Grid-Based Business Partnerships Using Service Level Agreements

Mike Boniface[1], Stephen Phillips[1], Mike Surridge[1]
[1] University of Southampton IT Innovation Centre

Abstract

The EU IST SIMDAT project is developing generic Grid technology targeted at business users from several representative industry sectors. We have developed an SLA (Service Level Agreement) Management service for GRIA middleware that allows service providers and customers to trade resources (applications, data, processing, storage) under the terms of bilateral SLAs. In this paper, we describe the SLA architecture, QoS model and how the service has been deployed to support Grid-based partnerships in the aerospace, automotive and pharmaceutical sectors.

1 Introduction

The SIMDAT project [1] is developing and deploying Grid technology to support industrial business partnerships within key economic sectors; aerospace, automotive, meteorology and pharmaceutical. An important characteristic of Grid technology is its ability to support inter-domain business partnerships allowing organisations share a wide range of business assets within the constraints of commercial polices for security and protection of IPR. Traditional academic Grids are based on collaborative resource sharing usually organised by service providers, who agree to share their resources to create a single 'resource pool' that can meet the needs of a common user community. However, this involves service providers and users signing up to a 'virtual organisation' with a common objective (that of the relevant user community), and agreeing unified policies for managing and granting user access to the shared resources. This arrangement is very expensive to establish and operate, does not meet the business needs of commercial service providers or consumers, and leaves no room for participants to compete openly on price or quality of service. We have developed an SLA (Service Level Agreement) Management service for GRIA [2] that allows service providers and customers to trade resources (applications, data, processing, storage) under the terms of bilateral SLAs allowing service providers to operate independently of each other, and compete as necessary to provide services to paying customers. Customers are able to control which services they consume, how much they are used, and by whom.

2 Background

The original GRIA project set out to provide business models for using the Grid, based on earlier work on the agent-based DISTAL framework [3]. In DISTAL, users were obliged to contact and negotiate with agents representing computing resource owners, specifying their requirements in terms of machine specifications and usage time. The resource owner's agent then responded with an "offer" indicating their machine specification and available usage time. The user could gather offers from several resource owners, select the best offer, and then submit their computations to the resource owner. DISTAL used a proprietary schema within a top-level FIPA agent protocol [4]. The GRIA 4 business model evolved from DISTAL to use a modified quality of service (QoS) model, in which a single "offer" can cover multiple computational "jobs" spread over a relatively long time interval. This approach requires far fewer message exchanges to negotiate QoS for composite applications such as video rendering or parametric optimisation. However, it also means it is not feasible to specify absolute job submission and completion times, because neither side can guarantee the timings for

collections of inter-dependent jobs. A simpler QoS model is therefore required, based only on the total usage across all jobs and data transfers over a specified interval. GRIA 4 provided a QoS service, separate from the job execution and data storage services, allowing user actions to be managed. Service providers respond to consumer QoS requests with offers based on predicted resource availability, allowing the user to detect which providers can meet their QoS needs, and accept offers only where capacity is sufficient for their needs. Users have to ask the QoS service to set up a job or data store, specifying the resource requirements, allowing the QoS service to check if these are within the agreed total before giving the user access to the job and data services themselves. The job and data services keep track of the resources used per job and data store, and terminate them when the declared usage limit is reached.

GRIA 4 was relatively easy to use, and has been extremely successful in the SIMDAT project and as a test platform in the NextGRID [5] and SFW projects [6]. However, it had some weaknesses that can be traced to the specific QoS model and management mechanisms. Users find it hard to predict the QoS requirements for each job and data store. They often overstate the true requirements to avoid premature termination by the service, but this in turn means they have to overstate their needs when asking for QoS offers. Service providers have to assume that QoS requests are realistic, and often decide that they cannot meet (user-inflated) QoS requests, so that no services are provided. Also, users often want to reuse data for new calculations in ways that could not easily be predicted in advance. Thus they frequently find they want to use resources beyond the time period originally envisaged, but are prevented by the QoS model from doing so. These drawbacks often cause service providers to refuse user requests, even when the resources available are enough to meet the user's true requirements, which in a commercial situation would produce lower resource utilisation and higher prices. The refusals often come when users don't expect it, making GRIA 4 services less usable to consumers. In SIMDAT, a new SLA management service was required for exploitation of Grid infrastructure by industry that addresses some of the imperfections in the original QoS model. The requirements include:

i) application service developers should be able to define their own qualitative QoS criteria in terms of their operations, including error and failure criteria;
ii) metrics should provide simple mechanisms to define quantitative QoS criteria and limits, and also resource usage and limits;
iii) metrics should include instantaneous rate measurements at some absolute time, and total usage over a defined period between two absolute times;
iv) metrics should provide a simple aggregation model for both QoS measurements and QoS limits, allowing SLA to cover multiple application services.

