Data Replication in Grid Computing

Russ Wakefield

1 Abstract

One of the fastest emerging technologies within the high performance computing environment is grid-based technologies. Achieving the performance requirements for these applications requires they be distributed among hundreds, sometimes thousands of nodes. Often the resource allocation of these nodes is dynamic, forcing the data required by these applications to be distributed dynamically as well. These needs have led to the concept of a *data grid* – grid systems focused on data. *Data replication* is the process of using *replicas* scattered to the various nodes to improve performance within grid applications as well as provide a level of fault tolerance with respect to that data. This paper compares and contrasts three systems that provide data replication capabilities within the grid environment.

2 Overview

2.1 Introduction

Grid computing is defined as coordinated resource sharing and problem solving within a dynamic, multi-institutional, virtual organization [2]. Grid computing theory indicates that resources that are unused within an environment could be added to the greater good. The promise of massively parallel systems has been with us for years – but the implementation of this promise is just beginning to evolve within the grid computing environment.

The virtual organizations defined in the keystone paper by Foster et al. [2] provide the coordination necessary to make the geographically-diverse heterogeneous resources owned by a multitude of institutions available to the members of the organization. This coordination involves the use of middleware packages to ensure the resources identified by the members of the virtual organization are made available in a manner that is timely as well as safely and securely.

Data grids provide the tools and infrastructure necessary to run distributed data-intensive applications that typically use large datasets stored using distributed storage [5]. The characteristics necessary in this environment include:

- The ability to identify which datasets on the data grid are available for their use.
- The ability to search through the available datasets to find the required dataset and the resources necessary to access that dataset,
- The ability to transfer these large-sized datasets in a timely manner to be available when the grid application is scheduled on the virtual organization's resources.
- The ability to schedule the resources within the virtual organization to consume the data resources.
- The ability to ensure the correct security elements – such as permissions - are associated with the appropriate datasets and cannot be subverted.

The evolution of requirements such as these leads us to the development of middleware packages that address these needs. This paper compares/contrasts three of the top sets of systems at use in industry today. The Globus consortium is associated with Ian Foster, one of the pioneers of grid computing and widely respected as a leader in the field. This consortium has produced the Globus toolkit which includes GridFTP and RLS as the components to address replica management. The San Diego Supercomputer Center has created a package called SRB (storage resource broker). The final system for compare / contrast is a middleware package called GFarm from a group headed by Osamu Tatebe – an associate professor at the University of Tsukuba and the

current head of the Open Grid Forum committee on data replication. The use of compare and contrast is intended to better illustrate the requirements of the data grid community, the state of the existing solutions, and the areas for continued research.

2.2 The Grid Environment

The term grid computing was coined in the late 1990s to describe a set of resources distributed over wide-area networks that can support large-scale distributed applications [2]. The similarities between a vision of readily available computer resources and the existing power grid boosted the analogy – the desire to be able to "plug" into the computational grid and have access to computation and data is likened to plugging an electrical device into an outlet and drawing power to operate it. The lure of being able to utilize these easily accessible resources has provided the focus for a tremendous amount of research and development. As more and more applications begin using the grid environments, data about the running the applications within that environment becomes available and the requirements to provide tools to aid in the use of the grid environments solidify and become clearer.

One of the biggest issues surrounding the grid environment is the complexity of the various components. There are potentially thousands of services, hardware components, nodes, software components, and administrative domains involved in the execution of an application within the grid. As with its predecessors in traditional computing, this has led to the development of all-encompassing packages to add layers of abstraction to reduce the complexity. These packages include the Globus toolkit [6] and the Parallel Virtual Machine [7]. The Globus toolkit contains one set of the packages used as a part of the compare/contrast section of this paper. PVM is still widely used – in fact there was a call for papers on their website among their users – but is a package less in the mode used in data grids and more in the mode of parallelizing algorithms and using message-based systems to solve computation-intensive scientific problems. These type of problems are pervasive in traditional high performance computing arenas – but not germane to the topic of this paper.

As I mentioned in the introduction, a keystone paper defining issues associated with the grid was [2]. *An Anatomy of the Grid* [Foster et al.] added the concept of a virtual organization to the definition of grid computing – defining the VO as a set of participants with various relationships that wish to share resources to perform a task. Resources are defined to include data, computers, scientific instruments, software, etc. The virtual organization is key to achieving quality of service, within this social framework occurs the Service Level Agreements (SLAs) necessary to commit these resources. According to *Anatomy*, "VOs enable disparate groups of organizations and/or individuals to share resources in a controlled manner, so that members may collaborate to achieve a shared goal". These virtual organizations have become a de facto standard used in discussing the grid and working through the social issues associated with using multiple owners of multiple resources each with its own set of policies.

