Early Experiences in Managing Inter-Site Storage Area Networks Using Secure Web Services

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

Fritz McCall Mike Van Opstal
University of Maryland Institute for Advanced Computer Studies
{fmccall,mvanopst}@umiacs.umd.edu

Hoot Thompson Kirk Hunter
Patuxent Technology Partners
{hoot,khunter}@ptpnow.com

Abstract

The NASA Goddard Space Flight Center and the University of Maryland Institute for Advanced Computer Studies have deployed a pilot system for managing distributed IP-Based Storage Area Networks of dynamically allocated SAN extensions in the Advanced Virtual Engine Test Cell (AVETEC) Data Intensive Computing Environment (DICE).

The system implements the framework for managing inter-site storage area networks using Grid and Web Services technologies presented at MSST06. It includes several new components including a basic registry and a java-based command line interface. The test-bed provides some early experiences with the reliability, usability, and security of the system, as well as some performance testing of the web services.

1. Introduction

At MSST06, we reported on the challenges that our administrators experienced while managing IP-based storage area networks that spanned geographically distributed and administratively independent sites including the NASA Goddard Space Flight Center, the University of Maryland Institute for Advanced Computer Studies, and the Gilmore Creek Alaska Ground Station Facility [1, 2]. We found that the cost of manually establishing a SAN extension was very high because it required a lot of coordination between experts at all of the participating sites. In order to simplify the operation of these inter-site storage area networks, we proposed a management framework that uses grid technologies to securely delegate administrative tasks to trusted users at the various sites. To demonstrate the feasibility of our approach, we presented a prototype system architecture built on Apache Tomcat [3] and secure web services [4] built on Apache Axis [5] and WSS4J [6].

In the past year, we continued to develop our prototype of the management framework with the primary goal of allowing users to dynamically establish SAN extensions on an as-needed basis and to remove them when they are no longer needed. A secondary goal of our prototype was to minimize the technical expertise required to establish a SAN extension to a needed resource. In order to achieve these goals, we developed two new software components:

1. A registry of information about the participating sites, shared devices, and shared zones that make up an Inter-Site Storage Area Network.
2. A Java-based command-line interface for dynamically managing iFCP connections between users and the devices that they need to access.

We installed these new components along with updated versions of the prototype software in the AVETEC DICE test-bed [7] at the NASA Goddard Space Flight Center where we evaluated their reliability, performance, usability, and security under a variety of network conditions.

Our previous paper at MSST06 described the Management Framework's overall architecture, discussing its capabilities, security mechanisms, authentication scheme, and authorization system. The design and implementation of SAN Extension technologies is well described in technical publications [8, 9], and papers at previous MSST conferences have described the performance of SAN extension technologies over local, metropolitan, and wide area networks using a variety of clustered file systems [2, 10, 11]. Rather than revisit these issues, this paper describes our early experiences with the management framework in the AVETEC DICE test-bed, and presents the new features that we have developed in the last year.

2. Design and Implementation of the Pilot System

2.1 Overview of the Management Framework

For our evaluations in the DICE test-bed, we developed a new java-based command-line user interface that queries the registry of inter-site SAN resources, and reconfigures the SAN routers through the management framework's Invocation Service, implemented as a secure web service acting as a proxy for calls to SAN Router Control Software. Figure 1. shows the interaction of the various software components.

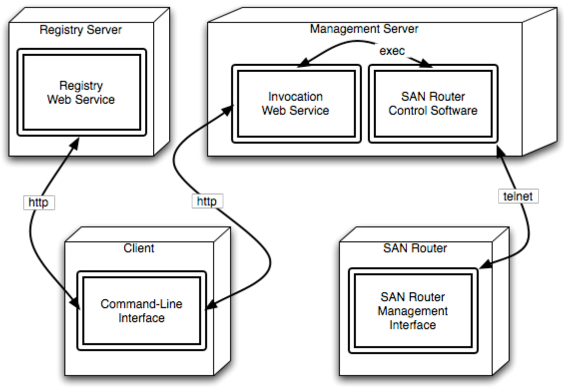


Figure 1. Software Overview

In this configuration, users can find and access the resources that they need without administrative intervention. The command line interface is installed and run without local administrative privilege on the client system. However, some administrative access may be needed to mount the file system depending on the operating system, application, and file system that will make use of the SAN extension.