The need for absolute times is because we must capture the temporal relationships between QoS measurements and their associated SLA limits, including different application services under the same SLA, and the relationship between multiple SLA and overall resource capacity.

3 Related Work

Current Grid technology focuses middleware to support service provision of a wide range of IT services. Web service technology is being increasingly adopted, and aspects such as security and quality of service are the focus of many developments. Within the Grid community, infrastructure technologies that support business partnership negotiation are relatively immature, with most current Grid systems using some sort of priority queue system, based around a low level job scheduler. Various OGF working groups are defining schema related to QoS models and metrics. The GGF Usage Record (UR) Working Group has defined a usage accounting record schema, designed to allow Grid sites to exchange usage data in a mutually understood format [7]. The core schema currently supports computational job accounting only with extensions for properties from individual sites or groups of sites. The GGF Job Submission Description Language (JSDL) Working Group has published a specification for describing the requirements of computational jobs for submission to resources, including but not limited to Grid environments [8]. Where the GGF Usage Record focuses on records of job resource usage during and primarily after execution, JSDL focuses on specifying resource

requirements (i.e. upper limits) at submission time. The UR and JDSL specifications are fairly comprehensive in their coverage, and provide a useful basis for describing computational resource requirements and usage. However, they are both highly specific to low-level job execution scenarios, and so cannot be used in a business-oriented QoS model for describing more generic consumer values.

WSDM [9] is an approved OASIS standard covering management of web services. WSDM MUWS Part 2 (Management Using Web Services) specification defines manageability capabilities for OperationStatus, State, Configuration and Metrics. The last of these includes a schema for associated attributes describing the temporal nature of the underlying measurements (e.g. measurements over an time interval, at a single instant, since the last reset, etc), how the result can change (e.g. counters versus gauges, etc), when they are updated (regular or irregular intervals, on demand, etc), and provides a way to associate metrics with metric groups. The WSDM metrics are quite generic, and at first glance appear useful for describing QoS in a very general way. However, it turns out that they describe only instantaneous values, even those that are collected over an interval. The "interval" is defined as an xsd:duration type, which is an amount of time (e.g. 1 hour) not an absolute interval (e.g. between 1605 and 1705 UTC on Tuesday 28 Feb 2006). The time-scope properties the metrics describe how a value is measured (e.g. a time average) and not its temporal relationship to other measurements or events. The WSDM MOWS (Management of Web Services) specification describes how MUWS can be applied to the management of web services via a MUWS-type management web service. MOWS defines metrics such as NumberOfRequests, NumberOfFailedRequests, NumberOfSuccessfulRequests, ServiceTime, MaxResponseTime, LastResponseTime. Clearly, these properties may be useful for some purposes, but they don't cover resource usage or limits, and they provide only a crude description of consumer-centric QoS.

4 QoS Model

4.1 Overview

The SLA Management service handles a general set of "metrics" describing service quality, delivery and resource usage. The SLA manager uses three sets of information based on these metrics:

- service and resource usage reports from application services, expressed as a set of values for named terms in the vector $Z \cup Y$;
- quality of service criteria from each SLA, expressed as a set of value constraints on the total value across all associated activities for named terms in the service usage vector Z;
- service provider capacity criteria, expressed as a set of value constraints on the total value across all activities for named terms in the resource usage vector Y.

The SLA management service compares service usage reports with the QoS "constraints" to decide whether a user should be allowed to start new activities or continue with existing activities. Constraints are similar to metrics, except that they represent future usage over some interval (the constraint is always set before the interval). Constraints are used to represent the total capacity of a set of service provider resources, the level of service to be provided in an SLA and the expected usage for a newly requested activity.

