes of 100B, 1KB, 10KB and the average response time and the comparison of the two scenario are presented in Table 18.1.

Parameter size	Traditional	Middleware	Latency	Relative
100 B	4.32	10.24	5.92	137.0%
1 KB	4.99	9.74	4.75	95.4%
10 KB	6.43	14.85	8.42	131.0%

Table 18.1 Comparing the nominal performance [133]

Columns *Traditional* and *Middleware* present the average response time in miliseconds for the corresponding scenario. Column *Latency* presents the latency that middleware introduces and column *Relative* the relative increase of the response time in middleware scenario compared to traditional.

Table 18.1 presents that the traditional scenario provides better performance than middleware.

In the following sections we analyze the web services' behavior both with and without load balancing for increased memory load, i.e. we increase the parameter size by 64B each, starting from 64B to 14.5KB.

18.3.2 Middleware without Load Balancing

In this experiment the clients invoke the middleware layer which works only in normal mode without load balancing. Figure 18.3 depicts the results of the experiments realized in traditional and middleware scenario without load balancing.

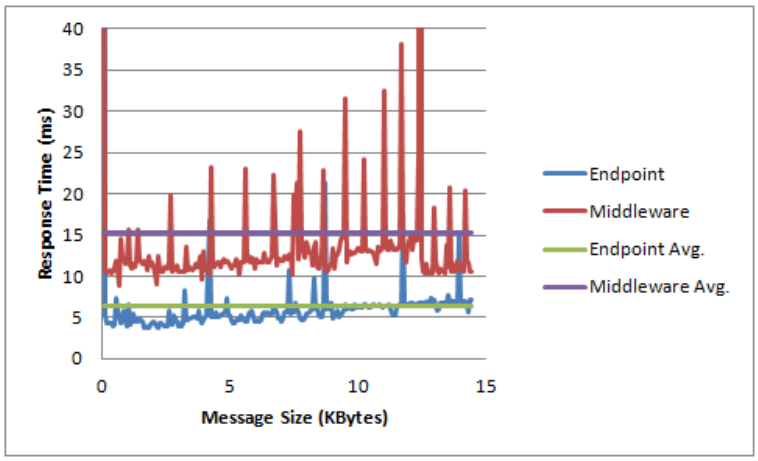

Fig. 18.3 Scenario comparison for middleware without load balancing [133]

The average response time for traditional scenario is 5.84*ms* and for middleware scenario without load balancing is 15.03*ms*. We can conclude that traditional scenario also provides better performance than middleware for increased memory load. Even more, the middleware scenario produces many peaks in response time, a lot more compared to the traditional scenario.

18.3.3 Middleware with Load Balancing and Peak Mode

In this experiment the clients invoke the middleware layer which works with load balancing in both modes. Figure 18.4 depicts the results of the experiments realized in traditional and middleware scenario with load balancing.

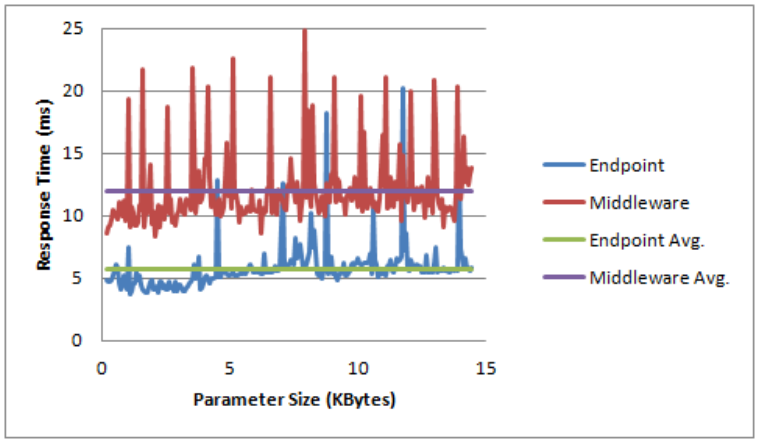

Fig. 18.4 Scenario comparison for middleware with load balancing [133]

The average response time for traditional scenario is similar to previously experiment, i.e. 5.78*ms* and for middleware scenario with load balancing is much lower than middleware without load balancing, i.e. 11.96*ms*. Traditional scenario also provides better performance than middleware for increased memory load even with load balancing. Also the middleware scenario makes many peaks in response time, much more than traditional scenario.

We can conclude that introducing load balancing in the middleware improves the overall performance, but still not enough as traditional endpoint web service.

18.3.4 Why (or when) to Introduce Middleware Layer?

The experiments from the previous section show that introducing middleware with load balancing for string concatenation provides worse performance compared to traditional client server concept. Then why to introduce the middleware strategy when it degrades the performance? Or maybe the more important issue is if there is any kind of web services and some payload such that introducing the middleware and additional resources will improve the overall web service performance?

We set a hypothesis that the additional latency that middleware produces can be compensated if the web service and its payload utilizes the server's processing unit, i.e. a huge part of the response time is due to the execution of web service methods rather than the invoke itself (the case for compute intensive calculations where computational time is dominant in comparison to the communication).

For this purpose in the string concatenation example, we develop another endpoint web service that sorts the input strings before concatenation. Figure 18.5 depicts the results of these experiments. Introducing middleware layer for this web service reduces average latency improving the overall performance compared to the same traditional endpoint web service.

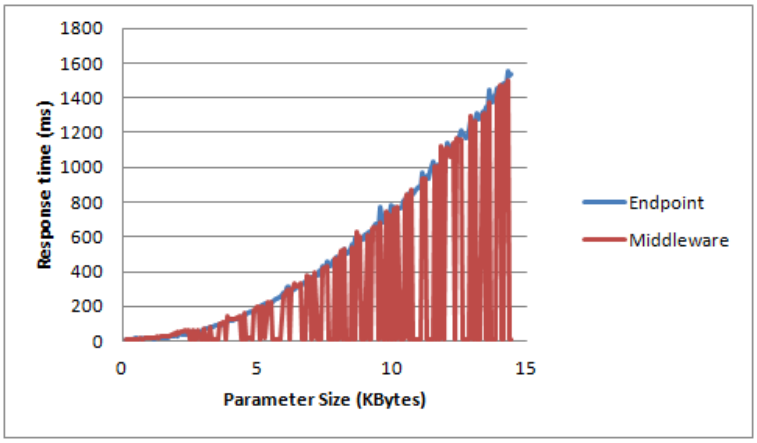

Fig. 18.5 Scenario with Overall Improvement introducing middleware [133]

Figure 18.5 clearly depicts that for this scenario the overall response time increases for bigger parameter size. Also middleware response time has two different response points, i.e. one similar to the traditional endpoint web service and the other is small response time.

Figure 18.6 depicts the responses from the traditional endpoint and from each middleware endpoint.

As depicted in Figure 18.6 the additional endpoint response time is much better than the endpoint on the active instance because the middleware is deployed there.

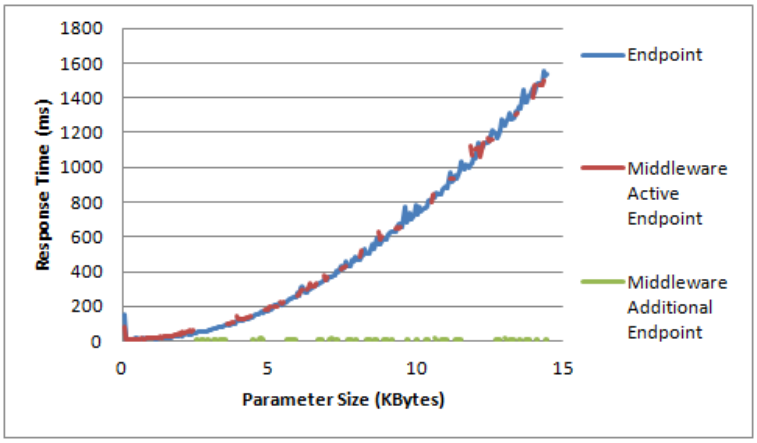

Fig. 18.6 Each endpoint responses

18.4 Pros and Cons

This section presents the pros and cons of our proposed middleware strategy.

18.4.1 Middleware Cons

The main deficiency of the middleware strategy is introducing additional latency in overall response time. The results show that introducing middleware layer even doubles the response time compared to traditional endpoint web service. We can conclude that introducing middleware achieves bad performance for web services with small response time comparable to additional latency that middleware produces. This happens for web services where communication costs are dominant in comparison to the computational.

Another deficiency is the middleware layer bottleneck which is not examined.

18.4.2 Middleware Pros

Several benefits can be achieved from introducing a middleware layer between the clients and endpoint web service in a cloud environment:

- **Better Performance - Smaller Average Response Time**. Our solution provides better performance for a web services when huge part of the response time is spent on web service methods execution rather than the invoke itself, such as the web service for string concatenation and sorting explained in Section 18.3.4. Fig-

ure 18.7 depicts the average response time of the two scenarios for compute intensive web services. Comparing the two trend lines we can conclude that the average response time when introducing middleware with load balancing is smaller than the traditional endpoint web service.

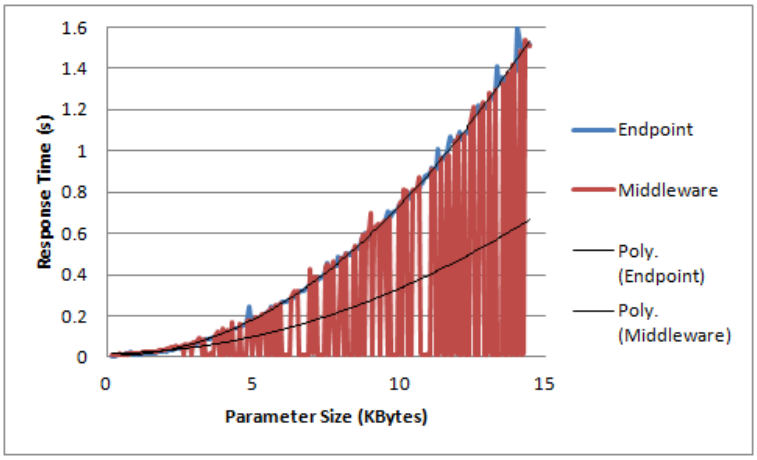

Fig. 18.7 Better performance introducing middleware [133]

- **Load Balanced Server Utilization**. Our solution provides load balanced server utilization for a compute intensive web services. Figure 18.6 clearly depicts that the two endpoint services in middleware scenario are less utilized than traditional endpoint web service. Even more, the load balancing can be reconfigured if the additional instance has less or more resources than the active instance.
- **Increased Service Availability**. Introducing middleware can greatly improve the web service availability. The single point of failure in traditional endpoint web service is improved in this scenario. If the additional instance of virtual machine fails to instantiate or becomes unavailable, then the middleware web service will not forward the requests in that direction.

18.5 Summary

Reducing the cost and improving the performance simultaneously is the imperative for each IT and quality manager. Our strategy proposed in this paper can provide it along with cost reduction that cloud computing paradigm offers.

We propose a middleware strategy to survive compute peaks loads in the cloud environment. Despite the latency for simple web services, the experiments prove that the middleware improves the performance of commute intensive web services

(where huge part of the response time is spent for service calculations). We believe that web services with implemented web service security standards, such as XML Signature and XML Encryption, are most promising for implementation of middleware strategy.

Chapter 19
Message Transformation for Better Web Service Performance in Cloud Computing

Abstract On-premise server performance depends on several parameters. Server's hardware resources, OS and runtime environment are persistent during server and service life cycle; they provide constant performance for even server payload. This feature changes if the server migrates in a dynamic multi-tenant cloud. The server's hardware resources usually are shared among several tenants which impacts server overall performance. For the purpose of this thesis research the authors in [132] analyzed what runtime environment and OS achieve the best performance for web services in cloud PaaS layer for peak loads, particularly when the increased load happens due to huge number of small messages or by huge message sizes. We propose a middleware strategy to survive the peak loads of huge number of small messages and also a model to transform huge messages into smaller chunks and send to the server as separated sub messages.

19.1 Cloud Testing Environment

The experiments are realized in cloud testing environment using OpenStack Compute project Cactus [106]. It is deployed in dual node as depicted in Figure 19.1, i.e. two servers connected with two networks.

Server1 is Controller Node that controls the network and volumes, and schedules instances. Server2 is Compute Node that runs the instances of virtual machines. Server1 has also Compute service as a backup.

Eth0 is public network where the activated instances of virtual machine communicate with the outside world. Eth1 network interfaces are a part of the private or service network where the virtual machines communicate among each other. Network and Port address translation is used to spare the IP addresses, i.e. private IP addresses are used for a network on Eth0 interfaces although it is public network.

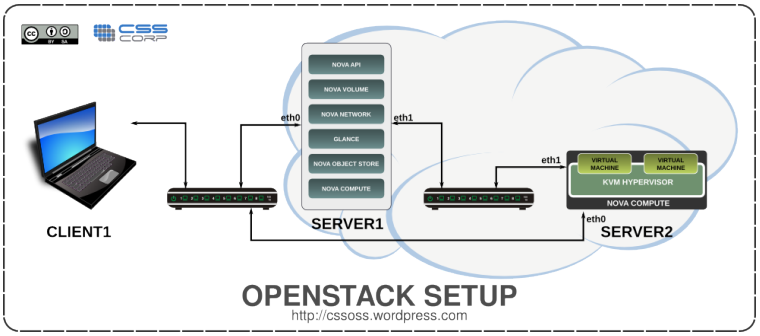

Fig. 19.1 Cloud Testing Environment [108]

19.1.1 The Infrastructure

Hardware Infrastructure consists of two servers. Server1 is HP Server ML110 G6 with 4GB RAM. Server2 is Dell Optiplex 760 with 4GB RAM and Intel Core(TM)2 Quad CPU Q9400 @ 2.66GHz.

The network consists of public and bridged private network as depicted in Figure 1.3. IP addresses of public pool are dedicated to virtual machine instances.

19.1.2 The Platform

Linux Ubuntu Server 11.04 64 bit is installed on both servers. One image of virtual machines is installed also with Linux 11.04 and another image of virtual machines is installed with Windows 2008 Server R2 Enterprise.

Apache Tomcat is installed as a runtime for web services both on the servers and in the virtual machines.

19.1.3 The Client

SoapUI is used to load web services with different message size and different number of concurrent messages.

19.2 The Message Transformation Algorithm

This section presents the message transformation algorithm that can gain better performance. It also presents which web services can use this algorithm to gain better performance with less resources.

Traditional client web service server model is described in Section 5.2.1. The arbitrary number of clients invoke in the same time one or more web services hosted on a web server installed in one instance of virtual machine in the cloud. This solution is not prone to peak loads. Either web server should be underutilized during the most of the time or there will be nosedive drawback in the performance of the web service and web server.

The proposed message transformation algorithm with middleware is depicted in Figure 19.2. Instead of renting one instance of virtual machine with more CPU and RAM resources at the beginning, we propose to rent one web server instance with minimum resources that cloud service provider offers and such that will satisfy the required performance. A middleware layer will be installed on this web server. It will receive all the requests from the customers and will forward the requests to the endpoint web service deployed in the same server. The operating system and runtime environment will be selected to provide the best performance for a nominal load as measured during the process of learning.

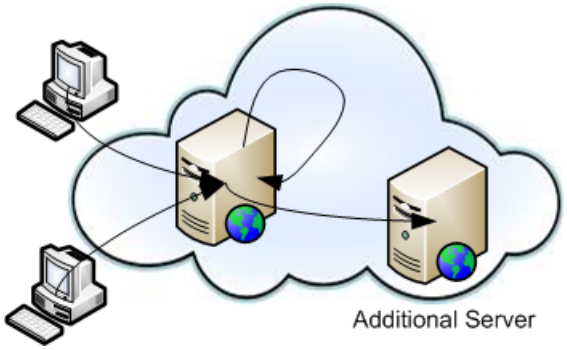

Fig. 19.2 The Message Transformation Algorithm [132]

19.3 New Solutions for Peak Loads

Besides the existing instance of virtual machine we propose a new solution that will rent additional server during the peak loads from different images with different platforms according peak load type. Peak load occurs when a high throughput is sent to the server. The high throughput can be produced by a huge number of con-

current messages or by huge messages, or even both. We propose a solution for each scenario in the following sections.

19.3.1 Peak load with huge number of concurrent messages

This scenario is more probable rather than the scenario in Section 19.3.2. Increasing the total number of users can often provide this peak. Even more, the peak is more weighty for compute or memory intensive web services. E-testing or E-voting are typical representatives of this scenario where a huge majority of users in the same time will load the web service. Only in short period of time the clients concurrently are taking the exams or they vote. Thus the web service will be overloaded with huge number of concurrent small sized messages. This scenario utilizes the web server's processor rather than occupying the memory.

We already proposed a solution for this scenario for compute intensive web services in previous Chapter 18, i.e. introducing a middleware layer between the clients and endpoint web service. The middleware starts and shuts down the instances if a peak load occurs.

We extend and even improve this solution. If the load of the middleware server reaches its limit then additional instance with web server will be started. Our new idea is what type of virtual machine image should be started? The authors in [119] found that Microsoft Windows OS provides better performance than Linux Ubuntu OS for messages above 10KB, i.e. huge messages. Opposite, Linux Ubuntu OS provides better performance than Microsoft Windows OS for huge number of concurrent messages and for small messages.

Therefore, we propose the middleware web server to be installed with Linux server based operating system. Further on, we extend the solution in two directions. That is, if the number of concurrent messages increases but the messages are small sized and the performance reaches its limits, then the additional web server that should be instantiated should be also installed with Linux as it performs better for huge number of small sized messages. Otherwise, if the messages are above 10K then additional server with Windows Server based operating system should be instantiated.

19.3.2 Peak load with huge messages

This scenario does not depend directly on the total number of users but depends on the message size and type that clients send to the web service. Implementing web service security standards always increases the original message size. The authors in [137] determine the message overhead increment both for XML Signature and XML Encryption. The size of signed SOAP message with XML Signature is always greater than the original message by a constant value regardless the size of the origi-

nal message. The size of encrypted SOAP message with XML Encryption increases linearly compared to the size of the original message for each message size. This scenario utilizes the web server's memory rather than the processor.

A huge size message can be provided if the message has a lot of parameters or the input parameters are huge. We focus for the latter case.

We propose a solution based on middleware strategy to improve the web service performance in this scenario. At the begging we propose the middleware layer to be implemented on front-end web server installed with Linux server based operating system. In normal mode the middleware layer forwards the requests to the endpoint deployed on the same server. If the load of the middleware server reaches it's limit then additional instance with web server will be started. If the middleware is invoked with a huge message then the middleware splits it to smaller chunks and forwards them to the endpoint web service. The middleware layer thus will create a connection to the endpoint only once to send all parts of the original request and will not cause a big latency to create a connection for each part of the original message.

This solution doesn't rent additional resources but transforms the original message to smaller chunks that web server with Linux server based operating system handles better.

19.4 The Performance Analysis and Discussion

This Section analyzes if our solution provides better performance than the same endpoint web service on one platform.

19.4.1 Peak load with huge number of concurrent messages

Figure 19.3 depicts the response time comparison for peak load with small messages of 0.2KB, for example, a message with two parameters, 3 bytes each. Both operating systems provide similar performance, Linux in front of Windows, for up to 500 messages per second. Increasing the load, Windows's performance reduces more than Linux's. For load of 1000 messages per second, Windows provides average response time of 162.74*ms* compared to Linux's 26.26*ms*. Our new proposed solution produces smaller latency compared to traditional endpoint, thus implementing middleware will provide better performance than traditional endpoint on Windows for peak loads with small messages.

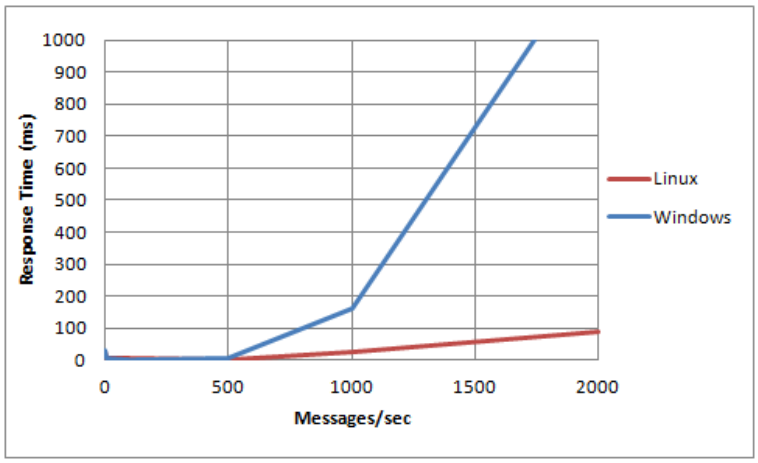

Fig. 19.3 Response time for peak load with small messages of 0.2KB [119]

19.4.2 Peak load with huge messages

Figure 19.4 depicts the response time comparison for peak load with huge messages of 1MB. Windows operating systems provides better performance than Linux for huge message of 1MB. The Linux's performance reduces more than Linux's increasing the number of messages per second. For example, for load of 5 messages per second, Windows provides average response time of 110.83*ms* compared to Linux's 470.76*ms*.

