

## Usage Policies at the Site Level in Grid

**Catalin L. Dumitrescu**

*Department of Computer Science  
The University of Chicago  
catalind@cs.uchicago.edu*

**Michael Wilde**

*Mathematics and Computer Science  
Division, Argonne National Laboratory  
wilde@mcs.anl.gov*

**Ian Foster**

*Mathematics and Computer Science Division, Argonne National Laboratory  
Department of Computer Science, The University of Chicago  
foster@mcs.anl.gov*

### Abstract

In the Grid3 [Grid3] context, resource *owners* grant to different Virtual Organizations (VOs) the right to use their resources. For example, scientists might provide access to their computers, storage, and networks to different scientific collaborations, which might then use those resources for large-scale data analysis tasks of interest. In industry, such scenarios can arise when resource consumers and providers engage in outsourcing service level agreements (SLAs).

Complex usage policy issues can arise in such scenarios. Owners want to express (and enforce) some usage policies under which resources are made available. VO representatives want to access and interpret policy statements published by resource owners to monitor agreements and guide their activities. Thus, both owners and VOs want to verify that policies are applied correctly. We address here the expression, publication, discovery, interpretation, enforcement, and verification of usage policies at the site level. We also describe our solutions and approaches developed over time in the Grid3 context, namely the site policy observation points (S-POPs) and site policy enforcement points (S-PEPs).

### 1. Introduction

In the scenario where resource owners want to grant to different virtual organizations (VOs) the right to use their resources, owners want to express and enforce different policies under which resources are made available as well. The problem we are taking on is the UP based resource sharing in Grid3 [Grid3] and layer down several mechanisms and solutions how a site can express, publicize, and enforce usage policies [ResourceAllocation,CPUAllocation]. Grid3 is a grid composed of about 30 sites that share resources to

achieve various VO goals. When there is no contention, all VOs get the resources they want. But when Grid3 is busy with VOs running their many workloads, contentions among workloads are a real issue seen in practice [Grid3,GriPhyN]. Another motivation is that different sites are supported or part of certain VOs, which want most of their resources when they need them. In this scenario where VOs compete for resources, each site wants to impose certain usage policies, not only security mechanisms that guarantee free access to the site.

The rest of the document is structured as follows. In section 2, we describe our approach in site usage policy specification for Grid3, and in the following section present the mapping of the S-PEP model to the specific RM models of Grid3. In section 4 we document what the schedulers do and how their behavior is controlled by specific configuration file entries. Section 5 contains various technical details about the UP components, future work and Grid3 gains. The experimental results gathered so far are incorporated in section6, and we end the document with similar work pursued in the Grid community.

### 2. Levels of Usage Policy

From our experience, site owners want convenient and flexible mechanisms for expressing the policies that determine how many resources are allocated to different purposes, for enforcing those policies, and for gathering information concerning resource usage. Particular concerns may include achieving the policy flexibility needed to meet different and time-varying demands, and being able to resolve disputes as to whether policies were correctly enforced [CPUAllocation]. Next, we detail how usage policies are handled at sites in Grid3.

## 2.1 Roles

There are two levels in our model where UPs are handled: site level and grid level. At the site level, resource owners state how their resources must be allocated and used by different VOs. This represents the *high-level GOAL a site's owner or manager policy (MP) wants to achieve*. The site administrators map MPs to different software RMs' syntaxes. The end product is the set of RM configuration files, named the *local POLICY or system configuration (SC)*.

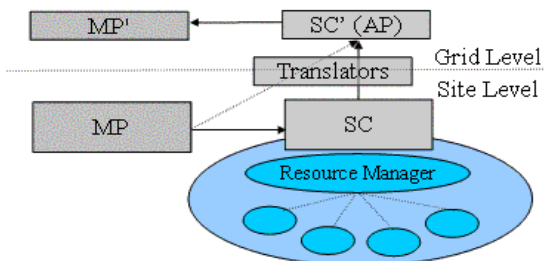
At the grid level, UPs are translated from SC by automated tools into an *abstract policy (AP)* set. SC descriptions are collected from the site RM configurations, filtered and, after translation, published through any standard Grid monitoring system, e.g., MDS or MonaLisa. [Condor,PBS,MAUI,ResourceAllocation].

