Enforcing Service Level Agreements using an Economically Enhanced Resource Manager

Mario Macías #[1], Garry Smith *[2], Omer Rana **[3], Jordi Guitart #[4], Jordi Torres #[5]

#*Technical University of Catalonia - Barcelona Supercomputing Center*
c/ Jordi Girona 29, 08034 Barcelona, Spain.
[1]mario.macias@bsc.es
[4]jguitart@ac.upc.edu
[5]jordi.torres@bsc.es

**School of Systems Engineering, University of Reading*
RG6 6BX, UK
[2]garry.smith@computer.org

***School of Computer Science, Cardiff University*
CF24 3AA, UK
[3]O.F.Rana@cs.cardiff.ac.uk

Abstract— **Traditional resource management has had as its main objective the optimisation of throughput, based on parameters such as CPU, memory, and network bandwidth. With the appearance of Grid Markets, new variables that determine economic expenditure, benefit and opportunity must be taken into account. The SORMA project aims to allow resource owners and consumers to exploit market mechanisms to sell and buy resources across the Grid. SORMA's motivation is to achieve efficient resource utilisation by maximising revenue for resource providers, and minimising the cost of resource consumption within a market environment. An overriding factor in Grid markets is the need to ensure that desired Quality of Service levels meet the expectations of market participants. This paper explains the proposed use of an Economically Enhanced Resource Manager (EERM) for resource provisioning based on economic models. In particular, this paper describes techniques used by the EERM to support revenue maximisation across multiple Service Level Agreements.**

I. Introduction

The Self-organising ICT Resource Management (SORMA) [1] is an EU IST [2] funded project aimed at developing methods and tools for efficient market-based allocation of resources, using a self-organising resource management system and market-driven models, supported by extensions to existing grid infrastructure. Topics addressed include Open Grid Markets, economically-driven middleware, and intelligent support tools.

Unlike traditional grid environments, jobs submitted to SORMA are matched with available resources according to the economic preferences of both resource providers and consumers, and the current market conditions. This means that the classic grid job scheduler, which is based on performance rules, is replaced by a set of self-organising, market-aware agents that negotiate Service Level Agreements (SLAs), to determine the 'best' resource allocation to fulfil both performance and business goals. In SORMA, an Economically Enhanced Resource Manager (EERM) exists at each resource provider's site, and acts as a centralised resource allocator to support business goals and resource requirements.

While a number of different economic models may be used to support resource management, this paper will focus on adaptation mechanisms to support revenue maximisation across multiple SLAs. In other words, when an EERM receives job/service reservations and associated SLAs, the EERM must allocate, monitor and enforce resource constraints in order maximise the number of jobs whose SLAs can be satisfied. However:

- the EERM does not have the ability to decide which jobs must be accepted or rejected. It is only used for consultative purposes. Even if the EERM advises that it cannot fulfill an incoming task, the economic agents could decide to send it to EERM to increase revenue.
- The EERM uses a predictive model to calculate the impact of a task execution. The prediction could be wrong, and the system would accept a job which could not be fulfilled, resulting in system overload.
- An abnormal situation could reduce the number of available resources; for example, some nodes of an available cluster could crash.

In the cases described before, the service provider would have a reduced number of resources, and the system becomes overloaded. The approach adopted in this work aims to minimise the economic impact of SLA violations, whilst at the same time attempting to enable as many jobs as possible to execute to completion.

The remainder of the paper is structured as follows: Section II presents related work. Section III defines a resource allocation scenario and describes revenue maximisation and SLA issues. Section IV describes the EERM's architecture and highlights important features. Section V contains an example scenario that shows the EERM in action. Finally, section VI concludes the paper and describes our future work.

II. Related Work

QoS has been explored in various contexts, such as for mobile devices [3] and multimedia applications [4]. Two types of QoS attributes can be distinguished: those based on quantitative, and qualitative resource characteristics. Qualitative characteristics refer to aspects such as service reliability and user satisfaction. Quantitative characteristics refer to aspects such as network latency, CPU performance, or storage capacity. Although qualitative characteristics are important, it is difficult to measure these objectively. Systems which are centered on the use of such measures utilise user feedback [5] to compare and relate measures to particular system components. Our focus is primarily on quantitative characteristics.

Sahai et al. [6] propose an SLA management entity to support QoS in the context of commercial grids. They envision the SLA management entity existing within the Open Grid Services Architecture (OGSA), with its own set of protocols for manageability and assurance; they also describe a language for SLA specification. Although an interesting approach, this work is still at a preliminary stage, and the general applicability of this work is not obvious.

