GangSim: A Simulator for Grid Scheduling Studies

Catalin L. Dumitrescu
Department of Computer Science
The University of Chicago
catalind@cs.uchicago.edu

Ian Foster
Mathematics and Computer Science Division
Argonne National Laboratory &
The University of Chicago

Abstract

Large distributed Grid systems pose new challenges in job scheduling due to complex workload characteristics and system characteristics. Due to the numerous parameters that must be considered and the complex interactions that can occur between different resource allocation policies, analytical modeling of system behavior appears impractical. Thus, we have developed the GangSim simulator to support studies of scheduling strategies in Grid environments, with a particular focus on investigations of the interactions between local and community resource allocation policies. The GangSim implementation is derived in part from the Ganglia distributed monitoring framework, an implementation approach that permits mixing of simulated and real Grid components. We present examples of the studies that GangSim permits, showing in particular how we can use GangSim to study the behavior of VO schedulers as a function of scheduling policy, resource usage policies, and workloads. We also present the results of experiments conducted on an operational Grid, Grid3, to evaluate GangSim's accuracy. These latter studies point to the need for more accurate modeling of various aspects of local site behavior.

1. Introduction

An important class of applications in Grid environments are those involving numerous and loosely coupled jobs that handle large data sets [3]. Grid-aware scheduling policies for such applications are both a necessity and a challenge. For example, Kavitha et al. [4] have developed a framework in which "*data movements are performed by a decoupled, asynchronous process on the basis of observed data access patterns and loads*."

While job scheduling and data replication algorithms remain challenging problems, we focus here on the impact of resource allocation policies adopted by sites and virtual organizations (VO) on the performance achieved by sites and individual VOs.

The starting point for this work was an exploration of distributed system monitoring conducted within the context of the GriPhyN and iVDGL projects [1,2]. We developed the VO Ganglia Monitoring Toolkit to gather resource characteristics, utilization data, and usage limits for a collection of sites. The toolkit enhanced the Ganglia Monitoring Toolkit [12] with components designed to evaluate the impact of different VO-level task assignment strategies and site usage policies on achieved performance. From there, it was a relatively easy step to replace "real sites" with "simulated sites," and thus to enable the evaluation of a wider range of site and VO policies and behaviors than was possible in a real system. The resulting GangSim system is the subject of this paper. The name GangSim reflects both the origins of the implementation and the fact that it can be used to simulate large "gangs" of sites and users.

The novelty of the GangSim simulator consists in its modeling of not only sites but also VO users and planners, and its ability to model usage policies at both the site and VO levels. The resulting simulation system allows various task assignment policies to be tested in conjunction with different usage policies in a range of different grid configurations and workloads. In addition, GangSim permits parallel execution and can combine simulated components with instances of a VO Ganglia Monitoring Toolkit running on real resources. Thus, one can in principle use GangSim to study how the behavior of an existing Grid may change with an increase in of the number of sites, Grid level components, or users.

The rest of this paper is structured as follows. We provide firstly a more detailed description of the context that provisions our work. In Section 3, we detail the concepts and simulation models captured by GangSim. Section 4 describes our modifications to the Ganglia Monitoring Toolkit, while Section 5 describes some experiments that we have conducted using the simulator. Section 6 describes an initial validation study involving a real grid, namely Grid3 [2]. We review other Grid simulators in Section 7 and conclude in Section 8.

2. Environment Overview

The environment that we want to simulate comprises potentially large numbers of resources, resource owners, and VOs: see, for example, Grid3 [2]. We may have hundreds of institutions and thousands of individual investigators that collectively control tens or hundreds of thousands of computers and associated storage systems [4]. Each individual investigator and institution may participate in, and contribute resources to, multiple collaborative projects that can vary widely in scale, lifetime, and formality [10].

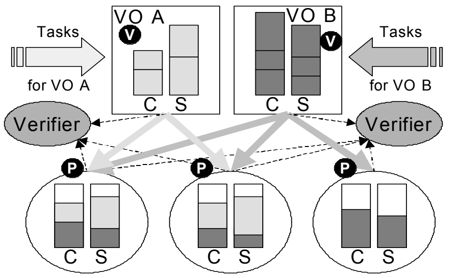

Figure 1: Resource Allocation Schematic

Figure 1 shows our model of resource allocation, in an architectural view, which we will return to in order to describe in more detail how we modeled this environment in our simulator. In the two VOs (squares) and three sites (circles), shaded elements indicate the compute (C) and storage (S) resources allocated to each VO at each site.

