Explicit Resource Usage Policy Management and Enforcement in Grid Computing

A PhD Thesis Proposal

Jun Feng
(jf4t@cs.virginia.edu)
Department of Computer Science
University of Virginia

Draft of Jan 28th, 2006
Project Summary

Grid computing can be defined as secure and efficient collaboration across geographically-distributed resources such as CPUs, data, storage and network, where the resources are not controlled or owned by a single institution. Often, a Virtual Organization (VO) is formed when multiple Physical Organizations (POs) each contribute resources to a single resource pool shared by all. In Grids, policy can refer to the preferences, requirements, constraints and obligations by which an entity, such as a VO, site or node shares information, share physical resources or engage services.

This thesis focuses on the policies to control resource usage at both site and VO level, which I call resource usage policies in general. Site level usage policies are used by resource owners to directly constrain local resource consumption by Grids. Such policies can express constraints such as “GridFTP service is open from 5:00PM to 9:00AM only”. VO level usage policies, in this research, refer to those policies regarding how the aggregate Grid resource consumption can be distributed among sites. In particular, this research will focus on the VO policies which can lead to (when strictly enforced) a “fair” (subject to further definition) distribution of the benefits and costs of resource sharing among sites. One example of such policies is “resource usage must be evenly distributed among the participating sites based on their capabilities”.

In contrast to the extensive research and wide use of explicit security policies in Grids, to date, not enough attention have been paid to describing, manipulating or enforcing resource usage policies in Grids. At the site level, most existing Grid systems only support limited types of usage policies for CPU resources, leaving the provisioning of other resources on the Grid constrained by security policies only. Without sufficient local usage policies, resource providers may be reluctant to participate in Grid collaborations fearing that the Grid activities can be intrusive to local activities. At the VO level, existing Grid systems generally do not have any resource usage policies or at best have some resource usage agreements which lack a concrete connection to the underlying Grid software. This type of VO can not provide the assurance to resource providers on what should or should happen to their resources, which may ultimately lead to the demise of VO itself when the participants find that their intentions to share resources can be violated or subject to potentially damaging ramifications.

This proposal lays out an 18-month plan to address above problems. First, some representative resource usage policies for resource providers and VOs will be identified. In particular, I will identify some fair share policies at VO level and their associated evaluation metrics. A flexible policy model and a XML policy language will then be devised to describe these policies. Second, a generic policy management system called MyPolMan will be developed to support secure policy manipulations on Grids, such as policy authoring, publishing and transferring. Third, policy enforcement mechanisms including algorithms and systems will be devised and implemented in Grid software and protocols, such as GridFTP and GRAM. These policies and mechanisms will be evaluated through simulation and on real production Grids, such as UVaCG.

The languages, software and algorithms generated in this research will lead to the more flexible and sufficient resource usage policies and hence the more efficient, controllable collaboration environments in Grids.
1 Introduction

Grid computing can be defined as secure and efficient collaboration across geographically-distributed resources such as CPUs, data, storage and network, where the resources are not controlled or owned by a single institution. In the context of Grid Computing, the Virtual Organization (VO) \[1\] is defined as a dynamic collection of users and resources working on a common problem or working through a common resource usage agreement. A VO can also be defined in terms of the set of physical organizations (PO) to which each user and/or resource in the VO belongs. Each physical organization in the VO offers some of its resources for the needs of the VO, thus they act as resource providers.

In Grid, policy can refer to the preferences, requirements, constraints and obligations by which an entity shares information, share physical resources or engage services. There could be many types of policies in Grids. One of the most common uses of explicit policies in Grid Computing today has been the security policies \([2][3][4][5]\). Using security policies, a resource provider is able to limit or deny resource use to a particular user or group typically based on the user (or group) identity, but sometimes also on the contents of the usage request or the authentication mechanisms. Besides security policies, policies can also be used to control resource usage in Grids. We call such policies resource usage policies. This type of policies defines constraints on the resource use that often incorporate information outside the context of user identity or a single request message between client and server. Thus the enforcement of this type of policies is often much complex than security policies.