The SLA management service compares resource usage reports with the service capacity to detect when resources are becoming overloaded, and may terminate some activities to prevent others failing due to this. It also compares QoS constraints in existing SLA with the service provider capacity constraints to determine whether to accept or refuse a new SLA. To do all this, we define metrics in relation to a model, so the SLA manager can:

- combine usage reports Z from individual activities and get total usage for all activities in an SLA, so this can be compared with SLA constraints on Z;
- combine usage reports Y from individual activities and get total usage for all activities hosted by the service provider, so this can be compared with capacity constraints on Y;
- predict when a set of activities governed by an SLA will collectively breach the usage limit specified by the SLA;
- calculate the resource usage bounds on Y corresponding to quality of service bounds

on Z, so the implications of a new SLA (in terms of Z) can be compared with capacity constraints on Y; and

- calculate the fees to be charged (normally defined as a value X in currency units), for a given usage Z.

This model has to be general, in the sense that it should not depend on what metrics are used by a given application service or constrained by its SLA. Only the mathematical treatment of metrics and constraints of various types should be defined in the model.

4.2 Mathematical Model

A significant complication that must be addressed in our model is that QoS and capacity constraints, and price calculations may be defined in terms of the rate of usage and also the total usage over time. Rate usage information is "instantaneous", relating to a single instant in time, while cumulative usage refers to an interval of time (i.e. has both a start and end time), and represents an integration over that time interval of the instantaneous usage rate. Note that cumulative usage does not appear related to "interval" metrics from WSDM, which are associated with duration (e.g. 1 hour) and not to a true interval between two absolute times.

Application services may deliver reports on instantaneous or cumulative usage, or both. We need a model that allows the SLA manager to derive a complete picture from the metrics available to it, and to combine usage reports from different activities at different times in a coherent fashion. The model should allow the SLA manager to compare usage with constraints on either type of usage metric, even though none of the usage reports coincide in time with the constraint or even with each other (See Fig 1). To support such comparisons, we use the series of mathematical assumptions.

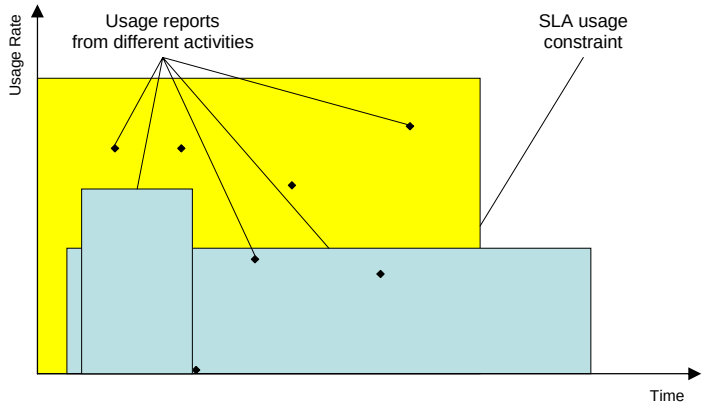

Fig 1. SLA Metrics Challenge

If a cumulative usage is required over some interval, but only rate measurements are available, we will assume that instantaneous rate reports refer to rate changes. Between two rate reports from an activity, the usage rate is assumed to be equal to that given in the earlier report. It is then easy to calculate the cumulative usage over a period during which reports have been received. To match this mathematical assumption, an application service that provides usage rate reports should do so whenever the usage rate of an activity changes. To match this mathematical assumption, an application service that provides usage rate reports should do so whenever the usage rate of an activity changes. Differentiation is used when we have cumulative reports covering a period, but we need a rate at some time in that period. The assumption we make here is that usage rates are constant in the interval covered by a cumulative usage report, but zero outside it. The rate can thus be assumed equal to the cumulative usage divided by the reporting interval duration, within the reporting time interval only. To match this mathematical assumption, an application service would have to provide cumulative usage reports whenever the usage rate changed, with each report covering the interval since the last change. This is not realistic, especially since cumulative usage reports are likely to be most popular when the usage rate is continuously varying. However, there is

no better assumption we could use in the absence of more data, so we have to accept that a rate calculated from a cumulative report using the above assumptions is an approximation. Because of this, usage rate reports will normally be more reliable than rates derived from cumulative reports, and should be used where available.