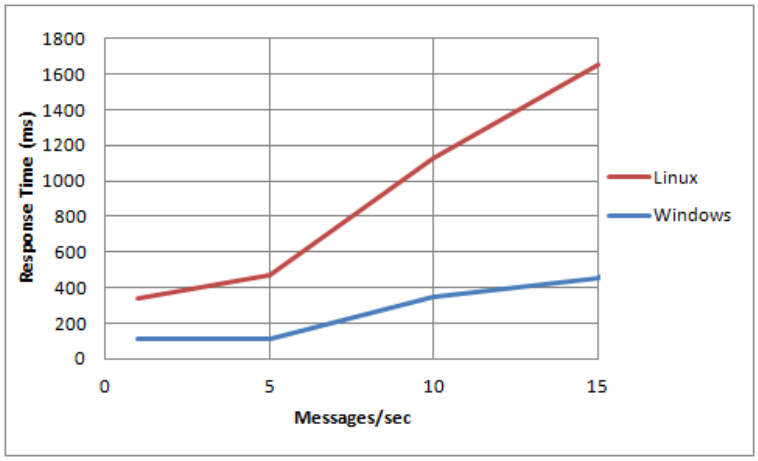

Fig. 19.4 Response time for peak load with huge messages of 1MB [119]

If we compare the results of Figures 19.3 and 19.4 we can conclude that our solution for this scenario has two benefits. Those messages that will be divided and forwarded to the same web server as middleware will use the better performance that Linux provides compared to Windows for peak load with small messages. The messages forwarded to the other instance of virtual machine with Windows operating system will use the better performance that Windows provides compared to Linux for peak load with huge messages.

19.5 Summary

This chapter describes solutions for two possible peaks in web service response time, peaks that appear due to increased number of concurrent requests and peaks with increased load due to huge message size. For the former peaks we propose a middleware based solution that will dynamically instantiate and shut additional instances. The middleware should be deployed on the same machine as the endpoint web service on Linux server based OS since it provides better performance than Windows OSs for small load. When peak load occurs, the middleware forwards the client requests among two endpoint web services, the first deployed on the same machine as the middleware and the other on the additional instance.

The middleware based solution for peak loads with huge message size will forward the requests from the clients to the endpoint web service deployed on the same machine. We assume that Linux server based OS will be installed for small number of huge messages. If the response time increases beyond the threshold, then the middleware strategy will split the input parameters into smaller chunks that Linux OS can process faster rather than the whole message. If the peak is even bigger, then the middleware will start additional instance installed with Windows Server based OS and forwards the client requests among two endpoint web services, the whole messages to Windows Server based OS and the messages divided into smaller chunks on Linux Server based OS.

Therefore, the additional latency that the middleware produces will be compensated with faster response from the endpoint web services. This solution will provide better performance for both peak loads and sometimes even with smaller resources for peak load with huge messages.

Part VI
Cloud Computing Security Challenges and Evaluation

Chapter 20
Web Service Performance when Introducing Security

Abstract Attaching signature and encryption headers to SOAP messages outcomes with message overhead. It requires complex cryptographic operations for each message and additionally parsing the increased XML message. This chapter presents the results of the experiments focused on understanding the performance impact of XML Signature and XML Encryption on Windows and Linux OS. The authors in [135, 137, 136] for the purpose of this thesis research. Increasing message size and the number of concurrent messages degrades server performance for services with and without security implementation. We continue the analysis to determine the optimal input parameters to obtain maximum throughput and web service performance drawbacks produced by XML Signature and XML Encryption varying server load.

20.1 The Testing Methodology

This section describes testing methodology including identification of environment, infrastructure and platform, test plan and design implementation details. Several steps were performed to create efficient and effective tests and results.

20.1.1 Test Environment Identification

The experiments are realized on traditional client-server architecture on the same hardware infrastructure but different platform. Two same web servers are used as VMs of with 1GB RAM and 2 CPU cores. Windows Server and Linux Ubuntu are installed on the VMs correspondingly. Eclipse Jetty is used as web server. SOAPUI [143] is used to create various server load tests. Client and server are in the same LAN segment to exclude the network impact shown in [82].

20.1.2 Test Plan

The test experiment basic goal is to measure the performance cost of implementing signature and encryption to SOAP messages. For that purpose, Concat web service defined in Section 17.1.3 will be used as a test data. The first part of the experiment consists of series of test cases to examine the impact of increasing the message size to the server response time. The second part of the experiment consists of series of test cases to examine the impact of increasing the number of concurrent messages to the server response time.

All test cases are performed (1) using regular SOAP messages; (2) signed messages; and (3) both signed and encrypted messages.

Every test case is run on this way: in a time of 60 seconds, N messages are sent with M bytes each, with variance 0.5. Parameters N and M are changed from test case to test case.

Response time is measured for various number and size of concurrent requests for various message types providing end-to-end security.

20.1.3 Test Environment Configuration

The sizes in kilobytes of the original (regular) SOAP messages sent in test cases are approximately: 0.2, 2, 10, 20, 60, 70, 100, and 1000. The server is loaded with various number of messages (requests) in order to retain server normal workload mode, that is, from minimum 1 to maximum to 1000 requests per second depending of message type and size.

20.2 Cost of Message Overhead

In this section we present the analysis for the cost of message overhead that the authors published in [137] for the purpose of this thesis research.

We measured the message overhead when creating (1) XML message without any security, (2) signed XML message, and (3) signed XML message and then encrypted. To compute the overhead, we extracted the message sizes for each security mechanism. Figure 20.1 depicts the message overhead in kilobytes for different message types.

We can conclude that security mechanisms that do not use encryption add a constant amount of bytes to the regular parameters. The mechanisms with encryption included add a linear amount of bytes comparing to parameters size. Signed and then encrypted messages add linear overhead not only to original SOAP message, but also to signed messages, which means that adding the encryption to regular or signed message adds a linear overhead.

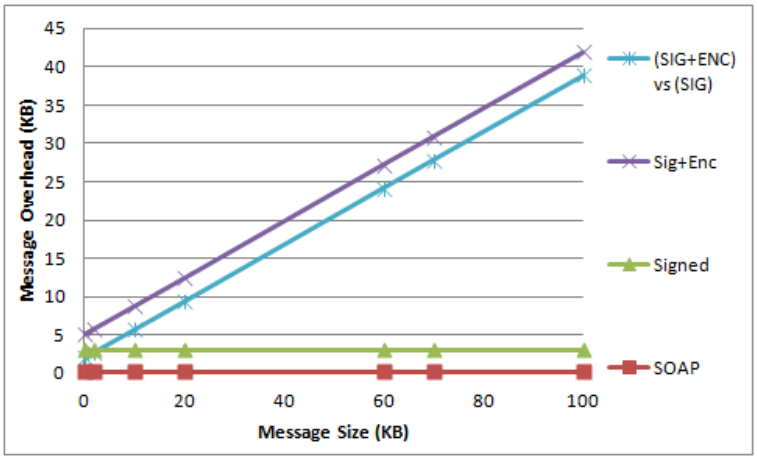

Fig. 20.1 Message overhead in kilobytes [137]

20.3 The Results on Windows OS

This section presents the results of the experiment realized on Windows OS for the three different web services varying server load with different message size and number of concurrent messages.

20.3.1 Web Service without Security

Figure 20.2 depicts the results of the experiments for Concat web service without implemented any security.

We can conclude that both input factors impact to the web service performance. Increasing the number of messages or message size increase the total amount of data sent to the service and thus increases the execution time to serve them. However, we can conclude that the number of concurrent messages degrades the performance more than their size.

20.3.2 Web Service with XML Signature

Figure 20.3 depicts the results of the experiments for Concat web service with XML Signature implemented.

We can conclude that both input factors impact to the web service performance, i.e. increasing the number of messages or message size increase the total amount of data sent to the service and thus increases the execution time to serve them.

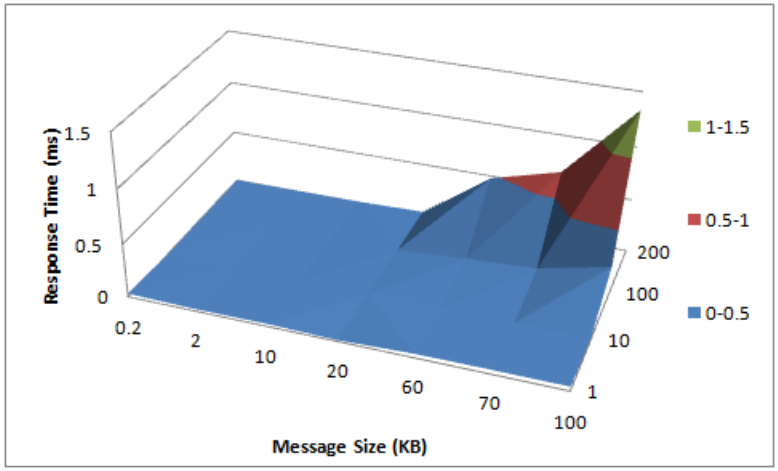

Fig. 20.2 Concat web service response time without security while hosted on Windows

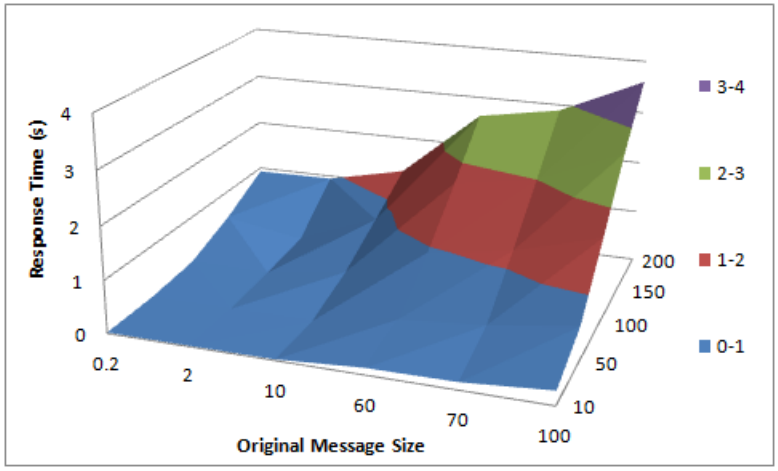

Fig. 20.3 Concat web service response time with XML Security while hosted on Windows

However, we can conclude that also the number of concurrent messages degrades the performance more than their size.

Comparing with the results for Concat web service without security we can conclude that adding security provides huge performance drawback.

20.3.3 Web Service with XML Signature and XML Encryption

Figure 20.4 depicts the results of the experiments for Concat web service with both XML Signature and XML Encryption implemented.

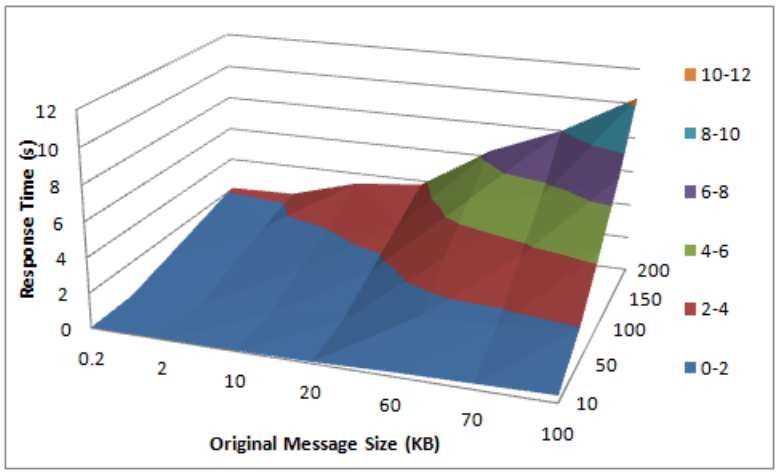

Fig. 20.4 Concat web service response time with both XML Security and XML Encryption while hosted on Windows

We can conclude that also both input factors impact similar to the web service performance, i.e. increasing the number of messages or message size increase the total amount of data sent to the service and thus increases the execution time to serve them. Also we can conclude that also the number of concurrent messages degrades the performance more than their size.

Comparing with the results for Concat web service with XML Signature we can conclude that adding additional encryption to already signed message provides even greater performance drawback than implementing only XML Signature.

20.4 The Results on Linux OS

This section presents the results of the experiment realized on Linux OS for the three different web services varying server load with different message size and number of concurrent messages.

20.4.1 Web Service without Security

Figure 20.5 depicts the results of the experiments for Concat web service without implemented any security.

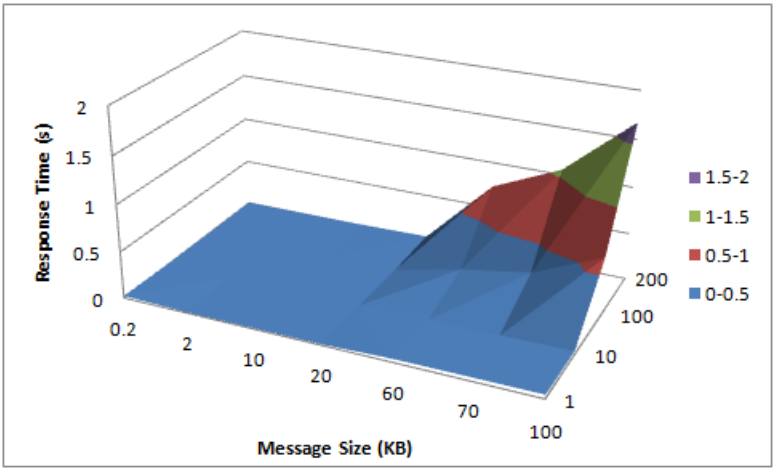

Fig. 20.5 Concat web service response time without security while hosted on Linux

We can conclude that both input factors impact to the web service performance. Increasing the number of messages or message size increase the total amount of data sent to the service and thus increases the execution time to serve them. However, we can conclude that the message size degrades the performance more than number of concurrent messages, even for 10 messages starting for 20K original message size.

20.4.2 Web Service with XML Signature

Figure 20.6 depicts the results of the experiments for Concat web service with XML Signature implemented.

We can conclude that both input factors impact to the web service performance, i.e. increasing the number of messages or message size increase the total amount of data sent to the service and thus increases the execution time to serve them. However, we can conclude that the message size degrades the performance more than number of concurrent messages.

Comparing with the results for Concat web service without security we can conclude that adding security provides huge performance drawback.

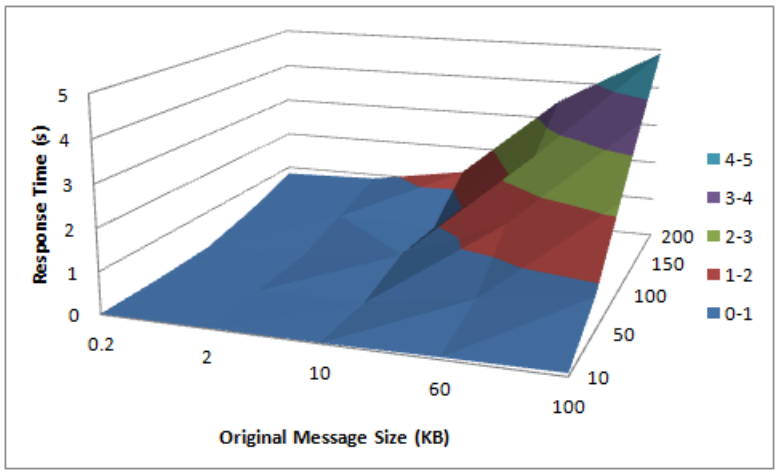

Fig. 20.6 Concat web service response time with XML Security while hosted on Linux

20.4.3 Web Service with XML Signature and XML Encryption

Figure 20.7 depicts the results of the experiments for Concat web service with both XML Signature and XML Encryption implemented.

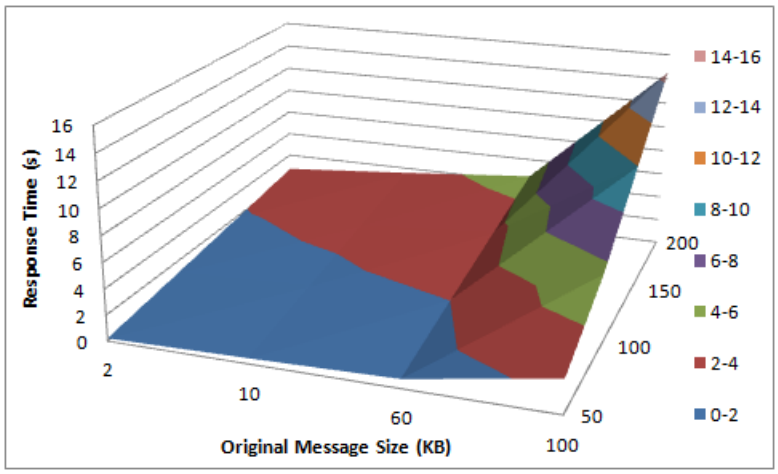

Fig. 20.7 Concat web service response time with both XML Security and XML Encryption while hosted on Linux

We can conclude that also both input factors impact similar to the web service performance, i.e. increasing the number of messages or message size increase the

total amount of data sent to the service and thus increases the execution time to serve them. Also we can conclude that also the number of concurrent messages degrades the performance more than their size.

Comparing with the results for Concat web service with XML Signature we can conclude that adding additional encryption to already signed message provides even greater performance drawback than implementing only XML Signature.

20.5 How to Measure Maximum Throughput

The throughput is defined as the amount of data processed in a particular time. It is measured in bytes per second. In this case the same amount of data can be realized differently, i.e. increasing the number of concurrent messages but decreasing their size and the opposite. We want to determine which server load will provide best server throughput and thus to maximize the server efficiency.

We analyzed the real vs theoretical throughput for each test case, and found the intersection of each two curves, as deppicted in Figure 20.8.

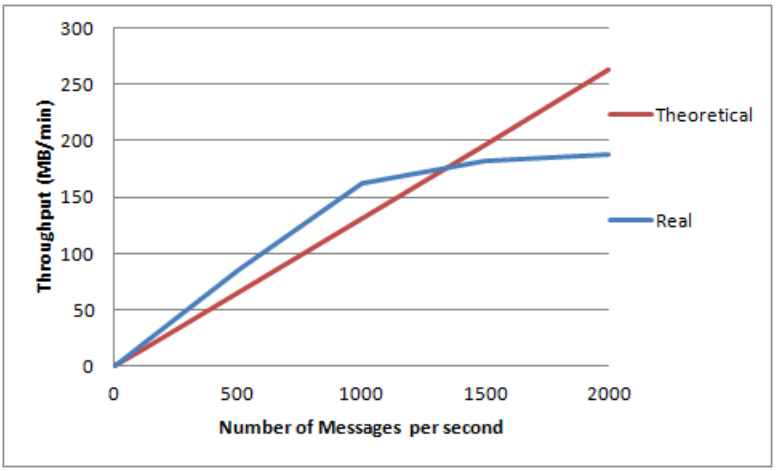

Fig. 20.8 The intersection of theoretical and real throughput for message 2K without security on Linux OS [119]

For smaller number of concurrent messages the server serves all the messages and even more than theoretical. Increasing the load, the server serves smaller number of messages. Thus, we use the intersection dot of each test as maximum throughput. This feature characterizes all the test cases we measured. Next sections presents the results of the evaluations for all web services on both OSs.

20.5.1 Max. Throughput without Security

This section analyzes the results of the experiments realized on both OSs for Concat web service without security for varying server load with different message size and number of concurrent messages.

Table 20.1 presents the measured maximum throughputs for particular message size on both OSs for Concat web service without security. We can conclude that Windows OS provides better maximum throughput than Linux for greater messages ($\geq 10KB$), and Linux for smaller messages ($\leq 2KB$).

Size (KB)	Conc. Mess. Win.	Max. Throughput Win.	Conc. Mess. Lin.	Max. Throughput Lin.
0.2	1063	12.44	2085	24.39
2	933	122.5	1344	176.52
10	800	489.16	621	379.98
20	415	502.81	397	480.56
60	155	561.34	109	393.45
70	118	512.59	84	362.22
100	87	534.93	59	363.74
1000	7	416.09	1	37.44

Table 20.1 Maximums for different message size without security on Windows and Linux OS without security implementation [135, 137]

Figure 20.9 depicts the maximum throughput comparison for a various message size on Windows and Linux OS.

