DI-GRUBER: A Distributed Approach to Grid Resource Brokering

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

*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 service level agreements (USLAs) within environments that integrate participants and resources spanning multiple physical institutions is a challenging problem. Maintaining a single unified USLA management decision point over hundreds to thousands of jobs and sites can become a bottleneck in terms of reliability as well as performance. DI-GRUBER, an extension to our GRUBER brokering framework, was developed as a distributed grid USLA-based resource broker that allows multiple decision points to coexist and cooperate in real-time. DI-GRUBER addresses issues regarding how USLAs can be stored, retrieved, and disseminated efficiently in a large distributed environment. The key question this paper addresses is the scalability and performance of DI-GRUBER in large Grid environments. We conclude that as little as three to five decision points can be sufficient in an environment with 300 sites and 60 VOs, an environment ten times larger than today's Open Science Grid.

1. Introduction

The motivating scenarios for our work are large grid environments in which *providers* wish to grant *consumers* the right to use certain *resources* for some agreed-upon time period. Providers might be companies providing outsourcing services, or scientific laboratories that provide different 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. USLA issues can arise at multiple levels in such scenarios. Providers want to express (and enforce) the USLAs under which resources are made available to consumers. Consumers want to access and interpret USLA statements published by providers, in order to monitor their agreements and guide their activities. Both providers and consumers want to verify that USLAs are applied correctly.

We present here a technique for constructing a scalable management service with support for USLA expression, publication, discovery, interpretation, enforcement, and verification [1]. This problem encompasses challenging and interrelated scheduling, information synchronization, 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 [1,25], and the scalability and performance measurements of various grid services [13]. GRUBER addresses issues regarding how USLAs can be stored, retrieved, and disseminated efficiently in a distributed environment. GRUBER has been implemented in both the Web Services (WS) and pre-WS versions of the Globus Toolkit (GT).

Here we introduce a two layer scheduling infrastructure, DI-GRUBER, capable of working over large grids. DI-GRUBER extends GRUBER by introducing support for multiple scheduling decision points, loosely synchronized via periodic information exchange. Our focus is on measuring both the capability and performance of such a framework, as well as gaining insights about the number of decision points required under a certain load.

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 about the tools used to perform these experiments, as well as the model used for USLA enforcement. Section 3 contains the description of the experiments, the results we achieved, and some improvements we consider necessary for our framework for dynamically re-configuring. Section 4 contains additional simulation results and enhancements proposed for future work. The rest of the paper focuses on related work and on our conclusions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SC|05 November 12-18, 2005, Seattle, Washington, USA
© 2005 ACM 1-59593-061-2/05/0011…$5.00

1.1. Problem Statement

This work targets grids that may comprise 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, requiring the efficient aggregation of large quantities of computing resources.

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. First, we investigate performance issues and service reliability. Then, we examine techniques for determining dynamically the number of decision points required for the large grid scenarios considered in this paper.

1.2. Performance Issues

How fast can a site selector service process requests? We address this question in detail later in the paper, but to provide some initial data we have performed several experiments using DiPerF, a distributed performance-testing framework designed to simplify and automate service performance evaluation (see Section 2.8).

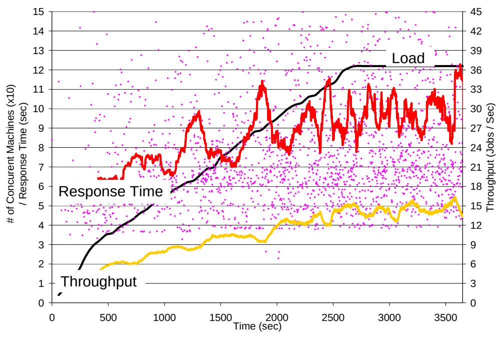


Figure 1: GT3.2 Service Instance Creation: Response time, Throughput, and Server Load

We use DiPerF to perform tests on service instance creation in a GT3 Java Web Service similar to the first GRUBER implementation as described in Section 2.2. We found a peak throughput of about 14 requests per second. (The factors limiting performance are primarily authentication and SOAP processing. While GT4 and the latest Apache Axis on which it builds provides significant improvements in both areas [29], limits will always remain.) Furthermore, service response time increases with load, from an average of about 4s under 'normal' load to about 10s under 'heavy' load.

Workloads on large grids such as the Open Science Grid (OSG: previously known as Grid3 [18]) require both higher job submission rates and reduced roundtrip times. Thus, we need to investigate other ways of building and organizing scheduling infrastructures for large grids with many submitting hosts, and to understand the implications of those alternative structures on performance and scheduling decision accuracy.

1.3. Service Reliability Issues

Another problem often encountered in large distributed environments concerns service reliability and availability. USLA service providers are subject to high load, due both the high request rates mentioned above and the need to support interactions relating to USLA modification. We cannot afford for this infrastructure to fail. The administrative costs of maintaining a USLA and scheduling infrastructure should not increase with the number of resource providers and VOs participating in resource sharing actions [10].

