

# Multi-criteria Grid Resource Management using Performance Prediction Techniques

Krzysztof Kurowski<sup>1</sup>, Ariel Oleksiak<sup>1</sup>, Jarek Nabrzyski<sup>1</sup>,  
Agnieszka Kwiecien<sup>2</sup>, Marcin Wojtkiewicz<sup>2</sup>, Maciej Dyczkowski<sup>2</sup>,  
Francesc Guim<sup>3</sup>, Julita Corbalan<sup>3</sup>, Jesus Labarta<sup>3</sup>

<sup>1</sup> Poznan Supercomputing and Networking Center  
{krzysztof.kurowski,ariel,naber}@man.poznan.pl

<sup>2</sup> Wroclaw Center for Networking and Supercomputing, Wroclaw University of Technology  
{agnieszkakwiecien, marcin.wojtkiewicz, maciej.dyczkowski}@pwr.wroc.pl

<sup>3</sup> Computer Architecture Department, Universitat Politècnica de Catalunya  
{fguim,juli,jesus}@ac.upc.edu

**Abstract.** To date, many of existing Grid resource brokers make their decisions concerning selection of the best resources for computational jobs using basic resource parameters such as, for instance, load. This approach may often be insufficient. Estimations of job start and execution times are needed in order to make more adequate decisions and to provide better quality of service for end-users. Nevertheless, due to heterogeneity of Grids and often incomplete information available the results of performance prediction methods may be very inaccurate. Therefore, estimations of prediction errors should be also taken into consideration during a resource selection phase. We present in this paper the multi-criteria resource selection method based on estimations of job start and execution times, and prediction errors. To this end, we use GRMS [28] and GPRES tools. Tests have been conducted based on workload traces which were recorded from a parallel machine at UPC. These traces cover 3 years of job information as recorded by the LoadLeveler batch management systems. We show that the presented method can considerably improve the efficiency of resource selection decisions

## 1 Introduction

In computational Grids intelligent and efficient methods of resource management are essential to provide easy access to resources and to allow users to make the most of Grid capabilities. Resource assignment decisions should be made by Grid resource brokers automatically and based on user requirements. At the same time the underlying complexity and heterogeneity should be hidden. Of course, the goal of Grid resource management methods is also to provide a high overall performance. Depending on objectives of the Virtual Organization (VO) and preferences of end-users Grid resource brokers may attempt to maximize the overall job throughput, resource utilization, performance of applications etc.

Most of existing available resource management tools use general approaches such as load balancing ([25]), matchmaking (e.g. Condor [26]), computational economy

models (Nimrod [27]), or multi-criteria resource selection (GRMS [28]). In practice, the evaluation and selection of resources is based on their characteristics such as load, CPU speed, number of jobs in the queue etc. However, these parameters can influence the actual performance of applications differently. End users may not know a priori accurate dependencies between these parameters and completion times of their applications. Therefore, available estimations of job start and run times may significantly improve resource broker decisions and, consequently, the performance of executed jobs.

Nevertheless, due to incomplete and imprecise information available, results of performance prediction methods may be accompanied by considerable errors (to see examples of exact error values please refer to [3,4]). The more distributed, heterogeneous, and complex environment the bigger predictions errors may appear. Thus, they should be estimated and taken into consideration by a Grid resource broker for evaluation of available resources.

In this paper, we present a method for resource evaluation and selection based on a multi-criteria decision support method that uses estimations of job start and run times. This method takes into account estimated prediction errors to improve decisions of the resource broker and to limit their negative influence on the performance.

The predicted job start- and run-times are generated by the Grid Prediction System (GPRES) developed within the SGIgrid[30] and Clusterix[31] projects. The multi-criteria resource selection method implemented in the Grid Resource Management System (GRMS) [23,24,28] has been used for the evaluation of knowledge obtained from the prediction system. We used a workload trace from UPC.

Sections of the paper are organized as follows. In Section 2, a brief description of the related activities concerning performance prediction and its exploitation in Grid scheduling is given. In Section 3 the workload used is described. The prediction system and algorithm used for generation of predictions is included in Section 4. Section 5 presents the algorithm for the multicriteria resource evaluation and utilization of the knowledge from the prediction system. Experiments, which we performed, and preliminary results are described in Section 6. Section 7 contains final conclusions and future work.

