A Framework for Managing Inter-Site Storage Area Networks using Grid Technologies

Fritz McCall
University of Maryland Institute For Advanced Computer Studies
fmccall@umiacs.umd.edu

Ben Kobler
NASA Goddard Space Flight Center
ben.kobler@nasa.gov

Mike Smorul
University of Maryland Institute For Advanced Computer Studies
toaster@umiacs.umd.edu

Abstract

The NASA Goddard Space Flight Center and the University of Maryland Institute for Advanced Computer Studies are studying mechanisms for installing and managing Storage Area Networks (SANs) that span multiple independent collaborating institutions using Storage Area Network Routers (SAN Routers). We present a framework for managing inter-site distributed SANs that uses Grid Technologies to balance the competing needs to control local resources, share information, delegate administrative access, and manage the complex trust relationships between the participating sites.

1. Introduction

1.1 A Background Introduction to SAN Routing

SAN Routers that use protocols like iSCSI, FCIP, and iFCP to interconnect geographically distributed SANs over high-speed IP networks are typically deployed to support business continuity and disaster recovery for applications that:

- Replicate data between data centers that are connected over wide-area IP networks in order to increase data availability or to support efficient Disaster Recovery [1]
- Extend SAN connectivity within the data center or to remote sites within a metropolitan area using IP networks as a ubiquitous interconnect for the purpose of server clustering, LAN-free remote tape backup, or remote data access. [1]
- Expedite adoption of networked storage by reducing the cost of the equipment and expertise needed to deploy a SAN infrastructure based solely on Fibre-Channel connectivity. [1]

Deployments like these represent the state-of-the-art for SAN extension technologies in production environments. Basic data transfers for these applications proved to be scalable and robust on 10-Gigabit networks with transcontinental latencies during data transfer exhibitions at SC2002 [2] [3].

There is also a growing interest in running shared file systems over IP SANs between geographically distributed sites in order to support high-performance, data-intensive, and grid computing. Although these applications are much more experimental than those described above, previous work presented at MSST2003 demonstrated the feasibility of running the CentraVision File System (CVFS) and the Global File System (GFS) over IP SANs that connected the Goddard Space Flight Center, the University of Maryland, and the Gilmore Creek Alaska Ground Station Facility [4]. Related work at the San Diego Supercomputer Center and on the Teragrid showed the feasibility of running the Global Parallel File System over 10 and 30 Gigabit-Ethernet IP networks that connected the San

Diego Supercomputer Center (SDSC), the National Center for Supercomputing Applications (NCSA) and the show floors at SC2003 and SC2004. [5]. In the past year, both the Teragrid and the Distributed European Infrastructure for Supercomputing Applications began to deploy the first production shared global file systems [6].

The Goddard Space Flight Center and the University of Maryland maintain a test bed as a continuation of the research presented at MSST2003, and in order to support new evaluations of shared file systems and IP SAN technologies within the Metropolitan-Area Network established by the Mid-Atlantic Crossroads. Figure 1 illustrates our test bed.

Our test bed is built on McData Eclipse Storage Routers installed at NASA Goddard Space Flight Center (GSFC) and the University of Maryland Institute for Advanced Computer Studies (UMIACS). Each site has existing high-end computing, data storage, and visualization systems in order to support experimentation on specific inter-site SAN applications. GSFC and UMIACS are connected with Gigabit Ethernet links through the Mid-Atlantic Crossroads Meta-pop (MaX) and high-speed wide-area connectivity through Internet2.