This thesis research focuses on the resource usage policies at both site (resource provider) and VO level. Typically, a site level resource usage policy will directly control how individual resources can be consumed by Grid. Such policies can express constraints such as “the GridFTP service can be accessed from 9am to 5pm” or “only 20% of the resource storage capacity can be used by the Grid”. On the other hand, since VO does not own any resources, VO level usage policy will usually deal with how the aggregate VO resource consumption should be distributed among sites. In particular, this research will focus on the VO policies which can lead to (when strictly enforced) a “fair” (subject to further definition) distribution of the benefits and costs of resource sharing among sites. One example of such policies is “resource usage must be evenly distributed among the participating resource providers based on their capabilities”.

To date, not enough attention has been paid to the describing, manipulation and enforcement of resource usage policies. At the site level, most existing Grid systems have either implicit resource usage policies (with ad-hoc enforcement mechanisms) or support only limited types of policies. Systems that do provide some support for resource usage policies typically consider only CPU resources, leaving the provisioning of other resources on the Grid unconstrained \([6][7]\). Providing more fine-grained control over resource usage to the resource owner is, I argue, necessary to build truly large-scale Grids. As for the VO, the emerging approach in Grid Computing today is essentially to define the VO as a particular set of users, whereby the equivalent of a “VO server” issues access tokens to humans attesting to their membership in the VO \([8]\). In this definition and implementation of VO, the policies of VO – in which, the responsibility of both resource providers and Grid users to contribute and consume is defined – is at best service-level agreements that lack a concrete connection to the underlying Grid software. The absence of VO policies, in particular, the absence of VO resource usage policies, makes VO a primitive collaboration whose behavior is hard to predict and manage. For instance, in Grids today, when an institution signs up for a VO, it has no idea what will happen to its resource because the VO can not provide any assurance such as “your site will not be filled up with job requests from other
sites”. Failure to provide such assurance will cause collaborators to be reluctant to engage Grid-level collaborations for the fear of discovering either in real-time or after-the-fact that their intentions with regard to resource sharing can be violated or subject to potentially damaging ramifications.

This thesis proposal lays out an 18-month plan to address above problems. In this research, I will identify appropriate policies for both resource providers and VOs for resource types such as CPU and storage. In particular, I will identify some fair sharing policies at VO level. I will then devise a policy model and policy language to express these policies explicitly. To manage these policies, I will design and develop a generic policy management system to securely create, publish and deliver these policies. I will also contribute algorithms and systems to enforce these policies. The benefits and costs of having these policies in Grids will also be thoroughly evaluated.

I expect the following contributions by the end of this research.

- A representative sample of usage policies at site level for CPU and storage resources and their enforcement mechanisms in Grid software and protocol such as GRAM and GridFTP.
- A policy model and an associated policy language to express explicit resource usage policies.
- A generic Grid policy management system called MyPolMan for secure policy authoring, discovery, transfer and other policy operations in Grids.
- A representative sample of VO level policies and their enforcement mechanisms.
- Metrics to evaluate the fairness of a Computation and Data Grid.

The remainder of this proposal is organized as follows. Section 2 describes existing work related to the goals and scope of this proposed research. Section 3 contains the details of the research design and methodology. Section 4 contains the specific plans and experiments I will use to evaluate the work. Section 5 contains the early progress towards this research direction. Section 6 is the summary, which includes a list of the key contributions and key research questions to be addressed.

## 2 Related Work

IETF [9] and DMTF [10] have a generic policy framework including four major components, the policy management tool, the policy repository, the policy enforcement point and the policy decision point. IETF and DMTF also define a common information model [9] for the policy rules. The underlying premise in this information model is that each policy should be considered as an expression of the form “if condition then action”. Policy research group [10] at the Global Grid Forum (GGF) has also adapted this IETF/DMTF policy framework by defining similar four software components. However, to date, there has no implementation of a policy system on Grids based on the GGF’s work.