## **2 Related work**

Prediction techniques can be applied in a wide area of issues related to Grid computing: from the short-term prediction of the resource performance to the prediction of the queue wait time [5]. Most of these predictions are oriented to the resource selection and job scheduling.

Prediction techniques can be classified into statistical, AI, and analytical. Statistical approaches are based on applications that have been previously executed. They can be time series analysis [6,7,8], categorization [4,1,2,22]. In particular correlation and regression have been used to find dependencies between job parameters. Analytical techniques construct models by hand [9] or using automatic code instrumentation [10]. AI techniques use historical data and try to learn and classify the information in order to predict the future performance of resources or applications. AI techniques

are, for instance, classification (decision trees [11], neural networks [12]), clustering (k-means algorithm [13]), etc.

Predicted times are used to predict resource information to guide scheduling decisions. This scheduling can be oriented to load balancing when executing in heterogeneous resources [14,15], applied to resource selection [5, 22], or used when multiple requests are provided [16]. For instance, in [17] authors use the 10-second ahead predicted CPU information provided by NWS [18,8]. Many local scheduling policies, such as Least Work First (LWF) or Backfilling, also consider user provided or predicted execution time to make scheduling decisions [19, 20,21].

### 3 Workload

The workload trace file was obtained in a IBM SP2 System placed at the UPC. It has 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. All nodes are connected through an SP Switch2 operating at 500MB/sec. The operating system that they are running is an AIX 5.1 with the queue system Load Leveler.

The workload was obtained from Load Leveler history files that contained around three years of job executions (178.183 jobs). Through the Load Leveler API, we convert the workload history files that were in a binary format, to a trace file whose format is similar to those proposed [21]. Fields in the workload are: job name, group, username, memory consumed by the job, user time, total time (user+system), tasks created by the job, unshared memory in the data segment of a process, unshared stack size, involuntary context switches, voluntary context switches, finishing state, queue, submission date, dispatch time, and completion date. More details on the workload can be found in [29].

Analyzing the trace file we can see that total time for parallel jobs is approximately an order of magnitude bigger than the total time for sequential jobs, what means that in median they are consuming around 10 times more of CPU time. For both kind of jobs the dispersion of all the variables is considerable big, however in parallel jobs is also around an order of magnitude bigger. Parallel jobs are using around 72 times more memory than the sequential applications, also the IQR is bigger<sup>1</sup>. In general these variables are characterized by significant variance what can make their prediction difficult.

Users submit jobs with various levels of parallelism. However, there is an important amount of jobs that are sequential (23%). The more relevant parallel jobs that are consuming more resources belong to three main number of processor usage intervals: 5-15 processors (31% of the total jobs), 65-128 processors (29% of the total jobs) and 17-32 processors (13% of the total jobs).

In median all the submitted LoadLeveler scripts used to be executed one time with the same number of tasks. This fact could imply that this last variable would be not

---

<sup>1</sup> The IQR is defined as  $IQR=Q3-Q1$ , where:  $Q1$  is a value such as a the exactly only 25% of the observations are less than it, and the  $Q3$  is a value such as exactly on 25% of the observations are greater than it's bigger on these first one.

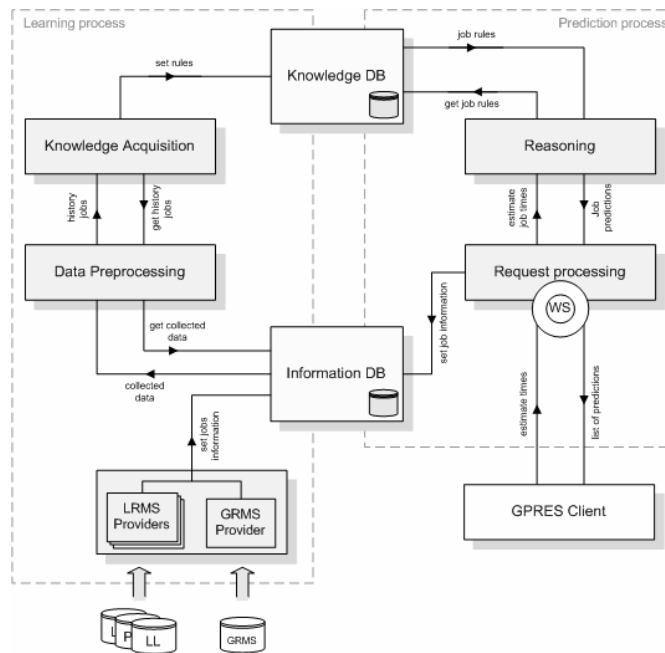
significant to be used for forecasting. However those jobs that were executed with 5-16 and 65-128 processors are executed in general more than 5 times with the same number of tasks, and represent the 25 % of the submitted jobs. This can suggest that this variable may be relevant.

