Grid Application Performance Prediction: a Case Study in BROADEN

Karim Djemame and Mohammed H Haji
School of Computing, University of Leeds, Leeds LS2 9JT, UK
Email: {karim,mhh}@comp.leeds.ac.uk

ABSTRACT

The BROADEN (Business Resource Optimisation for Aftermarket and Design on Engineering Networks) project aims to build a Rolls-Royce Grid as a proving ground for utilising Grid services technology. One of the central applications in this project is XTO ((eXtract Track Order) which is used to facilitate distributed diagnostics support for aircraft engines. One of the key elements in BROADEN is the ability to predict XTO applications run-time. In this paper a SNAP-base (Service Negotiation Acquisition Protocol) resource broker is extended to include a QoS (Quality of Service) Manager which utilises a run-time historical database that allows for such prediction to take place. The performance results presented in terms of predicted run-time that the use of this manager can provide show the potential of this approach on a Grid test-bed.

KEY WORDS: Grid computing, resource broker, Quality of Service, prediction methods, historical data.

1.0 INTRODUCTION

Grid computing has the potential to provide users with high performance, utility computing in a seamless virtual organisation (VO) (Foster 2003). In such environments users aim to negotiate with resource providers, often belonging to different administrative domains in order to achieve reliable, predictable application performance. More specifically, users will typically have commitments and performance requirements (e.g. time deadlines attached to their applications). A key goal of resource management within the Grid is to deliver commitments and assurances on top of the allocated resources, for example, compute and storage resources, security and network performance and assurance that a specified time deadline can be met. This is of particular importance to users from a commercial environment, since a failure to complete an application in a timely fashion may have severe implications. An example is the BROADEN (Business Resource Optimisation for Aftermarket and Design on Engineering Networks) project (Fletcher 2006), which is used in this paper to provide our experimental evaluation scenario.

The task of providing commitments and assurances is non-trivial, since a Grid system can integrate heterogeneous resources with varying quality and availability. This places importance on the system's ability to monitor the state of these resources. The Grid is a dynamic system where resources are subjected to changes due to system performance degradation, system failure, etc.

This paper focuses on Quality of Service provision in the Grid. To do so, it is important to address the dynamic and heterogeneous nature of the Grid. This is a key issue: 1) before an application begins execution for best resource selection; 2) during run-time in case support of adaptation is provided, and 3) post run-time as information regarding the application, run-time, environment etc is stored as historical data that can be used in future application runs for knowledge extraction, helping to determine whether the user's requirement will be met. It is envisaged that Grid implementations will operate within an economic framework (Buyya 2005). This means cost/performance trade-off decisions must be made pre run-time, requiring mechanisms to support performance prediction. The paper looks specifically into post run-time information regarding the application execution environment including execution time. A key feature of our approach is that the user is not required to install additional software, or make

alterations to their code requiring specialist Grid computing knowledge. This is achieved through a *Quality of Service (QoS) Manager* which is integrated within a Grid resource broker (Haji 2005).

The QoS Manager implementation in a real Grid environment is described in detail and performance results are presented in terms of predicted execution time that the use of this manager can provide. Our results show the potential of this approach on a Grid test-bed. The application used in these experiments is XTO (eXtract Tracked Orders) (Austin 2003) which is an engine data analysis tool that captures data monitored and recorded by the QUICK on-engine system during a flight (Nairac 1999), and is of considerable importance in the BROADEN project.

The paper is organised as follows. Related work is described in Section 2. This is followed by a discussion of the BROADEN project, which provides motivation and context for the research and introduces the application which is used in the experiments. Section 4 presents the QoS manager design and prediction methods used for run-time application. Section 5 presents experimental results, highlighting the key issues that must be accounted for in a Grid environment. Conclusions and future work are presented in Section 6.

2.0 RELATED WORK

Quality of Service (QoS) provision through resource management in the Grid has received considerable attention from the research community (Nabrzyski 2004). A number of projects have investigated scheduling on the Grid and include Nimrod-G (Buyya 2000), Condor-G (Thain 2005), Gridway (Huedo 2006) and Globus (Foster 2005). A number of works have also explored the problem of deadline scheduling in real-time systems (Aydin 1999). However, they mostly consider deadline parameters for individual jobs and are restricted to centralised resource management schemes and single administrative domain resources.

