

Grid Service Level Agreements Combining Resource Reservation and Predictive Run-time Adaptation

James Padgett, Karim Djemame and Peter Dew

School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom

Abstract

Resource reservation and performance prediction of execution run-times are key components in delivering timely application services for decision support systems. The provision of Service Level Agreements (SLA) and components to manage tasks such as resource negotiation, monitoring and policing are needed to help meet this requirement. This paper presents an SLA management architecture for use in Grid environments focusing on resource negotiation, monitoring and policing strategies for Grid application services. Included are methods for resource negotiation and generating execution run-time predictions for tracking application progress. Experiments to test the prediction model are presented and compared with a reference execution schedule.

1 Introduction

Grid computing offers scientists and engineering communities access to high performance computational resources and applications. Within these virtual organisations, users and resource providers often belong to different administrative domains. By definition, these resources are heterogeneous with varying quality and reliability. The ability to uphold commitments and assurances on top of the allocated resources is a requirement, sometimes referred to as Quality of Service. A key goal of Grid computing is to deliver management on top of the allocated resources which include for example availability of resources (compute resources, storage etc) and network performance (latency, throughput).

Commitments and assurances are implemented through the use of Service Level Agreements (SLA), which determine the contract between the user and the service provider stating the expectations and obligations that exist between the two. In systems based on timely service provision; to meet commitments and give reliable assurances, there is a need for some form of run-time monitoring and adaptation on top of resource selection and reservation. An additional requirement is the provision of a historical record of the execution plan for auditing.

To support Grid systems based on timely service response an SLA Manager incorporating resource reservation and run-time adaptation is desirable. The SLA Manager offers formal SLA contracts and negotiates for resources with an external resource broker. It monitors resource behaviour and detects when application

performance is likely to fall below a desired level. A system of corrective actions is defined which when implemented try to avert a violation taking place.

SLA management systems found in the literature have focused on specific actions such as negotiation and reservation, or reservation and monitoring; few extend support for run-time adaptation. The approach presented here deals with all these issues, but extends support for run-time adaptation including a system for recording violation warnings and notifications.

The paper presents a SLA management architecture for negotiation, monitoring and policing of SLA's. The SLA Manager negotiates a SLA for the rights to execute a Grid service. Once an agreement exists, management of the SLA involves monitoring which is achieved using performance measurement data obtained from a set of Grid monitoring tools. Policing is also performed using violation data obtained through monitoring of the SLA against real-time performance measurement data. Run-time adaptation is supported thanks to SLA policing mechanisms to enforce changes to the execution to meet SLA guarantees.

The aims of this paper are: firstly, to present a Grid SLA management architecture and show how an SLA can be specified. Secondly, taking an SLA for a compute service as a motivation, discuss how the interaction between the SLA manager and a resource broker is performed in order to guarantee the allocation of compute resources. Thirdly, show how the SLA manager provides rule based adaptation to generate run-time prediction and track application service progression. With this in mind, experiments are designed to test the performance of the prediction model and the rule based adaptation

mechanism when the application service is executing on a large distributed Grid infrastructure, the White Rose Grid (WRG); consisting of high performance computing resources at Leeds, Sheffield and York Universities [1]. An infrastructure such as the WRG exhibits heterogeneous resources and spans over multiple administrative domains.

2 SLA Architecture

The SLA Manager provides SLA and resource management support to virtual organisations such as the Distributed Aircraft Maintenance Environment (DAME) Diagnostic Portal [2]. In this example the SLA Manager is supporting a Grid based decision support system where timely service provision is a requirement. The user gains access to application services with the option of attaching time or performance constraints. The architecture is illustrated in Figure 1 and highlights component responsibilities.