Just like Internet protocols have layers enabling TCP and HTTP to use the common IP layer, providing a layered approach on the grid enables each of the layers to support different architectures and behaviors above it through the use of Applications Programming Interfaces (APIs). An open architecture enables many development approaches, the establishment of the common protocols and interfaces is essential to the success of implementing abstraction to work within the grid. A typical layered Grid architecture and how it relates to the similar layered Internet protocol looks like this [2] [8]:

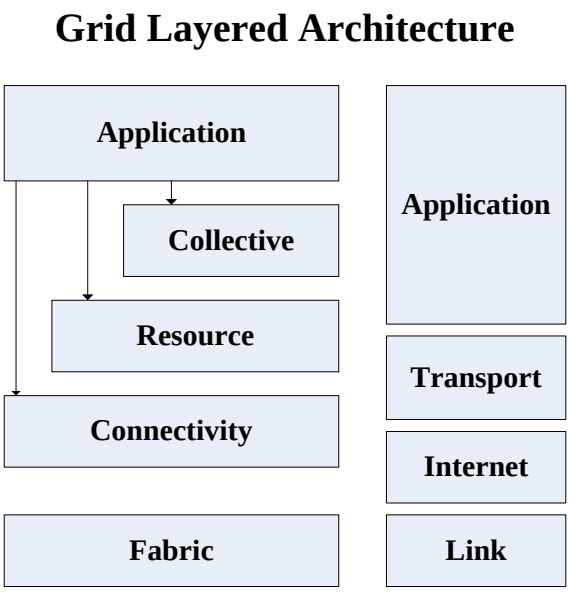

Figure 1: Comparison of Grid and Internet Protocol Layers

The fabric layer provides the nodes, services, software entities, etc. that are available for the applications to use and are mediated as a part of the grid protocol. The fabric components are responsible for implementing the operations the users of the resource need to accomplish their tasks. The connectivity layer is for communication and authentication in grid-specific transactions. Characteristics of authentication include single sign-on, delegation, and interoperability with resource-specific security, and cooperation within the virtual organization. The resource layer facilitates information gathering and control functionality at the level of individual resources; it is strictly concerned at the single resource level. The collective layer provides the communication between the resources to provide interaction between them, such as a directory service or a data replication service. On top of this stack is the application layer made up of the applications that participants within the virtual organization want to run within the grid.

My focus will be on satisfying the needs of the resource and collective layers – providing the resources necessary to satisfy data discovery and replication both as a single unit as well as resolving the issues to the individual resource level.

2.3 Data Grids

Data grids can be defined as data management systems that ties together storage systems distributed across administrative domains such as the virtual organizations defined by Foster et al. [9]. These data management systems are responsible for the state information created when the various entities register their resources within the virtual organization, creating a logical name space that denotes their membership in the organization. These storage systems can take many forms, including archival systems, caches, and databases. The act of registration creates a set of global persistent identifiers to identify the associated resource, as well as set of tools to access the storage in a uniform manner. In this manner, the storage resource can be treated much like an object in object-oriented programming in which the data being stored is within the object, and the methods provided in the interface allow access to that data.

While providing uniform access to the data stored within the individual systems is a requirement, another layer of abstraction is necessary for this data to be utilized. This layer of abstraction hides the infrastructure dependencies and provides a common interface within the data grid for discovering, accessing, and modifying the data. This interface also eliminates the need to understand the specifics of the mechanisms necessary for each individual storage system.

The logical name space created when the various members of the virtual organization register their storage components allows the use of a persistent global identifier when using those storage systems. These name spaces can be organized however the members of the virtual organization choose – the most logical being a hierarchical mechanism based on locality or an organization inherent to the data. This organization allows discovery and access methods that are more advanced than the individual access method provided by the resource component registering the data.

2.4 Definitions

The requirements for these data grids – and their associated interfaces - were laid out by Vengugopal et al.[5] and outlined here in the introduction in Section 2.1. These boil down to a minimum requirement of providing a timely and reliable high-performance data transfer mechanism and providing a management system that allows replica discovery and access. A *replica* is defined as a copy of a dataset created by a local user in differing geographic locations. This local copy is created to reduce the latency required to access the data – and may contain whole or partial copies of the original dataset depending on the application's needs. The *replica management system* allows the users to discover, create, register, and update the information contained within the replica. The rules by which the replica management system operates are called the *replication strategies*, which are based on the needs of the replicas, the need for locality of data, etc., basically the requirements for the management system and the use of the specific data. Information about the datasets is called *metadata* and is contained in the *replica catalog* along with information about the various members of the virtual organization and their usage of the replicas.

3 Data Discovery

3.1 Definition

Data discovery is defined as the act of providing information about the resources within a node to the potential users within the virtual organization. This could be as simple as providing the clock speed of the processors within a node to information about what datasets contain which data items when requesting a replica set. An important trait with data discovery is to keep the overhead for such an operation to a minimum – this is by definition a network intensive operation, and something to be minimized as much as possible.

3.2 Implementation

One of the key challenges with respect to data replication is the necessity to implement a robust and timely data discovery mechanism. As in most database technologies, the early mechanisms such as the Condor matchmaker [10] and the early implementation of the Globus Toolkit's Monitoring and Discovery Service (MDS) [11] – the technology revolved around a centralized server that provided all the information necessary about the replicas that existed within the data grid. While easy to implement and manage – a centralized server implementation led to a bottleneck within the grid as well as providing a single source of failure.

Recognizing these initial implementations had the aforementioned issues, the next generation was focused more on creating index servers providing a directory service to the members of the virtual organization. These index servers communicate using registration protocols that provide the information about the resources as well as understanding about each other and keeping the information contained within their particular repository available for the other index servers to access and use to update their own information if necessary. Users within the virtual organization can then query these index servers to obtain the location information necessary to accomplish their application. This type of implementation eliminated the single point of failure, at the cost of a set of index servers keeping themselves up to date.

Current research is looking at how data discovery can be accomplished using the concepts created in peer-to-peer networks. Since every participant in the virtual organization would be in control of the information contained about that participant – each would be responsible for maintaining a peer that provides that information to the rest of the participants within the virtual organization. This allows a much more dynamic structure where peers join and leave the structure on a regular basis. This research would also allow the current investigations in peer-to-peer networks to be directly applicable within the grid environment.

