Experiences in Running Workloads over Grid3

Catalin L. Dumitrescu[1], Ioan Raicu[1], and Ian Foster[1,2]

[1] Computer Science Department, The University of Chicago,
5801 S. Ellis Ave., Chicago, IL, 60637
cldumitr@cs.uchicago.edu

[2] Mathematics and Computer Science Division, Argonne National Laboratory,
9700 S. Cass Ave., MCS/221, Argonne, IL, 60439
foster@mcs.anl.gov

Abstract. Running workloads in a grid environment is often a challenging problem due the scale of the environment, and to the resource partitioning based on various sharing strategies. A resource may be taken down during a job execution, be improperly setup or just fail job execution. Such elements have to be taken in account whenever targeting a grid environment for execution. In this paper we explore these issues on a real grid, Grid3, by means of a specific workload, the BLAST workload, and a specific scheduling framework, GRUBER - an architecture and toolkit for resource usage service level agreement (SLA) specification and enforcement. The paper provides extensive experimental results. We address in high detail the performance of different site selection strategies of GRUBER and the overall performance in scheduling workloads in Grid3 with workload sizes ranging from 10 to 10,000 jobs.

1 Introduction

Grid3 represents a multi-virtual organization that sustains production level services required by various physics experiments. The infrastructure is composed of more than 30 sites and 4500 CPUs, over 1300 simultaneous jobs and more than 2TB/day. The participating sites are the main resource providers under various conditions. We consider in this paper that all these sites are governed by various usage SLAs [3]. We distinguish here between "resource *usage* policies" (or SLAs) and "resource *access* policies." Resource access policies typically enforce authorization rules. In contrast, resource usage SLAs govern the *sharing* of specific resources among multiple groups of users. Once a user is permitted to access a resource via a resource access policy, then the resource usage policy steps in to govern *how much* of the resource the user is permitted to consume. GRUBER focuses on computing resources such as computers, storage, and networks; owners may be either individual scientists or sites; and VOs are collaborative groups, such as scientific collaborations.

In this paper, we focus on the problems that can occur in Grid3, because various elements affect workload execution times. We measure the impact by means of *average resource utilization*, *average response time*, *average job completion*, *average job re-planning* (Replan), *workload completion time* (Time), and *job completion gain*

(Speedup) [1]. We present our detailed results for scheduling BLAST workloads over one of the largest US grid, the Grid3 [3] environment.

2 Goals

Running various size workloads over Grid3 can become a challenging problem when resources are shared by a large user community. For most environments, even short periods of overloading can decrease the performance the users get from the system. In any grid context, such overloading scenarios can be only partial as a grid is a large composition of resources spread in various administrative domains. Here we try to identify some of the main challenges users may face when submitting workloads in such environments and to provide also some simple means for improving their achieved performance. We assume in our scenario that various sites are used at particular times and we are interested in measure how well the overall performance maintains over larger time intervals.

2.1 Resource Providers and Consumers

Grid3 is composed of resources provided based on various usage SLA [1],[17]. Further, resources are aggregated at the VO level and provided on similar usage SLA means to groups and users. Our usage SLA scheduling framework, GRUBER, deals with two classes of entities: resource providers and resource consumers. A physical site is a resource provider; a VO is a consumer (consuming resources provided by a site) and a provider (providing resources to users, groups or workload types). We assume that each provider-consumer relationship is governed by an appropriate SLA.

2.2 Usage SLA-based Resource Sharing

The entire grid environment is described as follows:

- A grid consists of a set of resource provider *sites* and a set of *submit hosts*.
- Each site contains a number of processors and some amount of disk space.
- A three-level hierarchy of *users*, *groups*, and *VOs* is defined, such that, each user is a member of one group, and each group is a member of one VO.
- Users submit jobs for execution at submit hosts. A job is specified by four attributes: VO, Group, Required-Processor-Time, Required-Disk-space.
- A *site policy statement* defines site usage SLAs by specifying the number of processors and amount of disk space that the site makes available to each VOs.
- A *VO policy statement* defines VO usage SLAs by specifying the fraction of the VO's total processor and disk resources (i.e., the aggregate of contributions to that VO from all sites) that the VO makes available to different groups.