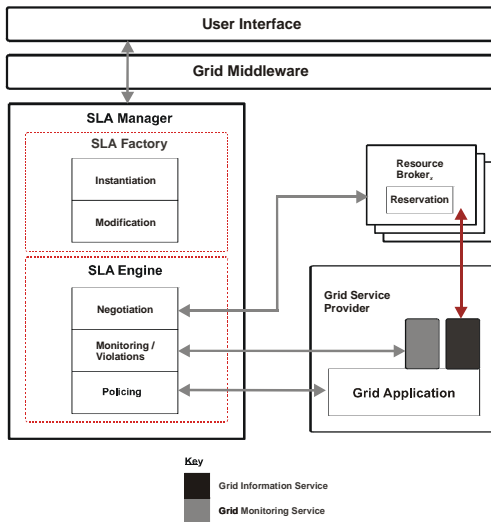


Fig. 1. SLA Management Architecture

2.1. Instantiation and Modification

The SLA Factory provides a template for SLA specification based on user identification of relevant time or performance constraints. Additionally, the factory records provenance data within the SLA during the policing phase. This provides a record of warnings and violations important for auditing after the agreement has terminated. Warnings are recorded when significant events are detected which may affect the SLA in the future but no actual violation has occurred. Violations are

recorded when actual breaches in time or performance constraints have occurred.

2.2. Negotiation and Reservation

In order to fulfil a time or performance constraint the SLA Manager is able to negotiate for resources using a Resource Broker which has responsibility for placing reservations with a resource provider. The architecture allows for a community of brokers providing reservations for resources of different types, e.g. compute, storage and bandwidth. The example used in this research is a SNAP-based resource broker [9], which provides reservations for compute resources. During the negotiation phase a user specifies a Task Service Level Agreement (TSLA), representing an agreement specification for a desired level of performance or time constraint for a Grid application service. Negotiation follows a request() / agree() protocol similar to that specified within the SNAP framework [5]. The SLA Manager enters into an agreement with a Resource Broker which provides a reservation guarantee, a Resource Service Level Agreement (RSLA), with a resource provider. Together the TSLA and RSLA form a Binding Service Level Agreement (BSLA) which binds the task to the potential resource capabilities promised in the RSLA.

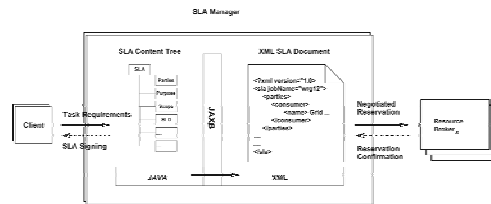


Fig. 2. Resource Reservation Process

Figure 2 illustrates the interaction between user, SLA manager and resource broker during resource negotiation. The task requirements are abstracted in a content tree which is marshalled into an XML document using the JAXB (Java Architecture for XML Binding) API. It is this document that is passed to the resource broker to request a RSLA. If resources are available which match the request an agree() response is sent. The SLA sends a signing request to the client before execution begins.

2.3. Monitoring

To demonstrate the concept of automated monitoring of a Grid infrastructure - the Globus Monitoring and Discovery System (MDS) is used. The SLA engine matches SLO's to

relevant Grid monitoring tools so that validation can be made against performance measurement data.

Alternatively, any Grid Monitoring Service [4] which provides resource and service information using local Grid Monitoring Tools [3] such as Net Logger [8] and the Network Weather Service [28] can be used. The use of a Grid Monitoring Tool (GMT) compliant with the Grid Monitoring Architecture (GMA) [25] would allow a publish / subscribe mechanism to query resource information. Tools such as these enable the SLA Engine to automatically monitor the time and/or performance constraints based on dynamic resource and process information.

2.4. Violations

A Grid infrastructure integrates heterogeneous resources with varying quality and availability which places importance not only on the ability to monitor the state of these resources but reliably predict and capture the violations. A record of warnings and violations is important for auditing once the agreement is in place. Warnings are recorded when significant events are detected which may affect the SLA in the future but no actual violation has occurred. Violations are recorded when it is beyond doubt that breaches in time or performance constraints have occurred.

2.5. Policing

The SLA Engine implements a predictive rule based decision maker. Rule based control strategy offers effective control where the control action does not have a continuous domain. The ability to effect the CPU processing potential is implemented either by migration onto a faster machine or one offering a load average which is less than that of the current resource.

Adaptation has the potential to significantly improve the performance and timely behaviour of an application bound with SLAs. Such an application can change its behaviour depending on available resources, optimising itself to its environment.

3 SLA Specification

The specification should include a form of job submission language [11]; a provenance record; and elements which will satisfy an economic architecture [6]. These provide the SLA with physical, historical and economic elements for describing application requirements.

