Web caching and content switching for performance of content delivery

Conference Paper · November 2015

DOI: 10.1109/TELFOR.2015.7377633

3 authors:

Ace Dimitrievski
Ss. Cyril and Methodius University in Skopje
20 PUBLICATIONS **153** CITATIONS
SEE PROFILE

Vladimir Trajkovik
Ss. Cyril and Methodius University in Skopje
278 PUBLICATIONS **1,708** CITATIONS
SEE PROFILE

Sonja Filiposka
Ss. Cyril and Methodius University in Skopje
140 PUBLICATIONS **840** CITATIONS
SEE PROFILE

Web caching and content switching for performance of content delivery

Ace Dimitrievski, *Member, IEEE,* prof. Vladimir Trajkovik, and prof. Sonja Filiposka

Abstract — **With the emergence of web 2.0 the web has evolved from informational posting board to rich application platform. Web applications have replaced desktop applications from office productivity to 3d rendering. In this paper we'll present issues that degrade user web experience and increase demand for resource as well as techniques for mitigating those issues. We'll specially focus on large legacy web sites with legacy infrastructure. The example that we will work on is the main site of a large international organization.**

Keywords —**Content delivery network (CDN), Content switching, Load balancing, Web**

I. Introduction

In this paper we will present several technologies to optimize performance and minimize slow page loads. As this is a broad subject, although most of what we will present can be applied in various web hosting scenarios, the focus of the paper will be on heterogeneous environments with large presence of legacy code and legacy web technologies for large enterprise web sites.

The pace of innovations in the web technologies can create a challenge for many organizations and corporations to keep up. When web sites are of a large scale and are built from many components it is challenging to make sure the entire site uses the latest technologies, instead a more common scenario is that the legacy code and infrastructure starts to get stretched, developers change parts of the applications to introduce new features such as better forms, better look and feel, ajax, responsive design, etc. On top of this as the world gets more connected to the Internet the user base increases. All this brings to the forefront the scalability challenges.

Performance for the sake of scalability is rightly focused on the backend. Database tuning, replicating architectures, customized data caching, and so on, allow web servers to handle a greater number of requests. This gain in efficiency translates into reductions in hardware costs, data center rack space, and power consumption [1].

In the next section we'll briefly touch on the related work on the subject. Next we make the analysis of the challenges and will present our concrete scenario. In the following section we will show the design of our solution. The section V presents the performance evaluation, here we present how the effect on user experience and resource utilization. The last section concludes the paper.

Ace Dimitrievski is with the Office of Information and Communication Technologies of the United Nations Secretariat (email: ace.dimitrievski@gmail.com)

Vladimir Trajkovik is professor at Faculty of Information Sciences and Computer Engineering, University of Ss. Cyril and Methodius, Karpos II b.b., 1000 Skopje, R. Macedonia (e-mail: trvlado@finki.ukim.mk).

Sonja Filiposka is professor at the Faculty of Information Sciences and Computer Engineering, University of Ss. Cyril and Methodius, Karpos II b.b., 1000 Skopje, R. Macedonia (e-mail: sonja.filiposka@finki.ukim.mk)

II. Related Work

Performance of web content delivery is known engineering challenge. The basic techniques of caching and load balancing are also addressed in research. In [6] we can see an overview and comparison of load balancing approaches for web servers. Several approaches are presented including the applicable scenarios as well as pros and cons for each. A research of large scale load balancing was presented in [7]. In this paper the authors present a novel class of algorithms for distributed load balancing. Web caching has also seen some novel research approaches. In [8] authors use artificial neural network, and particle swarm optimization and conclude that such techniques can yield improvements in performance of web cache. By using historical patterns and dynamic deviations in [9] is presented a system that performs favorably versus the popular least recently used algorithm.

III. Analysis

In this section we present in greater detail the performance issue of legacy applications as they are extended to produce more of the new technologies.

The main web site of the organization is typical example of organic growth beginning from the early days of the web. We should note that this paper only deals with the www subdomain, many other sites are hosted in different subdomain which benefit from this approach but we only use the www and its sub-sites to illustrate our research. As a complex multinational organization with offices around the globe and many departments the web presence reflected this very complex, hard to structure nature.

In the days of ASP web development a solution to use the performance of the Unix systems with this new web language was found in the ChilliSoft platform, compiled for Sun Solaris OS. This software was later renamed to Sun ONE Active Server Pages [2]. This platform currently doesn't exist and even the software cannot be found or recompiled. With most of the codebase running on such legacy platform which became a bottleneck, growing demand of the web expansion and growing user base as the world became more connected scalability was growing concern.

