Policy and Enforcement in Virtual Organizations

Glenn Wasson and Marty Humphrey
Department of Computer Science
University of Virginia
Charlottesville, VA 22904
wasson@virginia.edu, humphrey@cs.virginia.edu

Abstract

Arguably, the main goal of Grid Computing is to facilitate the creation of Virtual Organizations (VOs); however, to date, not enough attention has been placed on the policies and mechanisms by which these VOs will operate. The core of the VO--roughly, the responsibility of each Physical Organization (PO) in the VO to contribute and not unjustly consume resources in achieving the overall goal of the VO--is at best service-level agreements (SLAs) that lack a concrete connection to the underlying Grid software and at worst an implicit "in-spirit" agreement. Unfulfilled expectations and obligations on the part of each PO can have dire consequences and can ultimately lead to the demise of the VO itself. This paper identifies three general policies regarding resource utilization by which VOs might operate and presents the ramifications of each policy on the VO's day-to-day operations and the VO's ability to actually enforce the policy. A prototype implementation of a VO with the "you-get-what-you-give" policy is the basis of a concrete cost/benefit analysis of policy enforcement for this type of VO.

1. Introduction

A Virtual Organization (VO) is a dynamic collection of distributed resources that are shared by a dynamic collection of users from one or more Physical Organizations (POs). Many of today's virtual organizations [2][4][14][16][17] are formed to tackle large-scale scientific problems. Large computing centers typically provide the resources and domain scientists are selected as users. The emerging approach in Grid Computing [13] is essentially to define the VO as a particular set of users, whereby the equivalent of a "VO server" issues tokens to humans attesting to their membership in the VO (e.g., GroupMan [7], VOMS [26] and arguably CAS [18]). These tokens are then presented to the individual resources.

However, as VOs grow in scale, their creators will need to define their VOs in more complex and comprehensive ways than via low-level membership descriptors alone. Formal policy for VOs is becoming more realistic with the rise of specifications such as WS-Policy [5] and WS-PolicyAssertion [6] (and other languages, e.g. Ponder [12]). However, policy syntax will only be of limited use until the semantics of expressible policies are more understood.

There are many types of policies in Virtual Organizations. One of the most common uses of policy in grid computing to date has been security policy (either explicit, e.g. WS-SecurityPolicy [9] or implicit). Security policies typically express the type (or types) of tokens that a client needs for authentication with a particular resource. In some cases, the token used can be negotiated based on this policy. Security policies (e.g. "this service accepts only Kerberos tickets") are enforced "at the edges" of the VO by denying access to any client whose credentials do not take the correct form. Another common use of policy is for network configuration. The IETF's PCIM [19] provides syntax for specifying how network QoS should be altered based on the state of the network and a set of policy rules. Enforcement of network QoS characterization is well-defined via mechanisms such as DiffServ and support from network hardware.

Other types of policy for grid computing, however, may be more operationally complex. These policies may make assertions about users or resources outside the scope of a single domain or about interactions beyond a single client/server pair. Examples include:

- "There must always be at least 2 copies of the raw data from the linear accelerator kept somewhere in the VO."
- "Machines in the VO must be patched with the latest security bug fixes."
- "Sufficient attention must be devoted to making and keeping the VO *survivable.*"

Clearly, as these examples are meant to show, some policies are extremely challenging to implement; however, there are other policies that can be implemented via information about the dynamic state of the system as well as historical data. Still, ensuring compliance with such policies (i.e., enforcing them) is difficult because there is not necessarily a single, obvious point-of-enforcement or a well-understood enforcement model.

This paper concentrates on policies that describe the distribution of utilization of resources across the VO. For example, a policy might state that the disk utilization at site 1 and site 2 must be equal (in order to prevent the resources at either site from being unduly loaded). We refer to this type of policy as the *VO Resource Provisioning Policy,* and discuss this in more detail in Section 2. In Section 3, we discuss the issues of enforcing a VO Resource Provisioning Policy. In Section 4, we present the results from a prototype implementation of a VO (based on .NET) in which there is an explicit resource provisioning policy. This section provides a concrete cost/benefit analysis of the value of making such resource provisioning policy explicit. Overall, the value of this work is that it provides a novel treatment—both through a general discussion and through a concrete prototype—of the previously-neglected issues of VO-wide resource provisioning policy creation, monitoring, and enforcement.