3. Simulator Model

The main questions we wish to explore by means of simulation studies are: "*What site usage policies are appropriate in a Grid environment, and how do these policies impact achieved site and VO performance?*", "*What usage policy may be applied at the VO level?*", and *"What site selection policies are best suited for various Grid environments?"*

Our GangSim simulator simulates a policy-driven management infrastructure in which policies concerning the allocation of resources within communities (VOs) and the allocation of resources across VOs at individual sites interact to determine the ultimate allocation of individual computing resources (CPU, disk, and network). Usage policies are expressed in terms of rules associated with sites, VOs, groups, and users for different aggregations of available resources. For example, a VO policy might say "group A gets 50% of all resources available to this VO," while a site policy might say "VO B gets 20% of my resources."

GangSim models the following elements encountered in real Grids: a job submission infrastructure, a monitoring infrastructure, and a usage policy infrastructure. The principal GridSim components are external schedulers (ES), local schedulers (LS), data schedulers (DS), monitoring distribution points (MDP), site policy enforcement points (S-PEP), and VO policy enforcement points (V-PEP). Sites aggregate several computing nodes and VOs aggregate users, who may be further organized into groups. We describe each of these components in more detail in the next subsection.

3.1. Simulated Components

A *site* is characterized by the capacity and number of its CPU, disk space, and networks. Each characteristic is described through a configuration file and loaded during startup. Both intra- and inter-site network capacities are defined. Associated policy statements define the scheduling policy used by the site and how much CPU time, disk space, and network bandwidth each VO may use.

A *VO* is composed of a set of groups of users. Users submit jobs, which may be grouped into sets called workloads. In addition, a set of data files corresponding to data elements required for job execution are distributed among sites in the beginning. Associated with each group is a set of policies that speak to the resources that the VO will make available to that group.

External Schedulers, Local Schedulers, and Data Scheduler (ES, LS, DS) represent points where various scheduling decisions are performed. An ES queues user jobs and selects a best site candidate for each job. Once scheduled to a site, a job moves from the ES to the LS associated with the chosen site. Examples of such schedulers in practice are Pegasus and Euryale as ES, and Condor, PBS and LSF as LS [4,13,14,15].

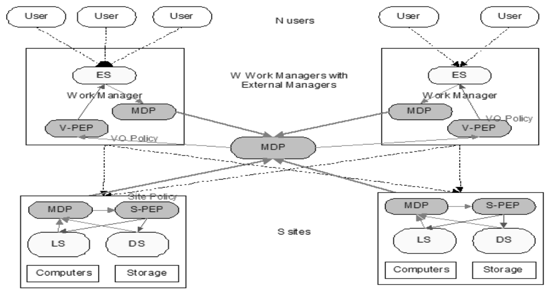

Figure 2: Simulated Environment

Monitoring Data Points (MDPs) represent the monitoring infrastructure "nodes" that carry out various metrics for Grid components' consumption. Such information is gathered from local and external schedulers, filtered, and delivered in a uniform manner.

Policy enforcement points (PEPs) are responsible for enforcing policies. 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 policy [6,7]. We distinguish two PEP types.

Site policy enforcement points (S-PEPs) reside at all sites and enforce site-specific policies. S-PEPs operate in a continuous manner, in the sense that jobs are immediately preempted or removed when policy requirements are no longer met. Jobs are not, however, necessarily restricted from entering site queues just because policy would prevent them from running.

VO policy enforcement points (V-PEPs) operate in a similar way to S-PEPs. They make decisions on a per-job basis to enforce policies regarding VO specifications for resource allocations to VO groups or to types of work executed by the VO. V-PEPs 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. V-PEPs interact with S-PEPs and schedulers to enforce VO-level policy specifications.

3.2. Strategies

During each simulation step, various algorithms and strategies are used to update the state of different components of the framework. There are algorithms for job selection, job assignment, and data file replication. There are algorithms for considering costs associated with various operations and for steering job flows through the framework. All of these algorithms are grouped in a single implementation module, and invoked every time a decision regarding the state of a component needs attention. Today schedulers allow sophisticated tuning possibilities. We tried to incorporate as much as possibly the characteristics of a generic site scheduler that allows policy or priority based scheduling and late planning.

3.2.1. Job flow: Jobs are submitted by users to ES queues according to its specific policy. (See Section 6.2 for a discussion of the policy used in studies described here.) Following site selection, they are moved to LS queues. A job may be rejected at the LS (e.g., because of no available disk space or local policy for the submitting VO), in which case the job is returned to the ES queue to re-enter the planning process. If the job requires larger files that the capacity of sites, it is rejected from the ES.

3.2.2. Costs: The simulator associates different time costs with each successful or failed operation. Thus, a job that starts after two rejections will incur the costs of two rejections and one successful submission, plus the time for movement and queue slot allocation. Overall, the following time intervals are counted during a job submission.