3.3

Discovering Data Items

Taking the implementation to the next step – how can the research in peer-to-peer network help in discovering data items in a grid environment? Imagine that given a logical identifier, the idea is to find the "best" physical entity matching that logical identifier based on some criteria. According to Casanova [2], the research in peer-to-peer networks associated with reducing the latency for such a query could be directly applicable within the grid environment. If the research improves the efficiency of that query, it could be used to locate the best replica within the virtual organization.

This is one example of something I have found throughout my research into the grid environment – that while the grid has unique characteristics, most of the issues in the grid environment can be directly associated to similar problems being addressed in networking, operating systems, and other disciplines within the field. I find myself wanting to learn more and more disciplines to understand their nuances and how they apply within the grid.

4 Data Replication

4.1 Introduction to Data Replication

In the definitions, *replica* was defined as a copy of a dataset contained within the virtual organization created by the user to provide greater locality of data – and therefore better performance when using that data. Fields such as high-performance computing create large datasets as a part of their application. As with any distributed system that contains multiple copies of the same data, the interactions with these datasets must be tightly managed to allow correctness to be maintained. As was mentioned in section 2.4, the minimum requirements for a data replication system include a timely and reliable high-performance transport system and a replica management system that identifies and maintains replicas within the system.

4.2

Replica Management system

A replica management is defined as a system that can create, identify, manage, and update replicas within the virtual organization. The components of a replica management system include a replication locater service, a metadata service providing information about the replicas, a data management component that handles versioning information, master copy management, and workflow management.

In addition, these replica management systems are beginning to use data replication strategies to guide their ability to meet the needs of their users. Below is an introduction to two of the recent proposals for these strategies.

4.3 Data Replication Strategies

The role of a data replication strategy is to identify when a replica should be created, identify which files should be replicated, and identify where should the replicas be placed [4] to enhance the performance of the distributed system. Replication promotes higher data availability, increased performance and lower bandwidth consumption through locality, and increased fault tolerance. The goal of manipulating the parameters above is to maximize these traits.

Both Ranganathan et al. [4] (2001) and Lamehamedi et al. [3] (2002) described dynamic strategies to minimize the overhead associated with the actions. The mechanisms introduced by Ranganathan have been incorporated into the Globus toolkit – probably the most widely used middleware package in use today. Before this time, the strategies defined were under the control of the user and often were overkill for their actual needs.

The strategies introduced by Ranganathan include concepts include best client, cascading replication, caching, caching plus cascading replication, and fast spread. Best client records the requests for each file and stores the file on the node with the highest request rate. Cascading replication is similar to best client by storing replicas on high usage sites, with

thresholds identified for the file moving it farther down the tree of nodes as the thresholds are reached. Caching is done at the user's request – but the local storage can only hold a few of these files at a time. Caching plus cascading replication combines these two strategies, allowing the client to store the file locally and periodically identifying high usage files to propagate downward. Fast spread stores a copy of each file as it propagates downward through the nodes.

The approach used by Lamehamedi et al. is to organize the data using alternative network topologies. Two different topologies, a fat tree and a ring were used based on the topology of the nodes. Each entity in the data grid maintains a replica set, initialized to empty. Each node also maintains an index list of the entities it is responsible for as well as the entities it is aware of and builds a replica connection graph using this information. The request for a file is then satisfied by the node closest to it based on the topology used, the runtime system associated with the model is responsible for updating the replica connection graph to allow this identification.

Both systems in their conclusions compared their results to the naïve approach in use before then. The additions to the Globus toolkit showed the caching plus cascading replication showed the best performance – which intuitively to me made sense. Cache what you can, move what you can't closer to the highest users. The network topology based system also showed better gains over the naïve approach – but had significant work to do to get it into a usable system.

The point of introducing these strategies is not so much to compare and contrast the two approaches as to illustrate the state of the current research in these areas. Both of these approaches are reusing research done previously in operating system and network disciplines to enhance the usability of these systems.

5

Introduction of Existing Systems

As previously mentioned, I chose three replica management systems to compare and contrast – RLS (Replica Locator Service) and GridFTP in use by the Globus toolkit, Storage Resource Broker (SRB) – which was created and is maintained by the San Diego Supercomputer Center and GFarm from a team headed by Tatebe.

5.1 Globus Toolkit

The Globus consortium identifies itself as a non-profit organization formed by global computing leaders who support the Globus Toolkit, the de facto standard for open source grid computing infrastructure [13]. The Globus Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability [6].

The Globus toolkit has several components to accomplish its replica management system. RLS is the component to address replica location and manage the replicas, GridFTP is the transport mechanism to relocate files.

5.1.1 Replica Locator Service

A replica location service such as those defined by Chervenak et al. [11][12] has several requirements – and one key restriction. I'll get into the requirements shortly, but wanted to cover the restriction first. Their definition of a replica is that they are read-only copies of the dataset. Updating the dataset is done by creating a new dataset and checking it in rather than using a more distributed database approach to update the records within the dataset. I viewed this as a limitation of the system - and hope that my continued research into this field will show me the rationale for implementing it this way.

Having said that – we know that the bulk of transactions in a distributed system – especially that of a compute intensive environment such as is currently being used within the grid – most of the transactions are read-only. The main requirement for a replication location service is

given a unique logical identifier; locate the physical locations of one or more copies of this dataset [11]. In addition, the replica location service maintains and provides access to the information about the physical locations of the copies. The RLS must scale to at least several hundred replica sites and must scale to provide good performance with respect to queries and updates. An RLS should provide security by protecting the privacy and integrity of the data maintained by the virtual organization. Finally a RLS should not introduce a single point of failure.