Policies are often encountered explicitly in the resource management systems, especially schedulers. As for policies at local site level, the Maui Scheduler [11] operates as a policy engine for controlling resource allocations to jobs while concurrently optimizing the use of managed resources. In Maui, scheduling policies can be fine tuned on several policy dimensions. Different scheduling policies (such as fair-share, queue time and QoS levels) are represented by priority factors and each factor is assigned a weight to control its impact on overall scheduling. However, in Maui, policies are restricted to scheduling policies instead of resource usage policies. Some Grid middleware can allow some resource usage policies at individual machine level. For example, in Legion, host administrators can specify some job admission policies such as “admit new job only when CPU usage in past 5 minutes is under 70%” [6]. In Condor, a few restriction
terms can be specified in its ClassAds language to better control the local CPU resource usage [7]. However, both Legion and ClassAds can only support policies at host level, and policies in these systems are generally restricted to CPU resources only.

There are some systems can consider policies at both the Grid/VO and site levels. Meta-schedulers, such as Silver [12] and CFS [13] can allow a few scheduling policies at the meta-scheduler level, such as round-robin and FCFS. CFS, in particular, can allow plug-ins to implement custom scheduling policies. Grid accounting systems, such as SGAS [14], DGAS [15], Gold [16] and GridBank [17], can assign each project a Grid-wide CPU-time allocation. In these systems, projects pay for their resource consumption with their allocations and may be denied resource access when their allocations have been spent. By manipulating these allocations, these systems can achieve a coarse resource allocation policy at Grid/VO level. Catalin et al have identified usage policies (UP) for both sites and VOs [18]. His research presents a policy model to express these UPs and a distributed architecture for UP-based scheduling in a Grid environment.

Wasson and Humphrey focus on the policies at the VO level [20]. They identify three general policies regarding resource usage 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 software simulation implemented the “you-get-what-you-give” policy. While arguably simplistic, it was able to conduct experiments in which it refined the issues and approach and began to quantify the costs and benefits of attempting to enforce Grid policy.

OASIS eXtensible Access Control Markup Language (XACML) defines a language for expressing access control policies [21]. In XACML, conditions and effects (results) are combined into rules, and rules are combined into policies. In typical cases, the decision of an XACML policy is to permit or deny access to the resource specified by the policy. The usage policies generally will have more types of decisions other than just permit/denial. In Web Service, WS-Policy [22] provides a flexible and extensible grammar for expressing the capabilities, requirements and general characteristics of entities in an XML Web Services-based system. It specifies a base set of constructs that can be used and extended by other Web service specifications to describe a broad range of service requirements and capabilities. Policies can be associated with Web Services metadata through the use of WS-PolicyAttachment[23]. Policies can be sought after via WS-MetadataExchange [24] which defines the request-response based metadata exchange interactions between Web services and clients.

Outside the Grid Computing, Autonomic Computing [25] is an approach towards the self-managed complex computing systems with a minimum human interference. The core component of Automatic Computing is often a policy-based management infrastructure to simplify the management and automation of complex systems. A particular policy system PMAC [26] focused on business policy decisions and conflicts resolution using an additional value element in each policy. In contrast to PMAC, my research considers the usage policies in Grid Computing instead of business policies.
3 Proposed Research

3.1 Policies for the Resource Provider

There are basically two types of resource usage policies for resource providers: configuration policies and action policies. Configuration policies define an intended configuration of the system, while action policies define an operation to be performed when some certain conditions are met. Configuration policies can be considered as static action policies which must apply in all conditions. Below I give some example configuration and action policies.

1. “No more than 2 streams can be used in one GridFTP data transfer session”
2. “Provide at most 40% of storage space at disk C of opteron8.cs.virginia.edu for Grid use”
3. “Accept new job only when the CPU usage in past 5 minutes is under 70%”
4. “Purge all Grid data over 7 days old at /scratch every night between 1:00AM – 3:00AM”
5. “If keyboard/mouse is touched, preempt all Grid jobs on this host immediately”

In these examples, 1-3 are configuration policies while 4-5 are action policies. Configuration policies are independent of individual request type/message and system conditions. Thus they can be easily expressed using name-value pairs. The only difficulty for this type of policies is to identify and define policy terms and their appropriate values for each type of resources. For example, if we can define a term “MaxStreams” as the maximum number of streams allowed, then the policy #1 can be expressed as “MaxStreams=2”. Complex configuration policies usually just involves more than one policy terms.

