International Journal of Grid and Distributed Computing
Vol.3, No.1, March, 2010

# Predicting the Performance of a GRID Environment: An Initial Effort to Increase Scheduling Efficiency

Nuno Guerreiro and Orlando Belo

*Department of Informatics, School of Engineering, University of Minho*
*4710-057 Braga, Portugal*
*{nmd.guerreiro@gmail.com; obelo@di.uminho.pt}*

***Abstract***

*GRID environments are privileged targets for computation-intensive problem solving in areas from weather forecasting to seismic analysis. Mainly composed by commodity hardware, these environments can deliver vast computational capacity, at relatively low cost. In order to take full advantage of their power we need to have efficient task schedulers with the ability to maximize resource effectiveness, shortening execution times. GRID schedulers must not only decide taking a snapshot of the GRID status into account, but should also consider the output involved in past decisions. In this work, we intend to show how resource usage can be analyzed, through the use of data mining techniques, to predict performance availability of a GRID environment, as a preliminary work to increase scheduling efficiency as well as adequate resource provisioning.*

***Keywords:*** *GRID environments, task scheduling optimization, resource availability, data mining, performance evaluation models.*

## 1. Introduction

Over the last decades, technological breakthroughs have changed the world so radically that we are now living in a global world, thirsty for information and unwilling to wait for that information to be discovered, captured and processed, before reaching its end consumer. Even though knowledge extraction algorithms and tools continue to evolve, processing relevant data requires usually expensive computational power. Over the last couple of decades a new trend of parallel architectures and distributed applications has emerged, presenting great advances on networking hardware and protocols, as well as in the evolution of entry-level equipments. Efforts like Beowulf [1] have turned commodity hardware into a feasible alternative for high performance computing, making it possible to connect multiple workstations in clusters and making them available for computation-intensive applications. Nonetheless, this approach still limited the maximum processing power, by requiring all nodes to be held by a single organization, on a single homogeneous data center. The Academia's work on fields like Cancer research or Weather analysis resulted on the development of applications whose processing power requirements are so extensive that cannot be met by a single computer. In response to those needs, universities and research institutes began sharing their resources regionally, then nationally and finally at a global scale. A new paradigm derived from this phenomenon: the GRID [2].

A GRID environment can be described as a cross-organization network of shared computing resources, potentially composed by both heterogeneous hardware and software. GRID resources tend to be geographically spread and may be used outside the GRID's scope,

rather than exclusively. In what concerns to coordination, GRIDs are managed in a distributed manner, at several levels, instead of following the traditional master-slave design, whose scalability and reliability, as we know, is ultimately limited by the master node. The complexity of task scheduling on the GRID is greater than on traditional clusters, since the set of resources changes throughout time, as does their availability to perform work for application purposes. Moreover, communication between resources and applications may span from extremely fast fiber-optic based networks to conventional long distance networks with higher latency and lower bandwidth. Hence, choosing a resource to execute a previously submitted task requires balancing the cost of using powerful, but remote resources in exchange for local, but more modest resources. Despite all the sophisticated scheduling algorithms currently available, scheduling software does not learn from its past decisions. Instead, the schedulers' decision processes are usually based on a snapshot of the GRID's status and the list of tasks it needs to execute. By gathering and analyzing a GRID environment's performance data over time, it is likely that more solid scheduling strategies may arise, therefore extracting more profit from the resources in place. Along these lines, data mining techniques enable both human and automated analysis of historical data with the goal of predicting the GRID's resources status throughout time. In this paper, we aim to demonstrate how can data mining techniques be used to extract valuable knowledge about resource's load over time in a GRID environment. In particular, we intend to show how humans and automated schedulers can benefit from such knowledge in order to increase scheduling efficiency and decrease execution times.

## 2. Executing Tasks on a GRID

The GRID has a decentralized coordination model, where multiple schedulers coexist with multiple resource managers, to provide a scalable and fault tolerant environment. GRID applications communicate with one of the available meta-schedulers and submit the tasks whose execution is required, along with its requirements (target OS, CPU architecture, minimum RAM memory, etc). It is up to the meta-scheduler to find the appropriate resources for executing the submitted tasks, using the GRID's middleware. Ultimately, the middleware may delegate the execution of a set of tasks on a single machine or on a cluster, whose task scheduling is done by a local cluster scheduler (figure 1).