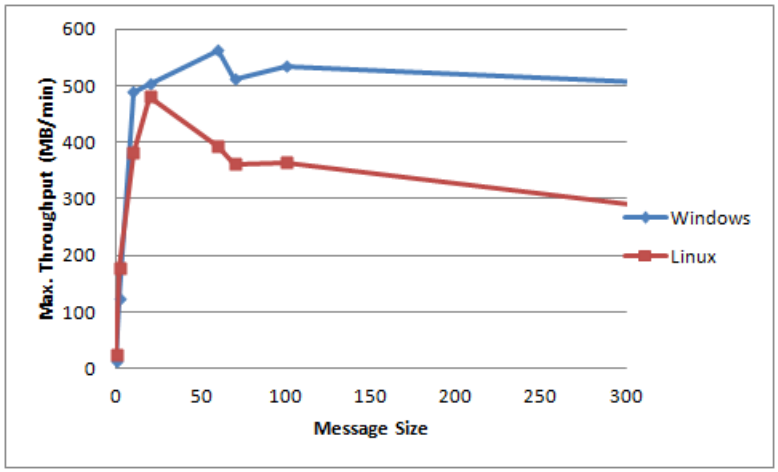

Fig. 20.9 Maximum throughput comparison for a various message size on Windows and Linux OS

We can conclude that the server provides the best throughput when it is loaded with medium number of concurrent messages with medium size for both OSs.

20.5.2 Max. Throughput with XML Signature

This section analyzes the results of the experiments realized on both OSs for Concat web service with XML Signature for varying server load with different message size and number of concurrent messages.

Table 20.2 presents the measured maximum throughputs for particular message size on both OSs for Concat web service with XML Signature. We can conclude that both OSs provide similar maximum throughput for small messages with Linux as a leader. Windows OS provides better maximum throughput for message size between 10K and 60K, and Linux for greater messages provides much better maximum throughput.

Size (KB)	Conc. Mess. Win.	Max. Throughput Win.	Conc. Mess. Lin.	Max. Throughput Lin.
0.2	118	13.1	126	13.97
2	117	27.03	121	27.91
10	80	56.67	73	51.67
60	21	81.27	18	69.39
70	14	63.3	18	79.65
100	1	1.53	14	84.05
1000	0	0	1	37.8

Table 20.2 Maximums for different message size on Windows and Linux OS using XML Signature [135, 137]

Figure 20.10 depicts the maximum throughput comparison for a various message size on Windows and Linux OS.

We can also conclude that the server provides the best throughput when it is loaded with medium number of concurrent messages with medium size for both OSs.

20.5.3 Max. Throughput with XML Signature and XML Encryption

This section analyzes the results of the experiments realized on both OSs for Concat web service with both XML Signature and XML Encryption for varying server load with different message size and number of concurrent messages.

Table 20.3 presents the measured maximum throughputs for particular message size on both OSs for Concat web service with both XML Signature and XML En-

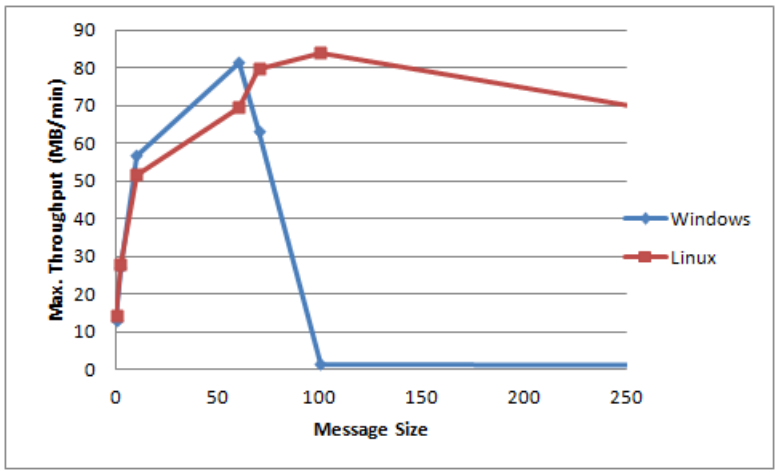

Fig. 20.10 Maximum throughput comparison for a various message size on Windows and Linux OS using XML Signature

cryption. We can conclude that Windows OS provides better maximum throughput for message size between 2K and 10K, and Linux provides much better maximum throughput for small and greater messages.

Size (KB)	Conc. Mess. Win.	Max. Throughput Win.	Conc. Mess. Lin.	Max. Throughput Lin.
0.2	61	14.31	51	20.24
2	60	23.74	19	19.72
10	50	52.14	15	27.56
60	10	51.84	11	55.81
100	1	9.21	4	30.57
1000	0	0	1	28.4

Table 20.3 Maximums for different message size on Windows and Linux OS using both XML Signature and XML Encryption [135, 137]

Figure 20.11 depicts the maximum throughput comparison for a various message size on Windows and Linux OS.

We can also conclude that the server provides the best throughput when it is loaded with medium number of concurrent messages with medium size for both OSs.

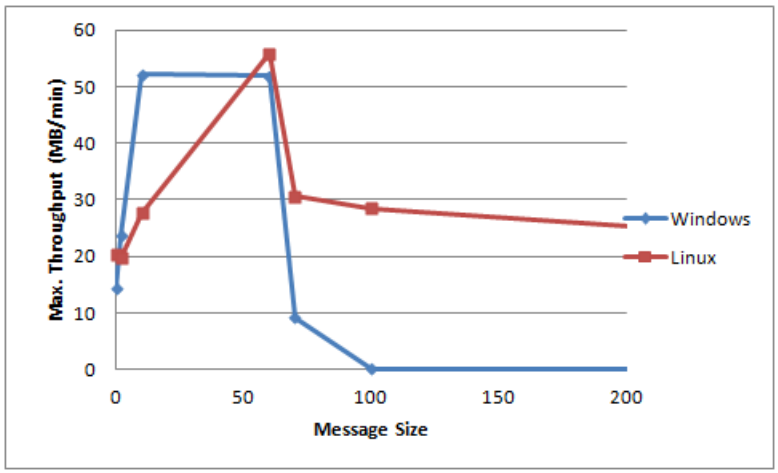

Fig. 20.11 Maximum throughput comparison for a various message size on Windows and Linux OS using both XML Signature and XML Encryption

20.6 Web Server Performance when Increasing Number of Requests

This section describes the results of the performed tests to measure the response time dependency of the message size and a given number of requests, for three implementations of Concat web service as defined in Chapter 20 and hosted on Windows OS.

Figure 20.12 depicts the response time for a given number of requests per second for different message size for the three Concat web service implementations.

20.6.1 Response Time Overhead without Security

For small sized messages without security, the web service has better response time when loaded with 10 or 100 messages per second, For bigger messages (over 30K), the system performance is as expected, that is, higher response time for a huge payload and bigger messages.

20.6.2 Response Time Overhead with XML Signature

For small sized signed only messages (smaller than 35K), the system has better performance when loaded with 10, instead of loaded with 1. For a load with 100

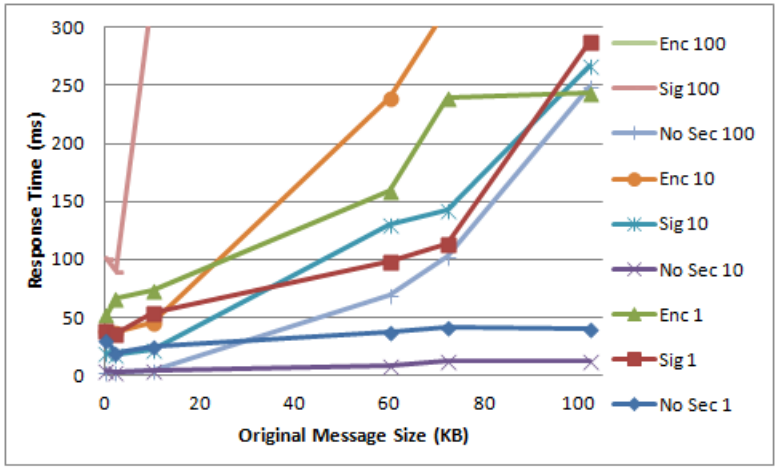

Fig. 20.12 Response time for a given number of requests in a second and message type, depending of message size [136]

messages per second, the system performance is as expected, that is, higher response time for a huge payload and bigger messages.

20.6.3 Response Time Overhead with Signature and Encryption

The results for signed and encrypted messages are similar as signed only messages. That is, for small sized messages (smaller than 25K), the system has better performance when loaded with 10, instead of loaded with 1. For a 100 messages per second, the system performance is as expected.

20.7 Web Server Performance for Different Message Security Type

This section describes the results of the performed tests to measure the response time overhead of the security implementation, for a given number of requests and various message size for three implementations of Concat web service as defined in Chapter 20 and hosted on Windows OS.

20.7.1 Response Time Overhead Implementing Security

In this section we analyze the response time overhead implementing signature to the different sized messages, compared to the same unsigned message for different payload of 1, 10 and 100 messages per second.

Figure 20.13 depicts the response time when implementing signature and both signature and encryption to the unsigned messages, for a given number of requests, and different message size for the three Concat web service implementations.

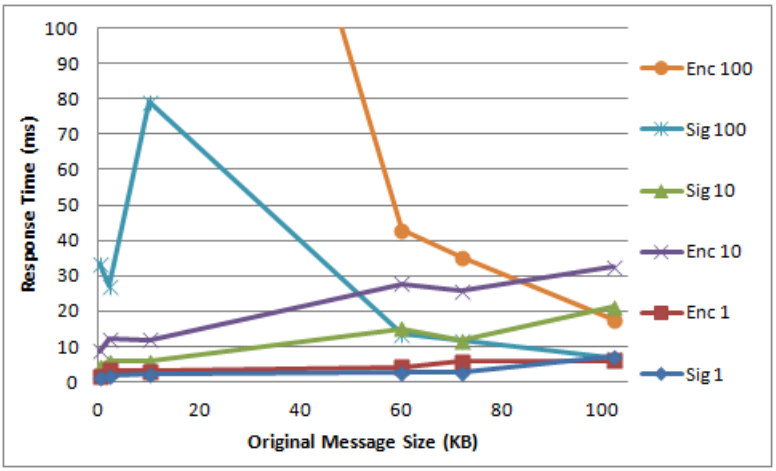

Fig. 20.13 Response time overhead (ratio) for implementing security for a given number of requests, depending of message size [136]

We can conclude that adding signature increases the response time ratio constantly, near linear, growing slowly when increasing message size, but only for a small number of requests, because for a huge number (100), the baseline payload (without security) has huge response time. Implementing both signature and encryption creates similar overhead to the no security messages.

20.7.2 Response Time Overhead Adding Encryption

In this section we analyze the response time overhead adding encryption to the signature for different sized messages, compared to the same signed only message for different payload of 1, 10 and 100 messages per second.

Figure 20.14 depicts the response time ratio when implementing both signature and encryption to signature only messages, for a given number of requests, and different message size.

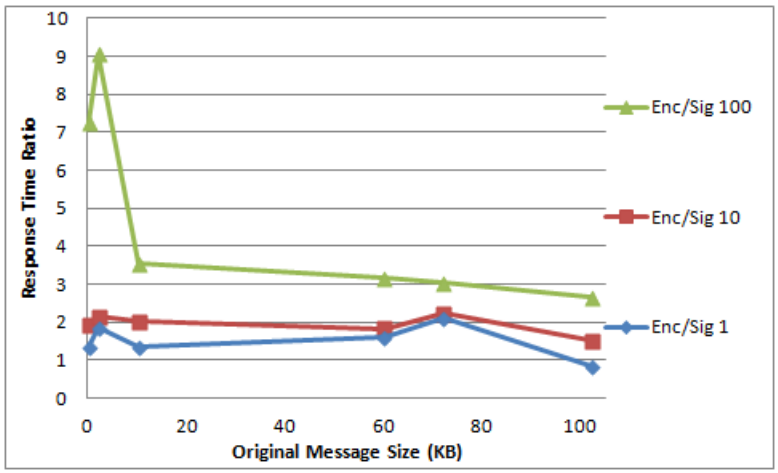

Fig. 20.14 Response time overhead (ratio) for adding encryption to the signature [136]

We can conclude that adding encryption to the signature increases the response time ratio constantly, near linear, but only for a message size above 10K, growing slowly when increasing message size. The ratio decreases at the range of a huge messages and especially huge number of requests due to huge response time of signed messages only.

20.8 Summary

This chapter analyzes the correlation of the two input factors, message size and number of concurrent messages on implementing XML Signature and both XML Signature and XML Encryption. The results show that increasing both parameters degrade the web service performance.

Platform environment is also analyzed. Linux OS handles better the number of concurrent messages and Windows the opposite, i.e. it handles better greater messages.

We analyze the maximum throughput via web services implementing different security-level mechanisms based on WS-Security, i.e. implementing XML Signature and both XML Signature and XML Encryption. We compare both platform and determine that Linux OS provides better throughput than Windows OS for small number of messages with and without security implementation. When implementing security Linux also provides better performance for huge messages. Windows provide better performance for middle sized messages.

Chapter 21
Cloud Security Standardization

Abstract Cloud computing providers and customers services are exposed to new security risks due to multi-tenancy, outsourcing the application and data, and virtualization, besides existing security risks that appear on-premise. Therefore, both the cloud providers and customers must establish even better information security system and trustworthiness each other, as well as end users. This chapter overviews main international and industrial standards targeting information security and analyzes their conformity with cloud computing security challenges. Almost all main cloud service providers are ISO 27001:2005 certified, at minimum [126]. As a result, an extension to the ISO 27001:2005 standard is proposed with new control objective and two controls about virtualization management, to retain generic, regardless of companys type, size and nature, that is, to be applicable for cloud systems, as well, where virtualization is its baseline.

21.1 General Security Standards and Audit and Assessment Guidance

This section presents the overview of main international and industrial standards targeting information security published by the authors in [127] for the purpose of this thesis research.

Many international standards, guidance, and best practices cover security issues. We overview their domain and comment their conformity to cloud computing security challenges.

21.1.1 NIST's 800-53 R3 Security Controls

The NIST's special publication 800-53 R3 [100] refers to Security Controls for Federal Information Systems and Organizations as another security control based guid-

ance. It provides guidelines for selecting and specifying security controls for information systems (ISs) supporting the executive agencies of the federal government to meet the requirements of FIPS 200 [40]. The guidance defines total of 205 controls grouped in 17 families of security controls for an information system and one family of program management controls to manage information security programs.

The standard focuses on managing risks aroused from information systems with risk management at the organizational level incorporated in NIST's Special Publication 800-39 [99].

21.1.2 ISO 27000 Standard series

ISO 27000 is series of standards specifically reserved for information security matters:

- **ISO 27001:2005**. ISO 27001:2005 [73] certification for information security management system (ISMS) can be considered as best solution for securing information assets and also to establish customer's trust in CSP's services. Microsoft proves that information security is central to its cloud operations [89]. The standard adopts the "Plan-Do-Check-Act" model applied to structure all ISMS processes. The model ensures that ISMS is established, implemented, assessed, measured where applicable, and continually improved. The standard defines 133 controls grouped into 39 control objectives and 11 clauses. These controls shall be selected as part of the process to establish ISMS suitable to cover the identified requirements. They are not exhaustive and additional control objectives or controls may also be selected, or some can be excluded, but the prospective candidate must justify the exclusion.
- **ISO 27002:2005**. ISO 27002:2005 [74] is complementary to ISO 27001:2005. It is a practical guideline for developing organizational security standards and effective security management practices and to help build confidence in inter-organizational activities.
- **ISO 27005:2011**. ISO 27005:2011 [75] provides guidelines for Information Security Risk Management (ISRM) in organization supporting the requirements of ISMS. ISRM process consists of context establishment, risk assessment, risk treatment, risk acceptance, risk communication and risk monitoring and review.

21.1.3 Audit and Assessment Standards and Guidance

A company must perform internal and external audits prior certification to obtain ISO 27001:2005 Certificate. There are several guidance and certifications for this purpose.

COBIT 4.1. COBIT [71] developed by ISACA provides a set of 34 high-level control objectives, one for each of the IT processes, grouped into four domains: Plan and Organize, Acquire and Implement, Deliver and Support, and Monitor and Evaluate. The structure covers all aspects of information and the technology that supports it. By addressing these 34 high-level control objectives, the business process owner can ensure that an adequate control system is provided for the IT environment. COBIT version 5 is the newest release.

SAS 70 (Audit) Type II. SAS 70 [2], developed by AICPA, does not specify a pre-determined set of control objectives or control activities that CSP must achieve, but it provides guidance to enable an independent auditor to issue an opinion on a CSP's description of controls through a Service Auditor's Report. SAS70 Type II certifies that CSP had an in-depth audit of its controls (including control objectives and control activities), which should relate to operational performance and security to safeguard customers data. This helps the CSP to build trust with its custoomers. Customers, on the other hand, with the Service Auditor Report from their CSP(s), obtain valuable information regarding the CSP(s) controls and the effectiveness of those controls. The standard SAS70 is now divided into parts and replaced by two new standards: (1) SSAE No. 16 for Service Auditors and (2) Clarified Auditing Standard for User Organizations. We have analyzed SAS 70 since many CSPs have SAS 70 compliance.

There are other security standards that cover specific areas. HIPAA [24] addresses the security and privacy of health data and intends to improve the efficiency and effectiveness of the health care system by encouraging the widespread use of electronic data interchange. PCI DSS V2.0 [111] is developed to encourage and enhance cardholder data security and facilitate the broad adoption of consistent data security measures globally. At high level it has 12 requirements to protect cardholder data, which may be enhanced with additional controls and practices to further mitigate risks at acceptable level.

21.2 Efforts in Cloud Security Standardization

This section presents the analysis of Cloud Security Standardization efforts published by the authors in [126] for the purpose of this thesis research.

Although general security standards can help CSPs in implementing information security system, there is a need for more efforts for cloud security standardization. CSA identified top threats to cloud computing in [30]. In order to mitigate the risks of threats ENISA identified and assessed the risk level as a function of the business impact and likelihood of the incident scenario [18].

NIST discusses the threats, technology risks, and safeguards for public cloud environments and provides the insight needed to make informed IT decisions on their treatment [97]. The main emphasis is set on security and data privacy.

The CSA's initial report V2.1 [29] contains a different sort of taxonomy based on 15 different security domains and the processes that need to be followed in an

overall cloud deployment. New candidate domains are proposed for version 3 [28] and are of the greatest interest to experienced industry consumers and security professionals. Core functionalities, optional features, services, addressed threats, and the challenges to be focused on are addressed for each candidate domain.

CSA puts a lot of efforts in its CSA GRC project [27]. A list of 98 controls grouped into 11 groups is defined in [26]. Each control is mapped into compliant control of other security standards or best practices.

21.2.1 Appropriate Standard for Cloud Security Challenges

The best solution for CSP's information security system is to cover and meet both the ISO standard and NIST guidance controls. But, it is not so simple. NIST's 800-53 [100] shows that a small number of controls are not covered in the other standard. Also, neither NIST's 800-53 security control subsumes ISO 27001:2005, nor opposite. There are many security controls with similar functional meaning, but with different functionality. Other security controls with similar topics are addressed in the same control objective (ISO) or family (NIST), but has different context, perspective, or scope. Another problem is that some controls from one standard are spread in several controls in the other standard.

The standards differ in their purpose and applicability, as well. While ISO 27001:2005 is general purpose and applies to all types of organizations, NIST's 800-53 is applicable for information systems supporting the executive agencies of the federal government.

The main concern here is: are the controls of both standards applicable to CSP and all cloud service layers? Do they cover all the traditional security challenges, as well as newly opened security issues in cloud? Are there any security challenges in cloud computing not covered with these controls?

ISO 27001:2005 is a general purpose standard and therefore, its control objectives are conformable to CSP. But the question remains: Are they enough for CSP's ISMS? Our further research is going into two directions: first, we measure the CC efforts to be taken for each ISO 27001:2005 control objective if their services are hosted on-premise or in the cloud. And second, we analyze if there should be any other security control to be included in the ISO 27001:2005 controls.

21.2.2 CSPs' Efforts towards Security

We continue with overview of the existing CSPs security certification and accreditations, as well as their security features. Table 21.2.2 presents the evaluation of the security standards certification that existing CSPs have, as well as their security features.

