A Scalability and Performance Evaluation of a Distributed Usage SLA-based Broker in Large Grid Environments

Catalin Dumitrescu*, Ian Foster*+, Ioan Raicu*

*Computer Science Department
The University of Chicago
{cldumitr,iraicu}@cs.uchicago.edu

+Mathematics and Computer Science Division
Argonne National Laboratory
foster@mcs.anl.gov

Abstract

Managing usage SLAs within large environments that integrate participants and resources spanning multiple physical institutions is a challenging problem. Maintaining a single unified usage SLA management decision point over hundreds to thousands of jobs and sites can quickly become a problem in terms of reliability as well as performance. Previous work developed GRUBER, a distributed grid Usage SLA-based resource broker that allows multiple decision points to coexist and cooperate in real-time. GRUBER ultimately addresses issues regarding how usage SLAs can be stored, retrieved and disseminated efficiently in a large distributed environment. The key question this paper addresses is the scalability and performance of GRUBER in large Grid environments. We conclude that as little as three GRUBER decision points could be enough in an environment with 300 sites and 60 VOs, an environment ten times larger than today's Grid3.

1. Introduction

The motivating scenario our work addresses consists of *providers* wishing to grant *consumers* the right to use certain *resources* for some agreed-upon time period in a large grid environment. Providers might be companies providing outsourcing services, or scientific laboratories that provide different scientific collaborations with access to their computers or other resources.

Providers and consumers may be nested: a provider may function as a middleman, providing access to resources to which the provider has itself been granted access by some other provider. Usage SLA issues can arise at multiple levels in such scenarios. Providers want to express (and enforce) the SLAs under which resources are made available to consumers. Consumers want to access and interpret SLA statements published by providers, in order to monitor their agreements and guide their activities. Both providers and consumers want to verify that SLAs are applied correctly. In summary, we address a technique for constructing a scalable management service with support for usage SLA expression, publication, discovery, interpretation, enforcement, and verification [1]. This problem encompasses challenging and interrelated scheduling, information synchronizations and scalability issues. We build on previous work concerning the specification and enforcement of local resource scheduling policies [2,3,4,5,6], the GRUBER broker [7,8,9,25], and the scalability and performance measurements of various grid services [13]. GRUBER ultimately addresses issues regarding how SLAs can be stored, retrieved and disseminated efficiently in a large distributed environment. *The key question this paper addresses is the scalability and performance of GRUBER in large Grid environments.*

The rest of this article is organized as follows. We first provide a more detailed description of the problem that we address. We then discuss the background information and related work. Section 3 describes succinctly the initial model for resource and workload management, namely GRUBER. The rest of the paper focuses on the problem of constructing a scalable framework infrastructure for usage SLA management, with section 4 covering our experimental results, and finally, with our conclusions.

1.1. Problem Statement

This work targets grids that may comprise of hundreds of institutions and thousands of individual investigators that collectively control tens or hundreds of thousands of computers and associated storage systems [11,12]. Each individual investigator and institution may participate in, and contribute resources to, multiple collaborative projects that can vary widely in scale, lifetime, and formality. At one end of the spectrum, two collaborating scientists may want to pool resources for the purposes of a single analysis. At

the other extreme, the major physics collaborations associated with the Large Hadron Collider encompass thousands of physicists at hundreds of institutions, and need to manage workloads comprising dynamic mixes of work of varying priority, some requiring the efficient aggregation of large quantities of computing and storage.

In this paper we focus on techniques for constructing a scalable service and measure its performance. It is important to understand the problems we face in order to come up with appropriate solutions. We initially focus on performance issues and on service reliability. In the end, we also discuss the problem of privacy issues for usage SLA at the VO level and beyond.

1.2. Performance Issues

We have performed several experiments using DiPerF, a distributed performance-testing framework designed to simplify and automate service performance evaluation [13]. DiPerF coordinates a pool of machines that test a target service, collects and aggregates performance metrics, and generates performance statistics. The aggregate data collected provide information on a service throughput, on the service 'fairness' when serving multiple clients concurrently, and on the impact of network latency on service performance.

