IBM

Introduction to Grid Computing with Globus

- Fundamentals and concepts
- Using the Globus Toolkit
- Grid demos and OGSA

Luis Ferreira,
Viktors Berstis,
Jonathan Armstrong,
Mike Kendzierski,
Andreas Neukoetter,
MasanobuTakagi,
Richard Bing-Wo, Adeeb Amir,
Ryo Murakawa, Olegario Hernandez,
James Magowan, Norbert Bieberstein

Redbooks

ibm.com/redbooks

International Technical Support Organization

Introduction to Grid Computing with Globus

December 2002

SG24-6895-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xiii.

First Edition (December 2002)

This edition applies to Globus Toolkit 2.2 running on Red Hat Linux Version 7.3.

Contents

Figures

Tables

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AFS®	IBM eServer™	pSeries™
AIX®	IBM®	Redbooks (logo)™
DataJoiner®	IMS™	Redbooks™
DB2®	Informix®	SP™
DFS™	iSeries™	WebSphere®
e-business on demand™	LoadLeveler®	xSeries™
Home Director™	Perform™	zSeries™

The following terms are trademarks of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both:

Lotus®	Word Pro®

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM Redbook is intended to give readers interested in the technical aspects of grid computing a hands-on introduction using the Globus Toolkit. This will include a discussion of the first basics of grid computing, applications to be run on the grid, and various grid products and architectures that are currently available.

This redbook is the first of several more books and papers on grid computing that are yet to come, both technical and non-technical. It serves as a good starting point and foundation before learning more about the next steps coming in grid computing, OGSA, e-business, and IBM's vision of the on-demand era.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Luis Ferreira, also known as “Luix”, is a Software Engineer at IBM Corporation - International Technical Support Organization, Austin Center, working on Linux and Grid Computing projects. He has 19 years of experience with UNIX-like operating systems, and holds a MSc. Degree in System Engineering from Universidade Federal do Rio de Janeiro in Brazil. Before joining the ITSO, Luis worked at Tivoli Systems as a Certified Tivoli Consultant, at IBM Brasil as a Certified IT Specialist, and at Cobra Computadores as a Kernel Developer and Software Designer.

Viktors Berstis is a Senior Software Engineer at IBM Corporation currently working on the Grid Computing Initiative in Austin Texas. His experience at IBM includes architecting the System/38 - AS/400, developing various compilers, research on high-level automated integrated circuit design, OS/2, JavaOS, and many other projects. He is a senior IEEE member and is an IBM Master Inventor. He received his Bachelor of Science in Mathematics and Physics and Master of Science in Computer Information and Control Engineering at the University of Michigan.

Jonathan Armstrong is a software engineer for IBM Global Service's e-Technology Center. He graduated from Texas Tech University in December of 2001 with a B.S. in Computer Engineering. He joined IBM in the e-Technology Center's Grid Computing Initiative at its inception. He has worked with many grid products and taken part in customer education workshops.

Mike Kendzierski is an IT Architect working within the Integrated Technical Services division of IBM Global Services. His focus is on IT Infrastructure and Security Architecture, where he has been working as an architect for over six years across a multitude of platforms and systems from networking, distributed systems, enterprise messaging, middleware, and security. Mike has also published several books on e-commerce technologies and computer systems.

Andreas Neukoetter is a Software Engineer for Grid Computing working at IBM Development Lab Boeblingen, Germany. He has a strong relationship to Linux/UNIX, over ten years, using only Linux for day to day business over the last five years. He joined IBM in 1995 and moved to the Boeblingen Lab in 1998. Six month ago, he was assigned to the Grid Team; this team's task is to drive the grid development and deployment inside and outside of IBM. He teaches at the University of Cooperative Education (Berufsakademie) in Stuttgart.

Masanobu Takagi is IT Engineer at IBM Global Services Japan. He has two years of Linux experience with project and customer support, and holds a MSc. Degree in Social Informatics from Kyoto University, Japan.

Richard Bing-Wo is an IT specialist with IBM Global Services, ITS, as a Systems Management and Test Specialist. His group, Migration Services/ESS, assists clients in managing existing systems as well as moving them to IBM platforms. Richard has 12 years of UNIX experience, the last two of which has been with IBM. He has a proven track record in applying technologies in various network/distributed computing environments with client/server applications.

Adeeb Amir is a Senior IT consultant for IBM Global Services, ITS. He has fifteen years of experience in systems management and network design and implementation. His areas of experience also include HA clusters, performance and tuning, troubleshooting, and server architecture. On the OS side, he is most experienced with UNIX ptx, AIX, Solaris, and Windows.

Ryo Murakawa is an IT Engineer at IBM Japan Systems Engineering Co., Ltd. He has seven years of Linux experience and 10 years of Unix-like experience, and five years of developing system applications. He holds a MSc. Degree in System engineering from Hiroshima University, Japan.

Olegario Hernandez is a former IBM Advisory Systems Engineer with 30 years of experience with IBM. He graduated as a Chemical Civil Engineer from the Universidad de Chile. During his time in IBM, as a regular employee, he was working in application development, disciplines of systems management, IS Architecture, CICS Application Interface, and business systems planning methodology (BSP). Has been assigned for residencies at different centers of the IBM International Technical Support Organization: ITSC Boeblingen for CICS Application Interface, ISC Gaithersburg for AD/Cycle, and ITSC Poughkeepsie, twice, for Systems Management and SystemView. After his retirement from IBM,

he has been a resident in the ITSO Austin for Architecting Secure Systems with Tivoli products and, now, for Grid Computing Architecture as an IT Systems Consultant for IBM Business Partners.

James Magowan is an IT Specialist with the Dynamic e-Business team within Hursley Services and Technology Department, UK. He has spent several years working with high performance computing and installing, troubleshooting, benchmarking, and tuning SP installations, before his involvement with high performance computing lead to working with Linux clusters and then into grid computing. James has been an active participant within the Grid Technical Community since January 2002, both through the Global Grid Forum and grid computing projects. His projects include European DataGrid, Open Grid Services Database Access and Integration Project (OGSA-DAI), and internal grid accounting investigations. His involvement with the Global Grid Forum includes chairing two Working Groups and most recently submitting the Draft Database Access Specification.

Norbert Bieberstein is a solution development manager at IBM Software Group in EMEA, located in Düsseldorf, Germany working with system integrators and ISVs for about five years. He has also worked as an IT architect at IBM's application architecture project office in Somers, NY and, before that, as a consultant for CASE and software engineering at IBM’s software development labs. This work resulted in a book on CASE technology that was published in Germany in 1993. He has organized several educational events for IBMers, business partners, and customers on IBM SW products. In 1997, he acted as the coordinating editor of the awarded IBM Systems Journal, 1/97 Edition. Before joining IBM in 1989, he worked as an application and system developer for a software vendor of CIM/ERP systems. Norbert holds a Masters degree in mathematics from University of Technology Aachen, Germany, and developed their evaluation systems for electron microscopes. He is currently studying for an MBA at Henley Management School.

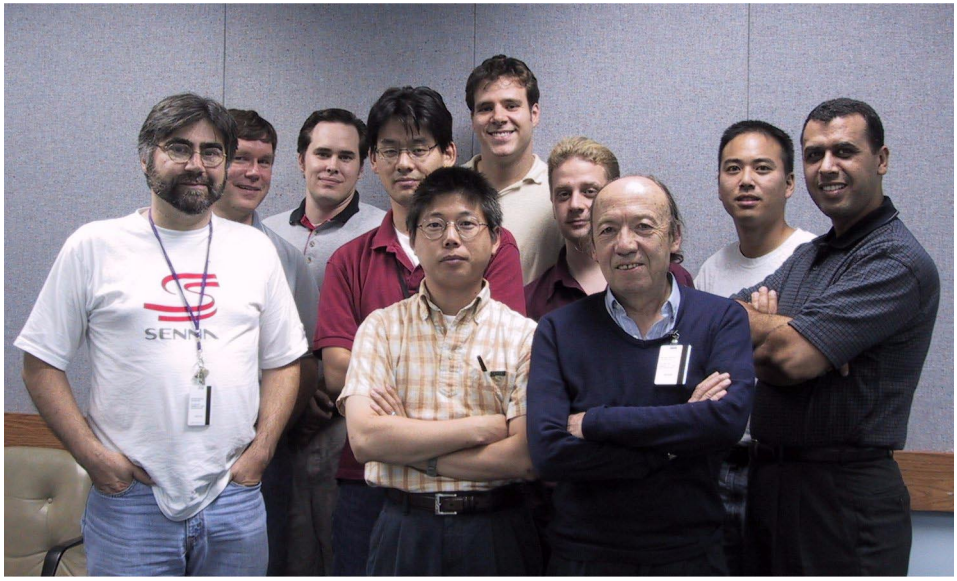

Figure 1 The Blue Grid-Tuxedo Team

Figure 1 shows the Blue Grid-Tuxedo Team. From left to right they are Luis, Viktors, Jonathan, Ryo, Masanobu, Mike, Andreas, Olegario, Richard, and Adeeb.

Acknowledgements

Thanks to the following people for their contributions to this project:

Lupe Brown, Bart Jacob, Wade Wallace, Julie Czubik, and Chris Blatchley
International Technical Support Organization, Austin Center

Cary Perkins, Candice Gilzean, and Chris Reech
Grid Computing Initiative, e-Technology Center, IBM Austin, USA

Benjamin Khoo
Linux, HPC, and Grid Computing IT Specialist at the Integrated Technology Services, IBM Singapore

Dr. Sebastian Wedeniwski
Consulting IT Architect, Software Solutions and Services, IBM Lab Boeblingen, Germany

Alan Fishman
Solution Manager, Linux Services Offerings, IBM Global Services, Seattle, USA

Matt Newton
IT Architect, IBM Global Services, T.J. Watson Research Center, USA

Jean Pierre Prost, Pierre Sabloniere, and Jean-Yves Girard
IBM Design Center for e-business on demand, EMEA ATS Products and Solutions Support Center Montpellier, IBM France

Atsuko Miyashita
Linux Support Center, System and Web Solution Center, IBM Japan

James Goethals
zSeries Networking Offering Manager, IBM RTP, USA

Eberhard Saemann, Tony Gargya, and Thilo Boehm
Grid Technologies and Solutions, IBM Boeblingen Lab, Germany

Art Cannon
IBM Global e-Business Solutions Center, USA

Chris Molloy
IBM Global Services, South SDC Technology Planning Department, USA

Christine L. Miller
IBM Grid WW Sales Executive - Life Sciences Sector, USA

Dennis Carden
SDC-NE Delivery Architecture Services - Strategic Outsourcing, IBM USA

Greg Kettmann
Grid Computing Architecture, IBM Americas

James Goethals
zSeries Networking Offering, IBM USA

Dr. Dave Watson
Program Director, Hursley Services and Technology, IBM UK

Joshua Horton
AIX/Linux Clusters Information Development, IBM USA

Mauro Gatti
EMEA xSeries Solution Architects, IBM Italy

Michael R. Haley
Global Solutions Executive - Grid Computing, IBM USA

Shawn Mullen
AIX - IP/Grid Security Development, IBM USA

Nina Wilner
LifeSciences TSM, Web Server Solutions Development, IBM USA

Paolo Di Napoli
IBM Learning Services, Italy

Ferry J Kubatz
EMEA - AP Curriculum Development Manager, IBM Learning Services, Netherlands

Rob Vrablik
IBM eServer Grid Marketing Strategy and Planning, IBM USA

Scott S. Denham
HPC Technical Architect, Worldwide High Performance Computing, IBM USA

Special thanks to the following people:

Ian Foster, **Steve Tuecke**, **Lee Liming** and the Argonne National Laboratory team for supporting us during the review process.

Charles Bacon, **Samuel Meder**, and **Larry Flon** from Argonne National Laboratory, for the excellent technical review.

Joanne Luedtke, **Paul Magnone**, and **John Adams** for the inspiration and John Adams' team (Grid Computing Initiative, and e-Technology Center) for all technical support provided during the project.

IBM's Grid Computing team in Sommers, in particular to **Matt Haynos** and **Andreas Hermelink**.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493

Part 1

Fundamentals

1

Grid computing

Grid computing, most simply stated, is distributed computing taken to the next evolutionary level. The goal is to create the illusion of a simple yet large and powerful self managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources.

The standardization of communications between heterogeneous systems created the Internet explosion. The emerging standardization for sharing resources, along with the availability of higher bandwidth, are driving a possibly equally large evolutionary step in grid computing.

The following major topics will be introduced to the readers in this chapter:

- What grid computing can do
- Grid concepts and components
- Grid construction
- The present and the future
- What the grid cannot do

1.1 What grid computing can do

When you deploy a grid, it will be to meet a set of customer requirements. To better match grid computing capabilities to those requirements, it is useful to keep in mind the reasons for using grid computing.

1.1.1 Exploiting underutilized resources

The easiest use of grid computing is to run an existing application on a different machine. The machine on which the application is normally run might be unusually busy due to an unusual peak in activity. The job in question could be run on an idle machine elsewhere on the grid.

There are at least two prerequisites for this scenario. First, the application must be executable remotely and without undue overhead. Second, the remote machine must meet any special hardware, software, or resource requirements imposed by the application.

For example, a batch job that spends a significant amount of time processing a set of input data to produce an output set is perhaps the most ideal and simple use for a grid. If the quantities of input and output are large, more thought and planning might be required to efficiently use the grid for such a job. It would usually not make sense to use a word processor remotely on a grid because there would probably be greater delays and more potential points of failure.

In most organizations, there are large amounts of underutilized computing resources. Most desktop machines are busy less than 5% of the time. In some organizations, even the server machines can often be relatively idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usage.

The processing resources are not the only ones that may be underutilized. Often, machines may have enormous unused disk drive capacity. Grid computing, more specifically, a “data grid”, can be used to aggregate this unused storage into a much larger virtual data store, possibly configured to achieve improved performance and reliability over that of any single machine.

If a batch job needs to read a large amount of data, this data could be automatically replicated at various strategic points in the grid. Thus, if the job must be executed on a remote machine in the grid, the data is already there and does not need to be moved to that remote point. This offers clear performance benefits. Also, such copies of data can be used as backups when the primary copies are damaged or unavailable.

Another function of the grid is to better balance resource utilization. An organization may have occasional unexpected peaks of activity that demand more resources. If the applications are grid-enabled, they can be moved to underutilized machines during such peaks. In fact, some grid implementations can migrate partially completed jobs. In general, a grid can provide a consistent way to balance the loads on a wider federation of resources. This applies to CPU, storage, and many other kinds of resources that may be available on a grid. Management can use a grid to better view the usage patterns in the larger organization, permitting better planning when upgrading systems, increasing capacity, or retiring computing resources no longer needed.

1.1.2 Parallel CPU capacity

The potential for massive parallel CPU capacity is one of the most attractive features of a grid. In addition to pure scientific needs, such computing power is driving a new evolution in industries such as the bio-medical field, financial modeling, oil exploration, motion picture animation, and many others.

The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts. A CPU intensive grid application can be thought of as many smaller “subjobs,” each executing on a different machine in the grid. To the extent that these subjobs do not need to communicate with each other, the more “scalable” the application becomes. A perfectly scalable application will, for example, finish 10 times faster if it uses 10 times the number of processors.

Barriers often exist to perfect scalability. The first barrier depends on the algorithms used for splitting the application among many CPUs. If the algorithm can only be split into a limited number of independently running parts, then that forms a scalability barrier. The second barrier appears if the parts are not completely independent; this can cause contention, which can limit scalability. For example, if all of the subjobs need to read and write from one common file or database, the access limits of that file or database will become the limiting factor in the application’s scalability. Other sources of inter-job contention in a parallel grid application include message communications latencies among the jobs, network communication capacities, synchronization protocols, input-output bandwidth to devices and storage devices, and latencies interfering with real-time requirements.

1.1.3 Applications

There are many factors to consider in grid-enabling an application. One must understand that not all applications can be transformed to run in parallel on a grid and achieve scalability. Furthermore, there are no practical tools for transforming arbitrary applications to exploit the parallel capabilities of a grid. There are some

practical tools that skilled application designers can use to write a parallel grid application. However, automatic transformation of applications is a science in its infancy. This can be a difficult job and often requires top mathematics and programming talents, if it is even possible in a given situation. New computation intensive applications written today are being designed for parallel execution and these will be easily grid-enabled, if they do not already follow emerging grid protocols and standards.

1.1.4 Virtual resources and virtual organizations for collaboration

Another important grid computing contribution is to enable and simplify collaboration among a wider audience. In the past, distributed computing promised this collaboration and achieved it to some extent. Grid computing takes these capabilities to an even wider audience, while offering important standards that enable very heterogeneous systems to work together to form the image of a large virtual computing system offering a variety of virtual resources, as illustrated in Figure 1-1 on page 7. The users of the grid can be organized dynamically into a number of virtual organizations, each with different policy requirements. These virtual organizations can share their resources collectively as a larger grid.

Sharing starts with data in the form of files or databases. A “data grid” can expand data capabilities in several ways. First, files or databases can seamlessly span many systems and thus have larger capacities than on any single system. Such spanning can improve data transfer rates through the use of striping techniques. Data can be duplicated throughout the grid to serve as a backup and can be hosted on or near the machines most likely to need the data, in conjunction with advanced scheduling techniques.

Sharing is not limited to files, but also includes many other resources, such as equipment, software, services, licenses, and others. These resources are “virtualized” to give them a more uniform interoperability among heterogeneous grid participants.

The participants and users of the grid can be members of several real and virtual organizations. The grid can help in enforcing security rules among them and implement policies, which can resolve priorities for both resources and users.

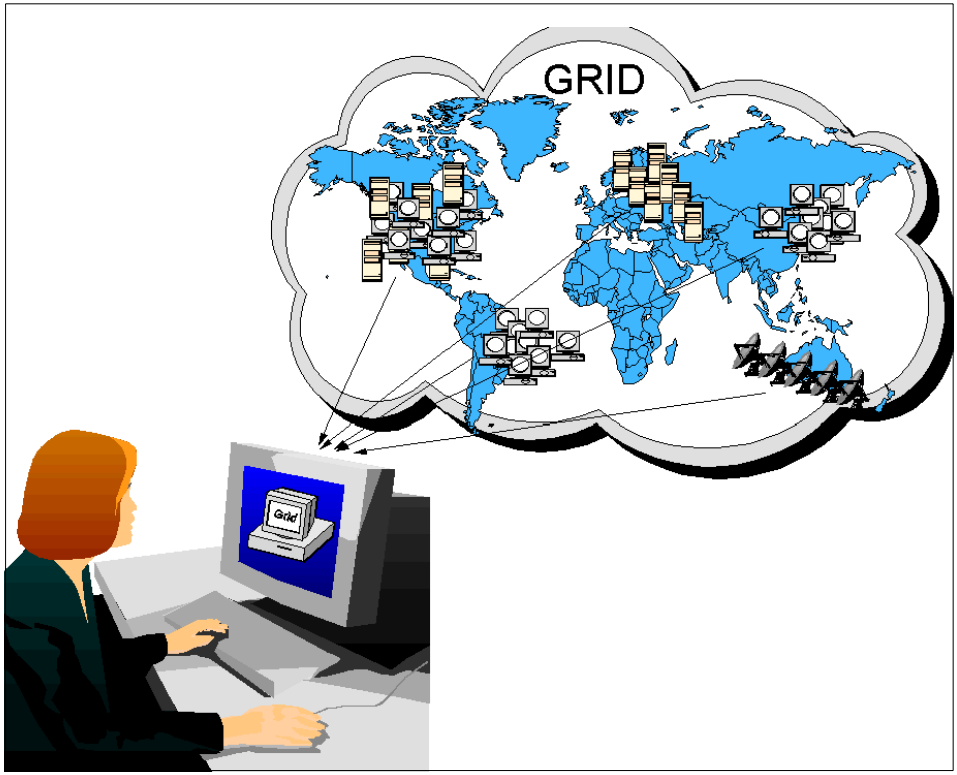

Figure 1-1 The grid virtualizes heterogeneous geographically disperse resources

1.1.5 Access to additional resources

In addition to CPU and storage resources, a grid can provide access to increased quantities of other resources and to special equipment, software, licenses, and other services. The additional resources can be provided in additional numbers and/or capacity.

For example, if a user needs to increase his total bandwidth to the Internet to implement a data mining search engine, the work can be split among grid machines that have independent connections to the Internet. In this way, the total searching capability is multiplied, since each machine has a separate connection to the Internet. If the machines had shared the connection to the Internet, there would not have been an effective increase in bandwidth.

Some machines may have expensive licensed software installed that the user requires. His jobs can be sent to such machines more fully exploiting the software licenses.

Some machines on the grid may have special devices. Most of us have used remote printers, perhaps with advanced color capabilities or faster speeds. Similarly, a grid can be used to make use of other special equipment. For example, a machine may have a high speed, self feeding, DVD writer that could be used to publish a quantity of data faster. Some machines on the grid may be connected to scanning electron microscopes that can be operated remotely. In this case, scheduling and reservation are important. A specimen could be sent in advance to the facility hosting the microscope. Then the user can remotely operate the machine, changing perspective views until the desired image is captured.

The grid can enable more elaborate access, potentially to remote medical diagnostic and robotic surgery tools with two-way interaction from a distance. The variations are limited only by one's imagination. Today, we have remote device drivers for printers. Eventually, we will see standards for grid-enabled device drivers to many unusual devices and resources. All of these will make the grid look like a large virtual machine with a collection of virtual resources beyond what would be available on just one conventional machine.

1.1.6 Resource balancing

A grid federates a large number of resources contributed by individual machines into a greater total virtual resource. For applications that are grid-enabled, the grid can offer a resource balancing effect by scheduling grid jobs on machines with low utilization, as illustrated in Figure 1-2 on page 9. This feature can prove invaluable for handling occasional peak loads of activity in parts of an larger organization. This can happen in two ways:

- An unexpected peak can be routed to relatively idle machines in the grid.
- If the grid is already fully utilized, the lowest priority work being performed on the grid can be temporarily suspended or even cancelled and performed again later to make room for the higher priority work.

Without a grid infrastructure, such balancing decisions are difficult to prioritize and execute.

Occasionally, a project may suddenly rise in importance with a specific deadline. A grid cannot perform a miracle and achieve a deadline when it is already too close. However, if the size of the job is known, if it is a kind of job that can be sufficiently split into subjobs, and if enough resources are available after preempting lower priority work, a grid can bring a very large amount of processing power to solve the problem. In such situations, a grid can, with some planning, succeed in meeting a surprise deadline.

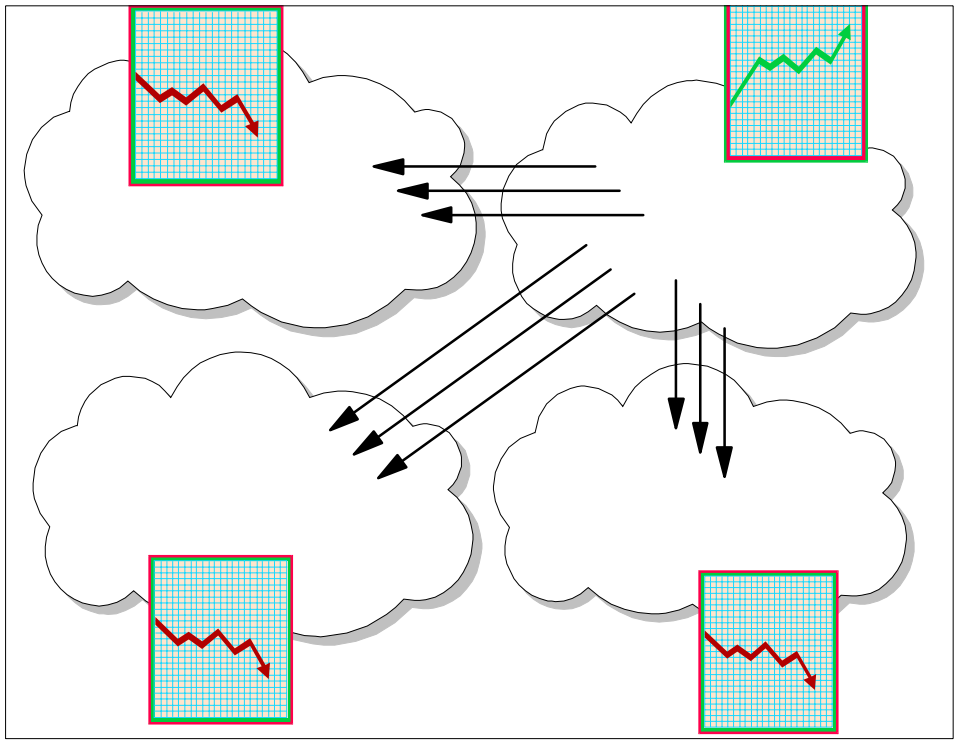

Figure 1-2 Jobs are migrated to less busy parts of the grid to balance loads

Other more subtle benefits can occur using a grid for load balancing. When jobs communicate with each other, the Internet, or with storage resources, an advanced scheduler could schedule them to minimize communications traffic or minimize the distance of the communications. This can potentially reduce communication and other forms of contention in the grid.

Finally, a grid provides excellent infrastructure for brokering resources. Individual resources can be profiled to determine their availability and their capacity, and this can be factored into scheduling on the grid. Different organizations participating in the grid can build up grid credits and use them at times when they need additional resources. This can form the basis for grid accounting and the ability to more fairly distribute work on the grid.

1.1.7 Reliability

High-end conventional computing systems use expensive hardware to increase reliability. They are built using chips with redundant circuits that vote on results, and contain much logic to achieve graceful recovery from an assortment of hardware failures. The machines also use duplicate processors with hot

pluggability so that when they fail, one can be replaced without turning the other off. Power supplies and cooling systems are duplicated. The systems are operated on special power sources that can start generators if utility power is interrupted. All of this builds a reliable system, but at a great cost, due to the duplication of high-reliability components.

In the future, we will see a complementary approach to reliability that relies on software and hardware. A grid is just the beginning of such technology. The systems in a grid can be relatively inexpensive and geographically dispersed. Thus, if there is a power or other kind of failure at one location, the other parts of the grid are not likely to be affected. Grid management software can automatically resubmit jobs to other machines on the grid when a failure is detected. In critical, real-time situations, multiple copies of the important jobs can be run on different machines throughout the grid, as illustrated in Figure 1-3. Their results can be checked for any kind of inconsistency, such as computer failures, data corruption, or tampering.

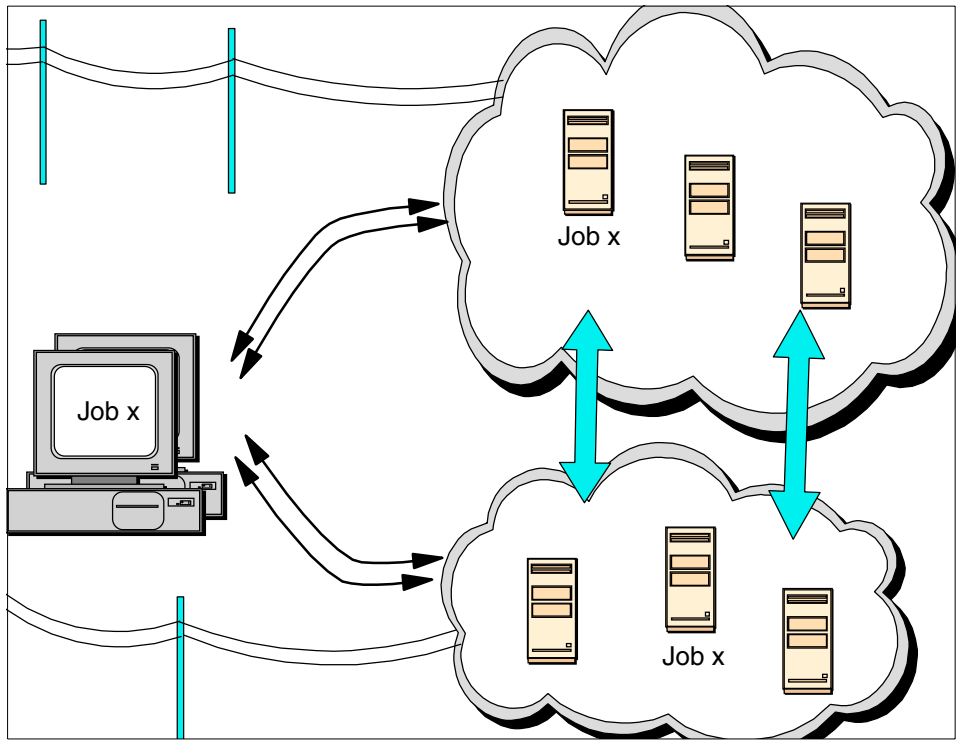

Figure 1-3 Redundant grid configuration

Such grid systems will utilize “autonomic computing.” This is a type of software that automatically heals problems in the grid, perhaps even before an operator or

manager is aware of them. In principle, most of the reliability attributes achieved using hardware in today's high availability systems can be achieved using software in a grid setting in the future.

1.1.8 Management

The goal to virtualize the resources on the grid and more uniformly handle heterogeneous systems will create new opportunities to better manage a larger, more disperse IT infrastructure. It will be easier to visualize capacity and utilization, making it easier for IT departments to control expenditures for computing resources over a larger organization.

The grid offers management of priorities among different projects. In the past, each project may have been responsible for its own IT resource hardware and the expenses associated with it. Often this hardware might be underutilized while another project finds itself in trouble, needing more resources due to unexpected events. With the larger view a grid can offer, it becomes easier to control and manage such situations. As illustrated in Figure 1-4 on page 12, administrators can change any number of policies that affect how the different organizations might share or compete for resources.

Aggregating utilization data over a larger set of projects can enhance an organization's ability to project future upgrade needs. When maintenance is required, grid work can be rerouted to other machines without crippling the projects involved.

Autonomic computing can come into play here too. Various tools may be able to identify important trends throughout the grid, informing management of those that require attention.

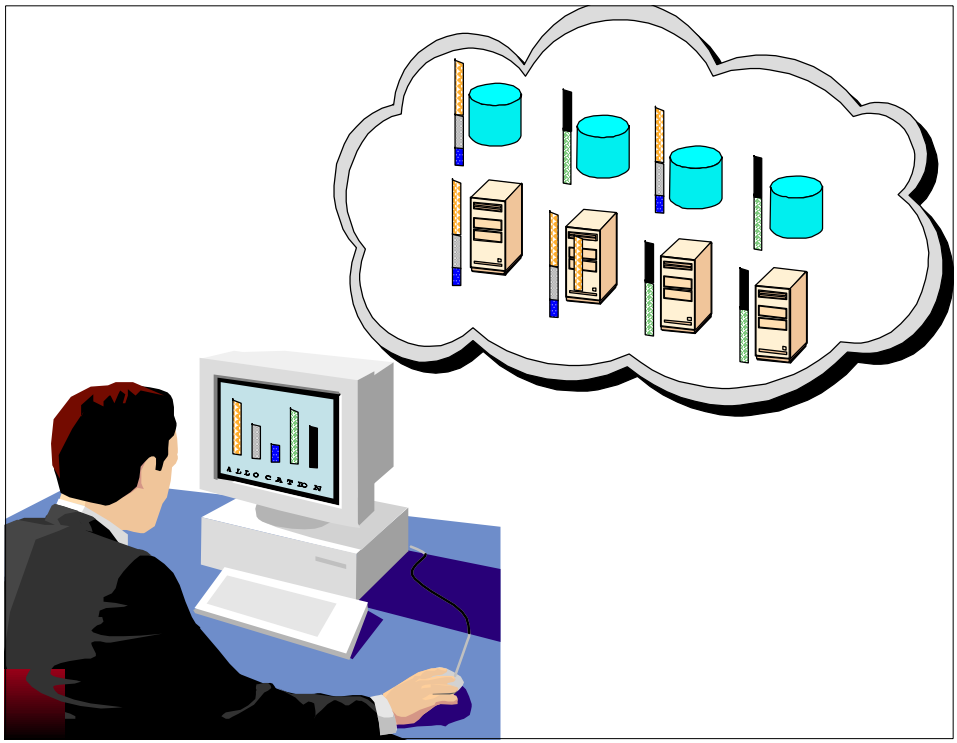

Figure 1-4 Administrators can adjust policies to better allocate resources

1.2 Grid concepts and components

In this section, we introduce the various grid concepts, components, and terms in more detail.

1.2.1 Types of resources

A grid is a collection of machines, sometimes referred to as “nodes,” “resources,” “members,” “donors,” “clients,” “hosts,” “engines,” and many other such terms. They all contribute any combination of resources to the grid as a whole. Some resources may be used by all users of the grid while others may have specific restrictions.

Computation

The most common resource is computing cycles provided by the processors of the machines on the grid. The processors can vary in speed, architecture, software platform, and other associated factors, such as memory, storage, and

connectivity. There are three primary ways to exploit the computation resources of a grid. The first and simplest is to use it to run an existing application on an available machine on the grid rather than locally. The second is to use an application designed to split its work in such a way that the separate parts can execute in parallel on different processors. The third is to run an application, that needs to be executed many times, on many different machines in the grid. “Scalability” is a measure of how efficiently the multiple processors on a grid are used. If twice as many processors makes an application complete in one half the time, then it is said to be perfectly scalable. However, there may be limits to scalability when applications can only be split into a limited number of separately running parts or if those parts experience some other contention for resources of some kind.

Storage

The second most common resource used in a grid is data storage. A grid providing an integrated view of data storage is sometimes called a “data grid.” Each machine on the grid usually provides some quantity of storage for grid use, even if temporary. Storage can be memory attached to the processor or it can be “secondary storage” using hard disk drives or other permanent storage media. Memory attached to a processor usually has very fast access but is volatile. It would best be used to cache data or to serve as temporary storage for running applications.

Secondary storage in a grid can be used in interesting ways to increase capacity, performance, sharing, and reliability of data. Many grid systems use mountable networked file systems, such as Andrew File System (AFS), Network File System (NFS), Distributed File System (DFS), or General Parallel File System (GPFS). These offer varying degrees of performance, security features, and reliability features.

Capacity can be increased by using the storage on multiple machines with a unifying file system. Any individual file or database can span several storage devices and machines, eliminating maximum size restrictions often imposed by file systems shipped with operating systems. A unifying file system can also provide a single uniform name space for grid storage. This makes it easier for users to reference data residing in the grid, without regard for its exact location. In a similar way, special database software can “federate” an assortment of individual databases and files to form a larger, more comprehensive database, accessible using database query functions.

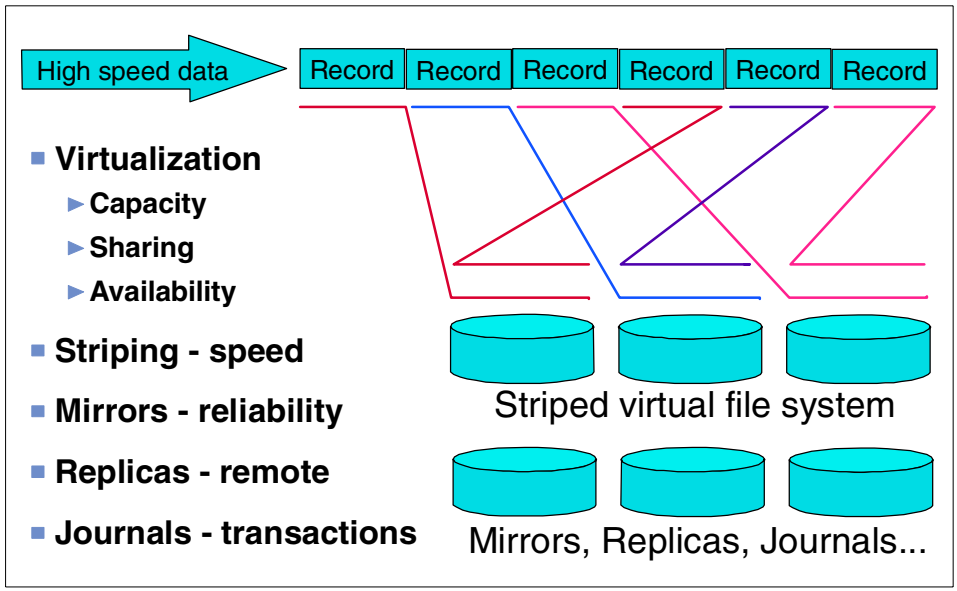

Figure 1-5 Data striping

More advanced file systems on a grid can automatically duplicate sets of data, to provide redundancy for increased reliability and increased performance. An intelligent grid scheduler can help select the appropriate storage devices to hold data, based on usage patterns. Then jobs can be scheduled closer to the data, preferably on the machines directly connected to the storage devices holding the required data.

Data striping can also be implemented by grid file systems, as illustrated in Figure 1-5. When there are sequential or predictable access patterns to data, this technique can create the virtual effect of having storage devices that can transfer data at a faster rate than any individual disk drive. This can be important for multimedia data streams or when collecting large quantities of data at extremely high rates from CAT scans or particle physics experiments, for example.

A grid file system can also implement journaling so that data can be recovered more reliably after certain kinds of failures. In addition, some file systems implement advanced synchronization mechanisms to reduce contention when data is shared and updated by many users.

Communications

The rapid growth in communication capacity among machines today makes grid computing practical, compared to the limited bandwidth available when

distributed computing was first emerging. Therefore, it should not be a surprise that another important resource of a grid is data communication capacity. This includes communications within the grid and external to the grid. Communications within the grid are important for sending jobs and their required data to points within the grid. Some jobs require a large amount of data to be processed and it may not always reside on the machine running the job. The bandwidth available for such communications can often be a critical resource that can limit utilization of the grid.

External communication access to the Internet, for example, can be valuable when building search engines. Machines on the grid may have connections to the external Internet in addition to the connectivity among the grid machines. When these connections do not share the same communication path, then they add to the total available bandwidth for accessing the Internet.

Redundant communication paths are sometimes needed to better handle potential network failures and excessive data traffic. In some cases, higher speed networks must be provided to meet the demands of jobs transferring larger amounts of data. A grid management system can better show the topology of the grid and highlight the communication bottlenecks. This information can in turn be used to plan for hardware upgrades.

Software and licenses

The grid may have software installed that may be too expensive to install on every grid machine. Using a grid, the jobs requiring this software are sent to the particular machines on which this software happens to be installed. When the licensing fees are significant, this approach can save significant expenses for an organization.

Some software licensing arrangements permit the software to be installed on all of the machines of a grid but may limit the number of installations that can be simultaneously used at any given instant. License management software keeps track of how many concurrent copies of the software are being used and prevents more than that number from executing at any given time. The grid job schedulers can be configured to take software licenses into account, optionally balancing them against other priorities or policies.

Special equipment, capacities, architectures, policies

Platforms on the grid will often have different architectures, operating systems, devices, capacities, and equipment. Each of these items represents a different kind of resource that the grid can use as criteria for assigning jobs to machines. While some software may be available on several architectures, for example PowerPC and x86, such software is often designed to run only on a particular

type of hardware and operating system. Such attributes must be considered when assigning jobs to resources in the grid.

In some cases, the administrator of a grid may create a new artificial resource type that is used by schedulers to assign work according to policy rules or other constraints. For example, some machines may be designated to only be used for medical research. These would be identified as having a medical research attribute and the scheduler could be configured to only assign jobs that require machines of the medical research “resource.” Others may be participate in the grid only if they are not used for military purposes. In this situation, jobs requiring a “military resource” would not be assigned to such machines. Of course, the administrators would need to impose a classification on each kind of job through some certification procedure to use this kind of approach.

1.2.2 Jobs and applications

Although various kinds of resources on the grid may be shared and used, they are usually accessed via an executing “application” or “job.” Usually we use the term “application” as the highest level of a piece of work on the grid. However, sometimes the term “job” is used equivalently. Applications may be broken down into any number of individual jobs, as illustrated in Figure 1-6 on page 17. Those, in turn, can be further broken down into “subjobs.” The grid industry uses other terms, such as transaction, work unit, or submission, to mean the same thing as a job.

Jobs are programs that are executed at an appropriate point on the grid. They may compute something, execute one or more system commands, move or collect data, or operate machinery. A grid application that is organized as a collection of jobs is usually designed to have these jobs execute in parallel on different machines in the grid.

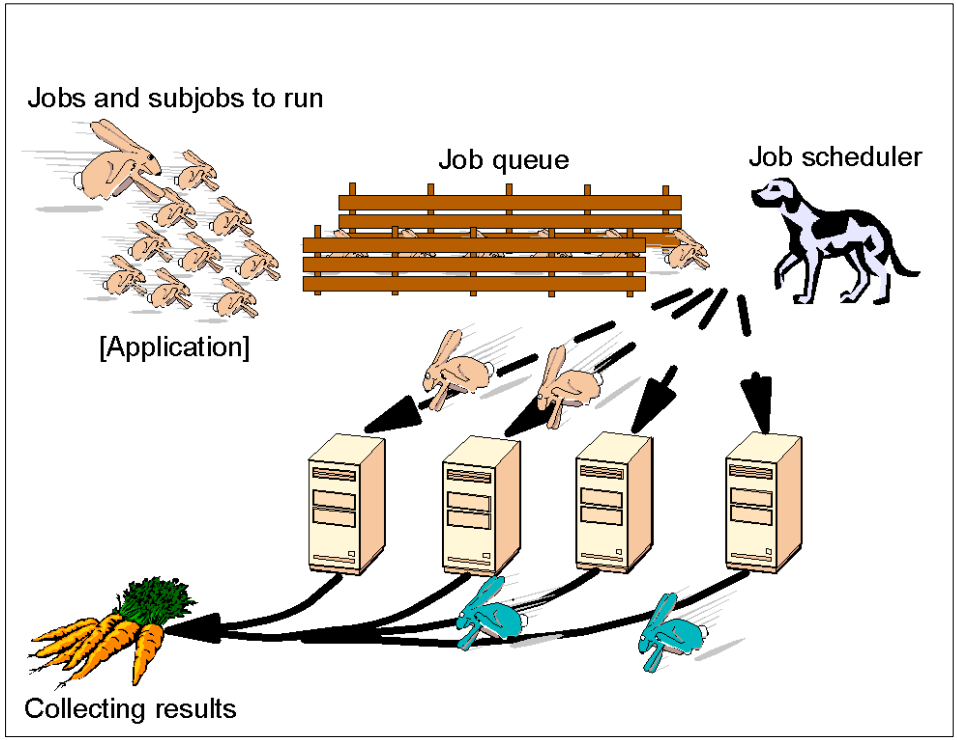

Figure 1-6 An application is one or more jobs that are scheduled to run on the grid

The jobs may have specific dependencies that may prevent them from executing in parallel in all cases. For example, they may require some specific input data that must be copied to the machine on which the job is to run. Some jobs may require the output produced by certain other jobs and cannot be executed until those prerequisite jobs have completed executing. Jobs may spawn additional subjobs, depending on the data they process. This work flow can create a hierarchy of jobs and subjobs. Finally, the results of all of the jobs must be collected and appropriately assembled to produce the ultimate answer for the application.

1.2.3 Scheduling, reservation, and scavenging

The grid system is responsible for sending a job to a given machine to be executed. In the simplest of grid systems, the user may select a machine suitable for running his job and then execute a grid command that sends the job to the selected machine. More advanced grid systems would include a job “scheduler” of some kind that automatically finds the most appropriate machine on which to run any given job that is waiting to be executed. Schedulers react to current

availability of resources on the grid. The term “scheduling” is not to be confused with “reservation” of resources in advance to improve the quality of service. Sometimes the term “resource broker” is used in place of “scheduler,” but this term implies that some sort of bartering capability is factored into scheduling.

In a “scavenging” grid system, any machine that becomes idle would typically report its idle status to the grid management node. This management node would assign to this idle machine the next job which is satisfied by the machine’s resources. Scavenging is usually implemented in a way that is unobtrusive to the normal machine user. If the machine becomes busy with local non-grid work, the grid job is usually suspended or delayed. This situation creates somewhat unpredictable completion times for grid jobs, although it is not disruptive to those machines donating resources to the grid.

To create more predictable behavior, grid machines are often “dedicated” to the grid and are not preempted by outside work. This enables schedulers to compute the approximate completion time for a set of jobs, when their running characteristics are known.

As a further step, grid resources can be “reserved” in advance for a designated set of jobs. Such reservations operate much like a calendaring system used to reserve conference rooms for meetings. This is done to meet deadlines and guarantee quality of service. When policies permit, resources reserved in advance could also be scavenged to run lower priority jobs when they are not busy during a reservation period, yielding to jobs for which they are reserved. Thus, various combinations of scheduling, reservation, and scavenging can be used to more completely utilize the grid.

Scheduling and reservation is fairly straightforward when only one resource type, usually CPU, is involved. However, additional grid optimizations can be achieved by considering more resources in the scheduling and reservation process. For example, it would be desirable to assign executing jobs to machines nearest to the data that these jobs require. This would reduce network traffic and possibly reduce scalability limits. Optimal scheduling, considering multiple resources, is a difficult mathematics problem. Therefore, such schedulers may use heuristics. These heuristics are rules that are designed to improve the probability of finding the best combination of job schedules and reservations to optimize throughput or any other metric.

1.2.4 Intragrid to intergrid

There have been attempts to formulate a precise definition for what a “grid” is. In fact, the concept of grid computing is still evolving and most attempts to define it precisely end up excluding implementations that many would consider to be grids. We will be pragmatic and not claim to make any definitive descriptions of

what a grid is and is not. Therefore, the following descriptions of various kinds of “grids” must be taken loosely.

Grids can be built in all sizes, ranging from just a few machines in a department to groups of machines organized as a hierarchy spanning the world. In this section, we will describe some examples in this range of grid system topologies.

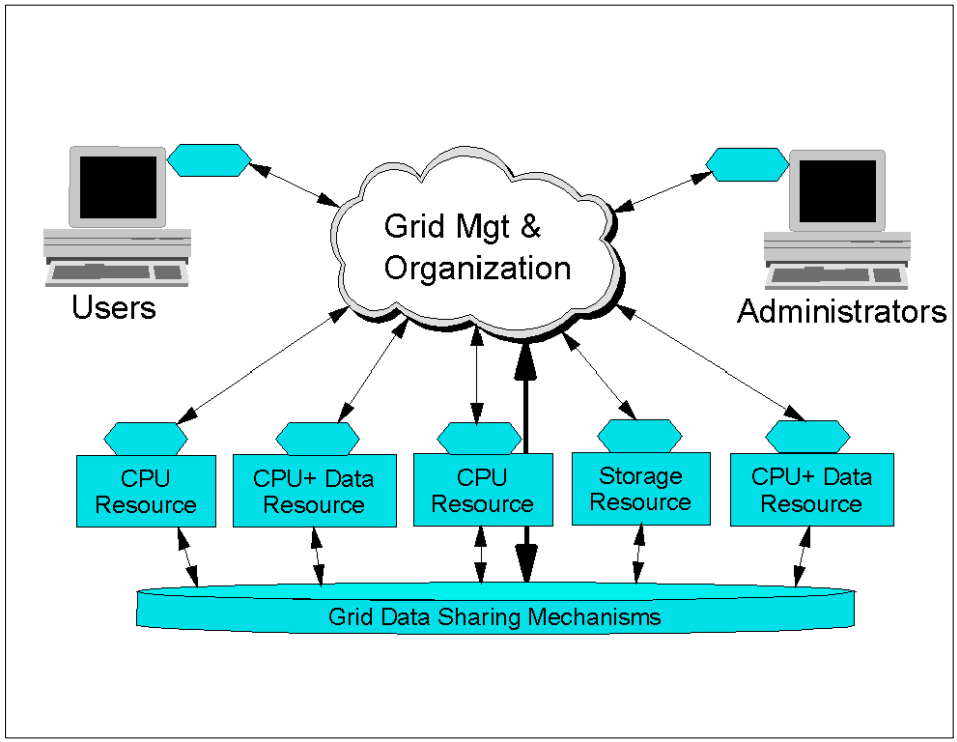

Figure 1-7 A simple grid

As presented in Figure 1-7, the simplest grid consists of just a few machines, all of the same hardware architecture and same operating system, connected on a local network. This kind of grid uses homogeneous systems so there are fewer considerations and may be used just for experimenting with grid software. The machines are usually in one department of an organization, and their use as a grid may not require any special policies or security concerns. Because the machines have the same architecture and operating system, choosing application software for these machines is usually simple. Some people would call this a “cluster” implementation rather than a “grid.”