Job submission elements include a *job description* which identifies the job, user and

resource requirements; plus any data staging requirements. SLO's represent the active guarantees within the SLA and are quantified by a corresponding Service Level Indicator (SLI). Elements describing the parties involved in the agreement may support a listing of users or providers.

Provenance elements support historical usage and economic functions elsewhere in the SLA. Warnings and violations are recorded to allow a historical or usage record to be constructed. These include violation occurrences; prediction and migration decisions; and checkpointing actions.

3.1 Example: Specification for a Compute Service

An example SLA for a Compute service is specified in Table 1. It gives an indication of the components which make up the Instance generated by the Factory.

Table 1. SLA Specification of a compute service

<i>Component</i>	<i>Observation</i>
Purpose	An application service with task requirements
Parties	Consumer, resource broker & resource provider
Scope	Compute service
Service Level Objective (SLO)	Ensure availability of resources satisfying task requirements for the duration of the Grid service task
SLO Attributes	Time and performance constraints
Service Level Indicators (SLI)	For each SLO attribute, its value is a SLI
Exclusions	Adaptation / reservation may not be included
Administration	SLO's met through resource brokering / adaptation

SLO's represent qualitative guarantees describing requirements such as time or performance constraints. Time constraints are expressed as an acceptable period to complete an application execution. Performance constraints are expressed as a desired level of performance in such metrics as CPU load and amount of memory (RAM). The latter comprises a set of SLI parameters which represent the quantitative level of guarantee. The SLI values may take a number of forms such as an upper and lower bound, or may represent a mean value which should be

maintained for the duration of the application execution.

4 Rule Based Adaptation

The SLA manager does not have authoritative control over all applications submitted to the Grid environment, thus has a limited number of control actions over the SLA bound application. The nature of control policy within a compute based Grid environment means the control actions available do not have a continuous domain. In other words the SLA Manager has at present been limited to 4 control actions over the application service: modify Time to Live (TTL), checkpoint, migrate or terminate. If no SLO's are being violated and none are predicted then no control action is taken.

The adaptive process involves a prediction module, a rule based decision maker and a grid monitoring service. The decision maker takes input from two sources, the scheduled remaining execution time, $T_{schedule}$ and the predicted remaining execution time, $T_{remaining}$. The latter is generated in the prediction module using process monitoring data from the Grid monitoring service, the former is taken from the SLA. The prediction model uses the CPU load to generate predictions for the remaining execution time of the application. The rule based decision maker infers a control action using the rule base to determine if any of the SLO's are being violated. The rules are derived from the guarantees specified in the SLA.

There are two control variables which describe the application state during run-time: (1) the scheduled remaining execution time $T_{schedule}$, and (2) the error E which is the difference between $T_{remaining}$ and $T_{schedule}$.

The rules are represented by *If* premise *Then* consequence. The premise represents the current system state and the consequents the control action for the execution. The rules are shown in tabular form below; the body of the table lists the consequents of each rule and the left column and top row represent the premise. As two control variables are being used, each with 4 states, there are 16 possible rules. For example:

If Error is -ve Large and
 execution schedule is < 100%
Then control action is
 migration

This rule quantifies the situation where the predicted remaining execution time $T_{remaining}$ is more than the scheduled remaining execution by a large amount and the application is more than

75% complete. For this situation the rule base indicates that a migration action be taken.

Table 2 The Rule Base

control action		execution schedule, $T_{schedule}$			
		< 25%	< 50%	< 75%	< 100%
error, E	-ve Large	O	T	C	M
	-ve Small	O	O	T	C
	+ve Small	T	O	O	T
	+ve Large	T	T	O	O

Actions

- O = none
- T = modify TTL
- C = checkpoint
- M = migrate

Modifying the TTL is particularly useful when combined with a charging model. It has the potential to optimise costs associated with resource usage and reservation.

5 Prediction Model

The prediction model assumes the user has a reasonable approximation of the application execution time prior to submission. With this assumption the model provides a run-time assessment of the remaining execution time based on the CPU load immediately prior to and during application run-time. Assessments are made as to whether the application will complete within a specified time through comparison with the scheduled remaining execution time. This is agreed upon in the SLA during negotiation. Even when no approximation can be made, monitoring techniques can determine with some accuracy that the execution will take significantly longer than anticipated. In this situation the SLO cannot be expressed as a time constraint and another metric should be chosen. A good choice is a request to maintain a specified percentage of mean CPU usage for the application during run-time.