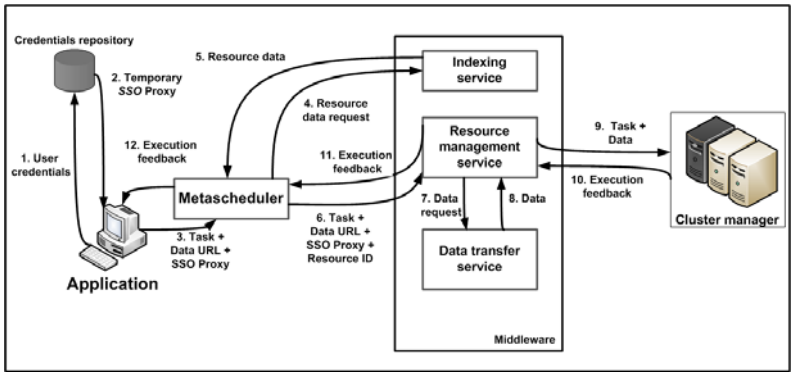


Figure 1. Performance availability of a GRID

Since GRIDs tend to be heterogeneous, software for them must be able to interoperate with its peers using standards [3], so that an organization is able to join the GRID without having

to adopt a complete new set of software. From a top-down perspective, meta-schedulers like Gridway [4] or Condor-G [5] interoperate with GRID middleware, as is the case of Globus Toolkit [6] or gLite [7]. It is mostly up to the middleware to provide support for the requirements that GRID applications set. Usually, the GRID's middleware does not execute the tasks itself. Instead, the middleware delegates task execution to local schedulers like Condor [8], Portable Batch System (PBS) [9] or Sun's GRID Engine [10]. The local scheduler manages local resources and a task queue and is ultimately responsible for executing tasks on the actual end nodes.

## 3. Evaluating the Performance of a GRID

Performance can be evaluated from multiple standpoints, depending on the type of analysis that is required. A GRID environment is no exception, since there are multiple metrics that can be thought of while evaluating the GRID's performance. Since the GRID is mostly used by applications whose execution time would be prohibitive for a single workstation, metrics that directly or indirectly relate to task execution time are among the most important. One of such metrics is performance availability, which measures the maximum amount of processing power a system is able to deliver on a given instant, considering its current load. When a system is fully loaded (i.e. cannot execute any more tasks) its performance availability is null. Conversely, if a system is idle, it is able to provide its maximum performance availability. In order to assess the GRID's performance, we chose to measure and evaluate performance availability. From our point of view, performance availability is a decisive measure that ultimately influences other ones, like execution time and success rate. It is our understanding that by scheduling tasks to resources with the best available performance, the GRID's efficiency is likely to improve.

Measuring and evaluating performance availability involves an objective measurement model, which needs to be based on resource status data, so that it matches the resources' performance. Considering the status data that it is possible to extract from resource indexing services, we decided to base our model on those that can be numerically evaluated and that have no direct relationship with other criteria. Therefore, we selected the following criteria to build an evaluation model: CPU specifications and number; memory specifications and amount; system load average.

$$Perf_{AVAILABILITY} = (kCPU \times kCPU_{ARCH} \times CPU_{FREQ} \times Free_{CPUs} + kMEM * Avail_{RAM}) \times Avail_{COEF}$$

Figure 2. Performance availability calculation

Our formula (figure 2) balances the importance of CPU and RAM memory by multiplying each fraction by a constant (kCPU and kMEM). Different CPU architectures deliver different performance values. So, we introduced a CPU architecture constant ($kCPU_{ARCH}$). The constants kCPU and $kCPU_{ARCH}$ are then multiplied by the number of free CPUs at each moment (it is assumed that all CPUs are equal on the same node) and by its operating frequency. This portion of the formula enables us to qualify the node's CPU in considerable detail. Different memory types may have different performance levels. However, it is not possible to retrieve such details from resource indexing services we analyzed. Therefore, we only considered the amount of memory available on each node ($Avail_{RAM}$). This value was

multiplied by the kMEM constant to obtain the memory's influence on a node's performance availability. Although the maximum capacity of a node is important, its real ability to execute work is far more significant in terms of performance availability, since it varies with the amount of load the node experiences. Therefore, a third factor was included in our model: system load average. This coefficient's ($Avail_{COEF}$) calculation formula can be found on figure 3. If the number of processes waiting for CPU time over the last 5 minutes is greater than the unit, a node is considered to have no availability, since it already has processes it cannot give CPU time to. On the contrary, if the value is less than one, the availability value is the difference to the unit. Therefore, $Avail_{COEF}$ varies between 0 and 1, that is, from null to maximum availability.