We note that this model is one of resource sub-allocation: resources are owned by sites, which apportion them to VOs. VOs in turn apportion their "virtual" resources to groups. Groups could, conceptually, apportion their sub-allocation further, among specific users. Without loss of generality, we simplify both this discussion and our implementation by sub-allocating no further than from VOs to groups.

3 Environment Settings

The execution environment is based on the VDS toolkit developed in the GriPhyN project context. The specific technicalities are as follows.

3.1 Euryale as Concrete Planner

Euryale [9] is a complex system aimed at running jobs over a grid, and in special over Grid3 [3]. The approach used by Euryale is to rely on the Condor-G capabilities to submit and monitor jobs at sites. It takes a late binding approach in assigning such jobs to sites. In addition, Euryale allows a simple mechanism for fault tolerance by means of job re-planning when a failure is discovered.

During a workload execution, DagMan executes the Euryale's pre- and post-scripts, the heart of Euryale concrete planner, which also contain the Euryale's execution logic. The prescript calls out to the external site selector, rewrites the job submit file, 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. To run things in the grid, Euryale needs knowledge about the available resources, or sites. An important feature of Euryale is its capacity to invoke external site selectors in job scheduling, such as our resource broker, GRUBER [18].

3.2 GRUBER as Resource Broker

GRUBER is the main component that we used for the site selection. It is composed of four principal components, as we now outline [18]. The *GRUBER engine* represents the main component of the architecture. It implements various algorithms for detecting available resources and maintains a generic view of resource allocations and utilizations.

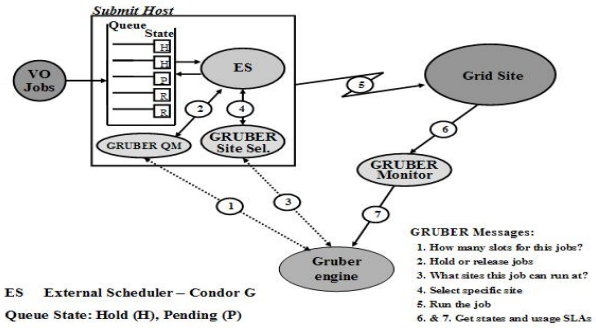

Fig. 1: GRUBER Architecture

The *GRUBER site monitoring component* is one of the data providers for the GRUBER engine. This component is optional and can be replaced with various other

grid monitoring components that will provide similar information, such as MonaLisa [10] or Grid Catalog [3]. So far, we are unaware of any complete replacement. *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 random assignment (*G-RA*), round robin (*G-RR*), least used (*G-LU*), or last recently used (*G-LRU*) task assignment policies. We note that one can use just the GRUBER engine and site selectors, without the GRUBER queue manager. This option makes GRUBER only a site recommender, without having the capacity to enforce any usage SLA expressed at the VO level, while enforcing the usage SLA at the site level by means of removing a site for an already over-quota VO user at that site.

3.3 Disk Space Considerations

Disk space management introduces additional complexities in comparison to job management [18]. For the experiments in this paper, we have extended GRUBER to deal with multiple resource issues and describe here in more detail our work. If an entitled-to-resources job becomes available, it is usually possible to delay scheduling other jobs, or to preempt them if they are already running. In contrast, a file that has been staged to a site cannot be "delayed," it can only be deleted. Yet deleting a file that has been staged for a job can result in livelock, if a job's files are repeatedly deleted before the job runs. As a consequence, a different approach has been devised. As a concrete example, a site can become heavily loaded with a one VO jobs and because of which other jobs are either in the local queue in an idle state waiting for their turn. But this does not stop the submission of more jobs. As a result, there may be lots of other input data on the site and the disk used space will keep on growing. On the other hand, the other jobs are not getting their turn to actually finish and delete their files. The rate of input data being copied over to the site is higher than the rate of completion of jobs, making the disk space to get full.

So far, we have considered a UNIX quota-like approach. Usually, quotas just prevent one user on a static basis from using more than his hard limit. There is no adaptation to make efficient use of disk in the way a site CPU resource manager adapts to make efficient use of CPU (by implementing more advanced disk space management techniques). More precisely, for scheduling decisions a list of site candidates that are available for use by a VO i for a job with disk requirements J, in terms of provided disk space, is built by executing the following logic.