## 2.2 Usage Policies

To simplify the example, we consider in this paper only what a site does from the perspective of a particular VO, e.g., "site X gives ATLAS 30% over a month". Thus, we have identified as important three places and descriptions for the UPs:

- MP: a description of what the site manager wants the policy to be for the site, e.g., MP(VOs) = "give ATLAS 30% over a month". We assume that simple English statements describe the MP set (site level).
- SC: an RM configuration: SC(VO) = <number of nodes, scheduler-type, scheduler-config>, which is written by the site administrators during RM configuration process (site level).
- AP: SC(VO) translated into an abstract policy, AP(VO,Site), that expresses SC(VO) in some more abstract, scheduler-independent format and which we will publish through MDS for general-purpose usages (grid level).
- MP': monitoring info about site's performance in order to understand what happens in reality, whether MP'(VO,Site) and MP(VO) match or not.

The key point is determining how an SC(VO) maps to an AP(VO,Site). The information flow is captured in a graphical way in Figure 1.



**Figure 1: The correlations between MP, SC, AP and MP'**

The additional items in Figure 1 are:

- Translators: tools that reverse SC to MP, but to a Grid understanding and trying to abstract to a common RM model (AP)
- AP + MP' + translators + others = UP Service / Site Recommender

## 3. RM Heterogeneity Considerations

To formalize the translation process, we look at a collection of RMs and develop a common description of the schedulers, their scheduling algorithms, and their configurations. The sites we used for analysis are PDSF, UBuffalo, UChicago and UMN. At PDSF, the RM is LSF; at UChicago the RM is Condor; at UMN the RM is PBSPro, and at UBuffalo is Maui with PBS [Grid3]. On each of these sites, the Grid3 settings are as follows: all users from a VO are mapped to a local common ID, and the local RM is configured to provide that local ID a specific fair-share priority relative to other users. Under these settings, we consider the following simple scenario: "there are three types of incoming jobs to balance from: one from CMS, one from ATLAS, and one from iVDGL. Let's call them CMS-Prod (*cmsprod*), ATLAS-Prod (*atlasprod*), and IVDGL (*ivdgl*). We want CMS-Prod and ATLAS-Prod to get an equal share of available CPUs, but we only want IVDGL to get a small fraction if there's contention (a 10<sup>th</sup> of what the others get)."

### 3.1 CPU Management Solutions

In Grid3 local RMs manage CPU access by means of priority and queuing mechanisms, under different scheduling policies. Usually, local RMs decide based on user priority and the amount of consumed CPU (among other things) how much CPU time a user gets in the future.

**3.2.1 Condor as RM:** For Condor, the `condor_userprio` is the recommended way to set the "priority factor" for each of user on Grid3. The lower a user's priority factor, the better their allocation. In our scenario, since the default priority factor for all users is "1.00", it can be left for both *cmsprod* and *atlasprod* in our example. But for *ivdgl*, the priority factor has to be changed to 10, so that it gets a 10<sup>th</sup> as many available CPUs as either the other two [Condor]:

```
% condor_userprio -setfactor ivdgltest 10
```

Now, if there are 100 CPUs and all three experiments simultaneously submit 100 jobs, *cmsprod* and *atlasprod* should each get about 46 CPUs, and *ivdgltest* should get about 8. If anyone of them has

received more CPUs recently, they will be given less if new contention arises, until their priorities re-balances. How long this re-balancing takes to happen can be controlled by the PRIORITY\_HALFLIFE configuration parameter.

In a formal way, if we have N users with priorities  $P_i$ ,  $i=1..N$ , then the minimal CPU allocation for each user is computed in our S-POPs as follows:

$$CF_{i(t)} = 1 / [ P_{i(t)} * \sum (1 / P_{k(t)}) ]$$

with  $k = 1..N$ . When there is no contention (some users do not use their allocations), the CPU allocation value becomes:

$$CF_{i(t)} = 1 / [ P_{i(t)} * \sum (1 / \text{MIN}(P_{k(t)}, M_{i(t)} / M)) ]$$

with  $k = 1..N$ . The individual priority re-balancing with PRIORITY\_HALFLIFE is:

$$P_{i(t+1)} = P_{i(t)} * [ 1 + (M_{i(t)} - M * CF_{i(t)}) / (M_{i(t)} - M * CF_{i(t)}) / 2 ]$$

with  $k = 1..N$ , and  $t$  as time. In the previous formula,  $M$  represents the total number of machines, while  $M_k$  the number of machines a user uses at time  $k$ . [Condor]

