

Grid computing performance prediction based in historical information¹

Francesc Guim¹, Ariel Goyeneche²,
Julita Corbalan¹, Jesus Labarta¹, Gabor Terstyansky²

¹ Computer Architecture Department, Universitat Politècnica de Catalunya,
{fguim, juli, jesus}@ac.upc.edu

² Centre for Parallel Computing, Cavendish School of Computer Science,
University of Westminster, 115 New Cavendish Street, London, W1W 6UW
{goyenea, terstyg}@wmin.ac.uk

Abstract. Performance prediction in Grid computing presents an important challenge due to Grid environments are volatiles, heterogeneous and not reliable. We suggest that the historical data related to applications, resources and users can provide an adequate amount of information for modelling and predicting Grid components behaviours. That information can be use to build a dynamic top-bottom Grid Service and Resources performance descriptions. In this paper we present the initial work that we have done in order to define prediction techniques. We show how the workload load analysis carried out in the previous stage of the presented work, has guided us in the design of prediction techniques, the mechanisms created for their evaluation and the performance predictors analysis.

1 Introduction

Performance refers to system responsiveness: either the time required to respond to specific events, or the number of events processed in a given time interval. For traditional information systems, performance considerations are often associated with usability issues such as response time for user transactions. On the other hand, the performance targets in Grid computing has had to be adapted to support heterogeneous and not reliable environments, where the analysis information can be limited, incomplete and non up-to-date.

Currently, at best, computational scientists, researchers and HPC users access Grid Services and Resources via Grid portals, which provide common components that can

¹ The integration work published in this paper has been supported by “CoreGrid”, network of excellence in “Foundations, Software Infrastructures and Applications for large scale distributed, Grid and Peer-to-Peer Technologies”, and by the Spanish Ministry of Science and Education under contract TIN2004-07739-C02-01.

securely help them to create, submit, and schedule Grid jobs. In addition, portal users may perhaps be allowed to track and monitor submitted jobs. However they still do not have the essential information in order to identify in advance the job performance behaviours and possible execution cost. At the Grid infrastructure level, performance prediction is fundamental for Scheduler and Brokers, which can improve their suggestions and decision having an approximate picture of the applications behaviour.

In order to answer some of the issues related to the Grid computing performance prediction area, we suggest that the historical data related to applications, resources and users can provide an adequate amount of information for modelling and predicting its behaviours. But, as we are aware of the complexity of this topic and the added difficulty of working with real workloads, we decided to carry out a preliminary stage that could guide us on the processes of designing such techniques. Mainly, the objectives of this stage are to understand the behaviours and characteristics of submitted jobs and to design and test performance prediction techniques.

The remainder of the paper is organized as follows. Sections 2 spots performance in Grid computing, Section 3 describes an approach of solving this issue, Section 4 tackles the paper's objectives by presenting general mechanisms in order to evaluate predictors and comparing their performance, designed predictors and analysis of their results. Finally, the current work conclusion and future work is described.

2 Performance in Grid Computing

Traditional computing performance research is focused in the exact qualification of software programs and resources, and an accurate prediction of its run-time performance. On the other hand, performance in Grid computing could be defined as the source of information to identify the most reliable source of capacity computing power.

The open question that needs to be addressed is how the performance knowledge, that has been a topic of much scrutiny for an important number of years, could be adapted to reflect the grid resources and what has to be changed, improved and added to fulfil these new characteristics.

2.1 Filling a vacuum

A snapshot of the most important traditional performance techniques may start with *execution-driven* [1] performance simulators. They can be acutely dismissed for Grid computing on the grounds of being extremely slow, and memory intensive. Techniques using *trace-driven* [2] performance analysis, where the inputs to test programs and resources are traces of dynamic instructions without full-function, are not an option given that the complexity of microarchitecture has increased dramatically, causing *trace-driven* performance simulation to become time consuming and, if having any result, too resource related. Also, *instruction-level* [3] methods that conceptually simulate all feasible paths on arbitrarily detailed timing models of the hardware plat-

form, assumes that input data is known and therefore only analyze a single path through the program associated with this input data.

Although there are examples [4] [5] [6] that try to improve some of the previous mentioned problems, all of them results in being too machine specific and non dynamic for Grid environments.