2. VO Resource Provisioning Policy

The resource provisioning type of policy describes the VO-wide distribution of resource utilization. For example, a policy that states "the compute load of the virtual organization is to be divided equally among all member sites" describes the VO's intended steady-state. Other policies might include:

- "All work is to be performed on large queuing systems from 9 am – 5 pm and on PC clusters after hours."
- "75% of data stored in the VO's data will be in the VO archive. The remaining 25% will be evenly distributed across the VO's storage resources."

In general, in order to be operational, policies must be more concrete—in effect, dictating policy by describing actions that will be taken to maintain the desired VO-state. For example, a policy such as "if any site is performing less than 25% of the work of the other sites, all new work will be scheduled on that site until the work load is equalized" describes an explicit *trigger condition* and an *action* that will be taken if that condition is met. Note that while such statements may allow for automated policy enforcement, the VO's response need not be "fully automated". That is, the appropriate administrator could be alerted to the relevant condition and/or a set of corrective actions might be suggested, allowing the human to make the final decision.

Note that VO-wide operational policy is different than policies used in [15], [18], [20], and [21]. These systems make permit/deny decisions based on (potentially multiple) access control policies and the accessor's identity / group membership. VO-wide resource provisioning policy refers to how a VO will allocate its resources given its workload. The policy service of [23] is primarily concerned with how local administrators can control the way in which their resources are used by the VO. In that system, VO policy covers the same scope as local resource policy and the policy actually used is a "least privilege" combination of the two. We believe that VO creators and administrators will want to express policies that are fundamentally different than those used by local administrators. This work concentrates on policy at the VO level, but we believe that the combination of VO and local policy is an important area of research and will be more difficult than simple combination.

2.1. Representative VO policies

We believe that there are three VO-wide resource provisioning policies that are implicit in today's virtual organizations. The policies differ in both their resulting resource usage patterns and the implications on enforcement of such a policy.

Policy 1: Each PO member opportunistically gives what it can to the VO [the you-give-what-you-can (*ygwyc*) policy]

We believe that this is the dominant policy implicit in many of today's scientific VOs[1]. But we also note that this is probably *not* the desired policy, but rather the only policy that is easily implemented (primarily because there is no required enforcement for this policy). The purpose of this paper is to propose *alternatives* to this implicit policy, which has been thrust upon users and resource providers by default, and so this policy will not be addressed in the remainder of this paper.

Policy 2: Resource utilization is divided equally among member resources [the 1/N policy]

We refer to this policy as the "*1/N policy"* because each resource in the VO is to perform 1/Nth of the total work of the VO (non-equal variations of this theme exist as well). This policy can apply to any resource that is distributed throughout the VO's member organizations: cycles, disk space, or other specialized resources. A 1/N policy is a common implicit desire in VOs where a PO's users are allowed to join a VO because it is assumed that the PO's resources will "pull their own weight". Typically, this policy is neither explicitly stated nor enforced.

Policy 3: Each PO member receives VO utilization credit for the resource utilization their PO provides to other VO users outside the PO [the you-get-what-you-give (*ygwyg*) policy]

[1] Although resource providers in scientific VOs are funded to provide resources to a grid user community, there is often no formal statement of how resources are divided between a provider's grid and non-grid users. Resources are provided as they are available.

Instead of requiring an equal distribution of resource utilization throughout the VO, this policy allows for users to utilize as much of the VO's resources as they wish provided they "repay" the VO by providing access for other members to resources they control. We contend that, arguably, this policy and policy 2 (1/N) are the desired policies in many emerging VOs.