We have used DiPerF to perform tests on service instance creation in a GT3 service (similar to the GRUBER implementation), and found a peak throughput of about 14 requests per second. Average service response time under 'normal' load was about 4s. Average service response time under 'heavy' load was about 10 seconds. We also observed that under heavy load the WS service does not allocate resources evenly among clients [13]. The results of

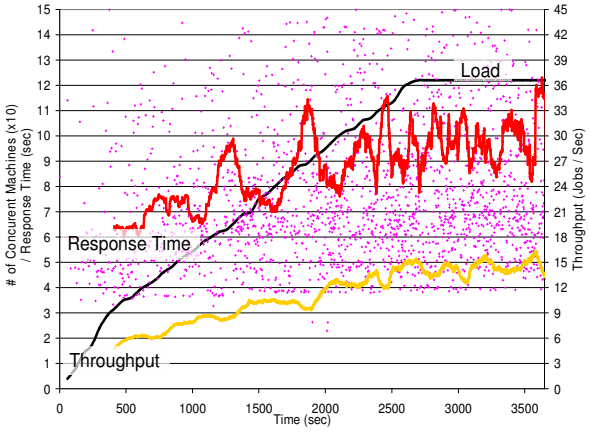

Figure 1 show how the service response time increases with the number of concurrent machines. As a consequence, there is a real need to investigate other ways of building and organizing a scheduling infrastructure for large grids with many submitting hosts, and to understand the implications this has for performance, reliability and scheduling decision accuracy.

GRUBER bypasses the OGSA service instance creation latency by providing the possibility of using generic and long term clients. One client is enough to handle all jobs submissions from at least six submission hosts. Unfortunately, in a large environment with many submission hosts, the performance problem still remains due to the high number of GRUBER clients that can run in parallel.

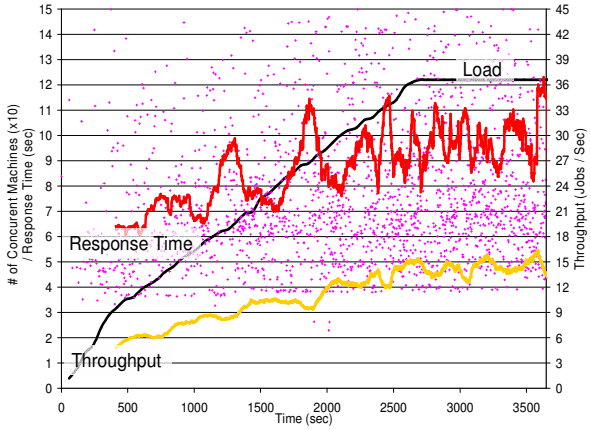

Figure 1: GT3.2 Service Instance Creation: Response time, Throughput, and Load

1.3. Service Reliability Issues

Another problem faced often in large distributed environments is the reliability and availability of a service. The interactions place undue burden on usage SLA service providers, who are forced to implement the SLA decision points for various communities. Furthermore, many interactions might be required as every usage SLA alteration can require interaction with the responsible decision point (also known as a VO policy enforcement point). As a consequence, the cost of maintaining a usage SLA and scheduling infrastructure should not increase with the number of resource providers and VOs participating in resource sharing actions. Administration issues must be bounded, essentially making the complexity proportional with the number of VOs and resource providers and not the size of the VOs [10].

Pearlman et al [10] address the scalability problem by introducing a third party, a *community authorization service* (CAS) that is responsible for managing the SLAs that govern access to a community's resources. While such an approach is

suited from an authorization purpose, it does not match the problem of resource scheduling and continuous usage SLA enforcement when resources are over-provided or contention occurs under more complex scenarios [1]. On the other hand, we approach the problem differently, leveraging from a peer-to-peer approach [14], where only partial information is exchanged among various decision points.

1.4. Privacy for Usage SLA Issues

Usage SLA specification, enforcement, negotiation, and verification mechanisms arise at multiple levels within VO-based environments. Resource providers want to establish, modify, enforce, and instrument usage SLAs concerning how their resources are made available to different participants and/or for different purposes.

In certain cases, users can require various privacy issues for the availability of information about their work (job types and priorities, data movement and characteristics). Thus, the maintenance of a *private* broker could be a better solution in such a situation. This issue can be encountered from the VO level on down to the individual scientists. The problem becomes even more sensible when dealing with commercial entities that have specific SLAs about their information [15]. Privacy for usage SLA issues is outside the scope of this paper; however it is nevertheless a very important topic for usage SLAs.

2. Background Information

2.1. Related Work

The Maui scheduler [4] for clusters and supercomputers is capable of enforcing complex SLA-driven scheduling schemas. It operates as a SLA engine for controlling resource allocations to jobs, and concurrently optimizes the use of managed resources. The scheduler manipulates several kinds of objects: jobs, nodes, reservations, QoS structures, policies, and composite objects.

The fair share scheduling strategies which were studied in depth in the early 1980s in the context of mainframe batch and timeshared systems, and then brought into the UNIX environment, represent another important body of work related to our problem. The purpose of fair share scheduling is to control resource distribution to allow greater predictability in process execution process [20,21].