The Service Negotiation and Acquisition Protocol (SNAP) [7] is a resource management model to negotiate resources in distributed systems such as grids. SNAP defines three types of SLAs that coordinate management across a desired resource set, and can be used to describe complex distributed service requirements. Resource interactions are mapped to well-defined, platform-independent, SLAs with the SNAP protocol managing resources across different administrative domains, via three types of SLAs: Task SLA (TSLA), Resource SLA (RSLA) and Bind SLA (BSLA). The TSLA describes the task that needs to be executed, and the RSLA describes the resources needed to accomplish this task. The BSLA provides an association between the resources from the RSLA and the application 'task' in the TSLA. The SNAP protocol requires the existence of a resource management entity that can provide guarantees on resource capability; for example, RSLA.

Keahey et al. [8] propose an architecture called Virtual Application Service (VAS) for managing QoS in computational grids. VAS is an extended grid service with additional interfaces for negotiation of QoS level and service demands. The key objective of VAS is to facilitate the execution of real-time services with specific deadline constraints. A client submits a request to VAS for advance or immediate reservation of a service; supplying only time constraints. Essentially, VAS is a deadline-bound system, and the client can only specify time constraints as a QoS metric; VAS requires the application to predict how long it will need to run. Subsequently, VAS computes the time needed for service execution, based on a prediction model and service metadata.
In our approach we make use of different allocation strategies to run user applications, based on whether they require a *Time-domain* or *Resource-domain* allocation strategy. For users who request a Time-domain allocation, 100% of the computational resources must be allocated to their jobs. Whereas VAS requires users to benchmark their applications by running them first on an unloaded CPU, we utilise the results of application execution times where a guaranteed service execution has been requested, and use these as a benchmark. Burchard et al. [9] also propose the use of SLAs to negotiate service execution parameters between resource managers. The SLA management is achieved via a Virtual Resource Manager (VRM). The VRM acts as a coordinator to aggregate SLAs negotiated with different sub-systems. Although the SLA management in this work is similar to our effort, the focus in our approach is on utilising the service paradigm, where the VRM is intended to integrate execution across a number of co-located clusters.

The General-purpose Architecture for Reservation and Allocation (GARA) [10] is the most commonly known framework to support QoS in the context of computational Grids. GARA allows users to specify end-to-end QoS requirements and provides advance reservations to various resources through a uniform interface. GARA's reservation is aimed at providing a guarantee that the client or application initiating the reservation will receive a specific QoS from the resource manager. Although GARA has gained popularity in the Grid community, it has limitations in coping with current application requirements and technologies, including: GARA is not OGSA-compliant; GARA does not support the concept of an agreement protocol to support the simultaneous allocation of resources; QoS monitoring and adaptation during the active QoS session is one of the most important and successful mechanisms to date in providing a quality guarantee [11], however, GARA does not provide adaptive functions to support this.

III. Scenario Definition

Multiple economic enhancements exist that could be applied to resource management. In this paper we focus on only those related to revenue maximisation across multiple SLAs. However, an aim of our work is to provide a framework that will allow grid economists to define their own rules to achieve their particular goals. Therefore the content of this paper should be considered as a particular view of how the system behaves.

The **SLA Satisfaction Function** determines if, for a set of n resources $R = \{R_1, R_2, \ldots, R_n\}$, an SLA S can be fulfilled (results *true*) or will be violated (results *false*).

The **Multiple SLA Satisfaction Function** determines if, for a set of n resources $R = \{R_1, R_2, \ldots, R_n\}$, a set of m SLAs $\{S_1, S_2, \ldots, S_m\}$ can all be fulfilled or if any SLA will be violated.

Consider the follwing scenario – a set of running jobs each with its own SLA, is assigned to a resource. Each time a new job/SLA pair arrives, the EERM must assign a portion of the resource bundle. There are two possible scenarios:

- There are enough free resources, so the Multiple SLA Satisfaction Function is *true*. In this case, it is trivial to allocate the incoming tasks to a suitable resource. This scenario will not be studied in this paper.

- There are not enough resources (Multiple SLA Satisfaction function is *false*), implying that an intelligent resource re-allocation mechanism is required for maximising revenue and minimising SLA violation penalties.

A. Revenue Maximisation in Resource-Limited Providers

The **revenue** Rev_i is the amount of money that a client will pay if a provider fulfills the SLA S_i. The revenue is specified in the same SLA and usually has a fixed value. On the other hand, we define **penalty** Pen_i as the amount of money that the provider must pay if the SLA S_i is violated. The penalty is also specified in the same SLA and can be a function with parameters specified as in section III-B.

The **gain** $G(S_i)$ is the economic benefit that the provider obtains with the execution of a job whose SLA is S_i. It is defined as $G(S_i) = Rev_i - Pen_i$ and it can be positive (provider earns money) or negative (SLA violation with high penalty costs).