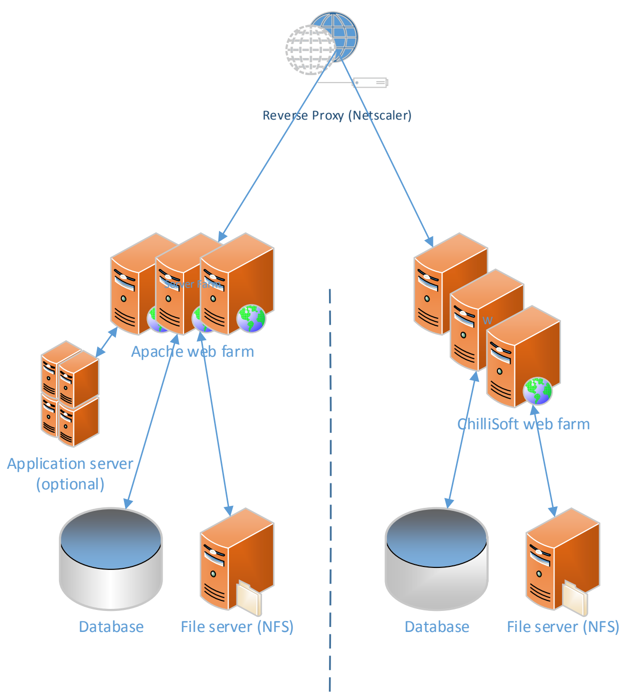

Fig. 1. Simplified web farms with reverse proxy

The original state can be represented by Fig. 1. Here we see that web sites hosting is split on legacy ASP hosting and Apache farm. In this scenario the only way to offload the legacy infrastructure is to rewrite parts of the application in newer technologies. Those parts would have to be data independent since the database is not shared and neither is the file share.

Additional problem that we faced is the obsolescence of the hardware and difficulty in procuring replacement hardware. As the software cannot be compiled for newer operating systems on new hardware scalability is limited to extending unsupported legacy hardware and software.

In order to improve performance and reduce cost we will utilize caching and content switching as the main approaches to address slow page loads. As part of the content switching strategy we implemented Internet Information Services (IIS) web farm and configured to work with ASP pages. As the code was written for alternative technology which had some notable differences the web applications couldn't be transferred without modifications.

Although more optimizations can be done on the application level by optimizing the code as shown in [1], here we'll focus solely on infrastructure approach.

IV. Design

In this section we propose two group of techniques for improving performance and reducing load on the legacy systems. The first technique is caching and the second one is content switching.

A. Caching

Caching is a technique of using a faster medium for data retrieval to temporary store data objects from slower medium. In the context of web services caching can be positioned in multiple places. We use three caching techniques on separate layers of the web request (see Fig. 2):

1. Web caching by using reverse proxy. In this layer entire web pages are cached on the reverse proxy / load balancer device.
2. Database caching. We optimize on this layer by caching database objects using Redis.
3. Client browser cache. This is accomplished by modifying the HTTP headers to make pages cacheable and improve duration of page validity.