The next progression would be to include heterogeneous machines. In this configuration, more types of resources are available. The grid system is likely to include some scheduling components. File sharing may still be accomplished

using networked file systems. Machines participating in the grid may include ones from multiple departments but within the same organization. Such a grid is also referred to as an “intragrid.”

As the grid expands to many departments, policies may be required for how the grid should be used. For example, there may be policies for what kinds of work is allowed on the grid and at what times. There may be a prioritization by department or by kinds of applications that should have access to grid resources. Also, security becomes more important as more organizations are involved. Sensitive data in one department may need to be protected from access by jobs running for other departments. Dedicated grid machines may be added to increase the quality of service for grid computing, rather than depending entirely on scavenged resources.

The grid may grow geographically in an organization that has facilities in different cities. Dedicated communications’ connections may be used among these facilities and the grid. In some cases, VPN tunneling or other technologies may be used over the Internet to connect the different parts of the organization. Security increases in importance once the bounds of any given facility are traversed. The grid may grow to be hierarchically organized to reduce the contention implied by central control, increasing scalability.

Over time, as illustrated in Figure 1-8 on page 21, a grid may grow to cross organization boundaries, and may be used to collaborate on projects of common interest. This is known as an “intergrid.” The highest levels of security are usually required in this configuration to prevent possible attacks and spying. The intragrid offers the prospect for trading or brokering resources over a much wider audience. Resources may be purchased as a utility from trusted suppliers.

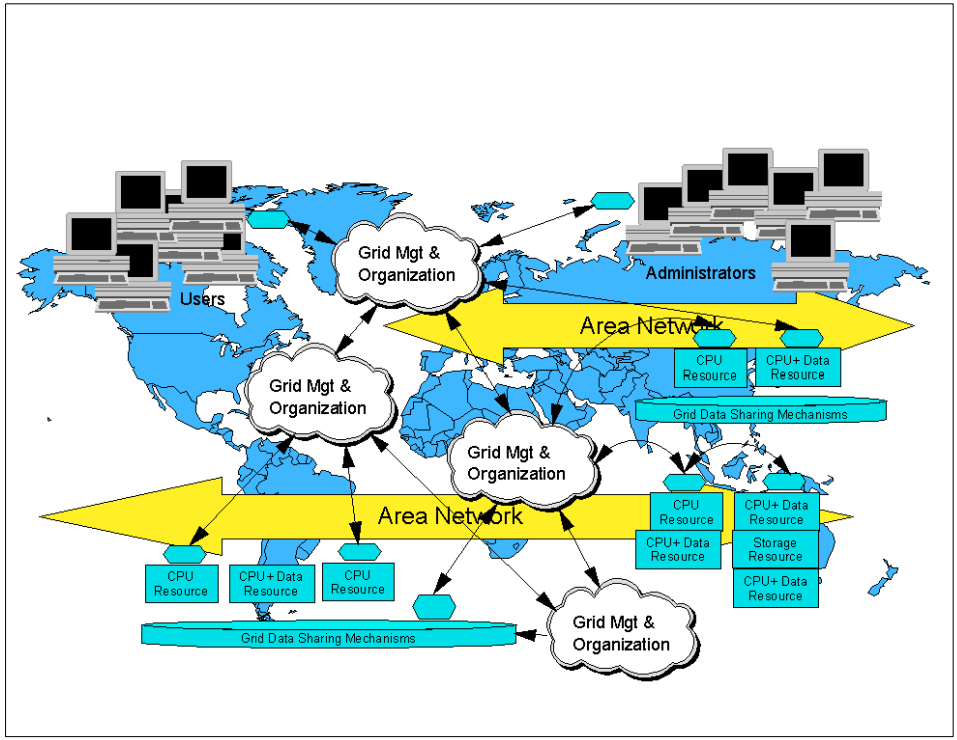

Figure 1-8 A more complex "intergrid"

1.3 Grid construction

An *ad hoc* grid may be installed by a few programmers in their spare time, but as the grid grows, and as users become more dependent on it for mission-critical work, a degree of planning is essential. It is best to understand the organization's requirements and choose grid technologies that best fit these requirements. This section discussed some of the planning considerations and grid components that address the requirements.

1.3.1 Deployment planning

The use of a grid is often born from a need for increased resources of some type. One often looks to their neighbor who may have excess capacity in the particular resource. One of the first considerations is the hardware available and how it is connected via a LAN or WAN. Next, an organization may want to add additional hardware to augment the capabilities of the grid. It is important to understand the

applications to be used on the grid. Their characteristics can affect the decisions of how to best choose and configure the hardware and its data connectivity.

Security

Security is a much more important factor in planning and maintaining a grid than in conventional distributed computing, where data sharing comprises the bulk of the activity. In a grid, the member machines are configured to execute programs rather than just move data. This makes an unsecured grid potentially fertile ground for viruses and Trojan horse programs. For this reason, it is important to understand exactly which components of the grid must be rigorously secured to deter any kind of attack. Furthermore, it is important to understand the issues involved in authenticating users and properly executing the responsibilities of a Certificate Authority.

Organization

The technology considerations are important in deploying a grid. However, organizational and business issues can be equally important. It is important to understand how the departments in an organization interact, operate, and contribute to the whole. Often, there are barriers built between departments and projects to protect their resources in an effort to increase the probability of timely success. However, by rethinking some of these relationships, one can find that more sharing of resources can sometimes benefit the entire organization better. For example, a project that finds itself behind schedule and over budget may not be able to afford the resources required to solve the problem. A grid would give such projects an added measure of safety, providing an extra margin of resource capacity needed to finish the project. Similarly, a project in its early stages, when computing resources are not being fully utilized, may be able to donate them to other projects in need. A grid also offers the ability for the organization's management to see the bigger priority picture and react more quickly in shifting resource utilization, priorities, and policies.

1.3.2 Grid software components

This section presents some of the key components that must be discussed before designing a grid computing architecture.

Management components

Any grid system has some management components. First, there is a component that keeps track of the resources available to the grid and which users are members of the grid. This information is used primarily to decide where grid jobs should be assigned.

Second, there are measurement components that determine both the capacities of the nodes on the grid and their current utilization rate at any given time. This information is used to schedule jobs in the grid. Such information is also used to determine the health of the grid, alerting personnel to problems such as outages, congestion, or overcommitment. This information is also used to determine overall usage patterns and statistics, as well as to log and account for usage of grid resources.

Third, advanced grid management software can automatically manage many aspects of the grid. This is known as autonomic computing, or “recovery oriented computing.” This software would automatically recover from various kinds of grid failures and outages, finding alternative ways to get the workload processed.

Donor software

Each machine contributing resources typically needs to enroll as a member of the grid and install some software that manages the grid’s use of its resources. Usually, some sort of identification and authentication procedure must be performed before a machine can join the grid. A Certificate Authority can be used to establish the identity of the donor machine as well as the users and the grid itself.

Some grid systems provide their own login to the grid while others depend on the native operating systems for user authentication. In the latter case, a user ID mapping system may be needed to match the user’s rights properly on different machines. This typically is manually maintained by a grid administrator. He determines which user ID a given user may possess on each grid machine and enters these IDs in a protected database or registry. In this way, when grid jobs are submitted to different machines for a user, the proper local machine user ID is used for determining the users rights.

In some grid systems, it is possible to join the grid without any special authentication. And in others, it is possible for any user to submit jobs to the grid. Such systems may be convenient to set up, but should be discouraged in larger deployments due to the serious security problems that they would open up.

The grid system makes information about the newly added resources available throughout the grid. The donor machine will usually have some sort of monitor that determines or measures how busy the machine is and the rate or amount of resources utilized. This information is “bubbled up” to the management software of the grid and used to schedule use of those resources accordingly. In a scavenging system, this information tells the grid management software when the machine is idle and available for work.

Most importantly, the software installed on a given machine can accept an executable job from the grid management system and execute it. A user

somewhere on the grid submits a job for execution on the grid. The grid management software must communicate with the grid donor software to send the job there. The donor grid software must be able to receive the executable file or select the proper one from copies pre-installed on the donor machine. The software is executed and the output is sent back to the requester. More advanced implementations can dynamically adjust the priority of a running job, suspend it and resume running it later, or checkpoint it with the possibility of resuming its execution on a different machine. These kinds of actions may be necessary to respond to load balancing problems or priority or policy changes in the grid.

Submission software

Usually any member machine of a grid can be used to submit jobs to the grid and initiate grid queries. However, in some grid systems, this function is implemented as a separate component installed on “submission nodes” or “submission clients.” When a grid is built using dedicated resources rather than scavenged resources, separate submission software is usually installed on the user’s desktop or workstation.

Distributed grid management

Larger grids may have a hierarchical or other type of organizational topology usually matching the connectivity topology. That is, machines locally connected together with a LAN form a “cluster” of machines. The grid may be organized in a hierarchy consisting of clusters of clusters. The work involved in managing the grid is distributed to increase the scalability of the grid. The collection and grid operation and resource data as well as job scheduling is distributed to match the topology of the grid. For example, a central job scheduler will not schedule a submitted job directly to the machine which is to execute it. Instead the job is sent to a lower level scheduler which handles a set of machines (or further clusters). The lower level scheduler handles the assignment to the specific machine. Similarly, the collection of statistical information is distributed. Lower level clusters receive activity information from the individual machines, aggregate it, and send it to higher level management nodes in the hierarchy.

Schedulers

Most grid systems include some sort of job scheduling software. This software locates a machine on which to run a grid job that has been submitted by a user. In the simplest cases, it may just blindly assign jobs in a round-robin fashion to the next machine matching the resource requirements. However, there are advantages to using a more advanced scheduler.

Some schedulers implement a job priority system. This is sometimes done by using several job queues, each with a different priority. As grid machines become available to execute jobs, the jobs are taken from the highest priority queues first.

Policies of various kinds are also implemented using schedulers. Policies can include various kinds of constrains on jobs, users, and resources. For example, there may be a policy that restricts grid jobs from executing at certain times of the day.

Schedulers usually react to the immediate grid load. They use measurement information about the current utilization of machines to determine which ones are not busy before submitting a job. Schedulers can be organized in a hierarchy. For example, a meta-scheduler may submit a job to a cluster scheduler or other lower level scheduler rather than to an individual machine.

More advanced schedulers will monitor the progress of scheduled jobs managing the overall work-flow. If the jobs are lost due to system or network outages, a good scheduler will automatically resubmit the job elsewhere. However, if a job appears to be in an infinite loop and reaches a maximum timeout, then such jobs should not be rescheduled. Typically, jobs have different kinds of completion codes, some of which are suitable for re-submission and some of which are not.

Reserving resources on the grid in advance is accomplished with a “reservation system.” It is more than a scheduler. It is first a calendar based system for reserving resources for specific time periods and preventing any others from reserving the same resource at the same time. It also must be able to remove or suspend jobs that may be running on any machine or resource when the reservation period is reached.

Communications

A grid system may include software to help jobs communicate with each other. For example, an application may split itself into a large number of subjobs. Each of these subjobs is a separate job in the grid. However, the application may implement an algorithm that requires that the subjobs communicate some information among them. The subjobs need to be able to locate other specific subjobs, establish a communications connection with them, and send the appropriate data. The open standard Message Passing Interface (MPI) and any of several variations is often included as part of the grid system for just this kind of communication.

Observation, management, and measurement

We mentioned above the schedulers react to current loads on the grid. Usually, the donor software will include some tools that measure the current load and activity on a given machine using either operating system facilities or by direct measurement. This software is sometimes referred to as a “load sensor.” Some grid systems provide the means for implementing custom load sensors for other than CPU or storage resources.

Such measurement information is useful not only for scheduling, but also for discovering overall usage patterns in the grid. The statistics can show trends which may signal the need for additional hardware. Also, measurement information about specific jobs can be collected and used to better predict the resource requirements of that job the next time it is run. The better the prediction, the more efficiently the grid's workload can be managed.

The measurement information can also be saved for accounting purposes, or to form the basis for grid resource brokering, or to manage priorities more fairly. The information can also be displayed in various forms to better visualize grid activity and utilization.

1.4 Using a grid: A user's perspective

This section describes the typical usage activities in using the grid from an user's perspective.

1.4.1 Enrolling and installing grid software

A user first enrolls as a grid user, and installs the provided grid software on his own machine. He may optionally enroll his machine as a donor on the grid.

Enrolling in the grid may require authentication for security purposes. The user positively establishes his identity with a Certificate Authority. This should not be done solely via the Internet. The Certificate Authority must take steps to assure that the user is in fact who he claims to be. The Certificate Authority makes a special certificate available to software needing to check the true identity of a grid user and his grid requests. Similar steps may be required to identify the donating machine. The user has the responsibility of keeping his grid credentials secure.

Once the user and/or machine are authenticated, the grid software is provided to the user for installing on his machine for the purposes of using the grid as well as donating to the grid. This software may be automatically preconfigured by the grid management system to know the communication address of the management nodes in the grid and user or machine identification information. In this way, the installation may be a one click operation with a minimum of interaction required on the part of the user. In less automated grid installations, the user may be asked to identify the grid's management node and possibly other configuration information. He may choose to limit the resources donated to the grid, the times that his machine is usable by the grid, and other policy related constraints. The user may also need to inform the grid administrator which user IDs are his on other machines that exist on the grid.

1.4.2 Logging onto the grid

To use the grid, most grid systems require the user to log on to a system using a user ID that is enrolled in the grid. Other grid systems may have their own grid login ID separate from the one on the operating system. A grid login is usually more convenient for grid users. It eliminates the ID matching problems among different machines. To the user, it makes the grid look more like one large virtual computer rather than a collection of individual machines. Globus, for example, implements a proxy login model that keeps the user logged in for a specified amount of time, even if he logs off and back on the operating system and even if the machine is rebooted.

Once logged on, the user can query the grid and submit jobs. Some grid implementations permit some query functions if the user is not logged into the grid or even if the user is not enrolled in the grid.

1.4.3 Queries and submitting jobs

The user will usually perform some queries to check to see how busy the grid is, to see how his submitted jobs are progressing, and to look for resources on the grid. Grid systems usually provide command line tools as well as graphical user interfaces (GUIs) for queries. Command line tools are especially useful when the user wants to write a script that automates a sequence of actions. For example, the user might write a script to look for an available resource, submit a job to it, watch the progress of the job, and present the results when the job has finished.

Job submission usually consists of three parts, even if there is only one command required. First, some input data and possibly the executable program or execution script file are sent to the machine to execute the job. Sending the input is called “staging the input data.” Alternatively, the data and program files may be pre-installed on the grid machines or accessible via a mountable networked file system. When the grid consists of heterogeneous machines, there may be multiple executable program files, each compiled for the different machine platforms on the grid. A nice feature provided by some grid systems is to register these multiple versions of the program so that the grid system can automatically choose a correctly matching version to the grid machine that will run the program. Some grid technologies require that the program and input data be first processed or “wrappered” in some way by the grid system. This may be done to add protective execution controls around the application or just to simply collect all of the data files into one.

Second, the job is executed on the grid machine. The grid software running on the donating machine executes the program in a process on the user’s behalf. It may use a common user ID on the machine or it may use the user’s own user ID, depending on which grid technology is used. Some grid systems implement a

protective “sandbox” around the program so that it cannot cause any disruption to the donating machine if it encounters a problem during execution. Rights to access files and other resources on the grid machine may be restricted.

Third, the results of the job are sent back to the submitter. In some implementations, intermediate results can be viewed by the user who submitted the job. In some grid technologies that do not automatically stage the output data back to the user, the results must be explicitly sent to the user, perhaps using a networked file system.

Scripts are also useful for submitting a series of jobs, for a parameter space application, for example. Some computation problems consist of a search for the desired result based on some input parameters. The goal is to find the input parameters that produce the best desired result. For each input parameter, a separate job is executed to find the result for that value. The whole application consists of many such jobs, which explore the results for a large number of input parameter values. Scripts are usually used to launch the many subjobs, each receiving their own particular parameter values. Parameter inputs can sometimes be more complex than simply a number. Sometimes a different input data set represents the “input parameter.” Scripts help automate the large variety of more complex parameter space study problems. For simpler parameter space inputs, some grid products provide a GUI to submit the series of subjobs, each with different input parameter values.

When there are a large number of subjobs, the work required to collect the results and produce the final result is usually accomplished by a single program, usually running on the machine at the point of job submission. If there are a very large number subjobs required for an application, the work of collecting the results might be distributed as well. For example, the subjob that submits more subjobs to the grid would be responsible for collecting and aggregating the results of the subjobs it spawned.

1.4.4 Data configuration

The data accessed by the grid jobs may simply be staged in and out by the grid system. However, depending on its size and the number of jobs, this can potentially add up to a large amount of data traffic. For this reason, some thought is usually given on how to arrange to have the minimum of such data movement on the grid.

For example, if there will be a very large number of sub-jobs running on most of the grid systems for an application that will be repeatedly run, the data they use may be copied to each machine and reside until the next time the application runs. This is preferable to using a networked file system to share this data, because in such a file system, the data would be effectively moved from a central

location every time the application is run. This is true unless the file system implements a caching feature or replicates the data automatically.

There are many considerations in efficiently planning the distribution and sharing of data on a grid. This type of analysis is necessary for large jobs to better utilize the grid and not create unnecessary bottlenecks.

1.4.5 Monitoring progress and recovery

The user can query the grid system to see how his application and its subjobs are progressing. When the number of subjobs becomes large, it becomes too difficult to list them all in a graphical window. Instead, there may simply be a one large bar graph showing some averaged progress metric. It becomes more difficult for the user to tell if any particular subjob is not running properly.

A grid system, in conjunction with its job scheduler, often provides some degree of recovery for subjobs that fail. A job may fail due to a:

- Programming error: The job stops part way with some program fault.
- Hardware or power failure: The machine or devices being used stop working in some way.
- Communications interruption: A communication path to the machine has failed or is overloaded with other data traffic.
- Excessive slowness: The job might be in an infinite loop or normal job progress may be limited by another process running at a higher priority or some other form of contention.

It is not always possible to automatically determine if the reason for a job's failure is due to a problem with the design of the application or if it is due to failures of various kinds in the grid system infrastructure. Schedulers are often designed to categorize job failures in some way and automatically resubmit jobs so that they are likely to succeed, running elsewhere on the grid. In some systems, the user is informed about any job failures and the user must decide whether to issue a command to attempt to rerun the failed jobs.

Grid applications can be designed to automate the monitoring and recovery of their own subjobs using functions provided by the grid system software application programming interfaces (APIs).

1.4.6 Reserving resources

To improve the quality of a service, the user may arrange to reserve a set of resources in advance for his exclusive or high priority use. A calendaring system analogy can be used here. Such a reservation system can also be used in

conjunction with planned hardware or software maintenance events, when the affected resource might not be available for grid use.

In a scavenging grid, it may not be possible to reserve specific machines in advance. Instead, the grid management systems may allocate a larger fraction of its capacity for a given reservation to allow for the likelihood of some of the resources becoming unavailable. This must be done in conjunction with tools that have profiled the grid's workload capacity sufficiently to have reliable statistics about the grid's ability to serve the reservation.

1.5 Using a grid: An administrator's perspective

This section describes the typical usage activities in using the grid from an administrator's perspective.

1.5.1 Planning

The administrator should understand the organization's requirements for the grid to better choose the grid technologies that satisfy those requirements. The following sections briefly describe the steps the administrator may take to manage the grid. It is suggested that one should start by deploying a small grid first, to learn about its installation and management, before having to confront more complicated issues involved with a large grid.

1.5.2 Installation

First, the selected grid system must be installed on an appropriately configured set of machines. These machines should be connected using networks with sufficient bandwidth to other machines on the grid. Of prime importance is understanding the fail-over scenarios for the given grid system so that the grid can continue operating even if any of the management machines fails in some way. Machines should be configured and connected to facilitate recovery scenarios. Any critical databases or other data essential for keeping track of the jobs in the grid, members of the grid, and machines on the grid should have suitable backups. Furthermore, public key certificates must be backed up and the private keys must be held in a highly secured place inaccessible by anyone else.

After installation, the grid software may need to be configured for the local network address and IDs. The administrator will usually require root access to the machines managing the grid. In some grid systems, he will also need root access to the donor machines be required to install the software on those as well. The software to be installed on the donor machines may need to be

customized so that it can find the grid management machines automatically and include pre-installed public keys for the grid. This software may be provided to potential donors on an FTP or equivalent server or be made available on physical media.

Once, the grid is operational, there may be application software and data that should be installed on donor machines as well. This software may have specific licensing restrictions that should be understood and adhered to. Some grid systems include tools to assist with grid-wide license management. This can both help in following the rules of the licenses and most efficiently exploit those licenses.

1.5.3 Managing enrollment of donors and users

An ongoing task for the grid administrator is to manage the members of the grid, both the machines donating resources and the users. Users may be further organized as project groups. The administrator is responsible for controlling the rights of the users in the grid. Donor machines may have access rights that require management as well. Grid jobs running on donor machines may be executed under a special grid user ID on behalf of the users submitting the jobs. The rights of these grid user IDs must be properly set so that grid jobs do not allow access to parts of the donor machine to which the users are not entitled.

As users join the grid, their identity must be positively established and entered in the Certificate Authority. The user and his certificate credentials must be added to the user list using the software appropriate for the grid system deployed. In some cases, the administrator must propagate the user information to several or all grid machines. Also, when the grid system depends primarily on the operating system for user login, the administrator may need to add entries to map the grid user to specific operating system user IDs on the donor machines.

Similar enrollment activity is usually required to enroll donor machines into the grid. The machine's identity is established and registered with the Certificate Authority. The administrator of the grid must have an agreement with the administrator of the donor machine about user IDs, software, access rights, and any policy restrictions. The administrator must enter the machine's identification credentials, addresses, and resource characteristics using the appropriate software for enrolling the donor machine into the grid. In some cases, the administrator may need to manually propagate this information to other machines in the grid.

Corresponding procedures for removing users and machines must be executed by the administrator.

1.5.4 Certificate authority

It is critical to ensure the highest levels of security in a grid because the grid is designed to execute code and not just share data. Thus, it can be fertile ground for viruses, Trojan horses, and other attacks if the grid system is compromised in any way. The Certificate Authority is one of the most important aspects of maintaining strong grid security. An organization may choose to use an external Certificate Authority or operate one itself. You must be able to trust the Certificate Authority to strictly adhere to its responsibilities.

The primary responsibilities of a Certificate Authority are:

- Positively identify entities requesting certificates
- Issuing, removing, and archiving certificates
- Protecting the Certificate Authority server
- Maintaining a namespace of unique names for certificate owners
- Serve signed certificates to those needing to authenticate entities
- Logging activity

Briefly, a Certificate Authority is based on the public key encryption system. In this system, keys are generated in pairs, a public key and a private key. Either one can be used to encrypt some data such that the other is needed to decrypt it. The private key is guarded by the owner and never revealed to anyone. The public one is given to anyone needing it. A Certificate Authority is used to hold these public keys and to guarantee who they belong to. When a user uses his private key to encrypt something, the receiver uses the corresponding public key to decrypt it. The receiver knows that only that user's public key can decrypt the message correctly. However, anyone could intercept this message and decrypt it because anyone can get the originator's public key. If the originator instead doubly encrypts the message with his private key and the intended recipient's public key, a secure communication link is formed. The receiver uses his private key to decrypt the message and then uses the sender's public key for the second decryption. Now the recipient knows that if the message decrypts properly, then only the sender could have sent it and furthermore, the sender knows that only the intended receiver can decrypt it. The beauty of all of this is that nobody had to securely carry an encryption key from the sender to the receiver, as must be done for conventional encryption systems, and any tampering with the communication is revealed. A similar exchange is used to get anyone's public key from the Certificate Authority, so that the user knows that he has received an unaltered public key for the desired user.

1.5.5 Resource management

Another responsibility of the administrator is to manage the resources of the grid. This includes setting permissions for grid users to use the resources as well as tracking resource usage and implementing a corresponding accounting or billing system. Usage statistics are useful in identifying trends in an organization that may require the acquisition of additional hardware, reduction in excess hardware to reduce costs, and adjustments in priorities and policies to achieve utilization that is fairer or better achieves the overall goals of an organization.

Some grid components, usually job schedulers, have provisions for enforcing priorities and policies of various kinds. It is the responsibility of the administrator to configure these to best meet the goals of the overall organization. Software license managers can be used in a grid setting to control the proper utilization. These may be configured to work with job schedulers to prioritize the use of the limited licenses.

1.5.6 Data sharing

For small grids, the sharing of data can be fairly easy, using existing networked file systems, databases, or standard data transfer protocols. As a grid grows and the users become dependent on any of the data storage repositories, the administrator should consider procedures to maintain backup copies and replicas to improve performance. All of the resource management concerns apply to data on the grid.

1.6 Using a grid: An application developer's perspective

Grid applications can be categorized in one of the following three categories:

- Applications that are not enabled for using multiple processors but can be executed on different machines.
- Applications that are already designed to use the multiple processors of a grid setting.
- Applications that need to be modified or rewritten to better exploit a grid.

The latter category is of interest to grid application developers. They will find a need for tools for debugging and measuring the behavior of grid applications. Such grid based tools are still in their infancy. It may be useful for developers to configure a small grid of their own so that they can use debuggers on each machine to control and watch the detailed workings of the applications. Since the debugging process can bypass certain security precautions, it may not always be wise to allow such debugging on a production grid.

Globus is more a developer's toolkit for building grid components rather than a comprehensive grid system. It has the basic components needed to build new facilities to manage grid operations, measurement, repair, and debug grid applications. Tools conforming to the emerging Open Grid Services Architecture (OGSA) interfaces will be usable on various vendor grid systems.

1.7 The present and the future

The Globus Toolkit is a set of tools useful for building a grid. Its strength is a good security model, with a provision for hierarchically collecting data about the grid, as well as the basic facilities for implementing a simple, yet world-spanning grid. Globus will grow over time through the work of many organizations that are extending its capabilities. More information about Globus can be obtained at http://www.globus.org.

Most grid systems include some job schedulers, but as grids span wider areas, there will be a need for more meta-schedulers that can manage variously configured collections of clusters and smaller grids. These schedulers will evolve to better schedule jobs, considering multiple resources rather than just CPU utilization. They will also extend their reach to implement better quality of service, using reservations, redundancy, and history profiles of jobs and grid performance.

Today, grid systems are still at the early stages of providing a reliable, well performing, and automatically recoverable virtual data sharing and storage. We will see products that take on this task in a grid setting, federating data of all kinds, and achieving better performance, integration with scheduling, reliability, and capacity.

Autonomic computing has the goal to make the administrator's job easier by automating the various complicated tasks involved in managing a grid. These include identifying problems in real time and quickly initiating corrective actions before they seriously impair the grid.

OGSA is an open standard at the base of all of these future grid enhancements. OGSA will standardize the grid interfaces that will be used by the new schedulers, autonomic computing agents, and any number of other services yet to be developed for the grid. It will make it easier to assemble the best products from various vendors, increasing the overall value of grid computing. More information about OGSA can be obtained at http://www.globus.org/ogsa.

1.8 What the grid cannot do

A word of caution should be given to the overly enthusiastic. The grid is not a silver bullet that can take any application and run it a 1000 times faster without the need for buying any more machines or software. Not every application is suitable or enabled for running on a grid. Some kinds of applications simply cannot be parallelized. For others, it can take a large amount of work to modify them to achieve faster throughput. The configuration of a grid can greatly affect the performance, reliability, and security of an organization's computing infrastructure. For all of these reasons, it is important for the users to understand how far the grid has evolved today and which features are coming tomorrow or in the distant future.

Part 2

Architecture

2

Application considerations

This chapter describes the considerations that need to be made when evaluating, designing, or converting applications for use in a grid computing environment

2.1 Application considerations

While a grid may offer many advantages, any given application may not necessarily benefit from a grid. For example, a word processor is tightly coupled with a user's interface, and does not consume a large amount of computing resources. Running it on a grid would likely degrade its performance by having to use a remote windowing system and being subject to more potential points of failure. However, other applications may be very suited for exploiting a grid.

The easiest use of the grid is to just run the application somewhere else when your own machine is too busy. Almost any kind of application can be executed in a grid environment this way. You will not see spectacular performance gains unless the machine it runs on is much faster than the machine you usually use. Applications that can be run in a batch mode are the easiest to handle. Applications that need interaction through graphical user interfaces are more difficult to run on a grid, but not impossible. They can use remote graphical terminal support, such as X Windows or other means.

Applications specifically designed to use multiple processors or other federated resources of a grid will benefit most. The following discussion is designed to stimulate analysis, which will show how various factors may help decide whether a given application should be deployed on a grid and what modifications, if any, might be considered.

2.1.1 CPU considerations

If you are considering grid computing, you should examine any applications that consume large amounts of CPU time. The following questions will help determine if such an application can be run on a grid as-is, needs modification, or just cannot benefit.

The most important step in grid enabling an application is to determine whether the calculations can be done in parallel or not. While HPC clusters (High Performance Computing) are sometimes used to handle the execution of applications that can utilize parallel processing, grids provide the ability to run these applications across a set of heterogeneous, geographically disperse set of clusters. Rather than run the application on a single homogenous cluster, the application can take advantage of the larger set of resources in the grid. If the algorithm is such that each computation depends on the prior calculation, then a new algorithm would need to be found. Not all problems can be converted into parallel calculations. As an oversimplified example, let us take the process of adding up a large list of numbers. The simple serial program may be written to start with the sum of zero and then add each of the numbers, one at a time, until the final sum is reached. Here each calculation depends on the prior one.

However, we can observe that the associative property of arithmetic shows us that we could break the list up into seven pieces, for example, with seven separate programs adding up the numbers in each list, and then a final eighth program adding the 7 sums to form the final answer. This is illustrated by Figure 2-1.

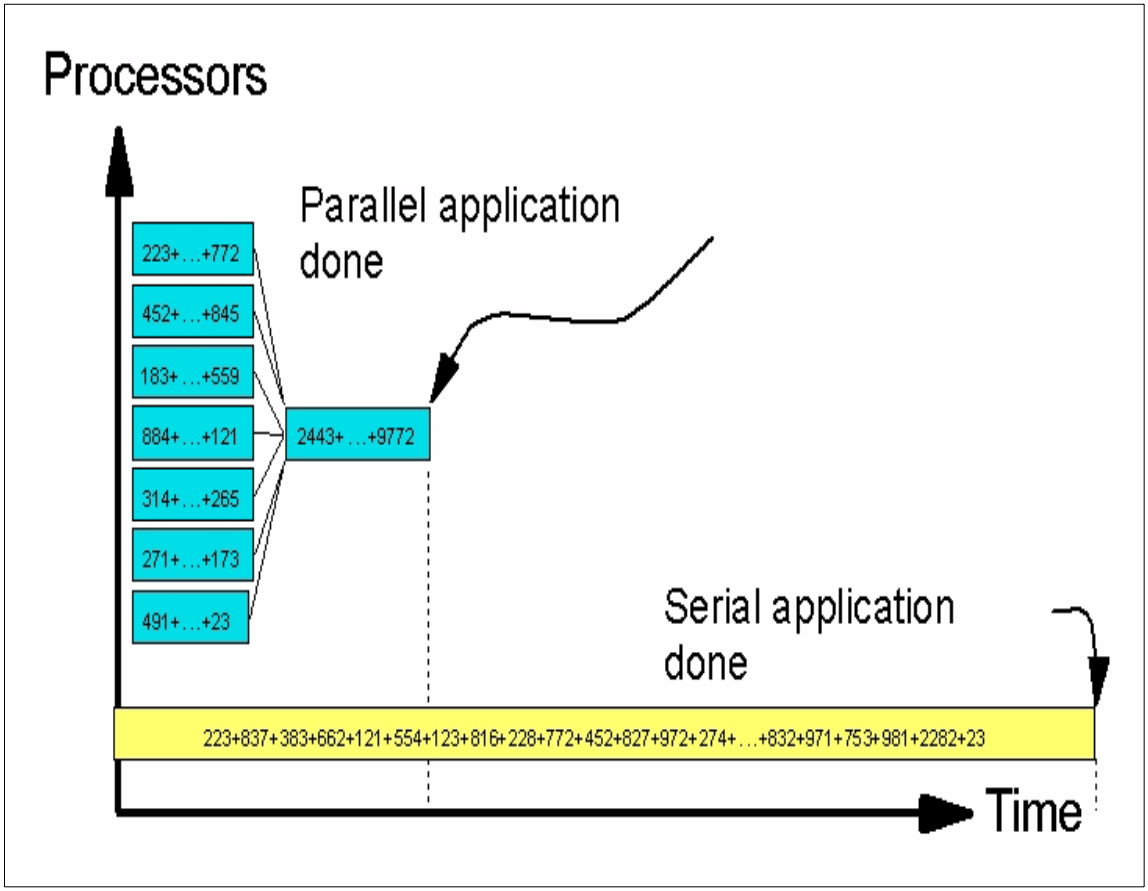

Figure 2-1 Rearranging computations to execute in parallel

On the other hand, some computations cannot be rewritten to execute in parallel. For example, in physics, there are no simple formulas that show where three or more moving bodies in space will be after a specified time when they gravitationally affect each other. These kinds of computations are done by simulating the motions of the bodies, applying Newton's (or Einstein's) laws to small time increments, and computing how the forces and bodies affect each other, given the new position of the objects after each tiny time increment, as illustrated in Figure 2-2 on page 42.

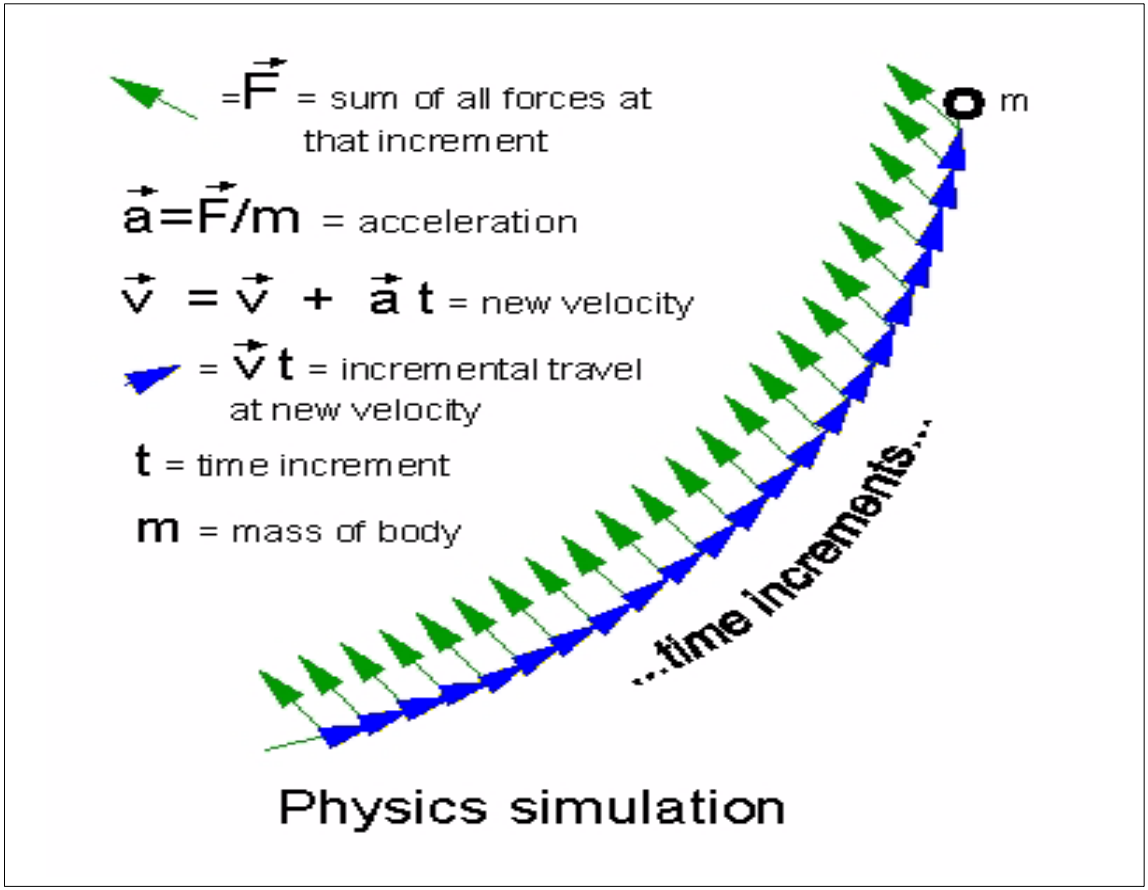

Figure 2-2 Simulation that cannot be made parallel but needs to run many times

This is repeated a great number of times until the desired time is reached. Each computation depends on the prior one. If it did not, then we would have discovered the difficult formula we did not have in the first place. Because the time increments are not infinitely small, after many increments, the small errors start adding up. The final computed position of the objects can be in error, perhaps ultimately causing a spacecraft to crash into a planet instead of going into orbit. To improve accuracy in such computations, we make the time increments much shorter. This increases the number of these increments to be computed and thus the overall computation time. Many simulations suffer from this type of difficulty.

As we saw above, in the list adding example, such computations can be performed in parallel while others, such as the 3-body physics problem, cannot. Often, an application may be a mix of independent computations as well as

dependent computations. One needs to analyze the application to see if there is a way to split some subset of the work. Drawing a program flow graph and a data dependency graph can help in analyzing whether and how an application could be separated into independently running parallel parts.

Going back to the space object example, let us say we are trying to find the correct trajectory to aim a rocket so that it loops around Venus, and then Earth, to reach Jupiter more quickly. We might try calculating to see what happens for a large number of different trajectories, pointing the rocket in slightly different directions and firing the engines for different durations. Each trajectory can be thought of as a separate calculation, and then in the end, a program chooses the best one. Here, we are able to perform work in parallel, even though the underlying computation for a single trajectory may be serial. Applications that consist of a large number of independent subjobs are very suitable for exploiting grid CPU resources. These are sometimes called parameter space searches.

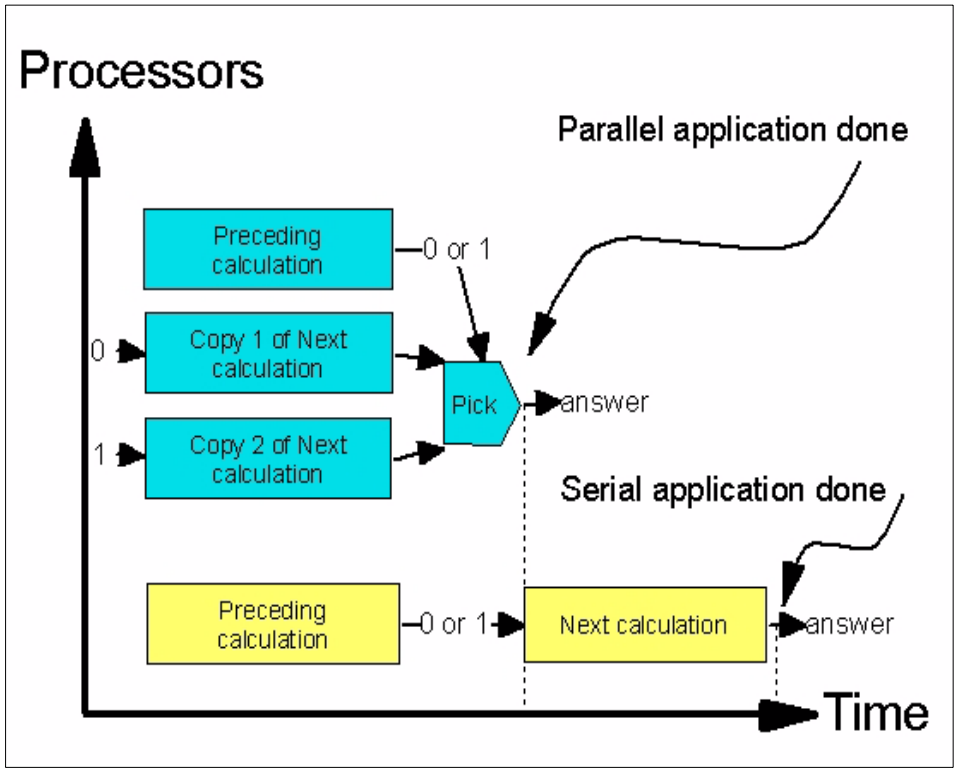

Figure 2-3 Redundant speculative computation to reduce latency

Another approach to reducing data dependency on prior computations is to look for ways to use redundant computations. If the dependency is on a subset of the

prior computations, it may be beneficial just to have each successive computation that needs the results of the prior computation recompute those results instead of waiting for them to arrive from another job. If the dependency is on a computation that has a yes/no answer, perhaps it is better to compute the next calculations for both of the “yes” and “no” cases and throw away the wrong choice when the dependency is finally known, as illustrated in Figure 2-3 on page 43. This technique can be taken to extremes in various ways. For example, for two bits of data dependency, we could make four copies of the next computation with all four possible input values. This can proceed to copies of the next calculation for N bits of data dependency. As N gets large, it quickly becomes too costly to compute all possible computations. However, we may speculate and only perform the copies for the values we guess might be more likely to be correct. If we did not guess the correct one, then we simply end up computing it in series, but if we guessed correctly it saves us overall real time. Here heuristics (rules of thumb) could be developed to make the best possible guesses. Furthermore, there may be many points in the application where we could use the speculative approach and if our guess rate is high enough, there might be an overall improvement in efficiency and parallelism. This same kind of speculative computing is used to improve the efficiency inside CPUs by executing both branches of a condition until the correct one is determined.

In many cases, an application is used to test an array of “what if” input values. For example, in the spacecraft trajectory problem, we might run simulations to see where a spacecraft goes if you alter, in small increments, its starting direction, starting speed, and apply various midcourse corrections. Each of these alternatives can be a separate job running the same simulation application, but with different input values. We call this a parameter space problem. A computation grid is ideally suited for this kind of problem, even though it might not be possible to make any single run more parallel in execution. The parallelism comes from running many separate jobs that cover the parameter space. Some grid products provide tools for simplifying the submission of the many sub-jobs in a parameter space exploration type of application.

Some parameter space problems are finite in nature, and some are infinite or so large that all possible parameter inputs cannot be examined. For these kinds of parameter space problems, it is useful to use additional heuristics to select which parts of the parameter space to try. This may not lead to the absolute best solution, but it may be close enough. The traveling salesman problem can be intractable in this way when there are many cities to be visited. However, various heuristics can be used to get reasonably close to an optimal solution. It may not be worth a month of additional computation to improve the answer from 98% to 99% of the best possible.

It may be acceptable to explore only a small part of the parameter space. One approach is to try a reasonable number of randomly scattered points in the

problem's parameter space first. Then one would try small changes in the parameters around the best points that might lead to a better solution. This technique is useful when the parameter space relates relatively smoothly to changes in the result. By analogy, this can be described as "hill climbing" to find the highest altitude point, in a perpetually fog shrouded region of land, on which to build a television broadcast antenna. You would put a set of people at random on the terrain. Then each would climb to the highest point near them. Whomever reached the highest point would then be declared to have found the highest hill in the land. They may not have found the absolute highest point if nobody started near that point, but they will probably find the nearly highest hill or one that is sufficient for their antenna tower. This kind of technique is useful when there are too few people and too many hills to visit all of them.

Often, mathematical calculations are commutative, associative, or linear in some way. The simple adding of a list of numbers example illustrates this. By altering some potentially unimportant rules in the computations involved in a calculation, we may be able to break the ordering requirement and thus make it possible to execute more of the application in parallel. For example, in a bank account, deposits and withdrawals are serially calculated and if the account ever goes negative, then the transaction may be rejected, a fine may be imposed, or the account may be frozen. If, however, the bank changes its rules and says that the account must simply be positive at the end of the day. Then withdrawals processed before the deposits would not cause a problem and all of these calculations could be broken up into separate parallel running jobs.

Many times, an application that was written for a single processor may not be organized or use algorithms or approaches that are suitable for splitting into parallel subcomputations. An application may have been written in a way that makes it most efficient on a single processor machine. However, there may be other methods or algorithms that are not as efficient, yet may be much more amenable to being split into independently running subcomputations. A different algorithm may "scale" better because it can more efficiently use larger and larger numbers of processors. Thus, another approach for grid enabling an application is to revisit the choices made when the application was originally written. Some of the discarded approaches may be better for grid use.

How you go about solving a problem may be quite different, depending on whether it is unique to be solved only once versus being solved repeatedly with different inputs. One might use a less efficient but more straightforward technique if the problem is only to be solved once, reducing debug time and making good use of a grid's ability to absorb momentary peaks of activity. On the other hand, if it is a one time problem, but is going to take a year of execution, more thought should be put into the problem before proceeding. The following are some additional things to think about.

Is there any part of the computation that would be performed more than once using the same data? If so, and if that computation is a significant portion of the overall work, it may be useful to save the results of such computations.

If we find that an application performs some sets of computations on the same input data every time it is run, produces the same output data, and takes a significant amount of time computing this output, how much output data would need to be saved to avoid the computation the next time? If there is a very large amount of output data, it may be prohibitive to save this data. Perhaps there are a large number of similar computations that might be saved. Even if any one computation's results does not represent a large amount of data, the aggregate for all of them might. One needs to consider this time-space trade-off for the application. One could presumably save space and time by only saving the results for the most frequently occurring situations. For example, in world class chess playing programs, the opening positions of the game of chess are usually stored in a database containing the best move to take in each such position. This information can be precomputed to a large extent and can save large amounts of computation time during a chess tournament. However, the number of possible chess board positions increases very rapidly with more moves into the game, so only the early move positions of the game or the end-game moves when there are few pieces left, are precomputed and saved.

In a distributed application, partial results or data dependencies may be met by communicating among subjobs. That is, one job may compute some intermediate result and then transmit it to another job in the grid. If possible, one should consider whether it might be any more efficient to simply recompute the intermediate result at the point where it is needed rather than waiting for it from another job. One should also consider the transfer time from another job, versus retrieving it from a database of prior computations.

2.1.2 Data considerations

When splitting applications for use on a grid, it is important to consider the amounts of data that are needed to be sent to the node performing a calculation and the time required to send it. After some analysis, one might discover that a different design would be better. If the application can be split into small work units requiring little input data and producing small amounts of output data, that would be most ideal. The data in this kind of case is said to be "staged" to the node doing the work. Sending this data along with the executable file to the grid node doing the work is part of the function of most grid systems. However, in many applications, larger amounts of input and/or output data are involved and this can cause complications and inefficiencies.