Aggregation of cumulative usage reports is used to calculate the total usage for a collection of activities in a given interval. We then have to add together the cumulative usage reports across those activities in that period (these usage reports may have been obtained via integration of rate reports). To do this, we use the cumulative usage differentiation assumption described above. It is then possible to work out what proportion of the cumulative usage from each report to include in the aggregate total, based on what proportion of the report interval overlaps the required aggregated interval.

Aggregation of rates is needed when we have a constraint on the rate of usage, and we need to determine whether the actual rate exceeds this. There are three scenarios:

- finding the total usage rate at the current time, e.g. to determine whether a new activity would take the total over a limit, or to determine when a cumulative limit will be reached;
- finding the total usage rate at a specified time in the past; and
- finding the time intervals during which the rate exceeded a specified limit, e.g. to determine whether surcharges should be imposed.

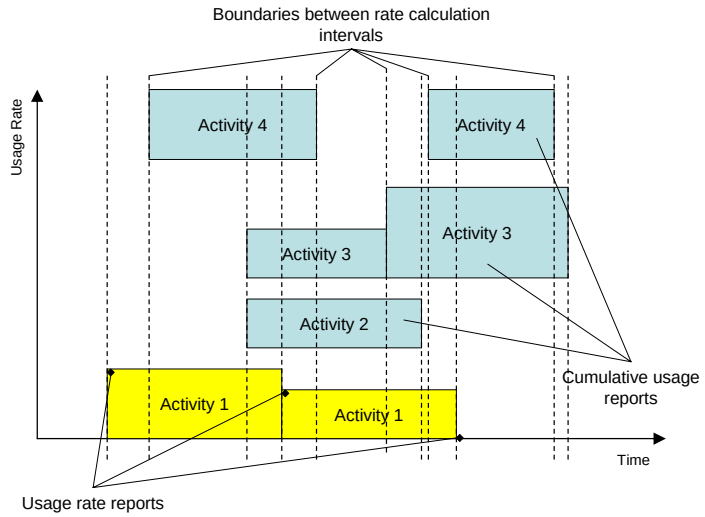

Fig 2. Instantaneous rate limit transgression times

The first scenario is the simplest. Cumulative usage reports only give information about past usage (since their reporting interval cannot extend into the future). They do not tell us anything about usage outside their reporting interval, which must include the current time. Thus the only way to get a current aggregated usage rate is to add together the latest usage rate reports from all the relevant activities. The second scenario is more complicated because if the specified time is in the past, we may have cumulative reports for some activities. The solution is to find the most recent rate report from each activity prior to the specified time and add them together. Then if there are any activities that did not produce rate reports, we should add the average rates from cumulative usage reports whose reporting intervals span the specified time. In this way, we ensure that directly reported rates are used from activities that provide such reports, but we still take account of activities that can't – e.g. because their usage rates are continuously varying. The last scenario is most complicated, as shown in Fig 2. To work out when the usage rate exceeds a certain level, we need to know how the usage rate of each activity changes, and what the rate was in the intervals between changes. Thus it is easier to work with "cumulative" usage data expressed over time intervals, even if these are actually derived from usage rate reports (as for Activity 1 in Fig 2). We then take the start and end times for all these intervals across all relevant activities, and define short intervals between them. For each of these intervals, we can find the total usage rate across activities by adding

together the rates (direct or derived) from activity reports that overlap in time with the interval. It is then trivial to compare with a usage rate limit and determine in which of these short intervals the usage exceeded the rate limit.

5 Architecture

The new model is implemented through the use of service level agreements that define QoS and other commitments by the service provider, in exchange for financial commitments by the consumer. As described above, the QoS model is very generic and is capable of handling a wide class of application services, allowing QoS to be expressed in user-centric as well as resource-centric terms. The SLA service acts as a "service level agreement manager" providing capabilities:

i) to provide access to service level agreement "templates" that can be filled in and submitted by a potential consumer as service level "proposals";
ii) to decide whether to enter into a new service level agreement (with a given QoS) when proposed by a consumer, and to respond accordingly;
iii) to decide whether a requested application service "activity" is covered by an existing service level agreement, and to detect when such an activity (even if covered) would exceed the capacity of the provider;
iv) to decide which service level agreement(s) should be breached when capacity is about to be exceeded, and to initiate load reductions in the corresponding application service(s);
v) to track the quality of service actually delivered, and to initiate charges when appropriate;
vi) to detect when a consumer is exceeding the limits of a service level agreement, and to initiate load reductions in the corresponding application service(s) to prevent this.