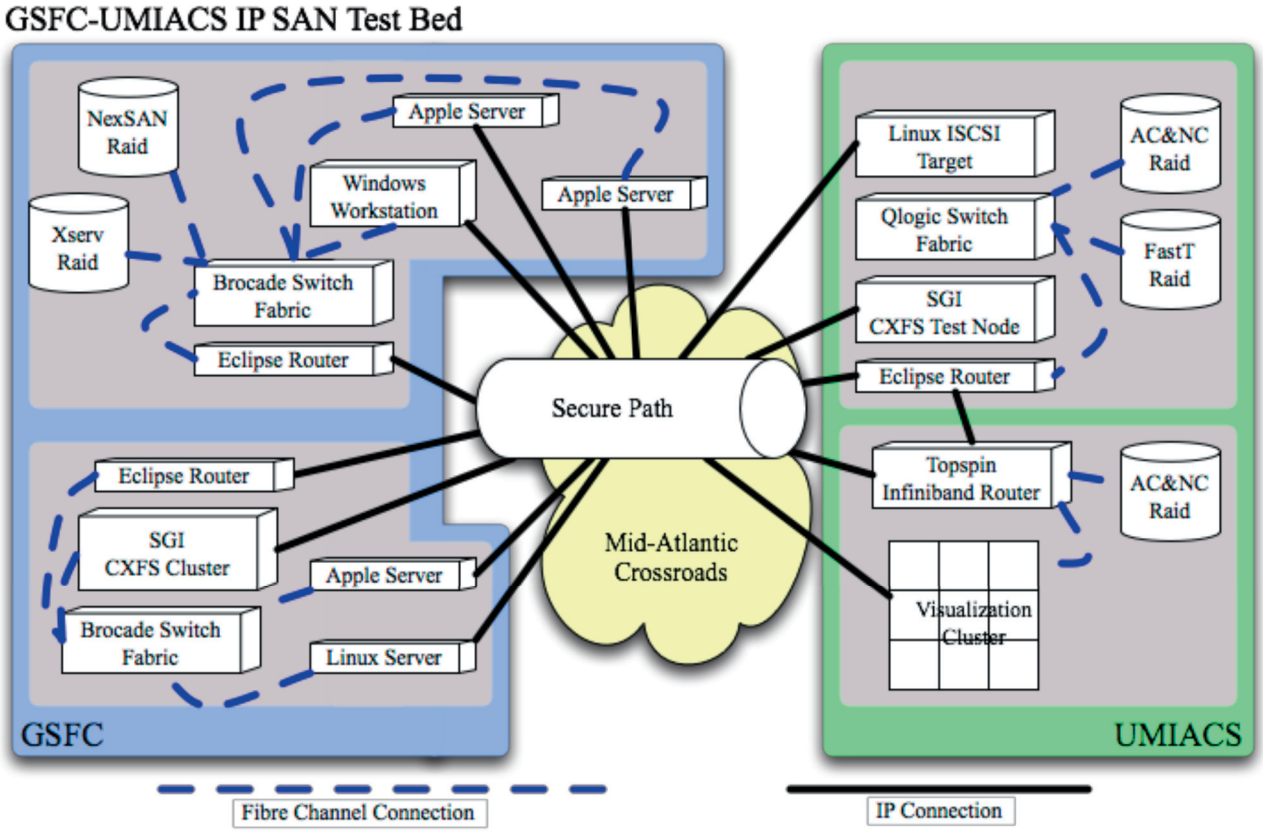

Figure 1. The GSFC-UMIACS IP SAN Testbed

In our test bed, we are particularly interested in developing mechanisms for managing IP SANs that span multiple independent administrative organizations, because distributed administration has been a particularly difficult aspect of our previous

work. Current management systems lack mechanisms to establish strong authentication and fine-grained trust between participating sites. Instead, most command-line interfaces provide just a few levels of authorization, such as read-only, read-write or administrative access. We also lack mechanisms to deal with the dynamic nature of inter-site SANs in which we assume that networks will change and sites will occasionally reconfigure or expand their environment. Finally, we need mechanisms that can reduce the level of inter-site coordination and administrative intervention needed to configure and maintain IP SAN.

This paper presents a prototype management system that allows participating sites to delegate administrative authority for specific administrative functions using secure Web Services. We discuss the overall design of our management system, its software components and security model, and a sample administrative application that will operate in the GSFC-UMIACS IP SAN test bed.

1.2 Our approach: Applying grid technologies to inter-site SAN management

The overall approach of our prototype management system is to apply Grid-computing technologies to the problem of delegating administration and managing trust between participants in an inter-site SAN. Each site runs a management server that supports remote procedure calls for two secure web services: an "invocation" service that allows authenticated users to invoke trusted administrative scripts at remote sites and a "rights" web service that applies fine-grained authorization policies to allow or deny authenticated user requests. Several technologies are needed to secure the system: all of its communications are encrypted using the Secure Sockets Layer, and both of its web services are secured according to the OASIS Web Services Security Specification (WS-Security). A Public Key Infrastructure (PKI) identifies participating users, sites, and services with X.509 certificates.

In practice, our system allows users to remotely execute shared administrative scripts through a command-line interface to the Invocation Web Service in order to manage systems at remote sites in an inter-site IP SAN. Their requests are authenticated through the Public Key Infrastructure and authorized by the Rights Web Service that maintains the database of the site's trusted users and their permissions. Our system also provides an interface that allows each site to independently define what administrative scripts they would like to share with specific remote users. Figure 2. Provides an overview of the major software components in our management system, their functionalities, and their relationships.