In et al. [22] proposed a framework for policy based scheduling of grid-enabled resource allocations. The framework provides scheduling strategies that (a) control the request assignment to grid resources by adjusting resource usage accounts or request priorities; (b) manage efficiently resources assigning usage quotas to intended users; and (c) supports reservation based grid resource allocation. This framework is incorporated as part of the SPHINX scheduling system from University of Florida. The difference with our approach consists in the fact that we do not assume a centralized point of usage SLA specification.

The Grid Service Broker, developed as part of the GridBus Project, mediates access to distributed resources by (a) discovering suitable data sources for a given analysis scenario, (b) suitable computational resources, (c) optimally mapping analysis jobs to resources, (d) deploying and monitoring job execution on selected resources, (e) accessing data from local or remote data source during job execution, and (f) collating and presenting results. The broker supports a declarative and dynamic parametric programming model for creating grid applications [23]. GridBus targets a higher degree of details about available resources (machine level), jobs and files compared to GRUBER, which inherently makes GRUBER more scalable in large environments. Also, GridBus does not take in account the notions of sites, submission hosts, and virtual organizations, groups or priorities associated with them.

Cremona is a project developed at IBM as a part of the ETTK framework [9]. It is an implementation of the WS-Agreement specification and its architecture separates multiple layers of agreement management, orthogonal to the agreement management functions: the Agreement Protocol Role Management, the Agreement Service Role Management, and the Strategic Agreement Management. Cremona focuses on advance reservations, automated SLA negotiation and verification, as well as advanced agreement management. GRUBER instead targets a completely different environment model, where the main players are VO and resource providers. They also base their actions on usage SLAs and a more opportunistic environment where free CPUs are acquired when available. Furthermore, GRUBER introduces the notion of adaptive usage SLAs, considering sites autonomous, adjusting to instantaneous sharing policies without triggering SLA violations when detected.

2.2. DiPerF

Since we used the DiPerF framework to perform the performance and scalability experiments in this paper, we decided to give an overview of DiPerF to outline its features.

DiPerF coordinates several machines in executing a performance service client and collects various metrics about the performance of the tested service. The framework is composed of a controller/collector, several submitter modules and a tester component. The tester is responsible for running remotely the service performance measurement clients and to report the results back. The controller is responsible for performing all the aggregation operations required and presented in this paper [13]. Figure 2 depicts the various components of the DiPerF framework and the relationship among the controller, the tester, the client code, and the service.

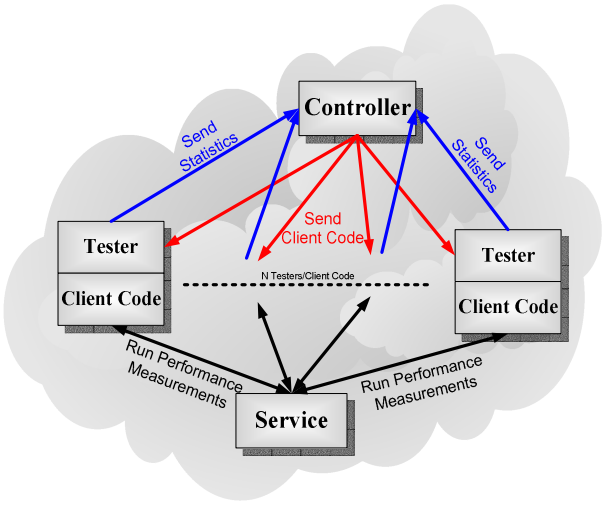

Figure 2: DiPerF framework overview

The client code distribution is automated as well as performance metric collection and result computation. Due to the time error rates seen in practice, DiPerF handles the time synchronization with a centralized time-stamp server that allows a time mapping to a common base; with a common unified global time, the framework is able to do metric aggregation accurately. Figure 3 depicts the aggregate view of the controller after the time has been synchronized at all testers. Additionally, due to the controlled delay in starting clients, DiPerF is able to report the maximum throughput a service supports as well as service response time as the load on the service varies.