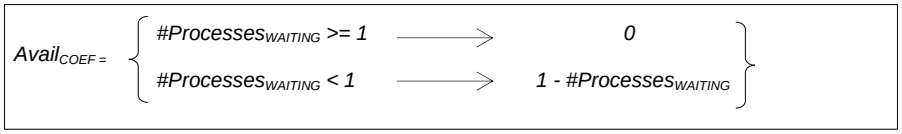


Figure 3. Availability coefficient calculation

## 4. The Performance Prediction Models

To evaluate the performance of the GRID environment we decided to use two types of data mining algorithms: clustering and decision trees. In order to do that, two models were created using each algorithm: one model for analyzing the entire GRID and other model for analyzing the behavior of each individual node.

Clustering algorithms are usually used for initial data exploration, in order to find patterns that help understanding the data set in hand. These models process data and group its instances that have similar values. In this particular case, we aimed to use clustering models to visually explore performance availability, by discovering patterns of similar performance and groups of machines that shared the same performance levels. Our clustering models were based on the KMeans algorithm. Decision tree algorithms are popularly used for prediction due to the ease of use and analysis. However, decision trees cannot predict continuous numerical values. Instead, the value to predict has to be transformed into a non-numerical value known as class or label. Therefore we categorized all performance availability values into six different classes (C0 - C5), using the following formula: round ($Perf_{AVAILABILITY}$ x 5) / (max($Perf_{AVAILABILITY}$)). A higher number of classes would introduce such detail that predictions would be inaccurate, despite being on neighbor prediction classes. In contrast, reducing the number of classes would increase prediction accuracy (ratio between accurate and total predictions), while decreasing the significance of each prediction. We used the DecisionTree algorithm, with minimum information gain of 0.005 and minimum confidence of 0.25. Tree depth was limited to a maximum of 30 levels. Our decision tree models were trained using approximately one third of the complete data set. The remaining data was left for prediction evaluation on two data sets: test and validation. RapidMiner's ClassificationPerformance operator was used to evaluate prediction accuracy.

In order to determine the effectiveness of our performance evaluation model, we simulated a typical Grid environment, using VirtualBox virtual machines on an Intel Quad Core Q9300 CPU (2.50GHz). Since VirtualBox assigns one OS process to each virtual machine, we limited the number of nodes to four, in order to allocate one of each of the four cores to a single virtual machine. Three of the nodes (gtnode-2, gtnode-3 and gtnode-4) received 1GB

of RAM memory, with gtnode-1 being granted 1.5GB of memory, due to the use of an X display server and some additional services. Communication between nodes was done via a set of simulated 100Mbit Ethernet interfaces. The four virtual machines were grouped into two virtual cluster sites (gtnode-1+gtnode-3 and gtnode-2+gtnode-4). We did not create distinct VOs, since it was irrelevant for our tests. To manage each of the clusters we used Torque 2.3.0, with Ganglia Monitor being chosen for monitoring the nodes' status. Although not mandatory, we installed the Globus Toolkit 4.0.6 on every node, to be able to gather data from each machine individually and also to have tasks being directly submitted to each machine, bypassing PBS if necessary. Ganglia's information was published on Globus' MDS Index service using the UsefulRP module. Data transfer between nodes was handled by RFT, with task management and resource allocation handled by WS-GRAM. In order to be able to submit tasks to both sites, we used Gridway 5.2.3 as the metascheduler. It was decided to install it on gtnode-1, due to its additional memory.

With the intention of simulating the load that real applications would introduce in our simulated Grid environment, we simulated three different applications, using the NAS Parallel Benchmarks [11]. We selected three benchmarks that target the processing ability for scientific applications (LU, SP and BT). Based on these benchmarks, we created three application profiles and developed a daemon that submitted tasks, for those profiles, using Gridway and its implementation of the DRMAA API. The daemon used probability values for each profile and period of the day, in order to determine if any tasks were to be launched for each profile. The three profiles were designed to simulate the activity of a fictitious organization, with different patterns of use throughout working days. On weekends, there was no load being injected on the Grid. The first profile (24h) represented a single-task application that could be executed with low probability over the entire day. The second profile (Work hours) simulated a single or double-task application used during work hours, whose probability levels try to mimic the organization's employees' productivity levels. The last profile (Nightly) simulated a batch application launched during the night, with four simultaneous tasks.