CSP	Amazon	Salesforce	Microsoft	Google	IBM
Security Compliance	PCI DSS Level 1, ISO 27001, SAS 70 Type II, HIPAA	ISO 27001, SysTrust, SAS 70 Type II	PCI DSS, HIPAA, SOX, ISO 27001, SAS 70 TYPE 1 and II	SAS 70 Type II, FISMA	ISO 27001
Security Features	AWS IAM, AWS MFA, Key Rotation	System status, Management Commitment for privacy	Access control, segmenting customer data	2-step verification	Rational AppScan OnDemand, Security Services for compliance

Table 21.1 Existing CSPs Security Certification and Accreditation, as well as Security Features [126]

As shown, all CSPs have one or many security certificates or compliances for their infrastructure. In addition, many CSPs not only they implement security features in their cloud systems, but they also offer security features to their customers to assess whether their services hosted in the CSP cloud are compliant to particular security standard.

21.3 ISO 27001:2005 (in)Compliance for Cloud Computing

In this section we present the analysis of the ISO 27001:2005 requirements' conformity to cloud computing security challenges published in [127] for the purpose of this thesis research. The authors analyze particularly new cloud computing security challenges, such as customer isolation, insider attacks, and security integration [44], due to cloud computing multi-tenancy, virtualization, and outsourcing the customers' data and applications. We evaluated that almost all main CSPs are ISO 27001:2005 certified. Due to new security challenges we analyze if CSP' ISO 27001:2005 Certificate will be enough to generate trust for customers that are secured in the rented infrastructure, platform or software.

21.3.1 Security Challenges due to Virtualization

Traditional on-premise data-centers security solutions do not comply with virtualized environment, because of the complex and ever-dynamic nature of cloud computing [67]. The virtualization by itself does not affect the security if it is used on-premise in a physical, logical and environmental isolated secured environment. IDS and IPS systems can secure the internal virtual and physical machines from the exterior environment, if they are into one autonomous system, that is, under same ad-

ministrative governance. Figure 21.1 depicts multitenant environment where cross VM attacks is possible.

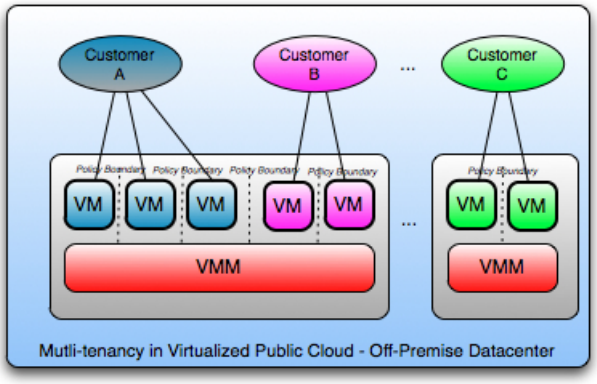

Fig. 21.1 Virtualized Multi-tenant Environment in IaaS and PaaS [29]

In cloud computing, especially in IaaS and PaaS, the resources are shared and rented to the different customers. Even more, the same physical machine can be shared to many different customers. The current virtualization is weak and can be easily attacked [1]. The security solutions for some flaws are found, but new security threats and vulnerabilities arise day by day. Thus, CSPs' security perimeter is broken from inside, making their IDS and IPS helpless. Therefore, CSPs must introduce effective isolation among the customers, although allowing physical resource sharing.

Multitenancy exists In SaaS cloud service layer, as well. There are three degrees of data isolation for SaaS applications presented in Figure 21.2. In Isolated environment each tenant has its own database. Tenants in Semi-shared environment share the database using a separate schema and in Shared environment share both the database and the schema.

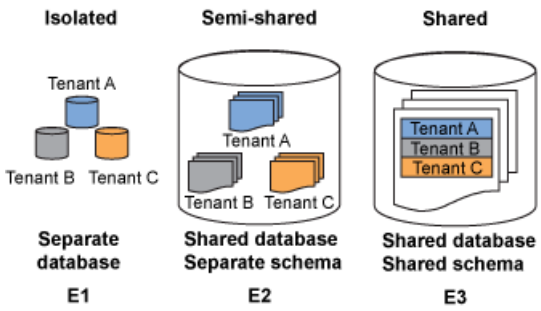

Fig. 21.2 Virtualized Multi-tenant Environment in SaaS [151]

We found several security solutions for virtualization challenges. The authors in [59] propose SEC2 solution which enables users to customize their security policy settings the same way they control their on-premise network. Virtualization-Aware Security Solution CloudSec which monitors volatile memory to detect and prevent for the kernel data rootkits is proposed in [68].

Analyzing ISO 27001:2005 requirements and their controls we concluded that there is no control for virtualization. Clause 11 that covers access control and also many standard controls, even the whole control objective, assume that operating systems are on separate real machines. But in the reality, issues such as trusting the VM image, hardening hosts, and securing inter-host communication are critical areas in IaaS [149]. Therefore, we propose to include a new control objective for *virtualization management*, with two controls: *virtualization* and *virtual machines control*. For the former we propose: *Information involved in virtual machines shall be appropriately protected* and for the latter: *Virtual machines shall be adequately managed and controlled, in order to be protected from internal and external threats, and to maintain information security in transit*. In addition to this, NIST defines the control SC-30 *Virtualization Techniques* in [100], which is not mapped to any control of ISO 27001:2005. NIST's control is far from enough to cover all security flaws due to multi-tenant virtualization in cloud computing.

21.3.2 Security, Data Protection and Privacy as-a-Service

Business does not fully accept cloud infrastructure, platform and software due to security, data protection and privacy, as well as trust issues. Combining the advantages of secured cloud storage and software watermarking through data coloring and trust negotiation, the authors in [65] propose reputation system to protect datacenter access at a coarse-grained level and secure data access at a fine-grained file level.

Such systems and solutions supersede and subsume the traditional security systems, and thus CSPs should implement them. Therefore, offering *Security-as-a-Service* (SECaaS) and *Data protection and privacy-as-a-Service* will speed up cloud market growth, both for the providers' offers and clients, as well as cloud trustworthiness. CSA offers 10 candidate domains for SECaaS [28].

Data privacy is treated in two controls in ISO 27001:2005 requirements. The control 6.2.3 requires the client data privacy (for cloud customers) and the control 15.1.4 requires from the CSP to ensure data privacy. These two controls obligate both the customers and the CSPs to manage the data privacy with higher importance.

As shown in Table 21.2.2, many CSPs are not only complained to some security standards, but they offer services to customers to help them in their security standard compliance, as well. Thus, the risks that arise from multi-tenancy and virtualization will be mitigated, and mutual trustworthiness will be established among CSPs, cusotmers and end users.

21.3.3 Performance challenges

All cloud computing security solutions and techniques, and many other as well, degrade cloud services performance. Implementing identity and access management, web and email security, intrusion management [28], as well as monitoring systems, data coloring, and other traditional security services, such as web service security produce data overhead and system latency which must be considered due to their negative impact to server performance, and thereby to the system availability.

21.4 Summary

Business managers know that risks exist in spite of all the benefits of each new technology or business model offers. There are a lot of regulatory violation, security, trust and privacy issues. Thus, each company that dives ahead using the benefits of cloud computing, should evaluate the risks found if moving its services into the cloud, compare to if retain to the traditional solutions.

This chapter overviews main international and industry standards towards security, and analyze their conformity to cloud computing. There are many different cloud security threats, vulnerabilities and control definitions, best practices, in order to standardize cloud security, as well.

ISO 27000 series (27001:2005, 27002:2005, and 27005:2011) of standards are defined as generic and they cover not only the technical solutions to technically identified threats and vulnerabilities, but take into account the operational, organizational and management vulnerability, as well. Due to its generality, as well as many open cloud security challenges, ISO 27001:2005 is not fully conformal with cloud information security system. Therefore, we propose a new control objective in ISO 27001:2005 requirements, *virtualization management*, with two controls covering *virtualization* and *virtual machines control*.

Chapter 22
Cloud Computing Security in Business information systems

Abstract This chapter presents the research published by the authors in [129] and [127] for cloud computing security challenges in business information systems. Cloud computing becomes the best offer in ICT for data storage and processing, offering flexible and scalable computing processing capacity. But, cloud computing may produce different risks with different impact to client company business than traditional IT solutions. CSPs must implement effectively information security management to reduce the security risks improving the cloud customer business continuity. With high-level risk-based approach this chapter addresses the risks of the security challenges in the cloud in order to improve the client company business continuity if migrates its services into cloud. The comparative analysis of main security benefits and detriments of the cloud that impacts the business continuity was not performed in the literature so far. The benefits that cloud computing offers to business continuity are presented in order to depreciate the risks to acceptable level.

22.1 Security Challenges Moving into Cloud

This section presents the security challenges for the company when moving into cloud published by authors in [129] for the purpose of this thesis research.

The security objectives of a company are a key factor to make decision about outsourcing their IT services, especially data and applications to a public cloud computing environment [97]. But, in some cases, cloud computing offers enhanced security benefits to the companies; for small companies with limited qualified IT administrators and security officers, and lack of business growth, it provides opportunity for overall security improvement.

Many organizations arent comfortable storing their data and applications on systems that reside outside of their on-premise datacenters [19]. This might be the single greatest fear of cloud clients. One approach to security challenges in the cloud is technical approach. Thus, [81] focuses on technical security issues arising from the

usage of cloud services and especially by the underlying technologies used to build these cross-domain Internet-connected collaborations.

[97] provides an overview of the security and privacy challenges pertinent to public cloud computing and points out considerations organization should take when outsourcing data, applications, and infrastructure to a public cloud environment. The advantages and disadvantages (in the context of data security) of using a cloud computing environment are presented in [68]. It also analyzes the data security risks and vulnerabilities which are present in current cloud computing environments. [149] illustrates the unique issues of cloud computing that exacerbate security and privacy challenges in clouds and discusses various approaches to address these challenges and explore the future work needed to provide a trustworthy cloud computing environment. [21] makes a step forward and proposes to extend control measures from the enterprise into the cloud through the use of Trusted Computing and applied cryptographic techniques to alleviate much of todays fear of cloud computing.

We found nice high-level approach of security management in cloud computing in [117] where the authors provide an overall security perspective with the aim to highlight the security concerns that should be properly addressed and managed to realize the full potential of cloud computing. Different cloud delivery and deployment models are matched up against some of the information security requirements.

However, so far there are no papers that analyze how cloud security vulnerabilities and threats impact to both the clients and providers business continuity and it is our challenge and main topic in this paper.

22.2 Risk-based Approach

This section presents the risk-based approach for the company when moving into cloud published by authors in [129] for the purpose of this thesis research.

Business managers know that risks exist in spite of all the benefits of every new technology or business model offers. Also, many issues like regulatory violation, security, trust and privacy appear. Thus, each company that dives ahead using the benefits of cloud computing, should evaluate the risks found if moving into the cloud, as well as if stay to the traditional solutions.

The main challenge when moving into cloud is security. Outsourcing data and application, virtualization and hypervisors, heterogeneity, lost security perimeter are some of the issues that should be addressed at least. The client company should define risk assessment mechanism to define levels of risk and make it part of the system development life cycle. Without preparation of risk assessment, it would be impossible to evaluate whether company systems are candidates for operating in the cloud and to assess the potential CSPs for their risk management practices. When this process is accomplished, the company systems and projects can have their risk assessments mapped with the CSP and a decision can be reached about whether moving into cloud is appropriate for the systems.

This risk assessment should not be a static one, but a dynamic, in order to meet the latest standards and trends. Both the cloud client and provider should evaluate the risks for the cloud services according the providers cloud design and the users service risk assessment. Also, hypothetically and eventually leaving the cloud, that is, moving back to the traditional solutions, should be covered in the risk assessment.

It is often possible for cloud clients to transfer the risks to the cloud provider, if applicable. However, neither always nor all risks can be transferred to the provider. If some risk leads to the incident scenario with business failure, serious damage to reputation or legal implications, it is hard or even impossible for any other party to compensate for this damage. Ultimately, you can outsource responsibility but you can't outsource accountability [18]. Therefore, the motivation of this research is to address the main security challenges in the cloud, especially those that can be disastrous to the cloud client business, and have impact to the business continuity.

22.2.1 Information Security Risk Management

Cloud features, such as scalability and flexibility, impacts both positive and negative to the security [18]. The massive concentrations of data, applications, servers and resources in the cloud provoke the hacker efforts to attack, but on the other hand, cloud-based defenses can be more robust, scalable, etc, offering better protection as the cost for traditional solutions defenses. Therefore, both the client and provider should perform the information security management.

The potential cloud clients should establish metrics and standards for measuring performance and effectiveness of information security management before moving into the cloud. Therefore, the risks of using cloud computing should be assessed and compared to the risks of staying with in-house solutions [18]. CSPs should also include metrics to assist customers in implementing their Information Risk Management requirements. The potential cloud clients should understand their current metrics and how they will change when operations are moved into the cloud, where a provider may use different (potentially incompatible) metrics [29].

A formal risk assessment process, as a part of the security risk management process, should be established that allocates security resources linked to clients business continuity and to compare the risks of using cloud computing with the risks of staying with traditional solutions.

22.2.2 Risk Assessment Process

Security risk assessment is activity where the risks are identified, quantified or qualitatively described, and prioritized against risk evaluation criteria and objectives relevant to the organization. This activity should be the main sub process during the Security Risk Management, because all the assets in cloud are exposed neither to

the same risks, nor to the same risk level as before moving into cloud. The same assumption can be made in the eventually reverse process, which is, moving back from the cloud to the traditional solutions.

Security risk assessment is critical process which helps the company identifying, selecting / excluding security controls during the process of establishing the ISMS for ISO 27000 certification candidates, or during the processes of reviewing ISMS and its improvement. Lack of attention in risk assessment when migrating into cloud, can increase the information security audit findings. Even more, some of the risks can be unidentified or remain untreated. This motivated us to deeper explore the effects of risk in business continuity when moving into cloud.

Besides the protection of information assets, detailed and technical security risk assessments in the form of threat modeling should be applied to applications and infrastructure as well, due to their outsourcing.

The risk evaluation is the next activity after the risk identification and estimation [75]. For each asset, the relevant vulnerabilities and their corresponding threats should be considered, and if there is vulnerability without a corresponding threat, or a threat without corresponding vulnerability, there is presently no risk (but precaution should be taken if the situation is changed eventually).

Also, other important issue is the business impact of the incident, as well as the likelihood to happen. [30] provides needed context to assist organizations in making educated risk management decisions regarding their cloud adoption strategies.

A matrix for risk quantification to successful risk measurement in a scale of 0-8 is defined in [75], annex E. The risks are rated as low (scale 0-2), medium (scale 3-5) and high risk (scale 6-8) as shown on Figure 22.1.

	Likelihood of incident scenario	Very Low (Very Unlikely)	Low (Unlikely)	Medium (Possible)	High (Likely)	Very High (Frequent)
Business Impact	Very Low	0	1	2	3	4
	Low	1	2	3	4	5
	Medium	2	3	4	5	6
	High	3	4	5	6	7
	Very High	4	5	6	7	8

Fig. 22.1 The risk level as a function of the business impact and probability of incident scenario [75]

With these rating, business managers can evaluate the risks before and after eventually moving into cloud, and they can measure whether the risks are acceptable and treated as planned.

22.3 Business Continuity vs Cloud

This section presents the business continuity challenges for the company when moving into cloud published by authors in [129] for the purpose of this thesis research.

This section briefly describes why Business Continuity and Disaster Recovery Planning is important and points the security benefits and detriments that impact to the cloud client business continuity. It also introduces proposals which minimize the impact to business continuity and the probability of incident scenario for each detriment. These main risks can be assessed appropriately and mitigated to the acceptable level by applying recommendations in these proposals according to matrix for risk level as a function of the business impact and probability of incident scenario [75].

22.3.1 Why Business Continuity and Disaster Recovery Planning?

The main goal of every company is to make the business growth. To achieve business growth, the company must have plans in place that will allow business continuity. The purpose of every business continuity / disaster recovery planning is to minimize the impact of any predictable / unpredictable interruption event on business processes. Business continuity and resiliency services helping businesses avoid, prepare for, and recover from a disruption. The cloud client business continuity and disaster recovery plan should include scenarios for loss of the cloud providers services, and even for the providers loss of its third party services and third party-dependent capabilities [29]. In BCP, cloud client must determine who to contact if security incident occurs, or other events that require investigation, identification, notification, reaction, or even eventually legal actions. This plan should be tested periodically together with the cloud provider. As the cloud provider becomes an external party the client relies, the cloud provider has to develop and approve BCP, mapped to the international standards, such as [75, 73]. The CSP must supply the client with providers:

- Documentation for assets and resources are assessed and audited, as well as the frequency of the assessments and audits.
- Incident management, business continuity and disaster recovery plans, policies, and processes and procedures
- Review of co-location and back-up facilities, if applicable
- Providers critical services, key performance indicators (KPIs), and the way they are measured

In order to make a decision if the business migrate the services from traditional solutions to the cloud, business managers should complete and sustain the risk assessment, establish risk acceptance and measure the security risks in both solutions, especially the situation if they impact to the business continuity. Table 22.3.1 as

pointed out in [3] lists many events which could have impact to the cloud client business continuity, some of them potentially disastrous.

Avalanche	Flood	Shooting
Severe Weather (heat, cold, blizzard, etc.)	Natural Gas Leak	Fuel Shortage (usually associated with a loss of main electrical power)
Biological Hazard	Heating Ventilation or Air Conditioning Failure	Bomb Threat
Civil Disorder	Hostage Situation	Kidnapping
Telecom Outage	Acts of Terrorism	Theft
Robbery	Train Crash or Derailment	Lightning Strike
Computer/Software Failure, Virus or Destruction	Employee/Union strike	Acts of Vandalism
Pandemic	Picketing	Power Outage
Fire Damage	Water Damage	Radiological Hazard

Table 22.1 Potentially Disastrous Events [3]

22.3.2 Business Continuity Benefits from the Cloud

This section presents the business continuity benefits for the company when moving into cloud published by authors in [129] for the purpose of this thesis research.

Several papers [81, 21, 18, 29, 149, 97] have mentioned many security vulnerabilities and threats, but this section summarizes all relevant efforts and introduces another dimension. Despite the security challenges and risks appeared to the business continuity if moving into cloud, we address several benefits that cloud computing has over traditional business continuity, as well. These benefits improve the clients BCP and depreciate the impact of the incidents to the clients business.

Eliminating downtime. SaaS offers advantages over traditional computing, for example, in the email services. Thus, SaaS ensures that email messages are never lost and makes the system outages virtually invisible to end users no matter what happens to your employees or infrastructure.

Better Network and Information Security Management. Company can outsource noncritical applications and its data to cloud, where they can run with better performance, which allows the company IT department to focus on critical applications. This also improves company network security and user access management.

Disaster Recovery Backup Management. The successful recover from a disaster depends mainly of the quality and the frequency of backups. Cloud offers much better layered backup strategy. This feature offers to have a better Recovery Point Objective (RPO).

Disaster Recovery Geographic Redundancy. Cloud Providers offer a built-in geographic redundancy in the form of regions and availability zones. This feature

offers to decrease the Recovery Time Objective (RTO). We must note that many of the events in Table 1 are geographically related.

Avoid or eliminate disruption of operations. Some clouds expose a hash (Amazon S3 generate an MD5 hash) when store an object, thus eliminating the need for forensic image verification time.

Increased Availability. The scalability feature of cloud computing facilities allows for greater availability. Redundancy exists all over the cloud environments and on-demand resource capacity increases service availability.

DoS Attack Depreciation. Redundancy and on-demand resource scalability also provide better resilience when facing distributed denial of service attacks, as well as for quicker recovery from serious incidents.

22.3.3 Business Continuity Detriments

This section presents the business continuity detriments for the company when moving into cloud published by authors in [127] for the purpose of this thesis research.

Cloud computing produces many open security issues to be assessed. Migrating company services into cloud moves their data and applications outside of the company security perimeter. This outsourcing opens new security issues and amplifies existing, thus increasing the company's security overall risk. Multi-tenancy, supported by virtualization, is another important security flaw producing new threats and vulnerabilities from inside, the co-tenants. The current isolation facility within clouds i.e. virtualization is weak and can be easily attacked [1]. The problem is even worse in the case of tenants are hosted on the same physical hardware. Thus, CSPs and customers must ensure the customer data and applications are "really" secured and the risks are mitigated to the customer's acceptable level.