In a pool of resources R, executing a set of SLAs S at concrete instant t we define the **punctual gain** as:

$$\triangle G(t, R) = \sum_{i=1}^{m} G(S_i) = \sum_{i=1}^{m} Rev_i - \sum_{i=1}^{m} Pen_i$$

which is the gain (or loss) obtained if the current jobs all execute and finish on the resources that were assigned at instance t.

When a new SLA S_i arrives and there are not enough resources, system overload will cause the provider to start violating SLAs. To avoid (or minimise) violation penalties and maximise revenue, we suggest two complementary solutions:

- Dynamic adaptation in terms of resource provisioning. Previous work [12] has demonstrated that we can increase both the throughput and the number of jobs completed, by dynamically adapting the share of available resources between the applications by a function of demand. This is feasible when several applications share a single multiprocessor platform (by assigning priorities and processors) or in virtualised environments [13], by dynamically assigning resources and priorities for each virtual machine).
- Task reallocation; finding a new resource assignation R' for each job i associated with the SLA S_i. The new gain will be defined as

  $$\triangle G'(t, R) = \sum_{i=1}^{m} G'(S_i) - M(S, R)$$

  where $M(S, R)$ is the economic cost of migrating the current running jobs S within the resource bundle R. When reallocating tasks, the main challenge for the EERM will be to find the highest $\triangle G'(t, R)$, by predicting the new gain for each possible assignment of resources, and trying to minimise the cost of resource reallocation $M(S, R)$.

B. SLA Violation

Monitoring SLA Violation begins once an SLA has been defined. A copy of the SLA must be maintained by both the client and the provider. It is necessary to distinguish between an ‘agreement date’ (agreeing on an SLA) and an ‘effective date’ (subsequently providing a service based on the Service Level Objectives (SLOs) that have been agreed). A request to invoke a service based on the SLOs (which are the SLA terms), for instance, may be undertaken at a time much later than when the SLOs were agreed. During provision it is necessary to determine whether the terms agreed in the SLA have been met. In this context, a monitoring infrastructure is used to identify the difference between the agreed upon SLO and the value that was actually delivered during provisioning. It is also necessary to define what constitutes a violation. Depending on the importance of the violated SLO and/or the consequences of the violation, the provider in breach may avoid dispatch or obtain a diminished monetary sanction from the client.

An SLA may be terminated in three situations: (i) when the service defined in the SLA has completed; (ii) when the time period over which the SLA has been agreed upon has expired; and (iii) when the provider is no-longer available after an SLA has been agreed (for instance, the provider’s business has gone into liquidation). In all three cases, it is necessary for the SLA to be removed from both the client and the provider. Where an SLA was actually used to provision a service, it is necessary to determine whether any violations had occurred during provisioning. As indicated above, penalty clauses are also part of the SLA, and need to be agreed between the client and the provider.

One of the main issues that the provider and the consumer will have to agree during the SLA negotiation is the penalty scheme or the sanctioning policies. Since both the service provider and the client are ultimately businesses (rather than consumers), they are free to decide what kind of sanctions they will associate to the various types of SLA breaches, in accordance with the weight of the parameter that was not fulfilled. We define the following broad categories of violation:

- ‘All-or-nothing’ provisioning: provisioning of a service meets all the SLOs – i.e. all of the SLO constraints must be satisfied for a successful delivery of a service;
- ‘Partial’ provisioning: provisioning of a service meets some of the SLOs – i.e. some of the SLO constraints must be satisfied for a successful delivery of a service;
- ‘Weighted Partial’ provisioning: provision of a service meets SLOs that have a weighting greater than a threshold (identified by the client).

Monitoring can be used to detect whether an SLA has been violated. Typically such violations result in a complete failure – making SLA violations an ‘all-or-nothing’ process. In such an event a completely new SLA needs to be negotiated, possibly with another service provider, which requires additional effort on both the client and the service provider. Based on this all-or-nothing approach, it is necessary for the provider to satisfy all of the SLOs. This equates to a conjunction of SLO terms. An SLA may contain several SLOs, where some SLOs (e.g. at least two CPUs) may be more important than others (e.g. more than 100 MBytes of hard disk space). During the SLA negotiation phase, the importance of the different SLOs may be established. Clients (and service providers) can then react differently according to the importance of the violated SLO. In the WS-Agreement specification [14], the importance of particular terms is captured through the use of a 'Business Value'.

Weighted metrics can also be used to provide a flexible and fair sanction mechanism, in case an SLA violation occurs. Thus, instead of terminating the SLA altogether it might be possible to re-negotiate, i.e. with the same service provider, the part of the SLA that is violated. Again, the more important the violated SLO, the more difficult (if not impossible) it will be to re-negotiate (part of) the SLA.