Compared with configuration policies, action policies are a little more complex to handle. Each action policy defines an action to be performed under certain conditions. A simple model for action policies can be “if (condition) then (action)”. Although this policy model is sufficient enough to describe action policies, for policy identification and classification purpose, it will be useful to introduce more elements to this model, such as the policy scope which describes the subject of the policy. By adding the policy scope, an action policy can then be expressed as “(scope) if (condition) then (action)”.

In the above policy model, the “scope” part is basically a string to identify which resource is governed by this policy. A resource can be expressed using URI. In some situations, a wild card can be used in URL to represent a set of resources with common properties, such as names. For example, the scope for policy #5 can be expressed as “opteron8.cs.virginia.edu/GridFTP”. A policy can have more than one scope. Unlike the simple “scope” part, the “condition” and “action” parts can be quite complex. Depends on the resource type, “conditions” and “action” will each have terms with associated types, values, logical and arithmetic operations.

I am open to certain possibilities that some existing policy languages may have abilities to express some subsets of usage policies and I will leverage these languages if possible. For instance, I have investigated on the capabilities of existing policy languages such as XACML and WS-Policy. Although XACML is limited to the security policies only, it supports a wide range of well defined terms with associated operations. Moreover, XACML can express security policies whose decisions can depend on some conditions, such as the user identify. A close observation on many configurations policies, such as the policy #1 above, reveal that many usage policy decisions are also either permit or deny, which will be in the ability scope of XACML language. For instance, the policy #1 can be written as “if the client requests more than 2 streams to be used, deny the request, and otherwise permit the request”, which can be expressed using XACML.
However, there are also many other usage policies which can not be expressed by XACML due to the following limitations.

First, XACML is limited to “permit/deny” decisions only while many usage policies involve decisions other than just “permit/deny”. For instance, in the policy #5 above, the decision is an action to be performed when the keyboard/mouse is touched. XACML can not express this type of decisions. Second, XACML policy model can only consider information in the security context, such as the static attributes of subjects and resources. The decision making process for many usage policies, such as the policy #2 above, will often requires information outside of the security context, such as the dynamic system load, resource consumption of a certain VO, and history usage which are all out of the expressive abilities of XACML. Third, XACML limits the interaction between the Policy Decision Point (PDP) and Policy Enforcement Point (PEP) to the request/response style only, in which the PEP must send a soliciting request to the PDP and then wait for the decision response. However, many usage policies require other types of interactions styles. For instance, the policy #4 requires a periodical timer based interaction initiated from the PDP first and the policy #5 implies an event based interaction initiated from PEP first. All these two interactions styles are not supported by XACML.

WS-Policy, on the other hand, has a simple framework defining “and” and “or” operations on some policy alternatives. It is easy to design a language to conform to WS-Policy specification. In general, a language that conforms to WS-Policy specifications usually is composed by these two operations on some domain specific policy terms.

My current plan is to incorporate some features of both languages with my own terms and rules to form a new policy language. This policy language will mainly base on the XACML grammar. New terms, new rules and new policy models, such as the event based policy model will be added to the XACML language. The XACML engine will be extended to support operations other than simple “permit/deny” and to consider information outside of the security context. Support for unsolicited interaction style between PDP and PEP, such as the event based interactions, will also be added. The new policy language will conform to WS-Policy if possible.

### 3.2 Policies at VO Level

Previously, we have identified two policies which we believe are implicit in VO (more policies will be identified as this research continues). The first one is the "1/N" policy [20]. The statement of this policy is "Resource usage is divided equally among member resources". 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 policy 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. Typically, this policy is neither explicitly stated nor enforced. Another policy is a so called “what-you-get-is-what-you-give” policy which stats “each PO member receives VO usage credit for the resource usage their PO provides to other VO users outside the PO”. Instead of requiring an equal distribution of resource usage 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. Although these two policies, lack a formal expression in some policy models, they are good examples from which I will derive more concrete VO level policies. In this research, I am interested in the VO level fair sharing policies. The research is inspired from the observation below.

Consider this scenario. A and B are two computation sites each with their own clusters. Seeing the opportunity to provide more available resources to their users, A and B decide to set up a Grid.
However, when the Grid is ready, users from site A find following problems: 1) Because some of the computation cycles of site A are now consumed by jobs from site B, users from site A now experience longer job turn-around time (or wait time or whatever metrics) on their own clusters 2) Although site A has given a fraction of its computation cycles to site B, A’s jobs submitted to site B are not treated equally well by B. For some reasons (such as site B has set its local scheduling policy to favor local jobs), jobs from site A can hardly compete for CPU resources with site B’s local jobs. Frustrated by these results, the administrators at site A decide to discontinue their participation in Grid.