The main goals of our prototype are to demonstrate:

- methods for strong authentication and fine-grained authorization that can support distributed storage management,
- a software environment in which local sites can retain control of their own resources even while delegating some administrative tasks to remote collaborators,
- tools to reduce the level of inter-site coordination and administrative intervention required to manage an IP SAN that operates between collaborating but independent organizations.

Our prototype addresses each of these goals with software components that have been implemented in the Grid Computing community.

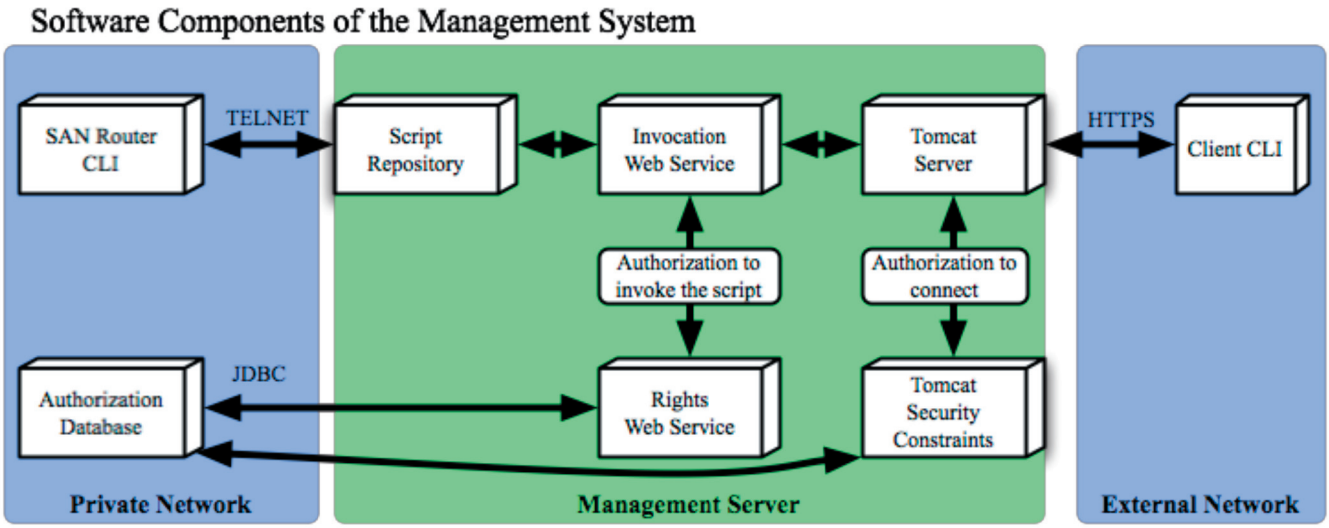

Figure 2. Software Components of the Management System

2. Description of Software components

2.1 The Management Server

The core of our system is the management server, which handles all of the inter-site management requests through its secure web services, manages the authorization databases, and executes the administrative scripts on behalf of remote users. It is built on a Secure Sockets Layer-enabled Apache Tomcat Server [7] with Java 1.5 and the Unlimited Strength Cryptography extensions from Sun Microsystems. We configure Tomcat with a number of software packages and security enhancements to support:

- Secure transport based on mutually authenticated connections through the Secure Hypertext Transport Protocol (HTTPS) in which both the client and the server must present trusted X.509 certificates in order to communicate,
- Apache Axis- an implementation of the Simple Object Access Protocol (SOAP) recommendation to the W3C that we use as a messaging layer to implement our Web Services. [8]
- Apache WSS4J- an implementation of WS-Security that we use to implement message level security for our web services. [9] [10]
- Security constraints within the Tomcat server that limit access to the web services to trusted users that have valid entries in our authorization database.

The software configuration that supports secure web services on the management server is very similar to the Producer-Archive Workflow Network's (PAWN) Receiving Server that was presented at MSST2005. [11]

The management server also hosts a MySQL relational database management system [12] as an authorization database. It contains a table of authorized users, which we represent by X.509 Subject Identifiers, and a table of permissions that grant or deny those users access to specific administrative scripts. Both Tomcat and the Rights Web Service use these databases to authorize user requests.