**3.2.2 Maui as Scheduler:** The configurations for the Maui/PBS for the same example are as follows (maui.cfg):

```
# use fair-share
FSPOLICY                ON
# favor fair-share component
FSWEIGHT                10
# no CPU limit when no contention
MAXJOBPERUSERPOLICY    OFF

USERCFG[cmsprod]        FSTARGET=10.0
USERCFG[atlasprod]      FSTARGET=10.0
USERCFG[ivdgltest]     FSTARGET=1.0
```

In this case, a good approximation for N users with a standard configuration is:

$$CF_i = P_i / \sum (P_k)$$

with  $k = 1..N$ , and when there is contention. And when there is no contention and extensible fair share enabled:

$$CF_i = P_i / \sum (\text{MIN}(P_{k(t)}, M_{i(t)} / M))$$

with  $k = 1..N$ . Unless we have other advance reservation rules or more complex policies in place, the two previous formulas characterize pretty well the extensible fair share policy [PBS,MAUI]. We assume here that priorities are fixed and there is no balance with time. Maui provides a finer-grained way of

describing policies and we consider that the syntax is sufficiently expressive for VO allocations.

### 3.1 Disk Space Management Solutions

An important issue in Grid3 is similar local RMs for disk. A site admin could implement quotas for example, and the allocations can be reported the same through UP monitoring in Grid3.

**3.3.1 Unix Quota:** 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 condor adapts to make efficient use of CPU. However this is not particularly adaptive, and requires some amount of effort for the users to contact every site admin to request special treatment.

**3.3.2 SRM:** Several concepts are introduced to support disk space sharing and reservation on the Grid. Files and locations are classified as permanent, durable and volatile. In the first case the user has total control on how the files are managed, while in the second the space has a limited life-time and reclaimed when expires. The third case is similar with the second one, but all files are lost when the reservation expires [SRM].

**3.3.3 Privilege:** In this project, the storage system is considered an independent service that provides a uniform interface for authorization and finer granularity operations at the disk level. Such operations include obligation policies (e.g., time to live, owner requirements). The process is quite complex, and involves negotiations among three different modules: GUMS, SAZ and OGS Auth. Service. The Storage Authorization Service interface introduces a well specified interface for operations request, which makes the system valuable in terms of portability. The project is still in its infancy and work is left until it is accepted on most Grid3 sites [Privilege].

As a conclusion, the only solution in our view for immediate deployment is the quota based disk management, which is available on any UNIX-based system.

## 4 Abstract RM Model

Now, we build our model starting from the observations that a Grid3 RM manages several worker nodes or some amount of resource capacity with time, the assignment policy is considered fixed, one from a predefined set of scheduling policies (fair-share, priority-based, opportunistic scheduling policies, or fixed allocation for disk). The other objects that the

RM manages are users and groups, depending on the scheduler implementation. The UPs for this abstract RM are expressed as usage time percentages or quotas.

Decoupled from the CPU management, is the space management.

#### 4.1 UP Syntax Example

To give intuition fast, we provide a concrete UP example, expressed in all the three forms introduced above, namely MP, SC and AP. We assume the following MP set for site X is:

- we have a cluster with 380 CPUs
- at any time: CMS has a 30%; ATLAS has a 30%; IVDGL-Test has just 10%
- when additional resources are available, Grid3's VOs can grab these resources

The Condor priorities (SC) used to realize the above description is:

- condor\_userprio -setfactor cms 2
- condor\_userprio -setfactor atlas 2
- condor\_userprio -setfactor ivdgltest 9

The Maui allocations (SC) are:

- USERCFG[cms] FSFACTOR=30
- USERCFG[atlas] FSFACTOR=30
- USERCFG[ivdgltest] FSFACTOR=10

In these settings, the Grid AP description becomes:

- RM type: Condor
- RM allocations: CMS: 30% ATLAS: 30% IVDGL-Test: 10%
- UP type: VOTFS

#### 4.2 Disk Space Extension

For disk space, the same allocations will be translated into the following rules (UNIX quota):

- edquota -u cms
- edquota -u atlas
- edquota -u ivdgltest

The Grid AP for disk becomes:

- DM type: quota
- DM allocations: CMS: 30% ATLAS: 10% IVDGL-Test: 10%
- UP type: VOTFS

This simple specification introduces practically two types of storage, the permanent storage (the hard limit quota limit) and volatile storage (the soft limit quota limit), as introduced by SRM project.