IV. Economically Enhanced Resource Manager

The overall aim of the EERM is to isolate SORMA economic layers from the technical ones and orchestrate both economic and technical goals to achieve maximum economic profit and resource utilisation. The main goals of the EERM are:

- To combine technical and economic aspects of resource management.
- Perform resource price calculations, taking into account current market supply and demand, performance estimations and business policies.
- To strengthen the economic feasibility of the Grid.

To provide a general solution that supports different scenarios and business policies, the EERM should provide flexibility in defining user (administrator) configurable rule-based policies, to support:

Individual Rationality
: An important requirement for a system is that it is individually rational on both sides, i.e. both providers and clients have to have a benefit from using the system. This is a requirement for the whole system, including features such as client classification or dynamic pricing.

Revenue Maximisation
: A key characteristic for SORMA providers is revenue (utility) maximisation. The introduced mechanisms can indeed improve the utility of both provider and client.

Incentive Compatibility
: Strategic behaviour of clients and providers can be prevented if a mechanism is incentive compatible. Incentive compatibility means that no other strategy results in a higher utility than reporting the true valuation.

Efficiency
: There are different types of efficiency. The first one considered here is that no participant can improve its utility without reducing the utility of another participant. The second efficiency criterion is allocative efficiency: i.e. the EERM must maximise the sum of individual utilities.

A. Architecture

The EERM's architecture is shown in Figure 1. To place the EERM in the context of the SORMA framework, we have also shown the SORMA Grid Market Middleware (GMM) [15], which provides the mechanisms to interact with the SORMA market. Once resource usage has been agreed in the SORMA market, a contract is sent to the EERM over the GMM. The contract provides the EERM with input for resource allocation, task execution and SLA enforcement activities. The EERM is comprised of the following components (see Figure 1):

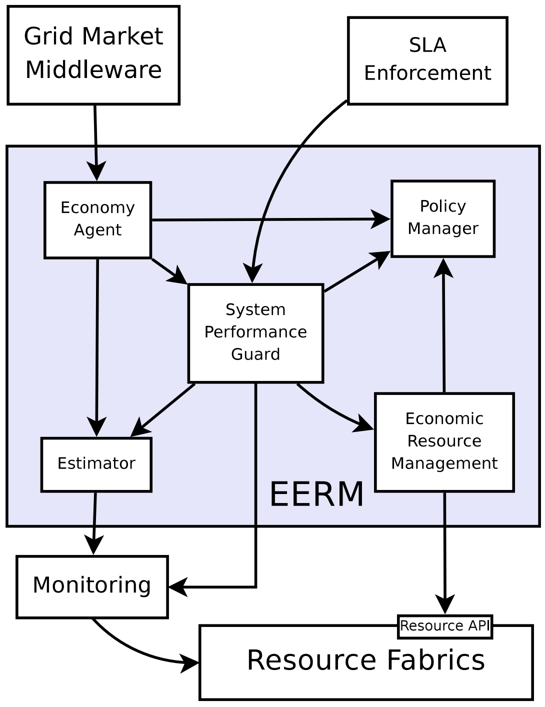

Fig. 1. EERM components

Economy Agent (EA)
: The EA receives requests from SORMA market agents over the GMM. For each request, the EA checks whether the job is technically and economically feasible and calculates a price for the job based on the category of client (e.g. a preferred customer), resource status, economic policies, and predictions of future resource availability (provided by the Estimator Component). The EA interacts with the upper SORMA economic layers in the SLA negotiation process.

Estimator Component (EC)
: The EC calculates the expected impact on the utilisation of the Grid and is based on Kounev, Nou, and Torres [16]. In short, the EC's task is to avoid performance loss due to resource overload [12].

System Performance Guard (SPG)
: The SPG monitors resource performance and SLA

violations. If there is a danger that one or more SLAs cannot be fulfilled, the SPG can take the decision of suspending, migrating or cancelling jobs to ensure the fulfilment of other, perhaps more important, SLAs with the aim of maximising overall revenue. Jobs can also be cancelled when additional capacity is required to fulfil commitments to preferred clients. The policies that dictate when to take action and which types of jobs should be killed, migrated or suspended are updated via the Policy Manager.

Policy Manager (PM)
: The PM stores and manages policies concerning client classification, job cancellation or suspension. Policies are formulated using the Semantic Web Rule Language (SWRL) [17]. The PM is an important part of the EERM in that it allows behaviour to be adapted at runtime. With the exception of the EC, all other EERM components use the PM to obtain policies that affect their decision making process.

Economic Resource Manager (ERM)
: The ERM interacts with local resource managers and is responsible for ensuring an efficient use of local resources. The ERM is described in further detail in Section IV-B.

Resource Monitoring (RM)
: The RM provides resource information for system and per-process monitoring. Resource information is used by the EC, SPG, ERM and SLA components. The RM is explained in further detail in Section IV-C.