The Replica Locater Service (RLS) component of the Globus toolkit is built on an architectural framework called Giggle (GIGa-scale Global Location Engine) [11]. This framework is structured with 5 basic mechanisms:

- Independent local state maintained in the Local Replica Catalogs (LRCs)
- Unreliable collective state maintained in the Replica Location Indices (RLIs)
- Soft state maintenance of RLI state
- Compression of state updates
- Membership and partitioning information maintenance

The Local Replica Catalog maintains information about replicas at a single replica site, including a mapping between logical file names and physical file names for those files located on its storage systems. It is also responsible for queries about that information, support authentication when satisfying requests, and propagate state information about itself to other catalogs.

The Replica Index is maintained to provide support user queries about multiple replica sites. These indices contain sets of {local file name, local replica catalog} entries. A soft state – information that times out periodically - is kept about the LRCs in the RLI. This state information is compressed for transmission and allows the RLI to be reconstructed easily after failures.

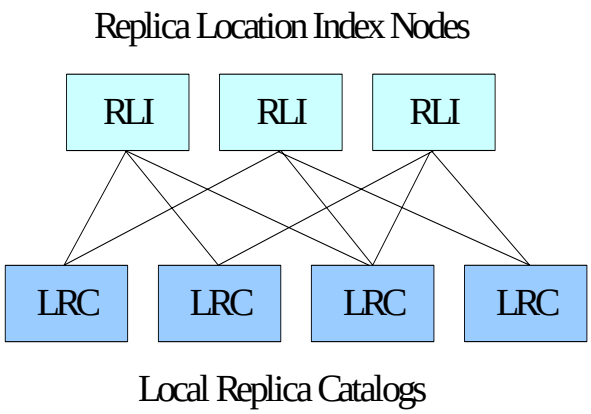

Figure 2. RLI to LRC mapping
From Globus website

Finally, the LRCs and RLIs that make up the RLS must be maintained as failures occur, new components join the system, etc. This is done by having each LRC and RLI server maintain a set of metadata about itself. This metadata is broadcast to its known locations periodically allowing the information to be reconstructed upon failure. This metadata includes such things as policies, partitioning information, etc.

A data file may have replicas in several different geographic and / or administrative domains with information about those replicas existing in multiple replica catalogs. With the RLIs, the system can be configured as either one large centralized index server or as a distributed set of index servers, each with a subset of the indices. This flexibility addresses the single point of failure requirement outlined for an RLS. [5]

The RLS component of the Globus toolkit is aimed at replicating data that is write once, read many as outlined previously. This matches the scientific applications with make up the bulk of their user community. Strict consistency is not maintained – the soft state information is adequate for their needs. In addition, the RLS component does not address file transfer or replica creation, it simply manages the index information about the replicas. For file transfer, we turn to the GridFTP component.

5.1.2 GridFTP

GridFTP is based on standard FTP with extensions to provide features necessary in a Data Grid environment. According to their website [14], these features include:

- Security
- Performance using parallel streams
- Performance using striping
- Partial file transfer
- Third party control / re-tartable data transfer
- Data extensibility
- Protocol extensibility

The extensions GridFTP provide enable a secure Public Key Infrastructure (PKI) interface as well as support for delegated authority using X.509 certificates to enhance the security necessary for the Data Grid environment. Delegated authority is a requirement in the grid environment where hundreds of resources can be utilized and single point sign-on is a necessity.

GridFTP also supports multiple TCP streams in parallel between source and destination. This is a common feature in high speed transfers these days allowing performance improvements of 3 to 5 times rates seen using FTP. To further enhance performance, it also supports striping – i.e. the ability to receive sections of data from multiple servers.

Another key to transferring large datasets is the ability to partially download a file. This is accomplished by identifying a byte position within the file to begin and end. GridFTP also supports restart capability, providing restart markers on a regular basis between the client and server. The last two features are extensibility for data and protocols. Because the interface abstracts the transfer from the underlying storage, other types of storage (such as SRB covered in section 5.2) can be put behind the transfer.

The server support is provided inside the Globus toolkit itself as a modified wu-ftpd server that supports most of the client's features. Evaluation of the GridFTP protocols – especially the usage of parallel threads – showed a marked increase in transfer speeds and much better link utilization that FTP. In their latest paper [15] new additional features released include pipelining for small files transfers, additional protocol support and network provisioning for binding transfers to optical paths.

Most of the features provided by GridFTP address the technical issues associated with speeding up transfers – but do not address the issue of diverging replicas. As was earlier stated, the Globus toolkit handles scientific applications for the most part that write once and read many. Diverging replicas is an issue that will arise as a future research direction.

5.1.3 Reliable File Transfer (RFT)

Although GridFTP provides the transport mechanism for the Globus toolkit, it provides a FTP like interface to the users for use. The Reliable File Transfer (RFT) components and the Data Replication Service covered in section 5.1.4 add the features to allow the user to manipulate the modules easier. RFT provides a web service interface to the user for use – as well as resolving an issue raised by the FTP protocol.