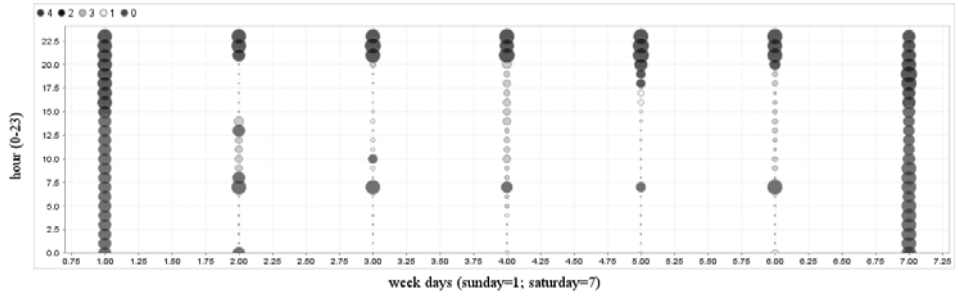


Figure 4. Grid's available performance over the week (clustering model)

In order to analyze the Grid's performance, we gathered status data during a full week, from Saturday to Friday, while the load injection daemon was executing. Since Globus Toolkit's MDS-Index module publishes node status data that can be accessed via any web services based application, we developed an additional daemon for gathering that data periodically (every 10 minutes), using Globus Toolkit's stub libraries. This daemon was installed on gtnode-1 and collected approximately 670 status snapshots in total. To perform an hour-by-hour analysis, we aggregated values on an hourly basis, by averaging values for

each machine. Status data for the entire Grid was generated by summing the values for all machines on a given hour. After collecting all the data, we built and applied the data mining models that were described in the previous section (Performance Evaluation Model), using RapidMiner.

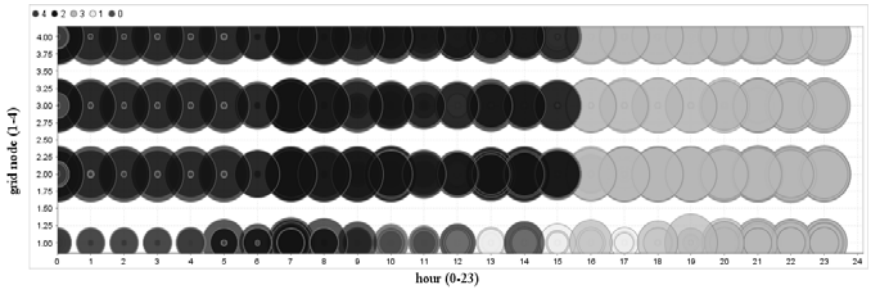


Figure 5. Grid nodes' available performance over the day (clustering model)

All models were applied to the Grid's aggregated data and also to each of the Grid's nodes. The most relevant results are shown below, for both the clustering and decision trees models. Figure 4 shows the Grid's available performance for each test day, determined using the clustering model. The x-axis shows the weekday (from Sunday to Saturday, 1-7). The y-axis represents the time throughout a particular day (from 0.0 to 23.99). Each of the values is represented by a bubble, whose size is relates to the available performance of the Grid, on a particular moment in time. Bubbles are shaded according to the cluster they belong. Again for the clustering model, results for each of the Grid's nodes are shown in figure 5. The x-axis represents the hour of the day (from 0-23). The nodes' identifiers can be found on the vertical axis (from 1-4). Once again, bubble size and shading refer to available performance and cluster membership, respectively. Instead of grouping values in clusters and providing a visual dispersion of these values, decision trees are typically used for classification and prediction. Table 2 shows the accuracy values for each the prediction classes, both per node and for the Grid as a whole, for the two data sets.

Table 2. Decision trees model prediction accuracy

| | Prediction accuracy | | | |
|---|---|---|---|---|
| | Per node | | Grid | |
| Performance class | Test set | Validation set | Test set | Validation set |
| C0 | 88,89% | 92,50% | 57,14% | 70,00% |
| C1 | 0,00% | 0,00% | 80,00% | 50,00% |
| C2 | 60,00% | 90,91% | 100,00% | 100,00% |
| C3 | 0,00% | 0,00% | 0,00% | 20,00% |
| C4 | 63,16% | 65,00% | 63,64% | 100,00% |
| **Global** | **75,70%** | **71,15%** | **65,52%** | **64,00%** |

## 5. Conclusions and Future Work

Clustering models are usually used to explore data. In this work, we neglected the constitution of the clusters in favor of performance patterns. From the results of the clustering

model for the overall Grid (figure 4), it is possible to say that Grid behaved as expected. Non-working days have maximum available performance, with two of the five clusters grouping high performance availability. In the nightly period of working days, the Grid experienced extremely low available performance periods, due to the simulated Nightly application. Even though task submission ended at 2 a.m., the Grid took, in average, almost five additional hours to complete all tasks. This can be explained by the large number of tasks being submitted during the period between 12 a.m. and 2 a.m. Finally, the Grid's behavior on week days is not absolutely homogeneous, since the number and interval between tasks is not deterministic, but random. The analogous analysis for each individual machine (figure 5) revealed that gtnode-1 displayed lower values of performance availability. This can be explained by the load induced by the status data gathering daemon as well as some additional services like Gridway and an X display server.

