Performance Analysis of GRID Middleware Using Process Mining*

Anastas Misev[1] and Emanouil Atanassov[2]

[1] University Sts Cyril and Methodius, Faculty of Natural Sciences & Mathematics Institute of Informatics, Skopje, Macedonia
[2] Bulgarian Academy of Sciences, Institute for Parallel Processing, Sofia, Bulgaria
anastas@ii.edu.mk, emanouil@parallel.bas.bg

Abstract. Performance analysis of the GRID middleware used in a production setting can give valuable information to both GRID users and developers. A new approach to this issue is to use the process mining techniques. Analyzing logs of the middleware activities, performed on the SEE-GRID pilot production Grid infrastructure, objective qualitative and quantitative information on what actually happens can be obtained. Using the appropriate tools like ProM to apply the process mining algorithms, many interesting findings and conclusions can be drawn. In this paper we describe our approach and show some of our conclusions.

Keywords: Grid, middleware, performances, process mining.

1 Introduction

Performance analysis of the GRID middleware can give valuable information to both GRID users and developers. Users gain by better understanding the workflow that is followed during job's lifecycle and the possible obstacles. Since the Grid middleware usually presents alternative ways to accomplish the same final result, the relevant performance information enables the users to optimize their choices and improve their throughput.

Developers can benefit by locating the bottlenecks and other problematic points during the job lifecycle and try to modify the middleware appropriately. They can also compare various implementations.

The performance of the GRID middleware can be analyzed from various aspects. As seen in [1], [2], analysis can be performed on the MDS, OGSA-DAI etc. All of this focuses mostly on the developers views of the middleware.

In this work, we analyze the performance from the logging and bookkeeping data obtained from the Logging and Bookkeeping (L&B) Service. In this way, we try to quantify the perception of reliability that the users get when they look at the final outcome (success/failure) of their jobs.

* This paper is based on the work done at the Institute for Parallel Processing at the Bulgarian Academy of Sciences, during the one month stay, supported by the FP6 project: Bulgarian IST Centre of Competence in 21 Century (BIS-21++), Contract no.: INCO-CT-2005-016639.

M. Bubak et al. (Eds.): ICCS 2008, Part I, LNCS 5101, pp. 203–212, 2008.

2 Description of the Logging and Bookkeeping Service and Database

The Logging and Bookkeeping (L&B) service [3] tracks jobs managed by the gLite WMS (workload management system) or the Resource Broker (RB). It gathers events from various WMS/RB components in a reliable way and processes them in order to give a higher level view, the status of job. Virtually all the important data are fed to L&B internally from various gLite middleware components, transparently from user's point of view.

Three main features of the system are events delivery, notifications and security, and access control. For a deeper understanding of them, refer to [3], [4].

All the data that the service receives is stored in a relational database. The diagram of the database is shown in the Fig. 1.