2.2. Utilization measurement

How can a policy enforcement service determine if a VO's policy is being fulfilled by the VO's member resources? To measure utilization for a *1/N* policy, a service must be able to assess the total resource demands on the VO and the current utilization at each member resource. To measure utilization for a *ygywg* policy, a service must determine the resources being consumed by a particular user and the resources being provided to other VO members by that user's PO.

Depending on the situation, measuring resource utilization can be performed via resource-centric mechanisms or via some measurement made at the time of the Grid Service request. For some resources requests (e.g., storing a file) the most effective way to determine the total demand throughout a VO is to have a single VO-wide interface by which users access that resource (e.g., a web-based portal by which users "upload files to the VO" from their desktops). Since all resource requests flow through this portal, total resource utilization is easily calculated as the sum of all granted requests. Constraining all Grid usage to be performed through a small collection of portals is unrealistic, so there must also be "daemons" monitoring and reporting resource usage, irrespective of *how* the grid work originated. This is particularly attractive for legacy applications that must "run in the VO" but cannot be (or should not be) modified to accommodate measurement. Work in the DMTF, such as the Common Information Model (CIM) [10] and the Desktop Management Interface (DMI) [11], provide a mechanism for utilization measurement by defining a common description language and a set of APIs for resources and clients to publish and receive resource information. Many queuing system support various processor usage statistics and tools such as NWS [25]can collect utilization data for resources including networks.

However, VO resource consumption can be more complicated than a simple sum over all resources. For example, it might be necessary to measure the *coordinated usage* over multiple resources, if the VO's policy is to "reward" such efforts (under the assumption that the VO's mission is being accomplished under such conditions). For example, assume that UVa is contributing part of one of its clusters to a VO that's trying to solve some large Physics problem. At a particular time, the non-UVa users want to use the UVa cluster as part of a coordinated VO-wide experiment. UVa suspends current jobs on its cluster that are being executed by UVa biologists to contribute to this VO-wide experiment. In this case, the UVa scientists should be granted comparable privilege across the VO at a later date.

Actual resource utilization may not be the only parameter that must be measured. For example, in a desktop PC grid VO with a *ygwyg* policy, does a user providing access to his PC actually need other users to run on the PC in order to receive credit or can they receive credit merely by making the PC available? Clearly time of day and length of contiguous availability are important also. Perhaps a VO also cares about reliability or security of the resources. In general, many parameters could be measured. It is useful to think of resources as providing a certain quality of service (QoS) to the VO's users, with the VO policy dictating how this is to be measured.

2.3. Accounting

We distinguish the measurement of resource consumption from the recording of such measured consumption (i.e., in some database). For any VO policy, some form of accounting service is needed to record the utilization data. The collection of the distributed resource utilization information can be handled by services such as the Globus project's MDS [8]. The accounting service then compiles the utilization information so that the enforcement system can check for enforcement conditions (discussed in Section 3).

One important accounting issue is the "exchange rate" for various resources. How can the utilizations of CPU time on various processors be compared? How much disk space does a user in a *ygwyg* VO need to donate to get 3 CPU hours on another machine? While VO policy needs to specify this, it is beyond the scope of this work. It is however being addressed by other projects in the Global Grid Forum [22]. Work on grid economies [1][24] is valuable here, particularly with respect to the *ygwyg* policy. Grid economic work concentrates on using economic models to determine the current "price" for a resource and mechanisms for brokering that resource to clients. The VO-wide resource provisioning policies presented here are complementary in that they specify the VOs "coin of the realm" (the resources themselves, i.e. utilization is paid for by providing utilization) and an enforcement model (see below).

It is worth noting that each VO needs some minimum standard that each member resource must meet in order to join. If a resource is sub-standard compared to others in the VO, no user will want to use that resource, and so its owner, will be unable to receive credit for its use. Again, the exchange rate is important because the providers of a large tape archive will not necessarily need to have processors competitive with those of a compute farm provider.

3. Enforcement of VO Resource Provisioning Policy

Of course, having a policy means having to enforce that policy. Where policies are implicit, they are usually enforced by site administrators, often through the configuration of site-specific security infrastructure. Explicit policies allow enforcement to be automated, and thus provide more fine-grained detail than a human would be able or interested in enforcing "by hand". We divide policy enforcement into *enforcement conditions* and *enforcement actions*.