GridFTP requires that the client in control of the remote transfer keep a control channel (a TCP/IP socket) to each of the participating servers during the transfer. This is not really feasible in a world of mobile clients or in the situation of network failures. The key service RFT provides is to act as the proxy for the user in this situation, maintaining the open socket for as long as necessary (sometimes weeks). In addition, it acts much like a batch submission agent, accepting a SOAP descriptor of a set of data transfers – sometimes hundreds of files – and stores the descriptor in a database to provide the ability to recover from failures. This database provides the user query capability of their transfer requests.

RFT removes the manual requirement for GridFTP, allowing the user to identify a set of files to be transferred, provides recovery

capability, and provides status information to the user about the transfer request. It also checks that the appropriate authentication is possible before acting as a proxy for the user. There is one last component of the Globus toolkit that makes up their Data Management System, the Data Replication Service.

5.1.4 Data Replication Service (DRS)

The Data Replication Service (DRS) component of the Globus Toolkit provides an interface that ties the RLS and the GridFTP together for the users. [16] This is a fairly new component with its beta release in the most recent release of the Globus toolkit (4.0). The functionality of this component includes the ability for users to identify a set of desired resources (files) in their virtual organization, to make local replicas of those files, and to register these replicas with the RLS. It also implements a query interface for determining the state of the replication activity and for determining the state of various resources.

5.2 Storage Resource Broker (SRB)

The next replication system examined is the Storage Resource Broker (SRB) system from San Diego Supercomputer Center. [18] The intent of the SRB system is to enable the creation of shared collections through management of consistent state information, latency management, load leveling, logical resource usage, and multiple access interfaces [5].

5.2.1 SRB Architecture

The architecture of SRB is laid out in three levels. Level 0 is the storage resource, level 1 is a middleware layer, and layer 3 is the application API and the metadata catalog (MCAT) [19]. A file system or database is managed as a physical storage resource (PSR) which sets of are combined to make up a logical storage resource (LSR). The layout of data in the SRB is organized in a hierarchical fashion akin to the UNIX file system. Collections of data are implemented using LSRs while the data within a collection can be located on any PSR. Information about the data in a PSR, such as access information, is stored with the system descriptive attributes which record user information and the attributes of the collections and PSRs. The metadata is stored within the MCAT and is accessed by queries against the MCAT interface.

Level 1 is a client-server based middleware layer that virtualizes the data space. The purpose of this layer is to provide a unified view across heterogeneous storage resources that exist on the network. It is made up of the SRB master daemon and the agent processes. Clients are authenticated to the master daemon which starts up an agent processes to handle the client requests. This agent contacts the MCAT database by talking to an MCAT Enabled Server (MES). MES translates the request into an SQL query against the MCAT database, which returns the location of the file requested by the client. The agent then contacts the appropriate PSR to have the file transferred to the user.

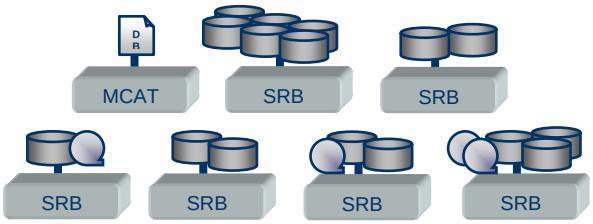

Figure 3. SRB Architecture
from SRB website

Level 2 is the Application API and MCAT database (although the MCAT database appears to span both Level 1 and Level 2). This provides a file system like interface through Windows, MacOS, or a set of Unix-like commands that is akin to the traditional file system interfaces in operating systems today. In addition, it provides a C and a web interface depending on the customer needs.[21]

The SRB is a federated system, with each instance of SRB managing/brokering a set of storage resources [20]. The configuration of

SRB allows for more than one server to manage/broker a storage resource, providing the redundancy necessary for failure recovery. It also provides location transparency because the users can connect to any SRB server to get to data from any other SRB in the federation. Replicas may be created on different storage systems on different hosts – providing the performance improvements desired in the data replication system. A mechanism is provided to determine a replication strategy – allowing for placement of data in distributed caches and access to archival copies.

The SRB provides a security model balanced against convenience. The system uses an Encrypt1 challenge/response, which is secure against network eavesdropping, while the use of user passwords is convenient and straight-forward for both users and administrators. User passwords can be stored on the host systems which opens the door for compromise. It uses the Globus Security Infrastructure component for authentication. GSI is also secure against network eavesdropping and somewhat less vulnerable against compromised hosts as only temporary delegation certificates are stored in files.

5.2.2 SRB I/O

The transport capabilities of the SRB are accomplished by the SRB I/O system. This system provides similar features to those in the GridFTP component of Globus toolkit. These features include parallel data transfers for high-speed bulk data transfers. It also supports multiple streams from multiple servers – a feature called slicing in GridFTP. Multiple files can be transferred as a single container and SRB I/O also supports staging files from tape or archival storage to disk.

Security for files being transferred by SRB I/O is handled by fine-grained access controls on the data. Credentials are authenticated through the MCAT and tickets are issues for users with read access to the data. These tickets can be time based or use based depending on the configuration. Controls can be placed at any level in the collection hierarchy – allowing the users to control the file system much like Unix. In fact, the organization of the security appears to be Unix-based with access list modifications added.

SRB I/O also supports remote procedure calls. This allows operations to be performed on the data without the data being moved. This is very useful for SQL queries, update of state information, providing attributes for filtering the data, and metadata queries. The users can specify that data can only be modified through the use of remote procedure calls – thereby allowing another level of security [5].

5.2.3 SRB features

With the ability to have collections of data, SRB will support the concept of virtual organizations much like the Globus toolkit. Data can be grouped into logical collections matching the participants of the virtual organizations. Once grouped into these logical collections, administration and management for these collections is accomplished within the existing framework [20].