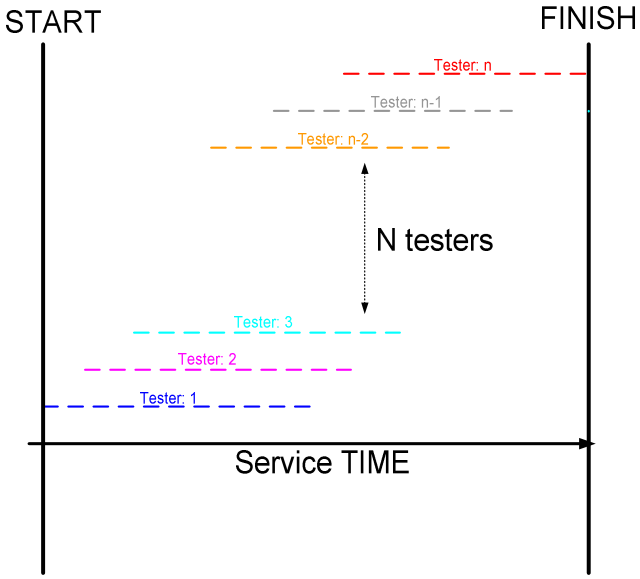

Figure 3: Aggregate view at the controller

Originally, DiPerF only supported the testing of a single service. An important improvement done to DiPerF for this set of experiments was the ability to test distributed services, such as the GRUBER deployment presented in Figure 4. Practically, each tester instantiates a GRUBER client with a specific GRUBER engine address for access, while later uses the site selectors and other GRUBER specific tools for actual job submission.

3. Usage SLA Enforcement Details

We consider important to overview the problem first at smaller sizes [1] and second to describe and analyze our solution for a scalable usage SLA management service.

3.1. Usage SLA Enforcement Model

The environment model which we used for our evaluation and experimentation is depicted in Figure 4 [16,17]. The main elements of our work are the *decision points* (previously known as policy enforcement points or PEPs), which are responsible for executing SLAs. They gather monitoring metrics and other information relevant to their operations, and then use this information to steer resource allocations as specified by the usage SLAs [1].

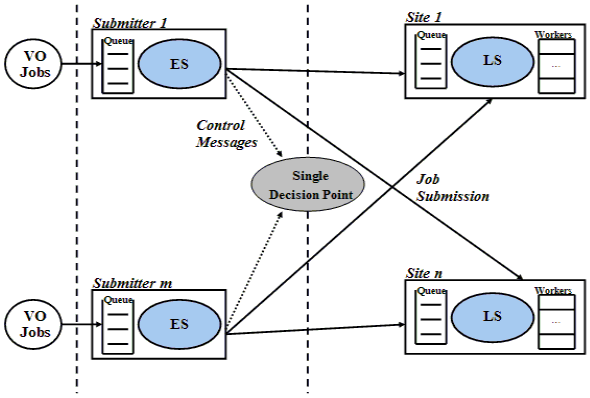

Figure 4: VO-Level Architecture

We distinguish between two types of PEPs. *Site policy enforcement points* (S-PEPs) reside at all sites and enforce site-specific policies. In our experiments, we did not take S-PEPs into consideration as they were outside the scope of this paper, and assumed the decision points have a total control over scheduling decisions.

VO policy enforcement points (decision points), associated with VOs, operate in a similar way to S-PEPs. They make decisions on a per-job basis to enforce usage SLAs regarding VO specifications for resource allocations to VO groups or to types of work executed by the VO. Decision points are invoked when VO planners make job planning and scheduling decisions to select which jobs to run, when to send them to a site scheduler, and which sites to run them at. Decision points interact with S-PEPs and schedulers to enforce VO-level SLA specifications.

We have already developed GRUBER [24,25], a prototype Grid decision point and S-PEP infrastructure that implements the usage SLA management model introduced before. GRUBER is composed of four principal components, the Gruber engine, the GRUBER site components, the GRUBER site selectors, and the GRUBER queue manager. In the GRUBER prototype, usage SLAs are specified through a specialized interface. The main components that we concentrate our performance measurements are the engine and the site selectors, as they are the main elements in providing adequate scheduling decisions when resources are available.

3.2. Information Dissemination Strategies

An important issue for a decentralized service is how usage SLAs and resource statuses are disseminated among components. The complexity is higher than for a file replica catalog, due to the necessity to correctly aggregated partial information gathered at several points; without the correct aggregation of the partial information, wrong decisions could generate workload starvations and resource under-utilization.

The first approach could allow both resource utilizations and usage SLAs to be exchanged among deployed decision points. The second approach might only allow utilizations to be exchanged among decision points. As possible variations on these two approaches, whenever new sites are detected in the exchanged data, their status is incorporated locally, assuming that each decision point has only a partial view of the environment. The third approach is one in which no usage information is exchanged and each decision point relies only on its own mechanisms for detecting the environment status, assuming the capacity to acquire global information about the environment.

For the experiments in this paper, we focused on the second approach with the assumption that each decision point has full "static" knowledge about available resources for its users, but not the latest resource utilizations. Practically, each decision point relies on information exchanges for updating only its view on the current utilizations. Another advantage of this approach is that it simplifies the implementation greatly, by avoiding the tracking of each usage SLA and allocation apparition time, as well as the entity to which it applies.