1.4. USLA Privacy Issues

Another problem faced in practice is the necessity for privacy when sensitive computing resources are shared. USLA specification, enforcement, negotiation, and verification mechanisms arise at multiple levels within VO-based environments. Resource providers want to establish, modify, enforce, and instrument USLAs 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 necessity in such a situation. This issue can be encountered from the VO level on down to individual users. The problem becomes even more sensible when dealing with commercial entities [15]. Privacy for USLAs is outside the scope of this paper, but is an important topic.

2. Background Information

We now introduce the main concepts and tools used in this paper.

2.1. USLA Enforcement Model

The environment model that we use in our work is depicted in Figure 2 [16,17]. Note in particular the *decision points* (also know as policy enforcement points or PEPs), which are responsible for executing USLAs. These components gather monitoring metrics and other information relevant to their operations, and then use this information to steer resource allocations as specified by the USLAs [1].

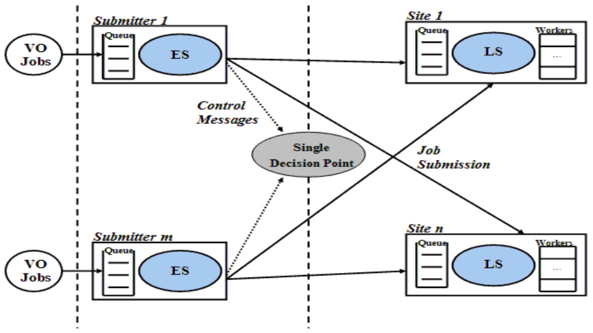


Figure 2: VO-Level Architecture

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 our scope, and assumed the decision points have 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 USLAs 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 USLA specifications.

2.2. DI-GRUBER Broker Decision Point

We have already developed GRUBER [25], a prototype Grid V-PEP and S-PEP infrastructure that implements the USLA management model introduced before. GRUBER is the main component that we used for the job scheduling over a hypothetical grid similar to Grid3 [18]. It is able to perform job scheduling based on notions such as VO, group VO, and USLAs at various levels. The main four principal components are described next.

The *GRUBER engine* is the main component of the architecture. It implements various algorithms for detecting available resources and maintains a generic view of resource utilization in the grid.

The *GRUBER site monitor* is a data provider for the GRUBER engine. This component is optional and can be replaced with various other grid monitoring components that provide similar information, such as MonaLisa or Grid Catalog.

A *GRUBER client* represents a standard GT client that allows communication with other GRUBER components and the GRUBER engine, such as the GRUBER site selectors that we introduce next.

GRUBER site selectors are tools that communicate with the GRUBER engine and provide answers to the question: "*which is the best site at which I can run this job?*". Site selectors can implement various task assignment policies, such as round robin, least used, or least recently used task assignment policies.

Finally, the *GRUBER queue manager* is a GRUBER client that resides on a submitting host. This component monitors VO policies and decides how many jobs to start and when. It interacts with the GRUBER engine to obtain site selection recommendations.

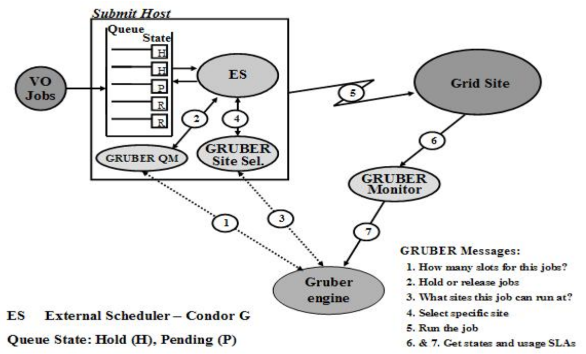


Figure 3: GRUBER Architecture

In the work reported here, we use the GRUBER engine and site selectors but not the queue manager. In this configuration, GRUBER is used only as a site recommender: it does not enforce VO-level USLAs, by for example removing a site for an already over-quota VO user at that site. In effect, we assume that all clients comply with the recommendations and that there is thus no need for enforcement.

GRUBER does not itself perform job submission, but as shown in Figure 3 can be used in conjunction with various grid job submission infrastructures. In the work we describe here, we interface with the Euryale tool.

2.3. GRUBER USLA Semantics

We review how USLAs are described. Much research from the web-service provisioning community deals with these issues in detail [5,6,8,9,10,15,23].

In the experiments described in this paper we use a USLA representation based on Maui semantics and WS-Agreement syntax [8,9,15]. Allocations are made for processor time, permanent storage, or network bandwidth resources, and there are at least two levels of resource assignments: to a VO, by a resource owner, and to a VO user or group, by a VO. We started from Maui semantics in providing support for fair-share rule specification [5]. Each entity has a fair share type and fair share percentage value, e.g., VO_0 15.5, VO_1 10.0+, VO_2 5.0-. The sign after the percentage indicates if the value is a target (no sign), upper limit (+), or lower limit (-).

We extended the semantics by associating both a consumer and a provider with each entry; extending the specification in a recursive way to VOs, groups; and users, and allowing more complex sharing rules as defined in the WS-Agreement. Further, we express allocations as WS-Agreement goals allowing the specification of rules with a finer granularity. We based our SLA specification on a subset of WS-Agreement, taking advantage of the refined specification and the high-level structure. We use a simple schema that allows for monitoring resources and goal specifications [5,8,9].