When the grid application is split into subjobs, often the input data is a large fixed set of data. This offers the opportunity to share this data rather than staging the

entire set with each subjob. However, one must consider that even with a shared mountable file system, the data is being sent over the network. The goal is to locate the shared data closer to the jobs that need the data. If the data is going to be used more than once, it could be replicated to the degree that space permits.

If more than one copy of the data is stored in the grid, it is important to arrange for the subjobs to access the nearest copy per the configuration of the network. This highlights the need for an information service within the grid to track this form of data awareness. Furthermore, one must be careful that the network does not become the bottleneck for such a grid application. If each subjob processes the data very quickly and is always waiting for more data to arrive, then sharing may not be the best model if the network data transfer speed to each subjob does not at least match disk speeds.

Shared data may be fixed or changing. For example, a database may contain the latest known gene sequences and be constantly growing. However, applications using this data may not need the latest gene sequence data the instant that it is available. This makes it easier and more efficient to share such a database because the updates to it can be batched and processed at off-peak usage times rather than contending with concurrent access by applications. Furthermore, if more than one copy of this data exists, and all of the copies do not need to be simultaneously updated, this improves performance because all applications using the data would not need to be stopped while updating the data. Only those accessing a particular copy would need to be stopped or temporarily paused.

When a file or a database is updated, jobs cannot simultaneously read the portion of the file concurrently being updated by another job. Locking or synchronizing primitives are typically built into the files system or database to automatically prevent this. Otherwise, the application might read partially updated data, perhaps receiving a combination of old and new data.

In some shared data situations, updates must not be delayed. For example, if the subjobs are processing financial transactions, they must be immediately updated in the master balances database. Furthermore, if there are copies of this database elsewhere, they must all be updated with each new item simultaneously. A number of scaling issues come into play here. There can be a large amount of data synchronization communications among jobs and databases. The synchronization primitives can become bottlenecks in overall grid performance. It is important to consider how the database activity can be partitioned so that there is less interference among the parts and thus less potential synchronization contention among those parts.

Applications that access the data they need serially are more predictable, so various techniques can be used to improve their performance on the grid. If each subjob needs to access all of the data, then shared copies might be desirable. Multiple copies of the data should be considered if bringing the data closer to the

nodes running the subjobs would help. If each part of the data is examined only once, then copies may not be desirable. However, if the access is serial, some of the retrieval time can be overlapped with processing time. There could be a thread retrieving the data that will be needed next while the data already retrieved is being processed. This can even apply to randomly accessed data, if there is the ability to do some prediction of which portions of data will be needed next.

One of the most difficult problems with duplicating rapidly changing databases is keeping them in synchronization. The first step is to see if rapid synchronization is really needed. Can the application be modified to work around this? If not, the synchronization mechanisms themselves may need to be changed. If the rapidly changing data is only a subset of the database, memory versions of the database might be considered. Network communication bandwidth into the central database repository could also be increased. Is it possible to rewrite the application so that it uses a data flow approach rather than the central state of a database? Perhaps it can use self contained transactions that are transmitted to where they are needed. The subjobs could use direct communications between them as the primary flow for data dependency rather than passing this data through a database first.

In some applications, various database records may need to be updated atomically or in concert with others. Locking or synchronization primitives are used to lock all of the related database entries, whether they are in the same database or not, and then are updated while the synchronization primitives keep other subjobs waiting until the update is finished. One should look for ways to minimize the number of records being updated simultaneously to reduce the contention created by the synchronization mechanism. One should exercise caution not to create situations which might cause a synchronization deadlock with two subjobs waiting for each other to unlock a resource the other needs. There are three ways that are usually used to prevent this problem.

- The first is easiest, but can be most wasteful. This is to have all waits for resources to include time-outs. If the time-out is reached, then the operation must be undone and started over in an attempt to have better luck at completing the transaction.
- The second is to lock all of the resources in a predefined order ahead of the operation. If all of the locks cannot be obtained, then any locks acquired should be released and then, after an optional time period, another attempt should be made.
- The third is to use deadlock detection software. A transitive closure of all of the waiters is computed before placing the requesting task into a wait for the resource. If it would cause a deadlock, the task is not put into a wait. The task should release its locks and try again later. If it would not cause a deadlock, the task is set to automatically wait for the desired resource.

It may be necessary to run an application redundantly for reliability reasons, for example. The application may be run simultaneously on geographically distinct parts of the grid to reduce the chances that a failure would prevent the application from completing its work or prevent it from providing a reliable service. If the application updates databases or has other data communications, it would need to be designed to tolerate redundant data activity caused by running multiple copies of the application. Otherwise, computed results may be in error.

3

Security

This chapter extensively describes the security issues, techniques, and solutions needed to provide a robust and secure grid computing environment.

The following topics are discussed:

- Grid Security Fundamentals
- Public Key Infrastructure Overview
- Globus Security Framework
- Grid Infrastructure Security
- Potential grid Security Risks

3.1 Introduction to grid security

Security requirements are fundamental to the grid design. The basic security components within the Globus Toolkit provide the mechanisms for authentication, authorization, and confidentiality of communication between grid computers. Without this functionality, the integrity and confidentiality of the data processed within the grid would be at risk. To properly secure your grid environment, there are many different tools and technologies available. This chapter will examine some of those technologies and the different components provided within the Grid Security Infrastructure (GSI) of the Globus Toolkit.

In order to better understand grid security, it is best to start with some basic fundamentals. Grid security builds on well-known security standards. We will discuss these first and they will help us understand some of the more difficult security concepts. During this chapter, we will be discussing the nuts and bolts of grid security and the underlying technologies that allow for grid security to work.

In reality, an entire book could be dedicated to the topic of grid security. Due to the scope of this chapter, we will only be covering the basics of some encryption and PKI standards and how they are applied to grid security. This will give you a good overview of the different components of a PKI environment and some of the encryption techniques and how they work within the grid security framework.

Along with the different responsibilities associated with securing a grid environment, there are many risks involved. While the Grid Security Infrastructure (GSI) components make it easy to install and configure, you should not be fooled into thinking that your environment will be secured without a little extra work. With this overview, you will need to have a better understanding of what is involved in properly securing your grid environment.

3.1.1 Security fundamentals

Security requires the three fundamental services: authentication, authorization, and encryption. A grid resource must be authenticated before any checks can be done as to whether or not any requested access or operation is allowed within the grid. Once the grid resources have been authenticated within the grid, the grid user can be granted certain rights to access a grid resource. This, however, does not prevent data in transit between grid resources from being captured, spoofed, or altered. The security service to insure that this does not happen is encryption.

The world of security has its own set of terminology. The International Organization for Standardization (ISO) has defined the common security services found in modern I/T systems. The list was first put in ISO 7498-2 (OSI Security Architecture) and later updated in ISO 10181 (OSI Security

Frameworks). To have a better understanding of security systems and services, some security terms with explanations are listed below:

Authentication Authentication is the process of verifying the validity of a claimed individual and identifying who he or she is. Authentication is not limited to human beings; services, applications, and other entities may be required to authenticate also.

Access Control Assurance that each user or computer that uses the service is permitted to do what he or she asks for. The process of authorization is often used as a synonym for access control, but it also includes granting the access or rights to perform some actions based on access rights.

Data Integrity Data integrity assures that the data is not altered or destroyed in an unauthorized manner.

Data Confidentiality Sensitive information must not be revealed to parties that it was not meant for. Data confidentiality is often also referred to as privacy.

Key Management Key management deals with the secure generation, distribution, authentication, and storage of keys used in cryptography.

The Grid Security Infrastructure (GSI) of Globus and a Public Key Infrastructure (PKI) provide the technical framework (including protocols, services, and standards) to support grid computing with five security capabilities: user authentication, data confidentiality, data integrity, non-repudiation, and key management.

3.1.2 Important grid security terms

During the course of this chapter, we will be going over many important security terms. While some of the terms covered within this section provide the background as to how grid security works, there are some important concepts that should be highlighted. This is due to the fact that some areas within grid security require a precise understanding of the security concepts. Also, some security components may work slightly different within a grid environment as opposed to a standard network. Below are some important security concepts that you should be aware of when reading this chapter. These concepts will be described in greater detail throughout the chapter.

- Symmetric encryption: Using the same secret key to provide encryption and decryption of data.

- Asymmetric encryption: Using a two different keys for encryption and decryption. The public key encryption technique is the primary example of this using a “public key” and a “private key” pair.
- Secure Socket Layer / Transport Layer Security (SSL/TLS): These are essentially the same protocol, but are referred to one another differently. TLS has been renamed by the IETF, but they are based on the same RFC.
- Public Key Infrastructure (PKI): The different components, technologies, and protocols that make up a PKI environment.
- Mutual Authentication: Instead of using an LDAP repository to hold the public key (PKI), two parties who want to communicate with one another use their public key stored in their digital certificate to authenticate with one another. This topic is covered in 3.2.2, “Grid security communication” on page 68.

These are all important concepts to remember and will give you a head start in understanding how grid security works.

3.1.3 Symmetric key encryption

Symmetric key encryption is based on the use of one shared secret key to perform both the encryption and decryption of data. To ensure that the data is only read by the two parties (sender and receiver), the key has to be distributed securely between the two parties and no others. If someone should gain access to the secret key that is used to encrypt the data, they would be able to decrypt the information. This form of encryption is much faster than asymmetric encryption.

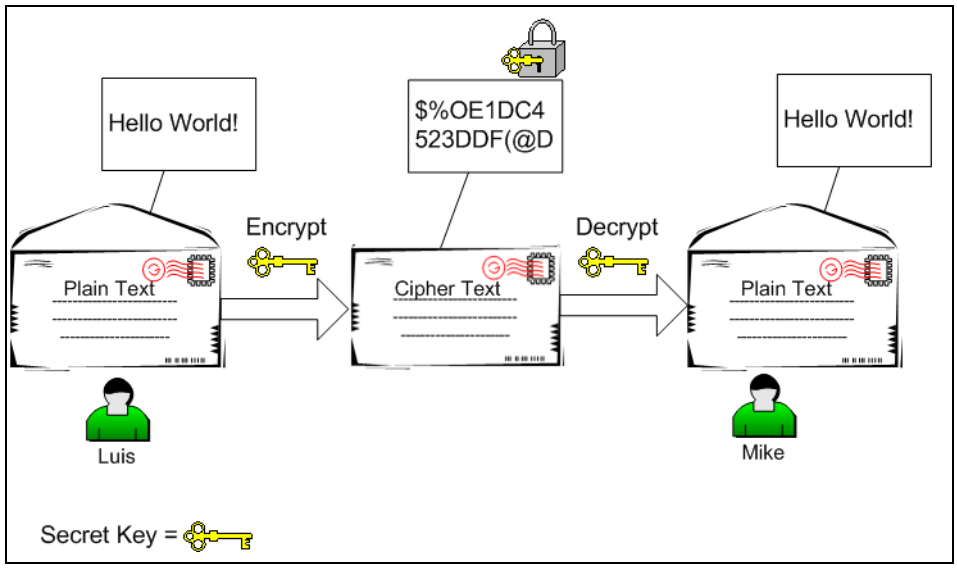

Figure 3-1 Symmetric key encryption using a shared secret key

Here are some commonly used examples of a symmetric key cryptosystem:

- Data Encryption Standard (DES): 56-bit key plus 8 parity bits, developed by IBM in the middle 1970s
- Triple-DES: 112-bit key plus 16 parity bits or 168-bit key plus 24 parity bits (that is, two to three DES keys)
- RC2 and RC4: Variable-sized key, often 40 to 128 bits long

To summarize, secret key cryptography is fast for both the encryption and decryption processes. However, secure distribution and management of keys is difficult to guarantee.

3.1.4 Asymmetric key encryption

Another commonly-used cryptography method is called public key cryptography. The RSA public key cryptography system is a prime example of this. In public key cryptography, an asymmetric key pair (a so-called a public key and a private key) is used. The key used for encryption is different from the one used for decryption. Public key cryptography requires the key owners to protect their private keys while their public keys are not secret at all and can be made available to the public. Normally, the public key is present in the digital certificate that is issued by the Certificate Authority.

The computation algorithm relating the public key and the private key is designed in such a way that an encrypted message can only be decrypted with the corresponding other key of that key pair, and an encrypted message cannot be decrypted with the encryption key (the key that was used for encryption). Whichever (public/private) key encrypts your data, the other key is required to decrypt the data. A message encoded with the public key, for instance, can only be decoded with the private key. One of the keys is designated as the public key because it is made available, publicly, via a trusted Certificate Authority, which guarantees the ownership of each of the public keys. The corresponding private keys are secured by the owner and never revealed to the public.

The public key system is used twice to completely secure a message between the parties. The sender first encrypts the message using his private key and then encrypts it again using the receiver's public key. The receiver decrypts the message, first using his private key and then the public key of the sender. In this way, an intercepted message cannot be read by anyone else. Furthermore, any tampering with the message will make it not decrypt properly, revealing the tampering.

The asymmetric key pair is generated by a computation which starts by finding two vary large prime numbers. Even though the public key is widely distributed, it is practically impossible for computers to calculate the private key from the public key. The security is derived from the fact that it is very difficult to factor numbers exceeding hundreds of digits.

This mathematical algorithm improves security, but requires a long encryption time, especially for large amounts of data. For this reason, public key encryption is used to securely transmit a symmetric encryption key between the two parties, and all further encryption is performed using this symmetric key.

3.1.5 The Certificate Authority

A properly implemented Certificate Authority (CA) has many responsibilities. These should be followed diligently to achieve good security. The primary responsibilities are:

- Positively identify entities requesting certificates
- Issuing, removing, and archiving certificates
- Protecting the Certificate Authority server
- Maintaining a namespace of unique names for certificate owners
- Serve signed certificates to those needing to authenticate entities
- Logging activity

Within some PKI environments, a Registrant Authority (RA) works in conjunction with the CA to help perform some of these duties. The RA is responsible for approving or rejecting requests for the certificate of public keys and forwarding the user information to the CA. The RA normally has the responsibility of validating that the user's information is correct before the signed digital certificate is sent back to the user. The Globus Toolkit includes a "simple" CA which can be installed for testing purposes. Within this scenario, the simple CA handles the job of both the CA and RA within the grid environment. As the number of certificates expands, these two jobs are normally separated.

One of the critical issues within a grid PKI environment is guaranteeing the system's trustworthiness. Before a CA can sign and issue certificates for others, it has to do the same thing to itself so that its identity can be represented by its own certificate. That means a CA has to do the following:

1. The CA randomly generates its own key pair.
2. The CA protects its private key.
3. The CA creates its own certificate.
4. The CA signs its certificate with its private key.

If a grid resource needs to securely communicate with another grid resource, it needs a certificate signed by a CA. The grid resource has to enroll with the CA by generating an unsigned digital certificate specifying his or her own information. The information submitted will be used by the CA to identify whether this grid resource is real and should be granted a certificate. The CA will then sign the digital certificate if the grid resource is eligible to receive the certificate. This certificate, after the CA signs the certificate, will be passed back to the requesting grid resource. So, one basic function of a CA is to create and issue certificates for a grid resource.

The CA's private key

The CA's private key is one of the most important parts in the whole public key infrastructure. It is used, for example, by the CA to sign every issued digital certificate within the grid network. Thus, it is especially susceptible to attacks from hackers. If someone were to gain access to the CA's private key, they would be able to impersonate anyone within the environment. Therefore, it is very important to protect this key. Knowing how sensitive the private key is to the rest of your grid environment, it is important to provide your CA server with any available security measures. This includes restricting physical and remote access and monitoring and auditing of the server.

CA cross certification

Generally within a single grid environment, a CA will provide certificates to a fixed group of users. If two companies or virtual organizations (VOs) need to

communicate and trust one another, this may require that both CAs trust one another or participate in cross certification. For example, Alice, an employee belonging to an organization with its own CA, may want to run a job on grid computer Mike, who is outside the organization, and who belongs to a different CA.

In order to do so, the following should be considered:

- Alice and Mike need a way to obtain each other's public key certificates.
- Mike needs to be sure that he can trust Alice's CA. Alice needs to be sure that she can trust Mike's CA.

Grid computers from different security domains or VOs will need to trust each others' certificates, so the roles and relationships between CAs have to be defined. The purpose of creating such trust relationships is to eventually achieve a global, interoperable PKI and enlarge the grid infrastructure. Once the relationship is established, both of the CA's can be configured to work with the grid system.

Managing your own CA

It is important to note that the simple CA is a fully functioning CA for a PKI environment, but it is only recommended for testing or demo purposes. For any type of production grid build, it is recommended that you evaluate some commercial PKI solutions that may better suit your needs and remove some responsibility for managing your own CA

3.1.6 Digital certificates

Digital certificates are digital documents that associate a grid resource with its specific public key. A certificate is a data structure containing a public key and pertinent details about the key owner. A certificate is considered to be a tamper-proof electronic ID when it signed by the Certification Authority for the grid environment.

Digital certificates, also called X.509 certificates, act very much like passports; they provide a means of identifying grid resources. Unlike passports, digital certificates are used to identify grid resources. Another difference between a digital certificate and a passport is that a certificate can (and should) be distributed and copied without restriction, while people are normally very concerned about handing their passports to someone else. Certificates do not normally contain any confidential information and their free distribution does not create a security risk.

The important fact to know and understand about digital certificates is that the CA certifies that the enclosed public key belongs to the entity listed in the

certificate. The technical implementation is such that it is considered extremely difficult to alter any part of a certificate without easy detection. The signature of the CA provides an integrity check for the digital certificate.

When a grid client wants to start a session with a grid recipient, he or she does not attach the public key to the message, but the certificate instead. The recipient receives the communication with the certificate and then checks the signature of the Certificate Authority within the certificate. If the signature was signed by a certifier that he or she trusts, the recipient can safely accept that the public key contained in the certificate is really from the sender. This prevents someone from using a fraudulent public key to impersonate the public key owner.

Contained in your digital certificate is the information about you and your public key. When you communicate with another party on the grid, the recipient will use your public key (contained in your digital certificate) to decrypt the SSL session ID, which is used to encrypt all data transferred between grid computers.

A digital certificate is made up of a unique distinguished name (DN) and certificate extensions that contain the information about the individual or host that is being certified. Some information in this section may contain the subject's e-mail address, organizational unit, or location.

Figure 3-2 on page 60 is a graphical depiction of the digital certificate.

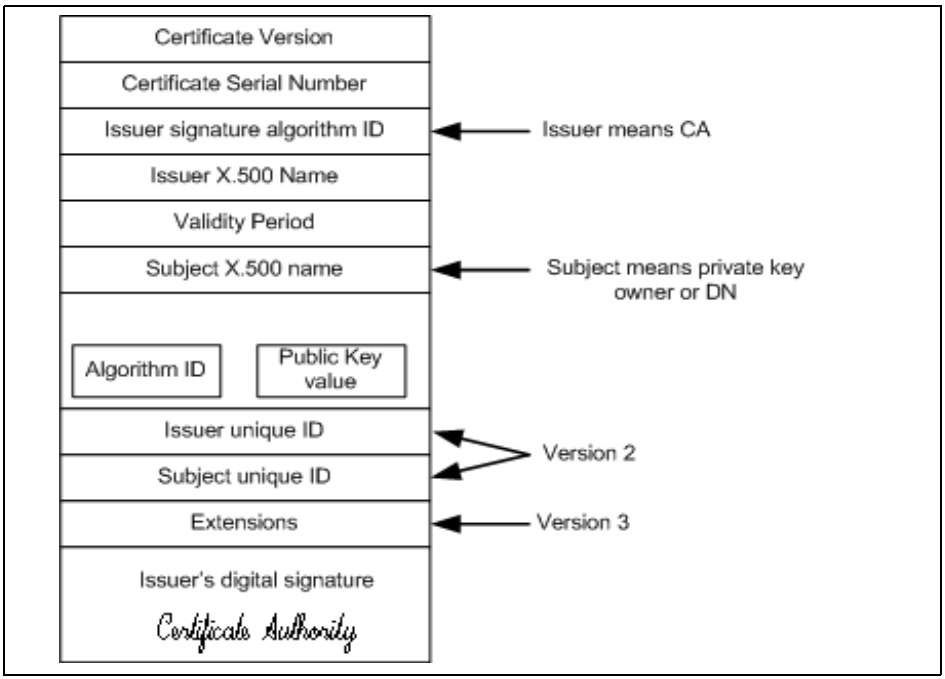

Figure 3-2 Digital certificate

Obtaining a client or a server certificate from a CA involves the following steps:

1. The grid user requiring certification generates a key pair (private key and certificate request containing the public key).
2. The user signs its own public key and any other information required by the CA. Signing the public key demonstrates that the user does, in fact, hold the private key corresponding to the public key.
3. The signed information is communicated to the CA. The private key remains with the client and should be stored securely. For instance, the private key could be stored in an encrypted form on a Smartcard, or on the user's computer.
4. The CA verifies the that the user does own the private key of the public key presented.
5. The CA (or optionally an RA) needs to verify the user's identity. This can be done using out-of-band methods, for example, through the use of e-mail, telephone, or face-to-face communication. A CA (or RA) can use its own record system or another organization's record system to verify the user's identity.

6. Upon a positive identity check, the CA creates a certificate by signing the public key of the user, thereby associating a user to a public key. The certificate will be forwarded to the RA for distribution to the user.

Verification of the user

The authentication described above is a one-time authentication for the purpose of certificate issuance. This can be compared to the process when a government authority issues a passport to an individual. The passport then serves as an authentication mechanism when this individual travels to foreign countries. Just like passports, digital certificates can subsequently be used in daily operations for authenticating subjects to other parties that require authentication.

Different types of certificates

There are two different types of certificates that are used within a grid environment. The first type of certificate is a user certificate that will identify different users on the grid. The second type of certificate is issued to grid servers.

User

As a grid user, you will need a user certificate to identify yourself within the grid. This certificate will identify your user name within the grid, not your server or workstation name. If your name is Mike Kendzierski, your digital certificate might have the distinguished name:

```
"/O=Grid/O=GridTest/OU=test.domain.com/CN=Mike Kendzierski"
```

Server

If you plan on running PKI enabled programs on your server, you will need to register a server certificate. This server certificate will register the fully-qualified domain name of your server to your certificate. For your certificate to work, your full-qualified DNS name will have to match your digital certificate. For example, If your server name was “Darkstar,” your server certificate would read:

```
/CN=Service/Darkstar.<domain>.<com>
```

Certificate revocation list

Since grid computing does not normally store digital certificates within a directory, there is no need for a certificate revocation list (CRL). During the authentication process, grid computers use mutual authentication that performs the digital certificate exchange and does not reference a directory store. For these purposes, the process of revoking grid certificates are handled manually.

In other PKI environments that use directory services to store the public key, a CRL is a means of notifying clients who wish to verify the revocation of

certificates. CRLs are issued to mark some certificates unusable, even though their expiration has not come yet.

Path validation

In order to verify that a certificate is valid, a check must be done to ensure that that the whoever signed the certificate is valid. This is how the path validation of a certificate is done. This is done to verify that the certificate path from the Root CA is valid and up the chain between the CA and grid client/server. This is especially important when explaining why delegated certificates are valid within the grid. This delegation is an extension to PKI and is not normally allowed. As long as the path is valid within the delegated certificate, the certificate will not be rejected.

For more information on certificate path validation, you should check out RFC 3280. You can find more information by accessing the following Web site:

http://www.ietf.org/rfc/rfc3280.txt

PKI directory services

Within some PKI environments, the signed keys are published to a public directory for easy retrieval. Instead of having the clients handle the mutual authentication, an external server is responsible for handling the authentication process. A good example of this process is the MyProxy server, which works as a grid Web proxy for Web portals. In this example, the user would authenticate to the Web portal, which would request the user's online credentials that are stored in the directory. Upon this authentication, the proxy would extract the DN within their digital certificate and match their credentials with the public key stored within the directory. If they two keys matched up, the user would be given access to resources within the grid.

3.2 Grid security infrastructure

Now that we have gone over some security fundamentals, explaining how the different grid security components interact will be much easier. In this section of the chapter, we will be going over how the different security components within the Globus Toolkit provide security services. We'll examine some different scenarios and walk through the various functions of the GSI.

3.2.1 Getting access to the grid

In order to build a grid environment using the GSI components, you have to create set of keys for public key cryptography and request your certificate from

the Certificate Authority and a copy of the public key of the CA. Figure 3-3 and the following procedure describe the steps to establish the GSI communication:

1. Copy the Certificate Authority's public key to your grid host with which you set up GSI.
2. Create your private key and a certificate request.
3. Send your certificate request to CA by e-mail or another more secure way if you are running a production system and need to positively identify the sender.
4. CA signs your request to make your certificate and sends it back to you.

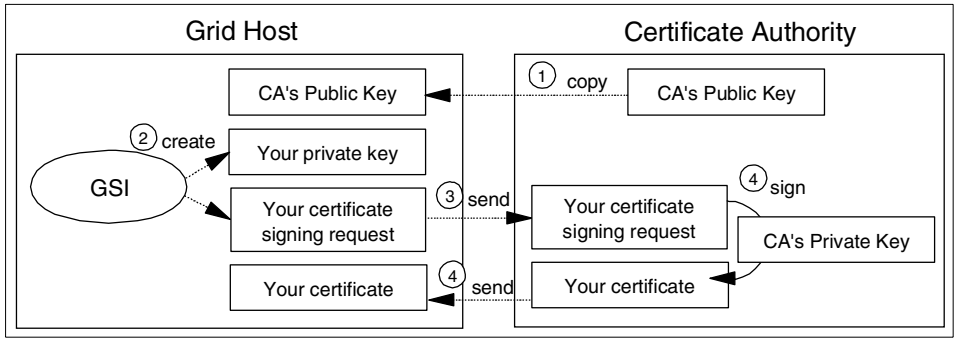

Figure 3-3 Preparation procedure for GSI

When that procedure has been completed and you have received your signed digital certificate, you will have three important files on your grid host. They are:

- The CA's public key
- The grid host's private key
- The grid host's digital certificate

In order to provide secure authentication and communication for your grid computer, you should not let others have access to your private key. An extra layer of security was added to the private key, which includes a secret passphrase that must be used when using your private key along with your digital certificate. This is to prevent someone from stealing your digital certificate and private key and being able to automatically use them to access grid resources. The host key is protected by the local operating system privileges within the grid server.

Authentication and authorization

Imagine a scenario where you need to communicate with another grid computer's application and you want to ensure that the data from the host is really from the host. Besides making sure that you can trust the grid host, you

want to make sure the grid host that you want to communicate with trusts your grid computer. In these cases, you can use the authentication function of GSI, as shown in Figure 3-4 on page 65. After you have authenticated with the remote grid resource, you then have the option of having the grid resource give you access to resources on your behalf. In this case, you can use the authorization function of GSI.

Through the steps described below, you (grid host A) are authenticated and authorized by grid host B. Almost all steps are for authentication, except the last authorization step:

1. Send your certificate to the other host (host B) by which you want to be authenticated.
2. Host B will get your public key and your subject from your certificate by using the CA public key.
3. Host B creates a random number and sends it to you (host A).
4. If you got the number, encrypt it with your private key (you may be requested to input password to open your private key) and send the encrypted number to host B.
5. Host B will decrypt the number and check that the decrypted number is really the one that it sent before. Then host B authenticates that the certificate is really yours, because only you can encrypt the number with your private key.
6. Your certificate is authenticated by host B, then your subject in the certificate is mapped to a local user name. The subject is in the form of Distinguished Name (DN) like "`O=Grid/O=Globus/OU=itso.grid.com/CN=your name`", and it is the name that is used by LDAP to distinguish the entries in the directory service. The subject is used to specify your user identity in a grid environment. Here you, the owner of DN, are authorized by host B to act as a local user on the host B.

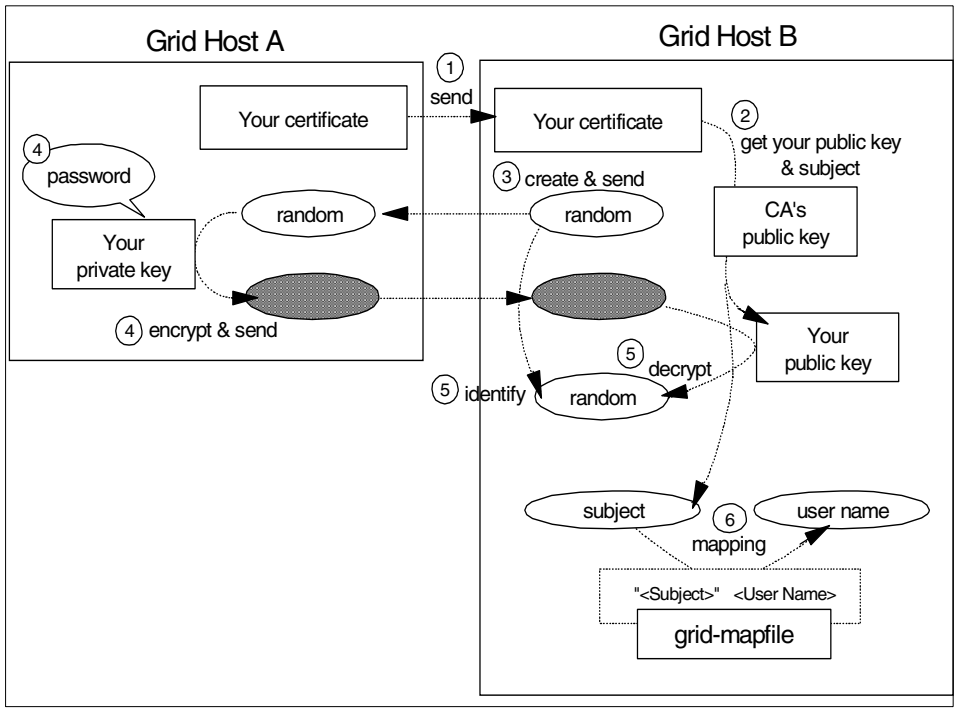

Figure 3-4 Authentication procedure

In grid environments, your host will become a client in some cases, and in other cases, a server. Therefore, your host might be required to authenticate another host and be authenticated by the host at the same time. In this case, you can use the mutual authentication function of GSI. This function is almost the same as explained above, and it proceeds with the authentication steps, and changes the direction of hosts and redoes the procedure.

Briefly speaking, authentication is the process of sharing public keys securely with each other, and authorization is the process that maps your DN to a local user/group of a remote host.

Delegation

Imagine a situation where you distribute jobs to remote grid machines and let them distribute their child jobs to other machines under your security policy. In this situation, you can use the delegation function of GSI, as shown in Figure 3-5 on page 67.

If you are on the side of host A, you can create your proxy at host B to delegate your authority. This proxy acts as yourself, and submits a request to host C on your behalf.

The next steps (see “Proxy creation” on page 66) describes the procedure to create your proxy (proxy creation) at a remote machine, and the procedure to submit a request (see “Proxy action” on page 66) to the other remote host on your behalf (proxy action).

Proxy creation

1. A trusted communication is created between host A and host B.
2. You request host B to create a proxy that delegates your authority.
3. Host B creates the request for your proxy certificate, and send it back to host A.
4. Host A signs the request to create your proxy certificate using your private key and sends it back to host B.
5. Host A sends your certificate to host B.

Proxy action

1. Your proxy sends your certificate and the certificate of your proxy to host C.
2. Host C gets your proxy's public key through the path validation procedure:
   a. Host C gets your subject and your public key from your certificate using CA's public key.
   b. Host C gets the proxy's subject and your proxy's public key from your proxy's certificate using your public key.
   c. The subject is a Distinguished Name similar to `"O=Grid/O=Globus/OU=itso.grid.com/CN=your name"`. The subject of proxy certificate is similar to its owner's (your) subject and is similar to `"O=Grid/O=Globus/OU=itso.grid.com/CN=your name/CN=proxy"`. So in order to validate the proxy certificate, Host C just has to check that the words that eliminate the words `"/CN=proxy"` from the proxy's subject is just the same as your subject. If it is validated, your proxy is authenticated by host C and able to act on your behalf.
3. The proxy encrypts a request message using its private key and sends it to Host C.
4. Host C decrypts the encrypted message using the proxy's public key and gets the request.
5. Host C runs the request under the authority of a local user. The user is specified using a mapping file, which represents the mapping between the grid users (subject) and local users (local user name).

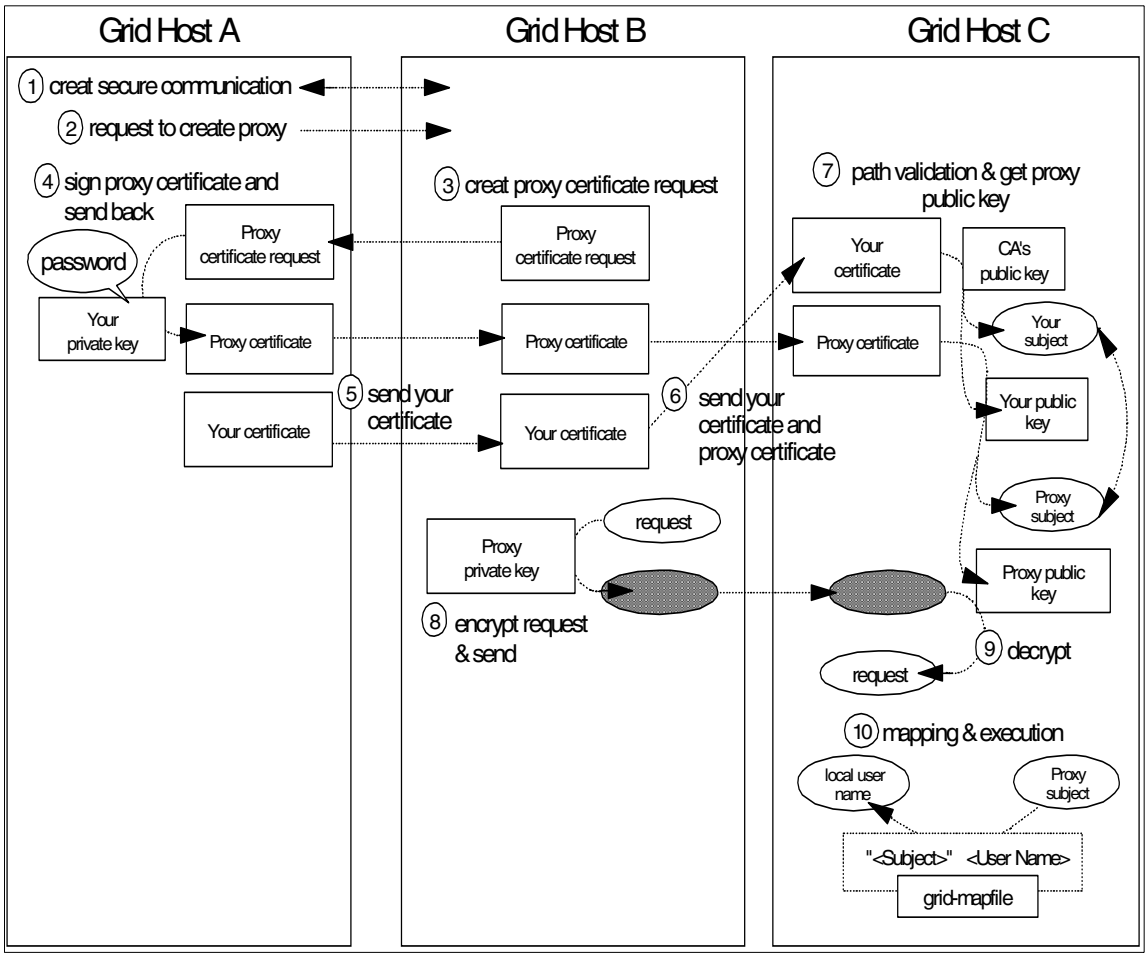

Figure 3-5 Delegation procedure of user's proxy

The procedure in Figure 3-5 represents the remote delegation, where a user creates a proxy at a remote machine. There is also a local delegation, where a user creates a proxy certificate at the local machine; for that task, Globus Toolkit uses the **`grid-proxy-init`** command and gatekeeper daemon mechanism.

When you make a proxy to a remote machine (in the remote delegation), the proxy's private key is on the remote machine, so the super-user of that machine can access your proxy's private key and conduct business under your name. This delegated credential can be vulnerable to attacks. In order to avoid this impersonation, it is recommended that the proxy attain restricted policies from its owner, as in the case with GRAM, for example. The standardization of this proxy

restriction is now going on under GSI Working Group of Grid Forum Security Area, and you can see more details in its Internet draft at the site below:

http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-03.txt

3.2.2 Grid security communication

While we have gone over the process of using PKI within a grid environment and the different functions of GSI, it is still important to understand the communication functions within the Globus Toolkit. By default, the underlying communication is based on the mutual authentication of digital certificates and SSL/TLS.

The digital certificates that have been installed on the grid computers provide the mutual authentication between the two parties. We will be going over this process in detail later on in this section. The SSL/TLS functions that OpenSSL provides will encrypt all data transferred between grid hosts. These two functions together provide the basic security services of authentication and confidentiality.

Mutual authentication

To allow secure communication within the grid, the OpenSSL package is installed as part of the Globus Toolkit. Within the Globus Toolkit, OpenSSL is a software package that is used to create an encrypted tunnel using SSL/TSL between grid clients and servers.

The process of mutual authentication begins when two grid resources want to share resources. Instead of using a key repository, each grid resource authenticates with one another based on their digital certificate. For example, one grid resource will attempt to establish secure communication with another grid resource. Before the recipient will allow the client access to their resources, they need to authenticate to one another. This process is documented below with the SSL handshake.

SSL handshake

In order to establish the secure communication between the grid server and grid client, a handshake must be established. This handshake is responsible for determining the SSL settings, exchanging public keys and the basis for the mutual authentication process. The handshake process is as follows:

1. A grid client contacts a remote grid server to start a secure session by using a digital X.509 ID certificate.
2. The grid client automatically sends to the server the client's SSL version number, cipher settings, randomly generated data, and other information the server needs to communicate with the client using SSL.

3. The grid server responds, automatically sending the grid client the site's digital certificate, along with the server's SSL version number, cipher settings, and so on.
4. The customer's client examines the information contained in the server's certificate, and verifies that:
   a. The server certificate is valid and has a valid date.
   b. The CA that issued the server certificate has been signed by a trusted CA whose certificate is built into the client.
   c. The issuing CA's public key, built into the client, validates the issuer's digital signature.
   d. The domain name specified by the server certificate matches the server's actual domain name.
5. If the server can be successfully authenticated, the grid client generates a unique “session key" to encrypt all communications with the grid server using asymmetric encryption.
6. The user's client encrypts the session key itself with the server's public key so that only the site can read the session key, and sends it to the server.
7. The server decrypts the session key using its own private key.
8. The grid client sends a message to the server informing it that future messages from the grid client will be encrypted with the session key. The grid server then sends a message to the grid client informing it that future messages from the server will be encrypted with the session key.
9. An SSL-secured session is now established. SSL then uses symmetric encryption (which is much faster than asymmetric PKI encryption) to encrypt and decrypt messages within the SSL-secured "pipeline."
10. Now that the first grid resources has authenticated, the second grid resource will now authenticate using the same process.
11. Once the session is complete, the session key is eliminated.

As long as both grid resources have a valid digital certificate, the process of mutual authentication will succeed. This is a good example of how grid security uses both symmetric and asymmetric encryption to authenticate and secure data transfer between grid resources. A grid client uses asymmetric encryption to authenticate and once it is authenticated, it passes symmetric encryption along with a shared secret key to encrypt and decrypt all data traffic between them.

Other grid communication

If you cannot physically access your grid client or server, it may be necessary to gain remote access to the grid. While your operating systems default telnet

program works fine for remote access, the transmission of the data is in clear text. That means that the data transmission would be vulnerable to someone listening or sniffing the data on the network. While this vulnerability is low, it does exist and needs to be dealt with.

To secure the remote communication between a client and grid server, the use of Secure Shell (SSH) can be used. SSH will establish an encrypted session between your client and the grid server.

GSI-SSH client

Depending on your operating system, there are also other standard SSH clients available. For more information you run a search on Google (`http://www.google.com`) using SSH as the keyword. You can also read the SSH charter page on the IETF Web site at:

`http://www.ietf.org/html.charters/secsh-charter.html`

3.2.3 Grid security step-by-step

In order to better understand the process for accessing grid resources, we have outlined the basic process from start to finish.

Local delegation

This program is used to get a session "proxy" certificate using your long term certificate.

The proxy certificate is used to authenticate the user and user programs to resources on the grid. For example, the user can run jobs on the grid with the **`globusrun`** command. The **`globusrun`** command is authenticated with the proxy certificate. The proxy certificate is created with the **`grid-proxy-init`** command. A proxy certificate must be created before jobs can be run on the grid. The proxy certificate is a session certificate with a limited or short-lived life time, which is signed by the user certificate. This is functionally equivalent to the Kerberos **`kinit`** program or DCE dce_login.

The motive behind this model is to provide for the single sign on. The single sign on is the **`grid-proxy-init`**. Once the grid proxy certificate or is created, this certificate is used for authentication on the grid.

This model works because it creates a certificate trust hierarchy, as shown in Figure 3-6 on page 71.

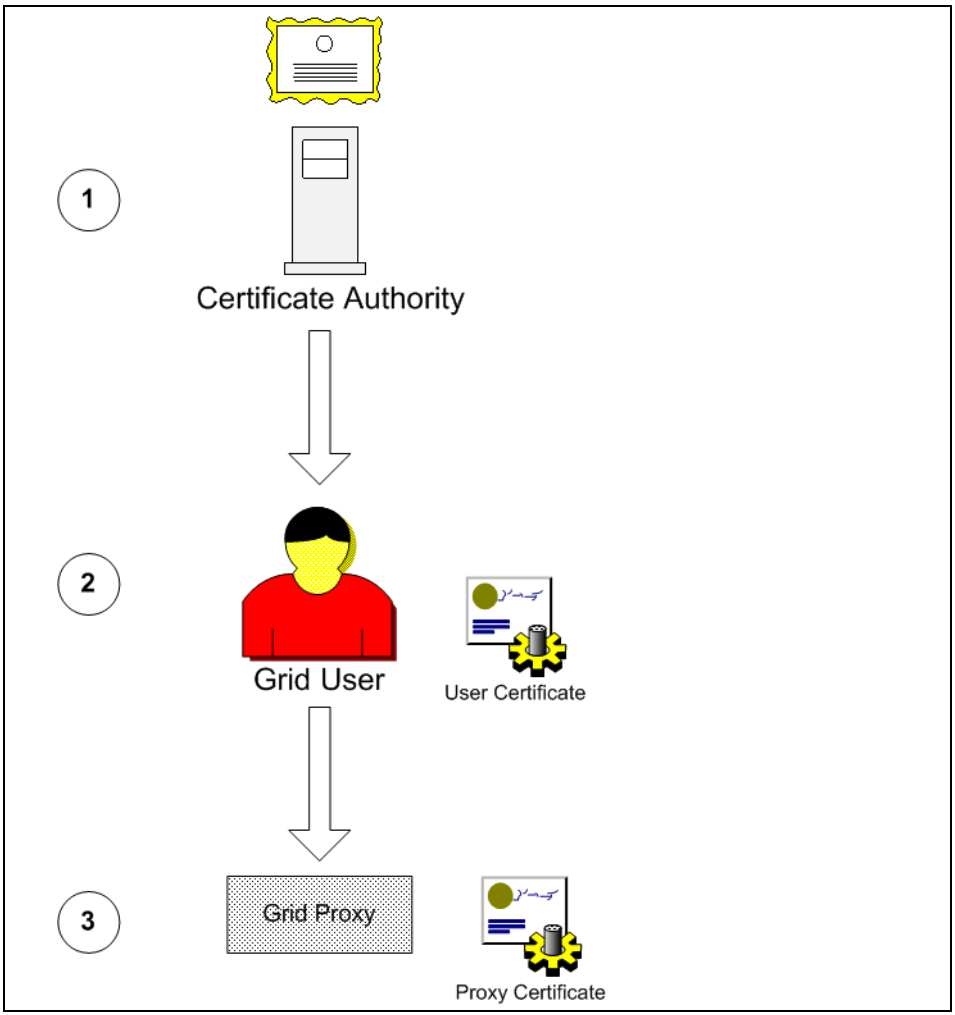

Figure 3-6 Authentication process

The hierarchy is as follows:

1. The remote grid resource trusts the CA. The remote grid resource trusts the CA because it placed the CA's certificate in /etc/grid-security/certificates.
2. The remote grid resource can authenticate the user certificate because it is digitally signed by the CA.
3. The remote grid resource can authenticate the user proxy certificate because it is digitally signed by the user certificate.

It is analogous to meeting three people at a party: CA, Alice, and Proxy. Proxy hands you a card that is similar to Figure 3-7.

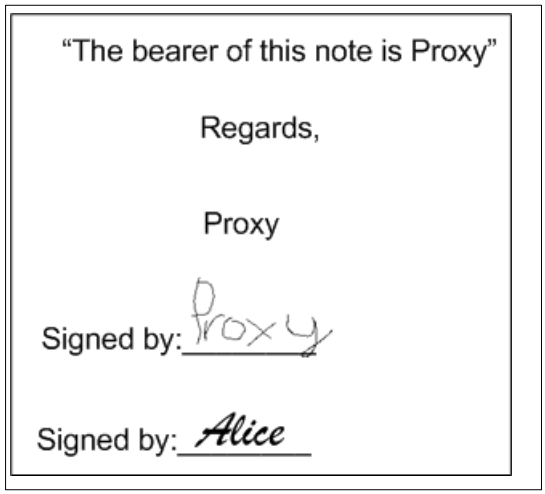

Figure 3-7 Certificate signed by Alice

You are not familiar with Alice's signature, so you take a card from Alice, which is similar to Figure 3-8.

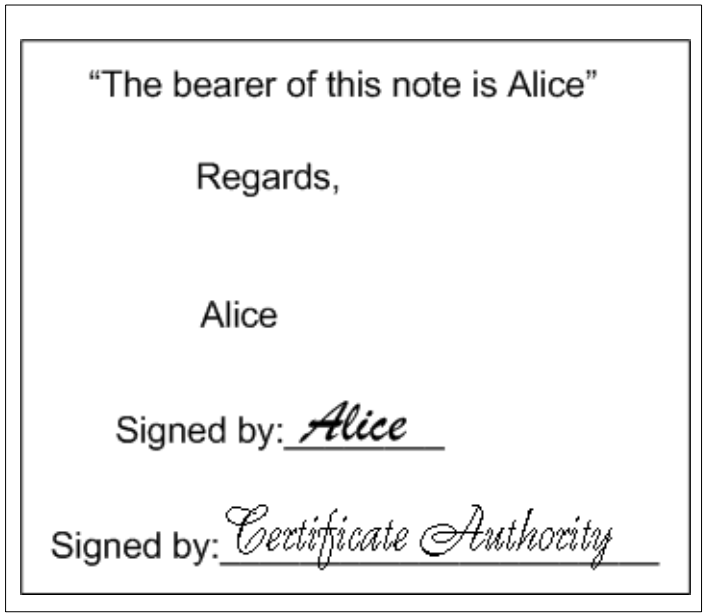

Figure 3-8 Certificate signed by CA

You keep a copy of CA's signature in you wallet. You compare the CA signature on Alice's card to the copy you keep in your wallet and they match. You now have an authenticated copy of Alice's signature which you compare to the signature on Proxy's card. They match, and you now trust you that you are talking to Proxy. You have authenticated that this person is Proxy.

