

Policy-Directed Data Movement in Grids

Jun Feng, Lingling Cui, Glenn Wasson and Marty Humphrey
Computer Science Department, University of Virginia, Charlottesville, VA 22904
{jf4t | lc3j | gsw2c | humphrey} @ cs.virginia.edu

Abstract

One of the guiding principles of the Grid is local (site) autonomy. Resource owners maintain control over their resources even when those resources are part of a larger Grid. In other words, resource sharing in Grids must be subject to the policies of the (local) resource owners. To date, not enough attention has been paid to describing, manipulating or enforcing explicit resource usage policies. Most existing Grid systems have either implicit resource usage policies (with ad-hoc enforcement mechanisms) or support only limited types of policies (e.g., security 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. This paper focuses on policies for Grid storage resources. We have identified classes of policies for Grid storage resource providers and have implemented an explicit policy-based architecture to manage and enforce them. This architecture consists of two components, MyPolMan, a service to express and manage policies, and .NET GridFTP, an enforcement mechanism for policy-based Grid data movement. We show how this system allows Grid users to access storage resources through a familiar API while allowing local system administrators to control resource utilization.

1. Introduction

Grid computing can be defined as secure collaboration across geographically-distributed resources. Often, a Virtual Organization (VO) is formed when multiple Physical Organizations (POs)

This work is supported in part by the Department of Energy Early Career Principle Investigator (ECPI) Program (PI: Marty Humphrey), and by the National Science Foundation under grants ACI-0203960, SCI-0438263, and SCI-0426972.

each contribute resources to a single resource pool shared by all. One of the main requirements of the POs is to maintain *site autonomy*, i.e. resource owners retain control over their own resources. This control is often embodied in terms of constraints on the use of associated resources by the VO, either explicitly or implicitly. These constraints can be viewed as a policy for the use of particular resources.

This paper addresses the management and enforcement of policies for controlling the utilization of *storage resources*. We call such policies resource utilization policies. There are many types of policies, e.g., security policies and scheduling policies, being used in Grids today. Security policy [1][2][3][4] allows a resource provider 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 mechanism. Resource utilization policy, on the other hand, defines constraints on resource use that incorporate information outside of the context of user identity or a single request message. Such policies can express constraints such as “the service can be accessed from 9am to 5pm” or “only 20% of the resource’s storage capacity can be used by the Grid”. Grid systems to date have provided only limited supports for this type of policies, typically only addressing CPU resources via scheduling [5] or job queues associated with single large clusters [6]. Providing more fine-grained control over resource utilization to resource owners is, we argue, necessary to build truly large-scale Grids.

We concentrate on data storage and data transfer policies because they are some of the most common Grid activities, often taking place as part of larger activities, e.g. running large computational jobs. We have identified some policies suitable for Grid storage resource providers by reviewing existing Grid projects and systems such as TeraGrid [7], LCG/EGEE [8] and SRB [9]. This lead to our explicit policy-based architecture, based on the IETF/DMTF generic policy framework [10][11]. A generic policy management

system, called MyPolMan, was developed to create, publish and provide policies to Policy Enforcement Points (PEPs). GridFTP [12][13] clients and services were chosen as ideal policy enforcement points because of the near ubiquity of their deployment in Grids today and the familiarity of their interface to most Grid users. While the GridFTP protocol and existing GridFTP implementations provide security for Grid data transfer, they do not utilize arbitrary policies such as resource utilization policies. Because there is no explicit policy support in GridFTP, the resource provider's (implicit) desires for sharing their resource(s) can be violated by Grid users. We have built an implementation of GridFTP on the Microsoft .NET framework which incorporates extensions to the GridFTP protocol. This service (called .NET GridFTP) can perform Grid data transfers to/from Windows machines and enforce the resource utilization policies that are stored in MyPolMan. Unless resource-sharing policies are treated explicitly, and services built to understand and enforce these policies, resource providers may begin to find that the Grid software does not facilitate sufficient local control. Therefore, when a site's shares a resource between local and non-local users and that resource becomes consumed by non-local users, the site may have no choice but to discontinue its participation in the Grid.