- time to enter the planner queue (one simulator step)
- time for site assignment: queue computations (one simulator step or time to wait for an available site)
- time for site transfer: network allocation and transfer for the executable (at least one simulator step)
- time for node assignment: queue computations (one simulator step or time to wait for an available node)
- time for job transfer: network allocation and transfer for the executable (time determined by data volume and network capacity, but at least one simulator step)

GangSim can also model costs associated with various components. So far, we model only LS latencies, as discussed in Section 6.1.

3.2.3. Simulator steps: GangSim is a discrete simulator, which means that every X seconds the simulator evaluates the state of all components in the system (jobs, queues, resource status, allocations, utilizations, etc). Each operation takes place only during the evaluation steps. If a jobs has a running time of n*X + X/2, the job will actually occupy a computational node for (n+1)*X seconds, but will use only n*X + X/2 CPU power seconds. As a consequence, the smaller the value of X, the higher the accuracy of the simulator. We have typically used a value of 10 seconds in our seconds, except that when the size of the simulated environment was large (hundreds of sites and dozens of VOs), we increased the simulation step to 30 seconds to accelerate simulation. Such a value for parameter X appears to be acceptable as long as an average job running time is greater than or equal to a few hundred seconds. Each job is submitted to an ES queue. From the ES queue it is scheduled to a site and transferred to the site queue. Afterwards, the site queue manager schedules the job to an execution site where it actually runs. The ES can implement both a global and a partial knowledge of the system, while each LS has only partial information about the state of the entire system. In all experiments described in this paper we simulated only the cases with ESs having global knowledge about the system.

3.2.4. Site Usage Policies (UPs): UPs are expressed as tuples. Under commitment policy, an UP statement considers two upper limits, the epoch limit R_{epoch} and the burst limit R_{burst}. For each limit, there is associated a time interval, T_{epoch} and, respectively, T_{burst}. A job is admitted if and only if (a) the average resource utilization for its VO is less than the corresponding R_{epoch} over the preceding T_{epoch}, or (b) there are idle resources and the average resource utilization for the VO is less than R_{burst} over the preceding T_{burst}. For example, if a VO has (3600, 10), (60, 20) as allocation, then its jobs cannot average more than 10% over an hour or 20% over a minute, but they can spike up to 100% for a brief period (say 10 seconds). This approach is also implemented by the Maui scheduler [17]. A language-based specification is still on the future work agenda for the simulator.

3.2.5. ES task assignment strategies: The effectiveness of a specific UP may also depend on the strategies used by ES components. So far, we have only experimented extensively with dynamic policies that take into account different site loads [10] and are steered by usage policies in picking the *best* site for a job. We are currently also exploring other approaches.

4. Implementation Details

GangSim replaces Ganglia reporters with specialized modules that model the appropriate environment components, using different algorithms as described above. In addition, we have developed several other tools for workload specification and Grid environment generation. Site usage policies can be either specified through a usage policy web interface in a centralized manner (once per simulator instance), or described in a configuration file with a higher level of granularity (once per site). Each component has the following functionalities:

- *Simulator Modules*: a set of Perl modules that keep track of grid workloads and generate appropriate metrics for feeding collectors. These modules simulate ESs, submitting hosts, LSs, and sites. MDP, the central module in Figure 2, is connected to these simulator modules in place of (or, in a mix-mode execution, as well as) real ESs, LS, S-PEPs, and V-PEPs.
- *Task Assignment Policies*: a set of algorithms that can be invoked for scheduling jobs to sites, scheduling jobs to nodes, or for selecting jobs to run. These algorithms are called from various components and places in Figure 2: S-PEPs, LSs, V-PEPs, and ESs.
- *Metric Aggregators*: a set of routines that aggregate metrics based on rules such as, string concatenation, integer median computation, or integer averaging.
- *Grid Components*: a set of functions and data structures for simulating Grid components. For example, queues are represented by arrays of structures for various metrics about jobs; sites are modeled as a list of physical node capabilities and instantaneous states; workloads are also maintained under various queue structures while they pass from one stage to another in the simulated environment.
- *Environment State Keeper*: a set of data structures that hold data used for system simulation. Stored information is mostly about workload status, grid component status, and current utilizations.
- *Interface*: a set of CGI scripts that gather GangSim status and present in a HTML form.

Simulation results are accessible either directly from tables saved by the simulator, or through a web interface. The web interface offers a simple and easy way to browse and view statistics about various components in the simulated environment. There are three main screens, the site view, the VO usage policy view and the planner level view. Each of these views has associated many sub-views for a particular component monitoring. For example, a user can inspect how a planner schedules jobs to a site.