The **grid-proxy-init** command uses the SSL library to create a proxy certificate that is stored in /tmp/<filename>, where <filename> is equal to x509up_u<uid>, where uid is equal to the UID of the user running **grid-proxy-init**. The permission of this file is “-rw------- owner group” of the user running the command.

This file is an X.509 certificate where the issuer is the user’s primary certificate. Basically, the users primary certificate acts like a CA to create this session or "proxy" certificate. The proxy certificate is considered a short-lived certificate. By default, it has a validity period of 12 hours, but this can be specified by the **grid-proxy-init** parameter -hours.

The proxy certificate, as with all X.509 certificates, contains a unique name and public key. The proxy certificate's unique subject name or distinguished name is the primary certificates unique name plus "CN=proxy" (limited proxy). This is best illustrated with the **grid-cert-info** and **grid-proxy-info** commands. If the **grid-cert-info** command is run with the filename of our primary certificate, the contents of the certificate is displayed:

```
%grid-cert-info -f.globus/usercert.pem -subject
/C=US/O=IBM/OU=GridLPP/OU=austin.ibm.com/CN=griduser
```

The -subject flag displays the subject or distinguished name (DN).

A complete description of X.509 certificates can be found RFC 2459. The /tmp/x509_up_u<uid> file created by **grid-proxy-init** contains two other components in addition to the proxy certificate. It also contains the private key of the proxy certificate and the user certificate.

The proxy certificates private key is only protected by the file permissions of /tmp/x509_up<uid>. Since the proxy certificate is short lived, a compromised or stolen certificate will become useless at the end of its life.

The user certificate's private key remains encrypted in the $HOME/.globus/userkey.pem file. It can only be accessed with the passphrase that is given when the user certificate is created with the **grid-cert-request** command.

3.3 Grid infrastructure security

Apart from the different GSI components and technologies, there are many other infrastructure security components that are needed to secure the grid. As in other areas of grid design, the grid infrastructure security builds on other security principles. While these security components are optional, they are considered standard within many production networks. We will explore some of these basic security concepts and see how they fit into a grid infrastructure.

3.3.1 Physical security

Once again, the security of grid infrastructure is based on other common security fundamentals. The basics involve solid physical security practices for all grid computers. The physical environment of a system is also considered a part of the infrastructure. If the servers are kept in an open room, no matter how secure the applications are designed or how complex the cryptographic algorithms are, the server services can easily be interrupted, such as being powered off, or otherwise tampered with. Therefore, physical access should be controlled and is part of the security policies that need to be defined.

The CA server should be located in a robust, dedicated, and locked room. All accesses should be logged and controlled so that only CA-related personnel can go in. The power supply to the servers should never be interrupted. This means an uninterruptable power supply (UPS) must be used. However, a UPS may still run out of electricity after a prolonged period. In such a case, the servers should be able to automatically back up the data and properly shut down. The room's entrance can also be monitored to check who has accessed the room.

For maximum security, the network segment where the PKI-sensitive server machines are installed should be physically and logically separated from the rest of the network. Ideally, the separation is done through a firewall that is transparent only for PKI-related traffic. Normally, PKI traffic is reduced to using only a few TCP/IP ports.

3.3.2 Operating system security

A review of the configuration files for each operating system and middleware component within the scope of the project determines how each effectively allows authorized users access based on your security policy and prevents and detects unauthorized access attempts at all times. You should:

- Remove any unnecessary processes from the servers. If the grid server does not need sendmail or an FTP server running, these processes should be disabled.
- Remove any unnecessary users or groups.

- Use strong passwords for all users on the grid server.
- Update your server with the latest updates and security FixPacks. This includes all software the has been installed as well.
- Restrict access to the /.globus directory.
- Consider using host IDS to monitor important directories on the server.
- Enable logging and auditing for the server.
- Use a uniform operating system build whenever possible.
- Enable file level restrictions on important files within the server.
- Make periodic reviews of the operating system every other month to ensure that nothing major has changed.
- Enable anti-virus protection.

3.3.3 Grid and firewalls

Firewalls can be used within networked environment to logically separate different sets of computers that require additional security. In a grid environment, this is no different. The use of firewalls within a grid design helps restrict network access to computers. The firewall is an important piece of the security infrastructure, so it needs to be carefully analyzed and understood before it is implemented

3.3.4 Host intrusion detection

A recommended option for further securing your grid computers is to invest in a host intrusion detection (IDS) product. As with any software application that stores important files within the local workstation, host intrusion detection can add a greater defense for anyone manipulating files on the workstation that should not be doing so. If the host IDS product detects a changed file on the server, it can send an alert to a central monitoring workstation to log and alert the necessary people.

An ID system gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization). An IDS uses vulnerability assessment (sometimes referred to as scanning), which is a technology developed to assess the security of a computer system or network.

Intrusion detection functions include:

- Monitoring and analyzing both user and system activities

- Analyzing system configurations and vulnerabilities
- Assessing system and file integrity
- Ability to recognize typical patterns of attacks
- Analysis of abnormal activity patterns
- Tracking user policy violations

Network intrusion detection

There can be a point made for network IDS within a grid environment, but some of that benefit would be lost due to the encryption between grid servers. While a network IDS would be able to use special signatures for standardized network traffic, the introduction of a network based IDS system would be lost because of the SSL/TLS encryption. While a network IDS system could not see the data payload portion of the packet that is encrypted, the network IDS could respond to events based on the packet header that is unencrypted. Network IDS is best suited for placement where it can analyze unencrypted traffic.

The use of any IDS is an optional component within an architecture, but is strongly recommended for good security practices.

3.4 Grid security policies and procedures

Good security policies and procedures are used to complement the variety of security components that make up a security infrastructure. This is no different in a grid environment, but may take on more importance since you may be dealing with networks out of your control. To help manage this risk, different policies and procedures should be used. These policies and procedures will help build a certain way of managing the security controls.

One of the first steps an organization has to consider when comprehensive security solutions are to be introduced is to define a feasible set of security policies. In the first place, this has little to do with a PKI because security policies need to be in place for any kind of I/T infrastructure. Only when the deployment of a PKI has been decided do some additional benefits and issues come up that need to be defined within security policies. The following subsections discuss security policies that primarily relate to a PKI.

3.4.1 Certificate Authority

A PKI must be operated in accordance with defined policies. The deployment of a PKI system in an organization requires the development of security policies and processes for that organization. The demo CA that is provided within the

Globus Toolkit provides the software in order to build a CA, but unfortunately none of the policies. In this section, we will examine some of the basic policies and expectations that a CA would normally be responsible for. For any type of production CA duties, it is suggested that you examine a commercial vendor to provide these services for you.

The standardization effort has been made to involve security policies in a PKI framework systematically, as outlined in RFC 2527, Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. According to X.509, a certificate policy is “a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.” A more detailed description of the practices followed by a CA in issuing and otherwise managing certificates may be contained in a Certification Practice Statement (CPS) published or referenced by a CA.

A certificate policies extension contains a sequence of one or more policy information terms, each of which consists of a registered object ID (OID) and optional qualifiers. Applications with specific policy requirements will have to recognize the OID meaning in at least the same security domain. If the required policy’s OID is not contained in the certificate extension field, or if any existing critical OIDs are not understood by the application, the application has to reject the client’s request. Security policies also result in processes that have to be in place and subsequently enforced. Processes describe (and/or mandate) the way an infrastructure is utilized by its administrators and users. Processes may include elements, such as:

- The certificate requesting, issuance, distribution, and revocation processes.
- The use of certificates for client authentication.
- The use of certificates for securing e-mail communication.
- The use of certificates for inter-organization communication.
- Procedures to follow when security violations are suspected.
- Handling guidelines for private keys and certificates.
- Application development guidelines for PKI exploitation (such as user authentication using certificates).

A PKI will alter many existing business processes and require many new ones to support it. These processes can cover technical, organizational, legal, and infrastructure elements of the whole workflow.

- CA key generation
  - Who is involved?
  - How is the process secured?

- CA key backup
  - How is a backup of the CA private key accomplished?
- CA Key restore
  - How is a key restored?
- CA Key compromise
  - What happens if the key is broken?
- User registration
  - How does a user obtain a certificate?
- Certificate revocation
  - How is a certificate revoked?

CA implementation

If you are planning on implementing your own Certificate Authority, you will need to build on some of the tools that are provided within the Globus Toolkit. The Globus Toolkit provides some of the basic tools for a demo CA within a lab or testing environment, but there is more to building a CA than installing a few scripts.

In order to manage and administer your own CA, you should be aware of some of the other resources and policies that are normally required. If you plan on managing a CA yourself, your plan for implementation must include:

- Required resources and skills
- Required PKI and security process additions and changes
- Recommended implementation timeline and dependencies
- Required changes to the technical infrastructure
- Adoption of the CPS, certificate, and security policies
- Required PKI and security policy additions and changes
- All required checkpoints and approvals

3.4.2 Security controls review

When building any new environment yo implement a new software application, it is always a good idea to perform a security health check. A security health check will help determine how these new changes will affect the overall security of the environment and any other areas of change. This can help provide guidance on the overall use of security controls or how you are managing security within your environment. A review of your security controls can help you better understand how security works for your passwords, administration, toolsets, auditing, and

monitoring within your environment. This will provide an in-depth review of the site security controls in place and the related processes used within the organization.

3.5 Potential security risks

Building a PKI environment will provide the necessary services along with the GSI to design a secure grid solution. This, however, does not guarantee that there are not any security risks. Within this section, we will examine some possible vulnerabilities to watch out for during your security design. This is by no means a laundry list for all security vulnerabilities or a cookbook for building a secure infrastructure. The importance of this section is to highlight some vulnerabilities that you may not have been aware of and allow you the option of taking the proper steps to improve their security.

Ultimately, it will be up to you to design, build, and test out your security infrastructure within your grid network. All of the security tools, processes, and policies in the world will not completely secure a networked environment. There is still some risk involved, but hopefully with the right people and tools at your disposal, you can reduce that risk to a negligible level.

3.5.1 PKI vulnerabilities

Just because you have built a PKI environment does not mean that your network is completely secure. There are still many vulnerabilities to be aware of. It is necessary to always keep an open mind and understand that with any networked environment there is going to be some risk involved.

Within a PKI environment, you constantly have to worry about the locations of your private keys and thefts of digital certificates. The following areas should be considered when dealing with a PKI environment:

- Impersonation: Obtaining a certificate through fraudulent means (either user or organization).
- Theft of private key: Unauthorized use of a private key associated with a valid certificate.
- Compromise of root CA private key: Using a CA key to sign fraudulent certificates or destroying a private key.
- Automatic Trust Decisions: Automated trust decisions can also automate fraud.

3.5.2 Grid server vulnerabilities

Any server or workstation that participates in the grid is a potential vulnerability to an external or internal hacker. Knowing this, it is very important to protect and isolate any grid computer from any network or resources that do not need explicit access to the grid. The most common way to isolate or protect your grid computers from unauthorized network access can be done through the use of good security policies and procedures. There are not any magic scripts or firewalls to protect your grid computers, but common sense can play a big part in how well your grid network is secured. The following areas within the grid server should be protected:

- Good physical security will limit the exposure of anybody walking up to the server and accessing the console.
- Protect any directories of the /.globus directory.
- Theft of the digital certificate and private key (along with the private key phrase).
- Any application vulnerabilities or processes that are running on the grid server.
- Any modification of the gridmap file.
- Latest operating system FixPacks.
- Any application FixPacks.

4

Design

This chapter provides architectural design considerations for grid computing, especially for the Globus toolkit. Other design topics that will be discussed are different grid topologies, grid infrastructure design, and grid architecture models.

At a glance, the following topics are discussed:

- Grid architecture design concepts
- Different grid topologies
- Grid architecture models
- Building a grid architecture
- Grid architecture conceptual model

4.1 Building a grid architecture

The foundation of a grid solution design is typically built upon an existing infrastructure investment. However, a grid solution does not come to fruition by simply installing software to allocate resources on demand. Given that grid solutions are adaptable to meet the needs of various business problems, differing types of grids are designed to meet specific usage requirements and constraints. Additionally, differing topologies are designed to meet varying geographical constraints and network connectivity requirements. The success of a grid solution is heavily dependant on the amount of thought the IT architect puts into the solution design.

Once the functional and non-functional requirements are known, the IT architect should readily be able to select the type of grid and the best topology required to satisfy the majority of the business requirements. When armed with this information, the high level grid design will be easier to complete, and by leveraging the use of known grid types and topologies, articulating the solution design will require much less effort.

It is important to focus on starting small and to begin building the basic framework of the design. Rather than setting out to build the desired end state grid solution all at once, consider building the grid solution in a phased approach. The milestone for the initial phase is to provide an “intragrid” solution, which is essentially a grid sandbox that supports a basic set of grid services. This solution would support a single location built upon the core grid components, such as a security model, information services, workload management, and the host devices. As long as this model supports the same protocols and standards, this design can be expanded as needed.

An easy way to begin the design is to start with the grid security model. The grid security model is typically built upon a Public Key Infrastructure (PKI) framework, and is the foundation for grid user authentication. Knowing the grid type, grid topology, and the desired security model is fundamental to the customization of the high level grid solution design. Given that the primary characteristic of a grid solution is that the network and hardware infrastructure is shared by multiple users and potentially multiple locations, it makes logical sense to make early architectural decisions based upon on the implications of the security requirements.

The first step of the design process is to build a graphical representation of the grid components. The subsequent phases of the design will be focused on the next level of architecture. This phase of the design is a starting point for architects, technical managers, and executives to understand the overall configuration of the architecture.

At a glance, the grid architecture design should offer the following:

- The “blueprint” for the detailed conceptual design.
- The use of open standards prescribed by the grid framework. For this redbook, the framework used follows the Globus toolkit patterns.
- A multi-dimensional tiered and layered view of the grid infrastructure, which demonstrates the ability to logically partition grid resources so that their service consumption does not impact other grid locations.
- The middleware components and subsystems for a grid infrastructure integration.
- A design for communication to both business and technical personnel, for budget and planning purposes, and to provide application development an illustration of how the shared grid infrastructure will impact the middleware solution design.
- The distribution of applications and subsystems.
- A means for identifying the necessary technical, infrastructural, and other middleware components and subsystems for a grid infrastructure.

4.1.1 Solution objectives

The design objectives provide a basic framework for building the grid infrastructure. The advantage of using design solution objectives is to start documenting certain areas that can affect the overall design. Within your design, you are going to need to make sure that the grid can provide a certain amount of security, availability, and performance. By documenting these different objectives or requirements, it will make your design a lot easier to follow. You will also be able to justify some of your decisions during the course of the design by being able to come back to certain objectives and making sure they were met.

Once the design objectives have been defined, you can separate them into individual subsystems. This allows each design objective to be worked on in parallel, but at the same time providing a cohesiveness for the overall architecture. Once you have documented the core subsystems of the design, you can focus on the different requirements that your grid design will be comprised of.

When you start building the initial pieces of your design, you need to make sure that your solution objectives line up with the customer’s requirements. For a grid design, this is especially important, as there are not only the standard infrastructure components to consider, but specialized middleware and application integration issues as well. Making sure that your solution objectives satisfy your stated requirements will allow you to design a working grid.

Security

Within any networked environment, there is going to be some risk and exposure involved with the security of your infrastructure. Unless the computers are unplugged in a locked room, there is the potential that someone may bypass the security and get access to protected resources. Whether the weaknesses are exploited in the infrastructure, application, configuration, or administration, there is some level of risk.

The security objectives are in place to help to reduce that risk to an acceptable level. While no design is 100% secure, the level of risk is reduced and controlled through the use of security controls. The goal of the security objectives are to examine the security requirements and implement the necessary tools and processes to reduce the risk involved.

The degree of security involved is based on the type of grid topology and the data the security will be protecting. The security requirements for a grid design within a bank will be completely different from those of an academic institution doing research. Whatever the security requirements may be, the security design objectives for the grid design need to be a central focus for the conceptual architecture.

Considering that the basic grid security model is based on PKI, it is imperative that the security components are designed and thought out carefully. While PKI has been around for a while, there are different components and necessary processes that should be identified. Rushing this process could lead to many problems in the future.

With the PKI architecture being the focus of the initial design, there are still areas that need attention. The infrastructure components (firewalls, IDS, anti-virus, and encryption) and the processes to manage these pieces are all part of the security objectives. Knowing which areas match up with your existing environment is the first step to robust security. The following bullet points are an example of some security questions that will be answered during the course of the design:

- Where will my CA be deployed and how will we manage it?
- Do I have the necessary processes in place to administer my own CA?
- What are the responsibilities for managing my own CA?
- How will I administer security on the local servers?
- Are my servers of a uniform build or common operating environment?
- Do I have a consistent software build across critical grid infrastructure systems?
- Which processes are running on my servers?

- Will any existing applications conflict or further expose my grid to any vulnerabilities?

Availability

Availability in its simplest terms commonly refers to the percentage of time that a site is up and servicing job requests. Determining how much availability should be built into the design are part of the availability objectives. This leads down the path of discovering how many potential single point of failure exist and how much redundancy should be built into the design. It is inevitable that some components will fail during a lifetime of usage, but this can be managed by using redundant components where possible.

Whenever you review various availability scenarios, there are always discussions about the amount of availability that is required. In this respect, a grid design is no different from any other infrastructure. A good start is to list the potential components within the design that should be resilient to failure. Once these components have been identified, you can seek out the specific availability options for those components. In the following examples, some different infrastructure options are described.

An important point that needs to be discussed is the availability of dynamic resources within a grid environment. Grid is not like a standard environment where resources are fixed and do not change regularly. Within Globus environments, the resources are constantly changing according to the membership and participation of the grid. When grid resources are active, they can register with the information services (GIIS) within the grid to alert the system of their state. It is important to make sure that when you design your grid that you keep this in mind.

Besides the grid middleware components, the different infrastructure components will also require different levels of availability. Some components will be more critical than others and it will be up to your design to make sure that you account for this. When going through the different availability requirements, make sure that you account for both the grid and infrastructure components. The following lists are some examples of availability resources that should be accounted for:

- Grid middleware
  - Workload management
  - Grid directory and indexing service
  - Security services
  - Data storage
  - Grid software clustering

- Networks
  - Load-balancing
  - High-availability routing protocols
  - Redundant and diverse network paths
- Security
  - Redundant firewalls
- Datastore
  - Mirroring
  - Data replication
  - Parallel processing
- Systems management
  - Backup and recovery
  - LDAP replicas
  - Alerts and monitoring to signal a failure within the environment

Every so often, different components necessary to the workflow process fail periodically and disrupt availability of the system. You can help mitigate the risk involved by eliminating the single points of failure within your environment through the use of redundant software or hardware components.

To give you a better idea of some different availability targets, the following list presents the expected system availability in a whole year:

- Normal commercial availability (single node): 99 - 99.5%, 87.6 - 43.8 hours of system down
- High availability: 99.9%, 8.8 hours of system down
- Fault resilient: 99.99%, 53 minutes of system down
- Fault tolerant: 99.999%, 5 minutes of system down
- Continuous processing: 100%, 0 minutes of system down

Keep in mind, however, that the redundancy that is added to the grid infrastructure will normally increase the costs within the infrastructure. It is up to the business to help justify the costs that would bring an environment from 99.9% availability per year up to 99.99% per year. While the difference in time between those two numbers is about eight hours, the costs associated may be too much to justify the increased availability.

Performance

The performance objective for a grid environment is to most efficiently utilize the various resources within the grid. Whether that includes spare CPU cycles, access to a federated databases, or application processing, it is up to you to match the performance goals of the business and design accordingly.

If your application can take advantage of multiple resources, you can design your grid to be broken up into smaller instances and have the work distributed throughout the grid. The goal is to take advantage of the grid as a whole in order to increase the performance of the application. Through intelligent workload management and scheduling, your application can take advantage of whatever resources within the grid are available. Part of the performance is based on the form of workload management to make sure that all resources within the grid are actively servicing jobs or requests within the grid.

4.2 Grid architecture models

There are different type of grid architectures to fit different types of business problems. Some grids are designed to take advantage of extra processing resources, whereas some grid architectures are designed to support collaboration between various organizations.

The type of grid selected is based primarily on the business problem that is being solved. Taking the goals of the business into consideration will help you choose the proper type of grid framework. A business that wants to tap into unused resources for calculating risk analysis within their corporate datacenter will have a much different design than a company that wants to open their distributed network to create a federated database with one or two of their main suppliers. Such different types of grid applications will require proportionately different designs, based on their respective unique requirements.

The selection of a specific grid type will have a direct impact on the grid solution design. Additionally, it should be mentioned that grid technologies are still evolving and tactical modifications to a grid reference architecture may be required to satisfy a particular business requirement.

4.2.1 Computational grid

A computational grid aggregates the processing power from a distributed collection of systems. A well known example of a computational grid is the SETI@home grid. This type of grid is primarily comprised of low powered computers with minimal application logic awareness and minimal storage capacity.

Rather than simply painting images of flying toasters, the idle cycles of the personal computers on the SETI@home grid are combined to create a computational grid used to analyze radio transmissions received from outer space in the “Search for Extra Terrestrial Intelligence.”

Most businesses interested in computational grids will likely have similar IT initiatives in common. While they probably will not want to search for extraterrestrials, there will likely be a business initiative to expand abilities and maximize the computer utilization of existing resources through aggregation and sharing. The business may require more computer capacity than is available. The business is interested in modifying specific vertical applications for parallel computing opportunities.

Additional uses for a computational grid include mathematical equations, derivatives, pricing, portfolio valuation, and simulation (especially risk measurement). Note that not all algorithms are able to leverage parallel processing, data intensive and high throughput computing, order and transaction processing, market information dissemination, and enterprise risk management. In many cases, the grid architecture model is not (yet) suitable for real-time applications.

Computational grids can be recognized by these primary characteristics:

- Made up of clusters of clusters
- Enables CPU scavenging to better utilize resources
- Provides the computational power to process large scale jobs
- Satisfies the business requirement for instant access to resources on demand

The primary benefits of computational grids are a reduced Total Cost of Ownership (TCO), and shorter deployment life cycles. Besides the SETI@home grid, the Distributed Terascale Facility (TeraGrid), UK, and Netherlands grids are all different types of computational grids. The next generation of computational grid computing will shift focus towards solving real-time computational problems.

4.2.2 Data grid

While computational grids are more suited for aggregating resources, data grids focus on providing secure access to distributed, heterogeneous pools of data. Through collaboration, data grids can also include a new concept such as a federated database. Within a federated database, as illustrated in Figure 4-1 on page 89, a data grid makes a group of databases available that function as a single virtual database. Through this single interface, the federated database provides a single query point, data modeling, and data consistency.

Data grids also harness data, storage, and network resources located in distinct administrative domains, respect local and global policies governing how data can be used, schedule resources efficiently, again subject to local and global constraints, and provide high speed and reliable access to data. Businesses interested in data grids typically have IT initiatives to expand data mining abilities while maximizing the utilization of an existing storage infrastructure investment, and to reduce the complexity of data management.

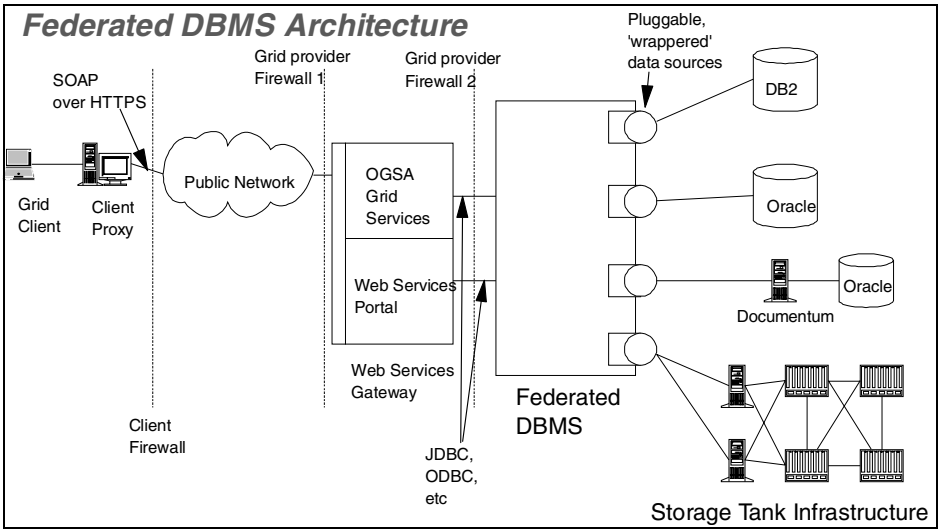

Figure 4-1 Federated DBMS Architecture

4.3 Grid topologies

A topology view (see Figure 4-2 on page 90) covers the following spectrum of grids:

- Intragrids
  - Single organizations
  - No partner integration
  - A single cluster
- Extragrids
  - Multiple organizations
  - Partner integration
  - Multiple clusters

- Intergrids
  - Many organizations
  - Multiple partners
  - Many multiple clusters

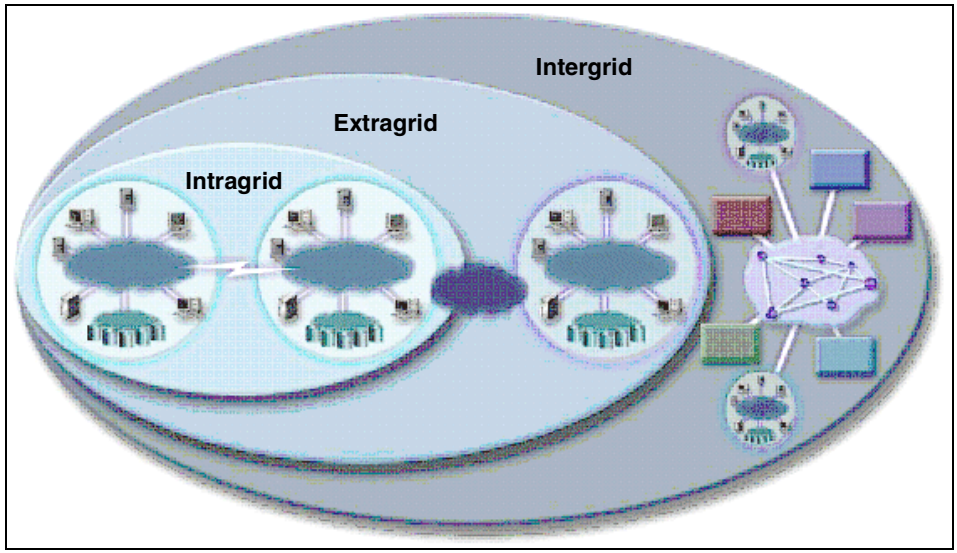

Figure 4-2 Intragrids, extragrids, and intergrids

The simplest of the three topologies is the intragrid, which is comprised merely of a basic set of grid services within a single organization. The complexity of the grid design is proportionate to the number of organizations that the grid is designed to support, and the geographical parameters and constraints. As more organizations join the grid, the non-functional or operational requirements for security, directory services, availability, and performance become more complex.

As more organizations require access to grid resources, the requirements for increased application layer security, directory services integration, higher availability, and capacity become more complicated. The philosophy behind grid computing is transparent, secure, and coordinated resource sharing and problem solving in dynamic, multi-institutional organizations. The resource sharing alluded to is not primarily file exchange but rather direct access to computers, software, data, and other resources, as is required by a range of collaborative problem-solving and resource-brokering strategies emerging in industry, science, and engineering. This sharing is, necessarily, highly protected, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs.

4.3.1 Intragrid

A typical intragrid topology, as illustrated in Figure 4-3, exists within a single organization, providing a basic set of grid services. The single organization could be made up of a number of computers that share a common security domain, and share data internally on a private network. The primary characteristics of an intragrid are a single security provider, bandwidth on the private network is high and always available, and there is a single environment within a single network. Within an intragrid, it is easier to design and operate computational and data grids. An intragrid provides a relatively static set of computing resources and the ability to easily share data between grid systems. The business might deem an intragrid appropriate if the business has an initiative to gain economies of scale on internal job management, or wants to start exploring the use of grid internally first by enabling vertical enterprise applications.

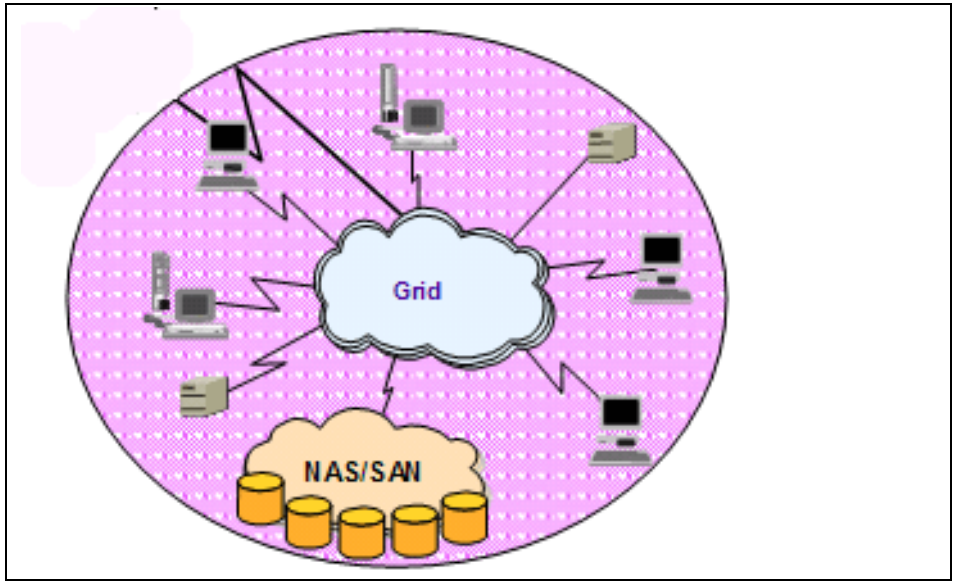

Figure 4-3 An intragrid

4.3.2 Extragrid

Based on a single organization, the extragrid expands on the concept by bringing together two or more intragrids. An extragrid, as illustrated in Figure 4-4 on page 92, typically involves more than one security provider, and the level of management complexity increases. The primary characteristics of an extragrid are dispersed security, multiple organizations, and remote/WAN connectivity. Within an extragrid, the resources become more dynamic and your grid needs to

be more reactive to failed resources and failed components. The design becomes more complicated and information services become relevant to ensure that grid resources have access to workload management at run time.

A business would benefit from an extragrid if there was a business initiative to integrate with external trusted business partners. An extragrid could also be used in a B2B capacity and/or to establish relationships of trust.

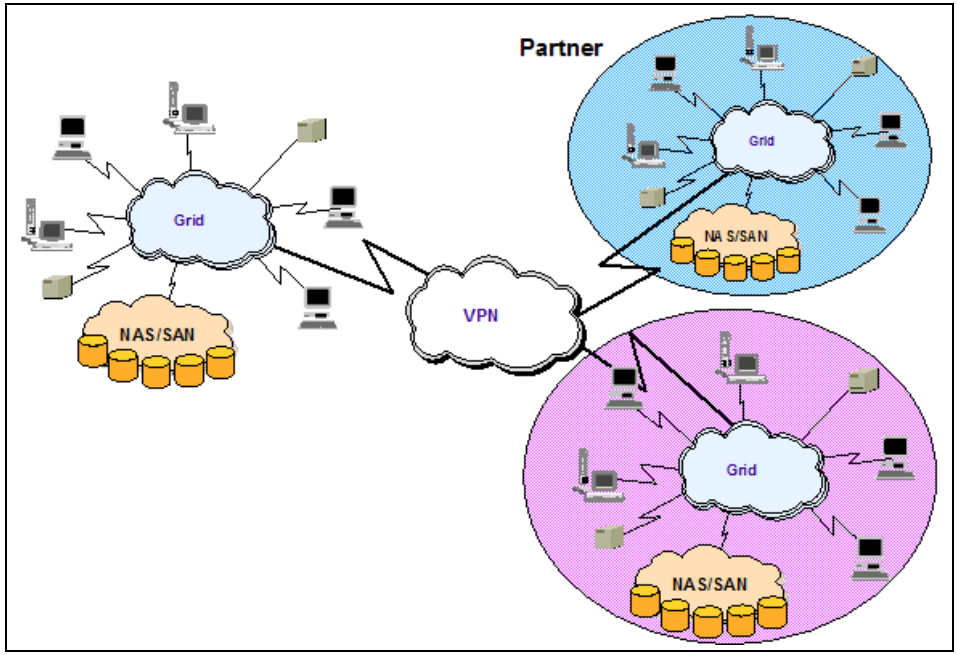

Figure 4-4 Extragrids usually exist in several organizations and security providers

4.3.3 Intergrid

An intergrid requires the dynamic integration of applications, resources, and services with patterns, customers, and any other authorized organizations that will obtain access to the grid via the internet/WAN. An intergrid topology, as illustrated in Figure 4-5 on page 93, is primarily used by engineering firms, life science industries, manufacturers, and by businesses in the financial industry. The primary characteristics of an intergrid include dispersed security, multiple organizations, and remote/WAN connectivity. The data in an intergrid is global public data, and applications, both vertical and horizontal, must be modified for a global audience. A business may deem an intergrid necessary if there is a need for peer-to-peer computing, a collaborative computing community, or simplified end to end processes with the organizations that will use the intergrid.

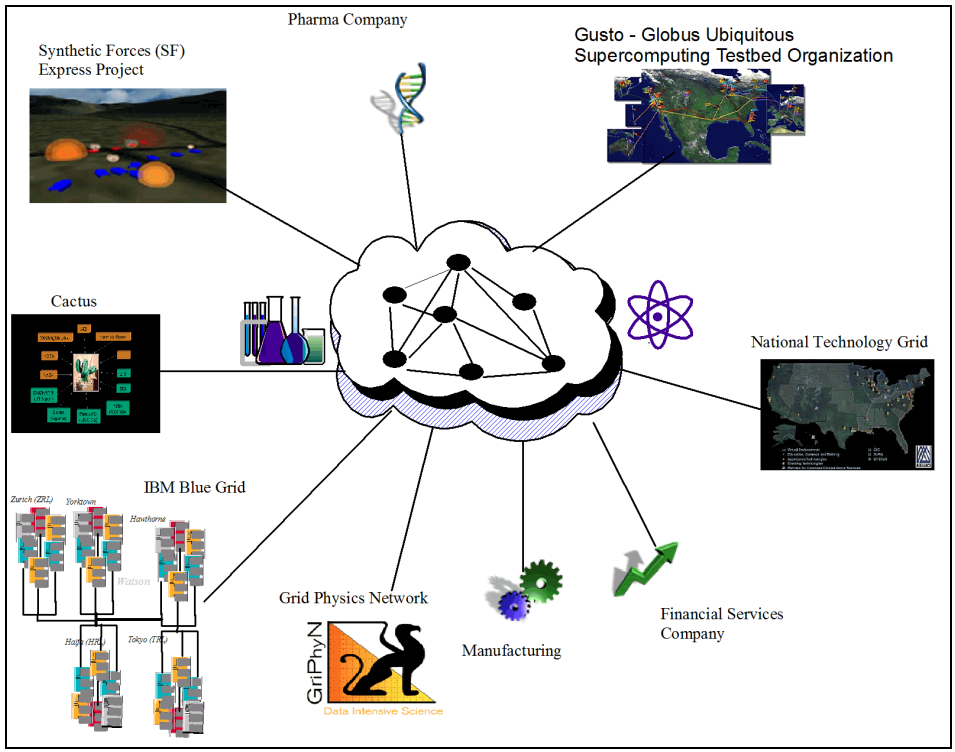

Figure 4-5 Intergrid

4.3.4 E-utilities

Instead of having to buy and maintain the latest and best hardware and software, customers will have the flexibility of tapping into computing power and programs as needed, just as they do gas or electricity. But enterprises are coming more and more to see the e-sourcing trend as a continuum - reaching beyond commonplace IT resources on demand to the delivery of business process and management functions integral to the way the organization works.

The "e-sourcing" business model is based on providing the components of IT function that are (largely) standardized and delivered through a service provider model. The attributes of this model include a distributed and shared environment, and generally standardized non-core business processes. The e-utility is used by consumers of the e-utility as building blocks for developing complex e-business solutions. The major properties of e-sourcing environments are a standard solution that requires minimal configuration, pooled resources used to serve multiple customers, capacity on demand and scalable, 24x7, always on, high availability, rapidly deployable, minimal operations overhead, shared systems

management, flexible pricing and billing based on either actual usage/consumption of resources, or a calculated flat rate subscription.

4.4 Phases and activities

Deciding which grid type and topology to chose from is just the first step in the grid architecture design. A mature end-to-end design methodology is comprised of distinct phases and activities. The activities in the architecture design phase of the project include a review of the detailed architectural decisions and design documentation for the current infrastructure, conducting interviews and workshops, the modification of the initial high level design based on new requirements and the results of the detailed assessment, the creation of a detailed nodular architecture design, and the creation of the implementation and transition plan.

4.4.1 Basic methodology

For building a grid architecture, using a basic methodology allows the design to follow a consistent path from beginning to end. A methodology is not a cookbook for building a grid architecture, but a way to trace the progress of the design from the kickoff meeting to the final end-state. The methodology follows a reproducible set of guidelines that can be used over again based on a set of successful guiding principals for architecture design. A methodology allows the architecture to follow a set of principals that can be documented from beginning to end throughout the design.

We define a basic design methodology for developing the grid conceptual architecture in the next three sections.

Understanding the business drivers

The first step of any design is to identify and document the business drivers that are the foundation behind building the grid. The business drivers will outline the investment and what the end state will accomplish. The business drivers or business strategy are the foundation or reasoning behind building the grid. Whether the goal is to tie together or build a federated database with your suppliers or tie together a set of computers to harness their overall processing power, you should have an end goal in mind before the design begins.

Requirements gathering

The requirements gathering process will help drive the architecture process by helping the technical team work within a set of guidelines for the architecture. By following this process, all of your decisions can be tied back to the basic

requirements and business drivers for the design. Along with your solution objectives, the requirements will offer a road map for you to follow work through the design phases.

- Business requirements

  The business requirements are a subset of the business drivers that are focused on solving a specific business need. The business requirements drive important areas within the design, such as the performance and availability of the environment. Helping to understand these key service levels are important part of the design.

- Infrastructure requirements

  The infrastructure requirements provide the basic framework for how the infrastructure will be designed. There are many different variables for how the grid architecture can be designed and, based on what the requirements will be, will shape how the environment will look.

- Application requirements

  There are many factors that need to be accounted for during the design and the application is one of them. Possibly one of the most important requirements that must be validated is to ensure that the application in question can be made “grid-aware.” Unless the application can take advantage of the grid resources or split the workload across multiple components, the power of the grid is wasted.

Validate requirements

During the course of some designs, the requirements can change at the last minute or may go undiscovered. Requirements also have a way of changing when you least expect them to, so it is always a good idea to validate them before you proceed. Validating the requirements one last time before the design phase begins is a good way to ensure that all parties agree with the direction of the design.

4.4.2 Recommended steps

The following sections deal with additional recommended methods for developing an optimal grid design. These methods include attending grid design workshops and building prototypes once the design has been completed.

Grid design workshops

The purpose of the grid design workshops is to help all of the parties involved to better understand the variables, options, and considerations that need to be taken into account when developing a grid infrastructure design. Many or most of the grid middleware, technologies, and system components are probably new to

many people within the design team and it is always a good idea to hear firsthand from experienced IT professionals the means by which grid infrastructures can be implemented, as well as any pitfalls to watch out for when designing environments for grid computing.

Documentation

An extremely critical means of communicating the design (your “solution”) of your grid infrastructure is via an architecture or solution document. The solution document should start with a high-level overview of the environment and subsequently should drill down into the most detailed configuration diagrams and descriptions as possible. You will want to include things like IP addresses, network routes, server names, server architectures, network hardware, and essentially everything you know about the infrastructure at the time your design is completed. In truth, architecture documents are often dynamic, changing as the needs of the system users change and as technologies mature, become obsolete, and are replaced by newer technologies. You should revise your architecture document upon further hardware and software updates so that it accurately reflects the state of the system. Without an accurate architecture document, the system implementation team may get easily confused and not produce the system that was originally designed. Additionally, anyone adding further design changes to the system after the original system architect has moved on will appreciate an up-to-date architecture document, as it will save him or her countless hours of information gathering that would be necessary without an architecture document.

Prototype

Building a prototype of a grid system can save significant time that would otherwise be spent debugging and re-tooling unforeseen system incompatibilities. Your goal in building a prototype should be to produce a small scale, end-to-end backbone of what your production environment will look like. It should include all interoperating technologies and/or architectures, so that if any incompatibility exists, it will be apparent before the production system is implemented. When all of the kinks are ironed out of your prototype, you will be confident that all of your components will work together properly in your designed infrastructure, and, additionally, you will have some experience in the implementation of such a system. Lessons learned from building the prototype should be reflected in your architecture document and any other directions provided to the implementation team.

4.5 A conceptual architecture

The purpose of the grid conceptual architecture is to establish a common understanding between the business owners and the people architecting and

designing the grid infrastructure by describing the grid architecture that will support the client business requirements.

This section will highlight some of the common components that you can choose from within the Globus Toolkit. If you are designing a grid architecture using different grid middleware software from Platform, DataSynapse, Avaki, or any other grid software provider, this section should still give you a head start on grid architecture. You will still be faced with decisions on the basic components, such as the security models, workload management, information services, and data sharing.

The conceptual model is a high-level framework consisting of the grid system components and nodes within the design. The nodes represent the different system components and grid middleware that make up the design. Normally, the conceptual model is the first graphical view of the grid infrastructure and is used as a stepping-stone to building a detailed configuration for the grid network. The graphic depiction of the grid environment will allow you to see how the requirements were gathered and how the many grid components will interact with one another.

4.5.1 Infrastructure

The infrastructure represents the physical hardware and software components used to interconnect different grid computers. These components help support the flow of information between grid systems and provide the basic set of services for connectivity, security, performance availability, and management. While many of these infrastructure components supply basic functionality to the grid, many are optional. It will be up to you to decide on the requirements and how well these components match up to the needs of your design.

Security

The use of firewalls can provide logical and secure segmentation between grid systems. You might want to use firewalls to protect your networks and grid servers by limiting the types of services and protocols that connect to your computers. By using firewalls within your grid design, you can help limit the network communication between grid systems and only use protocols that you specify that the firewall will support.

Firewalls are not the only answer to protecting your grid servers, but they do add an additional layer of defense from internal or external users trying to access your systems. Firewalls work by controlling access to network services that you grid computers will be running. Since the network offers a gateway to your grid systems, you want to make sure that you control exactly the services and protocols and from where to whom on your network.

For the most up to date information regarding the Globus Toolkit and firewalls, you should check out the firewall section on the Globus Web site at:

http://www.globus.org/security/

Some areas you may want to protect within your design are:

- Certificate Authority / Registrant Authority
- Globus Toolkit components, such as MDS, GRIS, and GIIS (for more information about these and other Globus Toolkit components, refer to 7.2, “Components of Globus Toolkit” on page 133.)
- Databases
- All grid servers

Networks

The network design within the grid architecture can take on many different shapes. The networking components can represent the LAN or campus connectivity or even WAN communication between the grid networks. Whatever the case may be, the network’s responsibility is to provide adequate bandwidth for any of the grid systems. Like many other components within the infrastructure, the networking can be customized to provide higher levels of availability, performance, or security.

Grid systems are for the most part network intensive due to security and other architectural limitations. For data grids in particular, which may have storage resources spread across the enterprise network, an infrastructure that is designed to handle a significant network load is critical to ensuring adequate performance.

Systems management

Any design will require a basic set of systems management tools to help determine availability and performance within the grid. A design without these tools are limited in how much support and information can be given about the health of the grid infrastructure. Some networks within a grid architecture can be dedicated to perform these functions as to not hamper the performance of the grid.

Storage

The storage possibilities are endless within a grid design. How that storage will be secured, backed up, managed, and replicated are some of the questions that the grid design will try to answer. Within a grid design, you want to make sure that your data is always available to the resources that need it. Besides availability, you want to make sure that your data is properly secured, as you would not want unauthorized access to sensitive data. Lastly, you want more

than decent performance for access to your data. Obviously, some of this relies on the bandwidth and distance to the data, but you will not want any I/O problems to slow down your grid applications. For applications that are more disk-intensive, or for a data grid, more emphasis can be placed on storage resources, such as those providing higher capacity, redundancy, or fault-tolerance.

4.5.2 Conceptual components

Within this section, we will incorporate the different infrastructure and grid middleware components with a sample grid architecture design. This design sets an example of the basic sets of services that the Globus Toolkit can offer and how the different infrastructure components interact with the grid middleware. Within this sample design, our basic framework is set within a single location and within a single security provider. Our basic design, as illustrated Figure 4-6 on page 100, begins with a simple understanding of the components that can then be expanded based on the requirements. To review, our design encompasses the following:

- Intragrid topology and departmental access
- Single security provider
- Basic set of Globus Toolkit components (see Chapter 7, “Components” on page 131 for more detail about Globus components)
  - Grid clients
  - Certificate Authority
  - Gatekeepers
  - MDS (GRIS/GIIS)
  - Digital certificates for authentication
- Basic workload management (job scheduler)
- Common infrastructure devices
  - Management network
  - Firewalls/host intrusion detection
  - Network connectivity
- Storage Area Network
- Network Time Protocol Server

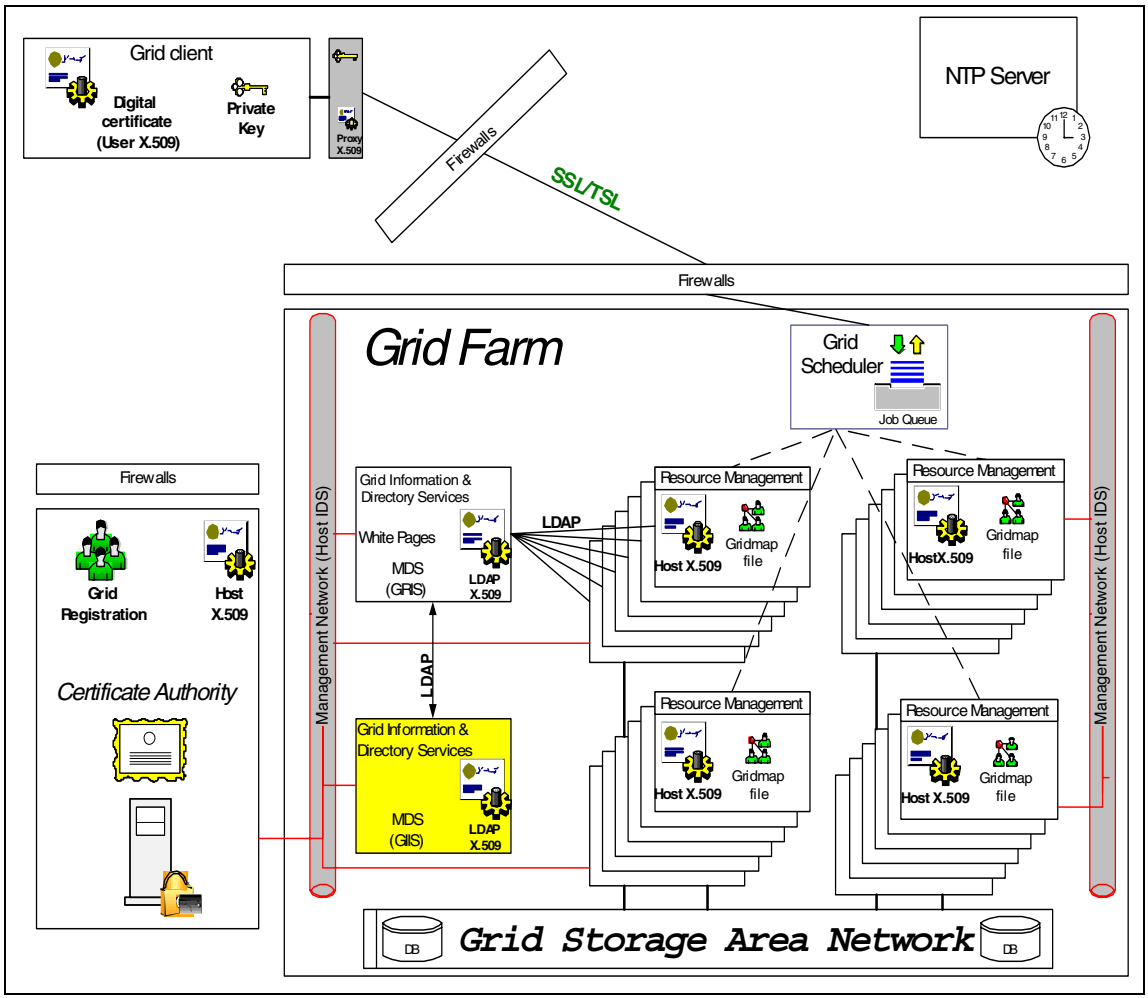

Figure 4-6 Conceptual components

Security functions

The security functions within the grid architecture are responsible for the authentication, authorization, and secure communication between grid resources. Based on a variety of security mechanisms, including PKI or Kerberos, clients gain access to the grid by registering with the security provider. To ensure the confidentiality and integrity of your data, encryption between grid systems is strongly encouraged.