2.2 Implementing the Registry Web Service

The registry is designed and implemented as a web service that helps users find and access the resources that they need through a variety of lookups based on high-level information such as descriptions, identifiers, or addresses for devices, sites, and zones. It also provides an administrative interface for registering information about shared resources. At a design level, it supports a data model that describes participating sites, shared devices, and zones. Figure 2. Illustrates the current data model for the registry.

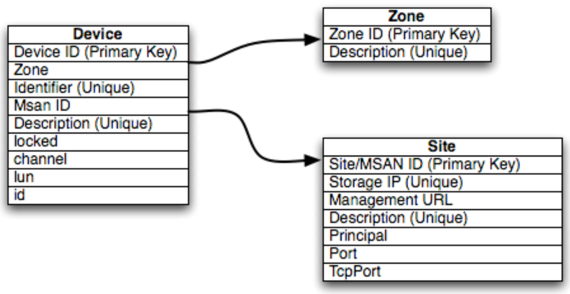


Figure 2. Registry Data Model

The registry is useful for finding resources within the Inter-Site SAN. We expect users to know only a short description of the resource that they need. For example, they may know that they want to access a disk named DICE-XRAID, or they may know that their application needs to connect to the GSFC-DICE zone. However, they do not need to know the larger set of low-level SAN extension connection parameters, because the registry will provide them.

At an implementation level, the registry web service provides remote methods built on Apache Axis. It also defines several complex types that map objects from the Java programming environment into the XML messaging format used by the Simple Object Access Protocol (SOAP). For example, the web service provides methods to find resources by unique Device Identifiers, Device descriptions, Site identifiers, and Site descriptions.

Most of the registry's methods return a connection object that contains the low-level information needed to create the connection. Every connection object contains information about the management framework, such as the IP address and TCP port of the remote site's management server. It also carries protocol-specific connection information that specifies information about a remote storage router's IP address, its TCP port, fiber channel port specifications, and SAN zoning information.

Locking is an important issue for any system that shares storage devices, but this is not the focus of our software. Most sites employ clustered or SAN file systems, like GPFS or SNFS, to manage locks when multiple hosts access the same shared device concurrently. Previous papers presented at MSST have demonstrated that these file systems and their locking mechanisms have been very successfully employed over Wide-Area Networks. For example, researchers at the San Diego Super Computer Center have demonstrated that GPFS file systems can deliver very high performance over the Teragrid's transcontinental high-speed network [12]. Since file locking is often efficiently handled at the file system layer, our software only provides a simple exclusive locking mechanism by which a client or an administrator can request exclusive access to a shared device.

The registry is stored in a relational database management system that enforces constraints on the stored data in order to:

1. Ensure unique identifier assignments for zones, SAN identifiers, and devices through the use of primary and secondary database keys.
2. Prevent conflicts for device and site addresses where multiple listings would not be valid.
3. Guarantee that consistent data is returned even as multiple consumers run queries and updates through the use of database transactions.
4. Enforce the validity of data relationships through the use of foreign keys.

We currently use the Mysql Database version 5.0.27 to implement the underlying data store.

2.3 Developing Router Control Software

Our router control software extends the prototype scripts for managing the McData Eclipse SAN Routers that we presented at MSST06. It uses perl-expect [13, 14] to add, remove, and list iFCP connections on McData Eclipse SAN Routers through their command line interface using the telnet protocol. It also provides public status and configuration information in a similar fashion. It currently supports several firmware revisions of the Eclipse models 1620 and 2460 as well their predecessors, the Nishan Systems Model 3300.