5. Achievable Results

In order to detail the simulation features of GangSim, we focus in this section on a few of the many results we have obtained with the system. We consider that each site has a number of CPUs, and each VO a number of groups that submit workloads. We use synthetic workloads to evaluate our usage policies. Each workload is composed of jobs, each corresponding to a certain amount of work and with precedence constraints determining the order in which jobs can be executed. Jobs arrive, are executed, and leave the system according to a Poisson distribution [8,9,10]. In all figures of this section, X axes represent simulated time interval, and Y axes represent CPU utilization percentages. The CPU utilization is defined as the total number of CPUs working on a job, while utilization percentage as the ratio of used CPUs to the total number of CPUs. For consistency, all simulations were performed over one hour and samples were taken every 10s.

5.1. Task Assignment Policies and Workloads

We present results for two types of workloads. In the first "synchronized" case, all VOs submit their jobs in bursts at approximately the same moment in time, while in the second "unsynchronized" case they submit their burst workloads at different moments in time. We consider these two scenarios important because they capture two common resource loads that can occur in practice: the moments before a conference dead-line, and repetitive workloads. The workloads overlay work for several VOs, each consisting of two groups. Each workload consists of 200 single jobs with a 250s average running, while the total running time was about 3000s. We consider that these workloads resemble closely for example the BLAST workloads that are run on the Grid3 resources. The interval among submissions was 700s in average.

The results captured in Figures 5-8 represent job executions and overall site utilizations when site grant VO resource requests using a simple FIFO strategy. Each task assignment policy schedules jobs on all sites with available resources. We note that the least used task assignment policy performs slightly better than the others, although a discussion of why is beyond the purpose of this paper. We present detailed analyses of such results elsewhere [9,10].

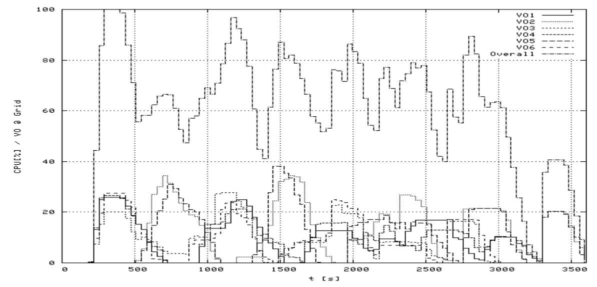

Figure 3: Round Robin Assignment Policy

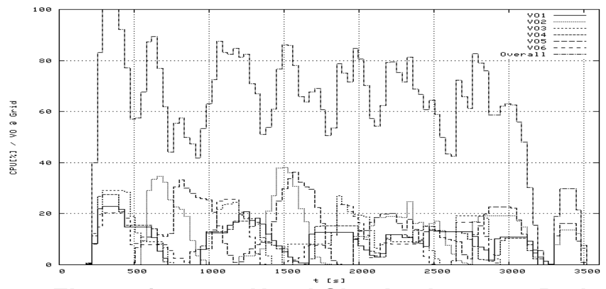

Figure 4: Least Used Site Assignment Policy

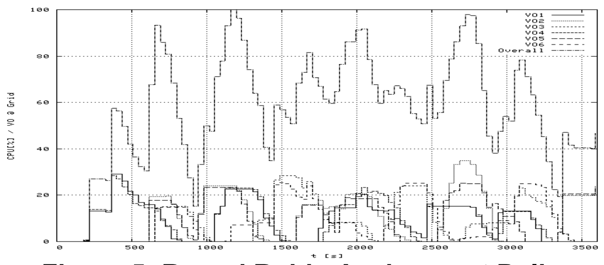

Figure 5: Round Robin Assignment Policy

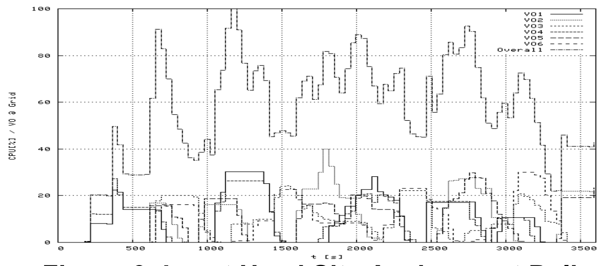

Figure 6: Least Used Site Assignment Policy

GangSim computes a few performance metrics automatically [10]. For example, we define the *average response time* (ART) for a VO as follows, with RT_i being an individual job response time, i.e., the time that elapses from job submission to job completion:

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

Table 1 captures ART values for the synchronized workloads presented above, while Table 2 captures the same metric for un-synchronized workloads.

Table 1: Synchronized Workloads – ART

Policy/Limit	No limit	Fix-limit	Ext-limit
Round Robin	11.09	19.39	11.32
Least Used	13.25	15.14	15.06

Table 2: Unsynchronized Workloads – ART

Policy/Limit	No limit	Fix-limit	Ext-limit
Round Robin	7.78	14.82	9.34
Least Used	10.57	13.68	11.37

These examples are illustrative of what GangSim can do. Other task assignment policies can be included, combined with various usage policies and data scheduling techniques [9,10].