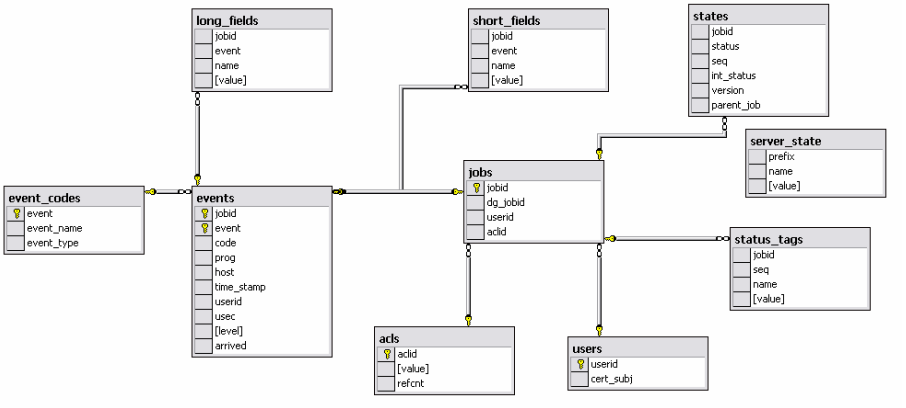

Fig. 1. Database structure of the L&B database

The unique user id, along with the cert data from the X509 certificate is stored in the Users table. Each job is assigned a unique identifier that is used as a foreign key in the related tables and is stored in the Jobs table. For each job, several events are created in the Events table. Each event has the job id, the sequence number, event code and a time stamp. Two more tables relate to the Events table: Short_fields and Long_fields. Both of them store pairs of (name, value) data, related to the event by the job id and event sequence number. The Short_fields contains shorter values, strings of up to 255 characters. For the longer values (entire JDL for a particular job, CE name, name of the queue, names of the files accompanying the job etc), which can be up to 16 millions of characters, the database uses the Long_fields table, with the same reference to the Events table (job id and event sequence number). When the job's lifetime ends, a record is added to the States table, referencing the job's final sequence number and large string field containing detailed description of the job lifetime.

3 The Rationale for L and B Based Performance Analysis

Various approaches have been proposed to tackle the performance analysis of the distributed systems. The work done by Margalef et al [5], proposes 3 different approaches: static, run-time and dynamic. In that context, L&B based analysis is a static, post-mortem analysis. As such, it has many advantages, but also some disadvantages. Main advantages are that it will not introduce any overhead to the production system, since the analysis is done off-line. Analyzing trace files can require lots of time, but since time is not an issue in off-line analysis, more comprehensive and in-depth analysis can be performed, helping even non-expert users to make fine tuning of their applications.

Main disadvantage of this approach is that such analysis require high level of details in the log files, which will require lots of resources (both processing and storage) for their manipulation. Also, since the analysis is static and post-mortem, it cannot cope with some dynamic application behavior that will occur during each execution.

Process mining has not yet been used as a tool to make performance analysis of GRID middleware. Other applications of the technique have proven very useful [7], [11].

4 Short Overview of Process Mining

Process mining techniques allow for extracting information from event logs [6], [8], [9]. It targets the automatic discovery of information from an event log. This information can be used to deploy new systems that support the execution of business processes or as a feedback tool that helps in auditing, analyzing and improving already enacted business processes. The main benefit of process mining techniques is that information is objectively compiled. In other words, process mining techniques are helpful because they gather information about what is actually happening according to an event log of an organization, and not what people think that is happening in this organization.

The type of data in an event log determines which perspectives of process mining can be discovered and specifies the type of questions that can be answered using the mining process:

1. The control flow perspective can be mined if the logs contain tasks executed by a process. The key elements in this perspective are processes and cases (process instances). This represents the “How?” questions.
2. If the log provides information about the persons/systems that executed the tasks, the organizational perspective can be discovered, giving answers to the “Who?” questions.
3. When the log contains more details about the tasks, like the values of data fields that the execution of a task modifies, the case perspective (i.e. the perspective linking data to cases) can be discovered. This relates to the “What?” questions.

We have chosen the ProM framework [10], [11] for several reasons: it is open-source, Java based, and it has big variety of available plug-ins. It is extensible with new plug-ins, if required.

The included plug-ins can be used either on logs only, called discovery plug-ins, or on logs and process models, called conformance and extension plug-ins. The discovery plug-ins can be used to discover the process elements only from the log files. They can then depict the process into various formats (Petri nets for example). The conformance plug-ins relies both on log data and a process model. They can be used to test the conformance of the data in the log to the proposed process model. The extension plug-ins also requires both logs and process model, but they discover the information that will enable to enhance the process model.

The ProM framework uses its own format to store the log data and additional attributes. The format is called MXML and is based on XML. Along with the ProM, there is an open source tool called ProMImport [12] that enables conversion from various well known log formats into MXML.

5 Application of Process Mining on the L and B Log Data

For the purpose of our analysis, we use the job identifiers as process instances (or cases) and events as audit trail entries. We also use the status field of the job (from the Events table) as the model element (or state in which on job can be in) and the combination of program and host name as originator (the entity performing the process element). For the future research, we will add more attributes to the analysis (CE name, detailed status of the job, queue and VO name, etc.)