2.4. Euryale as Concrete Planner

Euryale [28] is a system designed to run jobs over large grids such as OSG [18]. Euryale uses Condor-G [2] (and thus the Globus Toolkit GRAM) to submit and monitor jobs at sites. It takes a late binding approach in assigning jobs to sites, meaning that site placement decisions are made immediately prior to running the job, rather than in an earlier planning phase. Euryale also implements a simple fault tolerance mechanism by means of job re-planning when a failure is discovered. We use the Euryale planner as our job submission tool and GRUBER interface.

A tool called DagMan executes the Euryale prescript and postscript. The prescript calls out to the external site selector (i.e., in our case, GRUBER) to identify the site on which the job should run, rewrites the job submit file to specify that site, transfers necessary input files to that site, registers transferred files with the replica mechanism, and deals with re-planning. The postscript file transfers output files to the collection area, registers produced files, checks on successful job execution, and updates file popularity.

2.5. Information Dissemination Strategies

An important issue for a decentralized brokering service is how USLAs and usage information are disseminated among decision points. We need to aggregate correctly partial information gathered at several points; without a correct aggregation of the partial information, wrong decisions can result in workload starvation and resource under-utilization.

This problem can be addressed in several ways. In a first approach, both resource usage information and USLAs are exchanged among decision points. In a second approach, only utilization information is exchanged. As possible variations on these two approaches, whenever new sites are detected, their status is incorporated locally, which means that each decision point has only a partial view of the environment. In a third approach, no usage information is exchanged and each decision point relies only on its own mechanisms for detecting grid status.

For the experiments in this paper, we focus on the second approach and the assumption that each decision point has complete "static" knowledge about available resources, but not the latest resource utilizations. An advantage of this approach is the simplified implementation by avoiding USLA tracking.

2.6. Open Science Grid

Open Science Grid (OSG: previously known as Grid3 [18]) is a multi-virtual organization environment that sustains production level services required by various physics experiments. The infrastructure comprises more than 50 sites and 4500 CPUs, over 1300 simultaneous jobs and more than 2 TB/day aggregate data traffic. The participating sites are the main resource providers under various conditions. We consider in this paper an environment similar to OSG but ten times larger and with much higher rates of job scheduling [18]. GRUBER (and the DI-GRUBER enhancement) provides a USLA-based solution for job scheduling decisions for environments similar to OSG, by providing a means for informed site selection at the job level and beyond.

2.7. PlanetLab Testbed

PlanetLab [26,27] is a geographically distributed platform for deploying, evaluating, and accessing planetary-scale network services. PlanetLab is a shared community effort by a large international group of researchers, each of whom gets access to one or more isolated "slices" of PlanetLab's global resources via a concept called distributed virtualization. PlanetLab now comprised over 500 nodes (Linux-based PCs or servers connected to the PlanetLab overlay network) distributed worldwide. Almost all nodes are connected via 10 Mb/s network links (with 100Mb/s on several nodes), have processor speeds exceeding 1.0 GHz IA32 PIII class processor, and at least 512 MB RAM.

2.8. DiPerF

For all the experiments in this paper, we used the DiPerF tool, a distributed performance testing framework. 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 [13]. DiPerF was originally designed for testing a single point service. For the experiments reported here, we extended it to enable testing of distributed services such as DI-GRUBER. Simply, each tester instantiates a client connected to a single DI-GRUBER decision point. When scheduling a job, each such client interacts with its DI-GRUBER decision point to obtain site load information, and then executes site selector logic to determine the site to which the job should be dispatched. DiPerF allowed us to concentrate on the performance and scalability of DI-GRUBER rather than on how to perform large scale testing involving 100+ clients.

3. Empirical Results

In this section, we focus on measuring several characteristics of the DI-GRUBER implementation in a large environment. We present results for both GT3 and GT4 implementations of DI-GRUBER. Due to the fact that we do not have access to a large enough grid, we emulated the entire environment on PlanetLab [26,27].

3.1. Architecture Analysis

This section describes the performance analysis study we conducted to evaluate various DI-GRUBER configurations. In particular, we wanted to determine whether CPU resources could be allocated in a fair manner across multiple VOs, and across multiple groups within a VO, when using DI-GRUBER configurations that feature multiple loosely coupled GRUBER instances rather than a single centralized instance. The factors that we consider include both the number of GRUBER instances and the frequency of communication of state information among those decision points. Figure 4 depicts in a schematic way the layout of the scenarios we used for our performance measurements.

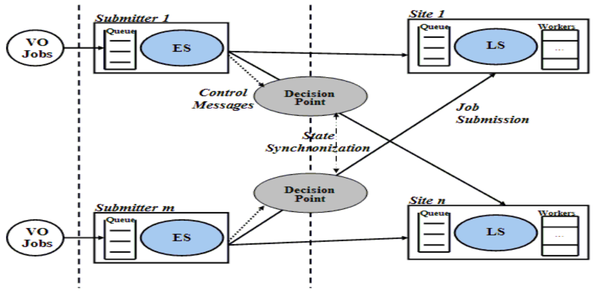


Figure 4: Multiple Decision Points

In either the single or multiple decision point scenario, each decision point maintains a full view of the resource usages and utilizations by monitoring scheduling decisions.