1. **for each** site s **in** site list G **do**
2. # Case 1: *over-used site by VO$_i$*
3. **if** $IA_i > IP_i$ for VO i at site s
4. **next**
5. # Case 2: *un-allocated site*
6. **else**
7. **if** $\Sigma_k(IA_k) < s.TOTAL - J$ **&&**
$IA_i + J < IP_i$ **then**
8. add (s, S)
9. **return** S

with the following definitions:

S = Site Set
k = index for any VO != VO_i
IP_i = Instantaneous Usage SLA for VO_i
IA_i = Instantaneous Resource Usage for VO_i
TOTAL = upper limit allocation on the site

The set of disk-available site candidates is combined with the set of CPU-available site candidates and the intersection of the two sets is used for further scheduling decisions.

4 Experimental Results

We first introduce the metrics [18] that we used to evaluate the alternative strategies, and then introduce our experimental environment, and finally present and discuss our results.

4.1 Metrics

We use five metrics to evaluate the effectiveness of workload performance execution.

- **Comp**: the percentage of jobs that complete successfully.

$$\mathbf{Comp = (Completed\ Jobs) / \#_{jobs} * 100.00}$$

- **Replan**: the number of performed replanning operations.
- **Util**: average resource utilization, the ratio of the per-job CPU resources consumed (ET_i) to the total CPU resources available as a percentage:

$$\mathbf{Util = \Sigma_{i=1..N}\ ET_i / (\#_{cpus} * \Delta t) * 100.00}$$

- **Delay**: average time per job (DT_i) that elapses from when the job arrives in a resource provider queue until it starts:

$$\mathbf{Delay = \Sigma_{i=1..N}\ DT_i / \#_{jobs}}$$

- **Time**: the total execution time for the workload.
- **Speedup**: the serial execution time to the grid execution time for a workload.
- **Spdup75**: the serial execution time to the grid execution time for 75% of the workload.

All metrics are important: an adequate environment will both maximize delivered resources and meet owner intents. For example, Table 1 presents a simple case scenario in which several VOs have various SLAs, resource requests, and resource utilizations at a particular resource provider. The column names have the following meanings:

- **Target:** the usage SLA for the consumer at the resource provider, as a percentage of site capacity.
- **Current:** the current utilization for the consumer at the provider, as a percentage.
- **Demand:** the current resource demand from the consumer for the provider, as a percentage (it cannot be less than **Current**).
- **Level**: how our criteria is met (OK when **Current** = MIN (**Target**, **Demand**)).

Table 1. Usage SLAs in our Scenario

VO	Target	Current	Demand	Level
USCMS	60	50	50	OK
USATLAS	20	15	30	Under
IVDGL	10	10	100	OK
LIGO	5	3	3	OK
SDSS	5	22	50	Over

4.2 Experiment Settings

We used a single job type in all our experiments, the sequence analysis program BLAST. A single BLAST job has an execution time of about 40 minutes (the exact duration depends on the CPU), reads about 10-33 kilobytes of input, and generates about 0.7-1.5 megabytes of output: i.e., an insignificant amount of I/O. We used these BLAST workloads in three different sets of workload configuration: (1) small workloads of 10, 50, and 100 jobs that get scheduled at once; (2) medium workloads of 500 to 1000 jobs that are submitted in several steps in order to honor the VO usage SLAs; and (3) large workloads of 10k jobs.

Table 2. Grid3 CPU Allocations on July 9, 2004

Site Name	# of CPUs	VO Allocations (in %)		
		iVDGL	Atlas	USCMS
T2cms0.sdsc.edu	76	0.62	24.74	0.40
nest.phys.uwm.edu	305	0.00	7.28	0.00
uscmsb0.ucsd.edu	3	11.68	11.68	11.68
xena.hamptonu.edu	1	25.00	25.00	25.00
garlic.hep.wisc.edu	101	3.01	3.01	3.01

We performed all experiments on Grid3, which comprises around 30 sites across the U.S., of which we used 15. Each site is autonomous and managed by different local resource managers, such as Condor, PBS, or LSF. Each site enforces different usage policies which are collected by the GRUBER site monitor. For example, Table 1 gives CPU allocations per VO on five Grid3 sites on July 9, 2004 as collected through the GRUBER site monitor. We submitted all jobs within the iVDGL VO, under a VO usage policy that allows a maximum of 600 CPUs. Furthermore, we submitted each individual workload under a separate iVDGL group, with the constraint than no one group can get more than 25% of iVDGL CPUs, i.e., 150.