4. Experimental Results

4.1. Architecture Analyses

This section describes the performance analysis study we conducted to evaluate various grid-wide resource allocation models. In particular, we wanted to determine whether CPU resources could be allocated in a fair manner across multiple VOs, and multiple groups within a VO, without requiring the centralized control that is impractical in large grid environments. The way in which information is communicated between various decision points has a critical influence on the way that information is used.

4.1.1. Evaluated Scenarios

Our study evaluates several ranges of service distributions, from a single decision point, to many multiple decision points. Figure 5 depicts in a generic way the layout of the scenarios we use for our performance measurements.

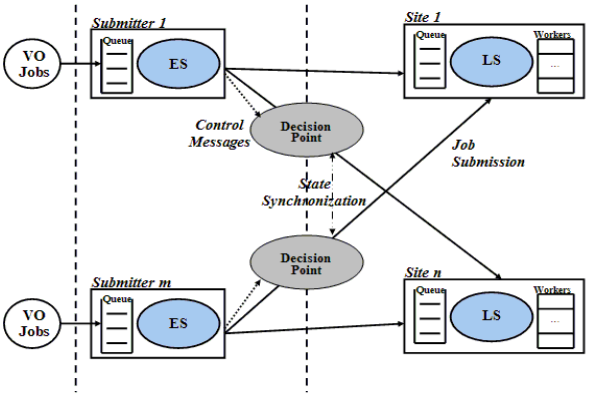

Figure 5: Multiple Unspecialized Decision Points

In either the single or multiple decision point (GRUBER) scenario, each decision point maintains a full view of the resource usages and allocations in the environment, by site monitoring and state exchanges among decision points (VO level queues, group level queues, and local usage SLAs).

We consider that for the scenario of one decision point per VO, the performance and reliability of the service is very similar to the case of *generic* decision points, and therefore we did not perform any additional tests for this case.

4.2. Empirical Results

We performed our tests on PlanetLab by deploying GRUBER decision points on several nodes. The complexity of the brokers' network used in this set of tests is specifically low in order to provide a simple case scenario that can easily be followed and later used.

GRUBER Infrastructure: We used one to ten GRUBER decision points deployed on machines around the world (mostly US and Europe). Each decision point maintained a view of the configuration of "simulated" global environment, while exchanging information about instantaneous utilizations.

Workloads and Environment: We used composite continuous workloads that overlay work for 60 VOs and 10 VO groups. The submission interval was one hour in all cases. Each submission site scheduled randomly one job to one GRUBER decision point on behalf of a pair (VO, group). The environment was composed of 300 sites (ten times larger than what Grid3 is today). The initial configurations were based on the Grid3 configurations.

***Usage SLA*:** For each virtual site, we used a similar approach in usage SLA specification by using the Grid3 policies as the initial configurations.

***Job states*:** The workload executions are based on a model where jobs pass through four states: 1) submitted by a user to a submission host; 2) submitted by a submission host to a site, but queued or held; 3) running at a site; and 4) completed.

***Evaluation Criteria / Measured metrics*:** We consider several metrics to evaluate the effectiveness of GRUBER in practice. We evaluate the effectiveness of different decision point deployment infrastructures by measuring *Average Response Time* (Response), *Average Throughput* (Throughput), *Queue Time* (QTime), *Average Resource Utilization* (Util), and *Average Scheduling Accuracy* (Accuracy) as a function of the environment complexity. All metrics are important as a good infrastructure will maximize delivered resources and meet owner intents.

We define **Response** as follows, with RT_i being the individual job time response:

$$\textbf{Response} = \Sigma_{i=1..N}\, \textbf{RT}_i \,/\, \textbf{N}$$

Throughput is defined as the number of requests completed successfully by the service averaged over a short time interval (per second or minute). We used in our measurements a second as the interval of measure.

We define **QTime** for an entire VO as follows, with RT_i being the individual job time response:

$$\textbf{QTime} = \Sigma_{i=1..N}\, \textbf{RT}_i \,/\, \textbf{N}$$

The difference between **Delay** and **QTime** is in their focus on different elements. While **Delay** measures the service responsiveness, **QTime** measures how fast a job is placed in execution after scheduling and provides a measure of the service accuracy in providing *good* scheduling decisions.