After we have imported the log data into the MXML format and loaded the log into the framework, we can proceed with log filtering. Log filtering enables us to select only the data that is relevant to the analysis. For example, we can define that only logs for jobs starting with REGJOB event will be used. Also, we can select that we will analyze only complete instances, so we can define another filter that will include only jobs with particular event as last event (DONE, CLEAR, CANCEL…).

It is possible to perform more advanced filtering using the advanced filtering tab. For example, we can filter out only events done by specific originators, which will help us to reduce the data for some of the analysis. This is important in our case since some of the events are reported by multiple originators to the service. Also, we can use remapping filter to remap sub-jobs to a parent job.

5.1 Log Summary

We have started the mining process of the L&B logs with simple log summary plug in. It gives an overview of the number of jobs (process instances) and events (audit trail entries). For each of the model element, a frequency is calculated and shown. Also, the model elements that are first in the audit trails (Starting log events) and last (Ending log events) are shown with the frequency of their occurrence. Finally, each of the originators is shown, along with the frequency of occurrences in the audit trails.

From here we can get basic notion about the data we are mining. For example, we can instantly see how many of the process instances finished successfully by looking

at the Ending log evens. We can also see the workload performed by various originators (program-service and host name combination).

5.2 Heuristic Miner

Another appropriate plug-in for analyzing data that is less structured or has instances that follow several different paths of execution is the Heuristic Miner. Using the tool, a heuristic network can be produced depicting the control flow in the given process model. A simplified example is shown in Fig. 2. The numbers in the boxes represent the number of occurrences of the specific event and the numbers on the links represent the frequency and the absolute number of occurrences of a specific transition.

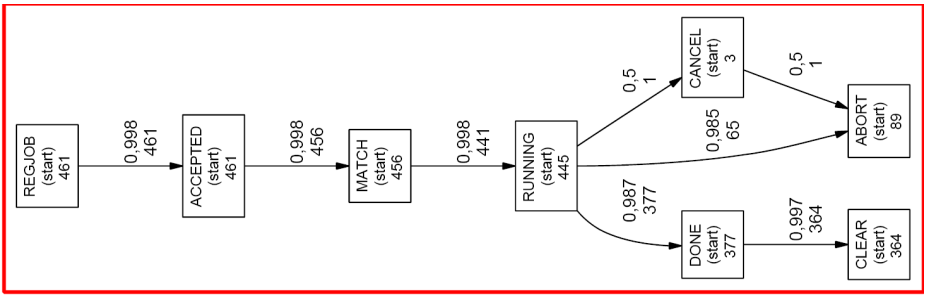

Fig. 2. Heuristic network

Using this network, we can easily recognize the frequencies of various transitions in the job's lifespan. The network can also be converted to a Petri net, as one of the most common formalism used to represent workflows.

5.3 Petri Net Performance Analysis

Once having a Petri net from the log, we can perform additional analysis. Especially useful is the Petri net performance analysis. As a result of this analysis, an interactive diagram is produced, helping in identifying the bottlenecks in the process model.

Different color coding of the places (circles) of the Petri net marks the different time needed in each one of them, as shown in Fig. 5. (Blue means low waiting time, yellow middle and purple high). Also, by selecting two transitions in the net, the tools shows the statistics (min, max and average time needed from one to the other).

5.4 Performance Sequence Diagram

Performance sequence diagram plug-in can be especially of help if you want to know what behavior in your processes is common, what behaviors are rare and what behavior may result in extreme situations (e.g. instances with extremely high throughput times).

An example of the output is given in the Fig. 4. We have used this output to identify the most common sequences of events that occur during job's lifetime, along

with their basic statistics. The diagram can be a full diagram, showing all the instances into time, or pattern diagram, grouping (by variable parameters) similar sequences into patterns. It also has a rich set of filtering options allowing us to select sets of process instances, or even individual ones. This plug-in, if used by end-users can help them visually identify the problems with their job submissions.

