Grid-Based SLA Management

James Padgett, Karim Djemame, and Peter Dew

Informatics Institute, School of Computing, University of Leeds,
Leeds LS2 9JT, United Kingdom
{jamesp, karim}@comp.leeds.ac.uk

Abstract. This paper presents an architecture for specifying, monitoring and validating Service Level Agreements (SLA) for use in Grid environments. SLAs are an essential component in building Grid systems where commitments and assurances are specified, implemented and monitored. Targeting compute resources, an SLA manager reserves resources for user applications requiring resources on demand. Methods for automated monitoring and violation capture are discussed showing how Service Level Objectives (SLO) can be validated. A SLA for a compute service is specified and experiments carried out on the White Rose Grid. Results are presented in the form of a SLA document and show the violations that were captured during task execution.

1 Introduction

Grids [1] offer scientists and engineering communities high performance computational resources supporting virtual organisations. In Grid environments, users and resource providers often belong to multiple management domains. Users need commitments and assurances on top of the allocated resources (this is sometimes referred to as Quality of Service), and it is the resource providers responsibility to deal with erroneous conditions, fail over policies etc. A key goal of Grid computing is to deliver the commitments and assurances on top of the allocated resources which include for example availability of resources (compute resources, storage etc), security and network performance (latency, throughput) [2].

Commitments and assurances are implemented through the use of Service Level Agreements (SLA), which determine a *contract* between user and Grid Service provider. A SLA is defined as an explicit statement of the expectations and obligations that exist in a business relationship between the user and the Grid Service provider. A formalised representation of commitments in the form of a SLA document is required, if information collection and SLA evaluation are to be automated. At any given point in time many SLAs may exist, and each SLA in turn may have numerous objectives to be fulfilled.

In the context of a Grid application, consolidation of management information is required when resources are spread across geographically distributed domains. SLAs may be distributed, and their validation depends on local measurements. With this is mind, the paper presents an SLA management architecture with automated SLA negotiation, monitoring and policing mechanisms. The current Open Grid Services Architecture (OGSA) [3] specification defines SLA management as a high level service supporting SLAs within the Grid. Thus, a Grid user accessing Grid services on demand and with quality of service (QoS) agreements enabling commitments to be fulfilled is a primary requirement. The user can specify a SLA which will guarantee resources, provide job monitoring and record violations if they are detected. The SLA document records agreement provenance allowing for auditing mechanisms after it has terminated.

The structure of the paper is as follows: in section 2, the Grid SLA management architecture is presented. In section 3 an SLA specification for a compute service is outlined. Section 4 details the interaction between the SLA manager and a resource broker. In section 5 the monitoring engine and violation capture mechanism are demontrated. Section 6 presents some experiments involving a performance evaluation of the SLA Manager, and discusses the experimental results obtained on the White Rose Grid [4]. Related work is described in section 7, followed by a conclusion and discussion on future work.

2 Service Level Agreement Management Architecture

The SLA Management Architecture (Figure 1) defines an SLA Manager that runs in parallel with Grid Services to provide SLA management in environments such as the DAME Diagnostic Portal [5]. A client uses portal access to invoke Grid services which have time or performance requirements for their execution.

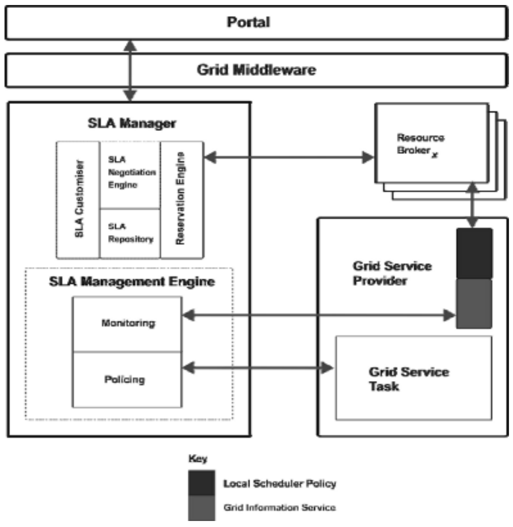


Fig. 1. SLA Management Architecture

2.1 SLA Management Interaction

Once the SLA Manager has instantiated an agreement and the Grid Service task is executing, interactions between the SLA Manager and the Grid Service task are maintained throughout the agreement life cycle to enforce the SLA guarantees.