The control software is a local program on the management server that is called by the Invocation Web Service on behalf of authenticated remote users. Its only role is to safely transmit commands and parameters to the SAN routers. In this context, “safety” requires:

1. Protection from malicious and accidental parameter injection attacks.
2. Assertions that we are interacting with the correct SAN router based on the expected system name and prompt.
3. Determination of an expected and compatible firmware level
4. Cleaning control characters from the router’s output so that the output is compatible with the XML messaging system that will return output and error streams to the remote user.

These checks help to ensure that the configuration commands have only their desired effects, and that output and error streams are accurately returned to the remote user.

2.4 Client Interface

We developed a java-based command line interface named flexzone that interacts with the registry and the invocation service in order to find and inter-connect shared SAN devices by dynamically creating the necessary SAN extensions. For example, a user can request a connection between the GSFC-DICE site and the DICE-Xraid disk with:

```
`flexzone –registry –device-description DICE-Xraid –site DICE1 `
```

This command encapsulates the more detailed parameters that are needed to setup the SAN extension. It is independent of the underlying SAN extension technologies, and it hides the SAN topology information that is needed to make the connections.

Depending on the information retrieved from the registry, this command may actually require two independent calls to the invocation service on two different management servers. For example, it could call:

```
InvocationWS https://172.16.230.206 invoke Flexzone-rtr.pl --addifcp --port 8 --zone 3 --ifcpip 10.8.1.50;
```

```
InvocationWS https://172.16.240.207 invoke Flexzone-rtr.pl --addifcp --port 7 --zone 3 --ifcpip 10.8.0.40;
```

The first call would set up the consumer side of the connection while the second would set up the device side of the connection. Figure 3. illustrates how the various components interact in this process.

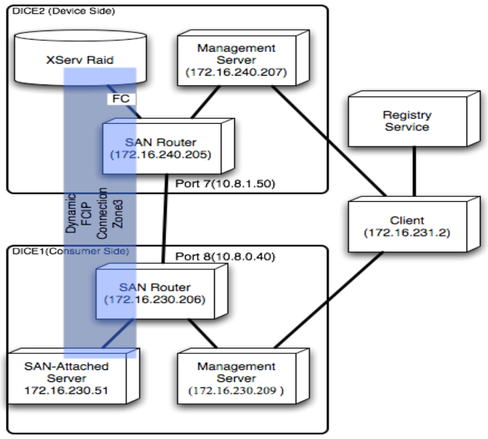

DICE2 (Device Side)
XServ Raid
Management Server (172.16.240.207)
FC
SAN Router (172.16.240.205)
Registry Service
Port 7(10.8.1.50)
Dynamic FCIP Connection Zone3
Client (172.16.231.2)
DICE1(Consumer Side)
Port 8(10.8.0.40)
SAN Router (172.16.230.206)
SAN-Attached Server 172.16.230.51
Management Server (172.16.230.209)

Figure 3. A Sample Inter-site SAN

3. Tests in the AVETEC DICE Test-bed

3.1 Overview of the AVETEC DICE Test-bed

Our testing environment in the DICE test-bed includes two logically separate sites. One site, DICE1, is configured as a consumer with a dual Intel Xeon processor Dell 2850 acting as a SAN attached server that will access the shared disk resources. It connects to a Mcdata Eclipse 1620 SAN Router through a McData Sphereon 4700 Switch. The other site, DICE2, is configured to share disk space from an Apple Xserve RAID through a McData Eclipse 2640 Router. It includes a Dell 6300 with four Intel Pentium 2 processors configured as a management server to control setup and tear down of connections. Figure 4. illustrates the installation of our hardware in the DICE test-bed.