SLA Enforcement (SLAE)
: The SLAE is tasked with monitoring SLA fulfillment. The SLAE uses monitoring data from the EERM and RM. When an SLA violation is detected, the SLAE takes reactive measures such as SLA re-negotiation or compensation retrieval based on SLA penalty clauses. This component is explained in further detail in Section IV-D.

B. Economic Resource Manager (ERM)

The ERM (Figure 2) is designed to interact with a range of execution platforms (e.g. Condor, Sun Grid Engine, Globus GRAM, or UNIX fork) and achieves this using Tycho [18] connectors that communicate over the network to Resource Agents (RA).

The RA translates XML messages from the ERM into messages understood by the underlying platform (e.g. Condor). In addition, RAs provide a consistent interface to the different underlying resource fabrics. This means that another platform can be adapted to SORMA by implementing an appropriate RA plug-in that performs translations to and from the underlying resource manager's native protocol. It is intended that access to the existing middleware be constrained by firewall rules, so that all interactions must go through the ERM. As a single point of access, the ERM can provide additional functionality that the underlying middleware may lack, for example, by providing support for advanced reservations.

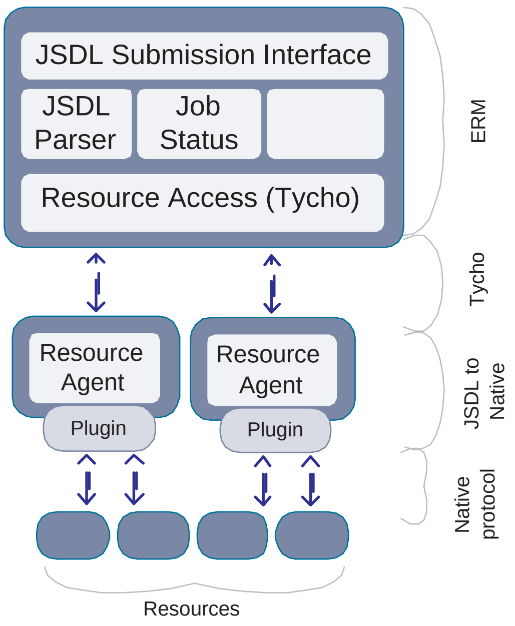

Fig. 2. ERM implementation

In the current prototype, resource agents include a plugin for launching JSDL [19] jobs using GridSAM [20]. The approach used to implement the ERM is complimented by a similar approach used for resource monitoring.

C. Monitoring

In order to enable SLA enforcement, an understanding of the current and recent state of the underlying resources is required. Resource availability and utilisation can be sampled periodically in a coarse-grained manner in order to provide a high-level understanding of general Quality of Service (QoS) indicators. At other times it may be appropriate to target particular and detailed attributes that reflect a given resources' ability to fulfil a particular action, e.g. the execution of a job. In addition notifications received from resources when a particular threshold has been exceeded can help to identify SLA violations. The EERM employs the GridRM [21] wide-area distributed monitoring system to gather data required for SLA enforcement.

The GridRM design employs gateways for gathering data from a number of different types of resources that make up the Grid. Resources of interest can include all manner of networked devices, from a remote sensor or satellite feed through to a computational node or a communications link. The Gateway is used internally, to a Grid-enabled site (the local layer), to configure, manage and monitor internal resources, while providing controlled external access to resource information. The EERM is bound to its local GridRM Gateway using the Tycho distributed registry and messaging system (see figure 3). The EERM queries the gateway for real-time and historical resource data, and registers interest to receive different types of events that reflect changes in resource state (e.g. completion of a submitted job, system load greater than a specified threshold).

Resources may already provide legacy agents e.g. SNMP, Ganglia, /proc, Condor. As long as the Gateway is installed with a driver that supports the agents native protocol, then all resource data provided by the native agent can be retrieved. In cases where an existing agent is not installed, a proprietary agent can be used for information gathering. Using a native agent means that existing resources can be monitored with little or no modification. Alternatively, installation of the proprietary GridRM agent implies some administrative overhead for each resource, but can result in improved performance and lower intrusiveness when gathering data.

Resource heterogeneity (agent and platform type) is hidden from GridRM clients and hence the EERM; the Structured Query Language (SQL) [22] is used to formulate monitoring requests, and a SORMA-specific schema based on the GLUE Schema [23] is used to group data and format the results into a consistent form (semantically and in terms of the values returned from different agents). Currently the SORMA consortium have identified a number of core attributes that are used for monitoring resources, enforcing SLAs, advertising resources on the market and for match making purposes. The core attributes include:

- CPU (architecture, number of, speed),
- Operating System (type, kernel version, shared libraries),
- Memory (total/free physical/virtual),
- Disk (total/free, network/local),
- Per-process execution statistics (start stop times, CPU time, memory footprint, exit status).