The rest of this paper is organized as follows. Section 2 describes policies for storage resource providers. Section 3 describes the two main components in our system, MyPolMan and .NET GridFTP. Section 4 illustrates how a policy regulated data transfer can be performed through an end-to-end usage scenario. Section 5 discusses related work and Section 6 concludes.

2. Policies

Identifying suitable and sufficient resource utilization policies for each type of resource in the Grid is a challenging problem. Storage resources are provided to the Grid for hosting data and as such, their utilization by Grid users consumes a fraction of the storage capacity and a fraction of network bandwidth to the resource provider. A storage resource provider may wish to place limits on this consumption to more precisely control how his resources are used (to, for example, reserve capacity for local, non-grid, users or to not exceed bandwidth limits allowed by a service-level agreement). In addition, a storage provider may have specialized abilities, such as backup, whose utilization can also be controlled through policy.

We have examined some existing Grid projects, such as the TeraGrid, LCG/EGEE and Data Grid systems such as SRB to understand their (implicit) storage utilization policies, and policy needs. These fairly large scientific computing Grids represent one use case. We have also studied appropriate storage resource policies for desktops and handheld devices which will inevitably be incorporated into Grids as their capabilities increase [15][16]. Based on our investigation, we have identified that the following type of policies should be necessary 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:\GridFTPRoot\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 relative 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 [17].

All of these policies can be uploaded, manipulated and retrieved using the MyPolMan service. Our GridFTP implementation now can enforce service availability policy, certain quota policies and the maximum number of streams policy.

3. A System for Policy-Directed Data Movement

Our policy system consists of two components, MyPolMan, a policy management system that can upload, manage and distribute policies and .NET GridFTP, a data transfer application that enforces these policies. This section describes these components.

3.1. MyPolMan

MyPolMan is a general policy management system for Grids that allows administrators (or users) to publish policies to a policy repository that can then be retrieved by other system components as needed. Figure 1 shows the three main components of MyPolMan the policy authoring tools, the policy repository services and the policy agents.

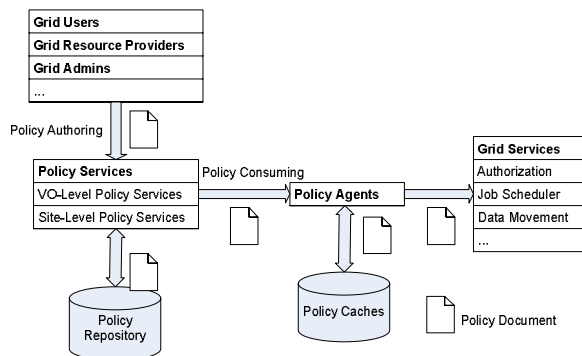


Figure 1. MyPolMan components

In MyPolMan, policies are expressed in XML that conforms to the WS-Policy [18] specification. Since MyPolMan can be used to manage many types of policy (not just data transfer policy), a variety of policy creation tools can be used to develop the policies that are published in MyPolMan. While a discussion of these tools is beyond the scope of this work, Figure 2 shows an example policy for limiting disk consumption. 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. For example, the policy of Figure 2 uses the <disk:MaxUsage> element to define the relative amount of a storage resource that can be used for Grid related activity.

```
<?xml version="1.0"?>
<wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:mpm="http://gcg.cs.virginia.edu/MyPolMan/PolicySchema" xmlns:disk="http://gcg.cs.virginia.edu/MyPolMan/Schema/disk">
  <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:AS>
      opteron8.cs.virginia.edu/GridFTP
    </mpm:AS>
    <mpm:ST>
      "Effective"
    </mpm:ST>
    <mpm:VT from="09/15/2004" to=""></mpm:VT>
  </mpm:PolicyProperties>
  <wsp:All>
    <disk:ResourceUtilization>
      <disk:ResourceName>
        C@opteron8.cs.virginia.edu
      </disk:ResourceName>
      <disk:MaxUsage relative="true">
        40
      </disk:MaxUsage>
    </disk:ResourceUtilization>
  </wsp:All>
</wsp:Policy>
```

Figure 2. A sample policy

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).