## 4 Prediction System

This section provides a description of the prediction system that has been used for estimating start and completion times of the jobs. Grid Prediction System (GPRES) is constructed as an advisory expert system for resource brokers managing distributed environment, including computational Grids.

### 4.1 Architecture

The architecture of GPRES is based on the architecture of expert systems. With this approach the process of knowledge acquisition can be separated from the prediction. The Figure 1 illustrates the system architecture and how its components interact with each other.



**Fig. 1.** Architecture of GPRES system

Data Providers are small components distributed in the Grid. They gather information about historical jobs from logs of GRMS and local resource management systems (LRMS, e.g. LSF, PBS, LL) and insert it into Information data base. After the

information is gathered the Data Preprocessing module prepares data for a knowledge acquisition. Jobs' parameters are unified and joined (if the information about one job comes from several different sources, e.g. LSF and GRMS). Such prepared data are used by the Knowledge Acquisition module to generate rules. The rules are inducted into the Knowledge Data Base. When an estimation request comes to GPRES the Request Processing module prepares all the incoming data (about a job and resources) for the reasoning. The Reasoning module selects rules from the Knowledge Data Base and generates the requested estimation.

## 4.2 Method

As in previous works [1, 2, 3, 4] we assumed that the information about historical jobs can be used to predict time characteristics of a new job. The main problem is to define the similarity of the jobs and to select appropriate parameters to evaluate it.

GPRES system uses a template-based approach. The template is a subset of job attributes, which are used to evaluate jobs' "similarity". The attributes for templates are generated from the historical information after tests.

The knowledge in the Knowledge Data Base is represented as rules:

*IF*  $A_1 op v_1$  *AND*  $A_2 op v_2$  *AND* ... *AND*  $A_n op v_n$  *THEN*  $d = d_i$ , where  $A_i \in A$ , the set of condition attributes,  $v_i$  – values of condition attributes,  $op \in \{=, >, <\}$ ,  $d_i$  – value of decision attribute,  $i, n \in N$ .

One rule is represented as one record in a data base. Several additional parameters are set for every rule: a minimum and maximum value of a decision attribute, standard deviation of a decision attribute, a mean error of previous predictions and a number of jobs used to generate the rule.

During the knowledge acquisition process the jobs are categorized according to templates. For every created category additional parameters are calculated. When the process is done the categories are inserted into the Knowledge Data Base as rules.

The prediction process uses the job and resource description as the input data. Job's categories are generated and the rules corresponding to categories are selected from the Knowledge Data Base. Then the best rule is selected and used to generate a prediction. Actually there are two methods of selecting the best rule available in GPRES. The first one prefers the most specific rule, with the best matching to condition attributes of the job. The second strategy prefers a rule generated from the highest number of history jobs. If both methods don't give the final selection, the rules are combined and the arithmetic mean of the decision attribute is returned.

## 5 Multi-criteria prediction-based resource selection

Knowledge acquired by the prediction techniques described above can be utilized in Grids, especially by resource brokers. Information concerning job run-times as well as a short-time future behavior of resources may be a significant factor in improving the scheduling decisions. A proposal of the multi-criteria scheduling broker that takes the advantage of history-based prediction information is presented in [22].

One of the simplest algorithms which requires the estimated job completion times is the Minimum Completion Time (MCT) algorithm. It assigns each job from a queue to resources that provide the earliest completion time for this job.