The current set of core attributes are a starting point and will evolve over time, as the requirements for more complex SLA enforcement are understood.

As well as real-time information a need exists to capture historical data so that the SLA enforcement component can determine the likelihood of an SLA violation, based on past resource provision at a given site. The gateway can be instructed to query particular core attributes at a given frequency and store the results in its internal database. The consistent view of resource data provided by the GridRM gateway means that the SLA enforcement component is not exposed to resource heterogeneity and hence can focus on performing its core duties of SLA monitoring and enforcement.

D. SLA Enforcement

The aim of the SLA enforcement component is to detect any SLA violation before it occurs, by evaluating the real time data about a provider to determine if a trend can be identified that fits in with a model that has shown, in the past using historical data, to result in a possible violation. Since each SLA would consist of several resource attributes, the monitoring data collection will be for a metric that would represent collected statistical information about the resource providers past and current resource provision.

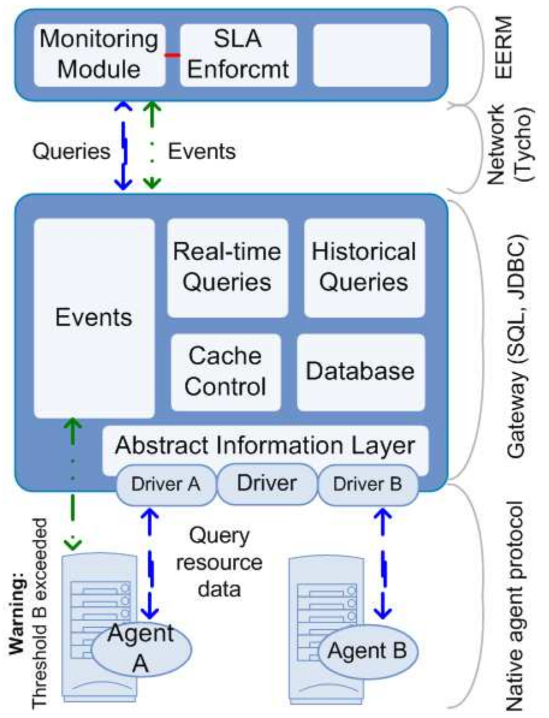

Fig. 3. The GridRM Gateway and relationship to the EERM

It is important to note that the SLA enforcement is not just for the providers to meet their commitments, it has to be monitored to validate that the consumers have met the SLA. One of the most important aspects to monitor that are relevant to consumers is the possibility of overuse of the resources than that agreed in the SLAs.

The interaction between the EERM and the SLA Enforcement component is described in figure 4. The process begins when SLA Enforcement component receives a contract from SORMA Contract Management (the element which creates the contracts once a negotiation is agreed between providers and customers). After this, the SLA is created and sent to the EERM, which watches for its fulfillment. The EERM takes the economic data from SLA Enforcement and the performance data from Monitoring components to detect if an SLA is being violated, and performs a selective violation of SLAs to maximise the revenue.

On violation, the SLA Enforcement component detects this and generates a notification for the SORMA economic layers, in order to negotiate a new contract or give clients the possibility of searching for another provider.

V. Example Scenario

To explain the operations of SLA fulfillment using the EERM, we have designed a simple conceptual scenario (see Figure 5): A resource provider wishes to sell the CPU time of four multi-processor machines. There are some free resources, and some running tasks whose revenues are specified in their SLAs. In order to simplify, there are two fixed economic parameters:

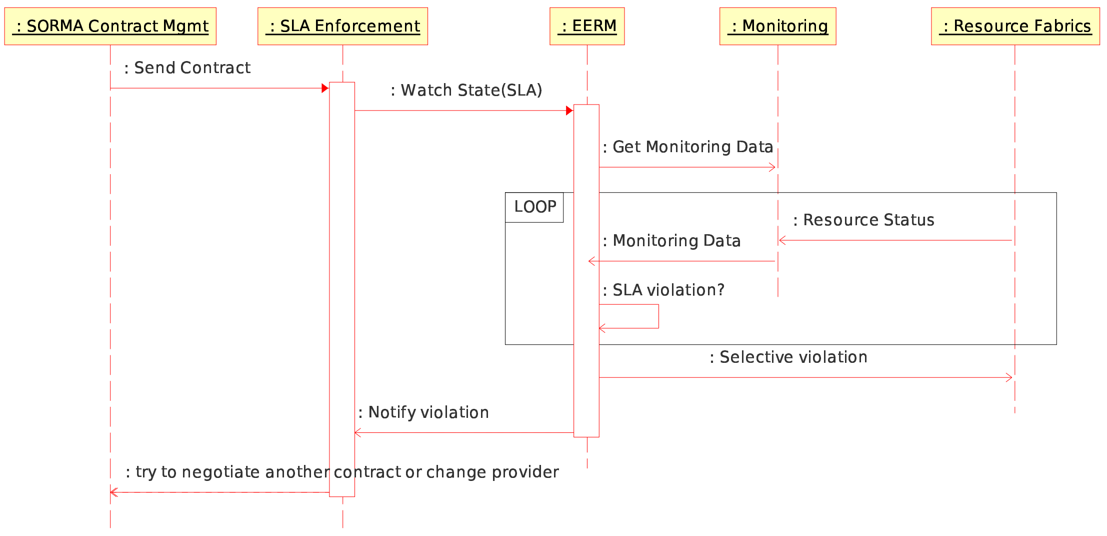

Fig. 4. SLA Enforcement and EERM components interaction

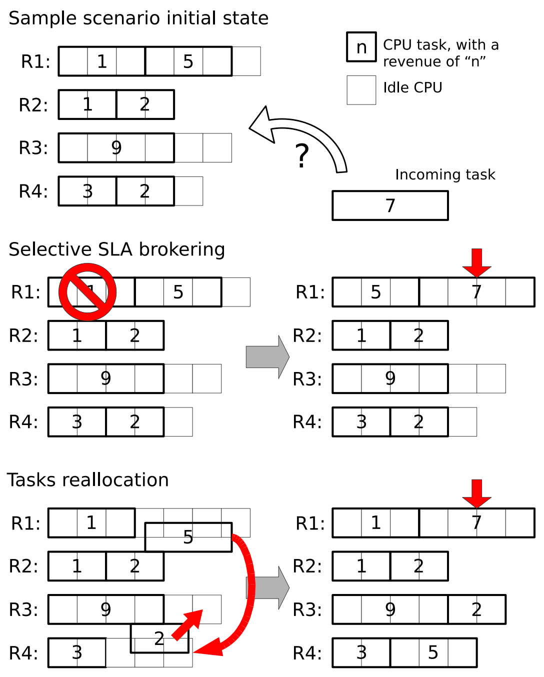

Fig. 5. Enforcement of SLA fulfillment example scenario

- Penalty for SLA violation: four currency units per violation, specified in each SLA.
- Task migration: one currency unit per migration; an indirect cost, calculated by the resource provider.

In the example scenario described in the upper schema of Figure 5, a new task arrives, and its SLA specifies a requirement for 4 CPUs and a revenue of 7. The incoming task does not fit in any resource, and therefore risks breaking the SLA. In response, the EERM could take three different actions:

1) Deny resource allocation for the incoming task. This is a non-economic response and means that the EERM has fallen back to the same behaviour as traditional resource management systems. Because the SLA has been agreed previously, if this response is taken the incoming task SLA will be broken and the provider will have to pay a penalty of 4. Using the formulas proposed in section III-A, the provider obtains a punctual gain of:

$$\triangle G(t,R) = \sum_{i=1}^{m} Rev_i - \sum_{i=1}^{m} Pen_i = 23 - 4 = \mathbf{19}$$

2) Perform a selective SLA violation. In the middle schema of Figure 5, the EERM determines that the first task in R1 can be terminated due to the low revenue associated with that task. As a result the incoming task is now able to fit into R1. The punctual gain for the provider is:

$$\triangle G'(t,R) = 29 - 4 = \mathbf{25}$$

3) Reallocate resources. In this particular case, there are 4 free CPUs, but they are scattered across the resource bundle. Reallocating tasks to provide a single machine with 4 CPUs may be cheaper than breaking the SLA. For example, the lower schema of 5, shows task migration which results in a new punctual gain of:

$$\triangle G'(t,R) = \sum_{i=1}^{m} G'(S_i) - M(S,R) = 30 - 2 = \mathbf{28}$$

By applying economic enhancements into resource management, a provider can dramatically increase its revenue (47% in the example) by choosing the correct policy for SLA brokering or task reallocation. Determining the optimal solution for a given scenario will depend on penalty and reallocation costs as well as current resource availability.

VI. Conclusions and Future Work

The work reported in this paper is motivated by the need to extend traditional resource management with economic parameters to support emerging grid markets. Within a grid market, resource providers must consider issues relating to current market conditions, QoS, revenue maximisation, economic sustainability and reputation, if they are to operate effectively.

In particular, this paper focuses on revenue maximisation using SLAs and describes how a strategic approach to managing SLAs can be used to secure optimal profit in situations where resources are scarce. Using our methods for selective SLA fulfilment and violation, the resource provider can determine which jobs should be pre-empted in favour of freeing up resources for more lucrative SLAs. For example, it may be more profitable to violate an existing SLA, and pay the associated penalty, than it is to checkpoint and redistribute existing jobs, so that all SLAs can be fulfilled.

A prototype EERM is introduced and its architecture described. The EERM is a first attempt at providing strategic SLA enforcement within a grid market and forms part of the market mechanisms currently being implemented by the SORMA project.