An important issue in policy enforcement is determining which entities (i.e. principals) will be effected by the enforcement. In other words, who gets credit/punishment for complying/not complying with VO policy? A supercomputer centers provides many resources and users to any VO to which it participates. If one such VO had a 1/N policy, who would see the effects of a supercomputer center's resources not being used as defined in the policy? Would it be all of that supercomputer center's users (or a subset)? Or would it be users from other POs who are affected by some automatic reconfiguration of the grid resources? In a *ygwyg* VO, there must be an association between users and resources whereby utilization of the resource provides "credit" to one or more users. However, since policy enforcement is based on the credit (or lack thereof) of a particular user, the issue of how to divide the credit for use of the resources of a super-computing center (or any similar organization) among its users is an important one.

3.1. Enforcement Conditions

Enforcement conditions describe the situations under which the VO should take some action to enforce its policy. The simplest conditions are based on tolerances from some nominal distribution of projected resource utilization. For example, for a 1/N policy, this might be whenever the difference between the most and least utilized resource exceeds some limit. For a *ygwyg* policy, this might be when a VO member has used a fixed amount more than their associated resources have provided.

In the event that some corrective action must take place, there is an issue regarding in-progress operations. Should an ongoing user operation be immediately terminated or allowed to complete when an appropriate enforcement condition is detected? What should be done with the (possibly partial) results of that operation? Should they be returned to the user, removed or held in escrow until the enforcement condition no longer exists?

Another matter is how conditions in the VO at startup affect enforcement conditions. In a *1/N* VO, policy should specify some minimum VO-wide workload below which the policy is not enforced. This prevents enforcement actions from being taken when there is not yet enough demand to warrant a change in the utilization distribution. In a *ygwyg* VO, the policy must specify the initial credit that is given to a new user. Such a VO depends on each user both consuming resources and providing them. If a user only provides resources that other VO members use, but the user does not consume any herself, eventually that user will receive a disproportionate amount of the total "credit" in the system. This can have the effect of starving out other users. Initial credit must be carefully selected to not allow users to perform large amounts of work without having to contribute resources, but not so small as to allow the VO to easily bog down when one user does not currently require resources.

It is interesting to note that *ygwyg* VOs have an inherent "risk". The VO's enforcement conditions must define a maximum possible "debt" that a VO member can accumulate before being completely denied access to VO resources. The virtual organization "risks" this much resource utilization whenever a new member joins the VO. If the user then leaves the VO (with their associated resource), the VO has lost the ability to collect service back from that user (see section 4.3 for a discussion of another similar kind of risk). Obviously decisions on membership in large VOs that span many POs must be made carefully.

3.2. Enforcement Actions

The two basic questions for a VO's enforcement actions are precisely *what* those actions are and *who* performs them. There are two types of actions that can be taken: *punitive* and *corrective*.

Punitive actions are those that reduce the quality of service that the VO provides to a user or set of users. The idea is to cause those users to enact a change in a particular resource's compliance (either directly for self-administered resources or through their local system administrators). Assume that UVa contributes 1 TeraByte of file space to a 1/N Physics VO. If a UVa system administrator for the cluster repeatedly removes files that have originated outside of UVa, then a punitive action might be for "the VO" to send all UVa computational jobs to a slow machine, until UVa proves that it is willing to let non-UVa files reside at UVa.

Corrective actions are actions in which the system attempts to alter the VO's distribution of resource utilization to achieve compliance. Assume UVa agrees to contribute part of one of its clusters to a 1/N VO. If there is a disproportionate amount of non-UVa jobs being executed on the UVa cluster, then jobs may be redirected to other resources in the VO to attempt to achieve the 1/N goal. It is important to note that in this context, "disproportionate" refers to the distribution of work across the VO, not some measure of UVa jobs to non-UVa jobs on the UVa cluster. Corrective actions are not meant to "punish" underutilized resources, but to alleviate the load on over utilized ones (and thus the amount of that resource available for local non-VO projects).