Created policies can be uploaded to the MyPolMan policy service. This service is based on CredEx [19], 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[21] 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. When a new policy is stored (or an existing policy is modified) in the policy service, the policy’s Applicable Scope field can be used to determine which policy agents should receive this new information. The policy can then be transferred to the policy agent who caches it on the local file system. This caching increases performance and allows multiple co-located policy-enforcement points to utilize the same policy agent. It should be noted that the GridFTP service itself could be used to transfer new policies (since policies are simply XML files). However, we have chosen to implement the policy agent web service because it is easier to configure a web service to securely interact with multiple policy services than it is to configure GridFTP to do the same using the gridmap file. Using an external authorization service, such as VOMS [20], may alleviate this problem.

3.2. GridFTP Protocol and Implementation on .NET

GridFTP [12] is a data transfer protocol for accessing distributed data on the Grid. It is based on the RFC 959 “File Transfer Protocol”, RFC 2228 “FTP Security Extensions”, and RFC 2389 “Feature Negotiation Mechanism for the File Transfer Protocol”. The GridFTP protocol is optimized for the high-performance, secure and reliable data transfer in high-bandwidth wide-area network.

We have implemented the GridFTP protocol on the .NET platform. Our implementation supports the main extensions of GridFTP v1 and some features of GridFTP v2. It also incorporates a new transfer mode specifically designed for the transfer of large amount of small files over wide area network. In addition, the GridFTP service has been modified to enforce data transfer policies on each operation requested by clients. Evaluation has shown that our implementation is both interoperable with and has comparable performance to the implementation included with the Globus Toolkit version 4 (GT4) [14].

On the server, .NET GridFTP runs as a Windows service and manages the storage resource. On the client side, the .NET GridFTP command line client can perform the GridFTP operations supported by the GridFTP service and report additional information if the requested FTP operations are rejected by the service due to policy violations.

4. Policy Enforcement

To understand how GridFTP enforces the policies stored in MyPolMan, consider the following scenario. A site administrator is contributing storage on the machine “opteron8.cs.virginia.edu” to the Grid. However, the administrator only wishes to contribute a maximum of 40% of the capacity of disk C to be used by Grid.

We have developed an ASP.NET-based web interface to compose quota policies. Using this interface, the administrator specifies the author name, policy name, applicable scope and other common properties of this policy. A second form allows the administrator to define quotas for each disk within the applicable scope (in this case on opteron8) using either absolute or relative volume. When the administrator finishes, a XML policy file, such as that shown in Figure 2, will be generated and displayed for final confirmation. If the administrator accepts the policy, it is published to a pre-configured MyPolMan policy service.

The MyPolMan policy service stores the policy in its policy repository and then pushes the policy to appropriate policy agents. These are determined based on the applicable scope property, in this case the policy will be sent to the policy agent on opteron8. Once the opteron8 policy agent has received the policy and saved it to its local policy cache, the policy is available to any policy enforcement points running on opteron8.

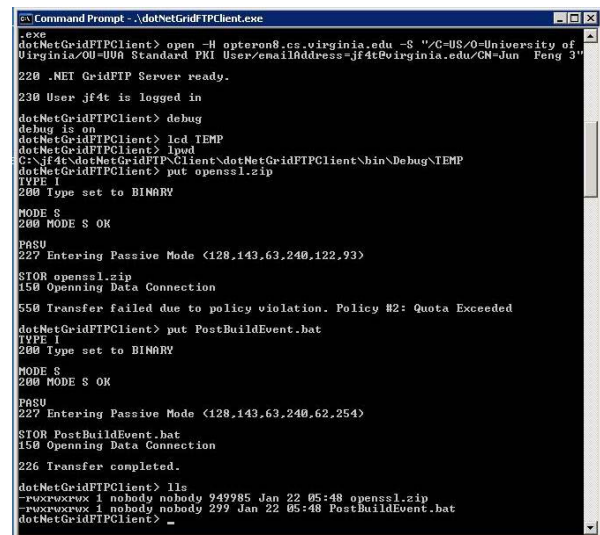
Any modifications made by the administrator to the policy will cause these steps to be repeated.