We also configured GRUBER to employ a replanning policy, by which a starving job was removed after a predefined time interval (20 minutes here) and resubmitted for rescheduling. If a job was submitted unsuccessfully 10 times or it was reported as

application level "failure" by a site, then it is considered a failure. All submissions were performed without withholding or setting any special priorities at sites, practically GRUBER had to find resources while all workloads ran in parallel. The VO usage SLA limited the submitting group to approximately 150 jobs at a time.

Regarding the site selectors, they were already introduced by Dumitrescu et al. [18] and used in a similar fashion. In a nutshell, G-RA represents GRUBER random assignment site strategy, G-RR represents GRUBER round robin site strategy, G-LU represents GRUBER least used site assignment strategy, and G-LRU represents least recently used site strategy [18].

4.3 Small Workload Results

Table 3 shows the results for the 1x10 jobs workloads. In the ideal case, these values are: Comp=100, Replan=0, Util=1.25, Delay=0, Time=3000, and Speedup=10. As can be seen, the speedup is 2.5 to 3.5 times smaller due to the probability of a jobs ending on sites with a local resource manager that do not behave as expected. The job starvation was "detected" after a time interval comparable with the execution time (20 minutes vs. 40 minutes). However, 75% of the jobs do complete in a time interval closer to the ideal case of a speedup of 10.

Table 3. Results and 90% Confidence Intervals of Four Policies for 1x10 workloads

	G-RA	G-RR	G-LU	G-LRU
Comp(%)	100	100	100	100
Replan	34.1 ± 5.51	47.5 ± 9.26	8.6 ± 1.83	13.6 ± 2.18
Util (%)	0.36 ± 0.05	0.31 ± 0.07	0.55± 0.10	0.50 ± 0.04
Delay (s)	3262 ± 548	4351 ± 824	1162 ± 376	801 ± 313
Time (s)	12436 ± 1191.4	13966 ± 2208.8	8787±158	7653 ± 205.9
Speedup	2.33 ± 0.25	2.21 ± 0.35	3.6 ± 0.6	3.46 ± 0.45
Spdup75	3.72 ± 0.59	3.46 ± 0.51	5.32± 0.67	5.66 ± 0.55

Table 4 shows the results for the 1x50 jobs workloads. In the ideal case, these values are: Comp=100, Replan=0, Util=6.25, Delay=0, Time=3000, and Speedup=50. The same situation as before was encountered in this set of experiments: several jobs starved and their execution time affected the overall speedup. The speedup of 75% of the jobs instead is more than the half of the ideal speedup, proving that most of the jobs do complete close to the optimal time.

Table 4. Results and 90% Confidence Intervals of Four Policies for 1x50 workloads

	G-RA	G-RR	G-LU	G-LRU
Comp(%)	100	100	100	100
Replan	35 ± 14	51.1 ± 28	48.8 ±10.8	78.8 ± 9.51
Util (%)	1.18 ± 0.25	1.44 ± 0.27	1.89 ±0.43	1.76 ± 0.18
Delay (s)	1420 ± 713	583 ± 140.4	653.8 ±202	1260 ± 528.7
Time (s)	8035 ± 990.4	9654 ± 603.5	8549 ±898	9702 ± 1247.3
Speedup	16.35 ± 1.17	14.12 ± 0.90	15.16 ±2.42	12.76 ± 0.71
Spdup75	30.84 ± 5.70	35.36 ± 2.79	35.41 ±2.48	24.36 ± 2.28

Table 5 shows the results for the 1x100 jobs workloads. In the ideal case, these values are: Comp=100, Replan=0, Util=12.50, Delay=0, Time=3000, and Speedup=100. Again, similarly to the 1x50 workloads, the execution performance is half for 75% of the workloads and drops for the entire workload.