Resource management

The resource management acts as an abstract interface to heterogeneous resources of the grid. This allows the grid computers to access dispersed resources within the grid, regardless of the platform. The resource management helps works with the different authentication mechanisms to map grid clients to resources within the grid.

Information services

The information services are an effective way for resources within the grid to cope with the dynamic nature of the grid. Within any grid, both CPU and data resources will fluctuate, depending on their availability to process and share data. As resources become free within the grid, they can update their status within the grid information services. This provide clients information to make intelligent decisions on which grid resources are free to use.

As opposed to the grid client itself sending off jobs to specific computers on the grid, a grid client would send job requests to the workload management component of the grid. The workload management would route the job to the necessary grid servers that are available to do the work. Based on a queuing system, the jobs would flow to a scheduler and be processed upon request. This area alleviates some performance bottlenecks with the grid by having the scheduler manage the routing of requests instead of the grid client.

Data movement

Your most important aspect asset within your grid is still your data. Whether you are building a computational grid or data grid, this still remains the same. Within your design, you will have to determine your data requirements and how you will move data around your infrastructure. Standardizing a set of grid protocols will allow you to communicate between any data source that is available within your design.

You also have choices for building a federated database to create a virtual data store or other design options for storing your data. Other options include Storage Area Networks, network file systems, and dedicated storage servers.

Part 3

Products

5

Grid software

This chapter presents an overview and introduction to several grid computing and grid related products. Included in this chapter is a discussion of available software platforms that can be used to implement an enterprise grid.

5.1 Grid computing products overview

If you did a search on the Internet for grid computing products, the results would be endless. It would be impossible to mention all of the products available in this redbook. Instead, we can only focus on a select number of products that are considered to be among the leaders in the grid computing arena. The first area in discussing grid computing products are those products that actually provide a software platform for computing grids. Included in this discussion are products available from vendors such as Avaki, Data Synapse, Entropia, United Devices, Globus, and Platform Computing. The second area focuses on those products that provide utilities or services on top of a computing grid. These components will be discussed in Chapter 6, “Additional components” on page 115.

What is meant by a product providing a software platform is that the product provides the base features one would need to construct a grid. There may be additional features that could be provided by products of the second type, but the first type immediately gives you the framework for creating a grid. There are different types of grids, which are discussed in Chapter 4, “Design” on page 81. The type of grid that most products today create is a computational grid. In creating this type of grid, there are different approaches one can take. For example, an enterprise may wish to scavenge CPU cycles from existing desktops throughout the enterprise. Alternatively, an enterprise may wish to have dedicated servers for use in a computational grid. Finally, an enterprise may choose to both scavenge existing desktops and establish dedicated resources for the computational grid. DataSynapse, Entropia, and United Devices are three vendors who supply products that take the desktop scavenging approach. Avaki and Globus supply products take more of the dedicated resource approach to their computational grid. Platform Computing has a suite of products that handle both desktop scavenging and dedicated resources. Another grid type discussed in Chapter 4, “Design” on page 81 is the data grid. Avaki supplies both a computational grid and a data grid.

5.2 IBM Grid Toolbox (Globus)

The IBM Grid Toolbox is a package available from IBM that includes the Globus Toolkit. Also included in the package is additional documentation and custom installation scripts written for IBM @server hardware running Linux and AIX. These install scripts include the installp images for AIX and RPMs for Linux. Also included in the documentation are the steps for the installation, configuration, and integration of LoadLeveler (AIX only), using a GASS server, and setting up MDS. Globus 2.0 on zSseries is also available as a part of the SuSE Linux Enterprise Server 8 (SLES 8).

The Globus Toolkit is an open source software with development being led by Argonne National Laboratory, the University of Southern California, and the University of Chicago. There are three main components in the Globus Toolkit:

- Resource management
- Information services
- Data management

The technologies used to realize these three components include Grid Resource Allocation Management (GRAM), Monitoring and Discovery Service (MDS), and Grid File Transfer Protocol (GridFTP). All of these components utilize the Grid Security Infrastructure (GSI) protocol for security at the connection layer. For a further explanation of these components, refer to the Chapter 7, “Components” on page 131.

5.3 Avaki

Avaki is a privately held software vendor located in Cambridge, Massachusetts. Avaki offers a grid platform, which supplies both a computational and a data grid. Avaki 2.5 is supported on Windows NT/2000, Linux, Tru64, AIX, Solaris, and IRIX. Avaki, as illustrated in Figure 5-1 on page 108, is designed more for dedicated resources and existing servers versus desktop scavenging. In the Avaki platform, there are a bootstrap host, service hosts, and command clients.

The bootstrap host is installed first to set up and initialize the grid. After the bootstrap host is installed, service hosts may be installed and join the grid established by the bootstrap host. Also, if a user just wishes to have access to the Avaki grid without enrolling their machine, they can install a client machine. Once the Avaki grid is established, the administrator may add users to the Avaki grid. Users may then log on to the Avaki grid to submit jobs and access data files located in the Avaki grid.

Avaki supports existing applications and does not require any modifications to be made to the code. A user may log in to the Avaki grid and then simply register their application in the grid. Once registered, users may run their application interactively or submit them as batch jobs to a queue. Avaki can also be configured to interoperate with existing schedulers or queues, like Platform’s LSF.

In addition to this computational grid capability, Avaki also supplies data grid capabilities. Avaki users may log in to the Avaki grid and either copy data files into the Avaki grid or share local data files in the Avaki grid. When these data files are copied in, they may be stored on any resource in the grid, but the user does not care which machine the file is located on, but is only concerned with the

path to the file in the global directory. The user may not even have access to the machine where the file is stored, but will have access to the file through the Avaki data grid.

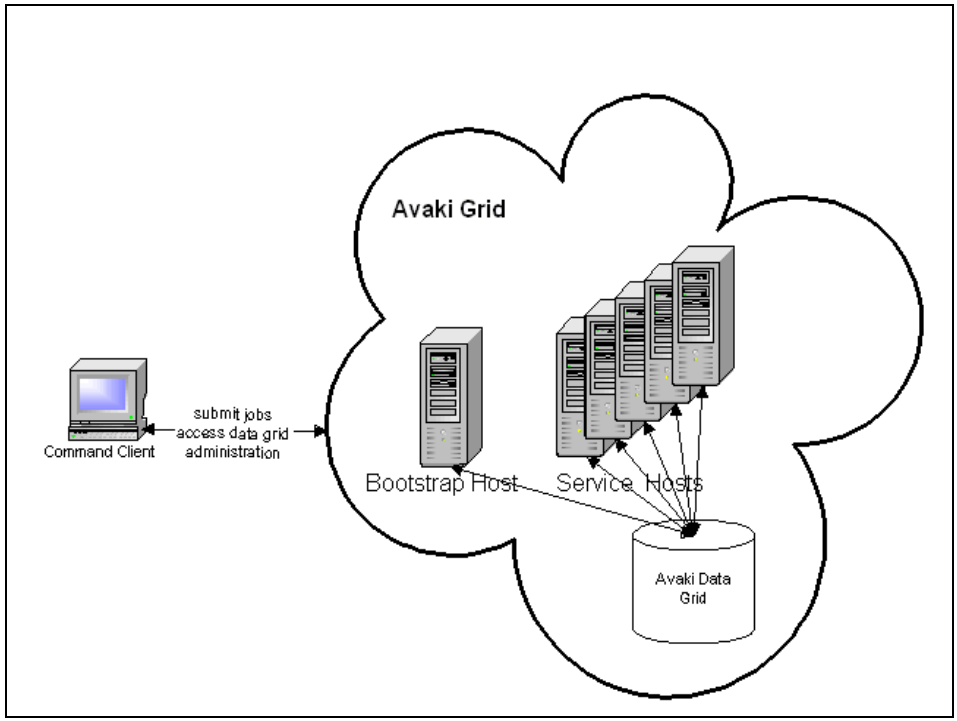

Figure 5-1 Avaki

For more information on Avaki, please visit http://www.avaki.com.

5.4 DataSynapse

DataSynapse is a privately held software vendor located in New York, New York. They supply a CPU scavenging platform known as LiveCluster. LiveCluster is supported on Windows, Linux, and Solaris. The LiveCluster platform, as illustrated in Figure 5-2 on page 109, consists of three major components: a Server, Engines, and a Driver.

The Server is responsible for system administration and scheduling. There is a Web console interface to the server that allows for performing system administration remotely. Engines can be installed on either desktops or dedicated servers. When an engine determines that a system is idle, it will request a job from the server. If an engine is interrupted by the desktop user

while working on a job, it will suspend the job and the server will restart it on another engine. This way, the desktop user's work is not affected. The driver is typically embedded in an application and is responsible for submitting jobs to the server and receiving the output. Drivers can also send administrative commands to the server to obtain status.

LiveCluster supports existing applications with only a little integration work required. In addition, existing applications may be rewritten, or new applications written, using the LiveCluster APIs to take full advantage of all of the features available on the LiveCluster platform. The LiveCluster APIs are available in Java, C++, and XML.

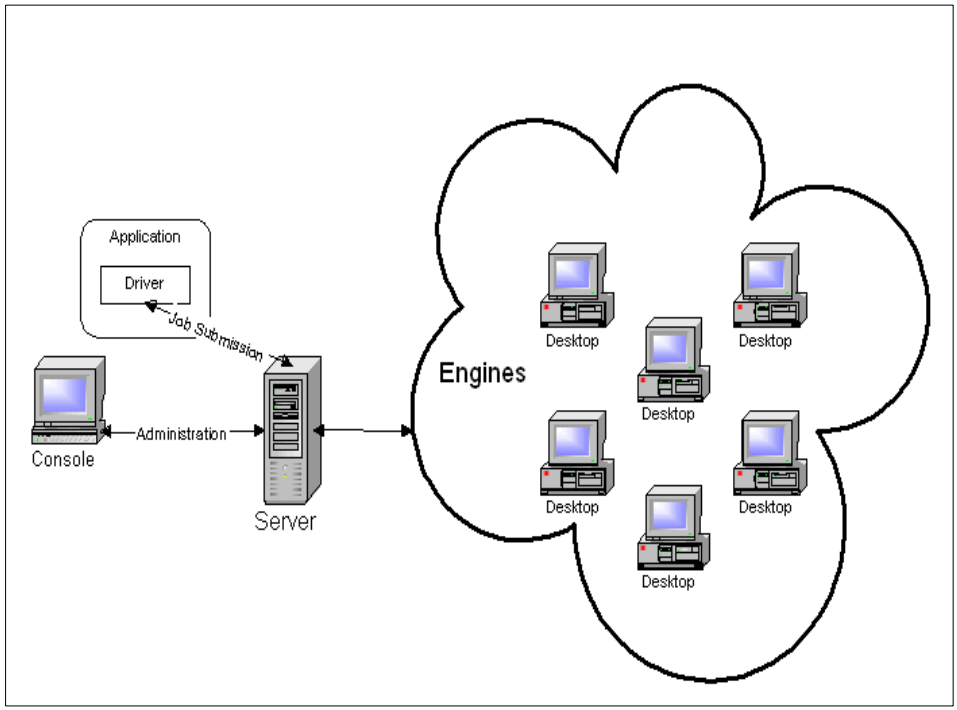

Figure 5-2 DataSynapse's LiveCluster

For more information, please visit http://www.datasynapse.com/.

5.5 Entropia

Entropia is a grid computing vendor located in San Diego, California. Entropia's product, DCGrid, provides a means to scavenge CPU cycles from existing Windows desktops in an enterprise. There are three main components in

DCGrid: the DCGrid Clients, DCGrid Manager, and the DCGrid Scheduler, as illustrated in Figure 5-3.

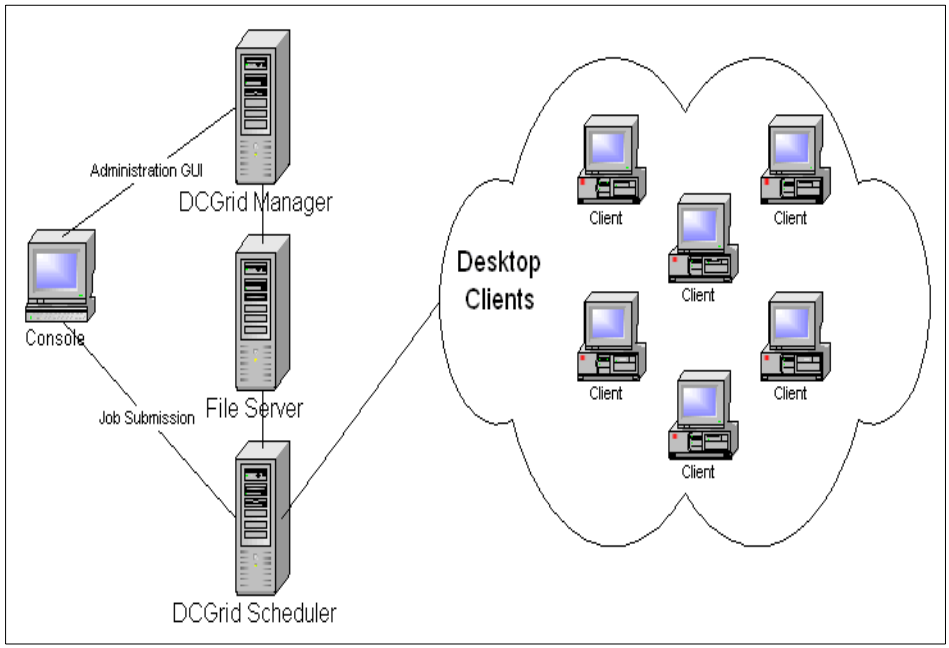

Figure 5-3 Entropia DCGrid

The DCGrid client is installed on all of the desired desktops in an enterprise. This client piece runs unobtrusively on the desktop, so the desktop user's normal work is not affected. The client will only work on a single job at any given time. Also, if a desktop machine cannot dedicate enough resources to that job because the desktop user is using too much of a resource (CPU, disk, and memory), the job will be paused or stopped and run on another client. The client also contains a “sandbox” for jobs to run in. This is an isolated area for the application to run in and the application cannot access any files outside of this “sandbox”. In addition, all application and data files stored on a client machine are encrypted with Triple-DES encryption. These two facts not only protect the application from the desktop user, but also protect the desktop user from the application. The server components of DCGrid consist of the DCGrid Scheduler and the DCGrid Manager. The DCGrid Scheduler is responsible for dispatching jobs to the client desktop. The DCGrid Manager offers a Web interface for managing the enterprise PC grid. This Web interface allows the administrator to monitor the status of jobs and clients, and manage user access control. The interface can also provide users with a Web based method for job submissions.

DCGrid supports any Win32 application and requires no alterations to the source code. However, not all types of applications are well suited for the DCGrid platform. Applications must exhibit certain characteristics to benefit from being run through DCGrid. Applications that are CPU intensive and highly parallel are good candidates for the platform. Serial applications and MPI applications are not good candidates for the DCGrid platform. The last two statements are true for any CPU scavenging approach.

For more information on Entropia please visit http://www.entropia.com.

5.6 Platform Computing

Platform is a privately held software vendor located in Toronto, Canada. Platform supplies many distributed computing products that relate to grid computing. This section will briefly discuss LSF, ActiveCluster, and MultiCluster. In addition to these products, Platform also offers Platform Globus. Platform Globus includes the software for the Globus Toolkit as well as documentation and support.

LSF, as illustrated in Figure 5-4 on page 112, is a product in Platform's suite of workload management products. LSF is supported on AIX, IRIX, Tru64, HP-UX, Solaris, Linux, and Windows. LSF can be installed across a heterogeneous group of servers and balance an organizations workload across these servers. LSF consist of three main components:

- Master host
- Server host
- Client host

Their is one master host per cluster and this host is responsible for scheduling all jobs on its cluster. A server host is any machine that is running the LSF daemons and can actually execute jobs. A client host is a machine that is not running any LSF daemons and can only submit jobs and run LSF commands. Client hosts cannot execute LSF jobs. LSF supplies multiple types of schedulers and also allows for custom schedulers to be developed to take an organizations policies into account. LSF supports existing applications and requires no changes to the code. LSF also provides a rich set of APIs for application development.

ActiveCluster is an extension of LSF that can be used for Windows desktop scavenging. ActiveCluster requires LSF, because it uses the LSF scheduler. A server host in an existing LSF cluster can be converted to an ActiveCluster host and is responsible for sending jobs to the ActiveCluster desktops. ActiveCluster provides a way to harness unused cycles from desktop machines in an enterprise.

MultiCluster is another extension of LSF that is used to link together multiple LSF clusters. A large enterprise may have multiple LSF clusters owned by different departments within the enterprise. MultiCluster allows an enterprise to balance its workload not only across the individual department cluster, but across all of the clusters in an enterprise. There are two approaches to this scenario available with MultiCluster. The first is a department whose cluster needs additional resources, so it may send jobs to a department whose cluster has idle resources. The second approach is for the department with underutilized resources to lease machines from its cluster to departments whose cluster needs additional resources.

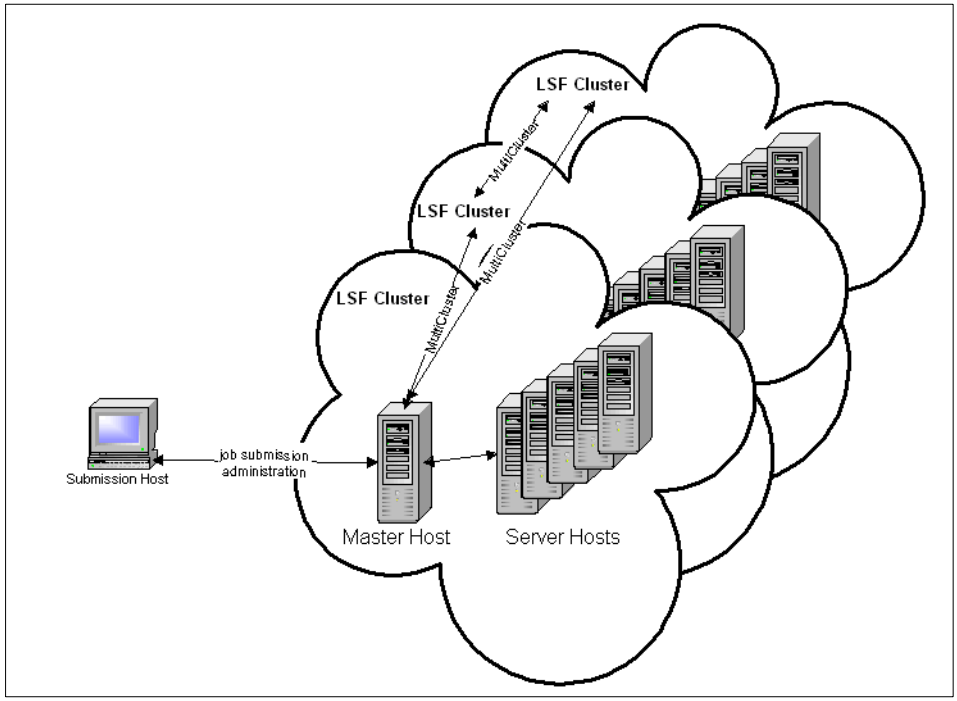

Figure 5-4 Platform LSF

For more information on Platform, please visit `http://www.platform.com/`.

5.7 United Devices

United Devices is a software vendor located in Austin, Texas. They supply a CPU scavenging platform called the MetaProccessor Platform. The MetaProcessor Platform, as illustrated in Figure 5-5 on page 113, consists of two main parts, a server part, known as the MP Server, and a client part, known as

UD Agents. In addition to the MP Server, there is a database server and a Web console to manage the MetaProcessor Platform.

The MP Server is supported only on Red Hat Linux, while the database server can be run on any operating system that is supported by IBM DB2. The UD agents are supported on Microsoft Windows 98, ME, NT, 2000, and XP, as well as Linux. The MP Server is responsible for scheduling jobs, collecting data, and managing the platform. The MP Console provides a Web interface to the server, which allows for remote management and administration. Jobs may also be submitted through this console and also through a command line job submission. The database server must be IBM DB2 and is used to store all the applications, and the statistics on jobs, users, agents, and so on. The UD Agents are installed on the desired desktops and are responsible for running the submitted jobs. The agent will only run one job at a time and at a low priority, so the desktop user's work is unaffected. The MetaProcessor Platform supports existing applications or applications can be written using the APIs to take full advantage of the platform.

United Devices has proven great scalability with their MetaProcessor Platform. Users can download and install the UD Agent and donate their spare desktop cycles to the United Devices community through the United Devices Web site. In turn, United Devices offers this processing power to non-profit research efforts, such as cancer research. There have been around 1.6 million downloads of the UD Agent, which shows this great scalability with the United Devices platform.

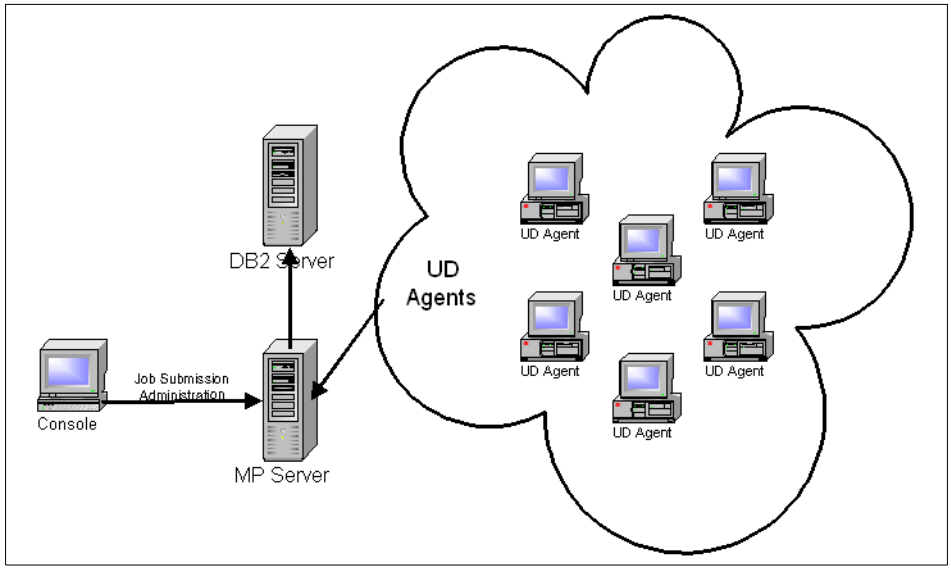

Figure 5-5 United Devices' MetaProcessor Platform

For more information on United Devices, please visit `http://www.ud.com`.

6

Additional components

This chapter provides information about some additional components that may be used in a grid computing environment.

At a glance, the following topics are discussed:

- Schedulers
- Data sharing
- Security
- Directory service
- License management
- Development tools

6.1 Schedulers

An important function in a grid is scheduling and load balancing. The grid platforms mentioned in Chapter 5, “Grid software” on page 105 all have scheduling capabilities, some more advanced than others. Some of the products, while having their own scheduler, also have the ability to be integrated with some existing schedulers. For example, the Globus Toolkit can be integrated with many schedulers, including PBS, Condor, and LoadLeveler. Some of these schedulers are discussed in the following sections. These schedulers are more cluster level schedulers. While the Globus Toolkit can be used to submit jobs to different local schedulers, there is still a piece missing. A higher level scheduler is needed to balance the load across these multiple clusters, as illustrated in Figure 6-1. This higher level scheduler is an area that is still being developed.

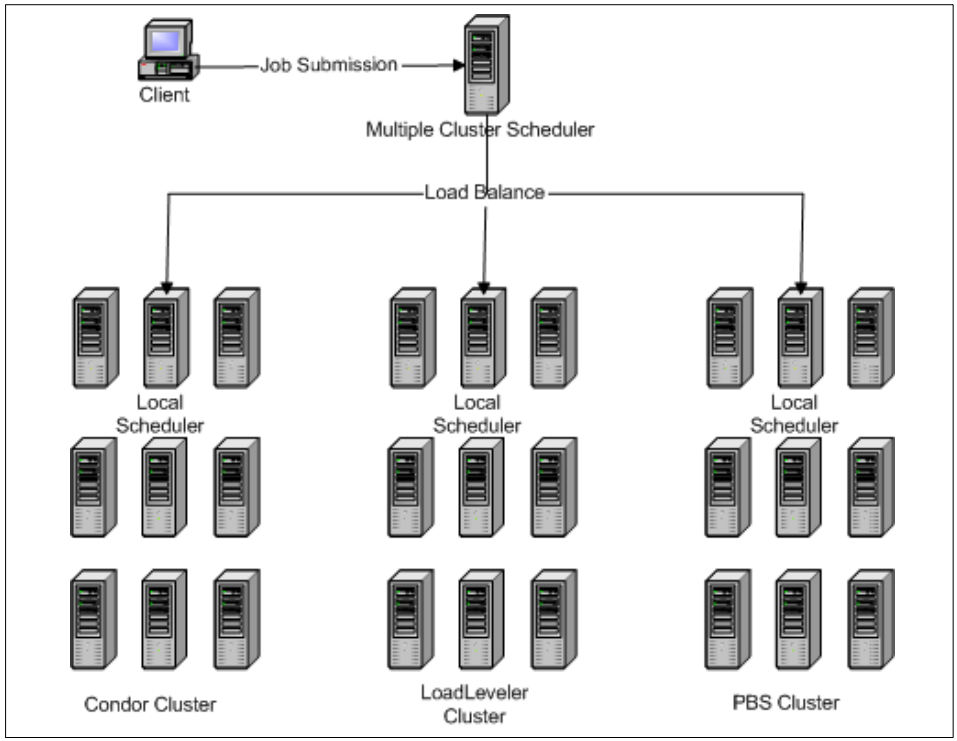

Figure 6-1 Local scheduler vs. higher level scheduler

6.1.1 Condor

Condor is workload management software that is developed by a research team located at the University of Wisconsin in Madison. The Condor research project

started in 1988. As illustrated in Figure 6-2 on page 118, there are three basic components in a Condor installation: a central manager, execution hosts, and submission hosts. A checkpoint server is an optional fourth component. The central manager serves two main functions in a condor cluster. The first function is collecting the status of all of the nodes in a Condor cluster. The second function is to match up resource requests for job submissions with a Condor node that can fulfill these requirements. Any node in a Condor cluster can be configured to be an execution machine and a submission machine, including the central manager. Execution machines are those nodes that can run Condor jobs and submission machines are those machines where Condor jobs can be submitted. An optional checkpoint server can be added to a cluster to store all of the checkpoint files for the jobs in the cluster.

There are nine daemons in a Condor cluster:

- There is a condor_master daemon, which watches over all other daemons. It is responsible for starting all daemons, restarting daemons that have stopped running, and stopping daemons at the request of the administrator. The condor_master daemon runs on all nodes in a Condor cluster.
- The condor_collector and condor_negotiator daemons both run on the central manager only. The condor_collector is the daemon responsible for keeping status for all of the nodes in a cluster. The condor_negotiator is responsible for matching up all resource requirements of jobs with available resources.
- The condor_startd daemon runs on any node in the cluster that is configured to be an execution machine. It contains the resource characteristics for a node and is used in matching to jobs.
- The condor_starter daemon is spawned by the condor_startd. This is the daemon that sets up the environment for the remote jobs and also monitors these jobs.
- The condor_schedd daemon runs on any machine in a cluster that is a submission machine. All user submitted jobs are submitted to queues that are managed by the condor_schedd.
- The condor_shadow daemon also runs on submission nodes and is responsible for managing running jobs.
- There is also an optional condor_ckpt_server daemon, which runs on the checkpoint server if a cluster is configured to use one. This daemon is responsible for storing and retrieving checkpoint files.
- There is a condor_kbdd daemon, which is only needed on Digital UNIX and IRIX machines. This daemon determines when there is keyboard and mouse activity.

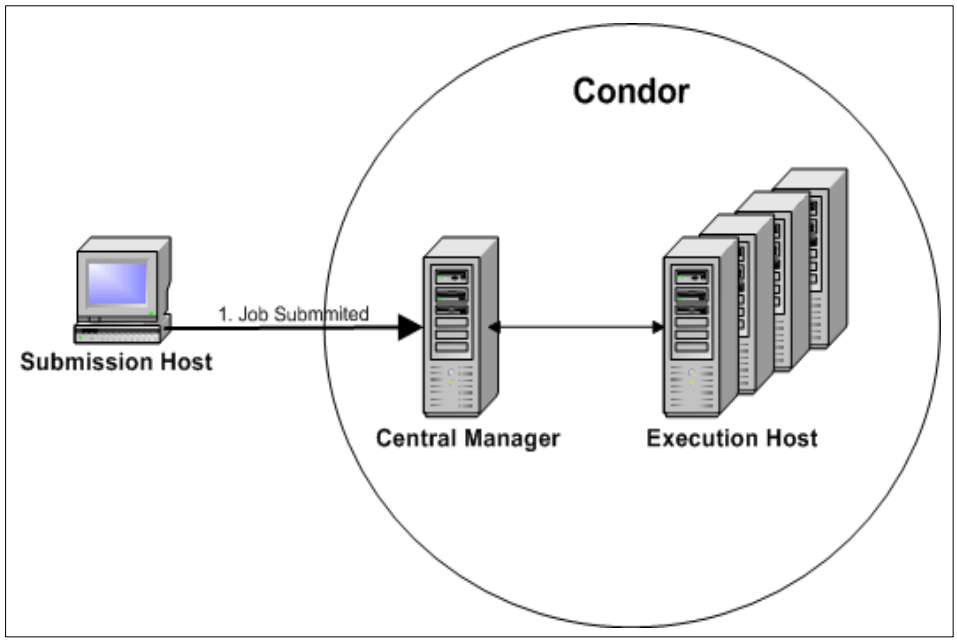

Figure 6-2 Condor

6.1.2 LoadLeveler

LoadLeveler is job management software available from IBM that can be used to schedule and load balance jobs across a cluster. LoadLeveler is supported on AIX. As illustrated in Figure 6-3 on page 119, a LoadLeveler cluster can be made up of four different types of machine servers: scheduling, central manager, executing, and submitting machines. One machine in a cluster is designated as the central manager. It is responsible for monitoring the status of the other nodes and also matching up job requirements with nodes that satisfy them. Any number of machines may be designated as submit only machines. This is a workstation, which is only able to submit jobs to a cluster and run commands that monitor the status of jobs and kill jobs. A submit only machine is not able to run any jobs submitted to a LoadLeveler cluster. Scheduling machines are cluster nodes that manage a queue and are responsible for scheduling submitted jobs. An executing machine is the node were a given job is run. A single machine may have more than one role.

There are six daemons in a LoadLeveler cluster:

- LoadL_master
- LoadL_schedd
- LoadL_startd

- LoadL_negotiator
- LoadL_kbdd
- LoadL_Smonitor

To illustrate the responsibility of each of these daemons, the job submission and execution process will be described. Jobs submitted to LoadLeveler are submitted to the schedd daemon. After being submitted, the schedd daemon notifies the negotiator daemon. The negotiator daemon is responsible for matching the job requirements with an available node in the cluster. The negotiator daemon monitors the status of all the nodes in a cluster and will use this information to make the match. Once the negotiator daemon matches a job with a cluster node, it will notify the schedd daemon of this match. The schedd daemon is then responsible for managing the job on this given machine. The startd daemon is responsible for spawning the job on the execution host. The startd daemon is also responsible for monitoring jobs and resources on the local node and reporting this to the negotiator daemon. The kbdd daemon monitors for keyboard and mouse activity on the local node. LoadLeveler can be configured to only run jobs if the keyboard and mouse are not being used. The gsmonitor daemon monitors for down machines in the cluster and will notify the schedd daemon so jobs that were running on machines that go down can be rescheduled. The master daemon runs on all nodes in a cluster and is responsible for all other daemons.

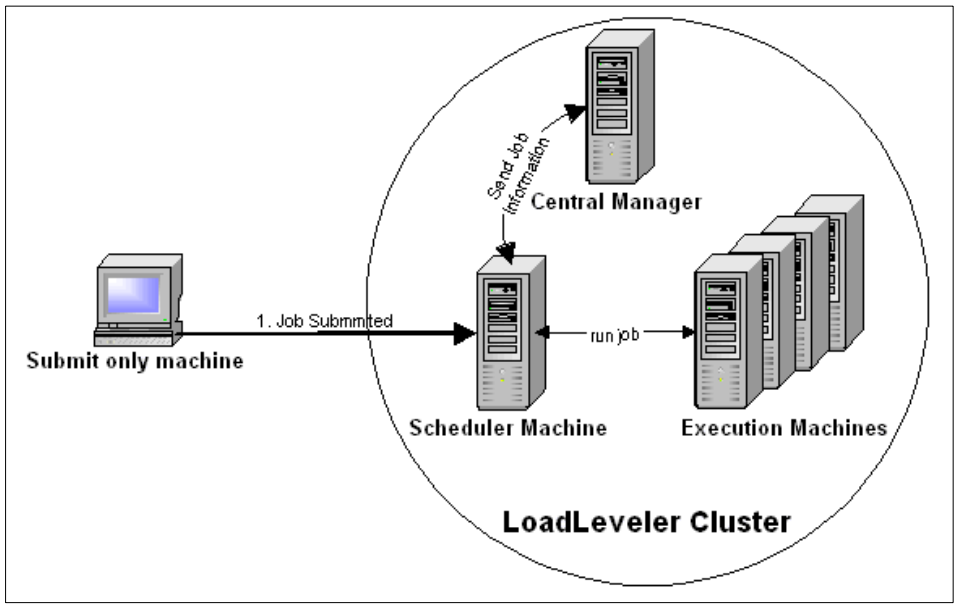

Figure 6-3 LoadLeveler

6.1.3 PBS

Portable Batch System (PBS) is a scheduler that was developed for NASA Ames Research Center by Veridian. There is an open source version available, OpenPBS, and a commercial version available, PBSPro. There is support available with the commercial version as well as additional features, such as better fault tolerance, advance reservation, and desktop cycle scavenging. The source is also available for the commercial version. As illustrated in Figure 6-4 on page 121, there are three main components that make up a PBS cluster:

- Job server
- Job executor
- Job scheduler

The job server is responsible for all batch processes in PBS, including creation, modification, running, and monitoring of batch jobs. A PBS cluster contains only one server machine.

The job executer is the machine in the cluster where a batch job is running.

The job scheduler is responsible for scheduling jobs on the cluster; it can query the status of execution nodes and also query the server to determine what jobs need to be run.

There are three daemons associated with these components:

- pbs_server
- pbs_mom
- pbs_sched

In a typical job submission, the pbs_server will notify the pbs_sched that a job needs to be run. The pbs_sched request status from the pbs_mom on all of the execution hosts. Once this status is returned to the pbs_sched, it will request the job requirements from the pbs_server. The pbs_sched decides where to run the job and sends the job to the pbs_mom on that machine.

In PBS, all jobs are submitted to queues. These queues and the scheduler can be configured using the policies of an organization. There are several different schedulers available in PBS. These schedulers may be configured by the administrator to comply with the organization's policies. Custom schedulers may also be developed by the administrator for a tighter integration with existing policies.

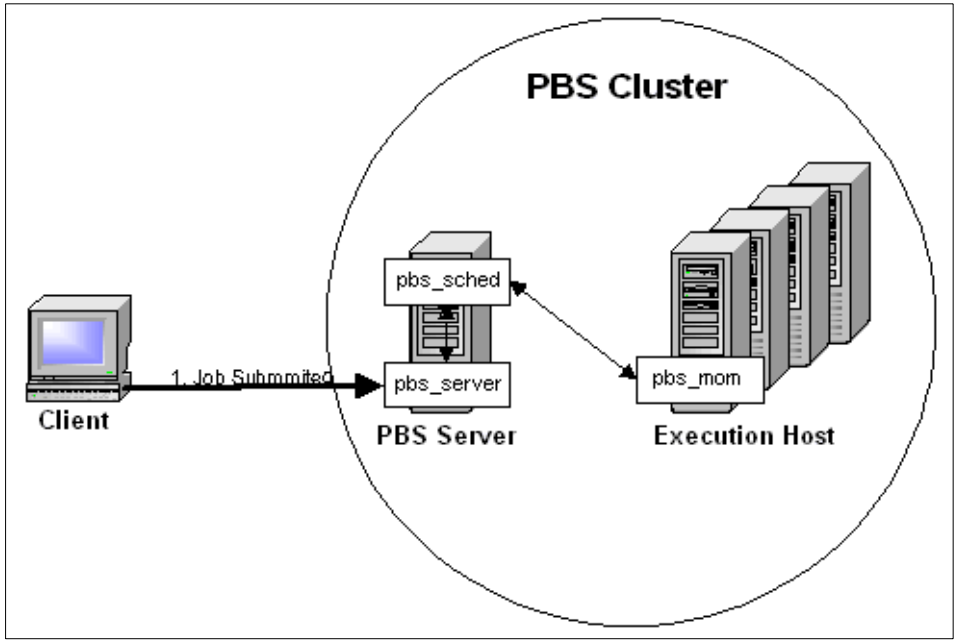

Figure 6-4 PBS

6.2 Data sharing

Computational grids are well suited for handling CPU intensive jobs, but when more data is involved, there is a greater need for a data grid. Data grids may include the sharing of data from many different data sources as well as many different locations. The more this can be virtualized, the better. For example, if somebody was looking for a certain kind of data, it would be nice if they could write a query that would span all the available data sources in all locations without that person knowing where the results are coming from and what kind of data source is used. Today, there are implementations of federated databases that realize this goal. For example, IBM offers some of this functionality with DB2. Also, a file system may be used in implementing a data grid. Distributed file systems can be used to virtualize storage across multiple machines. One example that will be discussed is IBM General Parallel File System (GPFS).

6.2.1 Federated databases

An organization may have data stored in many different data sources including Oracle databases, DB2 databases, flat files, and so on. If the organization wishes to share this data across the organization, it may be desirable to virtualize

all of these data sources to appear as one single data source. This way, a single query could be written that would locate the desired data across all of the various data sources. This is the concept behind a federated database. In existing relational database technologies, there is a database management system, a catalog, and data. A query is made from a client to the DBMS and resulting data is returned to the client. The results are all coming from the same data source in the database system (this is illustrated in Figure 6-5). In a federated database system, there is a federated DBMS, catalog, local data source, and remote heterogeneous data sources. A query is made from a client to the federated DBMS. The federated DBMS then uses wrappers to access a variety of data sources. The local data source can be used to store query results for better performance (this is illustrated in Figure 6-6 on page 123).

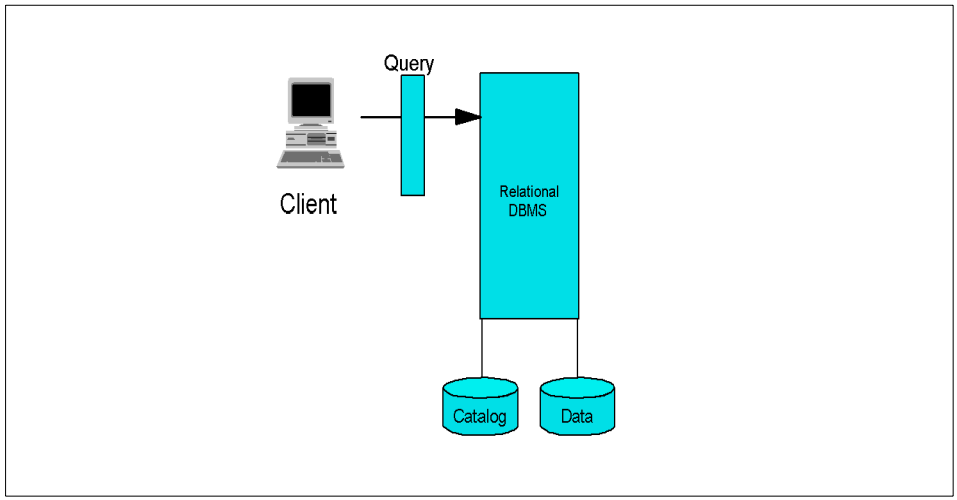

Figure 6-5 Relational database

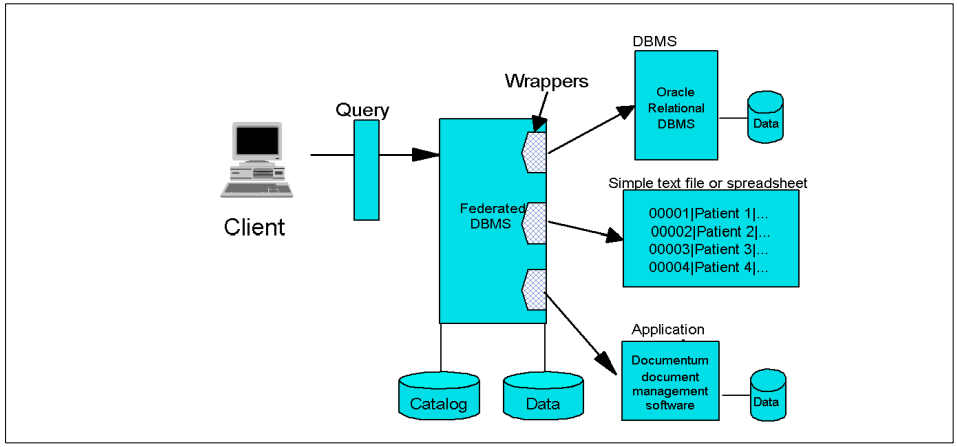

Figure 6-6 Federated database

There is a working group in the Global Grid Forum, Database Access, and Integration Services, which focuses on "promoting standards for the development of grid database services." Their home page can be found at http://www.gridforum.org/6_DATA/dais.htm. In the future, as OGSA is defined further, there will be data services defined. As the standards are defined, wrappers can be written to plug into any OGSA compliant data source.

IBM DB2

There are IBM DB2 products that have some of the federated database features. DB2 Relational Connect provides access to heterogeneous data sources like Informix, IDS, Oracle, Sybase, Microsoft SQL Server, and DB2. A DB2 instance is set up to be the federated database and its catalog contains information on the other available data sources. The other available data sources consist of a DBMS and data. Nicknames are created for tables and views in the other data sources. After the system is set up, clients can connect to the database and see all the data sources as one database. Relational Connect only allows for read access to the other data sources. DB2 DataJoiner is IBM database middleware that provides features similar to Relational Connect. The differences are that DataJoiner works with additional data sources and has insert, delete, and update capabilities. Included in this list of supported datasources is DB2, Informix, Oracle, Sybase, Microsoft SQL Server, IMS, and VSAM. In addition to these products, there is DB2 Life Sciences Data Connect, which supports even more data sources, including blast, Documentum, Excel, and Table-structured files.

6.2.2 Distributed file systems

Apart from sharing data from relational databases, an organization may want to share files. Again, it would be nice to virtualize the sharing of files as much as possible. It may be desirable to distribute files across many different storage devices. These many different storage devices could be virtualized to appear as one large disk. A user trying to access a file would not need to know what machine the file is stored on, only the path of the file in a virtual namespace. The following section will discuss one example of a distributed file system.

IBM GPFS

IBM General Parallel File System (GPFS) is a high performance shared-disk file system available for AIX and Linux. There are two types of configurations that can be applied to a GPFS cluster. First, there is the Network Shared Disk (NSD) Server Model. In this configuration, only one node, or two if redundancy is applied, is attached to a storage device. For this configuration, there is a potential for high network traffic, so a high speed network is recommended, for example, Gigabit Ethernet or Myrinet. This potential for high network traffic is due to the fact that every time a node requests to read or write to a file, the request is sent to the node that is directly attached to the storage device. After the attached node has completed the task, it will send the data back to the requesting node if it was a read request, or confirmation of completion, if it was a write request. The second configuration is Directly Attached Model. In this configuration, every GPFS node must have a direct connection to every storage device.

With GPFS, files can be stored across multiple nodes in a cluster, which allows for large file sizes. GPFS allows for multiple clients to access the same file at the same time and uses a token management system to ensure data consistency. Data and metadata can be replicated to achieve high availability. There are also many features to improve performance, including client side caching and data striping.

6.3 Security

Security is a very important topic in grid computing and is discussed in great detail in Chapter 3, “Security” on page 51. This section will briefly discuss the OpenSSL open source toolkit. OpenSSL includes a cryptography library and a library used to implement the Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols. The SSL/TLS protocols can be used to encrypt and decrypt communication between nodes in a grid system. The OpenSSL toolkit also provides a command line utility which can be used for many things, including creating certificates and keys and encryption and decryption. For more

information on OpenSSL and to obtain the latest release, please visit http://www.openssl.org.

6.4 Directory service

With the idea that a grid can be made up of many heterogeneous resources, it is vital to have a method of capturing what the grid looks like. One would want to be able to perform queries on both static and dynamic attributes of a grid. The static attributes may include the machines enrolled in a grid, the operation system version, CPU speed, physical memory, and so on. The dynamic attributes may include what machines are available, disk space available, current load, and so on. The larger the grid, the more important it is to be able to store and retrieve this kind of information. The Globus Toolkit component called Monitoring and Discovery Services (MDS) is implemented using OpenLDAP. Chapter 7, “Components” on page 131 describes further the MDS.