Enforcement of this policy requires information on how much disk space is used by Grid activities. In general, it is difficult to know how much space is used by “the Grid” because grid-related files can be created by multiple mechanisms. For example, a remote computation (typically considered a grid activity) can produce output through direct interaction with the host operating system. This output can appear at any time and be stored in any location on disk. While it may be possible to assume that every file owned by a user listed in the gridmap file is a “grid file”, this may not be appropriate in all circumstances. For this project, we have adopted a simpler approach based on file location. All GridFTP activities (storage/retrieval) take place within a GridFTPRoot directory and all data stored in this directory are considered grid related. This approach seems reasonable since appropriate permissions can be set to constrain remote (i.e. “Grid only”) users to work in this root directory and its sub-directories while allowing other users to have a combination of grid and non-grid files.

The GridFTP service acts as both a policy decision point (PDP) and a policy enforcement point (PEP), making decisions to accept or reject a file upload request based on current disk usage information and the incoming file size. Ideally, the PDP would make an accept/reject decision when a new file upload request is received. However, due to the FTP protocol’s limitation, the incoming file size is unknown until the file transfer completes. This means that the PDP is actually consulted after the file transfer operation completes and at that time the received file can either be accepted (and moved from temporary storage) or rejected (and therefore deleted). An ultimate solution for this problem will be a revision to the GridFTP protocol to make the incoming file size known by GridFTP server prior to the actual file transfer. We will let the Grid community know this problem and we expect the fix could appear in the new GridFTP v2 protocol.

If an operation is rejected due to a policy violation, the policy identifier and a short description will be returned on the GridFTP control channel, giving the Grid user information on why their requested operation did not happen. Figure 3 shows a screen output of a .NET GridFTP client trying to upload files to opteron8. In this figure, a Grid user first tries to do an FTP Put operation on a file that causes the Grid quota to be exceeded. This operation is rejected due to policy

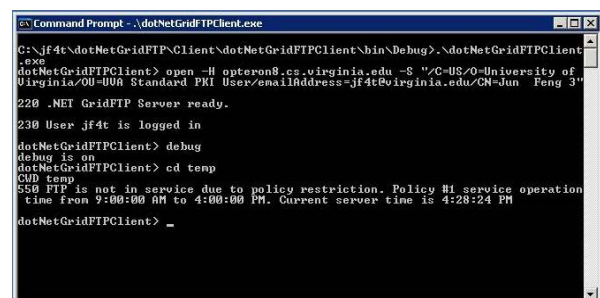
violation. The user then does a Put of another smaller file with success. The sizes of these two files are displayed using “lls” command (local list) in this figure.



```
.\dotNetGridFTPClient.exe
.exe
dotNetGridFTPClient> open -H opteron8.cs.virginia.edu -S "/C:/US/O-University of
Virginia/OU=UOA Standard PKI User/emailAddress=jf4t@virginia.edu/CN=Jun Peng 3"
220 .NET GridFTP Server ready.
230 User jf4t is logged in
dotNetGridFTPClient> debug
debug is on
dotNetGridFTPClient> lcd TEMP
dotNetGridFTPClient> lpwd
C:\jf4t\dotNetGridFTPClient\dotNetGridFTPClient\bin\Debug\TEMP
dotNetGridFTPClient> put opensecl.zip
dotNetGridFTPClient>
TYPE I
200 Type set to BINARY
MODE S
200 MODE S OK
PASU
227 Entering Passive Mode (128.143.63.240,122.93)
$TOR opensecl.zip
150 Opening Data Connection
550 Transfer failed due to policy violation. Policy #2: Quota Exceeded
dotNetGridFTPClient> put PostBuildEvent.bat
dotNetGridFTPClient>
TYPE I
200 Type set to BINARY
MODE S
200 MODE S OK
PASU
227 Entering Passive Mode (128.143.63.240,62.254)
$TOR PostBuildEvent.bat
150 Opening Data Connection
226 Transfer completed.
dotNetGridFTPClient> llS
-rwxrwxrwx 1 nobody nobody 949985 Jan 22 05:48 opensecl.zip
-rwxrwxrwx 1 nobody nobody 279 Jan 22 05:48 PostBuildEvent.bat
dotNetGridFTPClient> _
```