3.2. Performance Metrics

We use five metrics to evaluate the effectiveness of DI-GRUBER: *Average Response Time* (**Response**), *Average Throughput* (**Throughput**), *Queue Time* (**QTime**), *Average Resource Utilization* (**Util**), and *Average Scheduling Accuracy* (**Accuracy**). Each metric is important, as a good infrastructure will both maximize delivered resources and meet owner intents.

We define **Response** as follows, with RT_i being the individual job time response and N being the number of jobs processed during the execution period:

$$\mathbf{Response = \Sigma_{i=1..N}\, RT_i / N}$$

Throughput is defined as the number of requests completed successfully by the service per unit time.

We define **QTime** for an entire VO as follows, with QT_i being the individual job queue time, i.e., the time that elapses between the job being dispatched to a site and the job starting execution:

$$\mathbf{QTime = \Sigma_{i=1..N}\, QT_i / N}$$

Response and **QTime** focus on different elements. While **Response** measures the service responsiveness, **QTime** measures how fast a job is placed in execution after scheduling, and thus provides a more direct measure of the scheduling service's ability to *good* scheduling decisions.

We define **Util** as the ratio of the CPU time actually consumed by the N jobs executed during the period considered ($\Sigma\, ET_i$) to the total CPU time available over that time:

$$\mathbf{Util = \Sigma_{i=1..N}\, (ET_i) / (\#_{cpus} * \Delta t)}$$

Finally, we define the scheduling accuracy for a specific job (SA_i) as the ratio of free resources at the selected site to the total free resources over the entire grid. **Accuracy** is then the aggregated value of all scheduling accuracies measured for each individual job:

$$\textbf{Accuracy} = \Sigma_{i=1..N} (SA_i) / N$$

3.3. Experimental Environment

We used between one and ten GT3 DI-GRUBER decision points deployed on PlanetLab nodes [26,27]. Each decision point maintained a view of the configuration of the global DI-GRUBER environment, via the periodic exchange (in the experiments that follow, every three minutes) with other decision points of information about recent job dispatch operations. The decision points are connected in a mesh, a simple configuration that is adopted to simplify analysis and understanding.

We used composite workloads that overlay work for 60 VOs and 10 groups per VO. The experiment duration was one hour in all cases, and jobs were submitted every second from a submission host. Each of a total of about 120 submission hosts (“clients”) maintained a connection with only one DI-GRUBER decision point, selected randomly in the beginning—thus simulating a scenario in which each submission site is associated statically with a single decision point.

An important characteristic of our experimental architecture was that each client was configured to apply a 60s timeout to the requests that it dispatched to its designated DI-GRUBER decision point. If this timeout expires, the client’s site selector then selects a site at random, without considering USLAs. This strategy meant that site selection performance degraded gracefully in the event that a decision point reached a saturation state due to many requests in progress.

The emulated environment was composed of 300 sites representing 40,000 nodes (a grid approximately ten times larger than Grid3 today). Each site is composed of one or more clusters. The emulated configuration was based on Grid3 configuration settings in terms of CPU counts, network connectivity, etc.

We note that this emulated environment is already as big as some existing P2P networks. There are two layers of communication in this environment; the sites can be thought of as super-nodes from a P2P network, while the 40,000 nodes can be thought of as leaves from the P2P network. GT3 DI-GRUBER performance is determined primarily by the number of decision points used to answer queries, and not by the size of the environment in which it is deployed; therefore, we conclude that DI-GRUBER could support larger environments without a significant negative impact on its performance.

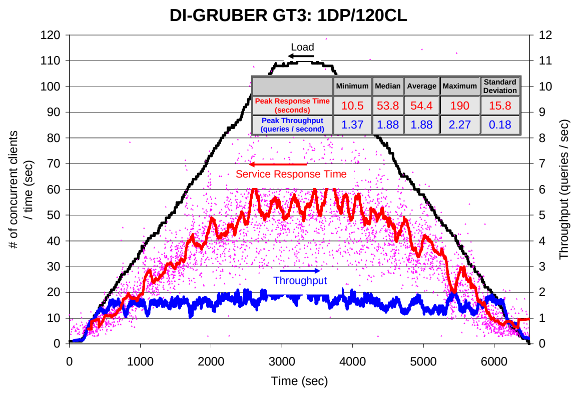
The workload executions are based on a model in which jobs pass through four states: 1) submitted by a user to a submission host; 2) submitted by a submission host to a site, but queued or held; 3) running at a site; and 4) completed.

3.4. GT3 DI-GRUBER Empirical Results

We now report the results of our PlanetLab experiments.

3.4.1. Infrastructure Scalability

We used DiPerF (described in Section 2.8) to vary slowly the participation of 110 clients. Figures 6-7 below present **Response** and **Throughput** as measured by DiPerF, as well as the number of active clients (**Load**), when using 1, 3, and 10 decision points, respectively. (Results presented in Section 4 suggest that performance gains obtained with more than 10 decision points would be marginal.)