OpenLDAP

OpenLDAP is an open source software package that implements the Lightweight Directory Access Protocol (LDAP). The OpenLDAP package includes an LDAP server, LDAP replication server, libraries for the LDAP protocol, and various utilities, including some sample clients. LDAP is a protocol used for directory access and works over TCP/IP. LDAP directories store information with a unique Distinguished Name and associated attributes. Each individual attribute is made up of a type and a value. For example, an attribute for a resource might contain the type “osname” and value “Linux”. These entries are stored in a hierarchical structure. In LDAP, the hierarchical structure is usually used to show some boundaries. For example, the top of the structure may be different organizations or regions, and branching out from each organization are the attributes associated with it. This structure can be seen in Figure 6-7 on page 126. LDAP queries allow for filters to be used, so the query may only search a portion of the structure. So if a user wanted to know only the servers located in Austin, Texas that are administrated by Sally, the only part of the directory structure that needs to be searched is located under Austin; the rest of the tree can be ignored. Clients must authenticate their identity to LDAP directory servers, so access to the directory server can be limited.

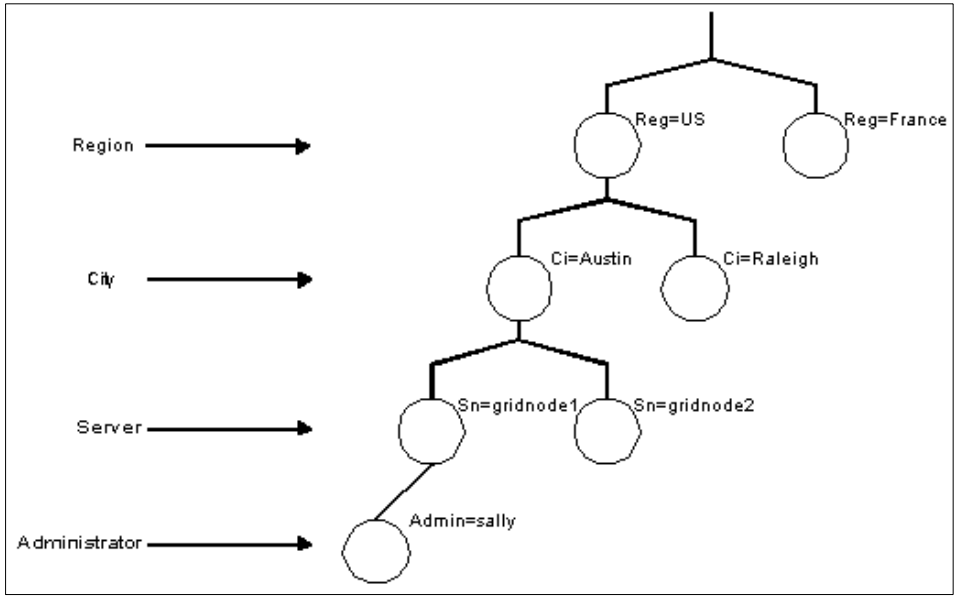

Figure 6-7 LDAP directory tree

LDAP follows a client/server model. The LDAP server contains the data that forms the LDAP directory tree. An LDAP client queries the server for data. The LDAP server returns the data to the client or the location if the data is stored on a different LDAP server. There are several different ways that LDAP directory servers can be set up. The simplest configuration would be a single LDAP server that contains all of the information for a local domain and is not aware of any other LDAP servers. In other words, the LDAP server will only return data that it is responsible for and will not refer a client to another LDAP server (this configuration is illustrated in Figure 6-8 on page 127). A second way would be to set up a LDAP server that returned requested data for its domain. But if the request was for another domain, it would return a referral back to the client for the LDAP server of the required domain. Another configuration may be to replicate the LDAP directory to another LDAP server for increased reliability or scalability.

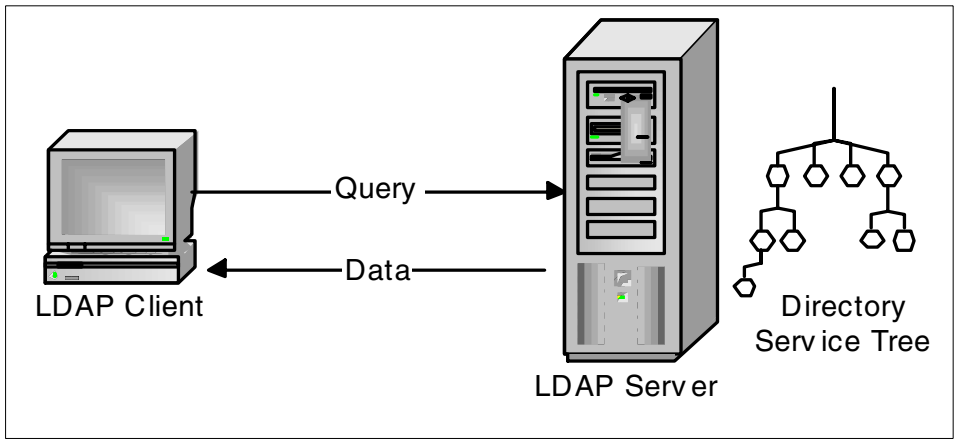

Figure 6-8 Simple LDAP configuration

The LDAP server, which is available with OpenLDAP (slapd), has many features. The security features for slapd include a wide range of access control information, authentication services, and privacy through the use of TLS or SSL. There are a wide variety of databases that can be configured to wok with slapd and it can be configured to use multiple database instances. slapd can also be replicated for high availability.

6.5 License management

License management is another important concept when discussing grid computing. Some software licenses may be set up to where they can be shared among multiple CPUs and geographies. In a computing grid, where several applications or users may need the same licensed product, one may not want to purchase a license for every user and application. Instead, it may be more desirable to buy less licenses and have a system that manages the use of these licenses. This system may implement a fair share scheme where all users and applications have equal access to these licenses, or a priority system may be established where higher priority applications or users are able to use the licenses when needed. More effective license management can be very cost effective.

Platform Global License Broker

One example of a license management product is Platform Global License Broker. Platform Global License Broker is capable of monitoring current license usage and distributing licenses based on an organization's policies. For example, it can be configured to use a fair share scheme giving all designated

users equal access to licenses. Alternatively, it may be configured to distribute licenses based on priority. For example, higher priority users will have higher access to a license than lower priority users. Platform Global License Broker can be combined with another product, Platform License Maximizer, to preempt running jobs to free up licenses. If a high priority job is waiting on a license, Platform License Maximizer can be configured to suspend a lower priority job using that license. Once the license is free again, the suspended job will be resumed.

6.6 Development tools

As computing grids develop and evolve, it will be important to have tools available to aid developers in creating and modifying applications to run on grids. Some of the grid software discussed in Chapter 5, “Grid software” on page 105 contained APIs and development kits that can be used to develop and integrate applications to run on their platforms. There are development tools available to help with development for the Globus Toolkit. Included in these tools are Commodity Grid (CoG) Kits. CoG Kits provide developers access to some of the Globus Toolkit’s grid services using different commodity technologies. Some of these technologies include Java, Python, Corba, Perl, and Matlab. Most of the CoG Kits are currently under development, but the source code is available. Please visit http://www-unix.globus.org/cog/ for more information. There is also a tool available for implementing MPI (Message Passing Interface) applications on a Globus Toolkit environment. MPICH-G2 is a grid implementation of the MPI v1.1 standard. MPICH-G2 allows for the Globus services, including security, to be used for MPI applications. MPICH-G2 documentation and source code is available from http://www3.niu.edu/mpi.

Part 4

Globus Toolkit

7

Components

This chapter presents some of the main components of the Globus Toolkit 2.2, which provides:

- Single sign-on, authorization, and authentication
- Job submission
- Resource monitoring, searching, and allocation
- Data movement

Also, the Globus Toolkit provides a set of tools for application programming (APIs) and system development kits (SDKs).

Extensive information about Globus Toolkit and Globus Project can be found at:

http://www.globus.org

7.1 Three pyramids

Globus Toolkit has three pyramids of support built on top of a security infrastructure, as illustrated in Figure 7-1. They are:

- Resource management
- Data management
- Information services

All of these pyramids are built on top of the underlying Grid Security Infrastructure (GSI). This provides security functions, including single/mutual authentication, confidential communication, authorization, and delegation.

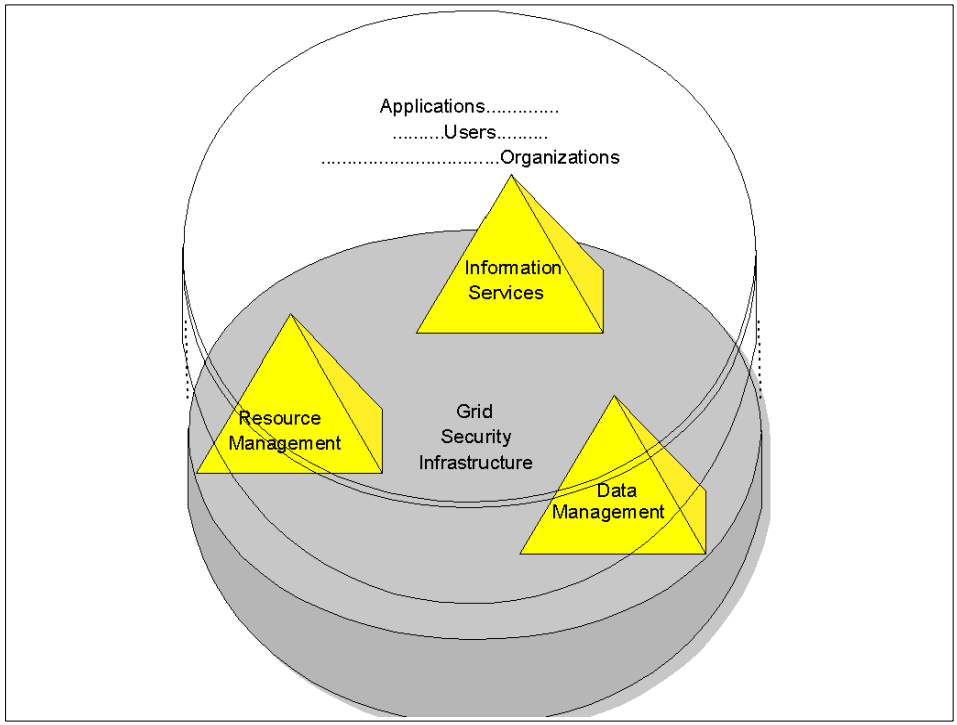

Figure 7-1 Three pyramids

Resource management

The resource management pyramid provides support for:

- Resource allocation
- Submitting jobs: Remotely running executable files and receiving results
- Managing job status and progress

Globus Toolkit does not have its own job scheduler to find available resources and automatically send jobs to suitable machines. Instead, it provides the tools and interfaces needed to implement schedulers and is often used with third-party schedulers.

Information services

The information services pyramid provides support for collecting information in the grid and for querying this information, based on the Lightweight Directory Access Protocol (LDAP).

Data management

The data management pyramid provides support to transfer files among machines in the grid and for the management of these transfers.

7.1.1 Open standards

Globus Toolkit is an open standard software developed and blueprinted by the Globus Project. Information about the collaborators of the project can be found at the following Web site:

http://www.globus.org/about/collaborators.html

In addition to the Globus Project, the Global Grid Forum has contributed a collection of references and standards. The GGF, as the Global Grid Forum is known, is a community initiative that includes participants of over 200 organizations in more than 30 countries. The GGF Web site is http://www.ggf.org/.

7.2 Components of Globus Toolkit

For each pyramid previously presented, Globus provides a component to implement resource management, data management, and information services, as illustrated in Figure 7-2 on page 134.

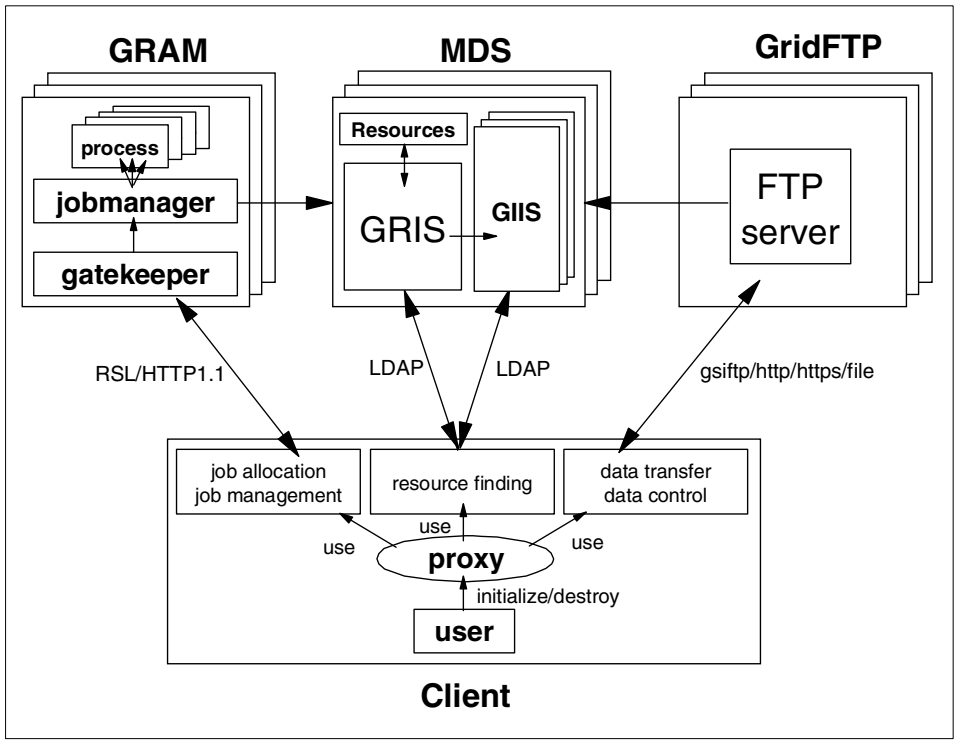

Figure 7-2 The system overview of Globus Toolkit

- GRAM/GASS

  The primary components of the resource management pyramid are the Grid Resource Allocation Manager (GRAM) and the Global Access to Secondary Storage (GASS).

- MDS (GRIS/GIIS)

  Based on the Lightweight Directory Access Protocol (LDAP), the Grid Resource Information Service (GRIS) and Grid Index Information Service (GIIS) components can be configured in a hierarchy to collect the information and distribute it. These two services are called the Monitoring and Discovery Service (MDS). The information collected can be static information about the machines as well as dynamic information showing current CPU or disk activity. A rich set of information providers is included with the Toolkit and the Globus users can add their own. The information provides an interface with the GRIS, which reports this information to a hierarchy of GIIS servers in the grid. The LDAP query language is used to retrieve the desired information.

- GridFTP

  GridFTP is a key component for the secure and high-performance data transfer. The Globus Replica Catalog and Management is used to register and manage both complete and partial copies of data sets.

  These three pyramids are modularized and can function in isolation; however, together, they complement each other.

- GSI

  All of the above components are built on top of the underlying Grid Security Infrastructure (GSI). This provides security functions including single/mutual authentication, confidential communication, authorization, and delegation.

7.2.1 Grid Security Infrastructure (GSI)

GSI provides elements for secure authentication and communication in a grid. The infrastructure is based on the SSL protocol (Secure Socket Layer), public key encryption, and x.509 certificates.

For a single sign-on, Globus add some extensions on GSI. It is based on the Generic Security Service API, which is a standard API promoted by the Internet Engineering Task Force (IETF).

These are the main functions implemented by GSI:

- Single/mutual authentication
- Confidential communication
- Authorization
- Delegation

The mechanism of GSI can be seen in detail in Chapter 3, “Security” on page 51, and at the following official GSI Web site:

http://www.globus.org/security/

7.2.2 Grid Resource Allocation Manager (GRAM)

GRAM is the module that provides the remote execution and status management of the execution. When a job is submitted by a client, the request is sent to the remote host and handled by the **gatekeeper** daemon located in the remote host. Then the **gatekeeper** creates a job manager to start and monitor the job. When the job is finished, the job manager sends the status information back to the client and terminates.

The official Web site of for the GRAM is:

http://www.globus.org/gram

Figure 7-3 depicts the conceptual view about GRAM. It contains the following elements:

- The **globusrun** command
- Resource Specification Language (RSL)
- The gatekeeper daemon
- The job manager
- The forked process
- Global Access to Secondary Storage (GASS)
- Dynamically-Updated Request Online Coallocator (DUROC)

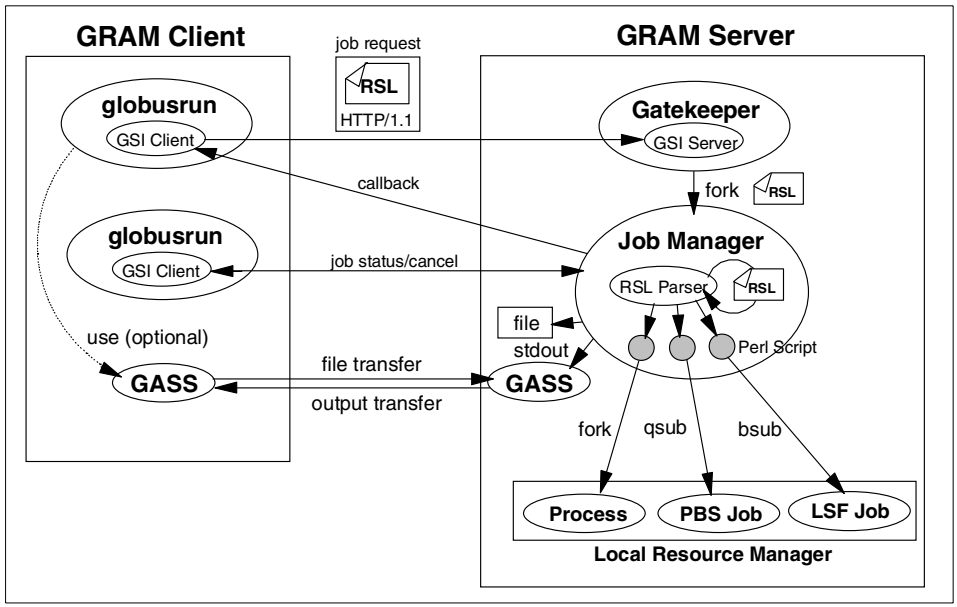

Figure 7-3 Overview of GRAM

The globusrun command

The **`globusrun`** command submits and manages remote jobs and is used by almost all GRAM client tools. This command provides the following functions:

- Request of job submission to remote machines

  Job submission uses security functions (such as GSS-API) to check mutual authentication between clients and servers, and also to verify the rights to submit the job.

- Transfer the executable files and the resulting job-submission output files

  The **`globusrun`** command can get the standard output of job results from remote machines. It uses GASS to provide the secure file transfer between grid machines.

Resource Specification Language (RSL)

RSL is the language used by the clients to submit a job. All job submission requests are described in RSL, including the executable file and condition on which it must be executed. You can specify, for example, the amount of memory needed to execute a job in a remote machine.

Gatekeeper

The gatekeeper daemon builds the secure communication between clients and servers. The gatekeeper daemon is similar to inetd daemon in terms of functionality. However, gatekeeper provides a secure communication. It communicates with the GRAM client (globusrun) and authenticates the right to submit jobs. After authentication, gatekeeper forks and creates a job manager delegating the authority to communicate with clients.

Job manager

Job manager is created by the **`gatekeeper`** daemon as part of the job requesting process. It provides the interfaces that control the allocation of each local resource manager, such as a job scheduler like PBS, LSF, or LoadLeveler.

The job manager functions are:

- Parse the resource language

  Breaks down the RSL scripts.

- Allocate job requests to the local resource managers

  The local resource manager is usually a job scheduler like PBS, LSF, or LoadLeveler. The resource manager interface is written in the Perl language, which easily allows you to create a new job manager to the local resource manager, if necessary.

- Send callbacks to clients, if necessary

- Receive the status and cancel requests from clients
- Send output results to clients using GASS, if requested

Global Access to Secondary Storage (GASS)

GRAM uses GASS for providing the mechanism to transfer the output file from servers to clients. Some APIs are provided under the GSI protocol to furnish secure transfers. This mechanism is used by the **`globusrun`** command, **`gatekeeper`**, and job manager.

Dynamically-Updated Request Online Coallocator (DUROC)

By using the DUROC mechanism, users are able to submit jobs to different job managers at different hosts or to different job managers at the same host (see Figure 7-4 below).

The RSL script that contains the DUROC syntax is parsed at the GRAM client and allocated to different job managers. The grammar and attributes of RSL and DUROC are explained in “Resource Specification Language (RSL)” on page 137.

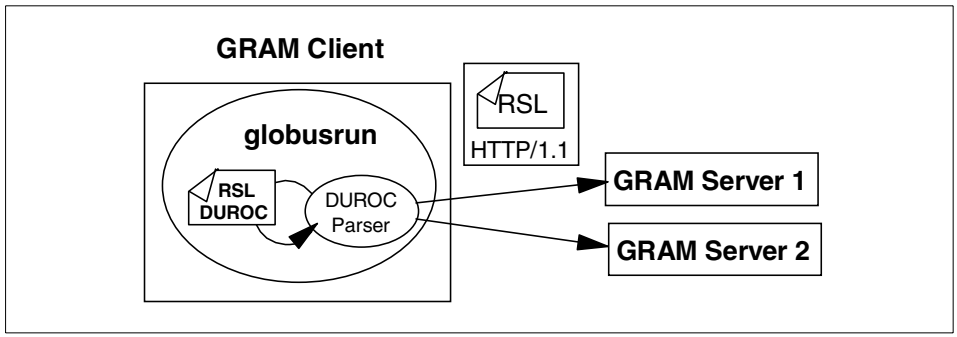

Figure 7-4 Overview of DUROC

7.2.3 Monitoring and Discovery Service (MDS)

MDS provides access to static and dynamic information of resources. Basically, it contains the following components:

- Grid Resource Information Service (GRIS)
- Grid Index Information Service (GIIS)
- Information Provider
- MDS client

The official site of MDS is:

http://www.globus.org/mds/

Figure 7-5 represents the conceptual view interconnection of the MDS components. As illustrated, the resource information is obtained by the information provider and it is passed to GRIS. GRIS registers its local information with the GIIS, which also registers with another GIIS, and so on. MDS clients can get the resource information directly from GRIS (for local resources) and/or a GIIS (for grid-wide resources).

The MDS uses LDAP, which provides the decentralized maintenance of resource information.

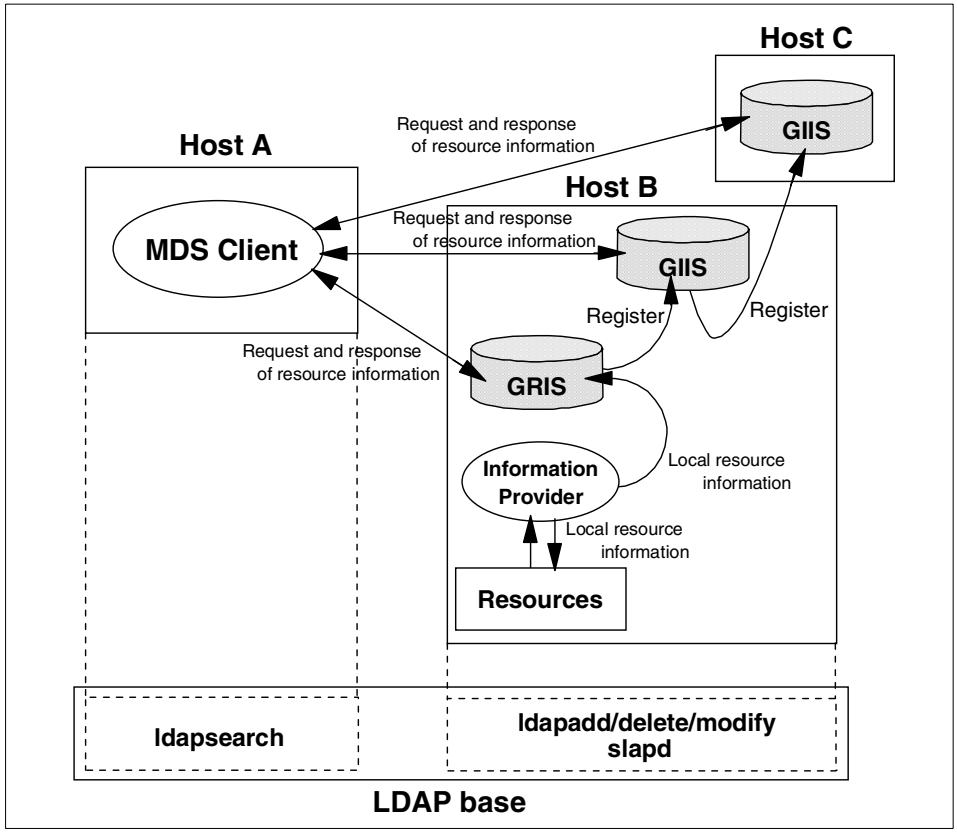

Figure 7-5 Overview of MDS

Resource information

Resource information contains the objects managed by MDS, which represent components resources (static and dynamic) as follows:

- Infrastructure components
  - For example, name of the job manager or name of the running job
- Computer resources
  - For example, network interface, IP address, or memory size

Grid Resource Information Service (GRIS)

GRIS is the repository of local resource information derived from information providers. GRIS is able to register its information with a GIIS, but GRIS itself does not receive registration requests. The local information maintained by GRIS is updated when requested, and cached for a period of time known as the time-to-live (TTL). If no request for the information is received by GRIS, the information will time out and be deleted. If a later request for the information is received, GRIS will call the relevant information provider(s) to retrieve the latest information.

Grid Index Information Service (GIIS)

GIIS is the repository that contains indexes of resource information registered by the GRIS and other GIISs. It can be seen as a grid wide information server. GIIS has a hierarchical mechanism, like DNS, and each GIIS has its own name. This means client users can specify the name of a GIIS node to search for information.

Information providers

The information providers translate the properties and status of local resources to the format defined in the schema and configuration files. In order to add your own resource to be used by MDS, you must create specific information providers to translate the properties and status to GRIS.

MDS client

The MDS client is based on the LDAP client command, **`ldapsearch`**. A search for a resource information that you want in your grid environment is initially performed by the MDS client.

Hierarchical MDS

The MDS hierarchy mechanism is similar to the one used in DNS. GRIS and GIIS, at lower layers of the hierarchy, register with the GIIS at upper layers. Clients can query the GIIS for any information about resources that build a grid environment.

7.2.4 GridFTP

GridFTP provides a secure and reliable data transfer among grid nodes. The word GridFTP can referred to a protocol, a server, or a set of tools.

GridFTP protocol

GridFTP is a protocol intended to be used in all data transfers on the grid. It is based on FTP, but extends the standard protocol with facilities such as multistreamed transfer, auto-tuning, and Globus based security. This protocol is still in draft level, so for more information, please refer to the following Web site (you must have Adobe Acrobat Reader to view the document):

http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf

As the GridFTP protocol is still not completely defined, Globus Toolkit does not support the entire set of the protocol features currently presented. A set of GridFTP tools is distributed by Globus as additional packages. Globus Project has selected some features and extensions defined already in IETF RFCs and added a few additional features to meet requirements from current data grid projects.

GridFTP server and client

Globus Toolkit provides the GridFTP server and GridFTP client, which are implemented by the in.ftpd daemon and by the **globus-url-copy** command, respectively. They support most of the features defined on the GridFTP protocol.

The GridFTP server and client support two types of file transfer: standard and third-party. The standard file transfer is where a client sends the local file to the remote machine, which runs the FTP server. An overview is shown in Figure 7-6.

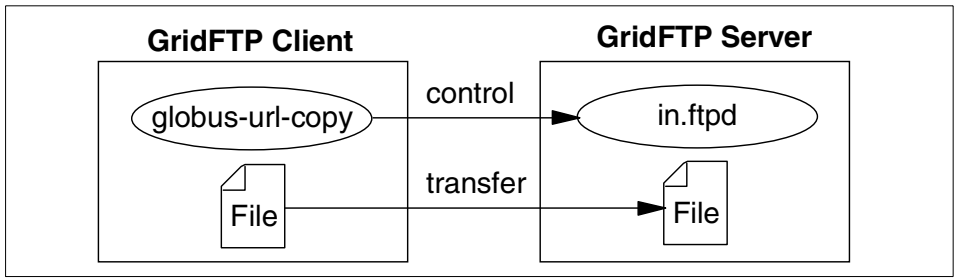

Figure 7-6 Standard file transfer

Third-party file transfer is where there is a large file in remote storage and the client wants to copy it to another remote server, as illustrated in Figure 7-7 on page 142.

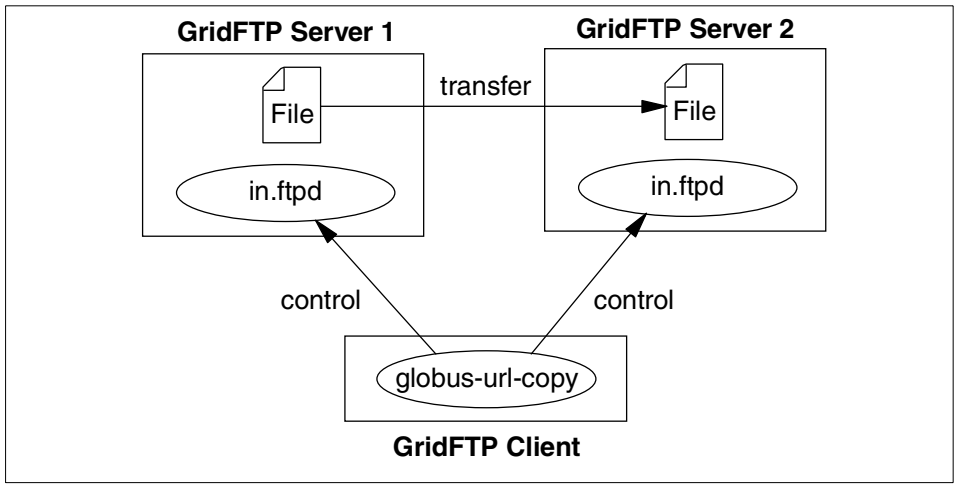

Figure 7-7 Third-party file transfer

GridFTP tools

Globus Toolkit provides a set of tools to support GridFTP type of data transfers. The gsi-ncftp package is one of the tools used to communicate with the GridFTP Server. This package is available at the following site:

http://www-unix.globus.org/ftppub/gt2/2.0/contrib/globus_gsincftp-0.2.tar.gz

The GASS API package is also part of the GridFTP tools. It is used by the GRAM to transfer the output file from servers to clients.

7.2.5 API and software developer's kit

Two other components are available to help develop Globus related grid applications:

- APIs
- Developer’s toolkit

API

Globus Toolkit APIs are basically implemented in the C language. Information about the APIs are at the following site:

http://www-unix.globus.org/api/c-globus-2.2/

Developer's kit

Globus provides a rapid development kit known as CoG (Commodity Grid), which supports the following technologies:

- Java
- Python
- Web services
- CORBA
- Java Server Pages
- Perl
- Matlab

More information about CoG can be found at:

http://www-unix.globus.org/cog/

8

Installation and setup

This chapter presents the necessary steps to install and configure Globus Toolkit 2.2.

The following topics are discussed:

- Requirements
- Installation
- Setup
- Advanced configurations
- Client interface

8.1 How to obtain Globus Toolkit

Globus Toolkit supports Red Hat Linux on xSeries, AIX on pSeries, and SuSE Linux Enterprise Server 8 (SLES 8) on zSeries, which contains the pre-compiled binary distribution of the Globus 2.0 code for Linux on zSeries.

As mentioned in 5.2, “IBM Grid Toolbox (Globus)” on page 106, an IBM Grid Toolbox package is available from IBM, which includes the Globus Toolkit.

For the purpose of this redbook, we selected the Globus Toolkit Version 2.2 running on Red Hat 7.3. This version of Globus Toolkit may be obtained at the official Globus Project site:

http://www.globus.org/gt2.2/download.html

Important: Globus Toolkit is distributed under the *Globus Toolkit Public License* (GTPL), a liberal open source license. You are allowed to use every tool and the source code in the Globus Toolkit as you like with no restriction. But it is *AS IS* with *NO WARRANTY*. You can find out more about the GTPL at:

http://www.globus.org/toolkit/download/license.html

For bug tracking, the Globus Project provides the following Web site:

http://bugzilla.globus.org/bugzilla/

For platform specific system requirements for Globus Toolkit 2.2, please refer to the following Web site:

http://www.globus.org/gt2.2/platform.html

8.2 Bundles and Grid Packaging Technology (GPT)

GPT is a package used for installation and distribution, which includes libraries, files, and modules to support package creation and installation. It supports the installation of Globus Toolkit bundles. The package contains the executable files, script files, and configuration files. There are two types of bundles: source bundles and binary bundles.

8.2.1 Source bundles

Table 8-1 on page 147 lists the source bundles available for Globus Toolkit Version 2.2. The source code must be compiled before installation.

Table 8-1 Source bundles of Globus Toolkit

	Client bundle	Server bundle	SDK bundle
Resource management	globus-resource-management-client-2.2.2-src_bundle.tar.gz	globus-resource-management-server-2.2.2-src_bundle.tar.gz	globus-resource-management-sdk-2.2.2-src_bundle.tar.gz
Information services	globus-information-services-client-2.2.2-src_bundle.tar.gz	globus-information-services-server-2.2.2-src_bundle.tar.gz	globus-information-services-sdk-2.2.2-src_bundle.tar.gz
Data management	globus-data-management-client-2.2.2-src_bundle.tar.gz	globus-data-management-server-2.2.2-src_bundle.tar.gz	globus-data-management-sdk-2.2.2-src_bundle.tar.gz

8.2.2 Binary bundles

The binary bundles contain the binary executable files that have been pre-compiled for specific platforms. Table 8-2 lists the binary bundles for a Linux Intel-based platform.

Table 8-2 Binary bundles of Globus Toolkit

Binary bundle	Contents
globus-all-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Client and server packages: Resource management, information services and data management
globus-all-server-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Server packages
globus-all-client-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Client packages
globus-all-sdk-2.2.2-i686-pc-linux-gnu-bin.tar.gz	SDK packages
globus-data-management-server-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Server packages for the data management
globus-data-management-client-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Client packages for the data management

Binary bundle	Contents
globus-data-management-sdk-2.2.2-i686-pc-linux-gnu-bin.tar.gz	SDK bundles for the data management
globus-information-services-server-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Server packages for the information service
globus-information-services-client-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Client packages for the information service
globus-information-services-sdk-2.2.2-i686-pc-linux-gnu-bin.tar.gz	SDK packages for the information service
globus-resource-management-server-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Server packages for the resource management
globus-resource-management-client-2.2.2-i686-pc-linux-gnu-bin.tar.gz	Client packages for the resource management
globus-resource-management-sdk-2.2.2-i686-pc-linux-gnu-bin.tar.gz	SDK packages for the resource management

Other platform-specific binary bundles are available at the following Globus FTP site:

ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/bundles/bin/

8.2.3 Additional bundles

Globus Toolkit provides additional software as GPT bundles.

- GSI:

  globus-gsi-2.2.2-src_bundle.tar.gz (source bundle), found at:

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/bundles/src/

- GRAM Job Manager Scheduler Support:

  globus_gram_job_manager_setup_condor-1.0.tar.gz (source bundle)

  globus_gram_job_manager_setup_pbs-1.0.tar.gz (source bundle)

globus_gram_job_manager_setup_lsf-1.0.tar.gz (source bundle)

globus_gram_job_manager_setup_remote-1.0.tar.gz (source bundle)

These are found at:

ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/extra/gram_job_manager/src/

- GRAM Reporter Scheduler Support:

  globus_gram_reporter-2.0.tar.gz (source bundle)

  globus_gram_reporter_setup_fork-1.0.tar.gz (source bundle)

  globus_gram_reporter_setup_condor-1.0.tar.gz (source bundle)

  globus_gram_reporter_setup_pbs-1.0.tar.gz (source bundle)

  globus_gram_reporter_setup_lsf-1.0.tar.gz (source bundle)

  These are found at:

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/extra/gram_reporter/src/

- File Replication Software:

  globus-replica-2.2.2-src_bundle.tar.gz (source bundle), found at:

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/bundles/src/

  globus-replica-2.2.2-i686-pc-linux-gnu-bin.tar.gz (Linux binary bundle), found at:

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/bundles/bin/

- GSI-enabled version of ncftp FTP client software:

  globus_gsincftp-0.6.tar.gz (source bundle), found at:

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/contrib/src/

- Globus Simple CA bundle:

  globus_simple_ca_bundle-0.7.tar.gz (source bundle), found at:

  ftp://ftp.globus.org/pub/gsi/simple_ca

- Globus Toolkit 2.2 Advisories

  There are the security-fix, bug-fix, and enhancement version of packages, which can be found at the following site:

  http://www.globus.org/gt2.2/advisories.html

8.3 Grid environment

Figure 8-1 introduces a conceptual grid environment presented after a Globus Toolkit installation. In this environment, there are three servers:

- demoCA

  It is the simple Certificate Authority.

- Host A and host B

  They are the grid nodes.

The user's names are different on host A and host B, but they share the same grid user ID, which is known as the Distinguished Name:

```
/O=Grid/O=Globus/OU=itso.grid.com/CN=ITSO grid user
```

We are going to explain how to install GPT and install and set up the servers demoCA and host A. The installation and setup of host B is explained in 8.5.3, "Adding a new grid server" on page 168.

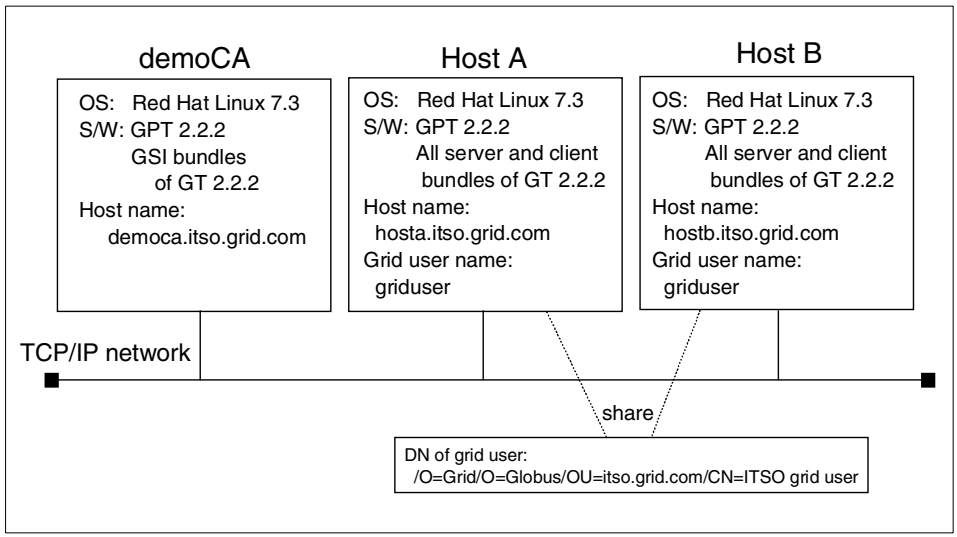

Figure 8-1 System overview after installation

8.4 Installation

You first need to install GPT into the servers and then install the server and client bundles.

Note: In order to build the grid environment presented in 8.3, “Grid environment” on page 150, install GPT on the demoCA and host A servers. After the installation of GPT, you must install the Certificate Authority to demoCA and the server and client bundles on host A.

8.4.1 Installation of GPT

The name of file you need is gpt-2.2.2-src.tar.gz. Before installing GPT, you are required to set an environment variable, $GPT_LOCATION, that points to the location where GPT itself is installed. You should set an environment variable, $GLOBUS_LOCATION, that points to the location where the Globus Toolkit build and install output is placed. The example is as follows:

```
export GPT_LOCATION=/usr/local/gpt
export GLOBUS_LOCATION=/usr/local/globus
```

Note: To save time upon subsequent logins, it is useful to set up the $GPT_LOCATION and $GLOBUS_LOCATION environmental variables in the profile file, such as ~/.bash_profile or /etc/profile.

Extract and unzip gpt-2.2.2-src.tar.gz and build the tools inside of it using the **build_gpt** command. The GPT commands are installed in the $GPT_LOCATION/sbin directory (see Example 8-1).

Example 8-1 Untar GPT source archive and build

```
[root@democa globus]# tar -zxvf gpt-2.2.2-src.tar.gz
gpt-2.2.2/
gpt-2.2.2/support/
gpt-2.2.2/support/Archive-Tar-0.22/
...(author omits lists)
[root@democa globus]# cd gpt-2.2.2
[root@democa gpt-2.2.2]# ./build_gpt
build_gpt ====> building support/Compress-Zlib-1.16
Up/Downgrade complete.
Up/Downgrade complete.
build_gpt ====> building support/Archive-Tar-0.22
build_gpt ====> building support/PodParser-1.18
build_gpt ====> building packaging_tools
[root@democa gpt-2.2.2]# ls $GPT_LOCATION/sbin
gpt-build                  gpt_save_flavor
gpt_build_config           gpt-setup
gpt-bundle                 gpt_sort_filelist
gpt_create_automake_rules  gpt-translate-interpreter
gpt-deps                   gpt-translate-interpreter-pl
gpt-edit                   gpt-undefines
```

```
gpt_extract_data           gpt-uninstall
gpt-flavor-configuration   gpt-update
gpt_generate_bin_pkg_data  gpt-verify
gpt_get_lib_names          gpt-virtual-pkg
gpt-install                pod2usage
gpt-perl-version           podchecker
gpt-pkg                    podselect
gpt-postinstall            ptar
gpt-query
```

8.4.2 Installation of bundles

In this next section, the general installation procedure of bundles (both source and binary) is explained. After that, the installation examples of host A and demoCA are shown.

Installation of source bundles

gpt-build is the command used to install source bundles. Many bundles distributed by Globus Project are built by using this command. The command to build a bundle is as follows:

```
gpt-build some_bundle.tar.gz <options> <flavors>
```

You can see some of the **gpt-build** flavors and options in Table 8-3 and Table 8-4 on page 153. The flavor option is the compiler option and depends on the C compiler which you want to use. The examples of flavors shown below are for the GNU C compiler.

Table 8-3 gpt-build flavors usually used

Flavor	Description
gcc32	Build bundles with GNU C 32-bit compiler as a single-thread without debug option.
gcc32dbg	Build bundles with GNU C 32-bit compiler as a single-thread with debug option.
gcc32pthr'	Build bundles with GNU C 32-bit compiler as a multi-thread without debug option.
gcc32dbgpthr	Build bundles with GNU C 32-bit compiler as a multi-thread with debug option.

Table 8-4 Options of globus-build command

Option	Description
-verbose	Tells gpt-build to print all of the build output. This is useful in debugging builds.
-log='directory path'	Tells gpt-build to keep a log of the install in 'directory path'.
-static	Tells gpt-build to link all of the programs to libraries statically, not to use any shared libraries.

Table 8-5 shows the standard combination of flavors and options for each bundle. In this case, there are no standard options.

Table 8-5 The standard combination of flavors and options

Bundle	Flavor
Data Management Client	gcc32dbg
Data Management Server	gcc32dbg
Data Management SDK	gcc32dbg
Information Services Client	gcc32dbgpthr
Information Services Server	gcc32dbgpthr
Information Services SDK	gcc32dbgpthr
Resource Management Client	gcc32dbg
Resource Management Server	gcc32dbg
Resource Management SDK	gcc32dbg
Replica	gcc32dbgpthr
GSI	gcc32dbg

Example 8-2 on page 154 is a sample procedure of installation of a source bundle (Data Management Client).

Example 8-2 A sample installation procedure (Data Management Client)

```
[root@hosta globus]# $GPT_LOCATION/sbin/gpt-build \
> globus-data-management-client-2.2.2-src_bundle.tar.gz \
> gcc32dbg
gpt-build ====> Changing to /home/globus/globus222/globus_core-2.6/
gpt-build ====> BUILDING FLAVOR gcc32dbgp
gpt-build ====> Changing to /home/globus/globus222
...(author omits lists)
```

Once you have installed all of the source bundles you wish to install, run the following command to complete your installation:

```
$GPT_LOCATION/sbin/gpt-postinstall
```

If you choose the installation of source bundle, you should install all server and client bundles in Table 8-1 on page 147 to host A. The SDK bundles are for development and are optional.

Installation of binary bundles

Use the GPT binary install command **`globus-install`** to install binary bundles. An example of how to install a bundle follows:

```
$GPT_LOCATION/sbin/globus-install \
> globus_data_management_bundle-server-linux-i686-gcc32.tar.gz
```

Once you have installed the binary bundles, you must run the following command to complete your installation:

```
$GPT_LOCATION/sbin/gpt-postinstall
```

Note: In order to build the grid environment presented in 8.3, “Grid environment” on page 150, you should install all binary server and client bundles shown in Table 8-2 to host A. The SDK bundles are optional.

Installation of a grid node

Example 8-3 shows the installation procedure of a binary bundle that contains both the server and client packages, globus-all-2.2.2-i686-pc-linux-gnu-bin.tar.gz, in host A.

Example 8-3 Installation procedure of grid node

```
[root@hosta globus]# $GPT_LOCATION/sbin/gpt-install \
> globus-all-2.2.2-i686-pc-linux-gnu-bin.tar.gz
globus_common-gcc32dbg-dev ver: 3.1 cmp id: 3.1.0 successfully installed.
globus_common-gcc32dbg-rtl ver: 3.1 cmp id: 3.1.0 successfully installed.
globus_common-gcc32dbgpthr-dev ver: 3.1 cmp id: 3.1.0 successfully installed.
...(author omits lists)
```

```
[root@hosta globus]# $GPT_LOCATION/sbin/gpt-postinstall
running /usr/local/globus/setup/globus/setup-globus-common...
creating globus-sh-tools-vars.sh
creating globus-script-initializer
...(author omits lists)
```

Installation of a simple Certificate Authority

There are two ways to use the Certificate Authority. One is to use the official Globus Certificate Authority; the other is to prepare your own CA in your grid environment.

If you use the official Globus Certificate Authority, its public key is in the GSI package. You do not have to build your Certificate Authority in your grid environment. Send your certificate requests, which are created by the **grid-cert-request** command, by e-mail to ca@globus.org. In about two days, you will receive a reply that contains your certificate signed by Globus CA.

Also, you can choose any CA package to build your own CA, such as OpenSSL. The server bundles of Globus Toolkit contains OpenSSL and some useful tools for building our own CA. The GSI bundle is enough to build a simple CA.

> **Note:** In order to build the grid environment presented in 8.3, “Grid environment” on page 150, a CA must be built on the demoCA server.

Example 8-4 shows the installation procedure of a simple CA on the demoCA server using the GSI bundle.

Example 8-4 Installation procedure of GSI bundle

```
[root@democa globus]# $GPT_LOCATION/sbin/gpt-build \
> globus-gsi-2.2.2-src_bundle.tar.gz gcc32dbg
gpt-build ====> Changing to /home/globus/globus/globus_core-2.6/
gpt-build ====> BUILDING FLAVOR gcc32dbg
gpt-build ====> Changing to /home/globus/globus
...(author omits lists)
[root@democa globus]# $GPT_LOCATION/sbin/gpt-postinstall
running /usr/local/globus/setup/globus/setup-globus-common...
creating globus-sh-tools-vars.sh
creating globus-script-initializer
...(author omits lists)
```

The GSI bundle, globus-gsi-2.2.2-src_bundle.tar.gz, is available at the following site:

ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/bundles/src/

8.4.3 Uninstallation

There is no command currently available to uninstall the Globus Toolkit bundles. GPT only provides the **gpt-uninstall** command to uninstall a package. A packages is a bundle with a dependency tree. Sometimes they might depend only on the packages in its bundle, while at other times they might depend on the packages in the other bundles. So, you need to uninstall all the related packages in order to keep the whole dependency tree. Fortunately, the **gtp-uninstall** command is able to identify the dependency. If one package is dependent upon other packages, this command does not uninstall it. But it does not support the automatic uninstallation of all the related packages in the dependency tree, so you need to check for the related packages to uninstall. Example 8-5 shows the uninstallation of the Data Management client bundle.