2.2 Automated SLA Negotiation

The SLA Manager provides SLA negotiation using the Service Negotiation and Acquisition Protocol (SNAP) [6]. It provides a protocol for the negotiation and reservation of resources in order to guarantee the SLA based on a set of task requirements. These requirements are formally captured in a number of Service Level Objectives (SLO). The type of reservations made can be resource based or service based and will be executed through the Reservation Engine. The first iteration of which will implement performance-based Service Level Objectives. Further development will see the functionality enhanced. Once an agreement has been reached the Service Level Agreement is formalised into an SLA document. The SLA is offered to all parties for signing before being stored in the SLA repository where it can be accessed whenever validation is needed. The SLA Customiser has the ability to change the SLA document after the agreement has been signed; for changes in state or recording violations.

2.3 Automated SLA Monitoring

Once a SLA has been agreed and the resources have been reserved, the Grid task can begin execution. The SLA management engine is tasked with automated monitoring of the metrics needed to enforce the guarantees in the Service Level Objectives. It uses an external Grid Monitoring Service [7] to select the Grid monitoring tools which will be needed to monitor the SLA, such as Net Logger [8] and the Network Weather Service [9]. Tools such as these enable the SLA management engine to automatically monitor the SLOs based on dynamic resource information.

2.4 Automated SLA Policing

The SLA policing engine will adapt a task execution using a number of adaptation outcomes, examples of which include modifying the TTL (time to live), resource re-negotiation, migration and termination. The adaptation feedback control mechanism will determine which of these outcomes is appropriate for a given situation. It will do this either in response to or to prevent violation of a SLO. The method used to adapt the Grid task is recorded in the SLA and based on the policies of the Grid resource. Adaptation has the potential to significantly improve the performance of applications bound with SLAs. An adaptive application can change its behaviour depending on available resources, optimising itself to its dynamic environment. For example, when resource load changes, a Grid system could seek to improve the quality of its compute resources or re-locate to another compute resource. To support adaptation feedback control, mechanisms for decision making are to be implemented using Fuzzy Control [10], however mathematical modelling, knowledge-based heuristics and reflection could be used.

3 SLA Specification

The SLA Document is XML based, whereas the SLA is represented internally by a content tree made up of Java objects. The SLA Manager supports a number of service guarantees through differentiated classes of service; best effort, best effort with adaptation, reservation, reservation with adaptation.

3.1 Example: Specification for a Compute Service

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

Table 1. SLA Specification of a compute service

Component	*Observation*
Purpose	A Grid job guaranteeing task requirements
Parties	Consumer, resource broker & compute resource
Scope	Compute service
Service Level Objective (SLO)	Ensure availability of resources satisfying task requirements for the duration of the Grid service task
SLO Attributes	CPU count, type, speed, usage. OS and OS version
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

The SLO's represent a qualitative guarantee such as CPU, RAM or HDD SLA. They comprise a set of SLI parameters which represent quantitative data describing the level of the SLO guarantee, such as CPU_COUNT or RAM_COUNT. The SLI values may take a number of forms, two which will be used are (1) a parameter distribution where the value of the SLI must be in a range or (2) a list where the parameter must equal a definite value

3.2 SLA Negotiation

The task requirements are based on the natural language definition given in Table 1. They are represented internally as a SLA content tree based and governed by a schema document. The negotiation engine parses the SLA content tree into an unsigned SLA document. This is passed to a resource broker which attempts to match the resource requirements by gathering information about the available Grid resources. A reservation decision is made based on the permissions and the suitability of the resource given the requirements. The SLA is signed by the SLA manager on behalf of the provider after which it is offered to the user for signing.