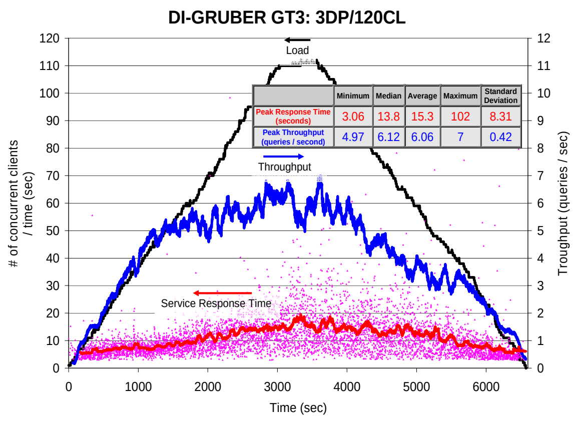


	Minimum	Median	Average	Maximum	Standard Deviation
Peak Response Time (seconds)	10.5	53.8	54.4	190	15.8
Peak Throughput (queries / second)	1.37	1.88	1.88	2.27	0.18

Figure 5: GT3 Centralized Scheduling Service

With one decision point (Figure 5), service response time increases steadily as the number of concurrent machines increases; during the peak period, the average service **Response** time was about 54 seconds. **Throughput** increases rapidly, but after about 15 concurrent clients, it plateaus at a little less than 2 queries per second; the throughput remains relatively constant at about 1.9 queries per second even when all 110 clients are accessing the service in parallel. (We note that a single GRUBER request involves several round trips, and the transport of significant state, as the site selector first requests information about current site availabilities and then informs the decision point about its site selection. Thus, the cost of a single "request" is considerably higher than in the simple case considered in Section 1.2.

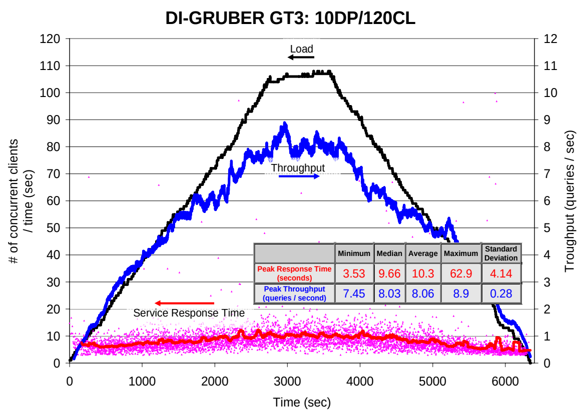
With three decision points (Figure 6), **Throughput** increases slowly to about 6 job scheduling requests per second when all testing machines are accessing the service in parallel. The service **Response** time is also smaller (about 15 seconds) on average, significantly less than with a single decision point (about 54 seconds).



	Minimum	Median	Average	Maximum	Standard Deviation
Peak Response Time (seconds)	3.06	13.8	15.3	102	8.31
Peak Throughput (queries / second)	4.97	6.12	6.06	7	0.42

Figure 6: GT3 DI-GRUBER, Three Decision Points

With 10 decision points (Figure 7), the average service **Response** time decreased even further to about 10 seconds, and the achieved **Throughput** reached about 8 queries per second during the peak load period.



	Minimum	Median	Average	Maximum	Standard Deviation
Peak Response Time (seconds)	3.53	9.66	10.3	62.9	4.14
Peak Throughput (queries / second)	7.45	8.03	8.06	8.9	0.28

Figure 7: GT3 DI-GRUBER, 10 Decision Points

The distributed service provides a symmetrical behavior with the number of concurrent machines that is 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 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 relative to the centralized approach.

3.4.2. Accuracy and Scheduling Performance

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. Thus, we analyze the performance of GT3 DI-GRUBER from the perspectives of achieved utilization, queue time, and accuracy.

Table 1 depicts overall GT3 DI-GRUBER performance for the three scenarios just discussed. We show not only **QTime**, **Util**, and **Accuracy**, but also the total number of operations requested by clients and the total number of operations "handled" by DI-GRUBER decision points. When the former number is greater than the latter, this means that DI-GRUBER decision points are becoming overloaded and timeouts are occurring, resulting in random site selection decisions. When timeouts occur, job submissions are delayed and thus the total number of job submissions is reduced during the time period considered—in addition to individual requests being scheduled less accurately.

While the values under the "All Requests" section provide an overall view of GT3 DI-GRUBER's performance, they do not reflect the job scheduling performance that would be achieved when the scheduling workload is adapted to the system capacity. The "Handled by GRUBER" data provide a better measure in that regard.

Table 1: GT3 DI-GRUBER Overall Performance

	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 DI-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 DI-GRUBER; 1) **Accuracy** shows significant improvement; 2) higher **Resource Utilization** when taking into consideration the percentage of requests handled by DI-GRUBER; 3) **QTime** is better; and 4) **Normalized QTime** (defined as the ratio between QTime and the total number of requests) is noticeably improved.