The fractional CPU usage F_i is measured over n samples at time T_i . $T_{100\%}$ is the time the task would take to execute if 100% CPU usage was maintained throughout application run-time. $F_{estimate}$ is the mean CPU usage over n samples.

$$T_{remaining} = \frac{T_{100\%} - \sum_{i=1}^{n-1} \Delta T_i F_i}{F_{estimate}} \quad (1)$$

$$F_{estimate} = \bar{F}_n = \frac{1}{n} \sum_{i=1}^n F_i \quad (2)$$

$T_{remaining}$ (1) is the predicted remaining execution time if $F_{estimate}$ (2) is maintained for the remainder of the execution. If F_i drops for a large proportion of this period the prediction will be invalid, subsequently, a new prediction is needed.

6 Experiments and Performance Results

Experiments are designed to test the policing phase of an application execution which operates in collaboration with the monitoring phase, after instantiation and negotiation is complete. Negotiation and monitoring/violation capture have been the subject of previous study in [20] and [19] respectively. When an SLA bound application is executing and a competing application is submitted, the resource may no longer be able to meet its RSLA and subsequently the time or performance constraint in the TSLA may be violated.

Once a SLA has been specified, the objective is to test the ability of the prediction model to generate reliable performance predictions for the remaining application execution time. The scenario uses the XTO (eXtract Tracked Orders) Grid application service used within the DAME project [2].

The prediction model assumes that the user knows *a priori* how long the application will take at 100% CPU usage. With this in mind the application service is executed at 100% CPU

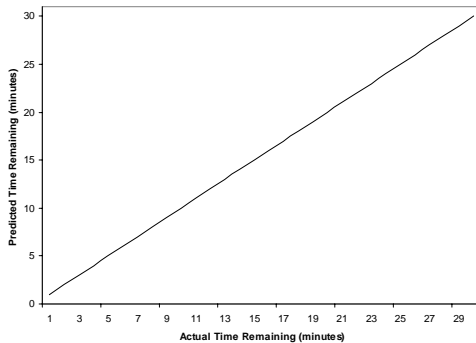


Fig. 3. Predicted time remaining vs. actual time remaining

usage to determine a reference execution time. Once this is known the application service is executed from start to finish. During the course of the execution the predicted remaining execution time is generated periodically using the prediction model and is compared with the reference remaining execution time. This experiment is used to determine the accuracy of (1).

Prior to the experiment the application was run and took 1800 secs to complete with a mean CPU usage of 98%, resulting in a value of 1764 secs for $T_{100\%}$. As a result T_{sla} was set at a reasonable 2100 secs.

The results in Figure 3 show the prediction model provides a good match with the reference remaining execution time suggesting that the model is accurate and can be used for the remaining experiments.

Following this, additional applications are submitted to the same resource to disrupt the SLA bound application. This creates a violation in the time constraint T_{sla} and initiates control actions as defined in the rule base. This is compared to the case where no adaptive technique is used.

The effect of submitting an additional application late in the execution can be seen in Figure 4. After approximately 20 minutes the CPU load decreases producing a knock on effect in the mean CPU load. This is reflected in Figure 5, which shows a decrease in the rate of change of $T_{remaining}$. At approximately 25 minutes $T_{remaining}$ exceeds $T_{schedule}$ and the decision maker sends a checkpoint and migration signal (marked C and M on Figure 5).

The effect of submitting an additional application early in the execution can be seen in Figure 6. After approximately 3 minutes the CPU load decreases, however, the effect on the mean CPU is greater at this stage in the execution. This is reflected in Figure 7 which shows a rapid increase in $T_{remaining}$. After approximately 6 minutes $T_{remaining}$ exceeds $T_{schedule}$ and the decision maker sends a warning signal (marked W on Figure 7). As a result a warning message is added to the SLA. Further into the execution $T_{remaining}$ drops below $T_{schedule}$ and the application completes within the time specified within the SLA.

Where no adaptive technique is used the application execution continues to violate its SLA and fails to deliver on its time constraint. It can be speculated that the predictive adaptation results in a shorter application execution time compared to the case when no adaptation is used.