Table 5. Results and 90% Confidence Intervals of Four Policies for 1x100 workloads

	G-RA	G-RR	G-LU	G-LRU
Comp(%)	100	100	100	100
Replan	228.7 ± 21	39.9 ± 13.8	124.7 ± 17	230 ± 20.3
Util (%)	2.86 ± 0.30	3.48 ± 0.59	3.51 ± 0.7	1.87 ± 0.46
Delay (s)	1691 ± 198	529 ± 92.67	640 ± 93.4	1244 ± 387.9
Time (s)	10350 ± 565.9	9013 ± 1025.1	9716±1130	7507 ± 2325.1
Speedup	22.43 ± 1.55	30.15 ± 3.43	28.02 ± 5.4	19.24 ± 1.56
Spdup75	47.38 ± 3.24	77.19 ± 3.26	73.54 ± 2.0	35.86 ± 3.72

4.4 Medium Workload Results

Table 6 shows the results for the 1x500 jobs workloads. Here, in the ideal case, the values are: Comp=100, Replan=0, Util=25.00, Delay=3600, Time=3000, and Speedup=150. The size of the workloads makes the execution performance to increase, and practically almost match the ideal speedup for the 75% of the workload and be only half for the overall workload.

Table 6. Results and 90% Confidence Intervals of Four Policies for 1x500 workloads

	G-RA	G-RR	G-LU	G-LRU
Comp(%)	100	100	100	100
Replan	925 ± 103.5	816 ± 245.6	680 ± 139.3	1024 ± 154.2
Util (%)	34.04 ± 4.55	33.19 ± 2.39	30.3 ± 4.7	25.41 ± 5.6
Delay (s)	9202 ± 1716.8	6700 ± 816.6	6169 ± 407	9125 ± 6117.8
Time (s)	28116 ± 2881	24225 ± 035.9	21362 ±1250	20434 ± 4100
Speedup	67.32 ± 5.6	60.22 ± 3.26	63.12 ±3.41	51.77 ± 5.94
Spdup75	98.43 ± 8.7	111.69 ± 9.81	113.2 ±8.82	101.48 ± 10.05

4.5 Large Workload Results

Next, we report on previous results were we used a more aggressive scheduling approach. In this approach, the number of retries was limited to five versus ten job retries. Also, in these measurements some of the sites were not properly configured and jobs failed immediately. These workloads provide also insights about GRUBER's scalability. Our results are captured in Table 7. Round-robin and random-assignment prove to achieve the best performance. The lower completion rates are explained by the number of low number of retries (5) and the missing of BLAST environment configuration at a few sites. All these factors are also an explanation for the lower performance achieved in these cases.

Table 7. Results of Four GRUBER Strategies for 1x1k workloads

	G-RA	G-RR	G-LU	G-LRU
Comp(%)	97	96.7	99.3	85.6
Replan	1396	1679	1326	1440
Util (%)	12.85	12.28	14.56	10.63
Delay (s)	49.07	53.75	50.50	54.69
Time (s)	29484	37620	33300	80028
Speedup	140.3+	113.1+	122+	101.4+
Spdup75	173.5	159.3	161.4	127.8

Further, in the 10k workloads, the completion rates drop even more, as the probability of failures increases linearly with the number of jobs (GRUBER maintains a constant load on the available sites).

Table 8. Results of Four GRUBER Strategies for 1x10K workloads

	G-RA	G-RR	G-LU	G-LRU
Comp(%)	91.75	91.88	77.88	73.58
Replan	18000	23900	27718	24350
Util (%)	24.3	23.3	20.0	17.6
Delay (s)	86.63	85.17	89.01	90.45
Time (s)	226k	260k	295k	349k
Speedup	137+	145.4+	134+	98.3+
Spdup75	156.2	163	139.6	98.3+

The results in Table 9 are the means across the four submitters. We see some interesting differences from Table 7. G-LU's completion rate drops precipitously, presumably for some reason relating to greater contention. The total execution times for G-RA and G-LU increase, although more runs are required to determine the significance of these results.