Like the Globus toolkit, SRB has been ported to a variety of platforms and operating systems. This is an absolute requirement to exist in the heterogeneous environment of the grid. Equal possibility exists that the users could want to access the data from a laptop as well as a supercomputer – the interfaces must be able to handle them.

In addition, much like a distributed database system, SRB provides versioning control and locking mechanisms necessary to manage distributed data. This is key to allowing multiple collaborators to share data to ensure consistency. Tbere exists a link between the version control / locking mechanism and the security authentication system to allow them to work in conjunction with each other.

5.2.4 SRBRack

Dedicated servers built for storing and brokers SRB data and information are called SRBRacks. These SRBRacks are stored at geographically

distributed sites and from a partnership among themselves. Each SRB server system includes two types of SRBRacks, MCAT information and the server racks themselves. The MCAT rack contains the information about a site and may have several instances for performance and redundancy. Master / slave database techniques are used to keep the information on multiple servers current.

Each of the server systems is a file server system running the SRB master daemon and the agents. They can either satisfy the request or collaborate with other SRB systems to move the data to a local resource. They may or may not have an archival component depending on the replication of the data contained.

5.3 Grid Datafarm

The third system that I looked at as a part of my research is the Grid Datafarm (GFarm) [22] created by team headed by Osamu Tatebe – an associate professor at the University of Tsukuba and the current head of the Open Grid Forum committee on data replication [23]. GFarm has an architecture that combines storage capabilities, I/O bandwidth, and processing capability to create a system that scales to handing petabytes (PB) of data. While the Globus toolkit and SRB systems that we looked at in earlier chapters are in wide use and have tens to hundreds of implementations, GFarm is a much newer system without that same exposure. For SC2003, they had set up a trans-pacific test bed to illustrate their implementation [24], and a review of their website [21] showed implementations using astronomical data currently in place.

As opposed to SRB and Globus, GFarm is touted not as data replication system, but as a network file system targeted as a replacement for NFS. As I read through the papers however, the implementation of a network file system designed to handle petabytes of data has the same characteristics as the data replication systems in use by the other two. Like the other two, GFarm provides a logical name space which is independent of the physical storage. It allows scalable I/O bandwidth by integrating the process scheduling with the data distribution. This is an important difference over the other two where the user was required to perform the scheduling themselves.

5.3.1 GFarm Filesystem

A GFarm file is a large file stored in the file system across multiple locations. The amount of the information at each location is arbitrary and can be stored on any node. These “fragments” that are stored can be replicated at a number of sites and like the other two systems, the metadata about the replicas is managed by the GFarm system. Unfortunately (or at least in my view), like the other two systems, the files are write-once and read-many with versioning control being done by the system [5].

The target user for a GFarm system is a data-intensive application in which the same program is executed over multiple data files. While executing the program, the scheduler dispatches the process to the node that contains the data – or a replica of the data. If the node that contains the data is overloaded for a variety of reasons, the GFarm system creates a replica of the file on a quiescent node and assigns the processing for that file to the new node. Because the processes and the data locality are under the control of the system, I/O bandwidth is conserved. Like any network file system, caching can be done to improve the overall performance of the system.

5.3.2 Target users

The earlier statements about how the processes and files are assigned to nodes lead us to the specificity of users that can take advantage of this system. GFarm is targeted for those systems that are tightly coupled yet require large scale computing. This targeting makes it ideal in the scientific computation community, where huge data files are continuously processed to provide analyzed results for the next phase of the research.

Examples of scientific applications that fit this criteria are astrophysics and high-energy physics in which the analysis is akin to “finding a needle in a haystack”.

5.3.3 Features and Requirements

The GFarm system uses a file system daemon called gfsd that runs on each GFarm node. These daemons provide authentication of the users for access control of the fragments as well as file replications, fast invocations, node resources status monitoring, and control[23]. Because of the nature of this system, high-speed bandwidth is a requirement for its success. Some alleviation of this occurs when the locality of the data matches the processing, but like all network applications, bandwidth is a key.

GFarm supports file recovery and regeneration as well. This is a critical in an environment where disk and node failures occur commonly as opposed to occasionally. These failures can take the form of hardware/software failures, but more commonly are caused by temporal failures where the node is off-line for a period of time for some purpose. As long as one fragment continues to be on-line, these are transparent to the user. When there is no replica fragment on-line, the GFarm fragments are recreated dynamically by re-computation. The necessary components to recreate the fragment are stored as a part of the metadata associated with the fragment and the lost file/fragment is recomputed. This procedure is not without strife however, the data must meet the non-deterministic requirements of the system for this to be accomplished successfully.

Like the other systems, GFarm supports parallel I/O streams as well as multi-thread, multi-server data streams. These are managed through the use of the fragments and the indexing done within the metadata. A C-based API is provided to allow the users to specify which and what they require which is fed into the scheduler and file system to create the necessary components.

6.0 Compare/contract existing systems

As I read through the papers and created the sections above, the first comparison I would make is the similarities to the implementations of these systems and the distributed database systems we are reviewing in class. If each file or fragment of a file (depending on the implementation) is treated as a data value in a distributed database system, the behavior of those systems matches that of those in the distributed database system. The biggest exception to that is that each of the three systems I reviewed did not support the ability to update existing information – rather they all provided versioning control in which updates were done at the file level. While I view this as a limitation – the environment in which these systems are utilized does not. Write-once, ready many systems where version control is used to handle updates is adequate in an environment where read-only data is the normal mode of operation.