AppLeS (Berman 2003) has been developed to enable resource selection based on meeting user QoS requirements (e.g. execution time, turnaround time) and adapt to changes in resource availability. However, performance prediction used is developed with specific applications in mind. In addition, applications must be “AppLeS enabled” which means making alterations to the user’s application code. GrADS (Grid Application Development Software) (Berman 2001) aim to provide a framework to enable efficient Grid application execution. It supports run-time monitoring and comparison with performance contracts, so that appropriate adaptive actions (e.g. application tuning, migration) can be performed during run-time. However, their approach to performance prediction differs in that historical compute resource performance and load variations are not accounted for (Vraalsen 2001). In addition, like AppLeS, the use of GrADS requires the user’s application to be altered in order to utilise the GrADS APIs.

Faerman et al (Faerman 1999) presents an application performance prediction method for determining file transfer times. However, the paper does not discuss resource selection, based on the behaviour (i.e. past and current performance) of compute resources. In (Smith 1998) methods for predicting run-times for parallel applications, using historical information, are discussed. Execution times of ‘similar’ applications run in the past are used to estimate the execution time. The use of historical information to determine resource reliability is not considered. Application of such methods to Grid scheduling is considered in [Smith 2004]. Specifically, run-time prediction of pending jobs is used to estimate execution start times by predicting queue waiting times. While their results indicate that this approach is useful in decreasing waiting times, it does not provide information as to the anticipated execution time of the user’s job. The performance prediction based tool, PACE (Performance Analysis and Characterisation Environment) (Nudd 2004), uses a combination of source code analysis and hardware modelling to provide an application performance prediction. The hardware models are static, this provides the advantage of reusability but does not account for dynamic changes to resource performance. The application of three learning techniques to estimate the resource requirements of any given application run before a scheduling decision is made is found in (Kapadia 1999).

3.0 BROADEN OVERVIEW

The importance for supporting run-time prediction can be seen by considering the requirements of a commercial Grid project such as BROADEN, a multi-site, DTI funded project which aims to build a Rolls Royce Grid as a proving ground for utilising Web/Grid services technology to fully exploit available IT resources. This is to support the following:

- Integrating diagnostic tools for health monitoring across development test and in-service aero-engines;
- Large scale and innovative numerical simulations of high fidelity Computational Fluid Dynamics (CFD) and design optimisation;
- Large scale agent-based modelling of aftermarket processes incorporating logistics and the supply chain.

BROADEN focuses on three distinct development areas – distributed diagnostics for engine health monitoring, high-performance computing for design, and agent-based software development for business modelling. On the distributed diagnostics for engine health monitoring side it builds on the DAME (Distributed Aircraft Maintenance Environment) (Austin 2003) e-science project, which aimed to develop a generic test bed for distributed diagnostics. The generic framework is deployed in a proof of concept demonstrator in the context of maintenance applications for civil aerospace engines. The essential theme of DAME was the use of real-time intelligent feature extraction, intelligent data mining and decision support techniques, where expertise and software tools are distributed across the Grid. The enormity of the databases and the need for distributed access to the data make this a particularly challenging problem for the Grid. DAME has demonstrated the potential of Grid-based diagnostics for health-monitoring applications and shown how data growth problems could potentially be addressed within scope of distributed data assets.

Access to distributed data is typically as important as access to distributed computational resources. Applications for distributed diagnostics require transfers of large amount of data between storage systems, and access to large amounts of data by many geographically distributed applications and users for analysis and visualisation.

The XTO (eXtract Track Orders) forms the central application within the project and is used to support distributed diagnostics support for aircraft engines. The XTO analyses vibration data produced by aircraft engines for features that can be used to support the process of making a fault diagnosis. This data captured is engine information monitored and recorded by an on-engine system during a flight, which is stored in a file system and consists of a single control file and a number of binary files that are all in the Rolls-Royce proprietary ZMOD format.

A simple scenario which forms part of a sophisticated process involved in using BROADEN to support engine fault detection and diagnosis is the following:

1. Data is downloaded from an aircraft;
2. A novel event is flagged and marked in the data set;
3. A search through historical data fails to identify the cause of the anomaly. Therefore the system operator launches a feature analysis session, which uses the XTO application;
4. A diagnosis is made, based on the features detected by this analysis.

In this scenario timely completion of the XTO application (and any others applications used in a diagnosis session) is critical since failure to make a diagnosis within the engine turnaround time could result, for example, in flight delays, leading a revenue loss. Thus the work is to look into ways of providing a run-time prediction mechanism for XTO application under various plug-ins and execution environments.