4 Resource Reservation

The SLA manager automates the SLA life cycle from negotiation to termination. An SLA document is specified in XML. The heterogeneous nature of XML makes it suitable for the SLA document in a service oriented architecture allowing the document to traverse the Grid along with the task execution. The SLA manager can be executed on all Grid resources as a high level Grid Service within any OGSA compliant middleware. It parses SLA documents and configures the SLA Management system for the task execution. Once parsed the SLA manager un-marshalls the SLA document into objects which represent the content and organisation of the SLA. The architecture allows an SLA document to be updated as the Grid execution is progressing, allowing for violations to be captured and recorded as they are detected.

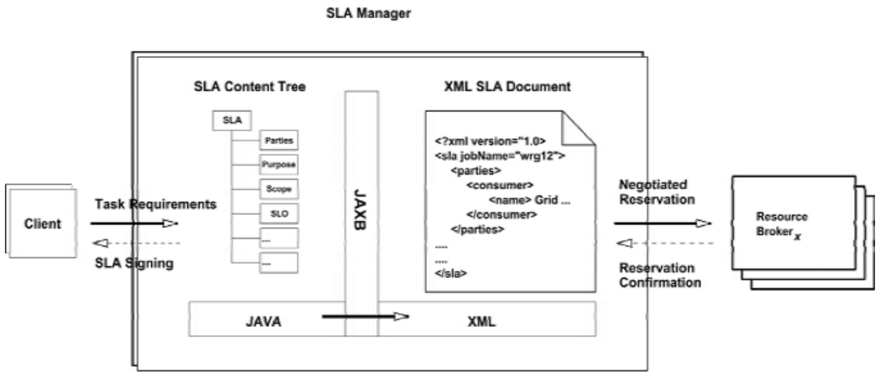


Fig. 2. Compute Service SLA Specification and Interation with Resource Broker

The SLA Manager's negotiation engine is able to interface with a resource broker providing reservations for different types of Grid resources as shown in Figure 2. It uses a resource broker to provision resources needed to fulfil the SLA. The example used in this work is the SNAP-based resource broker [11] which provides reservations for compute resources. This is similar to the Task Service Level Agreement (TSLA) defined within the SNAP framework which represents an agreement that specifies the desired performance of the task [6] where a task represents a job submission, execution or Grid service invocation. The SLA manager enters into an agreement with the resource broker service which provides a reservation guarantee, but it is the individual brokers which form the reservation agreements with the local Grid resources.

5 Automated Monitoring

5.1 Grid Measurement Tool

A Grid Information Service [12] (GIS) provides resource and service information from local Grid Monitoring Tools [13] that interface with low level Information Service Providers. The ability to use a number of low level Information Service Providers means that the Service can be used as a homogeneous interface to a number of Grid Monitoring Tools such as those mentioned in section 2.3. The use of a Grid Monitoring Tool (GMT) compliant with the Grid Monitoring Architecture (GMA) [7] allows a publish / subscribe mechanism to query resource information. To demonstrate the concept of automated monitoring of SLA's within a Grid - the Globus Monitoring and Discovery System (MDS) is used.

5.2 Monitoring Engine

The monitoring engine automates the SLA monitoring process. It matches SLO's to relevant Grid monitoring tools so that validation can be made against performance measurement data. A Grid integrates heterogeneous resources with varying quality and availability, this places importance on the ability to dynamically monitor the state of these resources and deploy reliable violation capture mechanisms.

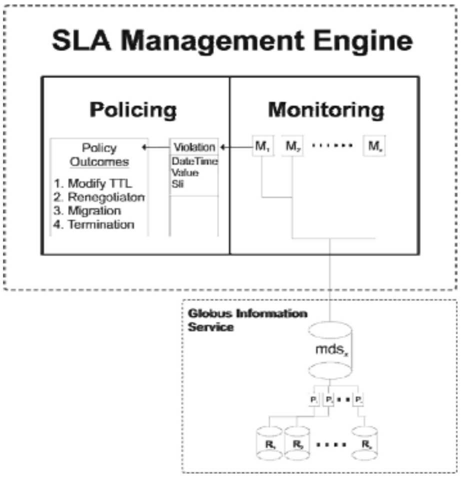


Fig. 3. Monitoring and Policing Mechanism

The monitoring engine launches a number of monitoring threads [M_x] corresponding to a SLI within each SLO (Figure 3). The threads periodically compare the measured value [m_x] taken from the Grid Information Service [12] [mds_x] with the value listed in the SLI, [sli_x]. When a violation is detected a violation object is created and passed to the policing engine.