Consequently, performance in a Grid computing has to be considered using a different approach: Our strategy aims to exercise a *top-bottom* self learning performance description of Grid Resources and Services that can be used to determine performance predictions. The dataset is build from the historical usability information of Grid resources, services.

The *top-bottom* means from the very simple Grid Services and Resources description towards a more complex description that reflects its compositions and details. The axiom of this idea may be seen as "The more Resources and Services are used, the better and more accurate the descriptive information should be". This practice aspires to keep always learning and converging, given the volatile aspect of Grid environments, the best confluence of data descriptive structures, metadata, and granularity for Grid components that provides the best analysis for performance prediction.

3 Grid Performance description models

The description of Grid services, resources and run-time environment plays a key factor in our strategy. Many of the few existent Grid performance technologies use Software Performance Engineering (SPE) [7] [8] to provide a representation of the whole system. SPE is the systematic process for planning and evaluating a new system's performance throughout the life cycle of its development. Its goals are to enhance the responsiveness and usability of systems while preserving quality. SPE includes techniques for gathering data, coping with uncertainty, constructing and evaluating performance models, evaluating alternatives, and verifying models and validating results. It also includes strategies for the effective use of these techniques.

SPE needs a number of different pieces of information in order to construct and evaluate early life-cycle performance models. These information requirements fall into the following categories: Workload, Performance objectives, Software characteristics and information model, Execution environment, Resource requirements and processing overhead

3.1 SPE and Grid Computing

Taking into consideration the Grid computing scenario, there are some points to be considered at the time of using SPE in Grid Computing:

- The definition of software and system execution models is rather difficult because of the dynamicity of the system and rights to access them.

- The complexity of accurate performance measures may significant given that SPE is pointed to address performance information precision when the models granularity is small.
- And, for some performance computing system, which involve concurrency and parallelism, the models is not completed and must be enhanced [9].

Performance Analysis and Characterization Environment (PACE) [10] is an example of the SPE enhanced and used in Grid computing by the inclusion of a parallel layer between the hardware and the application model. PACE can be used to produce predictive traces representing the expected execution behaviour of an application given appropriate workload information and has the motivation to provide quantitative data concerning the performance of sophisticated applications.

But, because PACE is a solution that has been migrated and afterwards extended from a traditional performance computing research to a Grid environment, it assumes several restricted points, such us the need of the source code, which has to be written in language C, of each software component in the Grid and the requirement of running the Resource tools in each new or modified Grid resource in order to generate the hardware model

Consequently, this example emphasize the idea of a Grid SPE that is able to, dynamically, auto-learn and consolidate in a Grid performance description model, which initially is contemporary to the environment where the information is coming from, and afterwards, this model could be enhanced in order to extrapolate it, by projecting the information, to other particular systems

3.2 Grid SPE: Performance Meta-data models

Therefore, the very first meta-data classification could be defined by the use of a fixed-scenario, based in a particular run in a particular scenario in a particular run-time environment or, alternatively, analytical approaches that can produce parametric models.

The performance model granularity and dynamism restricts the approach to collect and use meta-data information. A model that adds on top of a basic structure a statically meta-data that helps analytical approaches could be used in both scenarios. But the use of static structures in volatile environment produces a non up-to-date model that constantly increases in volume and complexity. As a result, the model has to use the meta-data information as a feedback to dynamically upgrade the description model.

At this point, there are several issues that have to be considered in order to reach the proposed approach, such as:

- How important a SPE component of a Grid Service or Resource Description Model is?
- Meta-data weight: Is it possible to assign a level of importance to the data collected from the Grid? Is this data influenced by the SPE component weight?
- Is the Meta-data historical information depreciated by time?
- It is also depreciated by Resource or Service variations?
- How much influence the time that takes to process the information could affect the model?

- How can we quantify the error of prediction?

In order to start tackling these open questions, in the following section an initial work done on prediction techniques is presented. We have focused on establishing mechanisms for designing and evaluating predictors in order to understand, how all the components in Grid computing can be put together to solve performance problems. We starting with the testing of simple prediction techniques based on historical data. Before design complex techniques, we studied simple predictors that try to exploit the idea that user and groups use behave similar in their executions.