Figure 3. Quota policy enforcement

In addition to disk quota policies, our current implementation can also enforce service availability policy and a policy on maximum number of streams. While the creation and distribution mechanisms for these policies are similar to the disk quota policy, the enforcement mechanism is different. To enforce the service availability policy, a policy decision is made on every FTP operations other than the command channel setup and close. The GridFTP server compares the current system time with the operation period specified in the policy, and then either rejects or accepts incoming FTP operations. Figure 4 shows this policy in action. The GridFTP service is configured using a service availability policy to only serve requests between 9:00AM – 4:00PM. Figure 4 shows a client attempting to do a change directory operation outside of this time window and being rejected.



```
.\dotNetGridFTPClient.exe
C:\jf4t\dotNetGridFTPClient\dotNetGridFTPClient\bin\Debug> .\dotNetGridFTPClient
.exe
dotNetGridFTPClient> open -H opteron8.cs.virginia.edu -S "/C:/US/O-University of
Virginia/OU=UOA Standard PKI User/emailAddress=jf4t@virginia.edu/CN=Jun Peng 3"
220 .NET GridFTP Server ready.
230 User jf4t is logged in
dotNetGridFTPClient> debug
debug is on
dotNetGridFTPClient> cd temp
C:\temp
550 FTP is not in service due to policy restriction. Policy #1 service operation
time from 9:00:00 AM to 4:00:00 PM. Current server time is 4:28:24 PM
dotNetGridFTPClient> _
```

Figure 4. Service availability policy enforcement

Enforcement of the maximum number of streams policy requires a policy decision when an FTP OPTS command, in which the client specifies the number of streams, is received. Figure 5 shows a GridFTP service that is configured to allow a maximum of 2 streams. A client requests 3 streams to transfer a file and is rejected. Then he tries 2 streams and succeeds.

```

C:\j4t\dotNetGridFTPClient> dotNetGridFTPClient.exe
dotNetGridFTPClient> open -H opteron8.cs.virginia.edu -S "/C:/US/0-University of Virginia/01-008 Standard PKI User/EmailAddress-j4t@virginia.edu/CN-Jun Feng 3"
220 .NET GridFTP Server ready.
230 User j4t is logged in
dotNetGridFTPClient> debug
debug is on
dotNetGridFTPClient> parallel 3
dotNetGridFTPClient> set opnssl.zip
TYPE I
200 Type set to BINARY
MODE E
200 MODE E OK
OPTS RETR Parallelism=3,3,3;
550 OPTS failed due to policy violation. Policy:#3 no more than 2 streams
transfer file failed
dotNetGridFTPClient> parallel 2
dotNetGridFTPClient> set opnssl.zip
TYPE I
200 Type set to BINARY
MODE E
200 MODE E OK
OPTS RETR Parallelism=2,2,2;
200 OPTS successful
PORT 128.143.63.240,14.137
200 Port command successful. 128.143.63.240:3721
RETR opnssl.zip
150 Opening Data Connection
226 Transfer completed.
dotNetGridFTPClient>

```

Figure 5. Stream policy enforcement

5. Related Work

IETF [10] and DMTF [11] have a generic policy framework including four major components, the policy management tool, the policy repository, the policy enforcement point (PEP) and the policy decision point (PDP). IETF and DMTF also define a common information model [22] 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”. We use this policy information model in our prototype implementation. The Policy Research Group [23] at the Global Grid Forum (GGF) has also adopted this IETF/DMTF policy framework by defining four similar software components. However, to date, there has been no implementation of a policy system on Grids based on the GGF’s work.