## 5 Technical Details and Grid3 Gains

At the site level, we introduce a specialized UP monitor. Because we consider no direct access to the MP description, we focus instead on the SC descriptions. These statements are Condor priorities, Maui allocations, LSF rules, quota rules or Privilege allocations. In the next step, these elements are translated locally to the uniform AP description and published into MDS.

### 5.1 UP Components

Instead of a complex S-PEP, we decoupled its functionalities. The enforcement is done thru the local RM, by means of several configuration rules. The monitoring and reporting (a.k.a. S-POP) is done through a few MDS providers. These MDS providers collect SC statements from the site RMs. The statements are Condor priorities, or Maui allocations. In the next step, these rules and quotas are translated at each site to the uniform AP description and published through MDS. As can be seen from the next section, there is a need for an S-POP and S-PEP at some sites, while others can be based on what current RMs offer.

To simplify the work of the site administrators in setting up UPs, we introduced also an UP console element [ResourceAllocation,S-POP.v3.0]. Through the console, site administrators can get easily translations from MP or AP to SC for usage policies.

### 5.2 Grid3 Gains

The main gains for the Grid3 users are:

- Additional usage information: it gives grid planners, like Euryale [Euryale], Pegasus [Pegasus] or Sphinx [Sphinx], hints about what sites to consider for job placement. For example, even if a site might look busy from a monitoring point of view, the current VO is still entitled to some amount of computing power.
- Time-based entitlement to resources: VOs are guaranteed under different FS policies that they get resources when they need them instead of maintaining constant workloads in order to burn their allocations.

## 6 Experimental Results

In order to measure the effectiveness of presented approach, we compare current mechanisms for UP specification with a continuous monitoring solution implemented earlier in with the VO-Centric Ganglia simulator [VO-Ganglia]. We focused on measuring how well RMs, such as Condor and Maui in conjunction with OpenPBS, are able to enforce the

extensible UP specification. Our interest was motivated by the necessity to be as non-intrusive as possible at the site level in Grid3.

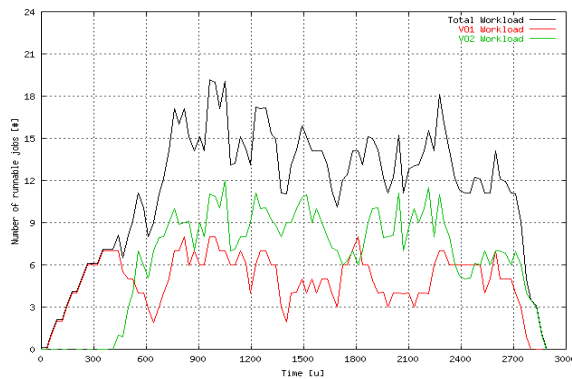
## 6.1 Continuous CPU Policy Enforcement

The S-PEP solution proposed initially was entirely external to the local schedulers, communicating with them via their interfaces. The S-PEP performed policy enforcement by invoking control commands specific to each scheduler to suspend, stop, and resume jobs.

We measured for a test case consisting of two VOs submitting randomly generated workloads to a single site Site<sub>1</sub>, and enforcing a usage policies in which overall CPU resources of Site1 were allocated 20% to VO<sub>1</sub> and 80% to VO<sub>2</sub>. The two VOs were allowed 30 second burst utilizations of 60% and 90% of the site's CPU resources, respectively:

```
[CPU, Site1, VO1, (3000s, 20%), (30s, 60%)]
[CPU, Site1, VO2, (3000s, 80%), (30s, 90%)]
```

Jobs were submitted to the scheduler via the Globus Toolkit™ 2.0 and we performed experiments with two different local schedulers, Condor and OpenPBS. The workload for this test consisted of jobs submitted by two virtual organizations. The workload is illustrated in Figure 2 which shows job arrival times and durations assuming ideal job execution times without queuing delays.