We define **Util** as the ratio of the CPU-resource actually consumed by users (ET_i) to the total CPU-resources available. We compute this quantity as follows:

$$\textbf{Util} = \Sigma\, (\textbf{ET}_i) \,/\, (\#_{\textbf{cpus}} * \Delta \textbf{t})$$

The last metric used for analysis is **Accuracy**. We first define scheduling accuracy (SAi) as the ratio of percentage of free resources at the selected site to the total percentage of free resources over the entire grid. Furthermore, we introduce **Accuracy** as the aggregated value of all scheduling accuracies measured for each individual job (where N is the total number of jobs scheduled in the considered period):

$$\textbf{Accuracy} = \Sigma_{i=1..N}\, (\textbf{SA}_i) \,/\, \textbf{N}$$

4.2.1. Infrastructure Scalability

By means of DiPerF we varied slowly the number of clients for our performance and scalability study. Figures below present **Response**, **Throughput** and **Load** as measured by DiPerF. The overall improvement in terms of throughput and response time is two to three times when a three-decision point infrastructure is deployed, while for the ten-decision point infrastructure the throughput increased almost five times.

The results of show service capacity decreases with the number of concurrent machines. **Throughput** increases immediately but does not go over a value of two jobs per second when all testing machines (120) are accessing the service in parallel. We also note the service response time decreases, once the number of machines decreases.

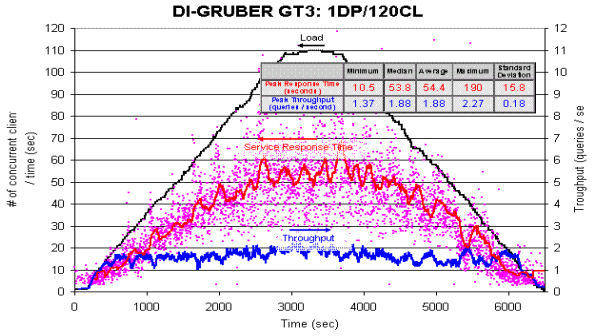

Figure 6: Centralized Scheduling Service

The results in show service capacity decreasing with the number of concurrent machines later. **Throughput** increases slowly and achieves a value of 6 job scheduling requests per second when all testing machines are accessing the service. **Response** is also smaller in average compared with the previous results. Once the number of machines starts decreasing, **Response** does not decrease in a symmetrical way, the service remaining somehow in a state that impedes it to answer as quickly as before. An explanation for this behavior is the fact that once jobs were scheduled for execution, they were considered long term running jobs. Each job was considered running for a longer time interval than the entire test time and the grid state was different (monotonically decreasing) in terms of free resources with each scheduled job.

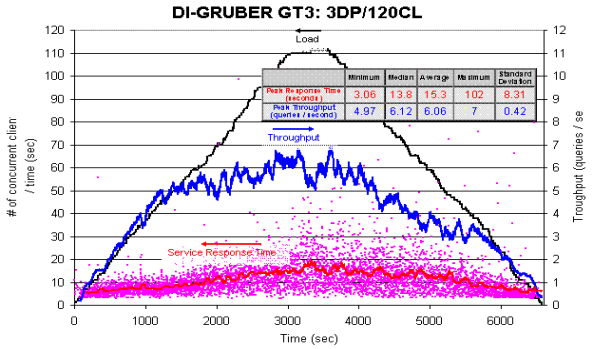

Figure 7: Distributed Scheduling Service with Three Decision Points

The results in are somehow different. The distributed service provides a symmetrical behavior with the number of concurrent machines independent of the state of the grid (lightly or heavily loaded). This result verifies the intuition that for a certain grid configuration size, there is an appropriate number of decision points that can serve the scheduling purposes under an appropriate performance constraint. The throughput achieved increases linearly with the number of decision points.

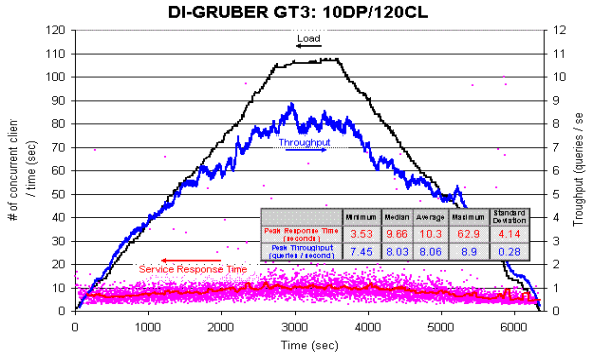

Figure 8: Distributed Scheduling Service with Ten Decision Points

While the performance of a service in answering queries is important, the accuracy of a distributed service in providing accurate scheduling decisions is even more important. In the next two sub-sections, we analyze the performance of our decision point's implementation, namely GRUBER, and the supporting algorithms in providing accurate scheduling decisions with both infrastructure complexity and synchronization interval between decision points.