4 Grid Performance predictors

Initially this section presents a global definition of the predictor performance, the second part presents in details each of them and their main objectives, and finally the performance evaluation of such predictors is described.

4.1 Evaluating the predictors

There are two different issues that have to be treated: the first one is to decide when a given prediction is a *hit* or a *miss*, and more important how to define the global performance of a given predictor with a set of predictions.

4.1.1 Hit / Miss definition:

Definition of what is a hit or what is a miss can be seen on the formula picture below(1). The definition of a hit or miss is based on the difference between the real and the predicted value: if the difference is less than 5% of the required error, the prediction is considered as a hit.

$$a = \text{predictedValue} / \text{realValue} \tag{1}$$

$$\text{hit} = a \cdot 100 < 100 - \text{error} \parallel a \cdot 100 > 100 + \text{error}$$

4.1.2 Predictor performance:

More complicated is to define the global performance of a predictor with a given set of executions. Our definition of global performance is based on the percentage of times the different applications are well predicted. This analysis is carried out for each of the predicted values, such as memory usage or total time, because a given predictor may predict better some them that the other.

When evaluating a prediction variable of a given predictor, we define a vector that has 11 intervals. The first interval will contains the percentage of applications that we are well predicted around the 0% of the times, the second one the percentage of applications that are well predicted from 1% to 10% of the times, the third one the number

of those that are well predicted from the 11% to 20% and so on. Those predictors that are carrying out better predictors will have higher numbers on the right side of the vector. For instance we could have a predictor that for a given variable in the interval 81-90% has a value of 50%, what means that from the 100% of the predicted applications the 50% of them are well predicted from 81% to 91% of the times.

We decided to create another interval that ponders each interval of the vector explained in the last paragraph for the amount of the predicted *variable* consumed by each application that belongs to the interval in all of its executions. For example, in the case of the system time, we ponder each interval for the amount of the time consumed by the applications that belongs to it. With this second interval we could realize how important the predicted applications were. For example we could see that 1% of applications are well predicted 5% of times, and they represent the 45% of the total time consumed by all the applications.

4.2 Predictors

For this work we have developed seven different predictors based on similar submission behaviours. We support that this conclusion will be also applicable to the Grid environment, because in general, the users that are submitting jobs to our centres are the same users that will submit jobs to our Grids.

- 4.2.1 Last value job: This predictor returns, for a given user and a given application, the amount of the queried resource prediction that the application executed by the same user used in its last execution. In this case the predictor stores for each resource, such as total time or memory usage, the last amount of this that the last execution for the application and user had consumed. With this predictor we expected to predict the applications are not executed with frequency, because other predictors, that require more historical data, would not be reliable.
- 4.2.2 Last value job indexed by tasks: Last value job indexed by tasks: This predictor is pretty similar as the last one, but it also indexes its predictions by the number of tasks used by the execution. A prediction will return the last amount of the queried resource that used the last execution for the given application, the given user and with the given number of tasks. We expected to achieve better prediction with those parallel applications where the amount of the used resource depends on the number of tasks used in the execution.
- 4.2.3 Mean job tasks: This predictor returns the mean for the queried resources of all the previous executions that used the given number of tasks of the given application and for the given user. We expected to hit applications that are having some variability on the amount of used resources in their executions, and that with the last value we miss the predictions due to this variability is higher than a 5%, or the used percentage of error, but in mean this variability is less than the error.
- 4.2.4 Median job tasks: This predictor returns the median for the queried resources of all the previous executions that used the given number of tasks of the given application for the given user. The goal of this predictor is the same as the

last one: catch those applications that are having some variability in the amount of used resources in their executions for a given user. However, as the mean is heavily influenced for the outliers, we wanted to use the non biased estimator median.