In general, whether a policy specifies punitive or corrective actions will depend on the ability of the VO to affect the individual member resources. The more independent the member resources are, the less likely it is that an enforcement service can correct for policy non-compliance. For example, if VO members can directly submit jobs to VO resources (as opposed to using a centralized scheduler), then those compute resources must be willing to accept a directive from an enforcement service to re-route submitted jobs to an under utilized resource. Corrective actions will also be limited by the requirements of any particular resource request. For example, not all compute nodes in a VO will have the same processor/OS and so it is not generally possible to route job requests to arbitrary nodes, even if they are under utilized.

VO policy will likely have multiple levels of enforcement actions depending on how far a particular resource is from compliance. Corrective actions in a 1/N VO will depend on the resource that must be equally distributed. To more evenly distribute compute cycle use, new scheduling approaches can be used to direct jobs to resources that must perform more work. To move evenly distribute space utilization, files can be migrated between storage resources after they are initially placed. In *ygwyg* VOs, corrective actions will typically involve some form of scheduling as various resources will need to receive use for their owners to receive credit. Some resources may require assistance from a VO level scheduler to prevent starvation. While the most obvious punitive action is to outright restrict a particular user's (or set of users') access to other resources in the VO if their associated resource is non-compliant, less stringent punitive actions include sending email to administrators or users merely alerting them to the problem (and thus give the opportunity for the end-users to correct the problem themselves).

The final issue is the identity of the entity that actually performs the enforcement actions. While an explicit VO policy in the long-term facilitates automated enforcement, in many cases, a human will be in the loop to authorize/schedule an enforcement action. No matter how humans are involved, there may be multiple enforcement services. Policy may specify that there is one per resource, one per user, one per action, or some other measure. For example, every time a UVa scientist attempts to submit a job to the Physics VO, it may be appropriate to "intercept and evaluate" the submission for proposed compliance with the VO policy. The selection of the number of independent enforcement services will depend on which method will most effectively scale with the virtual organization.

3.3. Security

An important issue is how all of the components that process and implement VO policy can ensure the veracity of the information they consume. As VOs grow larger and contain more members, there is a greater chance for accounting services to receive bogus utilization information or for enforcement services to receive bogus instructions to execute enforcement.

One scalable solution that we have been pursuing is to use digital signatures on messages exchanged between services. The WS-Security specification [3] defines a standard technique for signing XML messages. PKI cryptography and X.509 certificates can be used to generate signatures and different portions of the same message can even be signed by different entities. For example, any resource utilization report sent to the accounting system by a resource should be signed by that resource's private key. Note that this implies that each resource has its own certificate. While this signature would prevent a third party for modifying the utilization report on the wire, the resource itself may receive bogus user requests. This requires that all user requests be signed by the user, enabling the resource to verify that they are a member of the VO. Timestamps can be used to prevent third parties from artificially inflating a resource's utilization by replaying valid user request messages. There is a similar problem with messages sent to the VO's enforcement service. Any data that this service receives from the accounting service must be signed by the accounting service. Any resources contacted by an enforcement service to alter a member's privilege on that resource must similarly expect those messages to be signed by the enforcement service.

4. A Prototype Policy-based Virtual Organization

In order to further refine the issues and approach suggested thus far in this paper, we have implemented a policy-based VO prototype using Microsoft's .NET and Web Service Extensions (WSE). We chose .NET because of its proposed role in OGSA/OGSI, its extensive support of Web Services and its support of the emerging Web Services security specifications (we decided that evaluating this in the context of Globus directly was too difficult). This grid-system is composed of a set of web services that act as access points to resources, handle resource utilization accounting, and enforce the virtual organization's policy. The prototype system described here implements the *ygwyg* policy.

The policy-based VO prototype consists of 3 types of web services: *GateKeepers* that represent access points for resources in the VO, *Enforcers* that are in charge of carrying out the VO's enforcement actions, and a *Bank* that collects resource utilization data and holds information about member users and resources. All of these services are stateful and expect any request messages sent to them to be digitally signed by an authorized entity.