4.2.2. Infrastructure Performance

For scheduling decision performance analyses, we use three of the metrics introduced before, **QTime**, **Util** and **Accuracy**. In addition, we provide the total

number of requested operations and the actual total number of served operations.

As a side note, these metrics are even more important in our context because the GRUBER site selectors are able to provide a random solution whenever they are unable to contact a decision point service. Therefore, measuring the accuracy of the scheduling decisions has a higher weight than the previous analysis in order to capture what happens when the infrastructure is overloaded or information exchanges are not performed fast enough. As a reminder, in all cases status information was exchanged every ten minutes.

Table 1 depicts the overall performance of GRUBER in diverse scenarios. While the values under the "All Requests" section provide an overall view of GRUBER's performance, they do not reflect the actual performance in scheduling jobs when the scheduling workload is adaptable to the system capacity.

Table 1: Performance of decision points

	Decision Points	% of Req	# of Req	QTime	Norm. QTime	Util	Accuracy
Requests Handled by GRUBER	1	40%	8673	0	0.000	3%	99%
	3	53%	27486	921	0.033	24%	91%
	10	67%	37641	2405	0.063	33%	80%
Requests NOT Handled by GRUBER	1	60%	13009	0	0.000	2%	-
	3	47%	23507	993	0.042	27%	-
	10	33%	18391	2080	0.113	23%	-
All Requests	1	100%	21682	256	0.513	5%	84%
	3	100%	50993	5727	0.233	51%	63%
	10	100%	56032	7126	0.269	56%	60%

If we consider only jobs that were scheduled through one of the GRUBER decision points, the results look rather different. There are four notable differences when comparing the performance between the requests handled and those that were not handled by GRUBER; 1) the accuracy shows significant improvement; 2) higher resource utilization when taking into consideration the percentage of requests handled by GRUBER; 3) the QTime is orders of magnitude better; and 4) the Normalized QTime is noticeably improved between requests not handled by GRUBER and all the requests. It is interesting to note that the scenario with only 1 decision point has a very small QTime; this is due to the fact that within the 1 hour the tests were performed, the number of requests made was smaller than in the other cases due to lower throughput. With less resources being used, it was easier for the decision point to make good decisions, and hence we small QTime. We computed the normalized QTime in order to take into account both the number of requests and the resource utilization; we discovered that the deceivingly low QTime for the 1 decision point scenario now shows its worse performance when compared to the other two scenarios.

4.2.3. Scheduling Performance with Synchronization

The other important dimension in our analysis is the synchronization interval among decision points. We performed several tests using DiPerF, where the decision points were exchanging status information at predefined time intervals, namely 1, 3, 10, and 30 minute intervals. The results are presented in Figure 9. It is important to note that the higher frequency of information exchanges has also a negative draw-back for the GRUBER decision points, namely the lower percentage of jobs handled by GRUBER. Practically, each decision point spends some of its resources on sending and receiving other data without necessarily improving the performance in performing scheduling operations. According to our observations, for a three decision point infrastructure a 3 to 10 minutes exchange interval should be sufficient for achieving a 95% Accuracy.

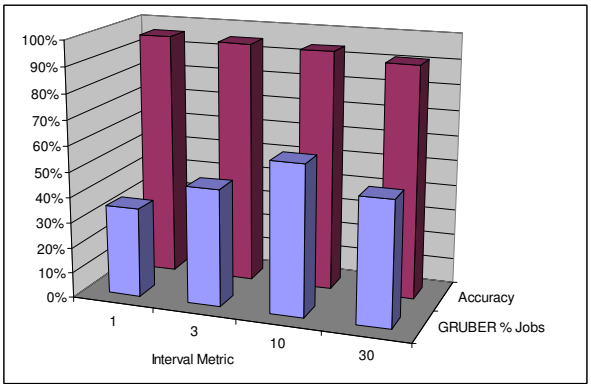

Figure 9: Accuracy and percentage of jobs handled by GRUBER as a function of the Exchange Time Interval

5. Conclusions and Future Work

Managing usage SLAs within large virtual organizations (VOs) that integrate participants and resources spanning multiple physical institutions is a challenging problem. Maintaining a single unified decision point for usage SLA management is a problem that arises when many users and sites need to be managed. We provide a solution, namely GRUBER, to address the question on how SLAs can be stored, retrieved and disseminated efficiently in a large distributed environment. *The key question this paper addressed was the scalability and performance of GRUBER in large Grid environments.* We evaluated a distributed architecture and SLA model for scheduling resources in large grid environments while satisfying resource owner and VO SLAs.

We achieved results in two dimensions – how well our proposed solution performed in practice and how to measure the success of GRUBER. We also introduced an enhancement to our GRUBER framework, namely the distributed approach in resource scheduling and usage SLA management.