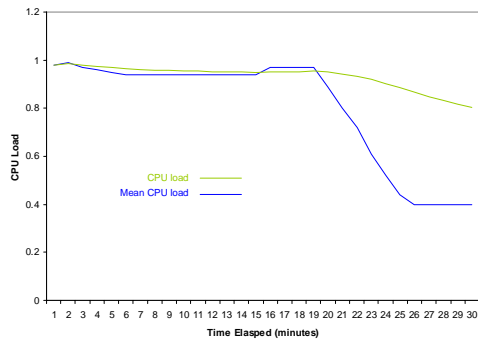


Fig. 4. CPU Load vs. Time – Disturbance added late in execution schedule

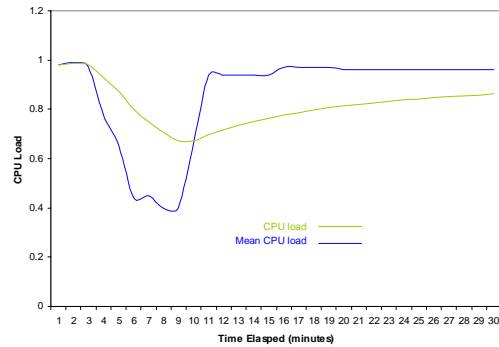


Fig. 6. CPU Load vs. Time – Disturbance added early in execution schedule

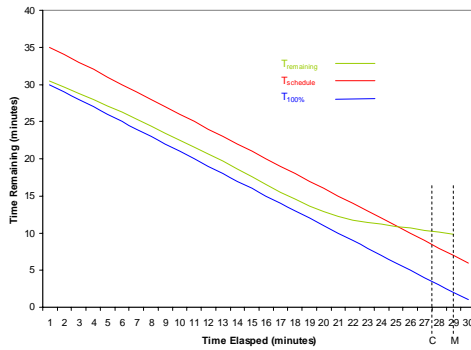


Fig. 5. Time remaining vs. Time elapsed – Disturbance added late in execution schedule

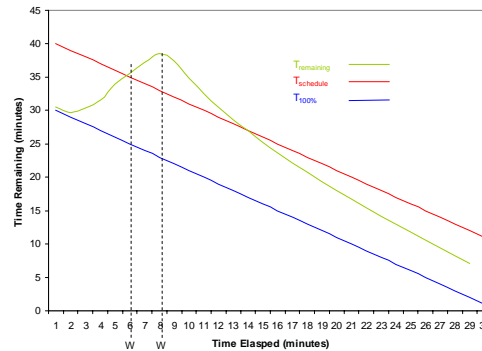


Fig. 7. Time remaining vs. Time elapsed – Disturbance added early in execution schedule

7 Related Work

There have been a number of attempts at defining SLA's and a management architecture for both Web and Grid services. Architectures from Sahai et al [21], Leff et al [12] and Verma et al [27] concentrate on service level agreements within commercial Grids. The service level agreement language used is that presented by Ludwig et al [14]. The Global Grid Forum have defined WS-Agreement [1]; an agreement-based Grid service management specification designed to support Grid service management. Two other important works are automated SLA monitoring for Web services [10] and analysis of service level agreements for Web services [22]. Contract negotiation within distributed systems have been the subject of research where business-to-business (B2B) service guarantees are needed [7]. The mapping of natural language contracts into models suitable for contract automation [16] exist but has not been applied to a Grid environment,

neither has it been applied as a SLA. An approach for formally modelling e-Contracts [15] exists at a higher level than the research by Ludwig et al [14]. Automated negotiation and authorisation systems for the Grid already exist [13] but involve no monitoring or run-time adaptation.

In [24] methods are discussed for predicting execution run-time for parallel applications using historical information. Execution times of similar applications run in the past, are used to estimate the execution time. Application of this method to executions run on a Grid using a local batch queuing system are considered in [23]. This work considers the prediction of execution start times for pending jobs in order to decrease the average job wait time. Although their results indicate that the approach is successful at reducing the average wait time it does not provide information as to the anticipated execution time of the user's job.