We connected these sites through Gigabit Ethernet on the DICE local area network (LAN). We also configured private non-routed networks for control of the McData Eclipse Routers' management ports. We simulate various wide area network conditions by passing the DICE LAN connections through a Dell 2850 with dual Intel Xeon processors running the Nist Net network emulation package (NistNet) [15].

Even though the management framework can control both sides of a dynamic connection (as shown in the previous client interface example section), our test environment provides just a single management server to dynamically control the device side of the connections. This simplified our test measurements and proved to be adequate and efficient for our performance tests.

3.2 Basic Evaluations in the AVETEC DICE Test-bed

The DICE test-bed provides several advantages for basic evaluations of the management framework's functionality. It is collaboratively supported by a group of systems, storage and network administrators who provide installation and management services for all of its various hardware and software systems. They play an important role in evaluating the management framework and they provide valuable feedback regarding its utility and usability.

A secondary goal of our evaluations in the DICE test-bed is to determine whether a system is sufficiently secure for integration into a production environment. As a first step toward this evaluation, our system was tested with a network security port scanner before it could be connected to the main test network. Using a McAfee Foundstone scanner, the security team has tracked vulnerabilities and potential risks in past versions of the Apache Tomcat code base. This interaction has been widely beneficial, both in better securing our software and in increasing familiarity with Java-based web applications and the Apache Tomcat Server among DICE participants.

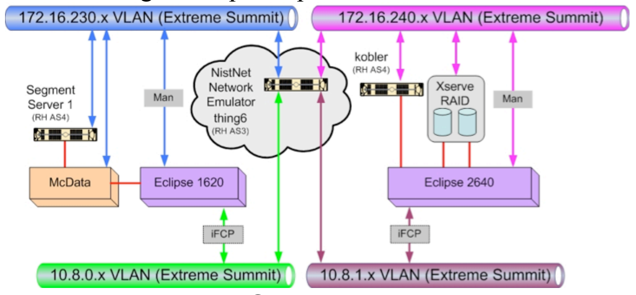

172.16.230.x VLAN (Extreme Summit)
172.16.240.x VLAN (Extreme Summit)
Segment Server 1 (RH AS4)
Man
NistNet Network Emulator thing6 (RH AS3)
kobler (RH AS4)
Xserve RAID
Man
McData
Eclipse 1620
Eclipse 2640
iFCP
iFCP
10.8.0.x VLAN (Extreme Summit)
10.8.1.x VLAN (Extreme Summit)

Figure 4. DICE Testing Environment

3.3 Usage and Test Cases

For our performance and reliability tests, we tested the ability to setup and tear down an iFCP-based SAN extension in order to access an ext3 file system on the Apple Xserve RAID using the following commands:

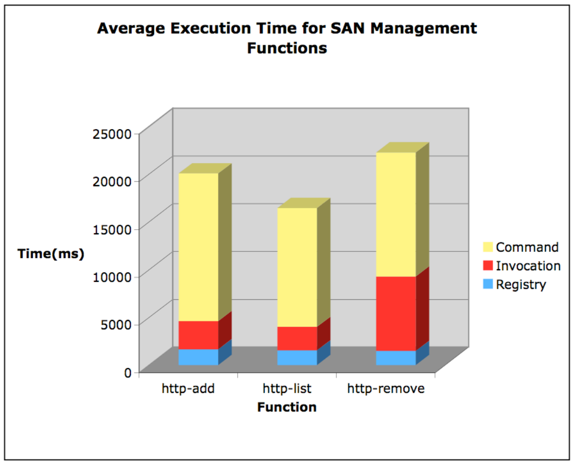
```
Flexzone –add –site dice1 –device-description
 Xraid0

Flexzone –list –site dice1

Flexzone –remove –site dice1 –device-description
 Xraid0
```

In order to better understand the cost of using Axis-based web services to manage SAN extensions, we instrumented our code in order to time the various steps involved in the process, including queries against the registry, requests to the invocation service, and calls to the router control software. We tested our software's performance under a variety of network conditions including high latency and high-loss using NIST Net .