5.2. Simulated Architecture Variations

GangSim can also model job scheduling decisions based on past observations about site behaviors. Such approaches can be important in situations where a site does not publish some or all of its usage policies. Our solution for such cases is to fall back to collected metrics about a site's performance. GangSim maintains two decoupled state structures about job execution, one at the site level that conforms to local policies and one at the VO level that conforms to job measured performance metrics and job queue times. We have also incorporated a performance characteristic associated with each site that controls "how fast" a site should respond to a new job submission request, as well as, "for how long" a job should remain queued before being placed in execution.

The first approach proves to achieve a higher job completion and site utilization (Figures 9 and 10) in the case of round robin job assignment policy. A detailed discussion of this observation is beyond the purpose of this paper is provided elsewhere [10,19].

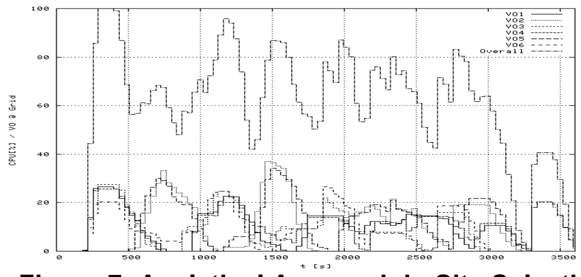

Figure 7: Analytical Approach in Site Selection

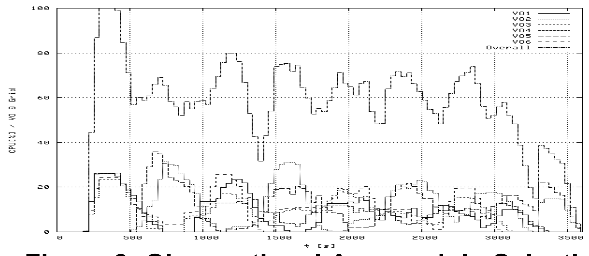

Figure 8: Observational Approach in Selection

5.3. Simulator Performance

Another important aspect of this work is the scale at which the simulator can still provide good results in the expected running interval. In Figure 9 we present a case involving 15 VOs and 100 sites. Scaling beyond this point overloads the simulator host and results become less accurate.

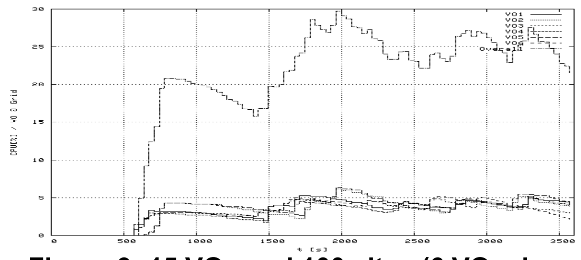

Figure 9: 15 VOs and 100 sites (6 VOs drawn)

GangSim can also be configured to run in a distributed mode in which several simulator instances run on different hosts, with sites and computational load distributed appropriately. This feature has the potential to provide greater scalability. However, we have not yet evaluated whether this potential can be achieved in practice.

6. Simulations vs. a Real Grid

A critical concern for a simulator such as GangSim is whether or not it provides an accurate simulation of the sorts of Grid environments encountered in practical situations. Thus, we have started a major effort aimed at comparing simulation results with the results of comparable experiments performed on a set of Grid3 sites [2]. We present preliminary results of this evaluation in this section. In our view, these results are encouraging in that they show that we can capture gross properties of Grid operation. However, they also make clear that we are far from having an accurate simulation of the Grid environment, due primarily to various idiosyncratic features of Grid3 LS and ES components that are not captured in our current GangSim models.

6.1. Basic Operations

As mentioned in Section 3.2.2, each state transition for a job incurs a cost. We are currently conducting experiments on Grid3 to determine costs for basic operations, so that accurate values for these costs can be incorporated into GangSim. In a first set of such experiments, we have measured the average delay incurred between submitting a job to a Grid3 site and that job starting execution; we observe the surprisingly high value of 321 seconds.

6.2. Site Level Comparison

We next compare GangSim and Grid3 within the context of a single site. For this comparison, we focus on FermiLab, a cluster running the locally developed FBSng as its local scheduler. (FBSng is a batch system for job management and load balancing). Out of 108 CPUs, only 50 were available.

For both GangSim and Grid3 we used four identical workloads, each comprising 120 jobs with an average runtime of 50 seconds and a standard deviation of 5 seconds. Jobs had data dependencies, with every eighth job requiring the data of all previous seven. Each workload was associated with a different VO, and started at a different time, as follows: iVDGL at 20 seconds, BTEV at 200, USATLAS at 400, and LIGO at 700.