Above scenario reveals a problem with current Grid implementation – sharing of resource on Grids can adversely affect some sites, in other words, can be unfair to some sites. Having a fair sharing policy implemented in Grid/VO level could alleviate the above problem. However, providing fair sharing at site level in Grids is not an easy task. First, there are no metrics to evaluate the fairness of a Grid today. Previous researches most focus on the fair sharing for individual jobs at a single cluster or CPU level. Second, current Grid software, such as schedulers, are mainly built to improve the Grid or cluster throughput, whether or not these software can lead to the fair sharing among sites remains unknown.

In this part of research, I will first identify some appropriate metrics to evaluate the fairness of a Grid/VO. From these metrics, I will then derive some fair sharing policies in their formal expressions. For example, if we consider an even distribution of available Grid resources among sites is a good indicator for the Grid/VO fairness, we then can define a fair share policy using the “if (condition) then (action)” model to “if (the difference between available Grid resource partition reaches a certain level) then (reduces some site’s Grid consumption rate)”.

I will investigate on the policy enforcement mechanisms in two typical Grid configurations, one with meta-scheduler and one without. In this research, I call the first Grid “meta-scheduler Grid” and the second Grid “hand-scheduling Grid”. In a hand-scheduling Grid, without the knowledge of the fair share policies and system status information, the jobs submitted by Grid users to a site may violate the fair share policies. The sites are responsible to detect policy violations, then execute some actions to bring the Grid back to the conformance with fair share policies. Possible actions are “priority adjustment”, “job rejection” and “job migration”. In a meta-scheduler based Grid, “priority adjustment” can still be used at both the meta-scheduler and local scheduler level to enforce the policy. However, “job rejection” and “job migration” will not be applied to meta-scheduler. Instead, the meta-scheduler will intelligently place jobs to enforce the policies.

I will evaluate my policy enforcement mechanisms using both real workload and synthetic workload in a simulation to evaluate the fairness of my enforcement mechanisms. The cost (or possible benefits) of the enforcement mechanisms will also be thoroughly evaluated using traditional metrics such as the average job response time and resource utilization rate.

I will also extend this work to deal with volatile Grid environment and storage resource sharing on Grids. Using the same methodology, I will develop similar metrics for the fair share of storage resources on Grids and then research on the policy enforcement mechanisms.

### 3.3 Policy Management System

A generic policy management system will be developed including the policy authoring tools, policy repository, policy agents and a policy transfer protocol. Figure 1 shows the main components of this system and the flow of policies in this system.
A policy repository provides a centralized storage for all policies in a single site or VO. The policy repository will have policy operation interfaces such as create, update, query, fetch, and delete. These interfaces will be compliant with emerging grid service specifications such as WSRF [28][29]. Policy authoring tools will be developed for Grid users to create their policies. Message level security mechanisms will be used to guarantee the secure transfers between policy repository services and other components of the policy management system. The policy repository service may also support policy access control list, against which all policy operations will be validated.

![Figure 1. A Policy Management System](image)

Between policy services and policy consumers, there are policy agents to facilitate policy distributions. The policy agent co-exists with other Grid services on the same machine to manage local policy copies. Policy services and policy agents execute a cache consistence protocol to assure the freshness of policies on the agent side. The use of caches on the side of the policy agent also reduces the burden on the policy service itself which will be beneficial for the scalability of the whole policy system.

### 3.4 Policy Enforcement

Having a policy means to be enforced. In this research, I will investigate the enforcement of resource usage policies on the existing Grid software and protocols, such as GridFTP and GRAM. The idea is to let Grid users to access the Grid through familiar APIs while allows the resource providers and VO to control resource consumption.

The basic steps to enforce a policy are to create a Policy Decision Point (PDP) first and then let a Policy Enforcement Point (PEP) to receive guidance from the PDP. If the policy is expressed using “if (condition) then (action)” model, the PDP will be responsible for the evaluation of the “condition” part, while the PEP will be responsible for the execution of “action” part.