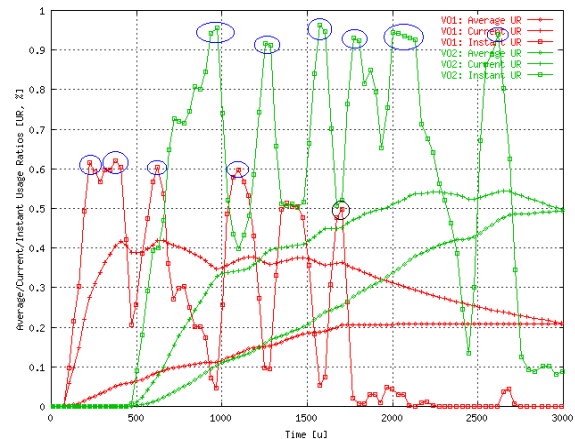
**Figure 2: Two VO workload for S-PEP measurements**

The test site was monitored by collecting load information every 10 seconds. The S-PEP performed policy enforcement actions every 30 seconds. The experiment ran for almost one hour.

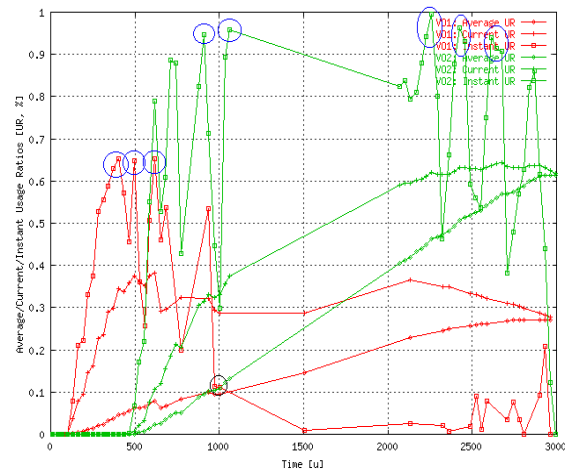
Figure 3 and Figure 4 show instantaneous, instantaneous average and average utilization of Site's CPUs over the test period for Condor and OpenPBS, respectively.

In these tests, S-PEP enforcement occurred several times, as represented by the blue and black circles. Blue circles represent job stopping or postponing

(when the burst CPU-usage limit was reached), while black circles represent the point at which all jobs for the virtual organization were deleted from the site queues (because the allocated CPU-usage limit for the major time period was exhausted). The fine-grained control actions of adjusting job priorities are not shown in the graph.



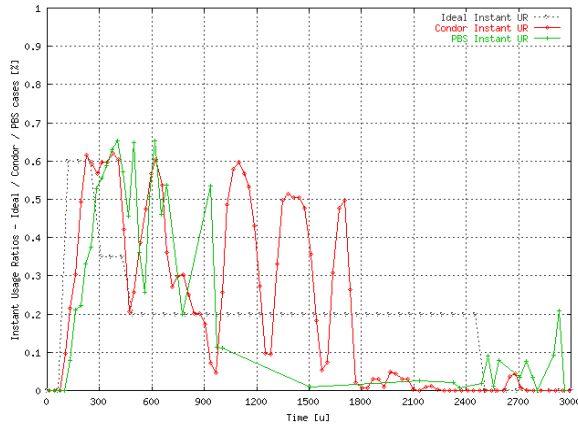
**Figure 3: Policy enforcement results for the Condor scheduler. X: time in secs; Y: Utilization of site processors. S-PEP enforcements are circled**



**Figure 4: Policy enforcement results for the PBS scheduler**

For comparing the accuracy of our S-PEP module, we have overlaid on a single graph the *Instantaneous Usage Ratios* for Condor and OpenPBS cases for one only (Figure 5). We observed that the policy enforcement module had similar effects for both Condor and OpenPBS at a coarse level of comparison. The ideal curve (shown in gray) cannot be achieved, due to the latencies incurred in submitting processes to site schedulers, and the subsequent scheduling delay. In addition, the monitoring sub-component achieves

different behaviors, being influenced by the capacity (or in-capacity) of the tested scheduler to return job information in a timely fashion.