2.2 The Invocation Web Service

The invocation web service provides the programmatic interface for our system's

remote management capabilities. When a user requests the remote execution of an administrative script, the Invocation Web Service processes the request as follows:

1. It receives the request as a SOAP message over a mutually-authenticated and encrypted SSL connection,
2. It uses WSS4J to ensure that the message is secure. It confirms that the message is signed by a valid X.509 certificate and that it has not been tampered with in any way.
3. It calls the Rights Web Service to confirm that the user is trusted and authorized to call the requested administrative script.
4. It checks the requested scripts arguments and inputs for disallowed characters or potentially malicious strings.
5. It executes the requested script and returns the standard output and error to the remote user.

It also provides an interface with which users can browse available scripts and an interface that allows each site to manage the repository of scripts that they wish to share.

2.3 The Script Repository

Our prototype administrative scripts are written in Expect [13], because it provides a convenient interface for configuring the McData Eclipse Routers using their command line interface through the telnet protocol. Expect is an extension of tcl that is frequently used by systems administrators to automate complex configuration tasks through terminal interfaces [14].

In our environment at the University of Maryland, most administrative scripts are implemented in Perl, Expect, or Shell Scripts, and we wanted to ensure that local administrators would be able to easily author, check, and update the scripts as needed. Our approach also allows us to share third-party binary programs whose functionality we might not be able to replicate in a home-grown program. Administrators and vendors already have existing tools for troubleshooting and managing filesystems, SANs, and IP networks, and we wanted to be sure that they could be use in our environment whenever it was possible and practical.

The careful development of administrative scripts is an important aspect of the system's overall security model. The security model for our administrative scripts is very similar to Common Gateway Interface scripts, which can be particularly vulnerable to input injection attacks if they are not carefully written and called. Given their administrative privileges on the SAN routers, it is particularly important that all of the shared administrative scripts be carefully written and reviewed before they are put into service.

2.4 The Rights Web Service

The rights web service provides the programmatic interface to determine what remote users and sites are authorized to access each administrative script. It allows us to associate certificate common names that identify users with the scripts that they are able to invoke through the invocation web service.

The rights web service's is responsible for the following steps:

1. It re-checks the validity of the client's certificate, even though the Tomcat server has validated the certificate during mutual authentication.
2. It checks that the client certificate is not listed in the Certificate Revocation List published by our Certificate Authority.
3. It extracts the Common Name from the client's X.509 certificate and queries the database to be sure that it is listed as a trusted user in our authorization database.
4. It ensures that the trusted user is authorized to invoke the requested administrative

script by querying the permissions table in out authorization database.

It should be noted that in a large grid installation, something like the Community Authorization Service would be a better choice because it would support greater functionality as an authorization system. However, we implemented a more basic authorization service because our test-bed is only intended to support just tens of sites with a relatively small number of test users.

2.5 The Classads Registry

We are also developing a central registry of administrative information about the distributed SAN based on the Classified Advertisements (classads) library developed as part of the Condor project at the University of Wisconsin.[15] Classads allow us to accurately describe resources in our distributed SAN as well as policies that should govern their use. The Classads language is particularly attractive for our administrative registry because it allows administrators at the invocation site to control how their site is described and the policy requirements for using their resource.

3. Ongoing Deployment in GSFC-UMD Test bed

3.1 A Sample Application

GSFC and UMIACS have developed and tested the prototype management framework and are in the process of deploying management servers in their IP SAN test bed. One of the main goals of this effort is to demonstrate the ability of our software to grant authority to a user so that he or she can bootstrap access to a remote SAN device and subsequently tear down that connection without any administrative intervention. In this section, we describe the configuration of the management system in our test bed, and our approach to delegate administrative control of a SAN logical unit number (LUN) from the test SAN at UMIACS to a user at GSFC.

3.2 Preparing the Local SAN environments at UMD and GSFC

In order to prepare the existing SAN environments for interaction with the new management system, each site's administrative staff will need to complete the following tasks:

- Strictly partition shared SAN devices from private SAN devices through a physical reconfiguration or through Zone or LUN-masking policies. Isolating shared resources helps each site retain control and ownership of their own systems. In this configuration, no SAN-attached device can ever be accessed by remote sites unless it is manually configured into the shared zone.
- Storage Routers must be configured for basic connectivity and functionality. Each router needs a public Ethernet connection that will communicate with other storage routers to move data across the inter-site SAN and a private Ethernet connection that needs to be isolated and protected from untrusted networks so that we can use it to safely administer the SAN Router using unencrypted protocols like Telnet or SNMP.
- Register and configure a unique mSAN identifier for each Storage Router and a unique Zone identifier for each shared SAN zone. Otherwise, identifier conflicts can occur and iFCP connections will fail.
- The Management Servers needs to be configured. It needs a public interface that remote users can access and it needs to be connected to the SAN router's private management network so that it can access the SAN router's command-line interface. It also needs to be configured with the router's login username and password.