#### Algorithm MCT

For each job  $J_i$  from a queue  
 For each resource  $R_j$ , at which this job can be executed  
 Retrieve estimated completion time of job  $C_{J_i, R_j}$   
 Assign job  $J_i$  to resource  $R_{best}$  so that

$$C_{J_i, R_{best}} = \min_{R_j} \left( C_{J_i, R_j} \right)$$

Nevertheless, apart from predicted times, the knowledge about potential prediction errors is needed. The knowledge coming from a prediction system shouldn't be limited only to the mean times of previously executed jobs that fit to a template. Therefore, we also consider minimum and maximum values, standard deviation, and estimated error (as explained in Section 4.2). These parameters should be taken into account during a selection of the most suitable resources. Of course, the mean time is the most important criterion, however, relative importance of all parameters depends on user preferences and/or characteristics of applications. For instance, certain applications (or user needs) may be very sensitive to delays, which can be caused by incorrectly estimated start and/or run times. In such case a standard deviation, minimum and maximum values become important. Therefore, a multi-criteria resource selection is needed to accurately handle these dependencies. General use of multi-criteria resource selection methods in Grids was described in [23].

In our case we used the functional model for aggregation of preferences. That means that we used a utility function and we ranked resources based on its values. In detail, criteria are aggregated for job  $J_i$  and resource  $R_j$  by the weighted sum given according to the following formula:

$$F_{J_i, R_j} = \frac{1}{\sum_{k=1}^n w_k} \sum_{k=1}^n w_k * C_k \quad (1)$$

where the set of criteria  $C$  ( $n=4$ ) consists of the following metrics:

$C_1$  – mean completion time ( $time_{J_i, R_j}$ )

$C_2$  – standard deviation of completion time ( $stdev_{J_i, R_j}$ )

$C_3$  – difference between maximum and minimum values of completion time ( $max_{J_i, R_j} - min_{J_i, R_j}$ )

$C_4$  – estimated error of previous predictions ( $err_{J_i, R_j}$ )

and weights  $w_k$  that define the importance of the corresponding criteria.

This method can be considered as a modification of the MCT algorithm to a multi-criteria version. In this way possible errors and inaccuracy of estimations are taken into consideration in MCT. Instead of selection of a resource, at which a job completes earliest, the algorithm chooses resources characterized by the best values of the utility function  $F_{J_i, R_j}$ .

Multi-criteria MCT algorithm

```
For each job  $J_i$  from a queue
  For each resource  $R_j$ , at which this job can be
  executed
    Retrieve estimated completion time of job  $C_{J_i,R_j}$ 
    and  $err_{J_i,R_j}$ ,  $stdev_{J_i,R_j}$ ,  $max_{J_i,R_j}$ ,  $min_{J_i,R_j}$ 
    Calculate the utility function  $F_{J_i,R_j}$ 
  Assign job  $J_i$  to resource  $R_{best}$  so that
```

$$F_{J_i,R_{best}} = \max_{R_j} \left( F_{J_i,R_j} \right)$$

## 6 Preliminary Results

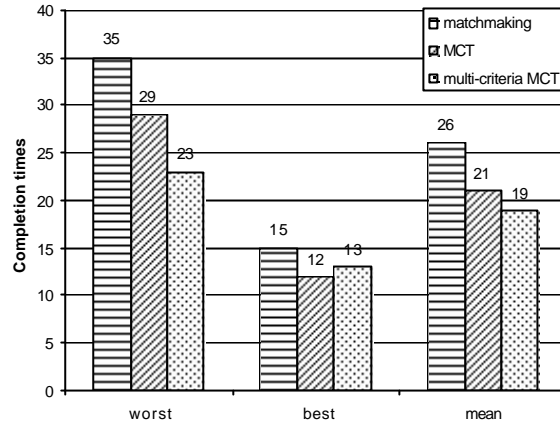
There are two main hypothesis of this paper defined. First, use of knowledge about estimated job completion times may significantly improve resource selection decisions made by resource broker and, in this way, the performance of both particular applications and the whole VO. Nevertheless, estimated job completion times may be insufficient for effective resource management decisions. Therefore, the second hypothesis is that results of these decisions may be further improved by taking the advantage of information about possible uncertainty and inaccuracy of prediction.