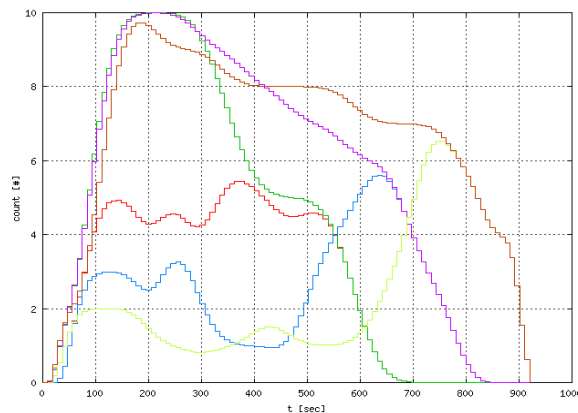


**Figure 5: Processor utilization levels for “ideal”, Condor, and PBS schedulers**

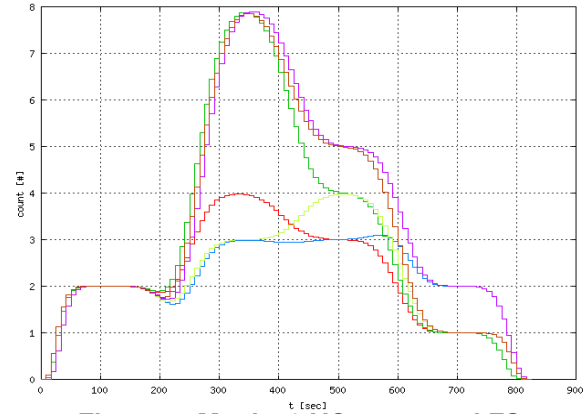
### 6.2 Local RM UP Enforcement

The test site was monitored by collecting Condor load information every 10 seconds. The experiment ran for 3000 seconds. Figure 6 and Figure 7 show instantaneous utilization and workload queued of Site’s CPUs over the test period for Condor and Maui, respectively. The three VOs had the following allocations: VO<sub>1</sub>: 50%, VO<sub>2</sub>: 30%, and VO<sub>3</sub>: 20%.

We observed that the policy enforcement of the RM had different effects for Condor and Maui at a coarse level of comparison. Maui behaved somehow differently due to other parameters that tried to achieve the “fair scheduling” policy.



**Figure 6: Condor – 3 VOs, unequal FS**



**Figure 7: Maui – 3 VOs, unequal FS**

Comparing these results with the ones in the previous subsection, we conclude that the continuity of enforcement at the local RM level is higher, while the external S-PEP module assures similar behaviours in all cases.

### 6.3 Grid3 Advantages

In order to prove the importance of taking in account UPs at different sites, we have conducted additional experimental tests on the Grid3 testbed. Based on the previous concepts and elements, we have introduced a specialized UP service that uses the APs and resource utilizations gathered by the S-POPs from each individual site. The UP service creates and maintains an internal image of the Grid3 resources and answers queries such as: “From the list *L* of sites, which is the subset *S* of sites where VO *V*’s workload is allowed to run?”, then “Which is the best site *X* to send VO *V*’s workload?”. The criteria for \*best\* site is in itself a complex problem and we focused so far on a few strategies: # of free CPUs, #of allocated CPUs, free disk space and site’s availability. The results we have gathered are captured in Table 1 and Table 2 [CPUAllocation].

**Table 1: Achieved Results / sync**

	ART	ARU	AJC
RA/NP	97.01	0.017	548
RR/NP	124.99	0.017	549
RA/SC	180.43	0.035	695
RR/SC	126.70	0.036	655

**Table 2: Achieved Results / un-sync**

	ART	ARU	AJC
RA/NP	69.19	0.017	551
RR/NP	74.53	0.015	554
RA/SC	78.22	0.028	630
RR/SC	118.10	0.027	603

In all cases our site recommender was used to provide site recommendations. The difference is that in the /NP cases recommendations were given based only on the free CPUs as reported by the monitoring components, while in the /SC case recommendations were given based on the usage policies and priorities enforced at sites as well.