Business continuity and Disaster recovery are only one domain of all the domains for CSA's SECaaS [28]. In this section we analyze the security detriments cloud computing offers and are aware that some benefits will also produce detriments. We overview some of the main risks that impact the business continuity together with some solutions that mitigates the risks to acceptable level:

- **Multi-tenant environment**. Although the cloud can offer better protection and defense for the same cost than traditional solutions it has a detriment as well. Different cloud tenants are serious potential threat in shared and multi-tenant environment and especially in the public clouds. This is not the case in the traditional in-house solution even if virtualization techniques are used. Each CSP should develop a methodology to evaluate the tenants and categorize them into categories with trustfulness purposes. This is especially important for IaaS and PaaS where a client can impact more to its own security, but also is threat to other tenants.
- **Heterogeneity, Complexity, Interoperability**. Business continuity depends not only on the effectiveness and correctness of system components, but also on the

interactions among them. Subsystem component heterogeneity leads to difficult interoperability. Number of possible interactions between components increases the system failure probability. Complexity typically relates inversely to security, with greater complexity giving rise to vulnerabilities [97]. Defining security standards for adapters, wrappers, transducers, and data transformation, as well as performance analysis can offer stable system solution and mitigate the risks.

- **Regulatory and Standards Compliance**. A CSP must provide an evidence that meets the standards and regulatory a company needs. Each CSP should permit the regular audits by the customers. A cloud customer should assess the risks and include them into risk acceptance plan if acceptable. If not, the services with unacceptable risks should stay in-house. ISO 27001:2005 covers these issues well in several controls.
- **Loss of Control**. A company must transfer some control of the assets, application, etc. to the CSP. Cloud customers must assure that their CSP can meet SLA requirements, and if not, they must assess the risks and include them into BCP. Also, we suggest to CSPs regulatory to obligate cloud customers to concern about security in SLA agreement.
- **Disaster Recovery - RPO and RTO**. Although the cloud can offer better RPOs and RTOs [129] we assume that maybe CSP had not defined these objectives or if defined they are worse than cloud customers would expect. The cloud customers must be ensured that CSP's RPOs and RTOs are defined in compliance with its own, as well as the CSP can satisfy such defined requirements.
- **Performance challenges**. All cloud computing security solutions and techniques degrade cloud services' performance. Implementing identity and access management, web and email security, intrusion management, [28], as well as monitoring systems, data coloring, and other traditional security services, such as web service security produce data overhead and system latency. They must be considered due to their negative impact to server performance and thereby to the system availability.
- **Data Protection, Privacy and Location**. Although replication produces security benefits in Disaster Recovery and system availability, it produces a security detriment. Thus, along with virtualization, it complicates the access control management and data privacy. Outsourcing only noncritical applications and its data to cloud, if applicable, shall provide the client company with even better data protection and management compared to traditional solutions. CSPs must ensure cloud customers into their operations and privacy assurance. Privacy-protection mechanisms must be embedded in all security solutions [149]. This risk directly impacts the regulatory compliance risk and company business reputation. Auditing and logging tenant's activities can reduce the risk of incidents, as well as including obligations in the SLA agreements. ISO 27001:2005 defines controls for audit and logging, but CSP must also include new controls we propose. In some cases the applications and data might be stored in countries where their judiciary concern and lead to regulatory incompliance. Keeping them in-house or in a hybrid cloud with the appropriate SLA can mitigate the risk.

22.4 Summary

No paper so far has presented business continuity aspects in detail of cloud computing and it challenged us to address the cloud computing model security detriments that depreciate the cloud customer business continuity: data privacy and protection, regulatory and standards compliance, loss of control, data location, heterogeneity, complexity, and interoperability, multi-tenant environment, and disaster recovery - RPO and RTO compliance and effectiveness.

We address cloud computing model security beneficial that improves the business continuity: eliminating downtime, better network and information security management, disaster recovery with both backup management and geographic redundancy. It also avoids or eliminates disruption of operations, increases service availability and DoS attack.

This chapter introduces proposals which minimize the impact to business continuity and the probability of incident scenario for each detriment. These main risks can be assessed appropriately and mitigated to the acceptable level by applying recommendations in these proposals according to matrix for risk level as a function of the business impact and probability of incident scenario [75].

Chapter 23
New Methodologies for On-premise vs Cloud Security Evaluation

Abstract The main question at the beginning was if the cloud concept can become basic ICT choice for the companies. Nowadays the main question is when the cloud will become basic ICT choice for the companies. However, not all companies will migrate their services in the public cloud. Some will keep the services on-premise and others will migrate into their own private clouds. Several security challenges migrating in the cloud are described in Section 22.1. This chapter presents the new methodologies that were published by the authors in [127] and [126] for security evaluation of the security on-premise or in the cloud and cloud service layers. ISO 27001:2005 control objectives are taken as a baseline for the evaluation.

23.1 ISO 27001:2005: On-Premise vs Cloud

This section presents the new methodology that was published by the authors in [127] for security evaluation of the security on-premise or in the cloud.

The first dilemma for IT security and business managers is what is the risk of moving into the cloud. Since previous sections in this part present that all CSP's are at minimum ISO 27001:2005 certified, this section evaluates it and proposes a model to measure the ISO 27001:2005 control objectives importance for both on-premise and cloud solutions. We assess and assign a quantitative metric for each control objective importance. With the qualitative and quantitative analysis we compare the applicability and importance of ISO 27001:2005 control objectives as a general purpose standard, and the fact that the cloud techniques subsume the on-premise ones.

23.1.1 Metric Definition

As the CSP becomes an external party that cloud customer relies, the cloud costomer must transfer some security issues to CSP, but also to increase the domain in SLAs. We define three possible values for the importance of each control objective in ISO 27001:2005, both for on-premise and in the cloud. Table 23.1 shows the explanation of each importance. We omit particular control objectives that has no effect if the services are hosted on-premise or in the cloud, i.e. operational or management control objectives.

#	Description
-1	Transfered partially to SLA and remain as Control Objective
0	Same importance
+1	Control Objective with increased importance

Table 23.1 Control objective importance metrics [127]

23.1.2 Evaluation of Control Objectives Importance

Comparison of the differences among cloud computing versus traditional on-premises computing can be carried through deducing which resources or services are executed by cloud customer or CSP. Such comparison is given in [22]. The responsibilities for all parts of the IT services hosted on-premises are on the resource owner, i.e. the customer. Going from IaaS, through Paas to SaaS cloud service layer, more and more responsibilities are transferred from the cloud customer to the CSP.

We evaluate each control objective importance on-premise and in cloud using the comparison and metric definitions in Table 23.1. According to control classification in [100] for control objectives, management and operational control objectives do not depend if the company services are hosted on-premise or in cloud. For example, the company must define security policy, no matter of information systems' size and type.

23.1.3 Analysis of Control Objectives Importance

The results of the evaluation are presented in Table 23.2. 18 control objectives depreciate their importance, 2 control objectives increase the importance and 7 control objectives retain the importance. We must emphasize that importance depreciation does not mean that a given control objective meaning is decreased or even irrelevant

or that particular control objective should be excluded, but the control objective obligations are somehow be transferred to the CSP, and should be integrated (partially or all controls of a given control objective) into SLA agreement signed between CSP and cloud customer. During the processes of establishing or reviewing ISMS and its improvement, the prospective cloud customer can use this evaluation to select / exclude the controls and control objectives to cover the identified requirements, and to put more effort to control objectives with higher importance.

Control Objective	Value
External parties	+1
Third party service delivery management	+1
Responsibility for assets	-1
Information classification	-1
Secure areas	-1
Equipment security	-1
System planning and acceptance	-1
Protection against malicious and mobile code	-1
Back-up	-1
Network security management	-1
Media handling	-1
Electronic commerce services	-1
Monitoring	-1
User access management	-1
Network access control	-1
Mobile computing and teleworking	-1
Security of system files	-1
Technical Vulnerability Management	-1
Reporting information security events and weaknesses	-1
Compliance with security policies and standards, and technical compliance	-1
Operating system access control	0
Application and information access control	0
Cryptographic controls	0
Management of information security incidents and improvements	0
Information security aspects of business continuity management	0
Compliance with legal requirements	0
Security in development and support processes	0

Table 23.2 Evaluation of ISO 27001:2005 Control Objectives [127]

Fig. 23.2 presents the percentages of control objectives that increase, decrease or retain the level of importance in cloud solution compared to on-premise. We conclude that $2/3$ of control objectives are with depreciated importance in cloud and only 7.41% increased the importance when moving into the cloud. Also, the number of control objectives with depreciated importance is 9 times greater than the one with increased importance.

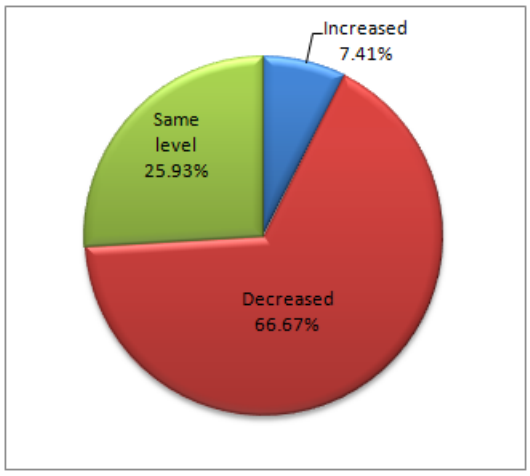

Fig. 23.1 Control objective comparison: On-premises computing versus cloud [127]

23.2 ISO 27001:2005 Evaluation in All Cloud Computing Service Layers

This section presents the new methodology that was published by the authors in [126] for security evaluation of the security on-premise or particular cloud service layers and their average.

In this Chapter we propose a model to measure the ISO 27001:2005 control objectives importance for both on-premise and cloud solutions, due to ISO 27001:2005's generality and because of almost all main CSPs' are ISO 27001:2005 Certified (Table 21.2.2). We assess and assign a quantitative metric for each control objective importance, with details in IaaS, PaaS and SaaS cloud service layers. With the qualitative and quantitative analysis we compare the applicability and importance of ISO 27001:2005 control objectives as a general purpose standard, and the fact that the cloud techniques subsume the on-premise ones.

23.2.1 Metric Definition

We define 6 possible values for the importance of each control objective. Table 23.3 shows the explanation of each importance. We put value "-" if particular control objective has no effect if the services are hosted on-premise or in the cloud, e.g. some operational or management control objectives. Values from 1 to 5 mean that particular control objective has different importance if the services are hosted on-premise or cloud, with given explanation for each importance value.

#	Importance
-	Irrelevant if service is hosted on-premise or cloud
1	Minimal importance (most part moved to SLA)
2	Partial importance
3	Important
4	High importance (almost always)
5	Highest importance (important for each company / IS)

Table 23.3 Control objective importance metrics [126]

23.2.2 Control Objectives Importance Evaluation

Comparing the differences among three service layers of cloud computing versus traditional on-premises computing can be carried through deducing which resources or services are executed by cloud customer and CSP. We can see such comparison on Fig. 1.2 [84]. The resources and services in responsibility of the CSP are shown in green boxes, while those in responsibility of the cloud customer are shown in red. Fig. 1.2 shows that SaaS solution may be used from anywhere and at any time, provided a client (web browser) and internet connection. These features make SaaS software most attractive for SMEs, and require no additional expensive and complex resources and hardware on customer's part.

In Fig. 1.2 is clearly presented that the responsibilities for all parts of the IT services hosted on-premises are on the resource owner, the customer in our case. Going from IaaS, through Paas to SaaS service layer in the cloud computing, more and more responsibilities are transferred from the cloud customer to CSP. Therefore, cloud customers should transfer the security responsibilities to CSP, as well, and thus, most part of ISO 27001:2005 control objectives shall depreciate their values going from On-premise, to Iaas, Paas or SaaS.

Using this comparison, and according to defined metrics in Table 23.3, we evaluate each control objective importance on-premise and in IaaS, PaaS and SaaS cloud service layers. At the beginning, using control classification in [100] for control objectives, that is, (1) Management, (2) Operational and (3) Technical class, we grouped the ISO 27001:2005 control objectives. For (1), management control objectives, we expect that their importance do not depend if the company services are hosted on-premise or in cloud. For example, the company must define security policy, no matter of information systems' size, type, hosting and number. For (2), Operational control objectives, we expect that their importance should be depreciated if hosted in cloud, due to cloud benefits, such as redundancy, scalability, geographic spread, etc. For (3), Technical control objectives, we also expect that their importance should be depreciated if services are hosted in cloud, due to technical benefits offered by the cloud.

The summary results of the evaluation are presented on Table 23.4. The first two columns are the control objectives with their codes, as defined in ISO 27001:2005.

The presented values in Table 23.4 are achieved on our evaluation of ISO 27001:2005s control objective importance factor, both for on-premise and the three cloud service layers, i.e. IaaS, PaaS and SaaS. Details of the evaluation are presented in 23.3. As we can see, many control objectives depreciate their importance, but also, few other control objectives increase. We must emphasize that importance depreciation does not mean that a given control objective meaning is decreased or even irrelevant or that particular control objective should be excluded, but the control objective obligations are somehow be transferred to the CSP, and should be integrated (partially or all controls of a given control objective) into SLA agreement signed between particular CSP and cloud customer. During the processes of establishing the ISMS for ISO certification candidates, or during the processes of reviewing ISMS and its improvement, the prospective cloud customer can use this evaluation to select / exclude the controls and control objectives to cover the identified requirements, and to put more effort and resources to control objectives with higher importance.

We must address that the evaluation is made for a SMEs, having their own information system, network and hardware equipment, since EU industry is mainly composed by SMEs [18].

23.2.3 Control Objectives Importance Analysis

After the evaluation, we proceed to qualitative and quantitative analysis on the results of the evaluation. In the quantitative analysis, we analyze the number of control objectives which importance increased / decreased for on-premise, as well as for each cloud computing service (IaaS, PaaS, SaaS) and their average. From the results shown in Fig. 23.2, we conclude that for each cloud computing service, the number of control objectives with depreciated importance for each cloud service layer (IaaS, PaaS, SaaS) is four to ten times greater than the number of the control objectives with increased importance when moving into the cloud.

In the qualitative analysis, we analyze the sum of control objectives importance value for each cloud service layer and the average, compared to the sum of the importance value of the same control objectives when hosted the services on-premise. From the results, shown in Fig. 23.3, we conclude that for each cloud computing service layer, total sum of control objective importance values into the cloud depreciates compared to on-premise, that is, before moving into the cloud. The percentages of the importance value depreciation are 7.76%, 18.10%, 47.41%, 18.10% for SaaS, IaaS, PaaS and Average, respectively.

We can conclude that both quantitative and qualitative analysis result in the control objectives importance depreciation, in all three cloud service layers, as well as in their average. The depreciation percentages show some paradox, that is, the distribution of control objective importance is not equal to the responsibilities of CSP and cloud customer, especially for SaaS, where it is expected to downgrade the importance near to zero. This is due to generality of ISO 27001:2005 and the fact that

#	Control Objective	On-premise	SaaS	IaaS	PaaS	Avg
5.1	Information security policy	-	-	-	-	-
6.1	Internal organization	-	-	-	-	-
6.2	External parties	1	5	5	5	5
7.1	Responsibility for assets	5	4	4	3	4
7.2	Information classification	5	4	4	3	4
8.1	Prior to employment	-	-	-	-	-
8.2	During employment	-	-	-	-	-
8.3	Termination or change of employment	-	-	-	-	-
9.1	Secure areas	4	2	2	2	2
9.2	Equipment security	4	3	3	3	3
10.1	Operational procedures and responsibilities	-	-	-	-	-
10.2	Third party service delivery management	1	5	5	5	5
10.3	System planning and acceptance	4	3	2	1	2
10.4	Protection against malicious and mobile code	4	4	3	1	3
10.5	Back-up	5	4	3	1	3
10.6	Network security management	5	2	2	1	2
10.7	Media handling	5	4	4	3	4
10.8	Exchange of information	-	-	-	-	-
10.9	Electronic commerce services	5	4	3	1	3
10.10	Monitoring	5	4	3	1	3
11.1	Business requirement for access control	-	-	-	-	-
11.2	User access management	5	4	3	2	4
11.3	User responsibilities	-	-	-	-	-
11.4	Network access control	5	4	3	1	3
11.5	Operating system access control	5	5	5	1	5
11.6	Application and information access control	5	5	5	5	5
11.7	Mobile computing and teleworking	4	5	4	1	3
12.1	Security requirements of information systems	-	-	-	-	-
12.2	Correct processing in applications	-	-	-	-	-
12.3	Cryptographic controls	3	4	4	1	3
12.4	Security of system files	5	4	3	1	3
12.5	Security in development and support processes	4	4	4	1	4
12.6	Technical Vulnerability Management	4	3	2	1	3
13.1	Reporting information security events and weaknesses	4	3	2	1	2
13.2	Management of information security incidents and improvements	5	5	5	5	5
14.1	Information security aspects of business continuity management	5	5	5	5	5
15.1	Compliance with legal requirements	5	5	5	5	5
15.2	Compliance with security policies and standards, and technical compliance	4	3	2	1	2
15.3	Information systems audit considerations	-	-	-	-	-

Table 23.4 Existing CSPs' Security Certification and Accreditation, as well as Security Features [126]

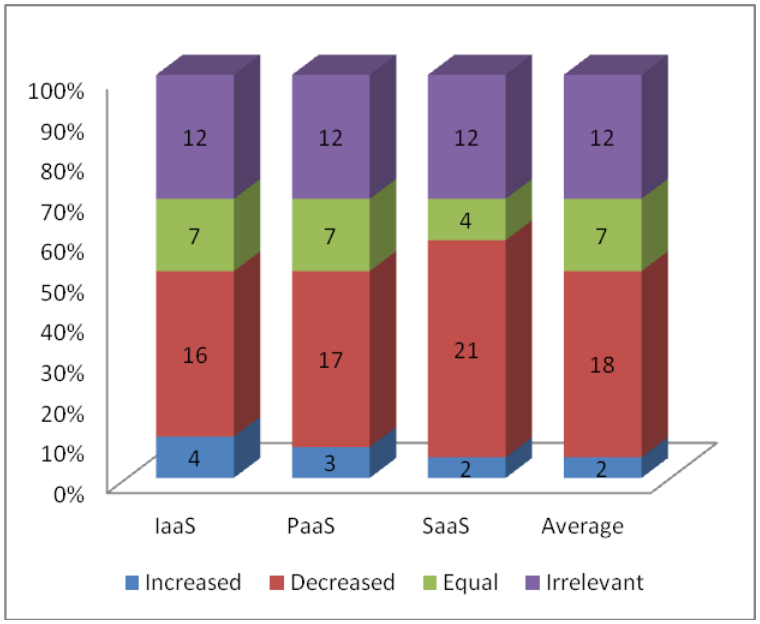

Fig. 23.2 Comparison between On-premises computing versus cloud service layers - IaaS, PaaS and SaaS [126]

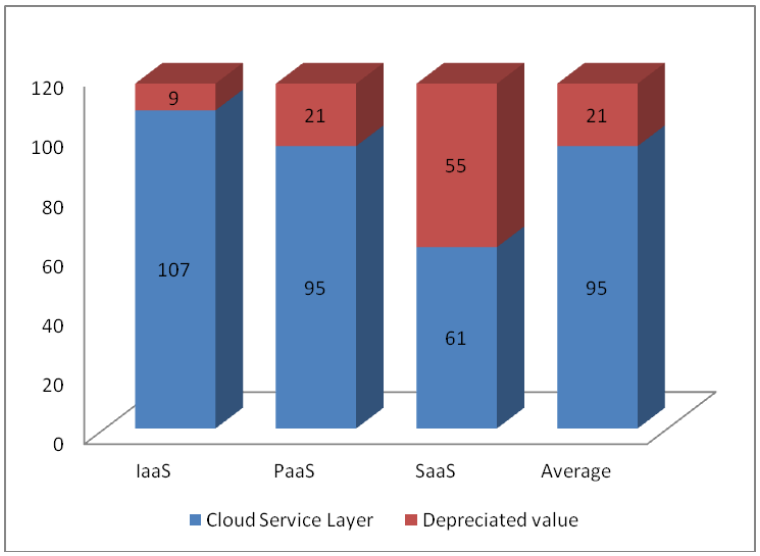

Fig. 23.3 Qualitative analysis on ISO 27001:2005 control objectives when moving into each cloud service layer [126]

cloud customers will still have information and assets on-premise, employees, legal issues, etc.

23.3 ISO 27001:2005 Quantification

Table 23.5 presents the details of importance factor evaluation for each ISO 27001:2005 control objective. The importance factor depends on fact whether the company services are hosted on-premise or in cloud.