An attribute they all shared as well – and one critical within the grid environment – is the ability to separate the logical name space from the physical. This attribute is necessary for the parallel I/O abilities that all three systems supported, including both multiple thread and multiple server support. This requirement – but not only this requirement – leads to the need for metadata about the file systems to be stored as a part of the system and to provide the capability for that metadata to be replicated for performance and redundancy reasons.

One key difference from a system view is their age. Both the Globus toolkit and the SRB were created before 2000 and their user base is substantial. GFarm had its initial papers released in 2002 and their trial data was published in 2004. Due to this, Globus and SRB are working on refinements and improvements to an established methodology where GFarm is at release 1.4 and still working on getting the initial design requirements up and debugged. Because the applications using Globus and SRB are similar in nature, work has been put in to integrate the two packages. Globus has an DCI (data control interface) to link to the SRB interface so Globus users can use the SRB back-end for heterogeneous storage. SRB has the ability to use the Globus authentication system to provide the credentials for its users when using either the two systems together, or using SRB alone. The GFarm website [22] mentioned some early collaboration with the Globus team, but the details were limited.

6.1 Features

6.1.1 Multiple Streams/Servers

In the grid environment, the ability to move data using multiple streams, as well as from replicas on multiple servers is an important feature and one that all three systems reviewed had implemented. The Globus implementation required an additional component of a proxy server to handle the control channel requirements associated with GridFTP – a procedure that appears to be bolted on rather than designed in – but it still accomplishes the task for the user. SRB has the features built into their process, and the GFarm system had the capabilities built into the process / data distribution system.

6.2 Replication Strategy

The Globus toolkit and SRB are primarily data storage and replication manager systems, where the GFarm system was more than a replication manager; it was also a process scheduling system. The Globus toolkit had implemented in its latest release the set of strategies called caching plus cascading replication described in section 4.3. SRB had a less sophisticated replication strategy, allowing for the placement of data in distributed caches. Since GFarm was closely tied with the process scheduling, its methodology was the most sophisticated of the three. It would determine the best location for the compute cycles based on location of the data, as opposed to the alternative employed by SRB and Globus – taking the data to the compute cycles. In addition, GFarm would recognize when a node was loaded, create a replica at a new site and move the cycles to that new site. This was a great strength of GFarm, but only possible because of the tight coupling to the application type.

In addition, GFarm has the ability to recreate data segments because of that same tight coupling. The metadata contained enough information for the system to recreate any missing datasets due to replicas being off-line or failures.

6.3 Security

All three systems had security components to protect the replicas being generated. Globus has an authentication component as a part of the toolkit that would allow delegated authority using X.509 certificates. SRB had a distributed Unix-like login for users that were more concerned with convenience that tight security – and provided the capability to use the Globus authentication system for the same distributed single sign-on capability. GFarm has security embedded in its metadata creation – which controlled the creation and access of the replicas.

7.0 Summary

SRB and the Globus toolkit are both systems that have been under development for over a decade, and their maturity reflects that. The GFarm system does not have the same development time – and focuses on solving a much narrower target with more features within that target.

The Globus toolkit provides a wide range of features and components due to its open source – when one of the participating forum members needs features, they work with the Globus team to develop. Even so, in the area of replication, the tools are having concepts such as replication strategy added to them as early as this year. This reflects their background as a computation system first and moving into the data grid environment as the system became more general.

The main strength of the SRB system is the interface to the heterogeneous storage system back-end. They have abstracted this interface to support a wide range of data storage devices and/or systems. Their background as a storage broker provides them with a design better suited to support the data replication methodology. They have also had the foresight to integrate their systems with the Globus systems allowing the best of both the data grid and the compute grid environments.

The GFarm system is the most intriguing to me. Because of their narrow target, I had originally felt this was a point solution – however, the concepts they are introducing can be carried forward to be more general purpose. The concepts of fragments stored on multiple systems can provide huge performance increases in distributed data replication systems.

8.0 Related work for data grids

Data grids are not only used for the scientific applications that are the focus of this paper. Any large scale distributed database could conceivably be called a data grid with its own needs and requirements. An excellent example of this is in the arena of digital libraries.

The focus of a digital library is the distribution of data and a set of persistent archives for the preservation of data. The basics of providing these capabilities are already present in the storage mechanisms – such as SRB reviewed earlier in this paper. Digital libraries can be implemented on top of data grids through the addition of tools that support library functions such as browsing, creation, and discovery. [27] In addition, support for bulk metadata loads, import and export of data in XML format, and management of collection hierarchies can be added to the existing framework to provide the capabilities necessary.

This arena would be a wide divergence from the focus of the traditional grid resources, namely scientific computing and high performance computing. With the advent of XML document wrappers – something I found a tremendous number of papers written about during my research – creating document libraries using the existing data grid technologies is a natural progression. Rebecca Moore – one of the primary authors of SRB – has turned the focus of her research to this area, with the first paper coming out in 2004.

9.0 Future work

Doing the research for this paper has been fascinating for me. My impressions are that the data grid environments have had some focus over the last 10 years, but that the sophistication of these systems is still being improved.

There exists two forums that I will continue my research in these areas as well as follow up on additional papers on the topics. The Open Grid Forum is a grid standardization body that is creating the framework for tool chains in the grid environment going forward. The other is the Global Grid Forum. "The objective of the Global Grid Forum is to promote and develop Grid technologies and applications via the development and documentation of "best practices," implementation guidelines, and standards with an emphasis on rough consensus and running code." [25] according to Ian Foster's website.