Example 8-5 Uninstallation of bundle

```
[root@hosta globus ] # tar -ztf \
>globus-data-management-client-2.2.2-src_bundle.tar.gz | \
>awk '{print "gpt-uninstall " $1}'FS=-|grep -v packaging_list \
>> uninstall.sh
[root@hosta globus ] # cat uninstall.sh
gpt-uninstall globus_gass_copy
gpt-uninstall globus_ftp_client
gpt-uninstall globus_gass_transfer
gpt-uninstall globus_ftp_control
gpt-uninstall globus_io
gpt-uninstall globus_gss_assist
gpt-uninstall globus_proxy_utils
gpt-uninstall globus_gssapi_gsi
gpt-uninstall globus_gsi_proxy_core
gpt-uninstall globus_gsi_credential
gpt-uninstall globus_gsi_callback
gpt-uninstall globus_gsi_sysconfig
gpt-uninstall globus_gsi_cert_utils
gpt-uninstall globus_gsi_openssl_error
gpt-uninstall globus_openssl_module
gpt-uninstall globus_gsi_proxy_ssl
gpt-uninstall globus_user_env
gpt-uninstall globus_trusted_ca_42864e48_setup
gpt-uninstall globus_openssl
gpt-uninstall globus_common_setup
gpt-uninstall globus_common
gpt-uninstall globus_core
[root@hosta globus ] #sh uninstall.sh
(execute this several times until no independent package exists.)
...(author omits lists)
```

8.5 Setting up the grid environment

After the installation of Globus Toolkit, each element of your grid environment must be set up. The first element is security.

You must request certificates for your servers processes from your CA. Next, set up your user certificate and validate the connection between the user applications and servers.

The following is a list of the needed files:

```
/etc/grid-security/certificates/<hash number>.0
/etc/grid-security/certificates/<hash number>.signing_policy
/etc/grid-security/hostcert.pem
/etc/grid-security/hostkey.pem
/etc/grid-security/ldap/ldapcert.pem
/etc/grid-security/ldap/ldapkey.pem
/etc/grid-security/grid-mapfile
<User's Home Directory>/.globus/usercert.pem
<User's Home Directory>/.globus/userkey.pem
```

The servers are then configured to act as daemon processes.

Attention: It is useful to set up the environmental variables PATH or LD_LIBRARY_PATH for Globus Toolkit. These are set at the command prompt or in a setup file like ~/.bash_profile or /etc/profile. For example:

```
PATH=$PATH:$GPT_LOCATION/sbin:$GLOBUS_LOCATION/bin:$GLOBUS_LOCATION/sbin
LD_LIBRARY_PATH=$GLOBUS_LOCATION/lib
```

It often happens that the command program cannot load libraries. It helps to check that the variable, LD_LIBRARY_PATH, is set correctly.

Globus Toolkit provides a shell script to set up these environmental variables. They can be sourced as follows:

```
source $GLOBUS_LOCATION/etc/globus-user-env.sh (sh)
source $GLOBUS_LOCATION/etc/globus-user-env.csh (csh)
```

If this shell script is added to the directory, /etc/profile.d/, then all needed environmental variables are set for all users.

8.5.1 Certificate Authority setup

Before setting up a CA, you should synchronize the system time of all the machines of your grid environment. GSI certificates use GMT and is very sensitive to the time. If the system time of your grid environment is not set

correctly, errors might occur when you use GSI certificates. For this reason, it is recommended that you set up a time server, NTP, in your grid environment and set the time correctly on all of your systems.

If your are building your own CA, the setup procedure using the GSI bundle is described below.

1. Create new CA, as shown in Example 8-6.

Example 8-6 Creation of new CA

```
[root@democa globus]# export SSLEAY_CONFIG="-config /usr/share/ssl/openssl.cnf"
[root@democa globus]# CA.sh -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Using configuration from /usr/share/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.....................................++++++
.....................................++++++
writing new private key to './demoCA/private/./cakey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Texas
Locality Name (eg, city) [Newbury]:Austin
Organization Name (eg, company) [My Company Ltd]:IBM
Organizational Unit Name (eg, section) []:ITSO
Common Name (eg, your name or your server's hostname) []:democa.itso.grid.com
Email Address []:ca@democa.itso.grid.com
```

Attention: The environmental valuable, $SSLEAY_CONFIG, is for the configuration of OpenSSL. The openssl.cnf file contained in the OpenSSL package of Red Hat 7.3 is used. You must install the OpenSSL package if you have not installed it on your system.

2. Create the files to be distributed.

   There are two files to be distributed to all grid machines. One is the public key file of the CA, and the other is the policy file for the CA. Example 8-7 on

page 159 shows how you can make the public key file for the distribution by using the **c_rehash** command.

Example 8-7 Creating the new CA public key

```
[root@democa CA]# $GLOBUS_LOCATION/bin/c_rehash demoCA
Doing demoCA
cacert.pem => cf9ff597.0
```

The name of policy file is in the form <hash number of key>.signing_policy. Example 8-8 shows the contents of a policy file.

Example 8-8 Example of policy file

```
access_id_CA      X509
'/C=US/ST=Texas/L=Austin/O=IBM/OU=ITSO/CN=democa.itso.grid.com/Email=ca@democa.
itso.grid.com'
pos_rights        globus        CA:sign
cond_subjects     globus     '"/O=Grid/O=Globus/*"'
```

The value of the access_id_CA attribute is the issuer name that was used to sign the certificates. This name should be the same as you used when creating the new CA (see Example 8-6 on page 158).

The CA public key and policy files, are shown below, respectively:

```
<CA's home directory>/demoCA/<hashed number>.0
<CA's home directory>/demoCA/<hashed number>.signing_policy
```

After the installation, the public key and policy file of CA should be distributed to all grid machines that use this CA.

Server certificates

All server bundles contain GSI. The first step is to initialize your GSI environment. After the installation of GSI, you are required to run the **setup-gsi** script (see Example 8-9). This script initializes the files that are necessary for security.

Example 8-9 Initialize GSI environment

```
[root@hosta globus]# $GLOBUS_LOCATION/setup/globus/setup-gsi
setup-gsi: Configuring GSI security
Making /etc/grid-security...
mkdir /etc/grid-security
Making trusted certs directory: /etc/grid-security/certificates/
mkdir /etc/grid-security/certificates/
Installing /etc/grid-security/certificates//grid-security.conf.42864e48...
Running grid-security-config...

   G S I   :   C O N F I G U R A T I O N   P R O C E D U R E
```

```
Before you use the Grid Security Infrastructure, you should first
define the DN (distinguished name) that should be used for your
organization's X509 certificates.  If you do not define a DN,
a default DN will be assigned to you.

This script will ask some questions about site specific
information. This information is used to configure
the Grid Security Infrastructure for your site.

For some questions, a default response is given in [].
Pressing RETURN in response to such a question will enable the default.
This script will overwrite the file --

     /etc/grid-security/certificates//grid-security.conf.42864e48

Do you wish to continue (y/n) [y] :
======================================================================

(1) Base DN for user certificates
         [ ou=itso.grid.com, o=Globus, o=Grid ]
(2) Base DN for host certificates
         [ o=Globus, o=Grid ]

======================================================================
(q) save, configure the GSI and Quit
(c) Cancel (exit without saving or configuring)
(h) Help
======================================================================
q
Installing Globus CA certificate into trusted CA certificate directory...
Installing Globus CA signing policy into trusted CA certificate directory...
setup-gsi: Complete
[root@hosta globus]# $GPT_LOCATION/sbin/gpt-verify
Verifying run-time dependencies...

Verifying setup dependencies...

Verifying setup packages...

The collection of packages in /usr/local/globus appear to be coherent.
```

The **gpt_verify** command verifies the installation.

Attention: For the setup of the CA, you don't need to run the setup-gsi script. This script creates a directory that contains the configuration files for security. CA does not need this directory, because these configuration files are for the servers and users who use the CA.

After the initialization, a directory that contains setup files for security, /etc/grid-security, is created. The public key and its policy file of CA should be copied to the /etc/grid-security/certificates directory.

All server bundles (GRAM, MDS, and GridFTP) use the GSI. Certificates must be created for all of them. GRAM and GridFTP use the same certificate and MDS uses a separated one.

GRAM and GridFTP certificates

The **`gatekeeper`** of GRAM and **`gsi-wuftpd`** of GridFTP use the same certificate (hostcert.pem) and private key (hostkey.pem). First, you need to make the private key and certificate request, as shown in Example 8-10.

Example 8-10 Creation of GRAM and GridFTP private key and certificate request

```
[root@hosta globus]# grid-cert-request -host hosta.itso.grid.com
Using configuration from /etc/grid-security/globus-host-ssl.conf
Generating a 1024 bit RSA private key
...(author omits lists)
```

Next, you need to send your certificate request to the CA and request it to sign your certificate. CA will send your signed certificate back to you. Place it in the /etc/grid-security/hostcert.pem file.

Note: In order to build the grid environment presented at 8.3, "Grid environment" on page 150, copy the certificate request file to the directory in which the demoCA directory is created. Rename it to newreq.pem and sign it, as presented in Example 8-11.

Example 8-11 Signing of GRAM and GridFTP certificate

```
(copy the hostcert_request.pem file from HostA to demoCA.)
[root@democa CA]# ls
demoCA  hostcert_request.pem
[root@democa CA]# cp hostcert_request.pem newreq.pem
[root@democa CA]# ls
demoCA  hostcert_request.pem  newreq.pem
[root@democa CA]# export SSLEAY_CONFIG="-config /usr/share/ssl/openssl.cnf"
[root@democa CA]# CA.sh -sign
Using configuration from /usr/share/ssl/openssl.cnf
```

```
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
organizationName      :PRINTABLE:'Grid'
organizationName      :PRINTABLE:'Globus'
commonName            :PRINTABLE:'host/democa.itso.grid.com'
Certificate is to be certified until Oct 17 15:47:35 2003 GM
T (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
...(author omits lists)
[root@democa CA]# ls
demoCA  hostcert_request.pem  newcert.pem  newreq.pem
[root@democa CA]# cp newcert.pem hostcert.pem
[root@democa CA]# ls
demoCA        hostcert_request.pem  newreq.pem
hostcert.pem  newcert.pem
(copy the hostcert.pem file to HostA.)
```

MDS certificate

MDS needs a certificate (server.cert) and a private key (server.key). Example 8-12 presents the process to create the private key and certificate request.

Example 8-12 Creation of MDS private key and certificate request

```
[root@hosta globus]# grid-cert-request -service ldap -host hosta.itso.grid.com
Using configuration from /etc/grid-security/globus-host-ssl.conf
Generating a 1024 bit RSA private key
...(author omits lists)
```

Next, you need to send your certificate request to CA, and request it to sign your certificate. The request file is /etc/grid-security/ldap/ldapcert_request.pem. The CA will sign and send your certificate back to you. Copy this to the file: /etc/grid-security/ldap/ldapcert.pem.

Note: In order to build the grid environment presented at 8.3, “Grid environment” on page 150, copy the certificate request file to the directory in which the demoCA directory is created. Rename it to newreq.pem and sign it, as shown in Example 8-13

Example 8-13 Signing of MDS certificate

```
(copy the ldapcert_request.pem file from HostA to demoCA.)
[root@democa CA]# ls
demoCA  ldapcert_request.pem
[root@democa CA]# cp ldapcert_request.pem newreq.pem
[root@democa CA]# ls
demoCA  ldapcert_request.pem  newreq.pem
[root@democa CA]# export SSLEAY_CONFIG="-config /usr/share/ssl/openssl.cnf"
[root@democa CA]# CA.sh -sign
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
organizationName      :PRINTABLE:'Grid'
organizationName      :PRINTABLE:'Globus'
commonName            :PRINTABLE:'ldap/democa.itso.grid.com'
Certificate is to be certified until Oct 17 15:50:54 2003 GM
T (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
...(author omits lists)
[root@democa CA]# ls
demoCA  ldapcert_request.pem  newcert.pem  newreq.pem
[root@democa CA]# cp newcert.pem ldapcert.pem
[root@democa CA]# ls
demoCA        ldapcert_request.pem  newreq.pem
ldapcert.pem  newcert.pem
(copy the ldapcert.pem file to HostA.)
```

User's certificate

The users need their key pair to be authenticated by the grid servers with a certificate (usercert.pem) and private key (userkey.pem).

Note: In order to build the grid environment presented at 8.3, "Grid environment" on page 150, create, on host A, the private key and certificate request for the user ID griduser, as presented in Example 8-14.

Example 8-14 Creation of user's private key and certificate request

```
[root@hosta globus]# useradd griduser
[root@hosta globus]# passwd griduser
Changing password for user griduser.
New password:
```

```
Retype new password:
passwd: all authentication tokens updated successfully.
[root@hosta globus]# su - griduser
[griduser@hosta griduser]$ export GPT_LOCATION=/usr/local/globus
[griduser@hosta griduser]$ export GLOBUS_LOCATION=/usr/local/globus
[griduser@hosta griduser]$ $GLOBUS_LOCATION/bin/grid-cert-request
Enter your name, e.g., John Smith: ITSO grid user
A certificate request and private key is being created.
You will be asked to enter a PEM pass phrase.
This pass phrase is akin to your account password,
and is used to protect your key file.
If you forget your pass phrase, you will need to
obtain a new certificate.

Using configuration from /etc/grid-security/globus-user-ssl.conf
Generating a 1024 bit RSA private key
.....++++++
.........................................++++++
writing new private key to '/home/griduser/.globus/userkey.pem'
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
...(author omits lists)
[griduser@hosta griduser]$ ls .globus
usercert.pem  usercert_request.pem  userkey.pem
```

The users' certificates are signed by the CA using the certificate request, <User's home directory>/.globus/usercert_request.pem. The CA will sign your certificate and send back you the signed certificate. Copy this to the file <User's home directory>/.globus/usercert.pem.

Note: In order to build the grid environment presented at 8.3, "Grid environment" on page 150, copy the certificate request file to the directory in which the demoCA directory is created. Rename it to newreq.pem and sign it, as presented in Example 8-15.

Example 8-15 Signing of user's certificate

```
(copy the usercert_request.pem file from HostA to demoCA.)
[root@democa CA]# ls
demoCA  usercert_request.pem
[root@democa CA]# cp usercert_request.pem newreq.pem
[root@democa CA]# export SSLEAY_CONFIG="-config /usr/share/s
sl/openssl.cnf"
[root@democa CA]# CA.sh -sign
Using configuration from /usr/share/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
```

```
Signature ok
The Subjects Distinguished Name is as follows
organizationName      :PRINTABLE:'Grid'
organizationName      :PRINTABLE:'Globus'
organizationalUnitName:PRINTABLE:'itso.grid.com'
commonName            :PRINTABLE:'ITSO grid user'
Certificate is to be certified until Oct 17 15:58:27 2003 GM
T (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
...(author omits lists)
[root@democa CA]# ls
demoCA  newcert.pem  newreq.pem  usercert_request.pem
[root@democa CA]# cp newcert.pem usercert.pem
[root@democa CA]# ls
demoCA       newreq.pem    usercert_request.pem
newcert.pem  usercert.pem
(copy the usercert.pem file to HostA.)
```

Testing certificate

After you set up the certificates, you can test them by initiating a **gatekeeper** connection, as follows:

```
grid-proxy-init
globus-personal-gatekeeper -start
```

This command will output a contact string like "`GRAM contact:hosta.itso.grid.com:32872:/O=Grid/O=Globus/OU=itso.grid.com/CN=ITSO grid user`". Substitute that contact string for "<contact>" in the following command.

```
globus-job-run "<contact>" /bin/hostname
```

You will see the host name of your system. If this fails, check the files below:

```
/etc/grid-security/certificates/<hash number>.0
/etc/grid-security/certificates/<hash number>.signing_policy
<User's Home Directory>/.globus/usercert.pem
<User's Home Directory>/.globus/userkey.pem
```

After the verification, you may stop the personal gatekeeper and destroy your proxy with:

```
globus-personal-gatekeeper -killall
grid-proxy-destroy
```

8.5.2 Services setup

GRAM and GridFTP services are daemons initiated by **xinetd** and MDS service is a LDAP daemon.

GRAM and GridFTP

The setup procedure is as follows:

1. Assign the service name `gsigatekeeper` to port 2119 and the name `gsiftp` to port 2811 in /etc/services as follows:

```
# for Globus Toolkit services
gsigatekeeper   2119/tcp                        # Globus Gatekeeper
gsiftp          2811/tcp                        # Globus wu-ftp
```

2. Create the gsigatekeeper and gsiftp configuration files in the /etc/xined.d/ directory. Sample contents are shown in Example 8-16 and Example 8-17. To notify **xinetd** that its configuration file has changed, restart the **xinetd** daemon.

Example 8-16 A sample of gsigatekeeper file

```
service gsigatekeeper
{
   socket_type     = stream
   protocol        = tcp
   wait            = no
   user            = root
   env             = LD_LIBRARY_PATH=/usr/local/globus/lib
   server          = /usr/local/globus/sbin/globus-gatekeeper
   server_args     = -conf /usr/local/globus/etc/globus-gatekeeper.conf
   disable         = no
}
```

Example 8-17 A sample of gsiftp file

```
service gsiftp
{
   instances       = 1000
   socket_type     = stream
   wait            = no
   user            = root
   env             = LD_LIBRARY_PATH=/usr/local/globus/lib
   server          = /usr/local/globus/sbin/in.ftpd
   server_args     = -l -a -G /usr/local/globus
   log_on_success  += DURATION USERID
   log_on_failure  += USERID
   nice            = 10
   disable         = no
```

```
}
```

3. Add the certificate subject corresponding to your user name to the grid-mapfile. The subject is the output of the **grid-cert-info -subject** command, and the user name is the output of the **whoami** command. Now, as root, create the file /etc/grid-security/grid-mapfile with an entry of:

   ```
   “<subject>”        <local username>
   ```

   An example entry is:

   ```
   “/O=Grid/O=Globus/OU=itso.grid.com/CN=ITSO grid user“    griduser
   ```

   Use the following command to make this mapfile:

   ```
   grid-mapfile-add-entry \
   > -dn “‘grid-cert-info -f /home/griduser/.globus/usercert.pem -subject‘” \
   > -ln griduser
   ```

4. Test GRAM and GridFTP with the execution of client commands, as shown in Example 8-18. This test is done under the griduser user ID.

Example 8-18 Test of GRAM and GridFTP

```
[griduser@hosta griduser]$ grid-proxy-init
Your identity: /O=Grid/O=Globus/OU=itso.grid.com/CN=ITSO grid user
Enter GRID pass phrase for this identity:
Creating proxy ........................................
Done
Your proxy is valid until: Wed Oct  2 22:38:57 2002
[griduser@hosta griduser]$ globus-job-run hosta.itso.grid.com /bin/hostname
hosta.itso.grid.com
[griduser@hosta griduser]$ ls
testa
[griduser@hosta griduser]$ globus-url-copy \
> gsiftp://hosta.itso.grid.com/home/griduser/testa \
> file:///home/griduser/testb
[griduser@hosta griduser]$ ls
testa  testb
```

If it fails, check the files below:

```
/etc/grid-security/certificates/<hash number>.0
/etc/grid-security/certificates/<hash number>.signing_policy
/etc/grid-security/hostcert.pem
/etc/grid-security/hostkey.pem
/etc/grid-security/grid-mapfile
<User’s Home Directory>/.globus/usercert.pem
<User’s Home Directory>/.globus/userkey.pem
```

MDS

The setup procedure is as follows:

1. Start the MDS daemon. The official Globus recommendation is that you should *never* run MDS as root. In fact, there is very little in the Globus Toolkit that needs to be installed as root, including OpenLDAP.

   ```
   $GLOBUS_LOCATION/sbin/globus-mds start
   ```

2. Send a test query to GRIS and GIIS under griduser user ID (see Example 8-19).

Example 8-19 Test of MDS

```
[griduser@hosta griduser]$ grid-info-search
version: 1

#
# filter: (objectclass=*)
# requesting: ALL
...(author omits lists)
```

If it fails, check the files below:

```
/etc/grid-security/certificates/<hash number>.0
/etc/grid-security/certificates/<hash number>.signing_policy
/etc/grid-security/ldap/ldapcert.pem
/etc/grid-security/ldap/ldapkey.pem
/etc/grid-security/grid-mapfile
<User's Home Directory>/.globus/usercert.pem
<User's Home Directory>/.globus/userkey.pem
```

Attention: If you had already started OpenLDAP server on the same host, you might be unable to start the MDS daemon. In this case, stop the LDAP daemon.

Note: The demoCA and host A servers are now set up according to the grid environment presented in 8.3, "Grid environment" on page 150.

8.5.3 Adding a new grid server

This section describes how to add a new server to your grid environment.

Note: In order to build the grid environment presented in 8.3, "Grid environment" on page 150, you need to add the host B server as a grid node, as explained in "Setting up a new server" on page 169.

Setting up a new server

Follow these steps:

1. Installation of Globus Toolkit

   Install the GPT and bundles to host B according to 8.4.1, “Installation of GPT” on page 151 and 8.4.2, “Installation of bundles” on page 152.

2. Setup of server certificates

   Set up the server certificates according to “Server certificates” on page 159.

3. Setup of user certificate

   Host A and host B share the same grid user ID, so you do not need to create a new user certificate. Copy the files for the user’s certification from host A:

   ```
   scp -r griduser@hosta.itso.grid.com:/home/griduser/.globus /home/griduser
   ```

4. Server setup

   Set up the server according to 8.5.2, “Services setup” on page 166. If you want to change the local user, you need to change the local user name in the grid-mapfile.

8.6 Additional configurations

Here we present some additional sections on how to configure the services daemons beyond the default configuration.

8.6.1 GRAM

Here we present two advanced configurations, adding a new job manager and adding and enabling a GRAM reporter.

Adding a job manager

The Globus Toolkit provides a job manager, in addition to the default **fork** job manager, to help you expand the job management capability. The job manager supports job submissions to some third-party vendor’s job schedulers, such as PBS, LSF, Condor, and LoadLeveler.

The job manager is created by the gatekeeper to satisfy every request submitted to the gatekeeper. By default, the job manager is the process that executes the job request from the gatekeeper as a "forked" child process.

Globus Toolkit 2.2.2 provides an extra packages to help you install the job managers for third-party schedulers. The packages are available at:

ftp://ftp.globus.org/pub/gt2/2.2/2.2.2/extra/gram_job_manager

As presented in Figure 8-2, the job manager for the PBS job scheduler has been chosen. PBS is installed on host B and host C and it contains three daemons:

- pbs_mom: Execution daemon
- pbs_sched: Scheduler daemon
- pbs_server: Management daemon

The pbs_mom daemon is needed on each system where jobs are expected to run (on host B and host C). The pbs_sched and pbs_server daemons are needed on the system where the job manager and job queues exist (on host B only).

When a client (on host A) submits a PBS job to a gatekeeper (on host B), the job manager for PBS runs and then PBS sends the job to executed in another machine (host C), where Globus Toolkit is not installed.

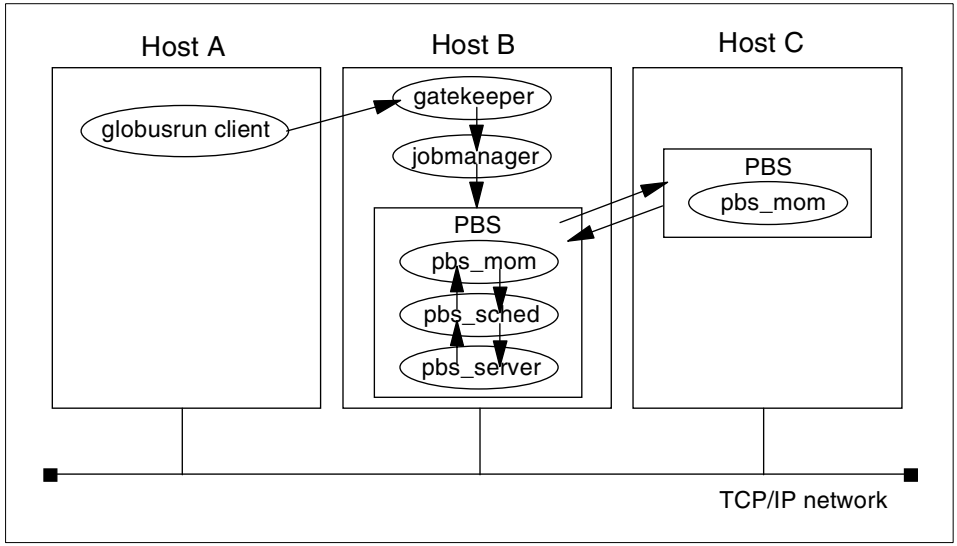

Figure 8-2 System overview after PBS installation

Adding the GRAM reporter

The GRAM reporter provides status information for the job manager. The GRAM reporter for the "**`forked`**" job manager is contained in the server bundle. This section describes the activation of GRAM reporter for the default job manager and for the PBS job manager.

Activation of GRAM Reporter for default job manager

Do these steps:

1. Edit the $GLOBUS_LOCATION/etc/globus-job-manager.conf file and add the following lines to the configuration file:

   ```
   -publish-jobs
   -job-reporting-dir=<GLOBUS_LOCATION>/tmp/gram_job_reporter
   ```

   Change the <GLOBUS_LOCATION>, depending on your environment.

2. Create the directory in which GRAM reporter places the temporary files. The command is as follows:

   ```
   mkdir <GLOBUS_LOCATION>/tmp/gram_job_reporter
   chmod 777 <GLOBUS_LOCATION>/tmp/gram_job_reporter
   ```

3. Restart the **xinetd** daemon:

   ```
   /etc/init.d/xinetd restart
   ```

Adding and activation of GRAM reporter for PBS job manager

The GRAM Reporter publishes the job manager status information into MDS. To enable it, you need to install the globus-gram-reporter package, plus one jobmanager-specific setup package, as presented in Example 8-20.

Example 8-20 Installing the globus-gram-reporter package

```
[root@democa gt222]# gpt-build globus_gram_reporter_setup_pbs-1.0.tar.gz
gpt-build ====> CHECKING BUILD DEPENDENCIES FOR globus_gram_reporter_setup_pbs
gpt-build ====> Changing to
/home/globus/gt222/BUILD/globus_gram_reporter_setup_pbs-1.0/
gpt-build ====> BUILDING FLAVOR
...(author omits lists)
[root@democa etc]# gpt-postinstall
running /usr/local/globus222/setup/globus/setup-globus-gram-reporter-pbs...
Setting up pbs gram reporter in MDS
----------------------------------------
loading cache /dev/null
checking for qstat... /usr/pbs/bin/qstat
updating cache /dev/null
creating ./config.status
creating /usr/local/globus222/libexec/globus-script-pbs-queue
Done
```

As illustrated in Figure 8-3 on page 172, GRIS now can get status information using the following command:

```
/usr/local/globus222/libexec/globus-gram-reporter -conf \
> /usr/local/globus222/etc/globus-job-manager.conf \
> -type pbs -rdn jobmanager-pbs -dmdn \
> Mds-Host-hn=democa.itso.grid.com,Mds-Vo-name=local,o=grid
```

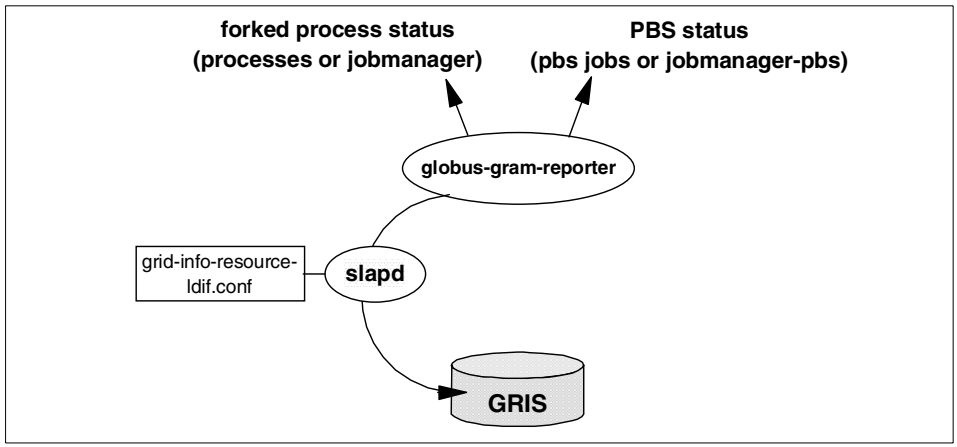

Figure 8-3 Behavior of globus-gram-reporter

8.6.2 MDS

MDS is based on OpenLDAP, allowing you to created your own configuration of hierarchical GIIS.

For more informations regarding MDS and LDAP customization, please refer to the following Web site at:

http://www.globus.org/

Configuration of hierarchical GIIS

MDS supports a hierarchical structure for GIIS similar to the Domain Name Servers hierarchy. In Figure 8-4 on page 173, we present an overview of a hierarchical GIIS. The boxes represent the configuration files and the ovals represent the programs and daemons that update the resource information.

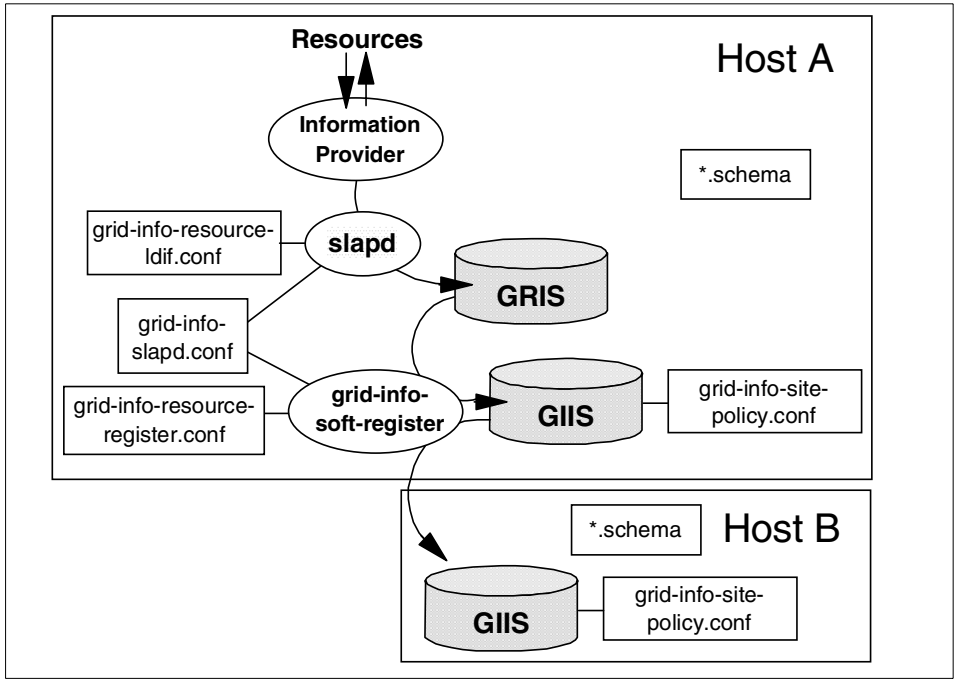

Figure 8-4 Overview of a hierarchical GIIS structure and configuration files

Programs and daemons

The slapd is one of two daemon processes that **`globus-mds`** executes. This triggers the information providers that output the local resource information and send it to the GRIS database, conforming to the grid-info-resource-ldif file format.

The grid-info-soft-register is the other daemon process that **`globus-mds`** executes. This is a shell script that registers the GRIS information with the GIIS database at regular intervals.

The information providers are the programs that output the local resource information in LDIF format (LDAP Data Interchange Format).

Hierarchical structure

An example of a hierarchical structure is presented in Figure 8-4, where GRIS (on host B) registers GIIS (also on host B) and GIIS (on host A) registers GIIS (on host B).

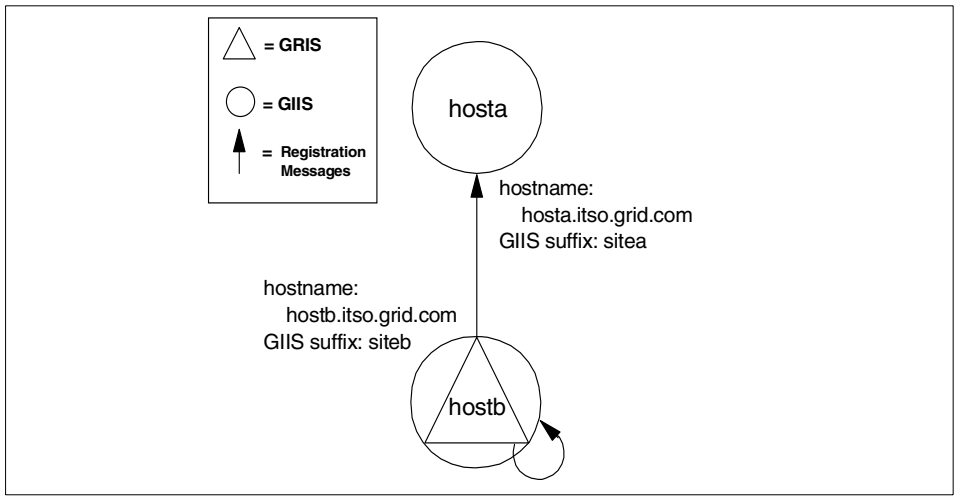

Figure 8-5 The abstract figure of a hierarchical GIIS

8.7 Client interface

This section presents the client interfaces for the following services of Globus Toolkit:

- GRAM
- MDS
- GridFTP

Before using any of these client tools, you must initialize your environment and create a proxy with the **`grid-proxy-init`** command.

Detailed information about all Globus Toolkit tools is described at:

http://www.globus.org/toolkit/

8.7.1 Client interface for GRAM

GRAM has the following client commands to submit and manage jobs on the grid environment.

globus-job-run

This is an online interface for job submissions. It is the easiest command to use to submit a job and returns the output of its result. This data-staging function is realized by the GASS server that stages the executables on remote machines.

The basic command syntax is:

```
globus-job-run <contact string> <command>
```

where <contact string> specifies a machine's host name, port, and service to which to send the request. The syntax of a contact string is host:port/jobmanager-name. The default port is 2119 and the default job manager's name is "jobmanager".

Here is an example of how to submit a program:

```
globus-job-run hosta.itso.grid.com -s program_name
```

globus-job-submit

This is an interface for batch job submissions. It will immediately return with an URL (with the job contact string embedded) that you can use to query the status of your job. The command is similar to **globus-job-run**, but the **globus-job-submit** command does not return the output of its result. To obtain the output, you need to run the job management commands (as seen in **globus-job-status**, **globus-job-get-output**, and **globus-job-clean/globus-job-cancel**) pointing to the URL generated as result of the **globus-job-submit** execution. An example job contact string is:

```
https://hosta.itso.grid.com:33318/1137/1033581329/
```

The basic command syntax is as follows:

```
globus-job-submit <contact string> command
```

globusrun

This is a command that gives access to the RSL, the language which provides a common interchange to describe resources. More information about RSL can be found at the following Web site:

http://www-fp.globus.org/gram/rsl_spec1.html

The **globus-job-run** and **globus-job-submit** commands are both shell script wrappers around **globusrun**.

The basic command syntax is:

```
globusrun <contact string> <RSL>
```

An example would be the following:

```
globusrun -o -r hosta.itso.grid.com \
> &(executable=/bin/ls)(arguments=-l)(count=1)
```

Here we have typical examples of using the **globusrun** command:

- How to run your program at a single job manager.

  The following example shows how to run myprogram for five times:

  ```
  globusrun -o -r hosta.itso.grid.com -s '&(executable=myprogram)(count=5)'
  ```

- How to write standard output to the remote file.

  The following example shows how to write standard output of myprogram to a remote file:

  ```
  globusrun -r hosta.itso.grid.com \
  > '&(executable=myprogram)(directory=/home/griduser)(stdin=myprog.output)'
  ```

- How to submit multi-requests to different hosts.

  The following example shows how to submit multi-requests to different job managers at different hosts:

  ```
  globusrun \
  > '+(&(resourceManagerContact="hosta.itso.grid.com")(executable=myprog1))
  >   (&(resourceManagerContact="hostb.itso.grid.com")(executable=myprog2)))'
  ```

globus-job-status

This is a job management command that returns a job status of one of the following:

- pending
- active
- done
- failed
- others

For example:

```
globus-job-status https://hosta.itso.grid.com:33318/1137/1033581329/
```

globus-job-get-output

This is a job management command that collects the output when the job finishes.

globus-job-clean/globus-job-cancel

This is a job management command that stops the job if it is running, and cleans up the cached copy of the output.

8.7.2 Client interface for MDS (GRIS and GIIS)

MDS has a client command to query for details about resources in the grid environment.

grid-info-search

This command sends one or more queries to GRIS and GIIS and displays the result in the standard output. The queries are RFC1558 compliant with the LDAP search filter, since the command embeds the **ldapsearch** command.

The basic command syntax is:

```
grid-info-search [options] <filter> [attributes...]
```

The following are typical examples of using the **grid-info-search** command:

- How to query all objects on a GRIS

  The following example shows how to display all of data objects and resources on a single, local machine set up as a GRIS:

  ```
  grid-info-search -x -b ‘Mds-vo-name=local,o=Grid’
  ```

  This search uses anonymous binding (-x) and runs on the local host with the default port of 2135. Mds-Vo-name=local means that the search will start on a GRIS.

- How to query a file system space on a GIIS

  The following example shows how to query for the amount of free file system space on all machines on a GIIS:

  ```
  grid-info-search -x -h hosta.itso.grid.com -p 2135 \
  > -b 'Mds-Vo-name=site,o=Grid' Mds-Fs-freeMB
  ```

  This search uses anonymous binding (-x) and runs on the host and port indicated. Mds-Vo-name=site means that the search will start on a GIIS that is named “site”.

- How to query CPU data on a single machine on a GIIS

  The following example shows how to query for CPU model and speed on a single machine on a GIIS:

  ```
  grid-info-search -x -h hosta.itso.grid.com -p 2135 \
  > -b 'Mds-Vo-name=site,o=Grid' \
  > '(&(objectclass=MdsCpu)(Mds-Host-hn=hosta.itso.grid.com))' \
  > Mds-Cpu-model Mds-Cpu-speedMHz
  ```

  This search uses anonymous binding (-x) and runs on the host and port indicated. Mds-Vo-name=site means that the search will start on a GIIS. Your GIIS may be named something other than the default of site. The objectclass

expression specifies CPU data on a specific machine on the GIIS. The last two arguments specify the CPU model and speed.

8.7.3 Client interfaces for GridFTP

GridFTP provides a client command to copy files between local and remote locations.

globus-url-copy

The basic command syntax is:

```
globus-url-copy [options] <sourceURL> <destURL>
```

where:

- <sourceURL> is the URL to source file, or '-' for standard input.
- <destURL> is the URL of the destination file, or '-' for standard output.

Basically, all protocols are supported by GASS, such as:

- local file
- http
- https
- gsiftp

Example URLs are:

```
gsiftp://hosta.itso.grid.com/home/griduser/testfile
file:///home/griduser/testfile
```

Here is an example of the execution of the **globus-url-copy** command:

```
globus-url-copy gsiftp://hosta.itso.grid.com/home/griduser/testa \
> file:///home/griduser/testb
```

9

Demo: Grid setup

This chapter is a cookbook for setting up the grid environment to be used to deploy the video conversion demo application. This includes installing Red Hat Linux, Network Time Protocol, Globus, and a Certificate Authority. Chapter 10, “Demo: Application” on page 197 explains the demo application, how to install it, and run it on this grid.

The following topics are discussed:

- Required software
- Hardware environment
- Operating system installation
- Globus installation and setup
- CA installation and setup

9.1 Required software

Globus Toolkit Version 2.2 is used in the demo. Globus Toolkit supports Red Hat Linux on xSeries and AIX on pSeries. Globus Toolkit 2.0 is also available as part of the SuSE Linux Enterprise Server 8 (SLES 8) on zSeries.

For the purpose of this redbook, we selected Red Hat Linux as our host operating system. Although the lab environment to produce the redbook had been configured with Red Hat Version 7.3, Version 8.0 can also be used with no restrictions.

This is the list of required files to be downloaded:

- Globus Packaging Technology
  - gpt-2.2.2-src.tar.gz
- Globus client
  - globus-all-client-2.2.3-i686-pc-linux-gnu-bin.tar.gz
- Server bundle
  - globus-all-server-2.2.3-i686-pc-linux-gnu-bin.tar.gz
- Certificate Authority
  - globus_simple_ca_bundle-0.9.tar.gz
- Network Time Protocol
  - ntp-4.1.1-1.i386.rpm

Place the Globus *.gz files in the /usr/src directory. These files can be downloaded from the official site of Globus Project at:

http://www.globus.org/gt2.2/download.html

or

ftp://ftp.globus.org/pub/gt2/2.2/

The NTP package may already be installed, perhaps at a newer level, depending on the version of Linux installed. If not, download and install it.

The **md5sum** command can be used to check if you are using the same versions of the files we used by comparing the signature values against this list:

```
765c5ec7f51b650eb8474baf009d8852  gpt-2.2.2-src.tar.gz
dd9453e0503350dcc93b9b1693682a89
                        globus-all-client-2.2.3-i686-pc-linux-gnu-bin.tar.gz
9ed2ac41dd2cf3ab27084f3d754341f5
                        globus-all-server-2.2.3-i686-pc-linux-gnu-bin.tar.gz
```

```
492e9386ac66f80c36fefe959af653a3  globus_simple_ca_bundle-0.9.tar.gz
f06d8293b290f92a137ce67450cf542f  ntp-4.1.1-1.i386.rpm
```

9.2 Setting up the environment

An Ethernet LAN and five Intel Pentium machines were used to build the demo application environment. In Figure 9-1 on page 182, we illustrate this environment with the host names and functions to be installed in each machine. The host names are alpha, beta, gamma, delta, and zeta. The machines should have a clock speed of at least 500 Mhz, at least 256 MB of memory, and at least 8 GB hard drives. The zeta machine should have at least a 40 GB hard drive for any serious video conversions. It is also recommended that the zeta machine have a 1Ghz processor or faster for improved video capture reliability.

The following steps should be performed in the order presented in this chapter. There are dependencies among them that required performing them in this order. The major steps to set up the grid environment include installing:

- Red Hat Linux on each machine
- Network Time Protocol server on one machine (for example, alpha) and configuring NTP clients for the others (beta, gamma, delta, and zeta)
- Globus Packaging Technology on each machine
- Globus Server on the alpha, beta, gamma, and delta machines
- Globus Client on zeta
- Globus Simple Certificate Authority on alpha

The grid is configured using these major steps:

- Sign the certificate requests from all components and users needing them
- Set up gridmap files for each system
- Set up automated grid startup
- Set up each GRIS to talk to one GIIS
- Set up MDS security

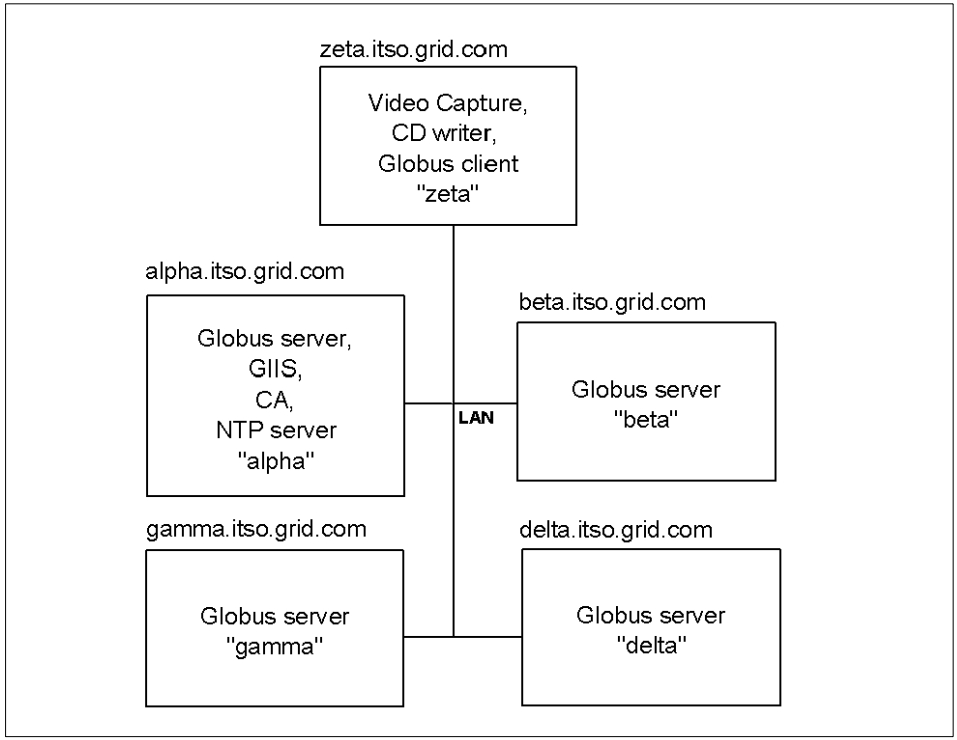

Figure 9-1 Hardware environment and software functions of each machine

9.2.1 Naming and addressing planning

Below, we show the tables we have filled for naming and address planning.

Table 9-1 summarizes the names of the machines to be used in the grid, their IP addresses, and the software to be installed on them.

Table 9-1 Host names and IP addressing

Host name	IP	Software
alpha.itso.grid.com	192.168.0.4	Globus server, CA, and NTPserver
beta.itso.grid.com	192.168.0.5	Globus server
gamma.itso.grid.com	192.168.0.6	Globus server
delta.itso.grid.com	192.168.0.7	Globus server

Host name	IP	Software
zeta.itso.grid.com	192.168.0.8	Globus client and video apps

Before you install the Globus Simple Certificate Authority, you must define the distinguished name (DN) that will be used by the CA in your environment. DN is defined by CCITT 1988 recommendation x.509, which determines a framework for authentication of objects included in a distributed directory service. Table 9-2 describes the distinguished name used for the Certificate Authority in our environment. In a real production environment, this should be globally unique. The distinguished names for the users and for the Globus services will be generated automatically.

Table 9-2 CA distinguished name and passphrase

Certificate Authority DN	Passphrase
`cn=my test CA, ou=alpha.itso.grid.com, ou=demotest, o=grid`	mycapw

The distinguished name (DN) and passphrase will be used by the Certificate Authority to sign certificate requests.

We recommend that you define the group and user IDs that you want to use before implementation. Table 9-3 has some suggested user and group IDs and passwords.