Note that the scenario with only one decision point has a small **QTime**; this is due to the fact that within the one hour the test was performed, the number of requests made was smaller than in the other cases due to lower throughput, so practically the number of jobs entering the grid was smaller in comparison with available resources (an expected behavior). With fewer resources being used, it was easier for the decision point to make good decisions, and hence the small **QTime**. We computed **Normalized QTime** in order to take into account both the number of requests and the resource utilization; we see that the deceivingly low **QTime** for the one decision point scenario now shows its worse performance when compared to the other two scenarios. (The low utilization also makes this clear.)

3.4.3. Accuracy with Synchronization

The other important dimension in our analysis is the interval at which decision points perform synchronization. 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. Figure 8 shows our results, in this case just for jobs handled by DI-GRUBER. We see that for the workloads considered, a three minute exchange interval is sufficient to achieve 95% Accuracy.

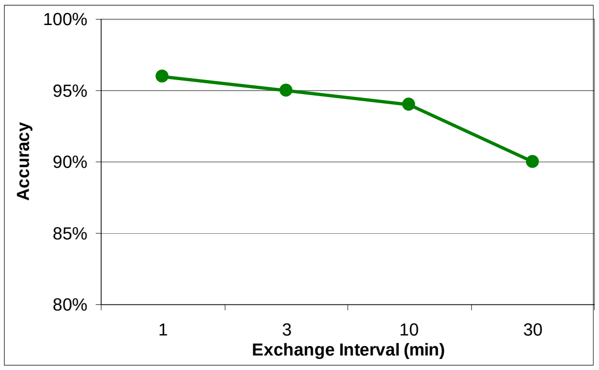


Figure 8: GT3 DI-GRUBER Scheduling Accuracy as Function of the Exchange Time Interval for Three Points

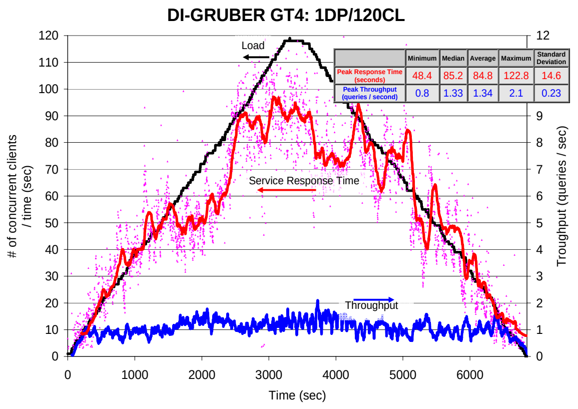
3.5. Results with GT4.0

We ported the DI-GRUBER implementation from GT3 to the GT3.9.5 prerelease of GT4. (This release is functionality equivalent to the final GT4.0 release, but provides somewhat lower performance than GT4.0, which is significantly faster than GT3.) GT4 is implemented quite differently from GT3, and thus experiments with this "GT4 DI-GRUBER" provide a means for further exploration of various parameters and behaviors based on infrastructure performance. We consider that it is like comparing two different resource brokers built for similar purposes but based on different technologies.

3.5.1. Infrastructure Scalability

Using DiPerF, we varied slowly the number of clients for this set of tests. Figures 9-11 below present as before **Response**, **Throughput**, and **Load** for 1, 3, and 10 decision points.

For one decision point (Figure 9) we see a steadily increasing service response time as the number of concurrent machines increases; during the peak period, the average service **Response** time was about 84 seconds. **Throughput** increases rapidly, but after about 10 concurrent clients, plateaus just above 1 query per second; the throughput remains relatively constant at about 1.3 queries per second even when all testing machines (close to 120 in this case) are accessing the service in parallel.



	Minimum	Median	Average	Maximum	Standard Deviation
Peak Response Time (seconds)	48.4	85.2	84.8	122.8	14.6
Peak Throughput (queries / second)	0.8	1.33	1.34	2.1	0.23

Figure 9: GT4 Centralized Scheduling

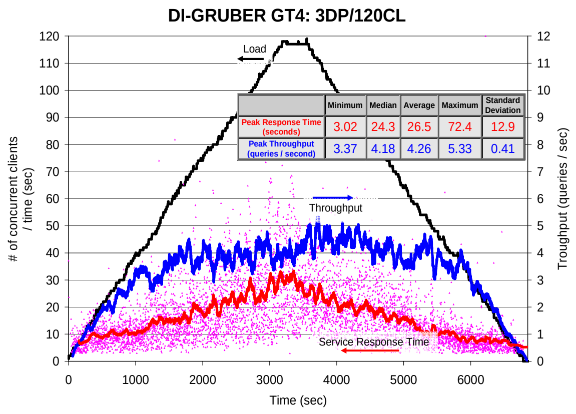


Figure 10: GT4 DI-GRUBER with Three Decision Points

With three decision points (Figure 10), **Throughput** increases slowly to about 4 job scheduling requests per second when all testing machines are accessing the service in parallel. The service **Response** time is also smaller (about 26 seconds) on average compared with the previous results (about 84 seconds).

With 10 decision points (Figure 11), the average service **Response** time decreased even further to about 13 seconds, and the achieved **Throughput** reached about 7.5 queries per second during the peak load period. 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.