It is important to note that our measurements do not include the time required for the operating system to mount the file system, because this depends heavily on the application, operating system, and file system. These issues are beyond the scope of this paper. Instead, we focused on the performance of the control connections needed to establish the SAN extension.

3.4 Overhead in Establishing SAN Extensions

We measured the overall time to setup and tear down an iFCP SAN Extension using the Hyper-Text Transfer Protocol (http). Graph 1. shows the average run time of our three test commands on a local area network without any measured delay or packet loss.

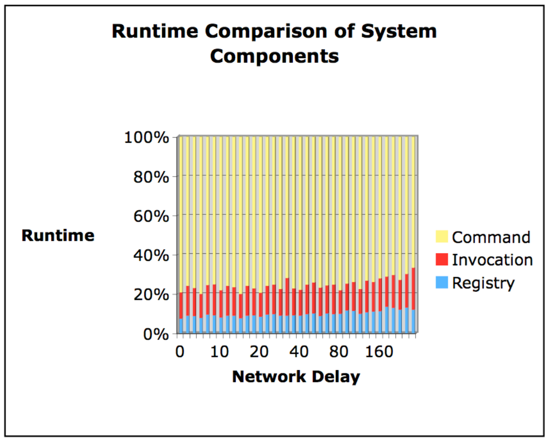


Graph 1. Aggregate Execution Times

Our tests show that we can typically reconfigure a SAN router in less than 25 seconds with requests to setup SAN extensions taking slightly less time than requests to tear them down. The router control software takes a majority of the setup time, so we consider that in the next section.

3.5 Cost of Reconfiguring SAN Routers

Graph 1. shows that reconfiguring the routers typically takes the most time of all our operations when we run on a local area network. We found that this holds true even as network latencies increase the runtime of our web services. Graph 2. illustrates the percentage of time spent in the Registry Service, the Invocation Service, and the SAN Router Control Software under varying network latencies.

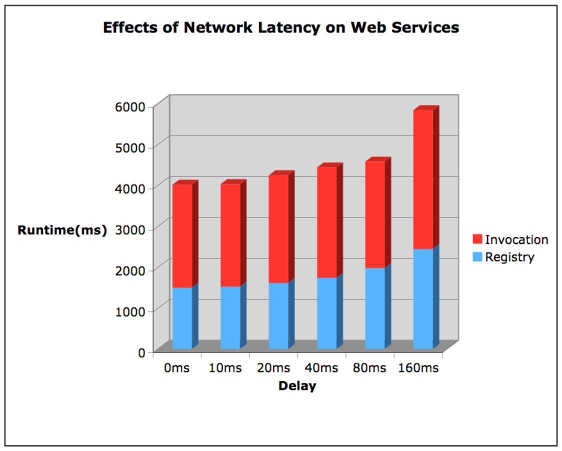


Graph 2. Software Runtime Comparison

While investigating the large command execution runtime, we found that the McData Eclipse SAN Routers take between 5 and 12 seconds to start a management session using the telnet protocol, which presents a relatively large overhead on our average operation, which is typically less than 25 seconds. However, no other device in the DICE test-bed showed similar delays, so our conclusion is that these delays are unique to the McData Eclipse SAN Routers.

3.6 Effects of Network Latency on Performance

We expect the management framework to operate over wide area networks that may present high latencies or high loss due to distance or congestion. In order to evaluate our system in these environments, we tested its reliability and performance under various network conditions using NISTnet. Increasing latency over a range of values between zero and one hundred and sixty milliseconds had no impact on the reliability of our software. Graph 3. shows the effects of network latency on the runtime performance of our web services. As expected, higher latencies increase the runtime of our web services, but only at a moderate and tolerable rate.