The number of jobs completed suggests the second approach gives a better job execution, even though the response time seems to be higher in most cases. From these results, we consider no need for further investigation of the importance of our UP service as long as it proved the difference between the two cases.

## 7 Related Work

There are several related streams of work that target the problem of usage policy at the site level. Next, we mention succinctly three of them, while other examples not mentioned here can be easily considered as related work.

The community authorization service (CAS) represents a similar solution for authorization purposes. A CAS server plays the role of a trusted third party that is responsible for managing both resource and VO policies that govern access to community's resources. The CAS server was designed to maintain entries for CAs, users, servers, resources and groups of entities. It also contains entries for policy statements that specify *who*, user or group, *which* resource or resource group the permission is granted on, and *what* permission is granted. Each CAS server is an independent instance ran by a VO that imposes a set of common interpretations of service's actions for all resources that recognize that server [CAS].

The Maui scheduler [MAUI] and Silver meta-scheduler are external job schedulers for use on clusters and supercomputers which are capable of enforcing complex policy-driven scheduling schemas. They operate as a policy engine for controlling resource allocations to jobs, and concurrently optimize the use of managed resources. Maui operates by guiding the scheduling decisions of other cluster management schedulers. It manipulates several kinds of objects: jobs, nodes, reservations, QoS structures, policies, and composite objects.

Lastly, the fair share scheduling strategies which were studied in depth in the early 1980s in the context of mainframe batch and timeshared systems, and then brought into the UNIX environment, represent another important body of work related to our problem. The purpose of fair share scheduling is to dynamic control the distribution of resources allowing a greater predictability for the execution of a given process. For example, [FairScheduling] introduced several measurements and strategies for the UNIX operating system for dealing with resource distribution to set of

processes.

Authors propose in [Spinx] a novel framework for policy based scheduling of grid-enabled resource allocations. The framework has several features. First, the scheduling strategy can control the request assignment to grid resources by adjusting resource usage accounts or request priorities. Second, efficient resource usage management is achieved by assigning usage quotas to intended users. Third, the scheduling method supports reservation based grid resource allocation. Fourth, Quality of Service (QoS) feature allows special privileges to various classes of requests, users, groups, etc. This framework is incorporated as part of the SPHINX scheduling system, currently under development at University of Florida. The difference with our approach consists in the fact that we do not assume a centralized point of usage policy specification, but a more distributed approach of specification and enforcements at the site level [Spinx].

## 8 Conclusions

We have presented and evaluated an approach to representing and managing resource allocation policies in a multi-site, multi-VO environment. We present results in three distinct areas. Firstly, we present what elements are we interested in and how are they interpreted. Secondly, we were interested to convince the audience that these elements are really providing the support for UP enforcement. Thirdly, we presented the main gains for Grid3 users and VOs in scheduling workloads. We have also presented a simple S-PEP for CPU sharing [VO-Ganglia]. The UP specification and enforcement were achieved through specific components tightly coupled with the monitoring infrastructure.

There are still problems and technical details not fully explored here. For example, our analysis did not consider the case of cluster administrators that *over-subscribe* local resources, in the sense of a local policy that states that 40% of the local CPU power is available to VO1 and 80% is available to VO2. A second issue not fully explored here is the disk allocation problem. We are interested to use the same policy specification for disk.

**Acknowledgements:** We thank Iwona Sakrejda, Jorge Rodriguez, Timothy Thomas, Leigh Grundhoefer, Mark Green, Jens Voeckler, Robert Gardner, and Ian Fisk for discussions and support in carrying on 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.