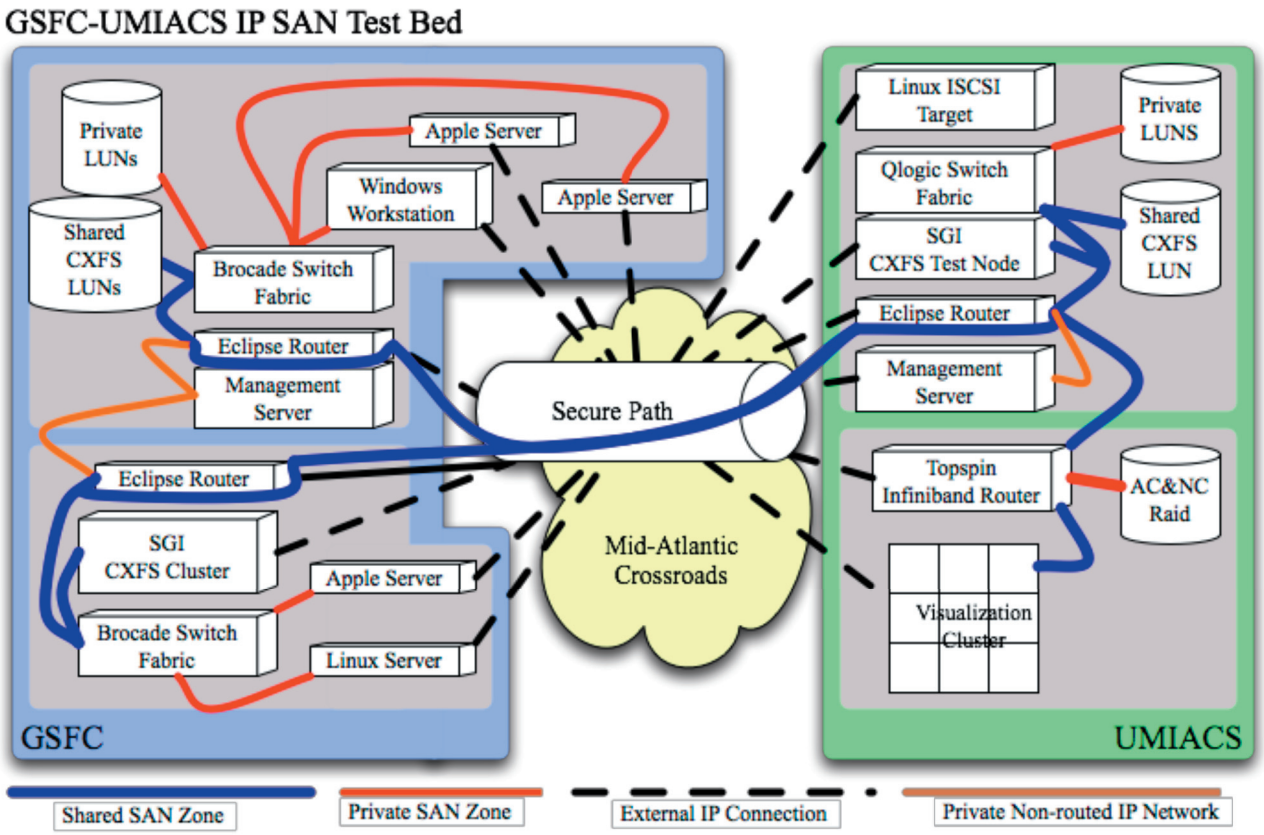

Figure 3. Preparing the GSFC-UMD SAN Test bed for our Management Software

Figure 3. illustrates how the test bed will be prepared for our management system. In particular, it shows the division of shared and private SAN zones, as well as how the management servers and participating hosts connect to each other over IP networks and to the SAN Routers. It also depicts the separation of the private management network at each site and the public network connectivity between each site.

3.3 Manually Sharing LUNS between UMD and GSFC

If we were to manually bootstrap an inter-site SAN connection between zones at UMD and GSFC without our management system in place, we would take the following steps:

1. We would determine the IP address and mSAN id of both the GSFC and UMIACS SAN routers.
2. We would ensure that the SAN routers have different mSAN ids.
3. We would agree on a common zone id to represent our new inter-site SAN on both storage routers. This zone id needs to match for bi-direction connectivity within the inter-site SAN, so this needs to be agreed upon before any configuration takes place.
4. Both sites would independently create a new zone with the common zone id.
5. Both sites would connect their storage router's R_PORT with the SAN fabric that they want to share, and associate the R_PORT with the newly created zone.
6. UMIACS would add the GSFC SAN Router's IP and mSAN identifier to the

access control list for the iFCP connections on our storage router, and we would specify that we will share our new zone over the connection.