#	Control Objective
5.1	**Information security policy**: The organization must define security policy, no matter if the services are hosted on-premise or in cloud. Therefore this control objective is not evaluated.
6.1	**Internal organization**: The information security must be managed within the organization for both solutions. Therefore this control objective is not evaluated.
6.2	**External parties**: The organizations information and information processing facilities are accessed, processed, communicated to, or managed by external parties in each cloud service layer and we evaluate each cloud service layer with 5. For on-premise hosting the organization does not use external party services and we evaluate with 1.
7.1	**Responsibility for assets**: The organization must manage appropriate protection of organizational assets on-premise regardless of company type (evaluate with 5). In Iaas and PaaS the organization transfers some of the assets (reduced to importance 4), and in SaaS most of the application data are completely transferred (reduced to importance 3, as data field is green in Figure 1). However, many other assets like mobile phones, paper accounting documents, usbs, etc the organization must manage.
7.2	**Information classification**: Each organization shall perform an appropriate level of protection to information on-premise (importance is 5). Some of the procedures for information handling are transferred to CSP for IaaS and PaaS, and more for SaaS as data field is green for SaaS.
8.1	**Prior to employment**: The organization must ensure that employees, contractors and third party users understand their responsibilities, and are suitable for the roles they are considered for, and to reduce the risk of theft, fraud or misuse of facilities for both solutions. Therefore this control objective is not evaluated.
8.2	**During employment**: Similar to 8.1 this control objective is not evaluated.
8.3	**Termination or change of employment**: Similar to 8.1 this control objective is not evaluated.

9.1 **Secure areas**: The organization must prevent unauthorized physical access, damage and interference to the organizations premises and information. Some organizations are tenants in secured areas and therefore we evaluate on-premise with 4. The organization transfers most of this facility to CSP (Networking, Storage and Servers are green in Figure 1) and the importance factor is 2 for all cloud service layers.

9.2 **Equipment security**: Similar to 9.1, but some equipment stays on premise and thus the equipment security is important in all cloud service layer.

10.1 **Operational procedures and responsibilities**: The organization must secure the information processing facilities for both solutions. Therefore this control objective is not evaluated.

10.2 **Third party service delivery management**: This control objective is very important for cloud solution as the CSP is external party and in many cases customer depends on third party service delivery (CSPs external parties like CSPs Internet service providers, power supply, etc). A customer must implement the control objective requirements into its BCP. Importance factor for on-premise is evaluated with 1 and for cloud with 5 since the external parties on-premise become third party in each cloud solution.

10.3 **System planning and acceptance**: For this control objective we put 4 for on-premise due to standard generality, i.e. not all organizations possess information systems or update them. Going from IaaS to SaaS the organization shall transfer to CSP with SLA the risk of systems failures. In IaaS the hardware resources are transferred to CSP, in PaaS operating systems and runtime, as well, and in SaaS the applications. Therefore we decrease the importance factor starting from 1 for on-premise and going from IaaS to SaaS.

10.4 **Protection against malicious and mobile code**: Same as 10.3 the importance factor is evaluated to 4 for on-premise due to standard generality. Protection level of the integrity of software and information for On-premise and IaaS solutions is the same. In PaaS solution CSP shall have some procedures to protect the customer (decreased importance by -1), and in SaaS CSP have the full responsibility of the operating systems and the applications (transferred in SLA, i.e. importance factor is 1).

10.5 **Back-up**: Similar evaluation as 10.4, except for on-premise where each organization regardless of its nature, type and size must perform backup on some assets (Legal requirements, Archive, Accounting etc).

10.6 **Network security management**: The organization in on-premise solution must ensure the protection of information in networks and supporting infrastructure (importance factor is 5). IaaS and PaaS are evaluated with 2 as the organization transfers more of the responsibility to CSP (there is network traffic, such as VPN or remote control from the organization to CSP). SaaS is evaluated with 1 solution the organization transfers the responsibility completely to CSP as defined in SLA.

10.7 **Media handling**: Some media are unnecessary for cloud solution, such as backup media. However, not all media are transferred to CSP; for IaaS and PaaS the organization shall prevent smaller number of media than on-premise (importance factor is high 4), and even less for SaaS (importance factor is 3).

10.8 **Exchange of information**: The security of information and software exchanged within an organization and with any external entity is not affected if the services are on-premise or hosted in the cloud. Therefore this control objective is not evaluated.

10.9 **Electronic commerce services**: The value of importance factor for on-premise solution is 5 if the organization uses e-commerce services. The importance in IaaS and PaaS decreases and for SaaS solution the CSP has completely responsibility, similar to 10.5.

10.10 **Monitoring**: The organization transfers the responsibility for monitoring to CSP going from IaaS to SaaS, similar to 10.9

11.1 **Business requirement for access control**: This is high-level control objective and access control policy shall be established regardless the solution. Therefore this control objective is not evaluated.

11.2 **User access management**: The organization transfers more of the responsibility to CSP (in SLA) going from IaaS to SaaS similar to 10.5. The difference is in SaaS since we evaluate it with 2 because the organization has to manage physical access.

11.3 **User responsibilities**: Users have the same responsibilities no matter where the information is. Therefore this control objective is not evaluated.

11.4 **Network access control**: The organization transfers the responsibility for monitoring to CSP going from IaaS to SaaS and therefore this control objective is evaluated similar to 10.5.

11.5 **Operating system access control**: For on-premise, IaaS and PaaS the organization has the responsibility for operating system access control (importance factor is 5). We evaluate the importance of SaaS solution with 1 as the organization transfers the responsibility to CSP as depicted in Figure 1.

11.6 **Application and information access control**: For both the solutions it is the responsibility to the organization for application and information access control. Therefore we evaluate both solutions with importance 5.

11.7 **Mobile computing and teleworking**: Hosting the services in IaaS increases the importance of this control objective. We evaluate this control objective with maximum importance 5 for IaaS. For PaaS CSP transfers come of the responsibilities to CSP and we decrease the importance. For SaaS this control objective shall be included in SLA and therefore we evaluate with 1.

12.1 **Security requirements of information systems**: Security must be an integral part of information systems regardless of the solution. Therefore this control objective is not evaluated.

12.2 **Correct processing in applications**: The organization must prevent errors, loss, unauthorized modification or misuse of information in applications regardless of the solution. Therefore this control objective is not evaluated.

12.3 **Cryptographic controls**: We evaluate on-premise with 3 due to standard generality. We evaluate IaaS and PaaS with increased importance 4 as the organization must implement cryptography to protect the confidentiality, authenticity or integrity of information. For SaaS the CSP has the responsibility.
12.4 **Security of system files**: Similar to 11.4
12.5 **Security in development and support processes**: Similar to 11.5
12.6 **Technical Vulnerability Management**: The organization transfers the responsibility to CSP going from IaaS to SaaS, similar to 10.3.
13.1 **Reporting information security events and weaknesses**: Similar to 12.6
13.2 **Management of information security incidents and improvements**: For both the solutions it is the responsibility to the organization for management of information security incidents with highest importance factor 5.
14.1 **Information security aspects of business continuity management**: There are many business continuity benefits and detriments that the organization must evaluate in its BCP. Therefore both the solutions are evaluated with maximum importance factor 5.
15.1 **Compliance with legal requirements**: For both the solutions it is the maximum responsibility to the organization to comply with the legal requirements. Both the solutions are evaluated with maximum importance 5.
15.2 **Compliance with security policies and standards, and technical compliance**: The organization transfers the responsibility to CSP going from IaaS to SaaS for compliance with security policies and standards, and technical compliance, similar to 13.1.
15.3 **Information systems audit considerations**: This is a high level organizational control objective and therefore this control objective is not evaluated.

Table 23.5: Details of ISO 27001:2005 Control objectives importance evaluation [126]

23.4 Summary

This chapter defines a methodology to quantify the ISO 27001:2005 Requirements grouped in control objectives, comparing on-premise and cloud environments. The evaluation and analysis of ISO 27001:2005 standard result in the importance transfer from cloud customer to CSP. Simultaneously cloud customer must provide a huge effort to implement all control objectives with decreased importance in SLA with its CSP.

We also define a methodology to quantify the ISO 27001:2005 Requirements grouped in control objectives, for on-premise and different cloud service layers. We evaluate that moving into cloud, 12 of 39 control objectives are for management, and are not affected if the services are on-premise or in cloud. Importance factor doesnt

change on average seven Control Objectives, depreciates on 18, and increases on only two of them. Thus, moving into cloud, cloud customers (SME) transfer the importance of the security to its CSP, and expect that their data and applications to be secured. Therefore, and due to emergent security challenges that cloud computing produces, cloud customers must re-evaluate their BCPs.

Chapter 24
New Methodology for Open Source Cloud Security Evaluation

Abstract After the decision to migrate its services in the cloud the company must select on which type of cloud to migrate. The expected dilemma is either the migration to be realized in the public or private, or both and develop hybrid cloud? Further on, should either the migration to be realized in the closed commercial public cloud or in open source cloud solutions. Several open source cloud software solutions offer a possibility to build own private cloud or even a hybrid cloud since many open source cloud solutions offer interfaces to commercial public clouds services. This chapter presents the cloud security challenges evaluation of OpenStack and other open source clouds realized by the authors in [128, 122].

24.1 OpenStack Security Assessment

In this section we present the assessment of the information security challenges in open source cloud softwares focusing on OpenStack cloud solution that the authors published in [128] for the purposes of this thesis research.

Tsai et al. address the system security, networking security, user authentication, and data security as information security issue of cloud computing in [153]. We add and analyze additional security issues in the security assessment.

24.1.1 User Access Management

User access management is based on *Role Based Access Control* (RBAC) in OpenStack. Each role has predefined set of permitted operations. Then roles are assigned to each user. The roles can be assigned when the user is created or editing the existing user profile.

Administrator is a project based role and has a privilege to add or remove an instance, to remove an image and to add a key. *IT security* is global role that per-

mits role holders to quarantine instances. *Project Manager* is default role for project owners and permits the role holders to add or revoke roles to users of the project, to manage network and all the privileges that Administrator role has. *Network Administrator* role allows particular user to allocate and assign publicly accessible IP addresses, and to create and modify firewall rules. *Developer* role is a global role that is assigned to users by default and user assigned with this role can create and download keys.

OpenNebula has the similar RBAC policy. Eucalyptus and CloudStack have more features such as LDAP integration and X.509 certificates.

24.1.2 Network Access Management

OpenStack Nova supports three kinds of networks: Flat Network Manager, Flat DHCP Network Manager, and VLAN Network Manager. The first two kinds of network assign the IP addresses of the subnet that is specified by the user with network administrator role. Therefore this control objective is not satisfied with these two kinds of networks.

In VLAN Network mode, Nova creates a VLAN and bridge for each project. Each project gets a range of private IP addresses that are only accessible from inside the VLAN. Each project gets its own VLAN, Linux networking bridge, and subnet in this mode.

A special VPN instance needs to be created in order for a user to access the instances of the virtual machine in their project. Certificate and key are generated to access the VPN which provides private network segment for instances of the project.

Berger et al. [15] develop a trusted virtual datacenter (TVDc) technology to address the need for strong isolation and integrity guarantees in virtualized, cloud computing environments. It clearly splits the management responsibilities between the cloud and tenant administrators.

All analyzed open source solutions have integrated firewall rules within the controller.

24.1.3 Operating System Access Control

Operating systems of the instances are controlled by the users. Those on the OpenStack physical servers are managed by the system administrator. The volumes, image and storage data are secured with project keys.

OpenNebula offers three authentication for the images: user-password scheme, x509 certificates based authentication and EC2. CloudStack has LDAP based authentication and Eucalyptus has LDAP and secret keys based authentication.

24.1.4 Application and Information Access Control

OpenStack has the application Dashboard for image, volume and instance management. The access is controlled via username and password. Khan et al. [83] propose and implement OpenlD-Authentication-as-a-Service APIs that allows users to use their OpenID Identifiers to log into the OpenStack Dashboard.

Other analyzed open source solutions have integrated LDAP and X.509 certificates.

24.1.5 Mobile Computing and Teleworking

OpenStack ensures information security when using mobile computing and teleworking facilities if it is configured in VLAN Network mode and if a special VPN instance is created by network administrator.

CloudStack has an option for creating VPN, Eucalyptus can create remote desktop, but without establishing VPN. OpenNebula can not create VPN.

24.1.6 Cryptographic Controls

OpenStack does not use cryptographic controls by default. Using OpenID can increase OpenStack overall security [83]. Neverthless, all OpenStack services does not use any SSL / TLS.

Other analyzed solutions provide different usage of cryptography.

24.1.7 Security of System Files

OpenStack ensures the security of customers' system files with instance access keys. The customer data is deleted after instance is shut-downed. The data saved in volumes is protected with access keys as well.

Other analyzed open source solutions have similar features.

24.1.8 Information Security Incident Management

Despite the fact that OpenStack allows cloud service providers to audit logs it provides, it does not possess some internal tool to manage the logs for incident management. However, there are additional tools that provide resource monitoring to

help in incident management. Nice tool is Zenoss [170] which manages OpenStack cloud servers, images, instances etc.

All other analyzed open source solutions have the systems for monitoring, both for users and administrators.

24.1.9 Backup and Disaster Recovery Procedure

OpenStack allows cloud service providers easily to create a snapshot of instances to store the customers data that are in the instances. It has a well designed disaster recovery procedure in case of a disk crash, a network loss, a power cut, etc.

All open source solutions have interfaces to Amazon's AWS providing good solution to backup and disaster recovery.

24.2 Open source Cloud Solution Security Evaluation with OpenStack

This section evaluates the security for open source cloud solutions described in Section 1.4 and compared to the OpenStack solution. The basics of security assessment is explained in Section 24.1 and the evaluation is based on assessment of ISO 27001:2005 [73] control objectives as a guideline for security evaluation.

24.2.1 Security Evaluation Metric Definition

In this section we define the metrics for security evaluation of the open source cloud solutions compared to OpenStack. Table 24.1 describes the metrics used in the evaluation. This assessment addressees only relative measures and is not attended to benchmark all solutions, but rather to analyze the OpenStack solution and its security in comparison to the other open source cloud computing solutions.

The suggested metrics include evaluation with 0 if the same level of security is found compared to the OpenStack solution, and correspondingly 1 or -1 if the security level is better or worse than OpenStack.

24.2.2 Security Evaluation

Open source cloud solutions are evaluated in comparison to OpenStack solution according to the metrics presented in Table 24.1. The evaluation is based on security

#	Description
0	The security level is the same as OpenStack
-1	The solution security level is better than OpenStack
+1	The solution security level is worse than OpenStack

Table 24.1 Metrics for security evaluation in comparison to OpenStack solution [128]

assessment in Section 24.1 and assessing each ISO 27001:2005 control objectives that depend of cloud solution.

The results are analyzed by calculation of the average of obtained values for each control objective and the average value of each analyzed open source cloud solution.

Table 24.2 presents the results of the evaluation. Column *ON* identifies OpenNebula, *CS* identifies CloudStack and *Eu* identifies Eucalyptus opensource cloud solution.

#	Control Objective	ON	CS	Eu	Avg
10.5	Back-up	0	0	0	0
10.6	Network security management	0	0	0	0
10.7	Media handling	0	0	1	0.33
10.10	Monitoring	0	0	0	0
11.2	User access management	0	1	1	0.67
11.4	Network access control	0	0	0	0
11.5	Operating system access control	0	0	0	0
11.6	Application and information access control	1	1	1	1
11.7	Mobile computing and teleworking	-1	0	-1	-0.67
12.3	Cryptographic controls	1	1	1	1
12.4	Security of system files	0	0	0	0
13.1	Reporting information security events and weaknesses	1	1	1	1
14.1	Information security aspects of business continuity management	0	0	0	0
	Average security level evaluation compared to OpenStack	**2**	**4**	**4**	**3.33**

Table 24.2 Security evaluation of open source cloud solutions [128]

24.2.3 Security Evaluation Analysis

Only 13 out of 39 or exactly $1/3$ of the ISO 27001:2005 control objectives depend on cloud solutions. These are presented in Table 24.2.

Figure 24.3 depicts the results of our qualitative analysis of the security evaluation. CloudStack and Eucalyptus achieve the best security with 4 security points ahead of OpenStack, meaning that in 4 out of 13 analyzed relevant control objec-

tives (30.77%) they achieve better security. OpenNebula achieves 2 security points, i.e. achieves better security in 2 out of 13 control objectives (15.4%) ahead of OpenStack.

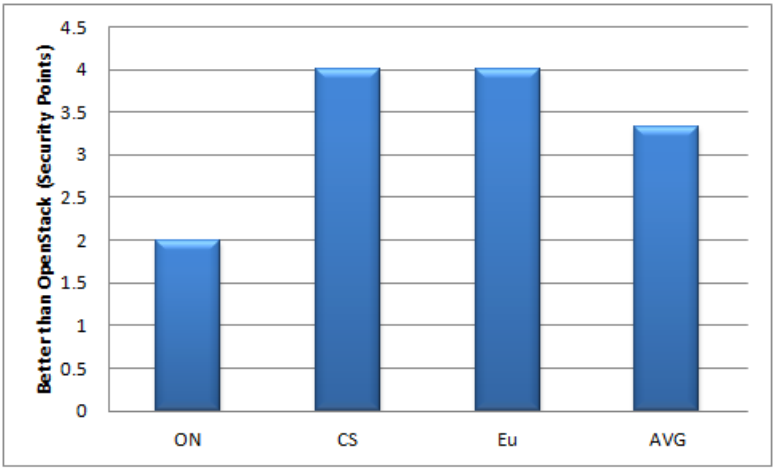

Fig. 24.1 The qualitative analysis of the security evaluation of other open source cloud solutions compared to OpenStack [128]

The conclusion is that OpenStack has the worst level of security.

Figure 24.2 presents another view on the security evaluation comparing open source solutions to OpenStack. It shows the average results of each solution according to the security evaluation compared to the OpenStack security level.

OpenStack achieves equal security level with other open source cloud computing solutions in 7 out of 13 analyzed relevant control objectives (53.85%) and worse security level in 5 relevant control objectives (38.46%).

Only in 1 out of all 13 analyzed relevant control objectives (7.69%) OpenStack achieves better security level than other open source cloud computing solutions. It concerns Mobile computing and teleworking.

24.3 OpenStack security pros and cons

This section presents pros and cons of OpenStack security as a result of our security assessment. A comparison with the other open source cloud solutions is presented.

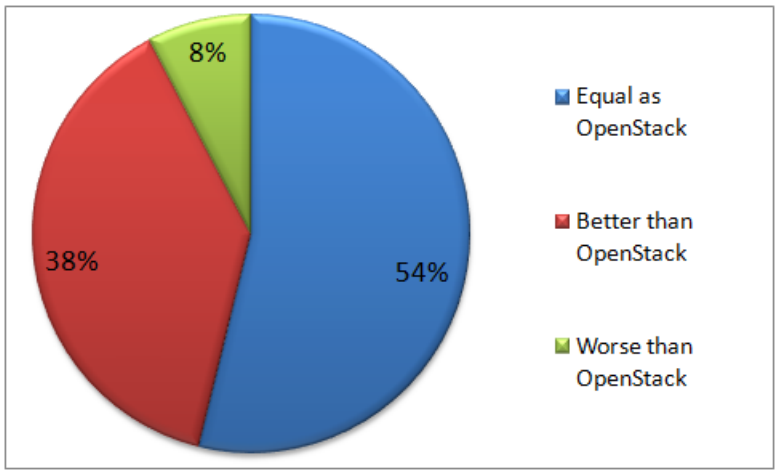

Fig. 24.2 The average of the security evaluation [128]

24.3.1 OpenStack Pros

Positive aspects of security that we found during the security assessment of OpenStack are:

- OpenStack manages user access securely with role based access control;
- OpenStack manages network access securely only if it is deployed in VLAN Network mode;
- OpenStack manages operating system access securely;
- OpenStack ensures information security when using mobile computing and teleworking facilities if it is configured in VLAN Network mode;
- OpenStack ensures the security of customers' system files;
- OpenStack provides log files for incident management but not the tools to process them in real time; and
- OpenStack provides disaster recovery procedure in case of disk crash, network loss a power cut etc.

Comparing with the other open source solutions we can conclude that OpenStack has better security level than CloudStack and Eucalyptus (same security level as OpenNebula) for mobile computing and teleworking facilities if it is configured in VLAN Network mode because it can configure VPN to the instances of virtual machines. For all the other OpenStack security pros the other open source cloud solutions provide equal or even better security level.

24.3.2 OpenStack Cons

Despite the advantages and security benefits, the OpenStack provides several security detriments, such as that OpenStack:

- can be configured in Flat Network Manager or Flat DHCP Network Manager where all instances of all projects are assigned IP addresses from one subnet, which does not provide segregation in networks and is incompliant with ISO 27001:2005;
- does not use cryptographic controls for its web services neither for authentication of web GUI for administration - OpenStack Dashboard; and
- does not provide a tool to monitor the system which is necessary for information security incident management.