## Bibliography

- [GriPhyN] Avery, P. and Foster, I. The GriPhyN Project: Towards Petascale Virtual Data Grids, 2001. [www.griphyn.org](http://www.griphyn.org).
- [DataGrid] 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). 187-200. 2001.
- [Grid3] Grid2003 Team, "The Grid2003 Production Grid: Principles and Practice", *Proc 13<sup>th</sup> IEEE Intl. Symposium on High Performance Distributed Computing*, 2004.
- [SNAP] Czajkowski, K., Foster, I., Kesselman, C., Sander, V. and Tuecke, S., SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems. in *8th Workshop on Job Scheduling Strategies for Parallel Processing*, (2002).
- [CAS] Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. in *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, (2002).
- [Decoup] Ranganathan, K. and Foster, I., Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications. in *International Symposium for High Performance Distributed Computing*, (Edinburgh, UK, 2002).
- [GARA] Foster, I., Fidler, M., Roy, A., Sander, V., Winkler, L., *End-to-End Quality of Service for High-end Applications*. in *Computer Communications*, 27(14):1375-1388, 2004.
- [Anatomy] Foster I., Kesselman C., Tuecke S., "*The Anatomy of the Grid*", International Supercomputing Applications, 2001.
- [CAS] Pearlman L., Welch V., Foster I., Kesselman C., Tuecke S., "*A Community Authorization Service for Group Collaboration*", Policy Workshop, June 2002.
- [Ten] Schopf J., Nitzberg B., "*Grids: The Top Ten Questions*", Special Issue of Scientific Programming on Grid Computing, 2002.
- [Agreements] Zhao T., Karamcheti V., "*Expressing and Enforcing Distributed Resource Agreements*", Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 2000.
- [Lupu-thesis] Lupu E., "*A Role-based Framework for Distributed Systems Management*", PhD Dissertation, Imperial College of Science, Technology and Medicine, University of London, Department of Computing, 1998.
- [FAIR] Henry G.J., "*A Fair Share Scheduler*", AT&T Bell Laboratory Technical Journal, Vol. 63, No. 8, October 1984.
- [FAIR2] Kay J., Lauder P., "*A Fair Share Scheduler*", University of Sydney and AT&T Bell Labs, 1988.
- [PolicyNetworking] D.C. Verma, "*Policy Based Networking, Architecture and Algorithms*", New Riders Publishing, November 2000.
- [PolicyNetworking2] D. Kosiur, "*Understanding Policy Based Networking*", Wiley Computer Publishing, 2001
- [Ganglia] Massie, M., Chun, B., Culler, D., The Ganglia Distributed Monitoring: Design, Implementation, and Experience. in *Parallel Computing*, May 2004.
- [Condor] Condor Team, A Resource Manager for High Throughput Computing, Software Project, The University of Wisconsin, [www.cs.wisc.edu/condor](http://www.cs.wisc.edu/condor).
- [PBS] OpenPBS Team, A Batching Queuing System, Software Project, Altair Grid Technologies, LLC, [www.openpbs.org](http://www.openpbs.org).
- [LSF] LSF Administrator's Guide, Version 4.1, Platform Computing Corporation, February 2001.
- [MAUI] MAUI, Maui Scheduler, Center for HPC Cluster Resource Management and Scheduling, [www.supercluster.org/maui](http://www.supercluster.org/maui).
- [ResourceAllocation] Dumitrescu, C., Wilde, M., and Foster, I., "Policy-based CPU Scheduling in VOs", iVDGL Technical Report, 2003.
- [CPUAllocation] Dumitrescu, C., and Foster, I., "Usage Policy based CPU Sharing in VOs", in *Grid Workshop*, 2004.
- [Grid3\_policy] Dumitrescu, C., "Usage Policy based Resource Allocation for GriPhyN and iVDGL", [http://people.cs.uchicago.edu/~cldumitr/Grid3\\_policy](http://people.cs.uchicago.edu/~cldumitr/Grid3_policy).
- [Euryale] Voeckler J., Euryale: Yet Another Concrete Planner, in *Virtual Data Workshop*, 2004.
- [Pegasus] Deelman E., Blythe J., Gil Y., Kesselman C., Mehta G., Patil S., Su M., Vahi K., Livny M., Pegasus : Mapping Scientific Workflows onto the Grid, in *Across Grids Conference*, Cyprus, 2004.
- [Sphinx] In, J., and Avery, P., Policy Based Scheduling for Simple Quality of Service in Grid Computing. in *International Parallel & Distributed Processing Symposium (IPDPS)*, Santa Fe, New Mexico, April '04.
- [VO-Ganglia] Foster, I. and Dumitrescu, C., "*VO-Centric Ganglia Simulator*", GriPhyN/iVDGL Technical Report, '04-31.