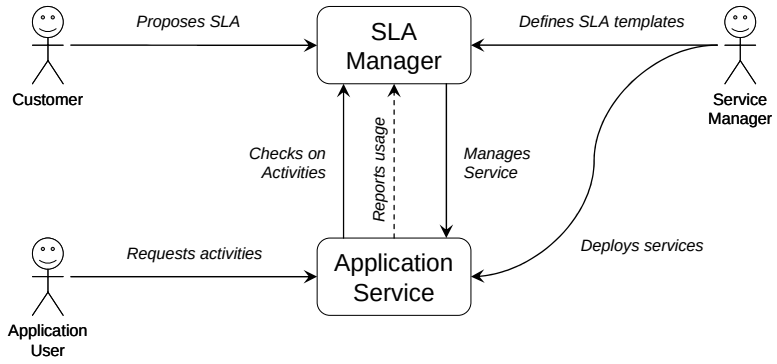

Fig 3. Interactions between the GRIA SLA manager and an application service

The Service Manager in Fig 3 is responsible for deploying application services, and making sure that the SLA templates available from the SLA manager cover the deployed services. A Customer can obtain one of these templates, fill it in, and send it to the SLA manager as an SLA proposal. At this point the SLA manager may accept the request, and start using the proposed SLA to manage services, or refuse the request. If accepted, the SLA manager will respond with a WS-Addressing Endpoint Reference (EPR) including a context reference for the SLA. The Customer must include this reference in the context header for subsequent requests to the SLA manager related to this SLA.
The Application User in Fig 3 interacts with application services provided under the terms of an SLA. The nature of these interactions obviously depends on the application service, but in general, the application service will consume resources in order to respond to the Application User. It should do so only to the level specified in the SLA, or the service provider resources may become overloaded and be unable to response to other Application Users to the required level. The Application User must specify the EPR of the SLA (i.e. the SLA manager address

plus SLA context reference) when initiating an interaction. If the user has no SLA, or specifies an SLA that doesn't cover the application service interaction, or to which this user has no access rights, the application service should reject the interaction.
The application services will ask the SLA manager when they need to check if the SLA requires them to respond to a user request. The SLA manager will ask the application service to terminate an interaction if it becomes necessary to stop the service from consuming resources under a given SLA. In addition, the application service should report on the usage of the service (Z) and the use of resources by the service (Y) – these reports can be sent asynchronously, since the application service doesn't need a reply.

6 Deployment by SIMDAT Industrial Partners

The GRIA SLA Management service has been deployed in SIMDAT to support business partnerships in the aerospace, automotive and pharmaceutical sectors. Aerospace companies have developed a prototype that demonstrates how Grid technologies can support pan-European inter-Enterprise collaborative development of complex products. Each organisation operates as a GRIA service provider offering specialised engineering services such as optimisation (University of Southampton), parameterised CAD generation (University of Southampton), aerodynamics (BAE SYSTEMS) aero-acoustics (EADS), and structural analysis (MSC). GRIA's explicit business process support for dynamic, bi-lateral SLAs allows project managers at aerospace companies to not only create distributed multidisciplinary engineering design teams but also quickly procure additional analysis services capabilities from suppliers if necessary [10]. In the pharmaceuticals sector, a significant challenge for companies is how to reduce cost and risk whilst increasing the probability of successfully developing new drugs. Innovative solutions to assist drug development exist both within large pharmaceutical and outside in the hands of hundreds of biotechnology businesses and academic institutions. The GRIA SLA Management service has been deployed to support business partnerships between GSK, Inpharmatica and academic service providers, fuelling the drug discovery pipeline. In the automotive sector, both Audi and Renault have deployed GRIA to support the product design process chain (CAE/CAD/CAT) including external engineering companies, developers and design suppliers.