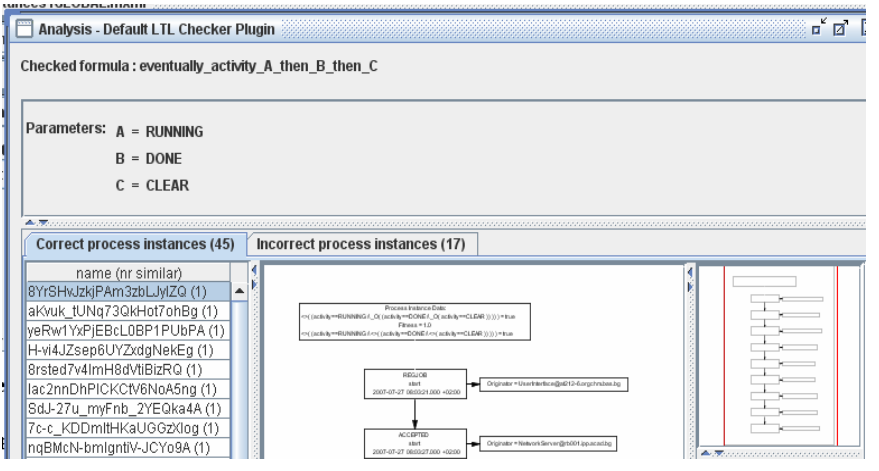

Fig. 3. LTL plug-in

5.5 Conformance Analysis

Conformance analysis plug-in requires both log data and a process model (Petri net for example). It replays the entire log and checks the conformance of each job with the model. It offers two perspectives: Log and model. The log perspective illustrates each separate job and indicates the ones that do not conform to the model. Model perspective shows the Petri net and indicates the non conformant points. It can also show the number of times an activity should occur (regarding the model) but actually didn't and vice versa.

5.6 LTL Plug-In

The Linear Temporal Logic (LTL) plug-in check validity of LTL formulas on the analyzed log. It has a rich set of options and predefined formulas. As a result it divides the set of process instances into ones that satisfy and others that not satisfy the formula. The example shown in Fig. 3 show the conformance of the processes to the formula "eventually activity RUNNING then DONE then CLEAR".

6 Some Important Findings about Middleware Performance Derived from the Process Mining

We have performed the analysis on different subsets of the L&B data. At the beginning, mostly for performance reasons, we have analyzed jobs from several users,

user by user. Subsequently, we have made a filtered data set from the whole database. We must note that some of the plug-ins requires much more time when working with large datasets, especially the ones that perform log replay. Most of the results that follow are from the overall analysis.

6.1 Percentage of Successful Jobs

From the performed analysis, we can conclude that the underlying infrastructure (SEE-GRID [13]) performs satisfactory. The overall percentage of the successful jobs is around 70%.

We identified several factors that influence the success rate of jobs. First of all, there is the human factor. Analysis of logs of jobs submitted by experienced users show greater percentage of success. If we analyze filtered logs from more experienced users, we can conclude that up to 80% end either with status DONE, or with status CLEAR (retrieved output). We have to make deeper analysis which require additional data attributes added to the logs to be analyzed, like status code, exit code etc. to better understand the percentage of the finished jobs and what is more important to discover the reasons why the other jobs failed.

Other factors include:

1. the "quality" of the Grid sites – usually larger sites in terms of number of CPUs have better support
2. software versions – the installation of a new middleware version or revision usually causes some hick-ups
3. Lack of resources or inappropriate job scheduling mechanisms – large percentage of failures are caused by jobs waiting in the queue for too long. The so-called proxy renewal mechanism did not work reliably until newer versions of the middleware solved the problem.