To do this, a policy engine will be developed to understand the policy language, evaluate the current system conditions and execute some actions. Figure 2 shows the interactions between Grid services and policy engine. Usage policies for resource providers, often only require information on the local site to evaluate the conditional part of a policy. The action part for resource provider policies, depends on the situation, could be accept/reject job requests in GRAM
or similar actions in GridFTP protocol. Modifications must be made to the certain places in the implementation of these protocols to enforce those policies.

![Figure 2. Interactions Between Grid Services and Policy Engine](image)

Although the enforcement of policies all follows these common steps, each policy may require different enforcement mechanism, and for some policies, the implementation of both PDP and PEP can be quite challenging. For example, the “1/N” policy will require the precise usage measurement across all the resources in the VO to evaluate the condition part. Such precise measurement may depend on the existing Grid monitoring and auditing software. In some situations, new capabilities may need to be implemented into these existing software to monitor some specific types of resource usage. As for the PEP of “1/N” policy, the enforcement mechanism can vary depends on the situations. For example, for a computational Grid, the PEP may migrate jobs from over-subscribed sites to under-utilized sites to enforce the policy. For a data Grid, the PEP may choose to replicate those “hot” data items to other locations to bring the Grid back to the conformance to the policy. This type of policy enforcement will require the coordination among several Grid services and PEPs.

For any VO level fair sharing policies, one challenge is to maintain good scalability for enforcement mechanism as the Grid grows in scale. Centralized approach, such as meta-schedulers, may not be appropriate to enforce some policies in large scale environment. I plan to use decentralized approach, such as peer-to-peer based communication protocols between sites, to enforce some challenging policies. The scalability of enforcement mechanisms will be evaluated either through simulation or on large production Grids if possible.

Another problem with VO-level fair sharing policies is potentially they could be in conflict with local resource provider policies in some situations. For example, if a VO PEP decides to migrate a job from a site to another site to enforce the “1/N” policy, a local policy at the destination site, such as “limit Grid active jobs in queue to at most 8” might prevent this enforcement action to happen if there is already 8 active Grid jobs in the queue. In this situation, how to deal with policy conflicts is a challenge problem. A simple resolution strategy such as “always in favor of VO policy when conflicts happen” may not be appropriate in some situations and will need to be thoroughly evaluated.

Grid is not a consistent resource sharing environment by any means. On Grids, machines may come and go and services may go down and up. This raises another great challenge for enforcement of some policies. As Grids are under faults, some parts of the Grid may become unreachable or the Grid usage measurements become outdate. In these situations, the strict enforcement of some policies may become impossible. How to recover from these situations and
bring the Grid back to the conformance is worth investigation. For example, in a computational Grid which implements a “1/N” policy, temporary site offline will possibly cause other live sites to be over-subscribed with jobs. How to correct this will be a challenge problem.

4 Plan and Experiments

I have plans for this research in the following steps and time schedules.

a) The first step is to identify some resource usage policies for resources such as CPU and storage. This part will be done through extensive reading on published policy documents in existing Grids and other distributed systems. I will also develop some metrics for evaluation of Grid/VO fairness, from which some formal expressions of fair sharing policies in VO level will be derived.

b) After these policies are identified, I will devise a policy model and create a policy language based on XACML and WS-Policy to describe these policies. I will evaluate this new policy language mainly on its expressibility for a wide range of resource usage policies.

c) In parallel with the first two steps, I will start to build a policy management system. I will begin with limited functionality, then gradually add more functions to both service and agents. The experiments of this part will start with one VO policy service, one site policy service and some policy agents. To begin, the policy agent will be able to fetch, catch, and auto-update policies from the policy service. The policy service will be able to store and distribute policies. I anticipate a pull-based auto-update protocol to periodically update local copies of policies from a remote policy service.

d) I will prototype Grid services to enforce these policies as early as possible. Previously I have built rFTP to move data faster on Grid by utilizing converging flows [31]. I also have recently built a .NET-based GridFTP service and client [32]. The plan is to incorporate all of these previous projects together with policy services proposed here to have an explicit policy aware data placement/movement service. Similar system will be built for Grid job execution using GRAM protocol as well.

e) As for the VO-level resource usage policies, I will devise enforcement mechanisms for both computational and data Grid scenarios. Through a simulation on real workload and synthetic workload, I will compare the fairness of my enforcement mechanisms with classical scheduling strategies using the fairness metrics identified by myself. The impact of my enforcement mechanisms on the system performance in terms of traditional metrics such as job response time and site utilization rate will also be thoroughly evaluated.

f) I will then extend my work on VO-level policies to deal with more complex Grid environments by taking into account of possible policy conflicts with local resource provider’s policies and volatile Grid resources.