Ian Foster is a recognized leader in the field of grid computing research. His website is loaded with references for those in my situation of learning about the grid environments – this is a resource for me to follow up on next semester as well.

Writing this paper was a lot of fun for me – another piece to the puzzle.

References

[1] Chervenak, A., Schuler, R., Kesselman, C., Koranda, S., & Moe, B. Wide Area Data Replication for Scientific Collaborations. November, 2005. In *Proceedings of the 6th IEEE/ACM Workshop on Grid Computing.*

[2] Casanova H. Distributed Computing Research Issues in Grid Computing. September 2002. *ACM SIGACT News, Vol. 33, Issue 3,* pp 50-70.

[3] Lamehamedi, H., Szymanski, B., and Shentu, Z. Data Replication Strategies in Grid Environments. 2002. In *Proceedings of the 5th Internation Conference on Algorithms and Architectures for Parallel Processing (ICA3PP '02)*

[4] Ranganathan, K., & Foster, I. Design and Evaluation of Dynamic Replication Strategies for High Performance Data Grids. September, 2001. *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics. Bejing, China.*

[5] Venugopal S., Buyya R., & Ramamohanarao K., A Taxonomy on Data Grids for Distributed Data Sharing, Management, and Processing. March 2006. *ACM Computing Surveys, Vol. 38,* Article 3.

[6] http://www.globus.org/toolkit

[7] http://www.csm.ornl.gov/pvm/

[8] D. Menasce, E. Casalicchio. *Quality of Service Aspects and Metrics in Grid Computing*. IEEE Internet Computing, Vol. 8, No. 4, July/August 2004.

[9] Rajasekar, A., Wan, M., Moore, R., Kremenek, G., & Guptil, T. Data Grids, Collections, and Grid Bricks. *Proceedings of the 20th IEEE NASA Goddard Conference on Mass Storage Systems and Technologies (MSS '03).* pp. 2-10

[10] Raman, R., Livny, M., and Solomon, M. Matchmaking: Distributed Resource Management for High Throughput Computing. *7th IEEE Internation Symposium on High Performance Distributed Computing (HPDC-7),* July 1998.

[11] Chervenak, A., Deelman, E., Foster, I., Guy, L., Hoschek, W., Iamnitchi, A., Kesselman, C., Kunst, P., Ripeanu, M., Schwartzkopf, B., Stockinger, H., Stockinger, K., & Tierney, B. Giggle: A Framework for Constructing Scalable Replica Location Services. 2002. *Proceedings of the IEEE/ACM Conference on Supercomputing*.

[12] Chervenak, A., Palavalli, N,, Bharathi, S., Kesselman, C., & Schwartzkopf, R. Performance and Scalability of a Replica Location Service. June, 2004. *High Performance Computing Conference (HPDC-13), Honolulu, HI.*

[13] http://www.globusconsortium.org/

[14]http://www.globus.org/toolkit/data/gridftp/advantage.html

[15] Bresnahan J., Link, M., Khanna, G., Imani, Z., Kettimuthu, R., Foster, I. Globus GridFTP: What's new in 2007 (Invited Paper), *Proceedings of the First International Conference on Networks for Grid Applications (GridNets 2007),* Oct, 2007

[16]http://www.globus.org/toolkit/docs/4.0/techpreview/datarep/DataRepFacts.html

[17] Allcock, W., Foster, I., Madduri, R. Reliable Data Transport: A Critical Service for the Grid. *Building Service Based Grids Workshop, Global Grid Forum 11,* June 2004.

[18] http://www.sdsu.edu/srb/

[19] Rajasekar, A., Wan, M., Moore, R., Schroeder, W., Kremenek, G., Jagatheesan, A., Cowart, C., Zhe, B., Chen, S., &Olschanowsky, R. October, 2003. Storage Resource Broker – Managing Distributed Data on the Grid.

Computer Society of Inda Journal, Special Issue on SAN. Vol. 33, No. 4. pp. 42-54

[20] Rajasekar, A., & Wan, M. SRB & SRBRack – Components of a Virtual Data Grid Architecture. April, 2002. *Advanced Simulation Technologies Conference (ASTC02), San Diego, CA.*

[21]http://www.sdsc.edu/srb/index.php/FAQ

[22] http://datafarm.apgrid.org/

[23] Tatebe, O., Morita, Y., Matsuoka, S., Soda, N. and Sekiguchi, S. 2002. Grid datafarm architecture for petascale data intensive computing. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid '02).* Berlin, Germany. IEEE Press, Los Alamitos, CA.

[24] Tatebe O., Ogawa H., Kodma, Y., Kudoh, T., Sekiguchi S, Matsuoka S., Aida K., Boku T., Sato M., Morita, Y., Kitatsuji, Y., Williams, J., & Hicks, J. 2004 The second trans-pacific Grid datafarm testbed and experiments for SC2003. In *Proceedings of the International Symposium on Applications and the Internet – Workshops (SAINT '04).* Tokyo, Japan

[25] http://www-fp.mcs.anl.gov/~foster/

[26]http://en.wikipedia.org/wiki/Data_replication

[27] Moore, R., Rajasekar, A., Wan, M. Data Grids, Digital Libraries, and Persistent Archives: An Integrated Approach to Publishing, Sharing, and Archiving Data. *Invited Paper - Proceedings of the IEEE, Vol. 93, No. 3.* March 2005