5.3 Capturing Violations

The monitoringThreads compare the observed value [m_x] with that expressed in the SLA, [sli_x]. If a violation occurs a dateTime stamp is added to the SLO_x and the violated value of [sli_x] is recorded. The violation is represented as a violation object. It is passed to the policing engine and the SLA Customiser so that the violation can be recorded within the SLA document.

6 Experiments

6.1 Overview

The SLA Manager is expected to provide a managed Grid execution with guarantees over best effort execution through resource reservation, automated monitoring and adaptation. Experiments are conducted as a proof of concept test of the SLA Manager in a production Grid environment; the White Rose Grid [4]. Once a SLA has been specified, the objectives are to: (1) Show the SLA manager can specify a SLA for a Grid compute service; (2) Test the effectiveness of the monitoring engine and the Grid monitoring tool; and (3) Determine if the violation capture mechanism is reliable and can keep pace with the monitoring engine.

The White Rose Grid was used, consisting of resources at the University of Leeds, Sheffield and York. The resources are configured with Solaris OS 5.8, Globus 3.0 middleware and MDS2.2 Grid information system. Communication occurs over a fast ATM network, called YHMAN. Network bandwidth and latency are not guaranteed within the SLA.

6.2 Experimental Design

Experimental Scenario

A user wants to execute a DAME Grid compute service task called, eXtract Tracked Orders (XTO) [14] with task requirements which they want formalising in a SLA. They would like guarantees on service performance to guarantee a time for completion of the task. The SLA Manager will take these requirements, specify a SLA, launch the Grid service, monitor its execution and record any violations. Two SLO's are specified: (1) the amount of RAM with an SLI of 3100MB; and (2) the free CPU_LOAD with a SLI of 0.25. The XTO Grid service is executed and monitored for a period of 130 seconds. Both SLO's are perturbed 80 seconds into the execution by two additional processes which are CPU and RAM intensive. The monitoring engine will launch two monitoring thread entries within the MDS: Mds-Memory-Ram-Total-freeMB and Mds-Cpu-Total-Free-5minX100.

6.3 Performance Results and Discussion

The experiments involve the submission and monitoring of the XTO Grid Service given a set of task requirements. Those requirements are to guarantee a minimum amount of free memory and CPU load on the compute resource. The monitoring threads sample MDS entries Mds-Memory-Ram-Total-freeMB and Mds-Cpu-Total-Free-5minX100 at each time interval from start to finish. Figure 4 shows the amount of free RAM and CPU_LOAD respectively during the task execution.

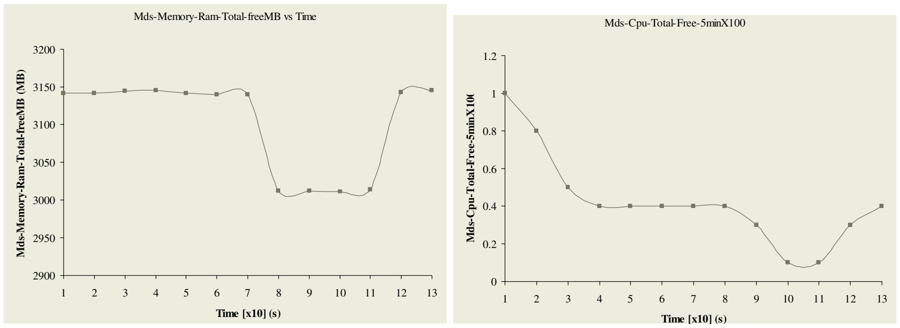
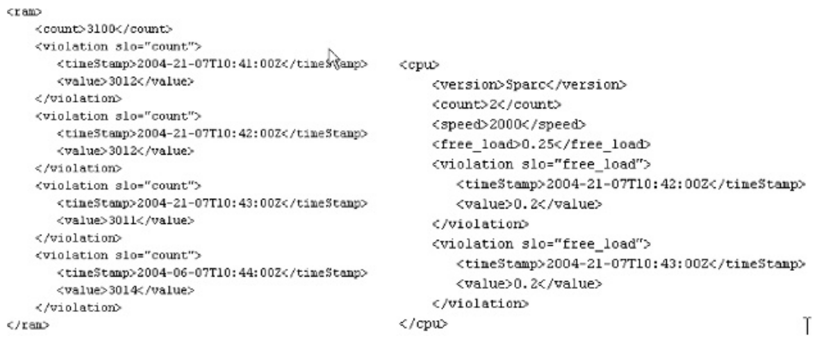