In order to check these hypothesis we performed two major experiments. First, we compared results obtained by the MCT algorithm with a common approach based on the matchmaking technique (job was submitted to the first resource that met user's requirements). In the second experiment, we studied improvement of results of the prediction-based resource evaluation after application of knowledge about possible prediction errors. For both experiments the following metrics were compared: mean, worst, and best job completion time. The worst and best job completion values were calculated in the following way. First, for each application the worst/best job completion times have been found. Second, an average of these values was taken as the worst and best value for comparison.

5000 jobs from the workload were used to acquire knowledge by GPRES. Then 100 jobs from the workload were scheduled to appropriate queues using methods presented in Section 5.

The results of the comparison are presented in Figure 2. In general, it shows noticeable improvement of mean job completion times when the performance prediction method was used.

The least enhancement was obtained for the best job completion times. The multi-criteria MCT algorithm turned out to be the most useful for improvement of the worst completion times. Further study is needed to test the influence of relative importance of criteria on final results.



**Fig. 2.** Comparison of job completion times for matchmaking, MCT, and multi-criteria MCT algorithms

## 7 Conclusion

In this paper we proposed the multi-criteria resource evaluation method based on knowledge of job start- and run-times obtained from the prediction system. As a prediction system the GPRES tool was used. We exploited the method of multi-criteria evaluation of resources from GRMS.

The hypotheses assumed in the paper have been verified. Exploitation of the knowledge about performance prediction allowed a resource broker to make more efficient decisions. This was visible especially for mean values of job completion times.

Exploitation of knowledge about possible prediction errors brought another improvement of results. As we had supposed it improved mainly the worst job completion times. Thus, taking the advantage of knowledge about prediction errors we can limit number of job completion times that are significantly worse than estimated values. Moreover, we can tune the system by setting appropriate criteria weights depending on how reliable results we need and how sensitive to delays application are. For instance, certain users may accept “risky” resources (i.e. only the mean job completion time is important for them) while others may expect certain reliability (i.e. low ratio of strongly delayed jobs).

The advantage of performance prediction methods is less visible for strongly loaded resources because many jobs have to be executed at worse resources. This drawback could be partially eliminated by scheduling a set of jobs at the same time. This approach will be a subject of further research. Of course, information about possible prediction errors is the most useful in case of inaccurate predictions. If a resource broker uses high quality predictions, knowledge of estimated errors becomes less important.

Although a substantial improvement of the performance were shown, these results are rather still far from users' expectations. This is caused by , among others, a quality of available information. Most of workloads (including the LoadLeveler workload used for our study) do not contain such essential information as number of jobs in queues, size of input data, etc. Exploitation of more detailed and useful historical data is also foreseen as the future work on improving efficiency of Grid resource management based on performance prediction.

## Acknowledgement

This work has been supported by the CoreGrid, network of excellence in "Foundations, Software Infrastructures and Applications for large scale distributed, Grid and Peer-to-Peer Technologies", the Spanish Ministry of Science and Education under contract TIN2004-07739-C02-01, and SGIgrid and Clusterix projects funded by the Polish Ministry of Science.

## References

1. Allen Downey, "Predicting Queue Times on Space-Sharing Parallel Computers". In International Parallel Processing Symposium, 1997.
2. Richard Gibbons. "A Historical Application Profiler for Use by Parallel Schedulers". Lecture Notes on Computer Science, pages 58-75, 1997.
3. Warren Smith, Valerie Taylor, Ian Foster. "Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance". In Proceedings of the IPPS/SPDP '99 Workshop on Job Scheduling Strategies for Parallel Processing.
4. Warren Smith, Valerie Taylor, Ian Foster. "Predicting Application Run -times Using Historical Information" In Proceedings IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.
5. I. Foster and C. Kesselman. Computational grids. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 15--52. Morgan Kaufmann, San Francisco, California, 1986.
6. R. Wolski, N. Spring, and J. Hayes. Predicting the CPU availability of time-shared unix systems. In submitted to SIGMETRICS '99 (also available as UCSD Technical Report Number CS98-602), 1998.
7. P. Dinda. Online prediction of the running time of tasks. In Proc. 10th IEEE Symp. on High Performance Distributed Computing 2001
8. R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15 (5-6):757-768 1999
9. J. Schopf and F. Berman. Performance prediction in production environments. In Proceedings of IPPS/SPDP, 1998.
10. V. Taylor, X. Wu, J. Geisler, X. Li, z. Lan, M. Hereld, I. Judson, and R. Stevens. Prophecy: Automating the modeling process. In Proc. Of the Third International Workshop on Active Middleware Services, 2001.
11. J.R. Quinlan. Induction of decision trees. *Machine Learning*, pages 81-106, 1986
12. D.E.Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back propagating errors. *Nature*, 323:533-536, 1986