6.2 Patterns of Job Control Flow in the Middleware

Using the Performance sequence diagram, we can obtain useful data about the patters of events that the jobs follow during their lifetime. Analyzing a RB log consisting of

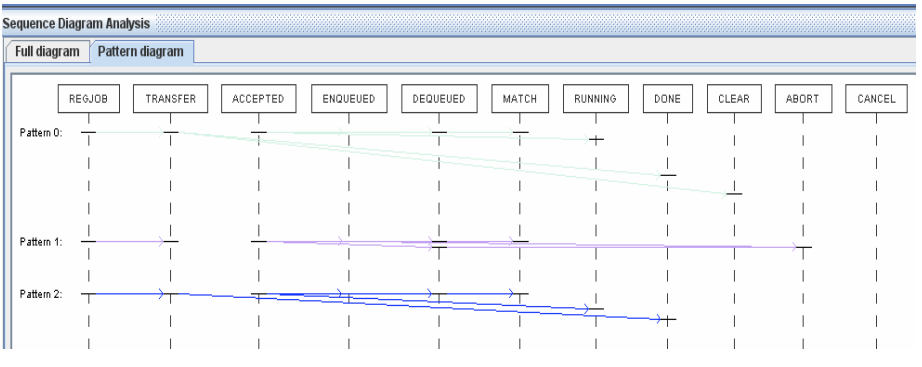

Fig. 4. Patterns of job control flow

around 18500 jobs, we have identified 85 different patterns of behavior. As shown on the Fig. 4, most of the jobs that finish successfully follow the first and the third pattern. They do it in average time of 27 hours and 11 hours respectively, with the former length is due to user intervention to pick up the results. This can be used as a good reference for the lengths of the proxy certificates.

Another conclusion that we can make from this results is the relatively large number of jobs following the fifth pattern (Pattern 4 in Fig. 4). Around 600 jobs have failed after waiting in the queues for an average of more than 280 hours. Using this data and examining the specific instances we could identify the reason for such failures.

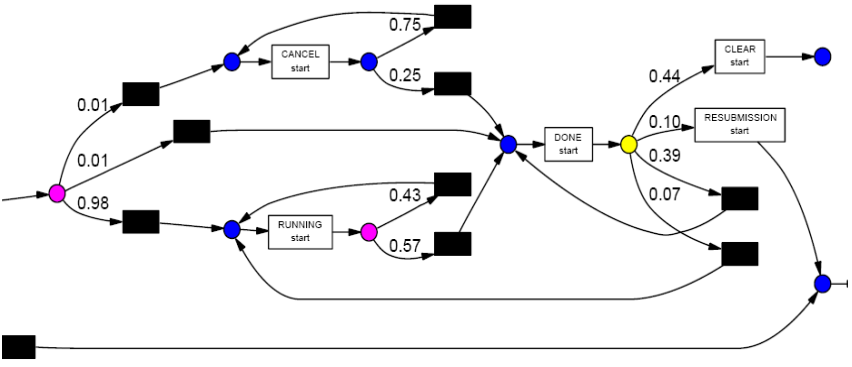

Fig. 5. Performance analysis with bottlenecks (details)

6.3 Bottlenecks in the Job Lifetime

Using performance sequence diagram, we have analyzed jobs from single user (for performance sake). Out of 470 jobs, almost 20% of them have been waiting in the queues with average time between 57 and 65 hours. All of them finished with ABORT (mostly due to proxy expired).

Using Petri net performance analysis we could also note that the point with the biggest waiting time (shown in purple in the Fig. 5).

We can see that the most time jobs spend waiting to start running. Other two bottlenecks include the running time (which greatly depends on the job itself) and time before the output is retrieved (shown in yellow), which is a human interaction.

7 Future Works

The work specified in this paper is only the beginning of deeper and wider performance analysis of the GRID middleware. Several issues that we will tackle soon include building custom import filter, based on the ProMImport framework to import the data directly from the L&B database, extend the data that is imported into the framework with additional elements (CE name, matching process results, some JDL attributes etc. to enable even more analysis), enable direct connection to the L&B web service interface, so the users can select particular jobs from within the ProM framework and get even more data from the service directly and propose a more

intuitive interface (possibly web) to the L&B data, to enable users to get better understanding.