- 4.2.5 Using mean and standard deviation: In [11] a formula (2) for memory usage prediction is presented. This formula uses the memory usage of all the past executions for the given user and application.

$$\begin{aligned} \mathbf{d} &= \text{executions}(\text{application}, \text{user}) \\ \text{prediction} &= \min(\max(\mathbf{d}), \text{mean}(\mathbf{d}) + 3 \cdot \text{stdev}(\mathbf{d})) \end{aligned} \quad (2)$$

- 4.2.6 Using median and the IQR : The formula explained on the last predictor was interesting, however as explained in the Mean predictor, the mean and the standard deviation are influenced by the outliers or extreme values, so we decided to implement the same formula but using non biased estimators (3), in order to check how the predictions could be affected by these values.

$$\begin{aligned} \mathbf{d} &= \text{executions}(\text{application}, \text{user}) \\ \text{prediction} &= \min(\max(\mathbf{d}), \text{median}(\mathbf{d}) + 3 \cdot \text{IRQ}(\mathbf{d})) \end{aligned} \quad (3)$$

4.3 Predictors performance

This section presents a general evaluation of the presented predictor performance focused on memory usage, total time and user time for the applications, users and groups contained in the workload.

The workloads were obtained from a Load Leveler three years history files obtained from an IBM SP2 System with two different configurations: the IBM RS-6000 SP with 8*16 Nighthawk Power3 @375Mhz with 64 Gb RAM and the IBM p630 9*4 p630 Power4 @1Ghz with 18 Gb RAM. A total of 336Gflops and 1.8TB of Hard Disk are available. The operating system that they are running is an AIX 5.1

4.3.1 Memory usage prediction in parallel applications

All the predictors had similar behaviour: they forecasting good predictions in 35% of the applications, they had poor prediction results in 30% of the cases, and the 35% left has been spread among the other intervals .

When pondering each application by its number of executions performance of all predictors is dropping substantially, what means that predictors did not do good memory prediction with applications that where executed some more times. Predictors last

value, last value indexed by task and median respect the others are achieving best performance. Worst results of predictor mean and predictor presented in 3.2.5 (the one that uses mean and the standard deviation) could be explained by possible outliers or extreme values that are affecting the biased estimators used in their predictions: mean and standard deviation. That implies that a future work should include analysis of outliers or values that are having a big impact on predictors based on biased estimators, perhaps pondering this extreme values by a factor of γ that decrease its weight in the overall prediction would be a way to decrease the effect of such values.

Finally, when taking into account the amount of memory usage of each application, the last value indexed by task respect the others is the one that is carrying out better predictions with such applications that are using more memory.

4.3.2 Total time prediction in parallel applications

Differences between predictors are less than 5 % in each interval, what mean that they are having similar performance in terms of how well are predicting each application.

Pondering each application for number of times that it was executed the last value indexed by task and last value predictors are having better performance with those applications that are more executed. We think that this fact could be caused because total time has some sort of linear relation with the number of used tasks.

Applications that spent more time to be executed are correctly predicted from 60% up to 89% of their total executions by three predictors: last value, last value indexed by tasks and mean. Predictors are doing good predictions with applications that are not executed many times (those that were executed more times were on the interval 30% to 70% for the same predictors). As a conclusion we can say that in general predictions can be improved substantially, but we have detected that an important subset of applications, those are executed few times, are good predicted for some of the predictors, that implies that we could use this kind of predictors for those predict those applications of which we have less historical data.

4.3.3 Memory usage prediction in sequential applications

Prediction performance for all the presented predictors is pretty similar when taking into account the percentage of times that a given application was well predicted. They split their predictions in two main groups of applications: 40% of applications are almost never well predicted, and another 40% is well predicted from 90% to 100% of times.

When pondering each application for its number of executions those applications that executed more times are well predicted from 1% -10% of times in the mean predictor. However median and last value predictors are achieving best results, this could mean that applications that are being executed more times have an amount of memory usage that are possible outliers or extreme values and are affecting the mean. Pondering applications for its total amount of memory used in general predictors had a poor performance for those applications that used more amount of memory.

Applications that are using more memory are never well predicted and applications that are more executed are well predicted at most the 40% of the time.

4.3.3 Total time prediction in sequential applications

As in the previous analysis, predictors had similar performance in terms of how well they predict the total time for the sequential applications. There are approximately a 50% of applications that are whose executions are well predicted around 90% -100% of times, and there are a 40% of applications that are almost never well predicted. However the importance of the applications that are well predicted is doubtful, due to those applications that are more executed or that are consuming more time are well predicted at most the 30% times of they executions.

5 Conclusions

One of the main problems regarding performance issues in Grid computing is the particular environment where Grid Services and Resources interact among them. We presents and justified a different strategy that intend to self learn from current and historical usability information of Grid Resources and Services in order to determine performance predictions.