13. C.Darken, J.Moody: Fast adaptive KMeans Clustering: some Empirical Results, Proc. International Joint Conference on Neural Networks Vol II, San Diego, New York, IEEE Computer Scienc Press, pp.233-238, 1990.
14. H.J.Dail. A Modular Framework for Adaptive Scheduling in Grid Application Development Environments. Technical report CS2002-0698, Computer Science Department, University of California, San Diego, 2001
15. S. M. Figueira and F. Berman, Mapping Parallel Applications to Distributed Heterogeneous Systems, Department of Computer Science and Engineering, University of California, San Diego, TR - UCSD - CS96-484, 1996
16. K. Czajkowski, I. Foster, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., JSSPP Whorskshop. LNCS #1459 pages 62-68. 1997.
17. C. Liu, L. Yang, I. Foster, D. Angulo., Design and Evaluation of a Resource selection Framework for Grid Applications. In Proceedings if the Eleventh IEEE International Symposium on High-Performance Distributed Computing (HPDC 11), 2002
18. R. Wolski. Dynamically Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service. In 6th High-Performance Distributed Computing, Aug. 1997.
19. D. Lifka, "The ANL/IBM SP scheduling system ". In Job Scheduling Strategies for Parallel Processing, D. G. Feitelson and L. Rudolph (eds.), pp. 295--303, Springer-Verlag, 1995. Lect. Notes Comput. Sci. vol. 949
20. D. G. Feitelson and A. Mu'alem Weil. Utilization and predictability in scheduling the IBM SP2 with backfilling. In Proc. 12th Int'l. Parallel Processing Symp., pages 542--546, Orlando, March 1998.
21. D.G.Feitelson. Parallel Workload Archive. <http://www.cs.huji.ac.il/labs/parallel/workload>
22. K. Kurowski, J. Nabrzyski, J. Pukacki, Predicting Job Execution Times in the Grid, in Proceedings of the 1st SGI 2000 International User Conference, Kraków, 2000
23. K. Kurowski, J. Nabrzyski, A. Oleksiak, and J. Weglarz., "Multicriteria Aspects of Grid Resource Management", In *Grid Resource Management* edited by J. Nabrzyski, J. Schopf, and J. Weglarz, Kluwer Academic Publishers, Boston/Dordrecht/London, 2003.
24. Kurowski, K., Ludwiczak, B., Nabrzyski, J., Oleksiak, A., Pukacki, J.: "Improving Grid Level Throughput Using Job Migration and Rescheduling Techniques in GRMS". *Scientific Programming. IOS Press.* Amsterdam The Netherlands 12:4 (2004) 263-273
25. B. A. Shirazi, A. R. Husson, and K. M. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems.* IEEE Computer Society Press, 1995.
26. Condor project. <http://www.cs.wisc.edu/condor>.
27. D. Abramson, R. Buyya, and J. Giddy. A computational economy for Grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems*, 18(8), October 2002.
28. Grid Resource Management System (GRMS), <http://www.gridlab.org/grms>.
29. F.Guim, J. Corbalan, J. Labarta. Analyzing LoadLeveler historical information for performance prediction. In Proc. Of Jornadas de Paralelismo 2005. Granada, Spain
30. SGIgrid project. <http://www.wcss.wroc.pl/pb/sgigrid/en/index.php>
31. Clusterix project. <http://www.clusterix.pcz.pl>