Although clustering models favor human analysis, decision tree models provide a set of rules that can be easily used in automated analysis. Table 2 shows that our decision tree model was reasonably accurate for predicting a node's performance availability class (more than 70% accurate). The accuracy is not higher mainly due to C1 and C3 classes, in which our model scored 0%. The low accuracy can be explained by the relatively low number of instances of these classes. This fact is due to the nature of our simulated applications which resulted in many more periods of high and low performance availability than those of medium performance availability Results for the entire Grid are not as good (around 65%). Per node predictions and Grid predictions show fairly different accuracy. This can be explained by the lower number of available data instances to analyze, since the Grid's status data instances were obtained by merging and averaging results from all nodes.

The results that we obtained have shown that clustering models can be used to determine which periods have better performance availability and which periods should be avoided if high performance availability is required by GRID applications. Besides exploring the GRID's available performance over time, we have also shown that it is possible to predict that available performance with reasonable accuracy, by using decision tree models to classify performance. These models have the edge over clustering models, since decision tree models can be easily translated to prediction rules, enabling them to be embedded into the GRID schedulers' decision process and potentially increasing efficiency.

Even though we presented valuable initial results, there is still a long way for improvement. Our simulated GRID was based on a low number of nodes. Besides, all nodes shared the same OS and hardware architecture. GRIDs are inherently heterogeneous so these models and techniques must be transposed to heterogeneous and real-scale GRID environments to confirm their worth in predicting the environments' available performance. Additionally, we only processed data for a period of seven days. Analyzing data of far more extended periods can also increase the support on the concepts we presented.

## References

[1] Sterling, T., Becker, D., Savarese, D., Dorband, J., Ranawake, U., Packer, C., 'Beowulf: A Parallel Workstation for Scientific Computation', In Proceedings of the 24th International Conference on Parallel Processing, pp.11-14, 1995.

[2] Kesselman, C., Foster, I., 'The GRID: Blueprint for a New Computing Infrastructure'. San Mateo: Morgan Kaufmann Publishers, 1998.

[3] Baker, M., Apon, A. Ferner, C., Brown, J. 'Emerging GRID Standards', IEEE Computer, vol. 38, no. 4, pp. 43-50, 2005.

[4] Huedo, E., Montero, R., Llorente, I., 'The Gridway Framework for Adaptive Scheduling and Execution on Grids', Scalable Computing: Practice and Experience, vol. 6, nr. 3, pp. 1-8, 2005.

[5] Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S., 'Condor-G: A Computation Management Agent for Multi-Institutional GRIDs', Cluster Computing, vol. 5, nr. 3, pp.237-246, 2002.

[6] Foster, I., 'Globus Toolkit Version 4: Software for Service-Oriented Systems', IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp. 2-13 2006.

[7] Laure, E., Fisher, S., Frohner, A., Grandi, C., Kunszt, P., Krenek, A., Mulmo, O., Pacini, F., Prelz, F., White, J., Barroso, M., Buncic, P., Hemmer, F., Meglio, A., Edlund, A., 'Programming the GRID with gLite', Computational Methods in Science and Technology, vol. 12, nr. 1, pp. 33-45, 2006.

[8] Litzkow, M., Livny, M., Mutka. M., 'Condor - a hunter for idle workstations', 8th International Conference on Distributed Computing Systems, pp. 104-111, 1998.

[9] Henderson, R., 'Job Scheduling Under the Portable Batch System', Lecture Notes in Computer Science, London: Springer-Verlag, vol. 949, pp. 279-294, 1995.

[10] Gentzsch, W., 'Sun GRID Engine: towards creating a compute power grid'. Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the GRID, pp. 35-36, 2001.

[11] Bailey, D., Harris, T., Saphir, W., Wijngaart, R., Woo, A., Yarrow, M., 'The NAS Parallel Benchmarks', The International Journal of Supercomputer Applications, vol. 5, nr. 3, pp. 63-73, 1991.