Future work will include the identification of policies and parameters suitable for enforcing revenue maximisation given a number of different resource scenarios. The aim is to understand how to determine an optimal solution (or sub-optimal if the computation cost is too great) across a resource pool when complex policies and multiple economic parameters are at play. Another line of work will address how EERMs can be used to provide input to the market so that the negotiation process between customers and providers results in the generation of more accurate SLAs.

Acknowledgement

We would like to thank Mark Baker, SSE, University of Reading, for his comments during the writing of this paper. This work is supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2004-07739-C02-01 and Commission of the European Communities under IST contract 034286 (SORMA). Thanks are also due to Martijn Warnier and Thomas Quillinan of Vrije University, Amsterdam, The Netherlands – for discussions about types of violations that could arise in SLAs.

References

[1] Self-organizing ICT Resource Management (SORMA). [Online]. Available: http://www.sorma-project.eu

[2] Information Society Technologies Programme. [Online]. Available: http://www.cordis.lu/ist

[3] A. Oguz, A. T. Campbell, M. E. Kounavis, and R. F. Liao, "The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking," *IEEE Pesronal Communications Magazine, Special Issue on Adapting to Network and Client Variability*, 1998.

[4] G. Bochmann and A. Hafid, "Some Principles for Quality of Service Management," Tech. Rep., 1996.

[5] V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian, "A quality of service management framework based on user expectations," *First International Conference on Service Oriented Computing (ICSOC), Trento, Italy*, December 2003.

[6] A. Sahai, S. Graupner, V. Machiraju, and A. Moorsel, "Specifying and Monitoring Guarantees in Commercial Grids through SLA," *Proceedings of the 3rd IEEE/ACM CCGrid2003*, 2003.

[7] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu, "Agreement-based Grid Service Management (OGSI-Agreement)," *Grid Forum, GRAAP-WG Author Contribution Draft*, June 2003.

[8] K. Keahey and K. Motawi, "The Taming of the Grid: Virtual Application Service," *Argonne National Laboratory Technical Memorandum No. 262*, May 2003.

[9] L. Burchard, M. Hovestadt, O. Kao, A. Keller, and B. Linnert, "The virtual resource manager: An architecture for sla-aware resource management," *In Proceedings of IEEE CCGrid 2004, Chicago, US*, 2004.

[10] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, "A distributed resource management architecture that supports advance reservation and co-allocation," *In Proceedings of the International Workshop on Quality of Service*, pp. 27–36, 1999.

[11] R. Al-Ali, A. Hafid, O. Rana, and D. Walker, "An Approach for QoS Adaptation in Service-Oriented Grids," *Concurrency and Computation: Practice and Experience Journal*, vol. 16, no. 5, pp. 401–412, 2004.

[12] R. Nou, F. Julià, J. Guitart, and J. Torres, "Dynamic resource provisioning for self-adaptive heterogeneous workloads in smp hosting platforms," *ICE-B 2007, International Conference on E-Business, Barcelona, Spain*, July 2007.

[13] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing performance isolation accross virtual machines in xen," *Middleware 2006, Melbourne, Australia*, Nov. 27 - Dec. 1 2006.

[14] Web Services Agreement specification. [Online]. Available: http://www.ogf.org/documents/GFD.107.pdf

[15] L. Joita, O. F. Rana, P. Chacín, I. Chao, F. Freitag, L. Navarro, and O. Ardaiz, "Application deployment using catallactic grid middleware," *Middleware for grid computing*, 2005.

[16] S. Kounev, R. Nou, and J. Torres, "Using QPN to add QoS to Grid Middleware," Universitat Politècnica de Catalunya, Tech. Rep., 2007.

[17] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "Swrl: A semantic web rule language combining owl and ruleml," W3C Member submission 21 may 2004, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/

[18] M. Baker and M. Grove, "A virtual registry for wide-area messaging," *IEEE International Conference on Cluster Computing*, September 2006.

[19] Job Submission Description Language (JSDL) Work Group. [Online]. Available: http://forge.gridforum.org/projects/jsdl-wg

[20] GridSAM, Grid Job Submission and Monitoring Web Service. [Online]. Available: http://gridsam.sourceforge.net/

[21] M. Baker and G. Smith, "Gridrm: an extensible resource monitoring system," *IEEE International Conference on Cluster Computing*, 2003.

[22] A. Eisenberg, J. Melton, K. Kulkarni, J. Michels, and F. Zemke, "Sql:2003 has been published," *SIGMOD Record*, vol. 33, no. 1, March 2004.

[23] S. Andreozzi, "Glue schema implementation for the ldap data model," Instituto Nazionale Di Fisica Nucleare, Tech. Rep., September 2004.