4.1. The GateKeeper service

The GateKeeper service is similar in spirit to the Globus Gatekeeper and provides access to a resource for VO members. In this prototype, members can request use of processor and/or disk resources and when the request is completed, the resource utilization (size of file or runtime of job) is sent to the Bank. The resources in this VO, a pool of PC-class machines, each "credit" one user for their utilization (the owner of the machine).
The GateKeeper interface provides a number of important methods with the `Register` and `RunJob` methods being the most interesting (`WriteFile` is another method similar to `RunJob`, but for accessing the disk resource). The `Register` method is used when joining the VO. When a GateKeeper service is started, this method is invoked to tell the Bank which user should receive credit for work done by this GateKeeper's resources. The message sent to the Bank contains the user DN to be credited and information about the resources of the GateKeeper's local machine (i.e. processor speed and disk capacity). The latter information is used by the Bank to a) determine if this resource meets the VO's minimum standards and b) calculate how much a user is debited for using the resources (better resources "cost" more). The GateKeeper signs all outgoing messages with the local machine's certificate.

The `RunJob` method provides access to the GateKeeper machine's processor resource. `RunJob` is invoked via a signed SOAP message from the user wishing to execute on the machine. If the GateKeeper recognizes the DN (Distinguished Name) in the signature as belonging to one of the current VO members, the invocation is allowed (see below for why a request from a VO member might also be disallowed). `RunJob` takes the name of a Windows executable and job arguments and executes that job on the local processor. (For simplicity and to satisfy certain trust issues, the Windows executables are already installed on the machines.) When execution completes, a resource utilization report is sent to the Bank. This report contains the original, signed request message from the user and the job's total processor time (in ms) all under the GateKeeper's signature. This prevents rogue GateKeepers from altering the credit/debt of users who did not request use of their resource (although a user must trust GateKeeper's they do use to produce correct utilization information).

4.2. The Bank Service

The Bank service contains information on every VO user and resource. For users, this information consists of their DN, the DN of the GateKeeper (i.e. of the machine's host certificate) whose utilization provides them credit and their current credit/debit for using resources in the VO. The Bank's resource information consists of resource statistics (processor speed and disk size), and well as the resource's DN and the DN of its associated user.

The most important method the Bank exposes is `UtilizationReport`. `UtilizationReport` is called by GateKeepers after they have served a user's request for resources. The requestor is debited and the resource owner is credited with an amount based on the utilization information (e.g. processor time) and the resource statistics (e.g. speed).

The Bank is also in charge of the Enforcement service, which carries out the VO's enforcement actions. For each new member that joins the VO, the Bank creates a new Enforcer (see below). As utilization reports arrive, the Bank monitors for the VO policy's enforcement conditions and instructs the appropriate Enforcer (via a signed message), to take a particular enforcement action on its associated user.

4.3. The Enforcement Service

The Enforcement service contains a set of "enforcer threads", which handle enforcement actions on particular users. There are two kinds of enforcement actions in the VO, one punitive ("cutoff") and one corrective ("redirect"). The "cutoff" action is when the enforcer contacts one or more GateKeepers and instructs them to deny access to their resources to the enforcer's associated user. The "redirect" action involves the enforcer contacting one or more GateKeepers to ask that resource requests they receive be re-directed to the enforcer's user's resource. This has the effect of providing more credit to a needy user. Space limitations prevent a discussion of the redirection mechanism, but it is a simple matter of forwarding SOAP messages.

Each enforcement action is taken by sending a signed message to a set of GateKeepers. The GateKeeper may reject the message if it is a redirect request and that GateKeeper is already redirecting. If the Bank has triggered an enforcement action, it will direct the enforcer to undo that action when the associated condition is no longer met. The enforcer will then recontact the necessary GateKeepers. Currently, the Bank initiates a redirect action when a user is below 15000 credits and a cutoff action when they reach 0 credits. Admittedly, these values are somewhat arbitrary and the subject of future research.