7. GSFC would add the UMIACS SAN Router's IP and mSAN identifier to the access control list for the iFCP connections on our storage router, and we would specify that we will share our new zone over the connection.

This is what we have done in the past, and it is easy to see how error prone this process can be between multiple sites with so much coordination of administrative information by phone and email. We feel that the limitations of these current management systems present a serious barrier to developing large-scale inter-site SANs.

3.4 Sharing LUNS between UMD and GSFC using the Management System.

Our management system simplifies the setup process significantly because it provides a secure registry for all of the requisite administrative information including IP addresses, mSAN identifiers, and Zone identifier for the participating SAN routers. Administrators can browse the available remote resources to support manual configurations of existing resources and be assured that they are choosing settings that will not conflict with other sites. Scripts can access the administrative information program-matically to support automatic configuration of new systems. In fact, new zones and SAN routers that are installed into the framework require even less coordination, because they can register unique identifiers when they are first installed.

The management system in our test bed will also demonstrate dynamic reconfiguration of the shared SAN. In the manual configuration described above, sharing LUNs between UMIACS and GSFC requires separate administrative intervention on both sides of the connection to share and unshare a LUN, and, unfortunately, both administrators end up running the same commands over and over again on their respective SAN Routers. Scripting these repetitive commands and making them available to trusted users through the Invocation Web Service enables users to reconfigure the inter-Site SAN and to access remote SAN devices as needed without any administrative intervention. We are excited about the possibilities for this type of dynamic inter-site storage area network.

4. Summary

Our pilot system shows some ways in which widely used web and Grid Computing technologies can be successfully employed to support the complex security issues and trust relationships that arise in inter-site SAN installations.

5. Acknowledgements

We would like to acknowledge the many engineers at GSFC and UMIACS who have supported this work, especially Hoot Thompson, Bill Fink, Paul Lang, Mike Van Opstal, Gary Jackson, Steve Willet, and Mike McGann.

6. References:

[1] T. Clark, "Designing Storage Area Networks: A Practical Reference for Implementing Fibre Channel and IP SANs." Addison Wesley Professional, March 2003

[2] H. Yang, "Fibre Channel and IP SAN Integration." MSST 2005: 101-113

[3] Phil Andrews, Tom Sherwin, Bryan Banister, "A Centralized Data Access Model for Grid Computing". MSST 2003: 280-289

[4] Hoot Thompson, Curt Tilmes, Robert Cavey, Bill Fink, Paul Lang, Ben Kobler: “Considerations and Performance Evaluations of Shared Storage Area Networks at NASA Goddard Space Flight Center.” MSST 2003: 135-145

[5] Phil Andrews, Bryan Banister, Patricia A. Kovatch, Chris Jordan, Roger L. Haskin: Scaling a Global File System to the Greatest Possible Extent, Performance, Capacity, and Number of Users. MSST 2005: 109-117

[6] Phil Andrews, Patricia A. Kovatch, Chris Jordan: Massive High-Performance Global File Systems for Grid computing. SC 20005: 53

[7] The Apache Tomcat Server, http://tomcat.apache.org/.

[8]The Apache Axis Project, http://ws.apache.org/axis/

[9] The Apache WSS4J Project, http://ws.apache.org/wss4j/

[10] A. Nadalin, C. Kaler, P. Hallam-Baker, and R. Monzillo. *Web Services Security: SOAP Message Security*, Oasis Standard 200401, March 2004.

[11] Joseph JaJa, Mike Smorul, Fritz McCall, Yang Wang: Scalable, Reliable Marshalling and Organization of Distributed Large Scale Data Onto Enterprise Storage Environments. MSST 2005: 197-201

[12] The MySQL Database, http://www.mysql.com/

[13] Libes, D., "Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Applications", O'Reilly & Associates, January 1995.

[14] Libes, D., "Using Expect to Automate System Administration Tasks," Proceedings of the Fourth USENIX Large Installation Systems Administration (LISA), Colorado Springs, CO, October 17-19, 1990.

[15] The Condor Project, http://www.cs.wisc.edu/condor/