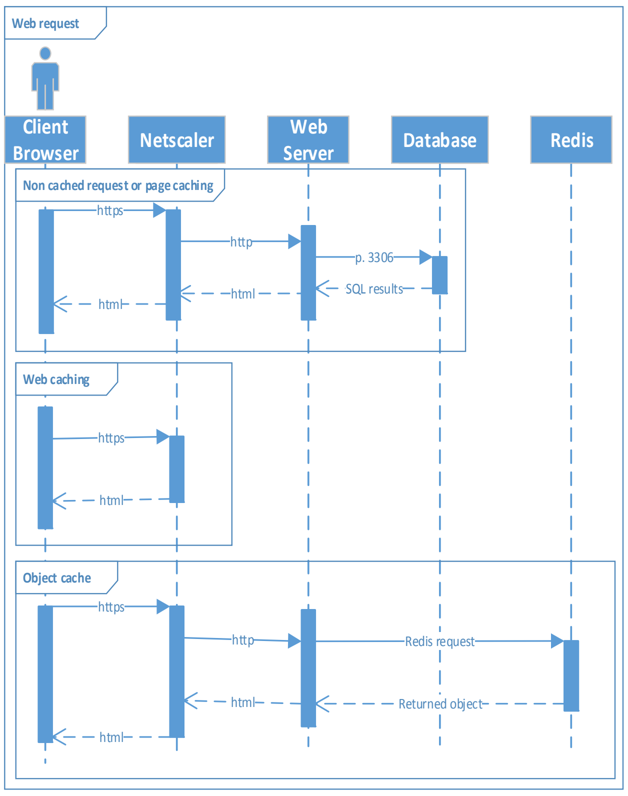

Fig. 2. UML sequence for caching in web delivery

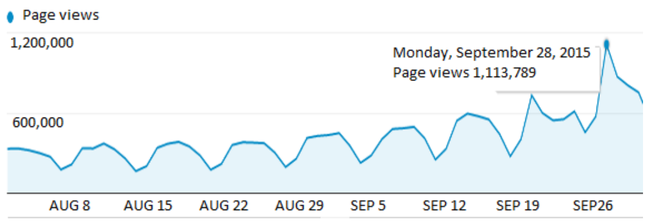

Fig. 3. Page views for 2 month period

As shown in [3] traffic congestion in page-request traffic on the Web is often predictable far enough in advance so that something can be done to mitigate its effects. This can be quite obvious in our scenario. In Fig. 3 we see daily page views of the main site for a period of two months from 3rd August to 3rd October 2015. We see an obvious spike in traffic in the week that is related to a high level event known for attracting increased number of visitors on the web sites, with highest value of 1.1m daily page views, and also notice that all weekends receive low traffic. Aside from this spike a slow but constant increase in page views can be seen over time.

In order to mitigate the negative effect on the infrastructure in preparation of such predictable events the caching of web pages is increased from 30 seconds to 5 minutes.

Database cache is done on the applications that were rewritten using some of the modern CMS systems with built in Redis support. Even though not directly benefiting performance of the legacy infrastructure such performance gain on the new web farm was used to offload more legacy web applications.

Client cache is implemented by rewriting headers of web pages on the reverse proxy. Even though a page returns no-cache header, this can be changed on the fly by rewrite engines. This reduces overall page views, as those requests do not reach our load balancers we cannot present statistical data on the impact.

B. *Content switching*

Web cluster architectures are a popular solution for supporting websites with large number of requests. The load balancing in the clustered web servers is performed on the basis of the requested content. The request for a resource contains several details regarding requested content. These load balancing algorithms use the content of requested web pages and the information stored in the request as a criteria to select the particular server [4]. We use a load balancing based on requested content, or content switching. Content switching is technique by which an application layer load balancer makes decision for routing of the HTTP request based on the type of content.

We use two modes of content switching:

1. Content switching based on the requesting URI path.
2. Content switching based on file type.

TABLE 1: FILE TYPE CONTENT SWITCH

File type	*Handled by*
Media (jpg, png, …)	Apache 2 or CDN
HTML	Apache 2
CSS	Apache 2
Java Script	Apache 2
SHTML	Legacy or IIS
ASP	Legacy or IIS

Table 1 presents a sample of file types and how they are handled. Fig. 4 provides the activity diagram of the content switch that we have implemented.

In addition to this, a CDN is used to directly offload images, documents and other media as well as CSS and JavaScript code. In order to use CDN links to media files must be changed to use the CDN subdomain. As we only address www, this optimization is not included in the data.

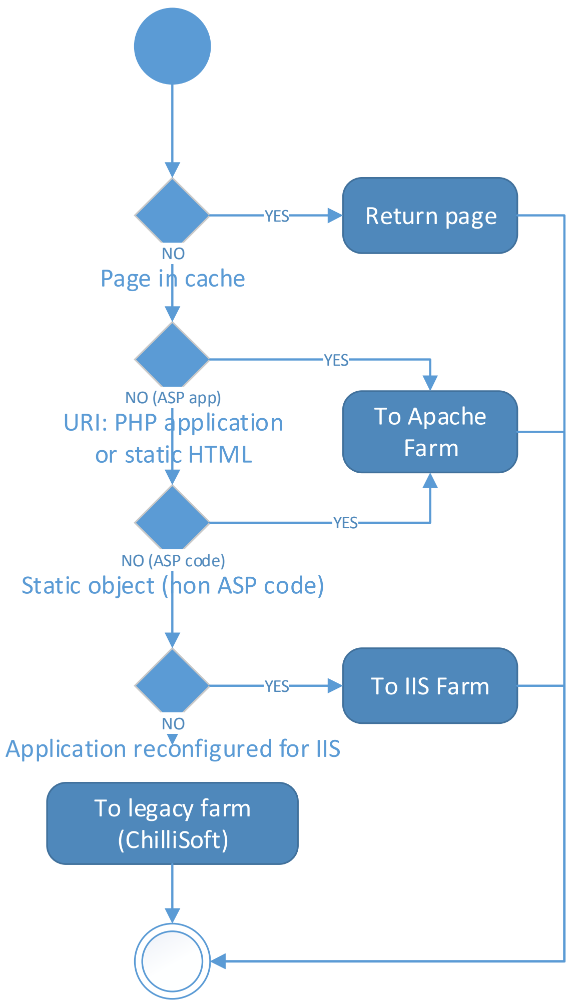
YES
Return page
NO
Page in cache
YES
NO (ASP app)
URI: PHP application or static HTML
To Apache Farm
YES
NO (ASP code)
Static object (non ASP code)
YES
To IIS Farm
NO
Application reconfigured for IIS
To legacy farm (ChilliSoft)