Below is the projected time table for this 18-month project plan.

<table>
<thead>
<tr>
<th>Table 1 Projected Time Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Identify resource usage policies at site and VO level</td>
</tr>
</tbody>
</table>
Define policy model and languages to describe policies & X & X & \\
Build, test and evaluate MyPolMan, Enforce resource providers policies in GridFTP and GRAM services & X & X & X & \\
Enforcement and evaluation of VO operational policies & X & X & X & \\
Thesis writing & & & X

## 5 Early Progress

I have made some progress [33] towards this research direction including a prototype policy management system MyPolMan and a data movement system to enforce some classes of policies for storage resources in Grids.

I have examined some existing Grid projects, such as the TeraGrid [34], LCG/EGEE [35] and Data Grid systems such as SRB [35] to understand their (implicit) storage usage policies, and policy needs. These fairly large scientific computing Grids represent one use case. I have also studied appropriate storage resource policies for desktops and handheld devices which will inevitably be incorporated into Grids as their capabilities increase [37][38]. Based on my investigation, I have identified that the following type of policies for storage resource providers.

- **Service Availability Policies**: Service availability defines when the service will be available to the Grid users. Storage resource provider can use this policy to limit the service operation time; for example, “the GridFTP service will be open for access from 5:00PM to 7:00AM Monday through Friday”.

- **Data Reliability Policies**: Storage resource providers can use this type of policy to define characteristics of the storage resource that effect how data stored on the resource is treated over time. Data reliability policy is subdivided into two policy classes, backup policy and purge policy. Backup policy allows a resource provider to specify which directory/drive will be backed-up and when/how often. For instance, “C:\GridFTP\Root\Pub is daily backed up” is a valid backup policy. Purge policy states how long a data item can stay on a storage resource. An example purge policy is “all data in /tmp is purged at midnight”.

- **Quota Policies**: Quota policy defines the space allocation strategies of resource providers. Policies can be either defined using relatively percentage of capacity or absolute data volume. Quota policies can be quite complex as the resource provider may provision disk on per directory or per user basis. However, more straightforward policies, such as “Grid data can not occupy more than 40% of disk C at host opteron8.cs.virginia.edu”, may be sufficient.

- **Network Consumption Policies**: Data access will consume network bandwidth. While this may not be an issue in high-end scientific clusters, it is likely to become more important as desktops (and smaller devices) begin making data available to the grid. Limiting network bandwidth consumption may also be needed to stay within the bounds of a service-level agreement (SLA) worked out between the storage provider and their network resource provider. Such policies can be used to limit the download/upload rate of GridFTP. Limits on the use of multiple streams in GridFTP may also be needed because it has been shown that multiple streams hurt the performance of other TCP applications [39].

Above policies will be expressed in XML that conforms to the WS-Policy [22] specification. Each MyPolMan policy contains a unique identifier (PolicyName) and a set of policy properties.
These properties include (but are not limited to) the policy author (AU), applicable scope (AS), valid time (VT) and current status (ST). In addition, each specific policy discipline will define its own terms that have meaning to the appropriate policy consumers. In this case, the storage resource of the C drive on the machine opteron8.cs.virginia.edu should allow a maximum of 40% of its total capacity to be used by the Grid. Note that the Application Scope (AS) is used to denote the services to which this policy applies (in this case, the GridFTP service on opteron8).

```
<?xml version="1.0" standalone="no" ?>
Xmlns:mpm="http://gcg.cs.virginia.edu/MyPolMan/PolicySchema"
<mpm:PolicyProperties>
<mpm:PolicyName>"UVaCG:MyPolMan:Sample:Policy#1"</mpm:PolicyName>
<mpm:AU>"/C=US/O=University of Virginia/OU=UVA Standard PKI User/emailAddress=jf4t@virginia.edu/CN=Jun Feng 3"</mpm:AU>
<mpm:ST>"Effective"</mpm:ST>
<mpm:VT from="09/15/2004" to=""></mpm:VT>
</mpm:PolicyProperties>
<wsp:All>
<disk:Resourceusage>
<disk:ResourceName>C@opteron8.cs.virginia.edu</disk:ResourceName>
<disk:Maxusage relative="true">40</disk:Maxusage>
</disk:Resourceusage>
</wsp:All>
</wsp:Policy>
```