3.1 BROADEN Tool, Service and Data Architecture

A generic distributed tool, service and data architecture is being developed with aero-engine health monitoring as the exemplar domain (Figure 1).

The architecture permits the integration of tools to the system with the minimum of change to the tools. Underlying the use of tools is the provision of services, which may be centralised, distributed or autonomous. The volume of the ZMOD data downloadable from the QUICK system may be up to 1 GB per engine per flight. It is envisaged the storage

requirements of an aero-engine application alone for an operational day is around 1 TB. Therefore the architecture permits the use of distributed nodes to store data (and services), and ensures other issues such as security, etc are satisfied. The Storage Resource Broker (SRB) (SDSC 2006) provides the mechanism to virtualise the storage across nodes.

FIGURE 1: BROADEN Tool, Service and Data Architecture.

The scenario used for BROADEN envisages geographically dispersed users using a workbench and tools which access distributed services and data. A middleware layer addresses work orchestration issues such as starting, stopping, monitoring and dynamic workflow capabilities. This is achieved thanks to: 1) the Enterprise Service Bus (Chappell 2004) which provides XML messaging, XML transformation, intelligent routing, connectivity, and is used within the architecture as a simple messaging and translation mechanism between the tools of the system, and 2) a Process Management Controller (PMC), a lightweight middleware application for managing services which act on distributed data in a generic manner and for providing high performance communication (Austin 2005).

Of particular interest in the resource broker which manages selection of Grid resources in keeping with specified Service Level Agreements (SLAs) (see Section 4).

An overview of the tools, services, and data including the issues and rationale behind the development of the architecture is found in (Fletcher 2006).

4.0 QUALITY OF SERVICE PROVISION

BROADEN applications can be executed on demand using computational resources that are hidden to the user. The type of applications can vary from computationally intensive simulations to high priority time critical executions. The resources on which these services are executed can vary in quality and reliability, particularly if demand for these application services is high. In this situation competition for resources is high and they can be easily swamped, leading to a drop in application performance. If this coincides with the execution of a time critical application, the results may be delayed. Therefore QoS requirements may be attached to applications in order to guarantee timely execution. In (Padgett 2006) we proposed a Service Level Agreement (SLA) management system incorporating resource reservation and run-time adaptation. For this system to function, commitments and assurances are specified using Service Level Agreements (SLA). An SLA is a contract between user and provider, stating the expectations that exist between them.

Sharing resources in a Grid is complicated in that it requires the ability to bridge the differing policy requirements of the resource owner, in order to create a consistent cross organisation policy domain that delivers the necessary capability to the end user, while respecting the policy requirements of the resource owner. Further complicating the management of Grid resources is the fact that Grid jobs often require the concurrent allocation of multiple resources. This need for simultaneous resource usage necessitates a structured

framework in which resources can be co-ordinated across administrative domains. This is addressed through the use of a resource broker developed within the SNAP framework (Czajkowski 2002). The SNAP-based resource broker (Haji 2005), unlike other brokers presented in section 2, caters for the three fabric QoS essentialities known as pre-runtime, run-time, and post-runtime phases, through its TSLA (Task Service Level Agreement) RSLA (Resource Service Level Agreement) and BSLA (Binding Service Level Agreement). An overview of the broker can be seen in Figure 2.

FIGURE 2: An overview of a SNAP-based resource broker.

The work presented in this paper extends the SNAP-based broker by integrating a QoS manager, which has the responsibility to gather historical data of past job execution run times. This is achieved through a Run-time Historical database that captures information as records including CPU load, speed and type, cache size, hostname, etc as well as the actual execution run-time. The responsibility of the QoS manager is to apply various statistical approaches (such as those mentioned in Section 5.0) on the data stored in the historical database and to predict the two most important factors surrounding the XTO application: 1) a run-time prediction, and 2) a storage capacity prediction needed to store the volume of output data.