The Grid3 ES used in this study implements the following scheduling policy: for each job, it first prefers sites with free CPUs and un-filled policy; then sites with free CPUs and extensible policy (e.g., Condor, but not PBS, which has with hard limitations); and finally sites with no free CPUs but un-fulfilled policy. If no site satisfies any of these criteria, it holds the job at the submission point. The Grid3 ES also implements a rescheduling policy, to deal with the fact that sites sometimes seem to hold jobs in the queues even though their policies suggest that they should be runnable. Specifically, if a job is pending for more than 20 minutes, the job is removed and returned for rescheduling.

The ES strategy implemented by GangSim differs from that just described in one important respect: if no site has free CPUs, then jobs are held at the ES rather than submitted to sites with un-fulfilled policy. In addition, GangSim does not provide for rescheduling, as this is not necessary given that GangSim's simulated sites always obey their policy correctly. Figures 12 and 13 present results obtained on Grid3 and our simulator, respectively. (All graphs presented in this section show CPU utilization percentages, as introduced in Section 5, as a function of time in seconds. Note that percentages in this case are relative to the 108 total CPUs.)

The GangSim and FermiLab executions both completed in close to the same time, but show rather different execution behavior. We are currently studying the reasons for these differences, with the goal of improving the simulator (or, potentially, the real implementation). Both GangSim and FermiLab schedule jobs under an equal fair-share strategy, but clearly there are aspects of the FermiLab scheduler that are not captured in GangSim.

We note that 5% of the FermiLab jobs failed for various reasons and thus had to be resubmitted. We should presumably be simulating such job failures in GangSim.

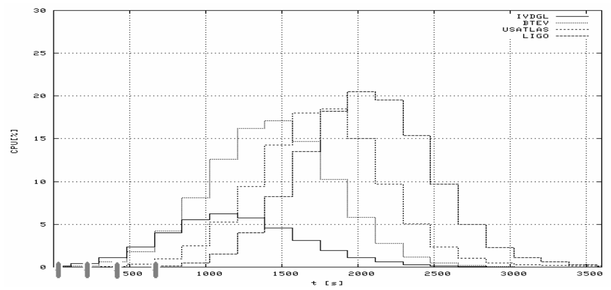

Figure 10: Per-VO, FermiLab (Grid3)

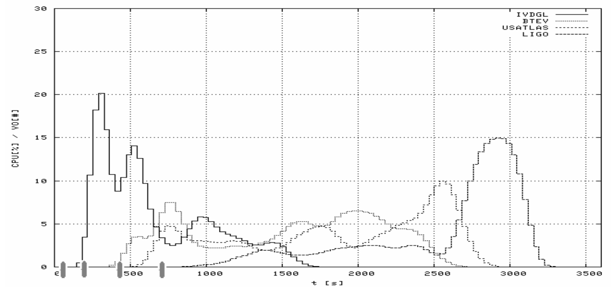

Figure 11: Per-VO, FermiLab (GangSim)

Other Grid3 sites use one of three other schedulers: Condor, Open-PBS/PBS-Pro, and LSF. Condor enforces an extensible fair-share policy [13], Open-PBS has its own language for describing various policy configurations [14], and LSF implements a hierarchical fair share strategy [15]. We have not yet studied simulation and experiment for these other schedulers, but will do so in the final paper.

6.3. VO Level Comparison

We next compare GangSim and Grid3 runs across 12 sites with a total of 1000 CPUs, of which 300 are available. For both GangSim and Grid3 we used eight identical workloads, each comprising 300 jobs with an average runtime of 150 seconds and a standard deviation of 50 seconds. Each workload was associated with a different VO and group, and started at different times: iVDGL-1 at 20 seconds, BTEV-1 and USATLAS-1 at 200, LIGO-1 at 700 sec, BTEV-2 at 800, iVDGL-2 at 1000, USATLAS-2 at 1500, and LIGO-2 at 1700.

Figures 14 and 15 present results obtained on Grid3 and GangSim, respectively, for aggregate CPU allocation per VO across all sites as a function of time in seconds. Once again, we see similar aggregate behavior but also major differences between Grid3 and GangSim, which we are currently working to explain.