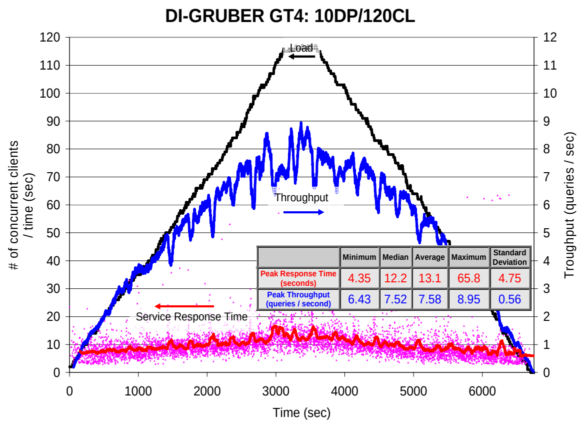


Figure 11: GT4 DI-GRUBER with 10 Decision Points

Overall, **Throughput** and **Response** improve by a factor of three when the number of decision points is increased from one to three, and by a factor of five when using five decision points. Again, practically, we see that for GT4 DI-GRUBER, three decision points are sufficient when around 120 clients are scheduling jobs. This result is confirmed analytically in Section 4.

3.5.2. Accuracy and Scheduling Performance

Next, we analyze the performance of the GT4 DI-GRUBER and its strategies for providing accurate scheduling decisions as in the GT3 case. Again, we look from both an infrastructure complexity and synchronization interval point of view.

Table 2 depicts the overall performance of GT4 DI-GRUBER in the scenarios introduced before. Again, the values under the "All Requests" section provide an overall view of the implementation's performance, but do not reflect actual performance.

Table 2: GT4 DI-GRUBER Overall Performance

	Decision Points	% of Req	# of Req	QTime	Norm QTime	Util	Accuracy
Requests Handled by GRUBER	*1*	53%	3852	0	0.000	3%	98%
	3	92%	24048	452	0.018	16%	90%
	10	93%	37593	2501	0.066	35%	75%
Requests NOT Handled by GRUBER	*1*	47%	3382	0	0.000	7%	-
	3	8%	1893	36	0.019	4%	-
	10	7%	2567	220	0.085	6%	-
All Requests	*1*	100%	7234	0	0.000	10%	94%
	3	100%	25941	660	0.025	20%	81%
	10	100%	40160	3017	0.075	41%	68%

If we consider only jobs that were scheduled through a single DI-GRUBER decision point, the results do not look that different, except for the one-decision point case. The explanation is that in the three and ten decision point cases, GT4 DI-GRUBER was able to handle almost all requests successfully, which is different from the GT3 DI-GRUBER.

3.5.3. Accuracy with Synchronization

Using DiPerF, we performed several tests where the decision points were exchanging status information at the same predefined time intervals (1, 3, 10, and 30 minutes). The results in Figure 12 show that in the GT4 case, for a three decision point infrastructure a three to ten minutes exchange interval is sufficient for achieving almost 90% **Accuracy**. This value depends also on the number of the jobs scheduled by the decision points.

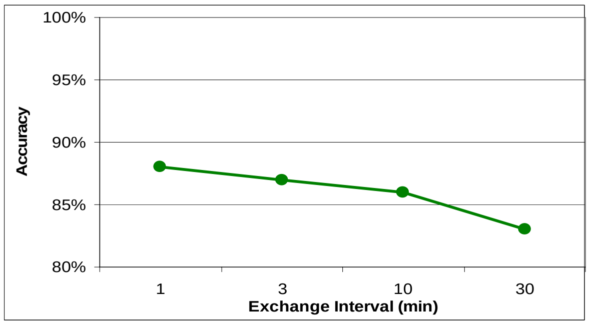


Figure 12: GT4 DI-GRUBER Scheduling Accuracy as Function of Exchange Time Interval for Three Decision Points

4. Infrastructure Dynamic Evaluation

While the above results are encouraging and meaningful for the settings we performed, one can reasonably argue that the appropriate number of decision points needed may depend on the dynamics, performance, and state of a particular grid.

Thus, we next focus on extracting meaningful elements that DI-GRUBER decision points can use to provide dynamic information (such as maximum load) that can be used to determine whether or not the saturation point was reached.

4.1. Evaluation Criteria

Saturation identification: The first element we want to identify is when the DI-GRUBER decision points get saturated. We use performance models created by DiPerF to establish an upper bound on the number of transactions that a decision point can handle per time interval. When this upper bound is reached, a decision point can trigger a "saturation" signal to a third party monitoring service responsible for handling these events.

Overall decision points needed per decision point set: Having information from each individual decision points about their state, a third party observer can decide dynamically what steps should be taken to reconfigure the scheduling infrastructure, for example by adding decision points or by rebalancing load among existing decisions points to avoid overloading.

In order to validate the proposed enhancements, we have developed a simple simulator (GRUB-SIM) capable of simulating DI-GRUBER decision points. We were interested in providing a simple means for dynamic identification of the number of required DI-GRUBER decision points starting from the logs we collected in the previous chapters. In essence, GRUB-SIM took the traces from the tests presented in the previous section, and attempted to identify the saturation points and the optimum number of decision points needed.