Fig. 4. Graph showing disturbance of free memory on a White Rose Grid resource

Figure 5 is the SLA after task completion. The monitoring engine successfully captures a timeStamp and value for each violation.



```
<ram>
   <count>3100</count>
   <violation slo="count">
      <timeStamp>2004-21-07T10:41:00Z</timeStamp>
      <value>3012</value>
   </violation>
   <violation slo="count">
      <timeStamp>2004-21-07T10:42:00Z</timeStamp>
      <value>3012</value>
   </violation>
   <violation slo="count">
      <timeStamp>2004-21-07T10:43:00Z</timeStamp>
      <value>3011</value>
   </violation>
   <violation slo="count">
      <timeStamp>2004-06-07T10:44:00Z</timeStamp>
      <value>3014</value>
   </violation>
</ram>
```

```
<cpu>
   <version>Sparc</version>
   <count>2</count>
   <speed>2000</speed>
   <free_load>0.25</free_load>
   <violation slo="free_load">
      <timeStamp>2004-21-07T10:42:00Z</timeStamp>
      <value>0.2</value>
   </violation>
   <violation slo="free_load">
      <timeStamp>2004-21-07T10:43:00Z</timeStamp>
      <value>0.2</value>
   </violation>
</cpu>
```

Fig. 5. SLA documents showing violations to RAM_COUNT and CPU_LOAD

7 Related Work

There have been a number of attempts at defining an SLA management architecture for both Web and Grid services. Architectures from Hewlett-Packard Laboratories [15] and IBM T.J. Watson Research Center [16] concentrate on service level agreements within commercial Grids. The service level agreement language used is that presented by Ludwig et al in [17]. The Global Grid Forum have defined WS-Agreement [18]; an agreement-based Grid service management specification designed to support Grid service management. Two other important works are automated SLA monitoring for Web services [19] and analysis of service level agreements for Web services [20]. Contract negotiation within distributed systems have been the subject of research where business-to-business (B2B) service guarantees are needed [21]. The mapping of natural language contracts into models suitable for contract automation [22] exist but has not been applied to Grid environments, nor has it been applied as a SLA. An approach for formally modelling e-Contracts [23] exists at a higher level than the research in [17]. Automated negotiation and authorisation systems for the Grid already exist [24] but involve no monitoring or policing ability post agreement.

8 Conclusion and Future Work

A formalised representation of commitments in the form of a SLA is required between users and service providers in order for QoS to be implemented in a Grid environment. SLA management is important in formalising QoS implementations within Grid services. Making simplifying assumptions regarding the task requirements, this work considers a SLA for a compute service, a mechanism for reserving resources and an automated monitoring / violation capture mechanism. The experiments were conducted in a Grid environment, the White Rose Grid. They show the SLA Manager is capable of negotiating and monitoring an SLA for a Grid compute service given a set of task requirements. The monitoring engine / violation capture mechanism is effective at detecting and recording violations within the SLA.

The SLA Management architecture has provision for three types of SLA guarantees: performance, usage and reliability. The work presented here is targeting service performance; service usage and reliability will be the subject of further research.