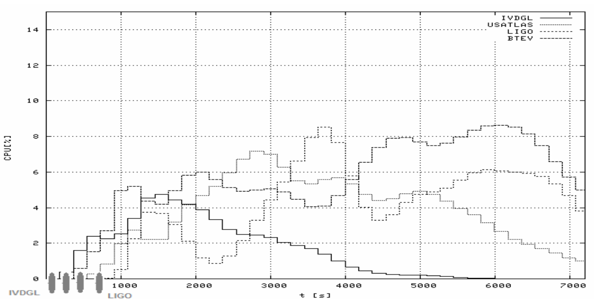

Figure 12: Per-VO, 12 sites (Grid3)

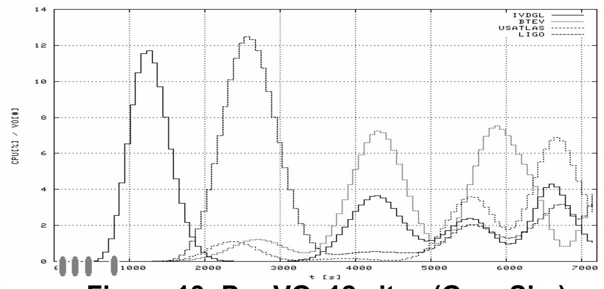

Figure 13: Per-VO, 12 sites (GangSim)

In Figures 16 and 17, we focus in on a single VO, and show aggregate resources obtained by the two iVDGL groups on Grid3 and GangSim, respectively. Each group had a pre-specified usage limit and jobs were scheduled under a FIFO policy.

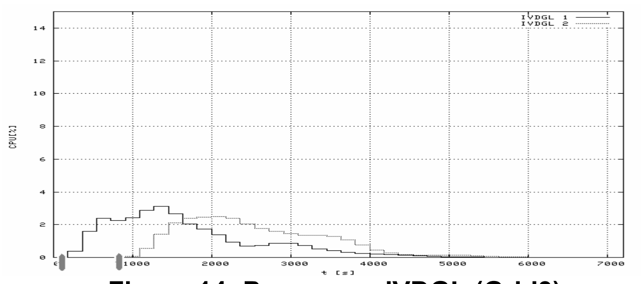

Figure 14: Per-group, iVDGL (Grid3)

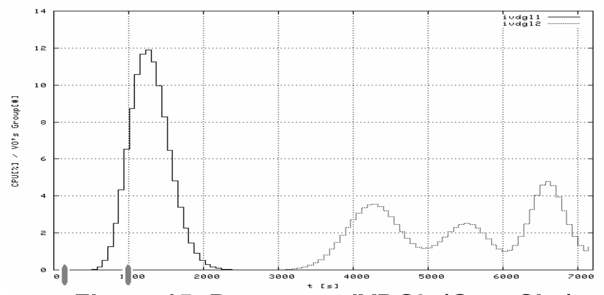

Figure 15: Per-group, iVDGL (GangSim)

6.4. Quantitative Comparison

We use three metrics to characterize and compare Grid3 and simulator performance: aggregated resource utilization, average response time, and average starvation factor. We define the *aggregated resource utilization* (ARU) as the ratio of the CPU resource actually consumed by users (ET_i) to the total CPU resources available. We define the *average response time* (ART) for an entire VO as follows, with RT_i being the individual job response time:

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

We define the *average starvation factor* (ASF) as the ratio of the resources requested and available, but not provided to a user, to the CPU resources consumed by the user (ET_i). We compute this quantity as follows, where ST_i is the CPU resource requested by a user but not provided, and RT_i is the total resources available:

$$\mathbf{ASF = \Sigma\ (\ MIN\ (ST_i, RT_i)\ )\ /\ \Sigma\ (ET_i)}$$

We present in Table 3 these three metrics as computed for the runs of Figures 14 and 15. The "site" column captures the metrics as measured for "FermiLab," as considering in Section 6.2, while the "VO" column captures the same metrics aggregated over the entire grid.

Table 3: Simulation (S) vs. Grid3 (G) Metrics

Level	Site		VO	
Metric	S	G	S	G
ARU	0.12	0.16	0.07	0.06
ASF	2.36	3.9	9.88	5.09
ART	1521.31	1100.7	1824.25	639.5

Important differences between our simulator and Grid3 are ART and ASF, which differ by a factor of almost two. We clearly need to investigate further to provide a complete explanation for these differences.

7. Related Work

ChicSim is a modular and extensible discrete event Data Grid simulation system built over Parsec that has been used to evaluate a wide variety of scheduling and replication algorithms [20]. Like GangSim, ChicSim models a Grid as a collection of sites. However, ChicSim does not include notions of VOs or groups and has no support for site usage policies. Similar comments apply to MONARC [21], a simulator developed to evaluate the performance of data processing architectures for physics data analysis, and GridSim [23], which models various components of distributed systems, but does not address the representation and evaluation of policies.

MicroGrid is a tool used to develop and implement a virtual grid infrastructure that can provide a convenient vehicle for scientific study of grid resource management issues. This tool enables repeatable, controllable experimentation with dynamic resource management techniques. Two types of experiments can be done on the MicroGrid, normal applications without Globus®, and Globus applications [22]. Again, MicroGrid does not address policy issues.