Migration of Grid applications using performance prediction is presented in [26]. Migration decisions are based on resource load,

potential performance benefits and time reached in the application. This technique is limited to MPI (Message Passing Interface) based programs but makes use of user specified execution times and tolerance thresholds.

A performance prediction based tool, known as PACE (Performance Analysis and Characterisation Environment) [17] has been developed at the University of Warwick. It uses a combination of source code analysis and hardware modelling to provide an application performance prediction. In [18], the use of PACE in the context of Grid resource scheduling is discussed. The hardware models used in this research are static, which provides the advantage of reusability but does not account for dynamic changes to resource performance.

8 Conclusions and Future Work

Performance prediction of execution run-times is a key component when delivering timely application services in decision support systems. The provision of a system of SLA's and components to manage tasks such as negotiation, monitoring and policing are needed to help meet these requirements. Making simplifying assumptions regarding the task requirements, this work considers a SLA management architecture focussing on a method of application run-time prediction and tracking.

Experiments were conducted in a Grid environment, the White Rose Grid [29]. Submitting an additional application on the candidate resource decreases the capability of the resource to satisfy the RSLA and starves the SLA bound application of processing power. The estimates produced by the prediction model show an increase in the remaining execution run-time. The rule based controller generates a series of warning messages which are recorded by the SLA Factory in the SLA. A violation is recorded resulting in a request to checkpoint and migrate the application service. Where no adaptive technique is used the application execution continues to violate its SLA and fails to deliver on its time constraint.

Future work will centre on a learning based technique to provide an initial prediction for $T_{100\%}$. It will use historical information based on execution run-times from previous runs of the same application service. Users will submit input parameters describing items such as dataset size and run-time flags in order to generate an initial prediction. Additional work comparing application and user level checkpointing will provide a deeper integration

of SLA Manager and SLA bound application service.

Acknowledgements

The work reported in this paper was partly supported by the DAME project under UK Engineering and Physical Sciences Research Council Grant GR/R67668/01. We are also grateful for the support of the DAME partners, including the help of staff at Rolls-Royce, Data Systems & Solutions, Cybula, and the Universities of York, Sheffield and Oxford.

References

- [1] Andrieux, A., K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, *Web Services Agreement Specification (WS-Agreement)*. 2004, Global Grid Forum.
- [2] Austin, J., T. Jackson, M. Fletcher, M. Jessop, P. Cowley, and P. Lobner, *Predictive Maintenance: Distributed Aircraft Engine Diagnostics*, in *The Grid 2: Blueprint for a new computing infrastructure*, I. Foster and C. Kesselman, Editors. 2004, Morgan Kaufmann: Elsevier Science: Amsterdam; Oxford.
- [3] Balaton, Z., P. Kacsuk, N. Podhorszki, and F. Vajda, *Comparison of Representative Grid Monitoring Tools*. 2000, Laboratory of Parallel and Distributed Systems, Hungarian Academy of Sciences, Budapest, Hungary.
- [4] Czajkowski, K., S. Fitzgerald, I. Foster, and C. Kesselman. *Grid Information Services for Distributed Resource Sharing*. in *High performance distributed computing*. 2001. San Francisco, CA: IEEE Computer Society Press.
- [5] Czajkowski, K., I. Foster, C. Kesselman, V. Sander, and S. Tuecke. *SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems*. in *Job scheduling strategies for parallel processing*. 2002. Edinburgh: Berlin.
- [6] GESA, *Grid Economic Service Architecture Working Group*, Global Grid Forum: <http://www.doc.ic.ac.uk/~sjn5/GGF/gesa-wg.html>.
- [7] Goodchild, A., C. Herring, and Z. Milodevic, *Business contracts for B2B*. 2000, Distributed Systems Technology Center (DSTC), Australia: Queensland, Australia.
- [8] Gunter, D., B. Tierney, B. Crowley, M. Holding, and J. Lee. *NetLogger: A Toolkit for Distributed System Performance Analysis*. in *Modeling, analysis and simulation of computer and*