Figure 3. A Sample Policy

Created policies can be uploaded to the MyPolMan policy service. This service is based on CredEx[40], an open-source, standards-based Web Service that facilitates the secure storage of credentials and enables the dynamic exchange of different credential types using the WS-Trust[41] token exchange protocol. We have extended CredEx to support management of arbitrary XML policies. This Tomcat-based web service exposes three key policy management operations:

- Storing or uploading a policy to the service
- Retrieving a policy from the service
- Removing a previously stored policy from the service

Policy uploaders can choose to associate a credential (either username/password or X.509 certificate) with the stored policy, requiring subsequent managers of this policy to present the same credential. Client side libraries for both Java and .NET platforms have been created to allow clients on multiple platforms to manipulate policies in MyPolMan. In MyPolMan, services called policy agents are used to facilitate policy distribution between the policy service and policy consumers i.e. policy enforcement points, such as the GridFTP service. Policy agents are implemented as .NET web services co-located with policy consumers. While policy consumers can “pull” policies directly from the policy service using that service’s API, the policy agents allows “push” based dissemination of policies.

I have implemented the GridFTP protocol on the .NET platform to enforce some classes of storage resource policies [32]. The GridFTP service act as both a policy decision point (PDP) and a policy enforcement point (PEP). The current implementation is able to understand and enforce
service availability policies, certain type of quota policies and a maximum stream number policy. Figure 4 shows a screen output of a .NET GridFTP client trying to upload files to opteron8. In this figure, a Grid user first try to do an FTP put operation on a file that causes the Grid quota to be exceeded. This operation is rejected due to policy violation. The user then does a Put of another smaller file with success. The sizes of these two files are displayed using “lls” command (local list) in this figure.

![Figure 4. Quota Policy in Enforcement](image)

6 Summary

The lack of explicit resource usage policy management and enforcement is a significant problem in Grids that is increasingly limiting the potential and deployment of the Grid technologies. While there are a number of small projects that address some aspects of the overall problem, there is no comprehensive, end-to-end solution for explicit resource usage policy in Grids today. This project makes important contributions in resource usage policy expression, management and enforcement for Grid systems.

There are a number of key questions that this research project will address:
- What policies are appropriate and sufficient for resource providers?
- What policies are appropriate and sufficient for VO?
- How these policies can be expressed explicitly and manipulated?
- How to enforce these policies?
- What are the cost and benefits of having these policies in Grids?

For each of these questions, I have corresponding work to investigate and I expect following contributions when the proposed research is done.
- Usage policies for storage and CPU resource provider on Grids
- Fair sharing policies at VO level
- Metrics to evaluate the fairness of a Grid/VO
• Policy model and language to express these policies
• Enforcement mechanisms of resource provider policies in the Grid software and protocol, such as GridFTP and GRAM.
• Enforcement mechanisms and systems for some VO fair sharing policies.
• Cost/benefits analysis of having these policies in Grids

References


[27] Marty Humphrey and Glenn Wasson. The University of Virginia Campus Grid: Integrating Grid Technologies with Campus Information Infrastructure, 2005 European Grid Conference (ECG 2005), Amsterdam, The Netherlands


[33] Jun Feng, Lingling Cui, Glenn Wasson, and Marty Humphrey, Policy Directed Data Movements in Grids, submitted for publication


[38] David. C. Chu, Marty Humphrey, Mobile OGSI.NET: Grid Computing on Mobile Devices, Fifth International Workshop on Grid Computing (Grid 2004), pp 182-191