In the following the approach taken is to formulate a run-time prediction for XTO application. A performance methodology is therefore developed to choose the most appropriate method to predict run-time. The run-time prediction from observations of past runs can be used by the BROADEN end-user. This performance information can be used in its other functionalities, including scheduling and contributing to the guarantee of QoS contracts. The run-time prediction is based on collecting information for each XTO application run and then applying predictive methods to the previous observations. The gathering of performance information does not affect the behaviour of the BROADEN system in any way. Three prediction methods are used to estimate future run-times (Dushay 1999):

1. *Last observation*: the most recent, single performance observation value is taken as the prediction. The last performance value is most likely to reflect the behaviour of future runs:

$$P_n = V$$

2. *Sample average*: the prediction is the mean average of the past performance values within a sample set. This set is defined by a sliding window of size x , which corresponds to the x most recent observations. This method is used when performance information is produced on a regular basis, and old value become less relevant. An average can be used with a maximum window size:

$$P_n = (\sum V_i) / x \quad i = 1..x$$

3. *Low Pass Filter*: recent performance data constitutes a better predictor than older data. This method uses an exponentially degrading function to obtain average of recent performance behaviour, thanks to the following formula:

$$P_n = w \cdot P_{n-1} + ((1 - w) \cdot V)$$

where:

P_n is the prediction and the new value of the low pass filter

P_{n-1} is the previous filter value

V is the most recent performance observation value

W is the weighting parameter and is a value between 0 and 1. It is generally equal to 0.95, thus decreasing the value of the weight as observation values grow older and increasing the prediction accuracy.

All three methods are simple, use a combination of past performance data for a given application run and require little processing time, in the order of seconds depending on the size of the historical database.

5.0 EXPERIMENTAL DESIGN AND PERFORMANCE RESULTS

5.1 Overview and Objectives

The prediction methods described in Section 4 have different capabilities in terms of the concepts that they can represent. Experiments were designed on a Grid test-bed to highlight their accuracy as well as their differences.

The experiments made use of ten different engine data with various input sizes, each using four different plugins (Fractional, Multiple, Residual_Energy and Step_Change) with a total of fifteen different combinations. The experiments ran for a period of several weeks at various times of the day, and generated over 45,000 different statistical records. The records were stored in the run-time historical database for use by the QoS manager (see Figure 2). Due to space limitation three different data engine sizes (large, medium and small) are presented in this paper, referred to as 7101xyz1, 7101xyz2 and 7101xyz3 respectively. With this in mind the experiments are designed on the basis of the following objectives:

- To gather statistical data on XTO application run-time based on engine data size using various plugins and to observe the accuracy of the prediction algorithms used, namely last observation, sample average and low pass filter;
- To gather statistical data on XTO application output data generated after each experimental run.

The experiments were performed on a Grid test-bed consisting of 10 machines, each having a Pentium IV processor (1.2GHz) and 256MB RAM. The operating system is Linux 2.4 and all machines have Globus 4 (Foster 2005) installed, with Sun Grid Engine (SGE) as the job scheduler.

5.2 Performance Results

The results in Figures 3-5 show the output data generated for three engine data size (large, medium and small) with 15 different plugins combinations. Clearly there is a pattern similarity in all three figures (apart from the obvious scale of the data output size) with the single plugin *Multiple* generating the largest output data, and *Residual_Energy* generating the smallest. It is this type of data that the QoS manager needs to capture for it to be able to predict and reserve sufficient disk space for future runs. This is particularly important for any new engine data introduced into the BROADEN system. The result of one plugin will enable the QoS manager to estimate the output data for the other 14 remaining combinations that will be applied to that engine.

FIGURE 3: Output data generated for engine number 7101xyz1 (large), based on various plugin combinations.

Figure 4: Output data generated for engine number 7101xyz2 (Medium), based on various plugin combinations.

Figure 5: Output data generated for engine number 7101xyz3 (Small), based on various plugin combinations.

Figures 6-8 present four performance results per plugin, namely the actual real run-time and the different predicted run-times generated by the QoS manager prior to the job execution. It can be seen in Figure 6 that the most accurate prediction for the majority of plugin combinations was the last observation. However in Figures 7 and 8 the best prediction method was the low pass filter. From the experiments it is observed that the most appropriate method for run-time prediction is based on how recent the data was captured and available to use by the QoS manager. Hence if it is recent in relation to the actual application execution (within a range of hours) then the last observation prediction method is appropriate, otherwise the low pass filter provides a better prediction. Further there is a pattern in the performance results presented in Figures 6-8; however it is not as significant and well pronounced as that in Figures 3-5. This shows output data provides little insight to run-time prediction. Conversely Figures 6-8 do provide the QoS manager information to estimate the run-time of a new engine with one set of plugin results entered into the system as the time range differences for all plugins based on a single engine data are marginally small.

Figure 6: Actual run-time compared to the predicted times for engine number 7101xyz1 (Large), based on various plugin combination.