Future work will centre on the implementation of a fuzzy controller for adaptation feedback control. It provides a methodology for implementing a more human heuristic control system by capturing qualitative control experience and applying it to the control algorithm. It is employed here in preference to a classic control method as they are inappropriate for Grid environments due to their heterogeneous nature. The use of a Fuzzy control technique is especially suited to this problem – where producing an accurate mathematic model for use in conventional control would be difficult.

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] Berman, F., *Grid computing : making the global infrastructure a reality*. Chichester: Wiley. 2003.

[2] Foster, I., C. Kesselman, and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 2001. **15**: p. 200-222.

[3] Foster, I., et al., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, The Globus Project, June 22.

[4] *The White Rose Grid* [Online], White Rose Consortium, Available from World Wide Web: http://www.wrgrid.org.uk/

[5] Foster, I. and C. Kesselman, *The Grid 2 : blueprint for a new computing infrastructure*. Amsterdam ; Oxford: Morgan Kaufmann : Elsevier Science. 2004.

[6] Czajkowski, K., et al. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. IN: *Job scheduling strategies for parallel processing*. D.G. Feitelson, L. Rudolph, and U. Schwiegelshohn. 2002. Edinburgh: Berlin.

[7] Tierney, B., et al., *A Grid Monitoring Architecture*, Global Grid Forum, August 27.

[8] Gunter, D., et al. 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] Wolski, R., N.T. Spring, and J. Hayes, The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generations Computer Systems*, 1999. **15**(5-6): p. 757-768.

[10] Passino, K. and S. Yurkovich, *Fuzzy control*. Menlo Park, Calif. ; Harlow: Addison-Wesley. 1998.

[11] Haji, M., et al. A SNAP-based Community Resource Broker using a Three-Phase Commit Protocol. IN: *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium*. 2004. Santa Fe, USA.

[12] Schopf, J.M., A General Architecture for Scheduling on the Grid. *Journal of Parallel and Distributed Computing*, 2002(Special Issue on Grid Computing).

[13] Balaton, Z., et al., *Comparison of Representative Grid Monitoring Tools*, Laboratory of Parallel and Distributed Systems, Hungarian Academy of Sciences, Budapest, Hungary, March 2000.

[14] Nadeem, S., P. Dew, and K. Djemame, *XTO Grid Services on the White Rose Grid: Experiences in building an OGSA Grid Application*, DAME Technical Report, Informatics Institute, University of Leeds, UK,

[15] Sahai, A., et al. Specifying and Monitoring Guarentees in Commercial Grids through SLA. IN: *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*. 2003. Tokyo: IEEE Computer Society.

[16] Leff, A., J.T. Rayfield, and D.M. Dias, Service-Level Agreements and Commercial Grids. *IEEE Internet Computing*, 2003. **7**(4): p. 44-50.

[17] Ludwig, H., et al. A Service Level Agreement Language for Dynamic Electronic Services. IN: *Advanced issues of E-commerce and web-based information systems*. 2002. Newport Beach, CA: IEEE Computer Society.

[18] Andrieux, A., et al., *Web Services Agreement Specification (WS-Agreement)*, Global Grid Forum, July 26.

[19] Jin, L., V. Machiraju, and A. Sahai, *Analysis on Service Level Agreement of Web Services*, HPL-2002-180, HP Laboratories,

[20] Sahai, A., et al., *Automated SLA Monitoring for Web Services*, HPL-2002-191, HP Labs,

[21] Goodchild, A., C. Herring, and Z. Milodevic, *Business contracts for B2B*, Distributed Systems Technology Center (DSTC), Austrailia, 2000.

[22] Milosevic, Z. and R.G. Dromey. On Expressing and Monitoring Behaviour in Contracts. IN: *Enterprise distributed object computing*. 2002. Lausanne, Switzerland: IEEE Computer Society.

[23] [Marjanovic, O. and Z. Milosevic. Towards Formal Modeling of e-Contracts. IN: *Enterprise distributed object computing*. 2001. Seattle, WA: IEEE Computer Society.

[24] 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