Table 9. Results of Four GRUBER Strategies for 4x1K workloads

	G-RA	G-RR	G-LU	G-LRU
Comp(%)	98.2	98.7	91.7	87.9
Replan	1815	1789	2409	1421
Util (%)	13.51	14.02	11.52	11.05
Delay (s)	66.62	64.41	63.96	68.97
Time (s)	40356	37800	48564	48636
Speedup	77.3+	74.1+	71+	60+
Spdup75	105.6	102.9	93.8	82.4

4.6 Failure Analysis

While for the small and medium workloads the number of failures was null, for large workloads the completion rates vary as can be observed from the tables in

+ Results for incomplete workloads.

subsection 0. The motivations for these failures are in most cases the small number of retries used during these tests (5 instead of 10), the temporary failure of the RLS server used to stage in and out data (overloading issues), in a few cases due to DagMan failure in managing jobs (application bugs), and gatekeepers overloading. Most of these errors were reported and fixes were performed or are expected in the future for the signaled problems.

Also, we have to note that Grid3 performance in executing BLAST workloads has already increased between the first set of experiments (large workloads) and the second ones (small and medium workloads). Either individual sites had undergone hardware upgrades or the job assignment policies to individual computing nodes became more job requirements aware (avoiding node overloads).

5 Statistical Analysis

While previous results provide useful insight about how Grid3 performs in executing workloads when GRUBER is the steering mechanism, further analysis is required to identify how the scheduling strategies have performed comparatively. Fig. 2 presents the speedup performance over all runs and the confidence intervals at 90%. Note the small confidence intervals for all runs, which express low standard deviation and the strength of our results across the runs and configurations.

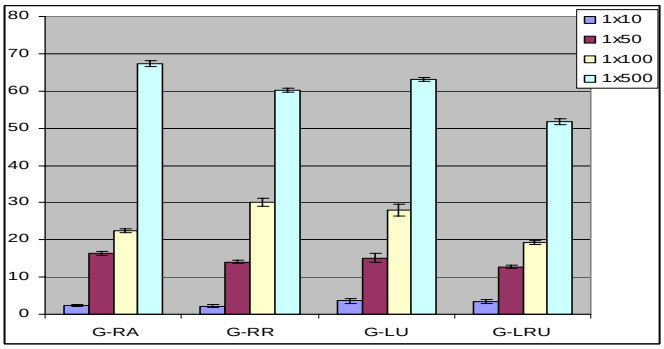

Fig. 2. Speedup Comparisons among Workloads

Further, we use the T-test to correlate the results of these experiments. The T-test is usually used for comparing the results of two alternative approaches with the claim that the results are significantly different. For comparison means, we use tournament trees and T-tests as comparison operator. These results are captured in Table 10. The null hypothesis and alternative hypothesis that we set up to conduct the t-test are:

- H_0 (null hypothesis): any given two runs have comparable performance;
- H_a (alt. hypothesis): prove H_0 is false; two runs do not have same performance.

The null hypothesis is the one that we want to reject as not being true, while the alternative hypothesis is the one that we want to accept as being true. Our alternative hypothesis is two sided since we test that the runs are just different, which implies either < or >; we essentially test 2 * P (t > critical_value) to be less than 0.05. The goal is to obtain a probability that the t value will be greater or less than the critical

value. The p value needs to be less than 0.05 for the results to be statistically significant, which implies that H_a is true with 95% confidence for the corresponding comparisons; the lower the p value, the better the confidence. If a p value cannot be found that is less than 0.05, then the sample space is not statistically significant, and hence more samples must be obtained.

Table 10. Tournament Tree (T-tests) Results

	G-RA vs. G-RR	**G-LU vs. G-LRU**	**G-RA vs. G-LU**
1x10	0.09 (?)	0.17 (?)	0.0005 (T)
1x50	0.0005 (T)	0.0005 (T)	0.0005 (T)
1x100	0.0005 (T)	0.0005 (T)	0.0005 (T)
1x500	0.0005 (T)	0.0005 (T)	0.0005 (T)

The results from Table 10 show that for all workloads other than the smallest one, the results are statistically significant with at least a 99.95% confidence. Regarding the smallest workload of 1x10 (for which we had 10 sample runs), the number of samples in our experiment do not seem to be enough, and hence more experiments would have to be performed for the 1x10 workload.