Fig. 4. UML activity for web content switching

V. PERFORMANCE EVALUATION

The new setup presented in the previous section yielded several benefits. Most importantly the page loading time for the content that was offloaded to newer infrastructure was decreased by 35% or more, the load reduction effect especially during peak times has produced page load decrease on the pages served by legacy infrastructure by 25% or more. The content split enabled divide and conquer approach to recode portions of the web page using newer content management systems (CMS) which speed up the migration away from legacy infrastructure.

As we described in section IV-a, web caching increase from 30 seconds to 5 minutes was deployed prior to expected web traffic increase. In Fig. 3 we saw doubling of page views, this was handled only by average of 65% server load increase. Without caching we have witnessed increase of 90-120% in server load.

When it comes to object caching using Radis we have seen multiple benefits, the reduction of database server loads as well as page loading times reduced to 30-50% of the loading time prior to implementing such cache.

The most effective reduction on legacy server load was accomplished by using content switching that we described in section IV-b. As we showed in Table 1 a content is switched depending on file type. In table 2 we present http requests for objects depending on file type for one day.

TABLE 2: DAILY OBJECT REQUESTS 28 SEPTEMBER 2015

File type	*Objects requested*
Total	42,718,796
*.asp	2,200,122
*.shtml	501,393
*.html	636,697
*.jpg	9,025,770
*.pdf	639,499

As ASP code can be present directly or indirectly included via SSI includes in asp, shtml, and html pages. The total of 3,338,202 requests are processed by asp engine, or 7.81% of all requests. This percentage can vary over time.

Not all of these requests go to the legacy servers. As we showed in Fig. 4, some applications were migrated to IIS farm and their respective paths no longer touch legacy servers. In table 3 we summarize page views over two months period and separate depending on their path.

TABLE 3: PAGE VIEWS OVER TWO MONTHS

Page views	*Objects requested*	*Percentage*
Total	25,755,238	
IIS farm	2,517,735	9.78
Apache farm	1,674,844	6.50

The pages that are handled by IIS should be calculated against the 7.81% which brings the object hits to estimated 7.05% of all objects as shown in table 4.

TABLE 4: IMPROVEMENTS SUMMARY

Property (requests)	*Percentage*
Pages with asp code	7.81
Handled by IIS farm	9.78
Pages handled by legacy	7.05

We should note that the 6.50% of the page views directed to the Apache farm reflects the URI paths that are completely routed to it, however as we previously said the actual handling of web requests is done mostly by Apache.

VI. CONCLUSION

The results presented in this paper show that using combined approach of web caching and content switching can significantly reduce load on legacy infrastructure enabling large organizations and corporations to increase the life span on their web applications, reduce outages caused by increased load and enable upgrades of the web systems in incremental phases.

Future work on this subject is to include data over longer span and track the progress of the values calculated in the previous section over time. Further offloading can be done using the cloud as described in [5]. Performance of such offloading should be calculated and compared to current results.

REFERENCES

[1] S. Sounders, “High-Performance web sites” Communications of the ACM, vol. 51, no. 12, pp. 36–41, Dec. 2008

[2] “Using Sun ONE Active Server Pages”. Available: http://docs.oracle.com/cd/E19636-01/817-6246/gsgasp.html

[3] Yuliy Baryshnikov et al., “Predictability of Web-Server Traffic Congestion” in Proc. 10th International Workshop on Web Content Caching and Distribution (WCW’05), 2005

[4] H. Mehta, et al., “Decentralized Content Aware Load Balancing Algorithm for Distributed Computing Environments” International Conference and Workshop on Emerging Trends and Technology, Mumbai, Feb 2011, pp. 370-375

[5] Inchul Hwang and Jonggyu Ham, “Cloud Offloading Methods for Web Applications”, 2nd IEEE International Conf. on Mobile Cloud Computing, Services, and Engineering, 2014, pp. 246-247

[6] Valeria Cardellini, “Dynamic Load Balancing on Web-Server Systems”, IEEE Internet Computing, May-June 1999 pp. 28-39

[7] Lu, Yi, et al. "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services." Performance Evaluation 68.11 (2011): 1056-1071.

[8] Sulaiman, Sarina, et al. "Intelligent Web caching using neurocomputing and particle swarm optimization algorithm." Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference on. IEEE, 2008.

[9] Kumar, Chetan, and John B. Norris. "A new approach for a proxy-level web caching mechanism." Decision Support Systems 46.1 (2008): 52-60.