Waiting time in the queues can be quite long. Some solutions to these problems that we will investigate further are:

- Providing separate queues for various types of jobs could strengthen the user's perception of the GRID,
- Providing end-to-end mechanism for jobs prioritization.

8 Conclusion

Using the process mining to analyze GRID middleware is not a new idea, but very little has been done to actually analyze the platform. Using the L&B database as a source of logging data was a natural choice. After researching for the appropriate tool, the ProM tool was chosen, mostly for the features mentioned. The initial results of the mining process are presented in this paper.

A very important conclusion from the analysis is that the underlying infrastructure performs satisfactory. With an overall job success rate of around 70% it is quite near the EGEE [14] average of 79% [15].

Since users experience affects the percentage of successful jobs, the education of the users about the underlying technology will increase the overall performance. The more aware the users are about the possibilities of the infrastructure and in the ways to evaluate certain sites, the better the success rate will be. In this context, measuring the success rate of each site can help users choose only the set of sites that promise higher throughput.

References

1. Zhang, X., Schopf, J.M.: Performance Analysis of the Globus Toolkit Monitoring and Discovery Service. In: MDS2, Proceedings of the International Workshop on Middleware Performance (MP 2004), part of the 23rd International Performance Computing and Communications Conference (IPCCC) (2004)
2. Jackson, M., Antonioletti, M., Chue Hong, N., Hume, A., Krause, A., Sugden, T., Westhead, M.: Performance Analysis of the OGSA-DAI Software. In: Proceedings of the UK e-Science All Hands Meeting, Nottingham, UK (September 2004)
3. EGEE User's Guide, Service Logging And Bookkeeping (L&B) (2007), `https://edms.cern.ch/document/571273/1`
4. Kouril, D., Krenek, A., Matyska, L., Mulac, M., Pospısil, J., Ruda, M., Salvet, Z., Sitera, J., Skrabal, J., Vocu, M.: Advances in the L&B Grid Job Monitoring Service (2007) (visited 06.08.2007), `http://lindir.ics.muni.cz/dg_public/lb2.pdf`
5. Margalef, T., Jorba, J., Morajko, O., Morajko, A., Luque, E.: Different Approaches to Automatic Performance Analysis of Distributed Applications. In: Getov, V., et al. (eds.) Performance Analysis and Grid Computing. Springer, Heidelberg (2004)
6. van der Aalst, W.M.P., Weijters, A.J.M.M. (eds.): Process Mining. Special Issue of Computers in Industry, vol. 53. Elsevier Science Publishers, Amsterdam (2004)

7. Rozinat, A., de Jong, I.S.M., Gunther, C.W., van der Aalst, W.M.P.: Process Mining of Test Processes: A Case Study, BETA Working Paper Series, WP 220, Eindhoven University of Technology, Eindhoven (2007)
8. Alves de Medeiros, A.K., Günther, C.W.: Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms. In: Sixth Workshop and Tutorial on Practical Use of Colored Petri Nets and the CPN Tools, Aarhus, Denmark (October 2005)
9. Process mining (2007), http://www.processmining.org/
10. ProM tool (2007), http://is.tm.tue.nl/~cgunther/dev/prom/
11. Alves de Medeiros, A.K., Weijters, A.J.M.M. (Ton): ProM tutorial, Technische Universiteit Eindhoven, The Netherlands (November 2006)
12. ProMimport, http://is.tm.tue.nl/~cgunther/dev/promimport/
13. SEE-GRID – South Eastern Europe GRID-enabled eInfrastructure Development (2007), http://www.see-grid.eu/
14. EGEE – Enabling Grids for E-sciencE (2007), http://www.eu-egee.org/
15. Monitoring and visualization tool for LCG (statistics for February 2008), http://gridview.cern.ch/GRIDVIEW/job_index.php