Comparing the other open source solutions we can conclude that all of them handles better the security issues that are cons for OpenStack. Even more, for the last two cons all other solutions have better security level (Table 24.2) than OpenStack.

24.4 Security Evaluation of Open Source Clouds

This section evaluates the security assessment realized by the authors in [122] for the purpose of this thesis research based on ISO 27001:2005 [73] control objectives.

24.4.1 Metrics Definition

This section defines the metrics for security evaluation of the open source cloud solutions. Table 24.3 describes the metrics used in the evaluation. The suggested metrics include evaluation with - if the particular control objective does not depend of cloud solution, with -1 if particular solution is not compliant, with 0 or 1 correspondingly if particular solution is partially compliant or partially compliant that can be upgraded to compliant, and with 2 for fully compliant solution.

24.4.2 Security Evaluation and analyses

Open source cloud solutions are evaluated according to the metrics presented in Table 24.3. The evaluation is performed for the purpose of this thesis research based on security assessment in [122] assessing each ISO 27001:2005 control objective. The results are analyzed by calculation of the average value of each analyzed cloud.

Table 24.4 presents the results of the evaluation. Column *OS* identifies OpenStack, *ON* identifies OpenNebula, *CS* identifies CloudStack and *Eu* identifies Euca-

#	Description
-	The particular control objective does not depend of the particular cloud solution
-1	The solution is not compliant with particular control objective
0	The solution is partially compliant with particular control objective
1	The solution is partially compliant but can be upgraded to be compliant with particular control objective
2	The solution is fully compliant with particular control objective

Table 24.3 Metrics for security evaluation [122]

lyptus cloud solution. Only 11 out of 39 or 28.2% of the ISO 27001:2005 control objectives depend on cloud solutions presented in Table 24.4.

Figure 24.3 depicts the results of qualitative analysis of the security evaluation. CloudStack earns average 2 security points, i.e. conforms with all 11 control objective. Eucalyptus and OpenNebula achieve average 1.82 and 1.73 average security points, i.e. more than 1.5 security points, and OpenStack achieves average 1.45.

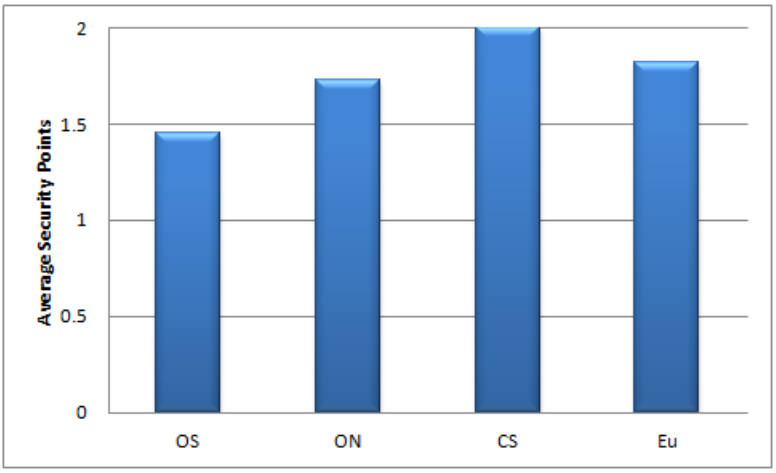

Fig. 24.3 The qualitative analysis of the security evaluation [122]

The conclusion is that CloudStack is the best solution to ISO 27001:2005 conformity ahead of Eucalyptus and OpenNebula. OpenStack is the worst for ISO 27001:2005 conformity.

Figure 24.4 presents another, quantitative view of the security evaluation. It depicts the comparison of security points distribution. As depicted, CloudStack earns 11 times 2 security points and nothing else. Eucalyptus earns 10 times 2 security points and only one 0 security point. OpenNebula has 9 times 2 security points and

#	Control Objective	OS	ON	CS	Eu
5.1	Information security policy	-	-	-	-
6.1	Internal organization	-	-	-	-
6.2	External parties	-	-	-	-
7.1	Responsibility for assets	-	-	-	-
7.2	Information classification	-	-	-	-
8.1	Prior to employment	-	-	-	-
8.2	During employment	-	-	-	-
8.3	Termination or change of employment	-	-	-	-
9.1	Secure areas	-	-	-	-
9.2	Equipment security	-	-	-	-
10.1	Operational procedures and responsibilities	-	-	-	-
10.2	Third party service delivery management	-	-	-	-
10.3	System planning and acceptance	-	-	-	-
10.4	Protection against malicious and mobile code	-	-	-	-
10.5	Back-up	2	2	2	2
10.6	Network security management	2	2	2	2
10.7	Media handling	-	-	-	-
10.8	Exchange of information	-	-	-	-
10.9	Electronic commerce services	-	-	-	-
10.10	Monitoring	1	2	2	2
11.1	Business requirement for access control	-	-	-	-
11.2	User access management	1	1	2	2
11.3	User responsibilities	-	-	-	-
11.4	Network access control	2	2	2	2
11.5	Operating system access control	2	2	2	2
11.6	Application and information access control	1	2	2	2
11.7	Mobile computing and teleworking	2	0	2	0
12.1	Security requirements of information systems	-	-	-	-
12.2	Correct processing in applications	-	-	-	-
12.3	Cryptographic controls	0	2	2	2
12.4	Security of system files	2	2	2	2
12.5	Security in development and support processes	-	-	-	-
12.6	Technical Vulnerability Management	-	-	-	-
13.1	Reporting information security events and weaknesses	1	2	2	2
13.2	Management of information security incidents and improvements	-	-	-	-
14.1	Information security aspects of business continuity management	-	-	-	-
15.1	Compliance with legal requirements	-	-	-	-
15.2	Compliance with security policies and standards, and technical compliance	-	-	-	-
15.3	Information systems audit considerations	-	-	-	-

Table 24.4 Security evaluation of open source cloud solutions [122]

by one times 0 and 1. OpenStack achieves only 6 times 2 security points, 4 times 1 point and only one 0 security points. No one solution gets -1 security point.

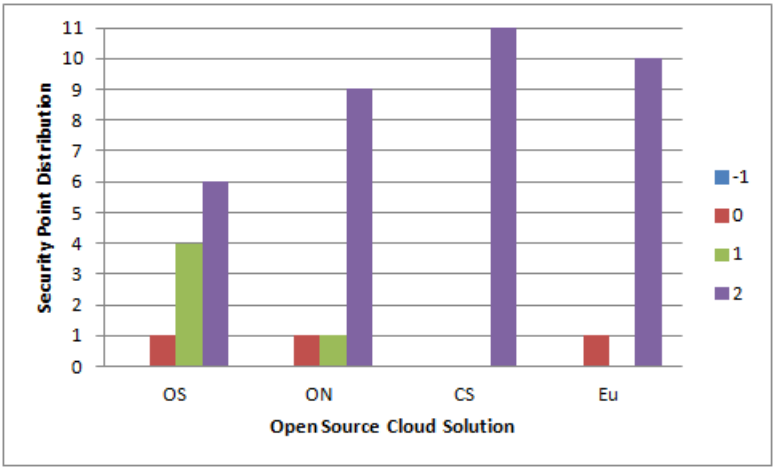

Fig. 24.4 The quantitative analysis of the security evaluation [122]

The same ranking is obtained in this analysis, i.e. CloudStack is the best solution in front of Eucalyptus, OpenNebula and the worst OpenStack solution.

24.5 Summary

IT quality managers have always a dilemma which platform should they select, commercial or open source, on which to drive their services. Both platforms have advantages and detriments. Commercial platforms usually offer more stable, tested, reliable, trusted and less riskiness solutions. However, usually they are more expensive than open source. Open source platforms usually offer acceptable stable solutions. Those solutions are not tested properly and a lot of new versions and sub versions are released in a short period of time. The main advantage of this platform is the small amount of money or even totally free for many products and services. Another huge advantage is the source code that can be used by the customer to redevelop the solution to its requirements.

Both commercial and open source platforms provide solutions to build a private cloud. Almost all commercial and some of the open source provide a solution to build new or interfaces to existing public clouds.

All key commercial cloud providers possess some security certificate as a company. Additionally all of them offer some security services to their customers [126].

Open source solutions provide a small number of security services to the clients or generally do not provide any.

Neither security assessment nor comparative security analysis of the cloud were not performed in the literature so far. We have analyzed the security issues that OpenStack cloud software possess and what other security tools can be integrated to improve its overall security. The evaluation of the security was realized by assessing the relevant control objectives defined by ISO 27001:2005 and comparing it to the other open source cloud computing solutions.

The results of our assessment show that Eucalyptus and CloudStack have integrated the maximum security level in front of OpenNebula. OpenStack has integrated the least security compared to others solutions.

General conclusion of the evaluation is that all open source clouds take care about some level of security. The results of the evaluations show that CloudStack is the best choice of all open source clouds to migrate the services and integrated the maximum security level in its architecture. It conforms with all ISO 27001:2005 11 control objectives that depends of the cloud solution. Eucalyptus and OpenNebula has also reached far in security. OpenStack is worst solution to migrate the services in the manner of security.

Although open source clouds heed the security, the company still have other 28 technical requirements, organizational and management requirements that should be conformed. Also, ISO 27001:2005 defines general requirements, i.e.management responsibility and establishing, managing, reviewing and improving the information security management system.

Glossary

AICPA American Institute of Certified Public Accountants.

COBIT Control Objectives for Information and Related Technology.

DHCP Dynamic Host Configuration Protocol.

ENISA European Network and Information Security Agency.

HIPAA Health Insurance Portability and Accountability Act.

ISACA Information Systems Audit and Control Association.

ISO International Organization for Standardization.

LDAP Lightweight Directory Access Protocol.

NIST National Institute of Standards and Technology.

PCI DSS PCI Data Security Standard.

REST Representational State Transfer (REST) is a style of software architecture for distributed systems.

SOAP Simple Object Access Protocol for exchanging XMLs between Web Services.

X.509 An ITU-T standard for a public key infrastructure (PKI).

References

1. Afoulki, Z., Bousquet, A., Rouzaud-Cornabas, J.: A security-aware scheduler for virtual machines on iaas clouds. Tech. Rep. RR-2011-08 (????), http://www.univ-orleans.fr/lifo/rapports.php?lang=en&sub=sub3
2. AICPA: SSAE 16 (Sep 2011), http://www.aicpa.org/Research/Standards/AuditAttest/Pages/SSAE.aspx
3. Alexander, P.: Information security: a managers guide to thwarting data thieves and hackers. PSI Business Security (2008)
4. Alharkan, T., Martin, P.: Idsaas: Intrusion detection system as a service in public clouds. In: Proceedings of The 3rd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012). pp. 11–17 (2012)
5. Amazon: Amazon elastic block store (Apr 2012), http://aws.amazon.com/ebs/
6. Amazon: Aws (2012), http://aws.amazon.com/ec2/
7. Amdahl, G.M.: Validity of the single-processor approach to achieving large scale computing capabilities. In: AFIPS Conference Proceedings. vol. 30, pp. 483–485. AFIPS Press, Reston. Va., Atlantic City, N.J. (Apr 18-20 1967)
8. An, B.S., Yum, K.H., Kim, E.J.: Scalable and efficient bounds checking for large-scale cmp environments. In: Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques. pp. 193–194. PACT '11, IEEE Computer Society, Washington, DC, USA (2011)
9. Anchev, N., Gusev, M., Ristov, S., Atanasovski, B.: Optimal cache replacement policy for matrix multiplication. In: to be published in ICT Innovations 2012. Springer Berlin / Heidelberg (2012)
10. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (Apr 2010)
11. Atanasovski, B., Ristov, S., Gusev, M., Anchev, N.: Mmcachesim: A highly configurable matrix multiplication cache simulator. In: to be published in ICT Innovations 2012, Web Proceedings, Skopje, Macedonia (2012)
12. Ballard, G., Demmel, J., Holtz, O., Lipshitz, B., Schwartz, O.: Communication-optimal parallel algorithm for strassen's matrix multiplication. In: Proceedinbgs of the 24th ACM symposium on Parallelism in algorithms and architectures. pp. 193–204. SPAA '12, ACM, NY, USA (2012)
13. Baun, C., Kunze, M., Nimis, J., Tai, S.: Cloud Computing: Web-Based Dynamic IT Services. Springer Publishing Company, Incorporated, 1st edn. (2011)
14. Bentley, J.L., McIlroy, M.D.: Engineering a sort function. Softw. Pract. Exper. 23(11), 1249–1265 (Nov 1993)
15. Berger, S., Cáceres, R., Goldman, K., Pendarakis, D., Perez, R., Rao, J.R., Rom, E., Sailer, R., Schildhauer, W., Srinivasan, D., Tal, S., Valdez, E.: Se-

curity for the cloud infrastructure: trusted virtual data center implementation. IBM J. Res. Dev. 53(4), 560–571 (Jul 2009)
16. Buyya, R., Sukumar, K.: Platforms for building and deploying applications for cloud computing. CoRR abs/1104.4379 (2011)
17. Castillo, P.A., Bernier, J.L., Arenas, M.G., Guervós, J.J.M., García-Sánchez, P.: Soap vs rest: Comparing a master-slave ga implementation. CoRR abs/1105.4978 (2011)
18. Catteddu, D., Hogben, G.: Cloud computing risk assessment (2009), http://www.enisa.europa.eu/publications/position-papers/position-papers-at-enisa/act/rm/files/deliverables/cloud-computing-risk-assessment
19. Chen, Y., Paxson, V., Katz, R.H.: Whats new about cloud computing security? Tech. Rep. UCB/EECS-2010-5, EECS Department, University of California, Berkeley (Jan 2010), http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html
20. Chen, Y.T., Cong, J., Reinman, G.: Hc-sim: a fast and exact l1 cache simulator with scratchpad memory co-simulation support. In: Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. pp. 295–304. CODES+ISSS '11, ACM, New York, NY, USA (2011)
21. Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., Molina, J.: Controlling data in the cloud: outsourcing computation without outsourcing control. In: Proceedings of the 2009 ACM workshop on Cloud computing security. pp. 85–90. CCSW '09, ACM, New York, NY, USA (2009)
22. Clayton, C.: Standard cloud taxonomies and windows azure (Sep 2011), http://blogs.msdn.com/b/cclayton/archive/2011/06/07/standard-cloud-taxonomies-and-windows-azure.aspx
23. CloudStack: Cloudstack opens source cloud computing (Apr 2012), http://cloudstack.org
24. CMS: HIPAA (Sep 2011), https://www.cms.gov/HIPAAGenInfo/
25. Conway, P., Kalyanasundharam, N., Donley, G., Lepak, K., Hughes, B.: Cache hierarchy and memory subsystem of the amd opteron processor. IEEE Micro 30(2), 16–29 (Mar 2010)
26. CSA: Cloud Security Alliance Cloud Controls Matrix (CCM), V1.2 (Sep 2011), https://cloudsecurityalliance.org/research/initiatives/cloud-controls-matrix/
27. CSA: CLOUD SECURITY ALLIANCE GROUP CSA-GRC Stack (Sep 2011), http://www.cloudsecurityalliance.org/grcstack.html
28. CSA: Security as a Service V1.0 (Sep 2011), https://cloudsecurityalliance.org/research/working-groups/secaas/
29. CSA: Security Guidance for Critical Areas of Focus in Cloud Computing V2.1 (Sep 2011), https://cloudsecurityalliance.org/research/initiatives/security-guidance/

30. CSA: Top Threats to Cloud Computing (Sep 2011), http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf
31. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the web services web: An introduction to soap, wsdl, and uddi. IEEE Internet Computing 6(2), 86–93 (Mar 2002)
32. DeFlumere, A., Lastovetsky, A., Becker, B.: Partitioning for parallel matrix-matrix multiplication with heterogeneous processors: The optimal solution. In: 21st International Heterogeneity in Computing Workshop (HCW 2012). IEEE Computer Society, IEEE Computer Society, Shanghai, China (May 21, 2012 2012)
33. Djinevski, L., Ristov, S., Gusev, M.: Superlinear speedup in gpu devices. In: to be published in ICT Innovations 2012. Springer Berlin / Heidelberg (2012)
34. Drevet, C.E., Islam, M.N., Schost, E.: Optimization techniques for small matrix multiplication. ACM Comm. Comp. Algebra 44(3/4), 107–108 (2011)
35. Duong, N., Cammarota, R., Zhao, D., Kim, T., Veidenbaum, A.: SCORE: A Score-Based Memory Cache Replacement Policy. In: Emer, J. (ed.) JWAC 2010 - 1st JILP Worshop on Computer Architecture Competitions: cache replacement Championship. Saint Malo, France (2010)
36. Edler, J., Hill, M.D.: Dinero iv trace-driven uniprocessor cache simulator (2012), http://pages.cs.wisc.edu/~markhill/DineroIV/
37. Eklov, D., Hagersten, E.: Statstack: Efficient modeling of lru caches. In: Performance Analysis of Systems Software (ISPASS), 2010 IEEE International Symposium on. pp. 55 –65 (march 2010)
38. Eucalyptus: Eucalyptus cloud (Apr 2012), http://www.eucalyptus.com/
39. Faber, V., Lubeck, O., White, A.: Superlinear speedup of an efficient sequential algorithm is not possible. Parallel Computing 3, 259–260 (1986)
40. FIPS 200: Minimum Security Requirements for Federal Information and Information Systems (Sep 2011), http://csrc.nist.gov/publications/PubsFIPS.html
41. Fraguela, B.B., Doallo, R., Zapata, E.L.: Automatic analytical modeling for the estimation of cache misses. In: Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques. pp. 221–. PACT '99, IEEE Computer Society, Washington, DC, USA (1999)
42. Furht, B., Escalante, A.: Handbook of Cloud Computing. Springer Science+Business Media, 233 Spring Street, New York, NY 10013, USA (2010)
43. Glaskowsky, P.: Nvidias fermi: the first complete gpu computing architecture. Tech. rep., NVIDIA (2009), white Paper
44. Glott, R., Husmann, E., Sadeghi, A., Schunter, M.: Trustworthy clouds underpinning the future internet. In: et al, J.D. (ed.) The future internet. pp. 209–221. Springer-Verlag, Berlin, Heidelberg (????)
45. Google: Google app engine (Apr 2012), https://developers.google.com/appengine/

46. Google: Google cloud storage (Apr 2012), https://developers.google.com/storage/
47. Gupta, R., Tokekar, S.: Proficient pair of replacement algorithms on l1 and l2 cache for merge sort. J. OF COMPUTING 2(3), 171–175 (Mar 2010)
48. Gusev, M., Ristov, S.: Matrix multiplication performance analysis in virtualized shared memory multiprocessor. In: MIPRO, 2012 Proceedings of the 35th International Convention, IEEE Conference Publications. pp. 264–269 (2012)
49. Gusev, M., Ristov, S.: The optimal resource allocation among virtual machines in cloud computing. In: Proceedings of The 3rd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012). pp. 36–42 (2012)
50. Gusev, M., Ristov, S.: Performance gains and drawbacks using set associative cache. Journal of Next Generation Information Technology (JNIT) 3(3) (31 Aug 2012)
51. Gusev, M., Ristov, S.: Superlinear speedup in windows azure cloud. Tech. Rep. IIT:06-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jul 2012)
52. Gusev, M., Ristov, S.: A superlinear speedup region for matrix multiplication. Tech. Rep. IIT:02-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jan 2012)
53. Gusev, M., Ristov, S., Velkoski, G.: Hybrid 2d/1d blocking as optimal matrix-matrix multiplication. In: ICT Innovations 2012. Springer Berlin / Heidelberg (2012)
54. Gustafson, J., Montry, G., Benner, R.: Development of parallel methods for a 1024-processor hypercube. SIAM Journal on Scientific and Statistical Computing 9(4), 532–533 (July 1988)
55. Gustafson, J.L.: Reevaluating amdahl's law. Communication of ACM 31(5), 532–533 (May 1988)
56. Gustafson, J.L.: Fixed time, tiered memory and superlinear speedup. In: Proceedings of the Fifth Distributed Memory Computing Conference. vol. 2, pp. 1256–1260 (1990)
57. Gustafson, J.L.: The consequences of fixed time performance measurement. In: Proceedings of the Hawaii Int. Conf. on System Sciences. vol. 25, p. 113 (1992)
58. Hake, J.F., Homberg, W.: The impact of memory organization on the performance of matrix calculations. Parallel Computing 17, 311–327 (June 1991), http://dx.doi.org/10.1016/S0167-8191(05)80116-4
59. Hao, F., Lakshman, T.V., Mukherjee, S., Song, H.: Secure cloud computing with a virtualized network infrastructure. In: Proc. of the 2nd conf. HotCloud'10. pp. 16–16. USENIX Ass., USA (2010)
60. Haque, M.S., Peddersen, J., Janapsatya, A., Parameswaran, S.: Dew: a fast level 1 cache simulation approach for embedded processors with fifo replacement policy. In: Proceedings of the Conference on Design, Automation and Test in Europe. pp. 496–501. DATE '10, European Design and Automation Association, 3001 Leuven, Belgium, Belgium (2010)