8. Conclusions

We have presented the design, implementation, and preliminary evaluation of GangSim, a Grid simulator that supports the analysis of different scheduling policies in a multi-site and multi-VO environment. The GangSim design combines discrete simulation techniques and modeling of important system components (calibrated by experiments on real Grids) to achieve scalability to Grids of substantial size. We demonstrate GangSim's use by describing studies of different VO-level scheduling policies in the presence of different local site resource allocation policies.

Comparisons of GangSim with equivalent runs performed on Grid3 suggest that our simulator captures some aspects of realistic Grid behavior but fails to capture detailed behaviors of local schedulers, local user jobs and other issues that tend to reduce performance in today's practical deployments. One may reasonably ask to what extent such "features" of today's Grid sites need to be captured in a simulator such as GangSim. However, we believe that it is important to be able to demonstrate better correlation with Grid3 results, and we plan to do so in the final paper.

Acknowledgements: We would like to thank Robert Gardner, Marco Mambelli, Michael Wilde, and Jens-S. Vöckler for their insights and discussions in support of this work. We also thank the Grid3 project. This work was supported in part by the NSF Information Technology Research GriPhyN project, under contract ITR-0086044.

9. References

[1] Avery, P. and Foster, I. "The GriPhyN Project: Towards Petascale Virtual Data Grids", 2001, www.griphyn.org.

[2] Grid2003 Team, "The Grid2003 Production Grid: Principles and Practice", *Proc 13th IEEE Intl. Symposium on High Performance Distributed Computing, 2004.*

[3] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S., "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets", *J. Network and Computer Applications* (23), '01.

[4] Ranganathan, K. and Foster, I., "Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications", *International Symposium for High Performance Distributed Computing*, Edinburgh, UK, 2002.

[5] Foster, I., Kesselman, C., Tuecke, S., "The Anatomy of the Grid", *International Supercomputing Applications*, 2001.

[6] Verma, D.C., *Policy Based Networking, Architecture and Algorithm*, New Riders Publishing, November 2000.

[7] Kosiur, D. *Understanding Policy Based Networking*, Wiley Computer Publishing, 2001.

[8] De Jongh, J. F. C. M., "Share Scheduling in Distributed Systems", Delft Technical University, 2002.

[9] Dumitrescu, C., Wilde, M., and Foster, I., "Policy-based CPU Scheduling in VOs", GriPhyN Tech. Report, 2003-31.

[10] Dumitrescu, C., and Foster, I., "Usage Policy based CPU Sharing in Virtual Organizations", IEEE/ACM *Grid Workshop*, 53-60, Pittsburgh, 2004.

[11] Dumitrescu, C., and Foster, I., "VO-Centric Ganglia Simulator", GriPhyN/iVDGL TechReport, 2004-31.

[12] Massie, M., Chun, B., Culler, D., "The Ganglia Distributed Monitoring: Design, Implementation, and Experience", *Parallel Computing*, May 2004.

[13] Condor Team, "A Resource Manager for High Throughput Computing", Software Project, The University of Wisconsin, www.cs.wisc.edu/condor.

[14] Open-PBS Team, "A Batching Queuing System", Software Project, Altair Grid Technologies, LLC, www.openpbs.org.

[15] LSF Administrator's Guide, Version 4.1, Platform Computing Corporation, February 2001.

[16] Foster, I., Fidler, M., Roy, A., Sander, V., Winkler, L., "End-to-End Quality of Service for High-end Applications", *Computer Communications*, 27(14):1375-1388, 2004.

[17] MAUI, Maui Scheduler, Center for HPC Cluster Resource Management and Scheduling, www.supercluster.org/maui.

[18] Dumitrescu C., Wilde M., and Foster I., "Usage Policy at the Site Level in Grid3", GriPhyN Technical Report, 2004-71.

[19] Dan, A., Dumitrescu, C., Ripeanu, M., "Connecting Client Objectives with Resource Capabilities: An Essential Component for Grid Service Management Infrastructures", *ACM International Conference on Service Oriented Computing (ICSOC'04)*, New York, 2004.

[20] Ranganathan, K., "The Chicago Grid Simulator", people.cs.uchicago.edu/~krangana.

[21] Models of Networked Analysis at Regional Centers for LHS Experiments, www.cern.ch/MONARC.

[22] Emulation Tools for Computational Grid Research, www-csag.ucsd.edu/projects/grid/microgrid.html

[23] Buyya, R., "GridSim: A Grid Simulations Toolkit for Resource Modeling and Application Scheduling for Parallel and Distributed Computing", www.buyya.com/gridsim.