6 Related Work

There are several other production workloads running over Grid3, such as the QuarkNet Project, SDSSCoAdd, GADU, or fMRIDC. While these workloads are important from the GriPhyN project point of view, they offer little elements for comparisons with the work described here. Firstly, these workloads are run mostly for their results and not for measuring various Grid3 execution capacities. The closest one is the SDSS/CoAdd workload in scope; however we do not have information to date about various metrics [20]. Besides the iVDGL workloads running over Grid3, there are other challenge problems solved. For example, the ATLAS "VO" and applications focus on Monte Carlo simulation of the physics processes that will occur in high energy proton collision at LHC; SDSS runs various problems related to galaxy clusters identification or pixel-level analysis of astronomical data, etc. [3].

7 Conclusions

Running workloads in grid environments is often a challenging problem due the scale of the environment and to the resource participation based on various sharing strategies. A resource may be taken down during job execution, be improperly setup or just fail job execution. Such elements have to be taken in account when targeting a grid environment.

In this paper we explored some of the issues that occur on a real grid, namely Grid3, by means of a specific workload, the BLAST workload, and a specific scheduling framework, GRUBER - an architecture and toolkit for resource usage service level agreement (SLA) specification and enforcement. During these experiments we faced various problems as described above, as well as quantified what performance a grid user should expect. In addition, we observed for our brokering

mechanism that for medium workloads, G-RA performs best with a 90% confidence interval, while G-LU performed best for smaller workloads. We also note that G-LRU performed worst for all tested workloads.

References

1. Dumitrescu, C. and I. Foster, "Usage Policy-based CPU Sharing in Virtual Organizations", in *5th International Workshop in Grid Computing*, 2004.
2. Foster, I., C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", in *International Journal of Supercomputer Applications*, 2001.
3. Foster, I., et al., "The Grid2003 Production Grid: Principles and Practice", in *13th International Symposium on High Performance Distributed Computing*, 2004.
4. 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)*, NY, 2004.
5. Altair Grid Technologies, LLC, *A Batching Queuing System,* Software Project, 2003.
6. Irwin, D., L. Grit, and J. Chase, "Balancing Risk and Reward in a Market-based Task Service", in *13th International Symposium on High Performance Distributed Computing*.
7. IBM, *WSLA Language Specification, Version 1.0.* 2003.
8. Pearlman, L., et al., "A Community Authorization Service for Group Collaboration", in *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*.
9. Voeckler, J., et al, "Euryale: Yet another Concrete Planner", Virtual Data Workshop, Presentation, 2004.
10. Legrand, I.C., et al. "MonALISA: A Distributed Monitoring Service Architecture", in *Computing in High Energy Physics*, La Jolla, CA, 2003.
11. 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)*, NY, 2004.
12. Cluster Resources, Inc., *Maui Scheduler*, Software Project, 2001-2005.
13. Henry, G.J., *A Fair Share Scheduler.* AT&T Bell Laboratory Technical Journal, 1984.
14. Kay, J. and P. Lauder, "A Fair Share Scheduler", University of Sydney, AT&T Bell Labs.
15. 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)*, New Mexico, '04.
16. Buyya, R., "GridBus: A Economy-based Grid Resource Broker", The University of Melbourne, 2004, Melbourne.
17. Dumitrescu, C. and I. Foster, "GangSim: A Simulator for Grid Scheduling Studies", in *Cluster Computing and Grid (CCGrid),* 2005, Cardiff, UK.
18. Dumitrescu, C., and I. Foster, "GRUBER: A Grid Resource SLA Broker", Euro-Par, 2005, Lisbon, Portugal.
19. Dumitrescu, C., I. Foster and I. Raicu, "A Scalability and Performance Measurements of a Usage SLA based Broker in Large Environments", iVDGL/GriPhyN Tech-Report, 2005.
20. Avery, P. and I. Foster, "GriPhyN Project Description", http://www.griphyn.org.
21. Dumitrescu, C., Wilde, M., and Foster, I., "A Model for Usage Policy-based Resource Allocations in Grids", in *IEEE/Policy Workshop*, 2005, Stockholm, Sweden.
22. Zhuge, H., The Future Interconnection Environment, IEEE Computer, 38 (4)(2005) 27-33.