Policies are often encountered explicitly in resource management systems, especially schedulers. For example, the Maui Scheduler [24] operates as a policy engine for controlling resource allocations to jobs while concurrently optimizing the use of managed resources. Legion can allow host administrators to specify some job admission policies such as “admit new job only when CPU usage in past 5 minutes is under 70%” [5]. Condor’s ClassAds mechanism [6]

can match the job’s requirements with advertised resources. In ClassAds, resource policies are expressed as “constraints” in their advertisement. However, all these systems are limited to managing CPU resources.

Catalin et, al have identified usage policies (UP) for both sites and VOs regarding resource allocations [25]. The work also presents the expression of these UPs and a distributed architecture for UP-based scheduling in a Grid environment. However, Catalin’s work also only addresses CPU resources. The general purpose policy management system MyPolMan, can support many different policy enforcement mechanisms and therefore can be used to manage policies for many types of resources.

Wasson and Humphrey focus on the policies at the VO level [26]. They identify three general policies regarding resource utilization by which VOs might operate and present the ramifications of each policy on the VO’s day-to-day operation and the VO’s ability to actually enforce the policy. A prototype software simulation implemented the “you-get-what-you-give” policy. This work began to quantify the costs and benefits of attempting to enforce Grid-wide policy. While arguably simplistic, it was useful in conducting experiments to refine the issues and approaches.

In the Web Service world, WS-policy provides a flexible and extensible grammar for expressing the capabilities, requirements and general characteristics of entities in a Web Services-based system. WS-Policy 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 [27]. Policies can be discovered via WS-MetadataExchange [28] which defines the request-response based protocol between Web services and clients. We are currently investigating the use of WS-MetadataExchange as a policy distribution mechanism between the policy repository and policy agents.

Outside the Grid Computing, Autonomic Computing [29] is an approach towards self-managed complex computing systems with minimum human interference. The core component of Autonomic Computing is often a policy-based management infrastructure that simplifies the management and automation of complex systems. A particular policy system PMAC [30] is similar to our system. However, our policy system and enforcement mechanisms focus on the usage policies which regulate the resource consumption by Grids, while Autonomic Computing

systems, such as PMAC, generally focuses on using policies to manage enterprise services to automatically react to certain system conditions, through which self-managed services can be achieved.

6. Conclusion and Future Work

The goal of this work was to increase the degree of control that resource owners have over Grid storage resources (beyond joining the Grid or not). We have identified classes of policies for Grid storage resource providers and have implemented an explicit policy-based architecture to manage and enforce them. We have shown that using this system allows Grid users to access storage resources through a familiar API (GridFTP) while allowing local system administrators to control resource utilization. In the future, we will add more policies for storage resources into the system and extend our work to include policy and enforcement for other resource types, such as advanced network bandwidth consumption policies. In particular, we are interested in using new technologies such as GMPLS [31], to enable the partitioning and reservation of network resources. Another important direction will be research on cross-site job and data scheduling that takes into account policy of multiple resources.