4.2. GRUB-SIM Results

GRUB-SIM automatically traces the **Response** metric and all overloads events, and simulates new decision points on the fly. The results are gathered in Table 3 and they provide the number of decision points required in each situation for achieving an adequate **Response** from DI-GRUBER. We see that for the GT3-based implementation, a total of 4 decision points was necessary. On the other hand, for the GT4 DI-GRUBER, a total of 5 decision points were needed.

Table 3: Required Decision Points

Decision Points	Additional Decision Points	
	GT3-based	GT4-based
1	3	4
3	1	2
10	0	0

While these results are encouraging, we do not have a DI-GRUBER implementation for such an approach. We hope to produce such an implementation in future work. However, these results strengthen our claim that only a few decision points, namely about 4 or 5, are enough to handle the scheduling for a grid that is 10 times larger than today's Grid3.

5. Related Work

A policy based scheduling framework for grid-enabled resource allocations is under development at the University of Florida [22]. This 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. One important difference of DI-GRUBER is the lack of an assumption of a centralized scheduling point.

The Grid Service Broker, a 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]. An important difference is that GridBus does not support the notions of sites, submission hosts, and virtual organizations or groups.

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. DI-GRUBER instead targets a different environment model, in which the main players are VO and resource providers with opportunistic needs (free resources are acquired when available).

6. Conclusions and Future Work

Managing USLAs within large virtual organizations that integrate participants and resources spanning multiple physical institutions is a challenging problem. Maintaining a single unified decision point for USLA management is a problem that arises when many users and sites need to be managed. We provide a solution, namely DI-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 addresses is the scalability and performance of scheduling infrastructures, DI-GRUBER in our case, in large Grid environments.*

The contributions of this paper are the results we achieved on two dimensions: how well our proposed solution performed in practice (three to ten decision points prove to be enough to handle a grid ten times larger than today's Grid3/OSG [18], with four to five decision points being sufficient as refined by GRUB-SIM) and a methodology to measure the success of such a meta-scheduler (performance metrics). We also introduced two enhancements to our previous GRUBER framework, namely the distributed approach in resource scheduling and USLA management, and an approach for dynamic computing the number of decision points for various grid settings.

We note that DI-GRUBER is a complex service: a query to a decision point may include multiple message exchanges between the submitting client and the decision point, and multiple message exchanges between the decision points and the job manager in the grid environment. In a WAN environment with message latencies in the 100s of milliseconds, a single query can easily take multiple of seconds to serve. We expect that performance will be significantly better in a LAN environment. However, one of DI-GRUBER's design goals was to offer resource brokering in a WAN environment such as grids.

While the transaction rate for the DI-GRUBER service is fairly low compared to other transaction processing systems [2,3,4], this rate proved to be sufficient in the Grid3 context [25]; furthermore, these other transaction processing systems were designed to be deployed in a LAN environment. Also, the transaction speed increases linearly with the number of decision points deployed over a grid. DI-GRUBER performance can be improved further by porting it to a C-based Web services core, such as is supported in GT4 [29]. The performance of DI-GRUBER could also be enhanced further simply by deploying it in a different environment that would have a tighter coupling between the resource broker (DI-GRUBER) and the job manager (Euryale); this approach would reduce the complexity of the communication from two layers to one layer.

We note that DI-GRUBER scaled almost linearly as we tested its performance with 1, 3, and 10 decision points respectively. The graphs (from Sections 3.4 and 3.5) that involve multiple decision points (especially with 10), it was expected to have a two-fold increase in clients with only one-fold increase in response time. For the graphs with only one decision point, the two-fold increase in clients resulted in roughly a two-fold increase in response time.

Also, by increasing the number of decision points (cooperating brokers that communicate via a flooding protocol) the throughput climbs to approximately 70 transactions/second with a low response time. This observation leads us to conclude that the required number of "decision" nodes to ensure scalability in a two-layer scheduling system like DI-GRUBER is relatively small.

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

Acknowledgements: This work was supported by the NSF Information Technology Research GriPhyN project, under contract ITR-0086044.

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, 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", in *Cluster Computing and Grid (CCGrid),* UK, May 2005.
25. Dumitrescu, C., Foster, I., "GRUBER: A Grid Resource SLA Broker", in *Euro-Par*, Portugal, September 2005.
26. A. Bavier et al., "Operating System Support for Planetary-Scale Services", Proceedings of the First Symposium on Network Systems Design and Implementation (NSDI), March 2004.
27. B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," ACM Computer Communications Review, vol. 33, no. 3, July 2003.
28. Voeckler, J., "Euryale: Yet Another Concrete Planner", in *Virtual Data Workshop*, May 18th, 2004.
29. M. Humphrey, G. Wasson, K. Jackson, J. Boverhof, M. Rodriguez, Joe Bester, J. Gawor, S. Lang, I. Foster, S. Meder, S. Pickles, and M. McKeown, "State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations", 4th IEEE International Symposium on High Performance Distributed Computing (HPDC-14), Research Triangle Park, NC, 24-27 July 2005.