- telecommunication systems*. 2000. San Francisco, CA: IEEE Computer Society.
- [9] Haji, M., I. Gourlay, K. Djemame, and P. Dew, *A SNAP-based Community Resource Broker using a Three-Phase Commit Protocol: a Performance Study*. Computer Journal, 2005. **48**(3): p. 333-346.
- [10] Jin, L., V. Machiraju, and A. Sahai, *Analysis on Service Level Agreement of Web Services*. 2002, HP Laboratories: Palo-Alto, CA.
- [11] JSDL, *Job Submission Description Language Working Group*, Global Grid Forum: <http://www.epcc.ed.ac.uk/~ali/WORK/GG/F/JSDL-WG/>.
- [12] Leff, A., J.T. Rayfield, and D.M. Dias, *Service-Level Agreements and Commercial Grids*. IEEE Internet Computing, 2003. **7**(4): p. 44-50.
- [13] Lock, R. *Automated contract negotiations for the grid*. in *Postgraduate Research Conference in Electronics, Photonics, Communications & Networks, and Computing Science*. 2004. University of Hertfordshire, UK: EPSRC.
- [14] Ludwig, H., A. Keller, A. Dan, R. King, and R. Franck, *A Service Level Agreement Language for Dynamic Electronic Services*. Electronic Commerce Research, 2003. **3**(1/2): p. 43-59.
- [15] Marjanovic, O. and Z. Milosevic. *Towards Formal Modeling of e-Contracts*. in *Enterprise distributed object computing*. 2001. Seattle, WA: IEEE Computer Society.
- [16] Milosevic, Z. and R.G. Dromey. *On Expressing and Monitoring Behaviour in Contracts*. in *Enterprise distributed object computing*. 2002. Lausanne, Switzerland: IEEE Computer Society.
- [17] Nudd, G.R., C. Junwei, D.J. Kerbyson, and E. Papaefstathiou. *Performance modeling of parallel and distributed computing using PACE*. in *19th IEEE International performance, computing and communications conference*. 2000. Phoenix, USA: IEEE Computer Society.
- [18] Nudd, G.R., H.N.L.C. Keung, J.R.D. Dyson, and S.A. Jarvis. *Self-adaptive and self-optimising resource monitoring for dynamic grid environments*. in *15th International workshop on database and expert systems applications*. 2004. Zaragoza, Spain: Institut Für Anwendungsorientierte Wissensverarbeitung.
- [19] Padgett, J., K. Djemame, and P. Dew, *Grid-based SLA Management*. Lecture Notes in Computer Science, 2005(3483): p. 1282-1291.
- [20] Padgett, J., M. Haji, and K. Djemame, *SLA Management in a Service Oriented Architecture*. Lecture Notes in Computer Science, 2005(3470): p. 1076-1085.
- [21] Sahai, A., A. Graupner, V. Machiraju, and A. van Moorsel. *Specifying and Monitoring Guarantees in Commercial Grids through SLA*. in *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*. 2003. Tokyo: IEEE Computer Society.
- [22] Sahai, A., V. Machiraju, M. Sayal, L. Jin, and F. Casati, *Automated SLA Monitoring for Web Services*. 2002, HP Laboratories: Palo-Alto, CA.
- [23] Smith, W., *Improving Resource Selection and Scheduling using Predictions*, in *Grid Resource Management: State of the Art and Future Trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2003, Kluwer Academic Publishers.
- [24] Smith, W., I. Foster, and V. Taylor, *Predicting Application Run Times Using Historical Information*. Lecture Notes in Computer Science, 1998(1459): p. 122-142.
- [25] Tierney, B., R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski, *A Grid Monitoring Architecture*. 2002, Global Grid Forum.
- [26] Vadhiyar, S. and J. Dongarra. *A Performance Oriented Migration Framework for The Grid*. in *Cluster computing and the grid; CCGrid 2003*. 2003. Tokyo: IEEE Computer Society.
- [27] Verma, D., M. Beigi, and R. Jennings, *Policy Based SLA Management in Enterprise Networks*, in *Policies for Distributed Systems and Networks*, M. Sloman, J. Lobo, and E.C. Lupu, Editors. 2001, Springer-Verlag. p. 137-152.
- [28] Wolski, R., N.T. Spring, and J. Hayes, *The network weather service: a distributed resource performance forecasting service for metacomputing*. Future Generation Computer Systems, 1999. **15**(5-6): p. 757-768.
- [29] WRG, *The White Rose Grid*, White Rose Consortium: <http://www.wrgrid.org.uk/>.