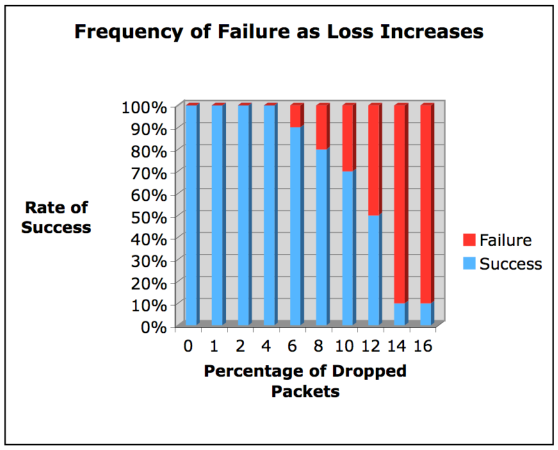


Graph 3. Effects of Network Latency on Runtime

Our conclusion is that large increases in network latency have a relatively small impact on the overall runtime of our system. This suggests that our approach will be suitable for most production wide-area networks.

3.7 Effects of Network Packet Loss on Reliability

We also tested the functionality of our client under network conditions with increasing loss. Although no failures were observed under high network latencies, high rates of network packet loss caused calls to the management framework to fail. Graph 4. shows the increasing rate of failed calls to the management framework as the network becomes less reliable.



Graph 4. Effect of Packet Loss on Reliability

We expect calls to the management framework to fail as the underlying TCP IP network becomes unreliable, but we feel that there is some room for improvement in this area. Our analysis of the failures shows that the failures resulted from socket errors, where reads or writes from the remote host could not be completed in a timely fashion. We are currently investigating ways of increasing the socket timeouts and adding additional retries to improve the reliability of our software on networks that drop between eight and twelve percent of network packets.

4. Future Work

We plan to continue our practical evaluations of the management framework by controlling SAN extensions in order to support GPFS and Ext3 file systems in the DICE test-bed. We also plan to begin testing integration with the Internet Small Computer Systems Interface (iSCSI) and the UDP-Based Data Transfer Protocol (UDT) [16].

Upon completion, we plan to make the software available to the public, and to identify collaborators to help test it with real-world applications.

5. References

[1] Ben Kobler, Fritz McCall, Mike Smorul: “A Framework for Managing Inter-Site Storage Area Networks using Grid Technologies.” MSST 2006: 15-18
[2] 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
[3] The Apache Tomcat Server, http://tomcat.apache.org/.
[4] A. Nadalin, C. Kaler, P. Hallam-Baker, and R. Monzillo. *Web Services Security: SOAP Message Security*, Oasis Standard 200401, March 2004.
[5] The Apache Axis Project, http://ws.apache.org/axis/
[6] The Apache WSS4J Project, http://ws.apache.org/wss4j/
[7] The Advanced Virtual Engine Test Cell Data Intensive Computing Environment, http://www.avetec.org/DICE/
[8] T. Clark, “Designing Storage Area Networks: A Practical Reference for Implementing Fibre Channel and IP SANs.” Addison Wesley Professional, March 2003
[9] T. Clark, “IP SANs: A Guide to iSCSI, iFCP, and FCIP Protocols for Storage Area Networks”, Addison Wesley Professional, December 2001
[10] 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
[11] Phil Andrews, Patricia A. Kovatch, Chris Jordan: Massive High-Performance Global File Systems for Grid computing. SC 2005: 53
[12] Phil Andrews, Chris Jordan, Hermann Lederer : “Design, Implementation, and Production Experiences of a Global Storage Grid” MSST06
[13] Perl Expect Module
http://sourceforge.net/projects/expectperl/
[14] Libes, D., "Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Applications", O'Reilly & Associates, January 1995.
[15] M. Carson, D. Santay, “NIST Net: a Linux-based network emulation tool” ACM SIGCOMM Computer Communication Review Volume 33: 111-12
[16]UDP-Based Data Transfer Protocol,
http://udt.sourceforge.net/