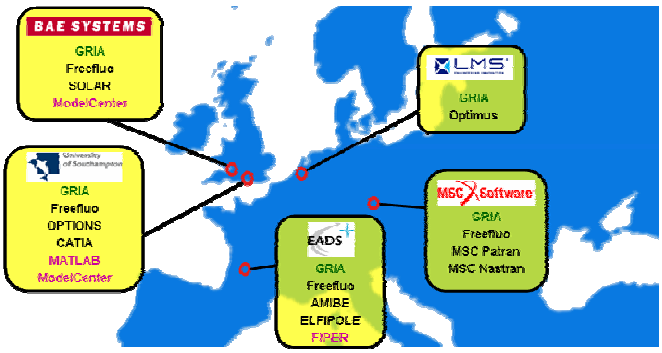

Fig 4. Aerospace business partnerships

7 Conclusions

In SIMDAT, GRIA has been successfully deployed to support business partnerships in the aerospace, automotive, and pharmaceutical sectors. The SLA Management service allows service providers to advertise SLA templates that are proposed by customers during SLA negotiation. The SLA describes quality of service (QoS) and other commitments by a service provider in exchange for financial commitments by a customer against an agreed schedule of prices and payments. The SLA management service supports a very generic QoS model that allows application service developers to define their own qualitative QoS criteria in terms of their operations, including error and failure criteria. Metrics usage can be monitored, constrained and billed for using instantaneous rate measurements at some absolute time, and

total usage over a defined period between two absolute times. The SLA negotiation is automated making GRIA services very responsive to new user needs, yet inexpensive to operate for providers. This is critical in a business Grid where human resources can easily become the largest cost when running services - in GRIA these costs are minimised, so services can be profitable for providers while still being affordable for customers.
Future work focuses on building upon GRIA's SLA service core to provide higher-level intelligent business services that can be used to drive SLA offerings and dynamic provisioning. GridEcon [11] is developing a market based Pricing Service and ArguGRID [12] is developing agent-based argumentation services to support SLA decision making and negotiation. Other advancements include dynamic service provider configurations, where applications are installed on demand to satisfy client requests. To manage such systems effectively requires enhancements to the SLA service to both record, and provide to other components, more information such as queue lengths and response times. These performance-based metrics are of great importance to consumers who need to make business decisions about service provider performance based on historical data. More dynamic cost models and the ability to constrain and monitor functions of the basic metrics will also be investigated in order to encode penalties for under-performance levied against the service provider.

The SLA Management service is part of GRIA's Service Provider Management package, which is available for download, free and open source, from www.gria.org.

References

1. EU IST SIMDAT Project, www.simdat.eu
2. Surridge, M., Taylor, S., De Roure, D. and Zaluska, E. (2005) Experiences with GRIA — Industrial Applications on a Web Services Grid, in Proceedings of the First International Conference on e-Science and Grid Computing, pp. 98-105. IEEE Press.
3. Addis, M. J., Allen, P. J. and Surridge, M. (2000) Simulation on Demand. E-business: Key Issues, Applications, Technologies pp. 906-912.
4. FIPA Interaction Protocol Specifications, http://www.fipa.org/repository/ips.php3
5. NextGRID Architecture Vision, http://www.nextgrid.org/download/publications/NextGRID_Architecture_White_Paper.pdf
6. Surridge, M., Payne, T. R., Taylor, S. J., Watkins, E. R., Leonard, T., Jacyno, M. and Ashri, R. (2006) Semantic Security in Service Oriented Environments. In Proceedings of UK e-Science Programme All Hands Meeting 2006 (AHM2006) (in press)
7. GGF Usage Record specification, http://www.psc.edu/~lfm/PSC/Grid/UR-WG/UR-Spec-gfd.58-ggf18.doc
8. Job Submission Description Language (JSDL) Specification, Version 1.0, http://www.gridforum.org/Public_Comment_Docs/Documents/July-2005/draft-ggf-jsdl-spec-021.pdf
9. WSDM 1.1 OASIS Standard Specifications, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
10. Upstill, C. and Boniface, M. J. (2005) SIMDAT. CTWatch Quarterly 1(4) pp. 16-24.
11. GridEcon, http://www.gridecon.eu/
12. ARGUGRID, http://www.argugrid.org/