References

- [1] M. Thompson. 2001. Akenti Policy Language. <http://www.itg.lbl.gov/security/Akenti/Papers/PoligyLanguage.html>.
- [2] L.Pearlman, V.Welch, I. Foster, C.Kesselman, S.Tuecke. A Community Authorization Service for Group Collaboration. *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
- [3] K.Keahey, V.Welch. Fine-Grain Authorization for Resource Management in the Grid Environment. *Proceedings of Grid Workshop 2002*
- [4] M. Lorch, D. Kafura, I. Fisk, K. Keahey, G. Carcassi and T. Freeman. Authorization and Account Management in the Open Science Grid. *Proceedings of Grid Workshop 2005*
- [5] Andrew S. Grimshaw, Adam Ferrari, Fritz Knabe, Marty Humphrey. Legion: An Operating System for Wide-Area Computing, *IEEE Computer*, 32:5, May 1999:29-37
- [6] Rajesh Ramen, Miron Livny and Marvin Solomon, Matchmaking: Distributed Resource Management for High Throughput Computing, *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, July 28-31, 1998, Chicago, IL
- [7] TeraGrid Policies. Available at http://www.teragrid.org/userinfo/guide_tgpolicy.html
- [8] LCG/EGEE Security and Availability Policies. https://edms.cern.ch/file/428008/LAST_RELEASED/LCG_Policy.pdf
- [9] Storage Resource Broker at <http://www.sdsc.edu/srb/>
- [10] The IETF Policy Framework Working Group. Charter <http://www.ietf.org/about/working/sla.php>
- [11] Distributed Management Task Force Policy Working Group. <http://www.dmtf.org/about/working/sla.php>
- [12] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, "GridFTP: Protocol Extensions to FTP for the Grid", 2001. <http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf>
- [13] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke, *Data Management and Transfer in High Performance Computational Grid Environments*, *Parallel Computing Journal*, Vol.28 (5), May 2002, pp 749 – 771
- [14] Jun Feng, Lingling Cui, Glenn Wasson and Marty Humphrey, Towards Seamless Grid Data Access: Design and Implementation of GridFTP on .NET, *Proceedings of Grid Computing Workshop (Associated with Supercomputing 2005)*, Nov 13-14, Seattle, WA
- [15] Andrew. A. Chien, Brad Calder, Stephen Elbert and Karan Bhatia. Entropia: Architecture and Performance of an Enterprise Desktop Grid System, *Journal of Parallel and Distributed Computing*. 63(5): 597-610, 2003
- [16] David. C. Chu, Marty Humphrey, Mobile OGSINET: Grid Computing on Mobile Devices, *Fifth International Workshop on Grid Computing (Grid 2004)*, pp 182-191
- [17] T.J. Hacker, B.D. Nobel, and B.D. Athey, The Effects of Systemic Packet Loss on Aggregate TCP flows, *Proceedings of Supercomputing 2002*, Baltimore, Maryland.
- [18] Don Box, et. al. Web Services Policy Framework (WS-Policy), 3 September 2004, BEA, IBM, Microsoft and SAP
- [19] D.Del Vacchio, J. Basney, N. Nagaratnam, and M. Humphrey. CredEx: User-Centric Credential Selection and Management for Grid and Web Services. *Proceedings of the 2005 IEEE International Conference on Web Services (ICWS 2005)*. July 12-15, 2005. Orlando, FL.
- [20] R.Alferi, R.Gecchini, V.Ciashini, L. dell Agnello, A. Frohner, A.Gianoli, K.Lorentey, and F.Spataro. VOMS: an Authorization System for Virtual Organizations, 1st European Across Grids Conference, Santiago de Compostela, February 13-14, 2003
- [21] S. Anderson, et al. Web Services Trust Language (WS-Trust), v. 1.1. May 2004 Available at <ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf>
- [22] B.Moore et.al., Policy Core Information Model – Version 1 Specification, RFC 3060, February 2001
- [23] Policy Research Group, at Global Grid Forum http://www.ggf.org/L_WG/wg.htm
- [24] MAUI, Maui Scheduler, Center for HPC Cluster Resource Management and Scheduling, www.supercluster.org/maui

- [25] C. Dumitrescu, I. Foster, M. Wilde, Usage Policy-based Resource Scheduling in VOs, *Grid Workshop 2004* (associated with Super Computing 2004)
- [26] G. Wasson and M. Humphrey, Policy and Enforcement in Virtual Organizations. In *4th International Workshop on Grid Computing (Grid2003)* (associated with Super Computing 2003). Phoenix, AZ. Nov 17, 2003.
- [27] Siddharth Bajaj, et. al. Web Services Policy Attachment (WS-PolicyAttachment), Available at <http://www-106.ibm.com/developerworks/library/specification/ws-polatt/>
- [28] K. Ballinger, D. Box, et. al, Web Services Metadata Exchange. <http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-metadataexchange.pdf>
- [29] R. Murch. *Autonomic Computing*, IBM Press, ISBN 013144025X
- [30] Policy Management for Autonomic Computing (PMAC) at <http://www.alphaworks.ibm.com/tech/pmac>
- [31] Generalized Multiprotocol Label Switching, Available at <http://www.iec.org/online/tutorials/gmpls/>