The current Enforcement service design provides GateKeepers with all the information necessary to make policy-compliant access control decisions. This has the benefit of allowing VO users to communicate directly with VO resources using no central mediating authority. However, it does allow for resources to get "out of sync" with the Bank, as enforcement actions propagate through the VO. The prototype does not address this issue because highly-sychronized VOs are beyond our current scope.

4.4. Evaluation

In order to evaluate our prototype VO, we performed an experiment to measure the cost of conforming to the *ygwyg* policy. The VO enforcers use the "redirect" enforcement strategy and thus redirects jobs to resources whose associated users have low (or no) credit. Because the resources in the VO had different performance characteristics, redirecting a job to a different resource than originally targeted can result in decreased performance in terms of the turnaround time for the user's requests (recall that the VO's policy does not attempt to optimize user's turnaround time, but rather a particular distribution of resource utilization).

In the remainder of this section, we focus on a representative scenario that illustrates the cost and benefits of the prototype implementation. We chose the following parameters to be representative of current Grid operations today. A single, representative case was chosen so that we could more concretely explain the details/issues regarding the prototype.

Our experimental VO consisted of 4 GateKeepers (referred to as A, B, C and D) representing hosts with relative processor speeds of A=2, B=1, C=1.5 and D=1.2. The user associated with each host ran a specific job, a certain number of times, on VO resources other than the one they control. To simplify the experiments, users launched jobs at a specific rate such that there was no contention for processors. Four work-sets of 10, 5, 7 and 6 jobs were used. For each experimental run, each work-set was mapped to specific user, e.g. the user associated with GateKeeper A (user A) runs 10 jobs, user B runs 7 jobs, user C runs 5 and user D runs 6 jobs. All possible mappings of work sets to users (24) were run. For the jobs in a user's work-set, the user alternates between executing on the two fastest resources other than their own (e.g. user A runs on resources C and D, user B runs on resources A and C, etc.). This scheduling pattern was chosen because it approximates the "hand scheduling" common among VO users today. Since B is the slowest resource, none of the other users will choose to schedule on it. However, in each run user B executes enough jobs that the Bank will detect the redirect condition (user B credit < 15000) and begins diverting jobs to resource B. These redirections (and similar redirections to resource D) cause reduced response time compared with the response times that would have been produced by the originally requested resource.

Table 1 and Table 2 show the result of running the 24 mappings of work-sets to users. Results both with and without policy enforcement are shown. Table 1 shows the absolute difference between the number of jobs run by a resource and the number requested of others by that resource's associated user (e.g. the difference between the number of jobs run on resource A and the number of jobs run by user A). Totals for all 24 runs are shown. There are two interesting effects to notice in Table 1. First, the policy enforcement mechanism has not reduced the difference between jobs run by a resource and jobs requested by a user to 0. This is because of the specific redirection strategy, in which only the resource with the most credit is selected to redirect jobs to other resources. If more resources were used to redirect jobs to low credit resources, the policy can be complied with more closely. However, the possibility of over-compensating (due to the time needed send messages to more GateKeepers to disable the redirection) and sending too many jobs to the resource increases. The second is that user D is actually further from compliance with the policy. This is due to redirection of jobs (mostly to resource B) which would have been run on D and kept it in closer compliance. Table 2 displays the average turn-around time of all the jobs run by each user in all 24 runs. This table shows the cost of policy enforcement in terms of the performance seen by the users. The decrease in performance seen in the policy-based VO is due to the redirection of jobs to resources B and D (which are slower). This redirection is necessary because of the number of jobs run by user B and user D requires the redirection to bring up the credit of those users.

	user A	user B	user C	user D
No policy	3.75	7.00	3.75	0.25
YGWYG	1.63	1.33	1.67	1.33

Table 1. Average difference between number of jobs run on a resource and run by that resource's associated user

resource	A	B	C	D
No policy	15.83	11.80	13.91	11.96
YGWYG	16.25	13.59	16.53	14.30
% slow down	2.7%	15.2%	18.8%	19.6%

Table 2. Average turn-around times for jobs run by each user (in seconds)

5. Conclusions

As grid computing evolves, virtual organizations will become more important, have more resources and members and be more difficult to administrate. Virtual organization policy assists the creators and maintainers of a VO by allowing them to specify a set of rules governing the virtual community. This work describes issues involved in policy specification and enforcement as well as a prototype implementation. In future work, we plan to expand the deployment and evaluation of the prototype in the context of our continuing work in supporting OGSI-compliant services in the .NET framework (and thus all OGSI-compliant services).