Figure 7: Actual run-time compared to the predicted times for engine number 7101xyz2 (Medium), based on various plugin combinations.

Figure 8: Actual run-time compared to the predicted times for engine number 7101xyz3 (Small), based on various plugin combinations.

6.0 CONCLUSION AND FUTURE WORK

A key goal of Grid resource management within BROADEN is to deliver commitments and assurances on top of the allocated resources, and assurance that a specified time deadline can be met pre application run-time. This is of particular importance to BROADEN users from a commercial environment, since a failure to complete an application in a timely fashion may have severe implications.

The approach taken in this paper is to predict XTO application run-time as well as output data size generated from a Grid application's perspective, based on past performance

data. A QoS manager makes use of three prediction methods, which may help a resource broker in scheduling decisions. This informed choice further contributes to guarantee QoS in the use of Grid middleware.

Experiments have been designed to gather statistical data based on engine data size using various plugins and to observe the accuracy of the prediction algorithms used, namely last observation, sample average and low pass filter. Using past performance data, the experiment results have been analysed. It has been found that the accuracy of the predictors was somehow different. The low pass prediction method was the most accurate method, with the last observation being on slightly less accurate but nevertheless appropriate if it is recent in relation to the actual application execution (within a range of hours).

The primary utility of this work is that it quantifies the accuracy of different prediction methods as well as the consistency of observed data from an operational Grid application. None of the prediction algorithms we used required on-going heavy computation, and the QoS manager does benefit the users of the BROADEN decision support system. The XTO application which was previously provided on a best-effort basis can now be offered with QoS guarantees.

The broker has been recently deployed on Rolls-Royce Grid which is heterogeneous in nature and consists of different clusters with various architectures and operating systems. Therefore further experiments will be designed to gather statistical data on XTO application run-time taking into consideration dynamic resource performance and volatility. A thorough evaluation of the historical observed data (CPU load, speed and type, cache size, hostname, run-time) is envisaged in order to further investigate performance prediction and nodes configurations.

Future work will consist of the integration of the QoS manager within an SLA management to monitor run-time application performance. Further the broker will be enhanced to cater for other Rolls-Royce applications such as compute intensive Message Passing Interface (MPI) Computational Fluid Dynamics.

Acknowledgements

The work reported in this paper was developed and undertaken as part of the BROADEN (Business Resource Optimization for Aftermarket and Design on Engineering Networks) project. The BROADEN project is part-funded via the UK Department of Trade and Industry under its Technology Innovation Programme. The work described was undertaken by teams at the Universities of Leeds, York, and Sheffield with industrial partners Rolls-Royce, EDS, Oxford BioSignals and Cybula Ltd.

References

- Aydin, H., Melhem, R.G., Mosse, D. and Mejia-Alvarez, P. (1999) Optimal Reward-Based Scheduling of Periodic Real-Time Tasks. In *Proceedings of the IEEE Real-Time Systems Symposium*, Phoenix, AZ, December 1999, pp. 79-89.
- Austin, J., Jackson, T., Fletcher, M., Jessop, M., Cowley, P. and Lobner (2003) P. Predictive Maintenance: Distributed Aircraft Engine Diagnostics. In: I. Foster and C. Kesselman (Eds.). *Grid2: Blueprint for a New Computing Infrastructure*. Chapter 5, Morgan Kaufmann, 2nd edition, 2003.
- Austin et al (2005) DAME: Searching Large Data Sets within a Grid-Enabled Engineering Application. *Proceedings of the IEEE*, Vol. 93, No.3, March 2005
- Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Mellor-Crummey, J., Reed, D., Torczon, L. and Wolski, R. (2001) The GrADS Project: Software Support for High-Level Grid Application Development. *International Journal of High Performance Computing Applications*, 15(4):327- 344, 2001

Berman, F., Wolski, R., Casanova, H., Dail, H., Faerman, M., Figueiera, S., Hayes, J., Obertelli, G., Schopf, J., Shao, G., Smallen, S., Spring, N., Su, A. and Zagorodnov, D. (2003) Adaptive Computing on the Grid Using AppLeS. IEEE Transactions on Parallel and Distributed Computing, Vol 14, No. 4, April 2003.