61. He, L., Sun, Y., Zhang, C.: Adaptive Subset Based Replacement Policy for High Performance Caching. In: Emer, J. (ed.) JWAC 2010 - 1st JILP Worshop on Computer Architecture Competitions: cache replacement Championship. Saint Malo, France (2010)
62. Hennessy, J.L., Patterson, D.A.: Computer Architecture: a Quantitative Approach. Morgan Kaufmann, 4 edn. (2006)
63. Hennessy, J.L., Patterson, D.A.: Computer Architecture, Fifth Edition: A Quantitative Approach (2012)
64. Hu, F., Qiu, M., Li, J., Grant, T., Tylor, D., McCaleb, S., Butler, L., Hamner, R.: A review on cloud computing: Design challenges in architecture and security. CIT 19(1), 25–55 (2011)
65. Hwang, K., Li, D.: Trusted cloud computing with secure resources and data coloring. IEEE Internet Computing pp. 14–22 (2010)
66. IBM: Deep blue supercomputer (May 1997), http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/
67. Ibrahim, A.S., Hamlyn-Harris, J., Grundy, J.: Emerging security challenges of cloud virtual infrastructure. In: Proc. of the Asia Pacific Cloud Workshop 2010 (co-located with APSEC2010), Sydney (2010)
68. Ibrahim, A.S., Hamlyn-Harris, J.H., Grundy, J., Almorsy, M.: Cloudsec: A security monitoring appliance for virtual machines in the iaas cloud model. In: Samarati, P., Foresti, S., Hu, J., Livraga, G. (eds.) NSS. pp. 113–120. IEEE (2011)
69. Intel: Inside intel core microarchitecture and smart memory access (Aug 2012), software.intel.com/file/18374/, white Paper
70. Iosup, A., Ostermann, S., Yigitbasi, M., Prodan, R., Fahringer, T., Epema, D.: Performance analysis of cloud computing services for many-tasks scientific computing. Par. and Dist. Syst., IEEE Trans. on 22(6), 931–945 (2011)
71. ISACA: Cobit 4.1 (Sep 2011), http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx
72. Ishii, Y., Inaba, M., Hiraki, K.: Cache Replacement Policy Using Map-based Adaptive Insertion. In: Emer, J. (ed.) JWAC 2010 - 1st JILP Worshop on Computer Architecture Competitions: cache replacement Championship. Saint Malo, France (2010)
73. ISO/IEC 27001:2005: Information security management systems - requirements (2012), http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=42103
74. ISO/IEC 27002:2005: Code of Practice for Information Security Management (Sep 2011), http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50297
75. ISO/IEC 27005:2011: Information security risk management (2012), http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=56742
76. Jackson, K.R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H.J., Wright, N.J.: Performance analysis of high performance

computing applications on the amazon web services cloud. In: Proc. of the IEEE CLOUDCOM '10. pp. 159–168. IEEE Computer Society, USA (2010)
77. Jaleel, A., Cohn, R.S., Luk, C.K., Jacob, B.: Cmpsim: A pin-based on-the-fly multi-core cache simulator. In: The Fourth Annual Workshop on Modeling, Benchmarking and Simulation (MoBS), co-located with ISCA'2008 (2008)
78. Jaleel, A., Theobald, K.B., Steely, Jr., S.C., Emer, J.: High performance cache replacement using re-reference interval prediction (rrip). SIGARCH Comput. Archit. News 38(3), 60–71 (Jun 2010)
79. Janapsatya, A., Ignjatović, A., Peddersen, J., Parameswaran, S.: Dueling clock: adaptive cache replacement policy based on the clock algorithm. In: Proceedings of the Conference on Design, Automation and Test in Europe. pp. 920–925. DATE '10 (2010)
80. Jenks, S.: Multithreading and thread migration using mpi and myrinet. In: Proc. of the Parallel and Distrib. Computing and Systems. PDCS'04 (2004)
81. Jensen, M., Schwenk, J., Gruschka, N., Iacono, L.L.: On technical security issues in cloud computing. In: Proceedings of the 2009 IEEE International Conference on Cloud Computing. pp. 109–116. CLOUD '09, IEEE Computer Society, Washington, DC, USA (2009)
82. Juric, M.B., Rozman, I., Brumen, B., Colnaric, M., Hericko, M.: Comparison of performance of web services, ws-security, rmi, and rmi-ssl. J. Syst. Softw. 79(5), 689–700 (May 2006)
83. Khan, R., Ylitalo, J., Ahmed, A.: Openid authentication as a service in openstack. In: Information Assurance and Security (IAS), 2011 7th International Conference on. pp. 372 –377 (dec 2011)
84. Kiroski, K., Gusev, M., Kostoska, M., Ristov, S.: Modifications and improvements on cen/bii profiles. In: Kocarev, L. (ed.) ICT Innovations 2011, Advances in Intelligent and Soft Computing, vol. 150, pp. 395–404. Springer Berlin / Heidelberg (2012)
85. Koh, Y., Knauerhase, R., Brett, P., Bowman, M., Wen, Z., Pu, C.: An analysis of performance interference effects in virtual environments. In: Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on. pp. 200 –209 (april 2007)
86. Lindholm, E., Nickolls, J., Oberman, S., Montrym, J.: Nvidia tesla: A unified graphics and computing architecture. IEEE Micro 28(2), 39–55 (Mar 2008)
87. Lira, J., Molina, C., González, A.: Lru-pea: a smart replacement policy for non-uniform cache architectures on chip multiprocessors. In: Proceedings of the 2009 IEEE international conference on Computer design. pp. 275–281. ICCD'09, IEEE Press, Piscataway, NJ, USA (2009)
88. Membrane: Load balancing http and web services (Jan 2012), `http://www.membrane-soa.org/soap-loadbalancing.htm`
89. MICROSOFT: Information Security Management System for Microsoft Cloud Infrastructure (2010), `http://www.globalfoundationservices.com/security/documents/InformationSecurityMangSysfor/MSCloudInfrastructure.pdf`

90. Microsoft: Microsoft hyper-v server 2008 r2 (Jan 2012), http://www.microsoft.com/en-us/server-cloud/hyper-v-server/
91. Microsoft: Microsoft live (Apr 2012), http://www.live.com/
92. Microsoft: Microsoft windows azure (Apr 2012), http://www.windowsazure.com/en-us/
93. Mizouni, R., Serhani, M., Dssouli, R., Benharref, A., Taleb, I.: Performance evaluation of mobile web services. In: ECOWS 2011. pp. 184 –191 (2011)
94. Moses, J., Aisopos, K., Jaleel, A., Iyer, R., Illikkal, R., Newell, D., Makineni, S.: Cmpsched$im: Evaluating os/cmp interaction on shared cache management. In: Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on. pp. 113 –122 (april 2009)
95. MSDN: Web service benefits (Mar 2012), http://msdn.microsoft.com/en-us/library/cc508708.aspx
96. Nickolls, J., Dally, W.J.: The gpu computing era. IEEE Micro 30(2), 56–69 (Mar 2010)
97. NIST SP800-144: NIST DRAFT Guidelines on Security and Privacy in Public Cloud Computing (Sep 2011), http://csrc.nist.gov/publications/PubsSPs.html
98. NIST SP800-145: The NIST Definition of Cloud Computing (Jyl 2012), http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
99. NIST SP800-39: Managing Information Security Risk (Sep 2011), http://csrc.nist.gov/publications/PubsSPs.html
100. NIST SP800-53: REVISION 3 (Sep 2011), http://csrc.nist.gov/publications/PubsSPs.html
101. NVIDIA: Cuda programming guide (Aug 2012), http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf
102. NVIDIA: Nvidias next generation cuda compute architecture: Kepler gk110 (2012), white Paper
103. OpenMP: (2012), https://computing.llnl.gov/tutorials/openMP/
104. OpenNebula: Opennebula cloud software (Apr 2012), http://Opennebula.org
105. Openstack: Openstack dual node (????), http://docs.stackops.org/display/documentation/Dual+node+deployment, [retrieved: May, 2012]
106. Openstack: Openstack compute (Feb 2012), http://openstack.org/projects/compute/
107. Openstack: Openstack network (Aug 2012), http://docs.openstack.org/trunk/openstack-compute/admin/content/libvirt-flat-networking.html
108. Openstack: Openstack setup (Aug 2012), http://docs.openstack.org/cactus/openstack-compute/starter/content/Introduction-d1e390.html

109. Padhy, R.P., Patra, M.R., Satapathy, S.C.: Windows Azure Paas Cloud: An Overview. International Journal of Computer Application 1, 109–123 (Feb 2012)
110. Patti, D., Spadaccini, A., Palesi, M., Fazzino, F., Catania, V.: Supporting undergraduate computer architecture students using a visual mips64 cpu simulator. Education, IEEE Transactions on 55(3), 406 –411 (aug 2012)
111. PCI Security Standards: PCI DSS v2.0 (Sep 2011), https://www.pcisecuritystandards.org/security_standards/
112. Pimple, M., Sathe, S.: Architecture aware programming on multi-core systems. International Journal of Advanced Computer Science and Applications (IJACSA) 2, 105–111 (2011)
113. Playne, D.P., Hawick, K.A.: Comparison of gpu architectures for asynchronous communication with finite-differencing applications. Concurrency and Computation: Practice and Experience 24(1), 73–83 (2012), http://dblp.uni-trier.de/db/journals/concurrency/concurrency24.html#PlayneH12
114. Pohorec, S., Zorman, M.: The challenges of the move from the desktop to the cloud. In: Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on. pp. 119 – 124 (june 2011)
115. Qureshi, M.K., Jaleel, A., Patt, Y.N., Steely, S.C., Emer, J.: Adaptive insertion policies for high performance caching. SIGARCH Comput. Archit. News 35(2), 381–391 (Jun 2007)
116. Rajan, S., Jairath, A.: Cloud computing: The fifth generation of computing. In: Communication Systems and Network Technologies (CSNT), 2011 International Conference on. pp. 665 –667 (june 2011)
117. Ramgovind, S., Eloff, M., Smith, E.: The management of security in cloud computing. In: Information Security for South Africa (ISSA), 2010. pp. 1 –7 (aug 2010)
118. Ravindran, R., Moona, R.: Retargetable cache simulation using high level processor models. Aust. Comput. Sci. Commun. 23(4), 114–121 (Jan 2001)
119. Ristov, S.: Analysis of Web Service Security and its Impact on Web Server Performance. Master's thesis, University Sts Cyril and Methodius, Faculty of Electrical Engineering and Information Technologies, Macedonia (May 2011)
120. Ristov, S., Gusev, M.: Achieving maximum performance for matrix multiplication using set associative cache. In: The 8th Int. Conf. on,Computing Technology and Information Management (ICCM2012), IEEE Conference Publications. ICNIT '12, vol. 2, pp. 542–547 (2012)
121. Ristov, S., Gusev, M.: Evaluating a superlinear speedup for matrix multiplication. Tech. Rep. IIT:03-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jan 2012)
122. Ristov, S., Gusev, M.: Open source cloud security audit. Tech. Rep. IIT:08-12, University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jul 2012)
123. Ristov, S., Gusev, M.: Performance gains and drawbacks in multiprocessor using set associative cache. Tech. Rep. IIT:10-12, University Ss Cyril and

Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering (Jul 2012)
124. Ristov, S., Gusev, M.: Superlinear speedup for matrix multiplication. In: Information Technology Interfaces, Proceedings of the ITI 2012 34th International Conference on. pp. 499–504 (2012)
125. Ristov, S., Gusev, M., Kostoska, M.: Information security management system for cloud computing. In: ICT Innovations 2011, Web Proceedings, Skopje, Macedonia (2011)
126. Ristov, S., Gusev, M., Kostoska, M.: Cloud computing security in business information systems. International Journal of Network Security & Its Applications (IJNSA) 4(2), 75–93 (2012)
127. Ristov, S., Gusev, M., Kostoska, M.: A new methodology for security evaluation in cloud computing. In: MIPRO, 2012 Proc. of the 35th Int. Convention, IEEE Conference Publications. pp. 1808–1813 (2012)
128. Ristov, S., Gusev, M., Kostoska, M.: Security assessment of openstack open source cloud solution. In: Proceedings of the 7th South East European Doctoral Student Conference (DSC2012) (2012)
129. Ristov, S., Gusev, M., Kostoska, M., Kiroski, K.: Business continuity challenges in cloud computing. In: ICT Innovations 2011, Web Proceedings, Skopje, Macedonia (2011)
130. Ristov, S., Gusev, M., Kostoska, M., Kjiroski, K.: Virtualized environments in cloud can have superlinear speedup. In: ACM Proceedings of 5th Balkan Conference of Informatics (BCI2012) (2012)
131. Ristov, S., Gusev, M., Osmanovic, S., Rahmani, K.: Optimal resource scaling for hpc in windows azure. In: to be published in ICT Innovations 2012, Web Proceedings, Skopje, Macedonia (2012)
132. Ristov, S., Gusev, M., Velkoski, G.: Message transformation to gain maximum web server performance in cloud computing. In: Proceedings of the Conference of Informatics and Information Technology CiiT (2012)
133. Ristov, S., Gusev, M., Velkoski, G.: A middleware strategy to survive peak loads in cloud. In: Proceedings of the Conference of Informatics and Information Technology CiiT (2012)
134. Ristov, S., Stolikj, M., Ackovska, N.: Awakening curiosity - hardware education for computer science students. In: MIPRO, 2011 Proceedings of the 34th International Convention, IEEE Conference Publications. pp. 1275 –1280 (may 2011)
135. Ristov, S., Tentov, A.: Performance comparison of web service security on windows platform message size vs concurrent users. In: X International Conference ETAI 2011 (2011)
136. Ristov, S., Tentov, A.: Security based performance issues in agent-based web services integrating legacy information systems. In: CEUR Workshop Proceedings. WASA 2011, vol. 752, pp. 45–51 (2011)
137. Ristov, S., Tentov, A.: Performance impact correlation of message size vs. concurrent users implementing web service security on linux platform. In:

ICT Innovations 2011. Advances in Intelligent and Soft Computing, vol. 150, pp. 367–377. Springer Berlin / Heidelberg (2012)
138. Ristov, S., Velkoski, G., Gusev, M., Kjiroski, K.: Compute and memory intensive web service performance in the cloud. In: to be published in ICT Innovations 2012. Springer Berlin / Heidelberg (2012)
139. SalesForce: Salesforce sales cloud (Apr 2012), http://www.salesforce.com/
140. Shi, Y.: Reevaluating amdahl's law and gustafson's law. Tech. Rep. MS:38-24, Computer Sciences Department, Temple University (Oct 1996)
141. Site, S.: Top 500 (Aug 2012), http://www.top500.org/
142. So, B., Ghuloum, A.M., Wu, Y.: Optimizing data parallel operations on many-core platforms. In: First Workshop on Software Tools for Multi-Core Systems (STMCS). pp. 66–70 (2006)
143. SoapUI: Soapui functional testing tool for web service testing (Jan 2012), http://www.soapui.org/
144. Srirama, S.N., Jarke, M., Prinz, W.: A performance evaluation of mobile web services security. CoRR abs/1007.3644 (2010)
145. Stolikj, M., Ristov, S., Ackovska, N.: Challenging students software skills to learn hardware based courses. In: Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on. pp. 339 –344 (june 2011)
146. Sun, X.H., Ni, L.M.: Another view on parallel speedup. In: Proceedings of Supercomputing '90. pp. 324–333 (Nov 1993)
147. Sun, X.H., Ni, L.M.: Scalable problems and memory-bounded speedup. Journal of Parallel and Distributed Computing 19(1), 27–37 (1993)
148. Suzumura, T., Takase, T., Tatsubori, M.: Optimizing web services performance by differential deserialization. In: Proc. of the IEEE Int. Conf. on Web Services. pp. 185–192. ICWS '05, IEEE Computer Society, USA (2005)
149. Takabi, H., Joshi, J.B.D., Ahn, G.: Security and privacy challenges in cloud computing environments. IEEE J. of Security & Privacy 8(6), 24–31 (2010)
150. Tao, J., Kunze, M., Nowak, F., Buchty, R., Karl, W.: Performance advantage of reconfigurable cache design on multicore processor systems. International Journal of Parallel Programming 36(3), 347–360 (Jun 2008)
151. Taylor, M., Guo, C.J.: Data integration and composite business services, part 3: Build a multi-tenant data tier with access control and security (Jan 2012), http://www.ibm.com/developerworks/data/library/techarticle/dm-0712taylor/
152. Tripathi, S., Abbas, S.Q.: Performance comparison of web services under simulated and actual hosted environments. Int. J. of Computer Applications 11(5), 20–23 (Dec 2010), published By Foundation of Computer Science
153. Tsai, C.L., Lin, U.C.: Information security of cloud computing for enterprises. AISS: Advances in Information Sciences and Service Sciences 3(1), 132–142 (2011), http://www.aicit.org/aiss/global/ppl.html?jname=AISS&yr=2011&vnidx=6

154. Tsilikas, G., Fleury, M.: Matrix multiplication performance on commodity shared-memory multiprocessors. In: International Conference on Parallel Computing in Electrical Engineering, PARELEC 2004. pp. 13 – 18 (sept 2004)
155. Valgrind: Valgrind (????), http://valgrind.org/, [retrieved: May, 2012]
156. vmware: Virtualization basics (????), http://www.vmware.com/virtualization/virtual-machine.html, [retrieved: May, 2012]
157. VMware: VMware and x86 virtualization (????), http://forums.techarena.in/guides-tutorials/1104460.htm, [retrieved: May, 2012]
158. VMware: VMware esxi 4 (Jan 2012), http://communities.vmware.com/community/vmtn/server/vsphere/esxi
159. vmware: Vmware esxi 5.0 @ONLINE (Jan 2012), http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html
160. VMware: Vmware vcloud (Apr 2012), http://www.vmware.com/products/vcloud/overview.html
161. Volkov, V.: Better performance at lower occupancy. In: Proceedings of GPU Technology Conference (GTC 2010) (2010)
162. Voras, I., Mihaljevic, B., Orlic, M.: Criteria for evaluation of open source cloud computing solutions. In: Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on. pp. 137 –142 (june 2011)
163. W3C: Web services architecture (Aug 2012), http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/
164. Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., Demmel, J.: Optimization of sparse matrix-vector multiplication on emerging multicore platforms. Parallel Comput. 35(3), 178–194 (Mar 2009)
165. cpu world: Amd opteron(tm) 8347 @ONLINE (Jan 2012), http://www.cpu-world.com/CPUs/K10/TYPE-Third\%20Generation\%20Opteron.html
166. cpu world: Amd phenom(tm) x4 9550 @ONLINE (Jan 2012), http://www.cpu-world.com/CPUs/K10/AMD-Phenom\%20X4\%209550\%20-\%20HD9550WCJ4BGH\%20(HD9550WCGHBOX).html
167. cpu world: Intel(r) core(tm)2 quad cpu q9400 (May 2012), http://www.cpu-world.com/sspec/SL/SLB6B.html, [retrieved: May, 2012]
168. cpu world: Intel(r) xeon(r) cpu x5680 (Jan 2012), http://www.cpu-world.com/CPUs/Xeon/Intel-Xeon\%20X5680\%20-\%20AT80614005124AA\%20(BX80614X5680).html
169. Xu, C., Chen, X., Dick, R.P., Mao, Z.M.: Cache contention and application performance prediction for multi-core systems. In: ISPASS'10. pp. 76–86 (2010)
170. Zenoss: Zenoss service dynamics (Mar 2012), http://www.zenoss.com/

171. Zhang, K., Wang, Z., Chen, Y., Zhu, H., Sun, X.H.: Pac-plru: A cache replacement policy to salvage discarded predictions from hardware prefetchers. In: Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. pp. 265–274. CCGRID '11, IEEE Computer Society, Washington, DC, USA (2011)
172. Zwick, M., Durkovic, M., Obermeier, F., Bamberger, W., Diepold, K.: Mcccsim - a highly configurable multi core cache contention simulator. Tech. rep., Lehrstuhl fr Datenverarbeitung, TU Mnchen (2009)

Index