For that reason, two early objectives have been defined: to understand the behaviours and characteristics of submitted jobs and to design performance prediction techniques.

The first goal was achieved studying the workload, provided by a Load Leveler history file, of our system of the lasts three years [12]. Something that has to be taken into account is that the presented study was not intended to provide an accurate description of the workload neither to provide specific workload models as can be found in other works[13], we characterized some issues that we considered useful for our purposes and we extracted and analyzed them from our system.

The second one has been completed by designing and evaluating simple prediction techniques. Before developing predictors based more complex and sophisticated techniques, such as those that can be found on [14], we decided to test how well predictors based on simple ideas or algorithms performed on our system.

We have realized that working with real workloads is a tedious and difficult task due to many factors has to be taken into account, and not always is possible achieve clear conclusions. Next phases will include study of synthetic workloads. This will help us to focus the prediction on set of different kind of applications, and we expected that will allow have more precise conclusions.

We have also concluded that there are no predictors that do good predictions for all the applications. It's necessary to design hybrid predictors or specific predictors for a given set of applications. We will need to characterize applications and find out which of their characteristics make them more suitable to be predicted by a specific predictor.

References

1. Poulsen, D.K.; Yew, P.-C, Execution-driven tools for parallel simulation of parallel architectures and applications, Supercomputing '93. Proceedings, 15-19 Nov. 1993 Page(s):860 - 869
2. Malloy, B.A.; Trace-driven and program-driven simulation: a comparison. Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, 1994., MASCOTS '94., 31 Jan.-2 Feb. 1994 Page(s):395 - 396
3. Embra: Fast and Flexible Machine Simulation, Emmett Witchel, Mendel Rosenblum, Massachusetts Institute of Technology and Stanford University. Sigmetrics 1996
4. PERFORM - A Fast Simulator For Estimating Program Execution Time, Alistair Dunlop and Tony Hey, Department Electronics and Computer Science, University of Southampton
5. Candice Bechem, et al; An Integrated Functional Performance Simulator, Carnegie Mellon University, 0272-1732/99/ 1999 IEEE
6. Brett H. Meyer et al; Power-Performance Simulation and Design Strategies for Single-Chip Heterogeneous Multiprocessors. IEEE Transactions On Computers, Vol. 54, No. 6, JUNE 2005
7. Connie U. Smith; Performance Engineering of Software Systems, Reading, MA, Addison-Wesley, 1990.
8. Connie U. Smith and Lloyd G. Williams; Software Performance Engineering: A Case Study Including Performance Comparison with Design Alternatives, IEEE Transactions on Software Engineering, Vol. 19, No. 7, July 1993
9. Stephen A. Jarvis, Daniel P. Spooner, Helene N. Lim Choi Keung, Graham R. Nudd Performance Prediction and its use in Parallel and Distributed Computing Systems.. High Performance Systems Group, University of Warwick, Coventry, UK
10. D. J. Kerbyson, J. S. Harper, A. Craig, and G. R. Nudd.. PACE: A Toolset to Investigate and Predict Performance in Parallel Systems. In European Parallel Tools Meeting, ONERA, Paris, October 1996. Keyword(s): PACE, Performance Prediction.
11. Anat Batat and Dror G. Feitelson, ``Gang Scheduling with Memory Considerations". In 14th Intl. Parallel & Distributed Processing Symp. (IPDPS), pp. 109-114, May 2000.
12. Francesc Guim Bernat, Julita Corbalan, Jesus Labarta; Analyzing LoadLeveler historical information for performance prediction. XXI Jornadas de Paralelismo. CEDI 2005
13. Evgenia Smirni and Daniel A. Reed, ``Workload Characterization of Input/Output Intensive Parallel Applications". In 9th Intl. Conf. Comput. Performance Evaluation, Springer-Verlag, Lect. Notes Comput. Sci. vol. 1245, pp. 169-180, Jun 1997
14. Warren Smith, Ian Foster, and Valerie Taylor, ``Predicting Application Run Times Using Historical Information". In Job Scheduling Strategies for Parallel Processing, Dror G. Feitelson and Larry Rudolph, (ed.), Springer Verlag, Lect. Notes Comput. Sci. vol. 1459, pp. 122-142, 1998.