6. References

[1] Abramson, D., Buuya, R., and Giddy, J. 2002. A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. Future Generation Computer Systems. Vol. 18(8).

[2] Alliance: National Computational Science Alliance. 2002. http://www2.ncsa.uiuc.edu/About/Alliance/

[3] Atkinson, B., et. al. 2002. Web Services Security (WS-Security). http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-security.asp

[4] ATLAS Grid. 2002. http://www.usatlas.bnl.gov/computing/grid/

[5] Box, D. et. al. 2002. Web Services Policy Framework (WS-Policy). http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policy.asp

[6] Box, D. et. al. 2002. Web Services Policy Assertion Language (WS-PolicyAssertions). http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-policyassertions.asp

[7] CalTech Virtual Organization Group Manager ("GroupMan"). http://groupman.sourceforge.net/

[8] Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C. 2001. Grid Information Services for Distributed Resource Sharing. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)

[9] Della-Libera, G. et. al. 2002. Web Services Security Policy Language (WS-SecurityPolicy). http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-securitypolicy.asp

[10] Distributed Management Task Force (DMTF). 1999. Common Information Model (CIM). v. 2.2. http://www.dmtf.org/standards/cim_spec_v22.

[11] Distributed Management Task Force. 1998. Desktop Management Interface Specification (DMI). v. 2.0. http://www.dmtf.org/standards/documents/DMI/DSP0001.pdf.

[12] Dulay, N., Lupu, E., Sloman, M. and Damianou, N. 2001. A Policy Deployment Model for the Ponder Language. Proc. IEEE/IFIP International Symposium on Integrated Network Management (IM'2001)

[13] Global Grid Forum. http://www.ggf.org.

[14] GriPhyN: The Grid Physics Network. 2002. http://www.griphyn.org/index.php.

[15] Keahey, K and Welch, V. 2002. Fine-Grained Authorization for Resource Management in the Grid Environment. Proceedings of Grid2002 Workshop.

[16] NASA Information Power Grid. 2002. http://www.ipg.nasa.gov/

[17] NPACI: National Partnership for Advanced Computing Infrastructure. http://www.npaci.edu

[18] Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S. 2002. A Community Authorization Service for Group Collaboration. Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks.

[19] Strassner, J., Ellesson, E., Moore, B. and Westerinen, A. 2001. Policy Core Information Model -- Version 1 Specification. RFC 3060.

[20] Sundaram, B., and B. Chapman. XML-Based Policy Engine Framework for Usage Policy Management in Grids. Proceedings of the Third International Workshop on Grid Computing (Grid 2002). Baltimore, MD, November 2002.

[21] Thompson, M. 2001. Akenti Policy Language. http://www-itg.lbl.gov/security/Akenti/Papers/PolicyLanguage.html

[22] Usage Record Working Group. Global Grid Forum. 2002. http://www.gridforum.org/3_SRM/ur.htm

[23] Verma, D., Sahu, S., Calo, S., Beigi, M. and Chang, I. 2002. A Policy Service for GRID Computing. GRID 2002, LNCS 2536. Springer: 243-255.

[24] Wolski, R., Plank, J., Brevik, J. and Bryan, T. 2001. Analyzing Market-based Resource Allocation Strategies for the Computational Grid. *Intl. Journal of High-performance Computing Applications*. Vol. 15(3).

[25] Wolski, R., Spring, N., and Hayes, J. 1998. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Journal of Future Generation Computing Systems*.

[26] Virtual Organization Membership Service. http://grid-data-management.web.cern.ch/grid-data-management/security/