Buyya, R., Abramson, D., Giddy, J. (2000) Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. In Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), May 2000, Beijing, China. IEEE Computer Society Press

Buyya, R., Abramson, D. and Venugopal, S. (2005) The Grid Economy, Special Issue on Grid Computing, Proceedings of the IEEE, Parashar, M. and Lee, C. (editors), Volume 93, Issue 3, Pages: 698-714, March 2005.

Chappell, D. (2004). Enterprise Service Bus – Theory in Practice. O'Reilly, 1st Edition, 2004

Czajkowski, K. Foster, I., Kesselman, C., Sander, V. and Tuecke, S. (2002) SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing, Edinburgh, Scotland, July 2002.

Dushay, N., French, J.C and Lagoze, C. (1999) Predicting Indexer Performance in a Distributed Digital Library. In Proceedings of the European Conference on Digital Libraries, Sept. 1999, pp. 142-166.

Foster, I. and Kesselman, C. (2003) The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 2004

Foster I. (2005). Globus Toolkit Version 4: Software for Service-Oriented Systems. In Proceedings of IFIP International Conference on Network and Parallel Computing, Hai, J., Reed, D. and Wenbin, J. (Eds.). Springer-Verlag LNCS 3779, pp 2-13, December 2005.

Fletcher, M., Jackson, T., Jessop, M., Klinger, M., Liang, B. and Austin, J. (2006) The BROADEN Distributed Tool, Service and Data Architecture. In Proceedings of the 2006 All Hands Meeting, Nottingham, UK, September 2006

Faerman, M., Su, A., Wolski, R. and Berman, F. (1999) Adaptive Performance Prediction for Distributed Data-Intensive Applications. In Proceedings of ACM/IEEE SC99 Conference on High Performance Networking and Computing, Portland, Oregon, November 1999.

Haji, M., Gourlay, I., Djemame, K. and Dew, P. (2005) A SNAP-based Community Resource Broker using a Three-Phase Commit Protocol: a Performance Study. Computer Journal, Vol. 48, No. 3, 2005, pp.333-346.

Huedo, E., Montero, R.S., Llorente, I.M. (2006) A Modular Architecture for Interfacing Pre-WS and WS Grid Resource Management. Future Generation Computing Systems, The International Journal of Grid Computing: Theory, Methods and Applications, Vol. 23, No. 2, pp.252-261, 2006.

Kapadia, N.H., Fortes, J.A.B. and Brodley, C.E. (1999) Predictive Application-Performance Modeling in a Computational Grid Environment. In Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC'99), pages 47--54, Los Angeles, CA, August 1999.

Nabrzycki, J., Schopf, J.M., Weglarz, J. (2004) Grid Resource Management: State of the Art and Future Trends. Kluwer Academic Publishers, 2004

Nairac, A., Townsend, N., Carr, R., King, S., Cowley, P. and Tarassenko, L. (1999) A System for the Analysis of Jet Engine Vibration Data. *Integrated Computer-Aided Eng.*, Vol. 6, pp. 53–65, 1999.

Nudd, G.R and Jarvis, S.A. (2004) *Performance-based Middleware for Grid Computing. Concurrency and Computation: Practice and Experience*, 2004

Padgett, J., Djemame, K. and Dew, P. (2006) Predictive Adaptation for Service Level Agreements on the Grid. *International Journal of Simulation*. 7(2), pp. 29-42.

Smith, W., Foster, I. and Taylor, V. (1998) Predicting Application Run Times using Historical Information. In *Job Scheduling Strategies for Parallel Processing*. Feitelson, D. G. and Rudolph, L. (Eds.), *Lecture Notes in Computer Science* 1459, pp. 122--142, Springer Verlag, 1998.

Smith, W. (2004) Improving Resource Selection and Scheduling using Predictions. *Grid Resource Management: State of the Art and Future Trends*. Nabrzyski, Schopf, Weglarz, Kluwer (Eds.), Academic Publishers, pp. 237-253, 2004.

SDSC (2006). San Diego Supercomputer Centre Storage Resource Broker Workshop, February 2006. <http://www.sdsc.edu/srb/Workshop/>

Thain, D., Tannenbaum, T., Livny, M. (2005) *Distributed Computing in Practice: the Condor Experience*. *Concurrency - Practice and Experience*, Vol. 17, No. 2-4, 2005, pp. 323-356

Vraalsen, F, Aydt, R.A., Mendes, C.L. and Read, D.A (2001) Performance Contracts: Predicting and Monitoring Grid Application Behavior. In *Proceedings of the 2nd International Workshop on Grid Computing*, November 2001.