There are certain issues that we did not address in this paper. For instance, our analysis did not consider certain methods of information dissemination among decision points. Furthermore, validating our results would involve performing tests on a considerably larger grid than exists today in practice.

In future work, we plan to perform a more extensive performance study in a wider range of scenarios and information dissemination strategies.

Acknowledgements: We thank Robert Gardner, Ruth Pordes, and Ian Fisk for insights, discussions, and support. We also thank the Grid3 project. Michael Wilde, Jens Voeckler, James Dobson, and Luiz Meyer provided code examples and technical support that made the results presented here possible. This work was supported by the NSF Information Technology Research GriPhyN project, under contract ITR-0086044.

6. References

1. Dumitrescu, C. and I. Foster, "Usage Policy-based CPU Sharing in Virtual Organizations", in *5th International Workshop in Grid Computing*, 2004, Pittsburg, PA.
2. Condor Project, *Condor-G*, www.cs.wisc.edu/condor/, 2002.
3. Altair Grid Technologies, LLC, *A Batching Queuing System, Software Project*, Software Project, 2003.
4. Platform Computing Corporation, *Administrator's Guide, Version 4.1*. February 2001.
5. Cluster Resources, Inc., *Maui Scheduler*, Software Project, 2001-2005.
6. Foster, I., et al., "End-to-End Quality of Service for High-end Applications", Computer Communications, 2004. 27 (14): p. 1375-1388.
7. S. Tuecke, et al., "Grid Service Specification".
8. Dan, A., C. Dumitrescu, and M. Ripeanu, "Connecting Client Objectives with Resource Capabilities: An Essential Component for Grid Service Management Infrastructures", in *ACM International Conference on Service Oriented Computing (ICSOC'04)*. 2004. New York.
9. Ludwig, H., A. Dan, and B. Kearney, "Cremona: An Architecture and Library for Creation and Monitoring WS-Agreements", in *ACM International Conference on Service Oriented Computing (ICSOC'04)*. 2004. New York.
10. Pearlman, L., et al., "A Community Authorization Service for Group Collaboration", in *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*. 2002.
11. Avery, P. and I. Foster, "The GriPhyN Project: Towards Petascale Virtual Data Grids", 2001.
12. Chervenak, A., et al., "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets", J. Network and Computer Applications, 2001(23): p. 187-200.
13. Dumitrescu, C., et al., "DiPerF: Automated DIstributed PERformance testing Framework", in *5th International Workshop in Grid Computing*, 2004, Pittsburg, PA.
14. Ripeanu, M. and I. Foster., "A Decentralized, Adaptive, Replica Location Service", in *11th IEEE International Symposium on High Performance Distributed Computing*. 2002. Edinburgh, Scotland: IEEE Computer Society Press.
15. Gimpel, H., et al., "PANDA: Specifying Policies for Automated Negotiations of Service Contracts", in *the 1st International Conference on Service Oriented Computing*. 2003. Trento, Italy.
16. Ranganathan, K. and I. Foster, "Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids", Journal of Grid Computing, 2003, 1 (1).
17. Ranganathan, K. and I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications", in *11th IEEE International Symposium on High Performance Distributed Computing*. 2002. Edinburgh, Scotland: IEEE Computer Society Press.
18. Foster, I., et al., "The Grid2003 Production Grid: Principles and Practice", in *13th International Symposium on High Performance Distributed Computing*. 2004.
19. Legrand, I.C., et al., "MonALISA: A Distributed Monitoring Service Architecture", in *Computing in High Energy Physics*. 2003. La Jolla, CA.
20. Kay, J. and P. Lauder, "A Fair Share Scheduler", University of Sydney, AT&T Bell Labs, 1998.
21. Henry, G.J., "A Fair Share Scheduler", AT&T Bell Laboratory Technical Journal, October 1984, 3 (8).
22. I In, J., P. Avery, R. Cavanaugh, and S. Ranka, "Policy Based Scheduling for Simple Quality of Service in Grid Computing", in *International Parallel & Distributed Processing Symposium (IPDPS)*. April '04. Santa Fe, New Mexico.
23. Buyya, R., *GridBus:* "A Economy-based Grid Resource Broker", The University of Melbourne: Melbourne, 2004.
24. Dumitrescu, C. and I. Foster, "GangSim: A Simulator for Grid Scheduling Studies", accepted for publication in *Cluster Computing and Grid (CCGrid),* Cardiff, UK, May 2005.
25. Dumitrescu, C., Foster, I., "GRUBER: A Grid Resource SLA Broker", GriPhyN/iVDGL Technical Report, 2005.