Table 9-3 User ID and group ID

User ID	Group ID	alpha pw	beta pw	gamma pw	delta pw	zeta pw
root	root	pwxjhsa	pwxjhsb	pwxjhsg	pwxjhsd	pwxjhsz
globususer	globus	pwcjsna	pwcjsnb	pwcjsng	pwcjsnd	pwcjsnz
snobol	snobol	-no id-	-no id-	-no id-	-no id-	kjsdnb
adminca	adminca	nslcxa	-no id-	-no id-	-no id-	-no id-

The root ID is used on all machines. A cell containing “-no id-” means that the corresponding machine does not have that user ID installed on it.

The globususer ID is used to run jobs on the grid for the user and to FTP files during installation. Note that since this user ID has more than eight characters, you will need to install it later, rather than installing it as part of the Linux install process. The other user IDs can be installed as part of the Linux installation or later.

The snobol ID is used to submit jobs to the grid.

The adminca ID is used to receive certificate requests for the Certificate Authority. This could be done via e-mail to this account. However, we will not describe the details of configuring e-mail. The adminca ID could be used to ftp the certificate requests to the alpha machine in our demo. The certificates will be signed using the root ID on machine alpha. In a production system, the Certificate Authority should follow more script procedures for identifying requesters and handling certificates. For the purposes of this demo, we will leave the decision of how to transfer certificate requests and signed certificates between the machines as an exercise for the reader.

9.2.2 Install Linux

Install Linux on all of the machines using the “custom” install, selecting “everything” and “no firewall.” Each system should use a fixed network IP address with a corresponding host name, per Table 9-1 on page 182, and do not use DHCP. Use Table 9-3 on page 183 to create user IDs (except globususer) at install time. Otherwise, the IDs can be added later. After the first boot, you will probably want to use the **ntsysv** command to enable **telnet** and **wu-ftpd**. This will make it easier to move files (using FTP) and to execute the installation procedures from one terminal (using **telnet**). To make FTP less restrictive, edit the file /etc/ftpaccess on each machine and comment out the guestuser line as follows:

```
#guestuser         *
```

To enable **telnet**, a reboot will be required after configuring with **ntsysv**. If you have trouble configuring **wu-ftpd** to your liking, you can try using **vsftpd** instead in Red Hat Linux 8.0.

9.2.3 Installing Network Time Protocol (NTP)

NTP needs to be installed because the grid needs the clocks on the systems to be synchronized. The security process creates proxy certificates that are valid for specific times. If the systems do not have their clocks synchronized, then the users may not be able to use the grid, because the proxy certificates may look like they have expired or are not yet valid.

On all of the grid machines, install NTP as follows using the root ID:

```
$ rpm -ivh /usr/src/ntp-4.1.1-1.i386.rpm
```

If the package is already installed as part of the Linux distribution, ignore the error message and continue to set up the NTP server. Proceed by setting up the server and daemons.

Edit the /etc/ntp.conf file on the machine designated to be the time server, machine alpha, and leave the following four lines as the only un-commented ones, commenting all others with a leading “#” character:

```
server 127.127.1.0 # local clock
fudge 127.127.1.0 stratum 10
driftfile /etc/ntp/drift
broadcastdelay 0.008
```

Also, on the NTP server machine (alpha), use the **ntsysv** command to enable the NTP daemon (ntpd) on the next reboot.

On the other machines in the grid (beta, gamma, delta, and zeta), change the /etc/ntp.conf file to leave only the following lines un-commented:

```
server alpha.itso.grid.com
driftfile /etc/ntp/drift
broadcastdelay 0.008
authenticate no
```

Next, execute the following command to have them check for the time from the above server machine alpha:

```
ntpdate -b alpha.itso.grid.com
```

This should be executed at least once per boot, and could be set up to run periodically using **crond** and **crontab**.

9.2.4 Set up other global items on each machine

Unless your systems are entered into a name server, you will want to list them in the /etc/hosts hosts file with the following lines, adjusting the IP addresses to match those assigned for your grid machines:

```
127.0.0.1 localhost
192.168.0.4 beta.itso.grid.com beta
192.168.0.5 zeta.itso.grid.com zeta
192.168.0.6 alpha.itso.grid.com alpha
192.168.0.7 delta.itso.grid.com delta
192.168.0.8 gamma.itso.grid.com gamma
```

Be certain to put the long name before the short name; otherwise, Globus will have some problems when it tries to do a reverse lookup of the fully qualified names for IP addresses.

Verify machine connectivity after the next reboot, using the **ping** command to ping each of the other machines by name.

As root, edit the /etc/profile file in each machine. pico is a simple convenient editor to use. Insert the following three lines after the line in /etc/profile that says "`export PATH USER ...`":

```
export GPT_LOCATION=/usr/local/gpt
export GLOBUS_LOCATION=/usr/local/globus
export PATH=$PATH:$GLOBUS_LOCATION/bin:$GLOBUS_LOCATION/sbin
```

Log off and log back on the machines after modifying the /etc/profile file so that the above settings take effect.

If the other user IDs in Table 9-3 on page 183 were not created while installing Linux, they can be added now.

Here is an example of how to add the globususer ID. Create a globus group and globus user ID under which the grid jobs will run for each server machine.

Add a group for globus by executing:

```
groupadd -g 900 globus
```

Add the user globususer (with password globususer) by executing:

```
adduser -u 900 -g globus -d /home/globususer -n globususer
```

Change the globususer ID's password from globususer to `pwcjsna` or other password by executing:

```
passwd globususer
```

Repeat the procedure on all of the machines, adding the IDs listed under the user IDs in Table 9-3 on page 183.

9.2.5 Installing the GPT

Log on as root and install GPT on all of the machines. Please ignore all warnings from Globus:

```
cd /usr/src
tar -xzvf gpt-2.2.2-src.tar.gz
cd gpt-2.2.2
./build_gpt
ls ${GPT_LOCATION}/sbin | wc -l
```

The final **`ls`** command should show 29 gpt-* executable files.

9.2.6 Installing a Globus server bundle

The following is used to install the server bundle on each server machine. Perform these steps on each machine that will be a server. In our demo, we will use machines alpha, beta, gamma, and delta as servers.

As root, run:

```
cd /usr/src
export PATH=$PATH:$GPT_LOCATION/sbin
gpt-install globus-all-server-2.2.3-i686-pc-linux-gnu-bin.tar.gz
gpt-postinstall
/usr/local/globus/setup/globus/setup-gsi
y
q
```

9.2.7 Installing a Globus client bundle

The following is used to install the client bundle on any machines that will be used to query or submit jobs to the grid. In our demo, we will install the client on the zeta machine.

As root, run:

```
cd /usr/src
export PATH=$PATH:$GPT_LOCATION/sbin
gpt-install globus-all-client-2.2.3-i686-pc-linux-gnu-bin.tar.gz
gpt-postinstall
/usr/local/globus/setup/globus/setup-gsi
y
q
```

9.2.8 Installing the Globus Simple Certificate Authority

To install the Globus Simple Certificate Authority, one of the Globus bundles (server or client) needs to be installed on the machine due to a dependency. We will install the CA and a Globus server on the “alpha” machine.

As root, run:

```
cd /usr/src
export PATH=$PATH:$GPT_LOCATION/sbin
gpt-build -nosrc gcc32
gpt-build globus_simple_ca_bundle-0.9.tar.gz gcc32
gpt-postinstall
...
Do you want to keep this as the CA subject (y/n) [y]: n
Enter a unique subject name for this CA:
   cn=my test CA, ou=alpha.itso.grid.com, ou=demotest, o=grid
```

```
Enter the email of the CA:
adminca@alpha.itso.grid.com
[default 5 years] 1825                    <- use the default by pressing enter
Enter PEM pass phrase:
mycapw
mycapw
[enter]
```

During the above process, a hash number is generated and used as part of the file name. Please note this number for use in the next steps. Run the script name printed at the end of the prior install, substituting the hex hash number printed by the above process in place of the <hash> shown below, adding the "-default" argument:

```
/usr/local/globus/setup/globus_simple_ca_<hash>_setup/setup-gsi -default
y
q
```

For your information, the /root/.globus/simpleCA/private/cakey.pem file is the CA's private key and should not be given out to anyone else. The /root/.globus/simpleCA/cacert.pem file contains the CA's public key.

The following is used to install the CA's certificate on each of the other grid machines. /root/.globus/simpleCA/globus_simple_ca_<hash>_setup-0.9.tar.gz is the file containing the public CA key and other information needed to participate in this grid. This must be copied to each of the other machines and installed using the **gpt-build** command.

First, on machine alpha, use ftp to copy the /root/.globus/simpleCA/globus_simple_ca_<hash>_setup-0.9.tar.gz file to the /usr/src/ directory of each of the other grid machines. This can be done in two steps by ftp-ing them to the /home/globususer directory on each of those machines using globususer ID. Then, using root, this file can be moved to the /usr/src directory. Next, issue the following commands on each of those machines as root:

```
gpt-build /usr/src/globus_simple_ca_<hash>_setup-0.9.tar.gz
gpt-postinstall
/usr/local/globus/setup/globus_simple_ca_<hash>_setup/setup-gsi  -default
y
q
```

9.2.9 Requesting and signing gatekeeper certificates for servers

On each of the server machines (alpha, beta, gamma, and delta), we perform the following steps to request and sign certificates:

```
grid-cert-request -host <hostname of requesting server machine>
```

Use ftp or e-mail (if available and using the adminca ID) to copy the /etc/grid-security/hostcert_request.pem file to the CA machine and put it into the `/root` directory. On the CA machine, as root, sign the certificate using the following:

```
grid-ca-sign -in /root/hostcert_request.pem -out /root/hostcert.pem
mycapw
```

Then, ftp the /root/hostcert.pem file back to the server machine and place it in the /etc/grid-security directory.

9.2.10 Requesting and signing user certificates

For each user who will use the grid (in our example, user snobol on the client machine zeta), the following procedure must be executed by the user and Certificate Authority. On the snobol user's logon, run:

```
grid-cert-request
<userspassphrase>              <- see paragraph below about this
<userspassphrase>
```

The user should make up his own passphrase for his certificate. He will use this same passphrase later with the **`grid-proxy-init`** command to authenticate with the grid. In our example, the snobol user's login password could be used here.

The user must then send the /home/<userid>/.globus/usercert_request.pem file to the Certificate Authority (machine alpha) for signing. On the CA machine (alpha), sign the certificate using root with the following command, adjusting the location of usercert_request.pem to point to wherever the above request file is now stored on alpha:

```
grid-ca-sign -in usercert_request.pem -out usercert.pem
mycapw
```

Securely send the usercert.pem file back the requesting user. The user should put the usercert.pem file into his /home/<userid>/.globus directory.

The user should also be added to the grid mapfile (on machine alpha under root) using the following command (note the backward apostrophe characters next to the double quote characters):

```
grid-mapfile-add-entry -dn "`grid-cert-info -f usercert.pem -subject`" -ln
globususer
```

Copy gridmapfile to /etc/grid-security/grid-mapfile to each of the other servers (beta, gamma, and delta) so that all of the servers have this file.

9.2.11 Setting up the gatekeepers

On each server (alpha, beta, gamma, and delta), add the following two lines to the /etc/services file:

```
gsigatekeeper 2119/tcp #globus gatekeeper
gsiftp 2811/tcp #globus wuftp
```

Create the file /etc/xinetd.d/gsigatekeeper on each server, containing the lines:

```
service gsigatekeeper
{
   socket_type     = stream
   protocol        = tcp
   wait            = no
   user            = root
   env             = LD_LIBRARY_PATH=/usr/local/globus/lib
   server          = /usr/local/globus/sbin/globus-gatekeeper
   server_args     = -conf /usr/local/globus/etc/globus-gatekeeper.conf
   disable         = no
}
```

Create the file /etc/xinetd.d/gsiftp on each server, containing the lines:

```
service gsiftp
{
   instances       = 1000
   socket_type     = stream
   wait            = no
   user            = root
   env             = LD_LIBRARY_PATH=/usr/local/globus/lib
   server          = /usr/local/globus/sbin/in.ftpd
   server_args     = -l -a -G /usr/local/globus
   log_on_success  += DURATION USERID
   log_on_failure  += USERID
   nice            = 10
   disable         = no
}
```

Now reboot all of the machines.

9.3 Setting up MDS

We will configure the Monitoring and Discovery Service (MDS) to have one Grid Information Index Service (GIIS) in the alpha machine, which collects the data reported by the Grid Resource Information Servers (GRIS) in all of the machines. The GRIS servers send information about their respective servers to the GIIS. In

the demo application, we will use this to find machines that are not busy. The user will be able to query the GIIS from the zeta client machine.

To set up this structure, we need to modify several configuration files. These files name the GIIS and GRIS, and show how these components should register with each other. Figure 9-2 shows the relationship among the MDS components in our example.

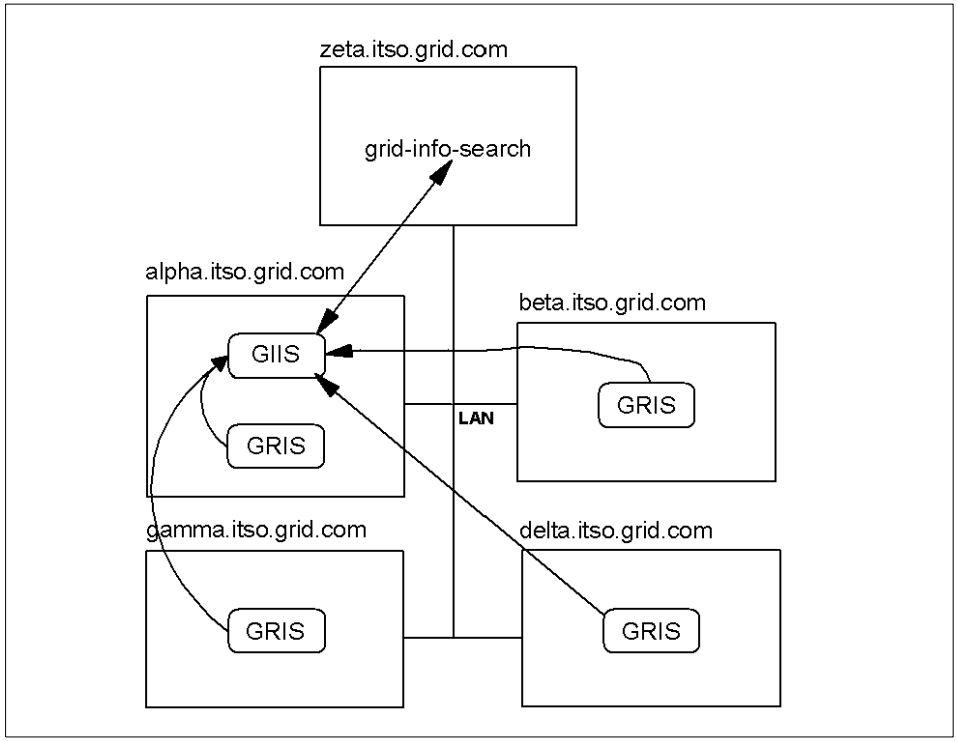

Figure 9-2 MDS configuration

9.3.1 Setting up the GIIS and GRIS on the alpha machine

On alpha, make the following modifications to the conf files in the $GLOBUS_LOCATION/etc directory.

In the grid-info-slapd.conf file, name the GIIS on machine alpha. Change the second of the lines:

```
database     giis
suffix       "Mds-Vo-name=site, o=Grid"
```

to

```
database     giis
suffix       "Mds-Vo-name=alpha.itso.grid.com, o=Grid"
```

In the grid-info-site-policy.conf file, allow registrations from the domain. Change the second of the lines:

```
policydata:
(&(Mds-Service-hn=site) (Mds-Service-port=2135))
```

to

```
policydata
(&(Mds-Service-hn=*.itso.grid.com) (Mds-Service-port=2135))
```

In the grid-info-resource-register.conf file, tell the alpha GRIS to register with the alpha GIIS. Change the two matching lines to the settings shown below:

```
dn: Mds-Vo-Op-name=register, Mds-Vo-name=alpha.itso.grid.com, o=grid
reghn: alpha.itso.grid.com
```

9.3.2 Setting up the GRIS on beta, gamma, and delta

On all of the other server machines (beta, gamma, and delta), make the following modifications to the conf files in the $GLOBUS_LOCATION/etc directory.

In the grid-info-slapd.conf file, remove the GIIS server from these machines. Remove the block of lines starting with the following lines:

```
database     giis
suffix       "Mds-Vo-name=site, o=Grid"
```

In the grid-info-resource-register.conf file, tell the GRIS which GIIS to register with. Change the two matching lines as shown below:

```
dn: Mds-Vo-Op-name=register, Mds-Vo-name=alpha.itso.grid.com, o=grid
reghn: alpha.itso.grid.com
```

9.3.3 Start the MDS on all of the servers

Start the MDS on all of the servers (alpha, beta, gamma, and delta) using:

```
globus-mds start
```

This can be automated by putting it in /etc/rc.d/rc.5 per the usual conventions. Copy the **`globus-mds`** script into the directory /etc/init.d/. Then create two symbolic links as follows:

```
cp $GLOBUS_LOCATION/sbin/globus-mds /etc/init.d/
cd /etc/rc.d/rc5.d/
```

```
ln -s /etc/init.d/globus-mds S92globus-mds
ln -s /etc/init.d/globus-mds K92globus-mds
```

9.3.4 Setting up the MDS client zeta

Modify the $GLOBUS_LOCATION/etc/grid-info.conf file lines shown below so that searches go to the GIIS on machine alpha:

```
GRID_INFO_HOST="alpha.itso.grid.com"
GRID_INFO_ORGANIZATION_DN="Mds-Vo-name=alpha.itso.grid.com, o=Grid"
```

9.3.5 Setting up a secure MDS

So far, we have set up an MDS that permits anonymous access. The **grid-info-search** command should use the -x flag to indicate an anonymous search request. However, the MDS can be secured so that only certified users can access the GIIS and only certified server GRISs can register to send information to the GIIS. The following steps should be performed.

Request and sign certificates for each server machine

For each of the server machines (alpha, beta, gamma, and delta) request LDAP certificates, sign them using the Certificate Authority on alpha, and copy the signed certificates to the proper location. The steps for one of the servers (beta) is shown here.

On the server machine (beta) under root, run:

```
grid-cert-request -service ldap -host beta.itso.grid.com
```

Copy the request certificate from /etc/grid-security/ldap/ldapcert_request.pem to the Certificate Authority machine (alpha) using ftp or any other desired method. Sign the certificate using root on alpha substituting the correct locations for the request certificate and signed certificates:

```
grid-ca-sign -in ldapcert_request.pem -out ldapcert.pem
```

Copy the resulting signed certificate file ldapcert.pem from the Certificate Authority machine (alpha) to the file the server machine (beta) location /etc/grid-security/ldap/ldapcert.pem.

Change the conf files

Change the following configuration files on the servers.

Change $GLOBUS_LOCATION/etc/grid-info-slapd.conf to change the anonymousbind setting(s) as follows:

```
anonymousbind   no
```

Change the $GLOBUS_LOCATION/etc/grid-info-resource-register.conf files on the servers to require authentication when registering:

```
bindmethod: AUTHC-ONLY
```

After making all of these changes, the server machines should be rebooted or the following should be used to restart the MDS on each of the servers (alpha, beta, gamma, and delta):

```
globus-mds stop
globus-mds start
```

9.4 Checking the installation

Check the installations on each machine as root using the command:

```
$GPT_LOCATION/sbin/gpt-verify
```

The following commands can be used on a server machine to see if the GRAM and GridFTP are listening on their respective ports:

```
netstat -an | grep 2119
netstat -an | grep 2811
```

From the client machine (zeta) logged on as the user snobol, do the following:

This command sets up the environment so that Globus commands can be issued by the user. One may want to add this line to one's login profile:

```
. $GLOBUS_LOCATION/etc/globus-user-env.sh
```

This command refreshes the proxy certificate for the user (snobol):

```
grid-proxy-init
<userspassphrase>
```

The following commands send a simple job to the server machine and has it return its host name. This test whether jobs can be submitted to each of the server machines:

```
globus-job-run alpha.itso.grid.com "/bin/hostname"
globus-job-run beta.itso.grid.com "/bin/hostname"
globus-job-run gamma.itso.grid.com "/bin/hostname"
globus-job-run zeta.itso.grid.com "/bin/hostname"
```

The following command tests whether a file can be copied to the target server machine and then submits a job to list the directory contents:

```
echo hello >testfile
globus-url-copy file://$HOME/testfile \
   gsiftp://alpha.itso.grid.com/home/globususer/testfile
```

```
globus-job-run alpha.itso.grid.com "/bin/ls" "-trlc"
```

The following submits an insecure MDS search request. Remove the "-x" parameters for the secure version:

```
grid-info-search -x "(objectclass=MdsCpu)" Mds-Cpu-Free-1minX100
```

To refine the search to look for processors that have more than 90% free of CPU utilization for the last minute, use:

```
grid-info-search -x \
   "(&(Mds-Device-Group-name=processors)(Mds-Cpu-Free-1minX100>=90))" \
   Mds-Cpu-Free-1minX100
```

You should now be ready to install and run the video conversion demo application.

10

Demo: Application

This chapter is intended to introduce a parallel demo application that can be run on a grid. A video conversion application is introduced and installation and implementation steps follow.

10.1 Video conversion application overview

The video conversion application was built to show a potentially practical use of the grid as well as to give a feel for the problems that may be encountered with data intensive applications.

The demo takes a home video tape and converts it to a VideoCD that can be played on DVD players supporting this format. Commercial programs exist for this application, but they are set up to use a single system for the conversion. The compression of the raw captured video data into an MPEG-1 or MPEG-2 data stream can take an enormous amount of time, which increases with higher quality conversions. Depending on the quality level of the video capture, the data required for a typical one hour tape can create over 10 GB of video data, which needs to be compressed to approximately 650 MB to fit on a VideoCD. The compression stage is CPU intensive, since it matches all parts of adjacent video frames looking for similar sub-pictures, and then creates an MPEG data stream encoding the frames. The MPEG data stream can be thought of as an instruction stream on how to reconstruct the video frames from prior (and sometimes future) video frames and compressed data that represent differences in portions of the images. The audio is compressed as well. This compressed data is written in a standard format to a CD-R or writable DVD disk and can then be played on DVD players or other media players supporting that format. At higher quality levels, more data is initially captured and enhanced algorithms, which consume more time, are used. The compression process can take a day or more, depending on the quality level and the speed of the system being used. For commercial DVD quality, conversions are typically done by a service company that has developed higher quality conversion algorithms. Such conversions may take weeks. Hence, grid technology is ideal for improving the process of video conversion.

Our demo uses a Linux system to capture the video from a FireWire enabled camcorder or VCR that is used to play the video tape to be converted. The captured video file is then split into a number of smaller segments. These segments are sent via Globus to Linux grid systems for compression. The compressed segments are then reassembled and a CD is written in the VCD format. This process can be seen in Figure 10-1 on page 199. Currently, the compression programs run only on Linux x86 systems. The process described here can be modified to handle more hours of video, and use different levels of quality with different media formats. However, the procedure described here has a limit of about 1 hour of video, which fits on one CD written in the VCD format.

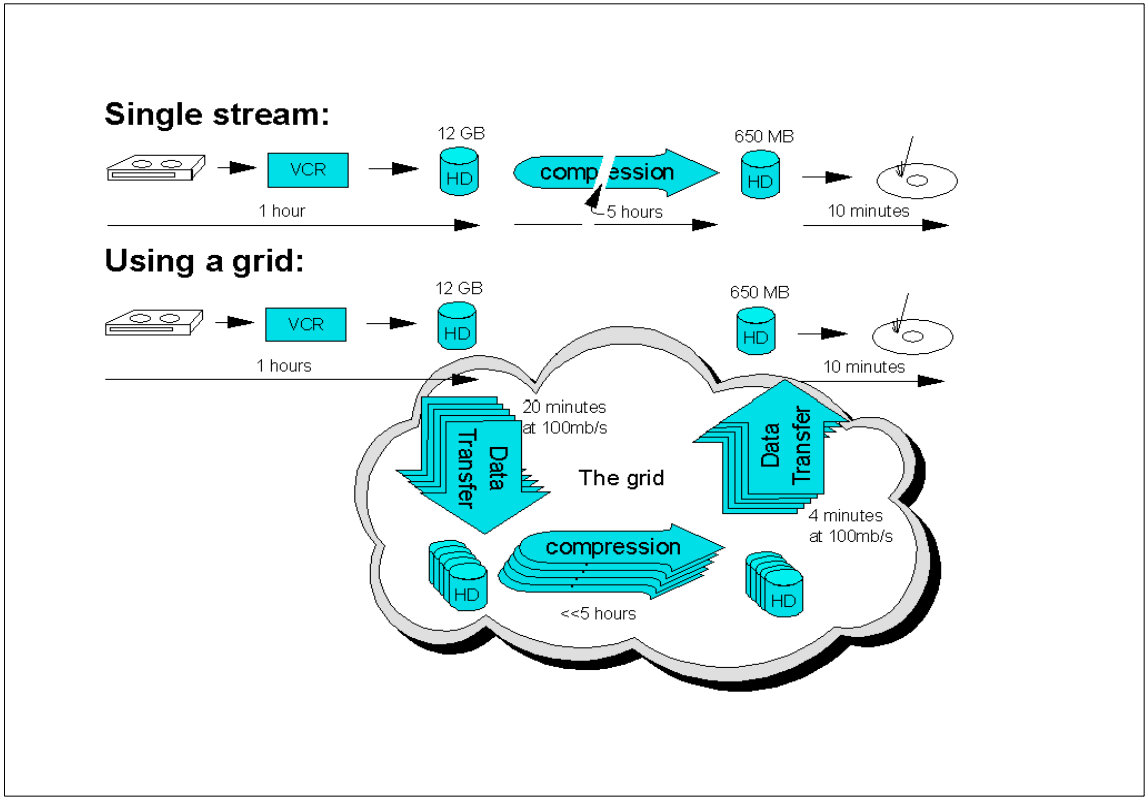

Figure 10-1 Video conversion demo application

Sending many gigabytes of data from one computer to another takes a considerable amount of time, even with a 100 Mb Ethernet connection. When a file is copied into a distributed file system, it may be stored on a system other than the one running the conversion for that segment. This can cause yet another large transfer of the data to get it to the system doing the conversion. Thus, for grid programs processing large amounts of data, it is imperative to understand the network topology and to have the means of keeping the data near the processing node that is using it.

To improve this grid application, one could install more network cards in the grid machines that split the video data. These machines would be arranged in a hierarchical network topology. With these, one could split the captured video into fewer pieces initially, and have other grid machines split the files further and distribute them to several machines in parallel. Another optimization would be to split the data while it is being captured.

10.2 Pre-installation

As described in the previous section, the video conversion application performs three major tasks: capture a video, conversion of the video, and creation of a VCD. This demo will use the Globus demo installation described in Chapter 9, “Demo: Grid setup” on page 179. Changes can be made to the scripts presented so they work on different configurations of the grid. This application was tested on Red Hat Linux 7.3 and 8.0, but should work on any distribution with suitable adjustments. One machine in this group (machine zeta) will be used to capture the video and write the VCD and the other four (alpha, beta, gamma, and delta) will be used to perform the conversion. There are several open source software packages used for this demo application. These packages and the location from which to download them are listed in Table 10-1. The versions listed were the ones that were tested, but the process should work with later versions, possibly requiring small changes in parameters. In addition to the software packages, some additional hardware will be needed. To perform the video capture, a FireWire enabled DV camcorder or FireWire VCR is needed, as well as a FireWire card for machine zeta. The demo application was tested on a FireWire card with the TI chipset, but other OHCI compatible chipsets may also work with the drivers. A CD-R drive is needed to write the VCD and must be able to write in “disk-at-once” mode. If video capture hardware is not available, one could use a AVI file captured somewhere else in DV format. Furthermore, if you do not have a CD-R drive, the MPG file created by the procedure can be played by many media players.

Table 10-1 Software packages needed for video conversion.

Package	Version	URL
libraw1394	libraw1394_0.9.0.tar.gz	http://sourceforge.net/projects/libraw1394
libdv	libdv-0.98.tar.gz	http://sourceforge.net/projects/libdv
libavc1394	libavc1394-0.3.1.tar.gz	http://sourceforge.net/projects/libavc1394
dvgrab	dvgrab-1.1b2.tar.gz	http://www.schirmacher.de/arne/dvgrab/index_e.html
mjpegtools	mjpegtools-1.6.0-1.i386.rpm	http://sourceforge.net/projects/mjpeg
vcdimager	vcdimager-0.6.2-1.i386.rpm	http://www.vcdimager.org/
cdrdao	cdrdao-1.1.7.bin.x86.linux.tar.bz2	http://cdrdao.sourceforge.net/index.html

The above files should be downloaded into the /usr/src directory on machine zeta. The mjpegtools will be installed on the server machines (alpha, beta, gamma, and delta), so they should be downloaded to the /usr/src directory on those machines.

The **md5sum** command can be used to check if you are using the same versions of the files we used by comparing the values against this list:

```
56fc0bc6f00efdebb635dcc52d91f7bc  libraw1394_0.9.0.tar.gz
9b536c093869f171de1b4179885e43fb  libdv-0.98.tar.gz
7e761f05b310392d6e07a7186b973cb5  libavc1394-0.3.1.tar.gz
b63cce79baca8baca6d9fe88a9251b1d  dvgrab-1.1b2.tar.gz
324494caa95b8ae01f0fd435db8c487f  mjpegtools-1.6.0-1.i386.rpm
8b585534df6005beee1606b3a2924663  vcdimager-0.6.2-1.i386.rpm
8c045b665f67f7c9665cb8eae3a7c814  cdrdao-1.1.7.bin.x86.linux.tar.bz2
```

10.3 Installation

There are four main parts to the installation of the video conversion demo:

- Globus Toolkit
- Capture software
- Conversion software
- VideoCD creation software

Install Globus Toolkit

Please follow the instructions given in Chapter 9, “Demo: Grid setup” on page 179 to obtain and install the Globus Toolkit in the configuration required for this demo application. This text will assume a total of five machines are used for the application. The scripts presented can be modified to work for more or fewer machines if needed.

Install capture software

There are several packages that are needed to capture a video using a FireWire card and a FireWire video source. If you are using Red Hat Linux 8.0, the following packages will not need to be installed, because they are included in the “custom” installation of “everything” as part of the Linux install. Otherwise, the following packages should be download, as listed in the previous section:

- libraw1394_0.9.0.tar.gz
- libdv-0.98.tar.gz
- libavc1394-0.3.1.tar.gz
- dvgrab-1.1b2.tar.gz

Libraw1394 is used to access the IEEE 1394 bus directly. The libdv libraries, also known as the Quasar DV codec, are used to decode video in the DV format. DV video is defined by the IEC61384 and SMPTE314M standards and is most commonly used in digital camcorders. The libavc1394 libraries are needed for

utilities to be able to control IEEE-1394 devices. The dvgrab package is used to capture DV input into AVI files. To install these packages, please follow the steps in Example 10-1. The packages must be installed as root on the machine that will be used for the video capture. This is the zeta machine, on which the FireWire card is installed.

Example 10-1 Installing video capture packages on zeta

```
cd /usr/src
tar -vzxf libraw1394_0.9.0.tar.gz
cd /libraw1394-0.9.0
./configure
make
make dev
make install

cd /usr/src
tar libdv-0.98.tar.gz
cd libdv-0.98
./configure
make
make install

cd /usr/src
tar -vzxf libavc1394-0.3.1.tar.gz
cd libavc1394-0.3.1
./configure
make
make install

cd /usr/src
tar -vzxf dvgrab-1.1b2.tar.gz
cd dvgrab-1.1b2
./configure
make
make install

chmod 666 /dev/raw1394
```

Test capture machine

A simple test can be done to determine if all of the capture software was properly installed. To do this task, connect the video camera to the FireWire card in the capture machine. dvgrab does not contain any remote control features, in other words, it cannot start and stop the camera. The camera must manually be started. Also an environment variable LD_LIBRARY_PATH must be set to include /usr/local/lib. The following example illustrates how to test the capture software. It should be performed while the video source is playing.

In Example 10-2, the **modprobe** command is used to load the ohci1394 and raw1394 modules. The **dvgrab** command is then used to capture video from the FireWire card. The autosplit and frames options indicate that the resulting capture is to be split into multiple video files, each containing 750 frames or about 25 seconds of video. The format option indicates that the captured videos should be in dv2 AVI format. If you are using Red Hat Linux 8.0, leave off the "--index small" parameter; otherwise, that version of **dvgrab** will give an error message. The index option is used to choose between the older "small" size file structure or the newer "large" file format for files larger than 1 GB. The parameter myvideo is used to indicate the name of the files created. Since multiple files are created, they will follow the naming convention myvideo001.avi, myvideo002.avi, and so on, as shown in Figure 10-2.

Example 10-2 Testing the capture software

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
modprobe ohci1394
modprobe raw1394
dvgrab --autosplit --frames 750 --format dv2 --index small myvideo
(Stop the capture using ctrl-C.)
```

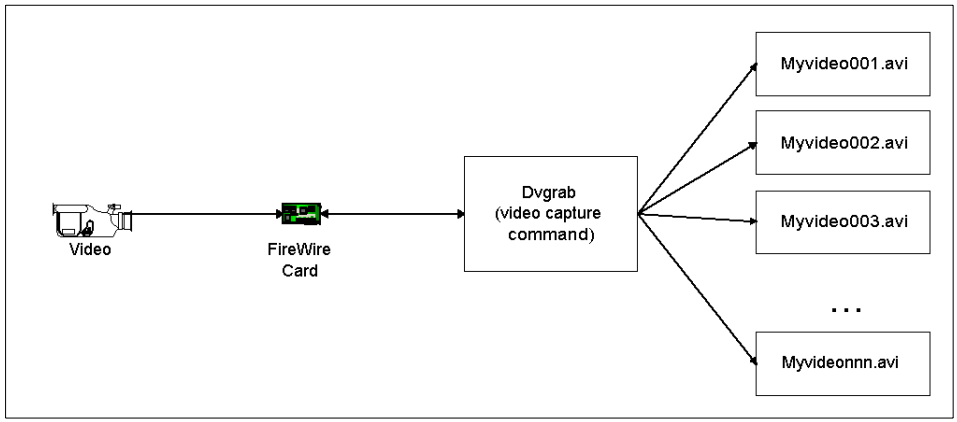

Figure 10-2 Video capture example

Install video conversion packages

There is a package that must be installed on each of the server machines to perform the video conversion. This consists of the mjpegtools-1.6.0-1.i386.rpm file, which should have been downloaded to the /usr/src/ directory of each of the server machines (alpha, beta, gamma, and delta).

Use the steps shown in Example 10-3 to install this package on each of the server machines.

Example 10-3 Video conversion software installation on alpha, beta, gamma, & delta

```
cd /usr/src
rpm -ivh --nodeps mjpegtools-1.6.0-1.i386.rpm
```

mjpegtools contains the commands necessary to do the mpeg compression. The **lav2wav** command is used to extract the audio from an captured AVI file. The **mp2enc** command is used to convert the WAV audio file produced from the **lav2wav** command into a compressed, MPEG layer-2 audio file. The **lav2yuv** command is used to convert the mjpeg-encoded video produced by dv2jpg into a raw yuv format. The **yuvscaler** command is used to generate VCD compliant frames from the video produced by **lav2yuv**. The **mpeg2enc** command is used to convert the video into MPEG format. The **mplex** command is an audio/video multiplexer used to combine the compressed audio produced by **mp2enc** with the video produced by **mpeg2enc**. The flow of commands for the video conversion is illustrated in Figure 10-3. There are two approaches that can be taken to installing the video conversion software. One way is to install the software on all of the nodes in the system that will be performing conversions. A second way could be to install the programs on a single machine and stage the binaries and library files to the remote machines for conversion. For simplicity, we will simply preinstall the mjpegtools on all of the servers.

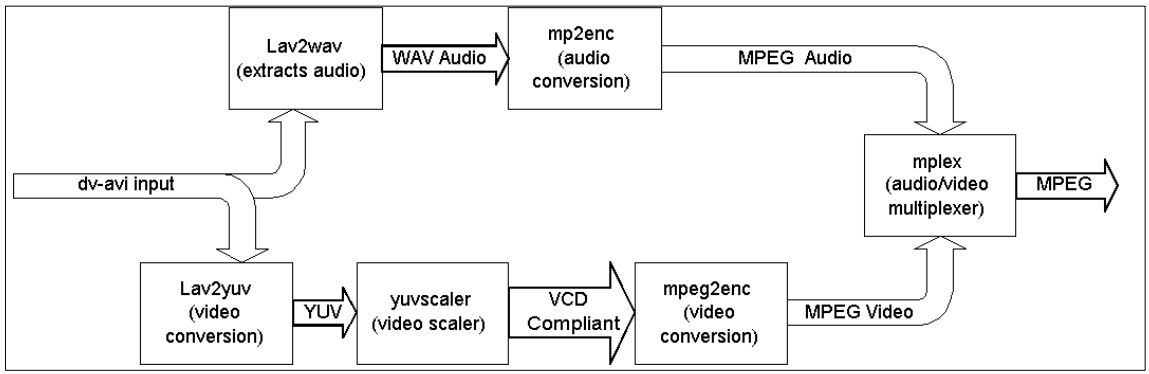

Figure 10-3 Video conversion process

Install VideoCD creation software

The final two packages are used to create the VideoCD:

- vcdimager-0.6.2-1.i386.rpm
- cdrdao-1.1.7.bin.x86.linux.tar.bz2

These packages should be downloaded from the sources described in Table 10-1 on page 200, as shown in Figure 10-4. The **vcdimager** command converts a video in MPEG format to a VCD image, in a BIN/CUE format suitable for use by the cdrdao CD-R writing program. The installation instructions for both are listed in Example 10-4. Both packages must be installed by root. These packages should be installed on the zeta machine, which has the CD-R drive.

Example 10-4 VideoCD creation software installation on zeta

```
cd /usr/src
rpm -Uvh vcdimager-0.6.2-1.i386.rpm

tar -xvjf cdrdao-1.1.7.bin.x86.linux.tar.bz2
cd cdrdao-1.1.7.bin.x86.linux
cp cdrdao /usr/bin/
# the following gives everyone permission to write CD-Rs
chmod 666 /dev/sg?
```

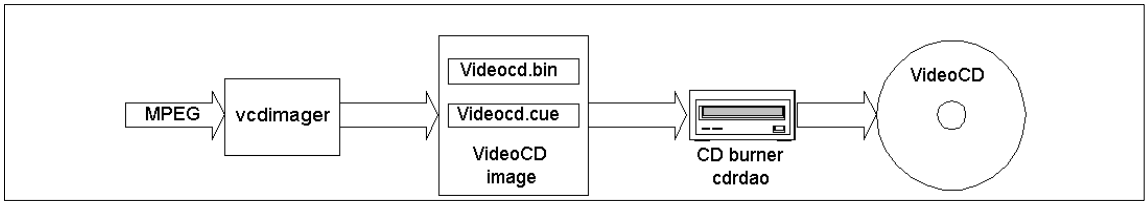

Figure 10-4 VideoCD creation using vcdimager

10.4 Setup

There are three main steps that must be completed for the video conversion demo: video capture, video conversion, and VideoCD creation. A script for each step will be presented along with a fourth script used to tie the other three together.

Video capture setup

The video capture must take place on the zeta machine with the FireWire card and the capture software installed, as shown in Example 10-5 on page 206. The **dvgrab** command will be used to capture the video. This command will continue to run until the process is killed, so it must be scripted to kill the process when the desired amount of video is captured. The **dvgrab** command also allows for capturing video in multiple files. In this example, the video will be captured in four files, each containing 750 frames. Each video should be approximately 25 seconds, for a total of almost two minutes of video.

Example 10-5 Place this file in /home/snobol/videocapture.sh on zeta

```
#!/bin/sh
#videocapture.sh

#set up capture environment
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
#the following three commands need to be performed by root:
#chmod 666 /dev/raw1394
#modprobe ohci1394
#modprobe raw1394

rm -f videocap0*.*

#start the video capture 1800 frames = 1 minute
# Note: for Red Hat 8.0 with preinstalled dvgrab, remove the "--index small"
#       parameter
dvgrab --autosplit --frames 1800 --format dv2 --index small videocap &

#wait for capture to finish
# The following lines limits the number of captured files to 4.
# It does this by killing the capture when the 5th file is created.
flag=0
while [ $flag -eq 0 ]
do
   sleep 15
   flag=`ls videocap005.avi 2>/dev/null | wc -l`
done

#kill dvgrab process
for i in `ps -efw | grep dvgrab | cut -c10-14`
do
   kill -9 $i
done

rm -r videocap005.avi
```

Video conversion setup

The video conversion script will be run on all four conversion machines. Each machine will convert a different captured avi file into MPEG format, as shown in Example 10-6. The argument passed to videoconversion.sh is the name of the avi file to be converted.

Example 10-6 Place this file in /home/snobol/videoconversion.sh on zeta

```
#!/bin/sh
videoconversion.sh
lav2wav $1 | mp2enc -V -o sound.mp2
```

```
lav2yuv $1 2>/dev/null | yuvscaler -O VCD 2>/dev/null | mpeg2enc -s -r 16 -o
video.m2v 2>/dev/null
mplex -f1 sound.mp2 video.m2v -o $1.mpg
rm -f video.m2v
rm -f sound.mp2
rm -f $1
```

VideoCD creation setup

The third script will use the **vcdimager** command to create a VCD image, as shown in Example 10-7. The input to **vcdimager** is the concatenated set of four MPEG video files and the output consists of the bin and cue files suitable for writing the VCD image to a CD-R disk. Then cdrdao will be used to write the VCD to a blank CD-R disk. The -device 0,0,0 option is the device number for the CD-R drive. If this is unknown, the following command can be used to determine the device number:

```
cdrdao -scanbus
```

Furthermore, if you are using a different type of CD-R writer, the --driver parameter may need to be changed to match your hardware. Use the --help parameter of the cdrdao program to find out more about this.

Example 10-7 Place this file in /home/snobol/videocd.sh on zeta

```
#!/bin/sh
#Either run as root or give permissions to use cd writer using:
#   chmod 666 /dev/sg?
#The --driver --device --speed parameters may need to be changed to match your
# cd writer
vcdimager videocap.mpg
cdrdao write --driver generic-mmc --device 0,0,0 --speed 8 --eject videocd.cue
rm -f videocd.cue
rm -f videocd.bin
```

Main script setup

The final script (Example 10-8) will be used to tie the other three together. The Globus toolkit will be used to stage the necessary files to the remote machines and run the conversions on the remote machines.

Example 10-8 Place this file in /home/snobol/main.sh on zeta

```
#!/bin/sh
# main.sh

#First, capture video from DVcamera
./videocapture.sh
```

```
#set environment variables
target_dir=/home/globususer
curdir=`pwd`

#stage code and video to remote machines
ndx=1
# List machines to be used here and below (one conversion per machine):
for target_host in alpha.itso.grid.com beta.itso.grid.com \
                gamma.itso.grid.com deltra.itso.grid.com
do
   echo Setting up demo on host $target_host
   globus-url-copy file:${curdir}/videoconversion.sh \
                gsiftp://${target_host}:2811${target_dir}/videoconversion.sh
   echo sending video ${curdir}/videocap00${ndx}.avi
   globus-url-copy file:${curdir}/videocap00${ndx}.avi \
                gsiftp://${target_host}:2811${target_dir}/videocap00${ndx}.avi
   globus-job-run ${target_host} /bin/chmod 755 videoconversion.sh

   echo Building RSL for $target_host
   echo +>demo_rsl${ndx}
   echo "( &(resourceManagerContact="${target_host}")" >>demo_rsl${ndx}
   echo " (subjobStartType=strict-barrier)" >> demo_rsl${ndx}
   echo " (label="videocap00${ndx}")" >> demo_rsl${ndx}
   echo " (executable= ${target_dir}/videoconversion.sh)" >> demo_rsl${ndx}
   echo " (arguments = videocap00${ndx}.avi )" >> demo_rsl${ndx}
   echo ' (stdout= $(GLOBUSRUN_GASS_URL) # "'$curdir/videocap00${ndx}.out'")' \
                >> demo_rsl${ndx}
   echo ' (stderr= $(GLOBUSRUN_GASS_URL) # "'$curdir/videocap00${ndx}.err'")' \
                >> demo_rsl${ndx}
   echo ")" >> demo_rsl${ndx}


   echo submiting job to $target_host
   globusrun -w -f demo_rsl${ndx} &
   ndx=`expr $ndx + 1`
done
   echo waiting for all conversions to complete
   wait

        echo getting result files now

rm -f videocap.mpg
ndx=1
for target_host in alpha.itso.grid.com beta.itso.grid.com gamma.itso.grid.com
deltra.itso.grid.com
do
   globus-url-copy
gsiftp://${target_host}:2811${target_dir}/videocap00${ndx}.avi.mpg \
                file:${curdir}/videocap00${ndx}.avi.mpg
```

```
   cat videocap00${ndx}.avi.mpg >> videocap.mpg
   rm -f videocap00${ndx}.*
   rm -f demo_rsl${ndx}
   ndx=`expr $ndx + 1`
done

# Now create the video cd (VCD)
./videocd.sh
```

10.5 Operation

Once all of the above steps have been completed and the necessary scripts have been written, the demo is ready to run. Before running the main.sh script, it is necessary to set up a proxy for the globususer ID to be able to run the Globus Toolkit commands. This can be done with the **grid-proxy-init** command. This command will prompt for a passphrase that was set up by the user. The DVCamera needs to be connected to the capture machine via FireWire and manually started. As soon as the video playback is started, the main.sh script should be started, which in turn starts the video capture process and all of the other conversion steps. While the process is starting, a blank CD-R disk should be loaded into the CD writer.

This assumes that all of the scripts are located in the same directory on the capture machine. This example creates four video segments, each one minute long, and sends them to each of the four sever machines for conversion. The resulting VCD can be played in DVD players that support the VCD format. The videocap.mpg file can also be played directly using a media player on a personal computer.

To summarize, the following steps are used to convert about four minutes of video and make a VCD out of it:

- Put a blank CD-R in your CD writer.
- Then, get your DV camcorder connected and ready to play.
- Issue the commands shown in Example 10-9.
- As you enter your passphrase for the **grid-info-cert** request, press Play on your camcorder.

Example 10-9 Operation commands

```
# As user ID root (once per boot):
modprobe ohci1394
modprobe raw1394
# As user ID snobol:
```

```
chmod 755 *.sh
grid-proxy-init
<passphrase>             <- This is the passphrase you set for snobol: kjsdnb
./main.sh
```

10.6 Improvements

As mentioned in the overview, there are some changes that can be made to improve the performance of the demo application. One of these changes is to begin the video file transfer during the capture, as illustrated by Figure 10-5 on page 211. Since **dvgrab** creates multiple video files while capturing the video, this improvement can be easily scripted. Once the first video is completed, it can be staged to the remote machine and the conversion can begin. This process is repeated as each individual AVI file is captured. A parameter is specified in the new script, which states the number of minutes of video to capture and convert. Furthermore, using MDS, we can locate a machine on the grid that is not very busy and send the video file to it. In this way, we can use a variable number of machines.

To accomplish this, we can use the following script files in place of the ones above. Also, these are configured for using anonymous MDS **grid-info-search** commands. For a secure MDS, remove the "-x" parameter form the **grid-info-search** commands.

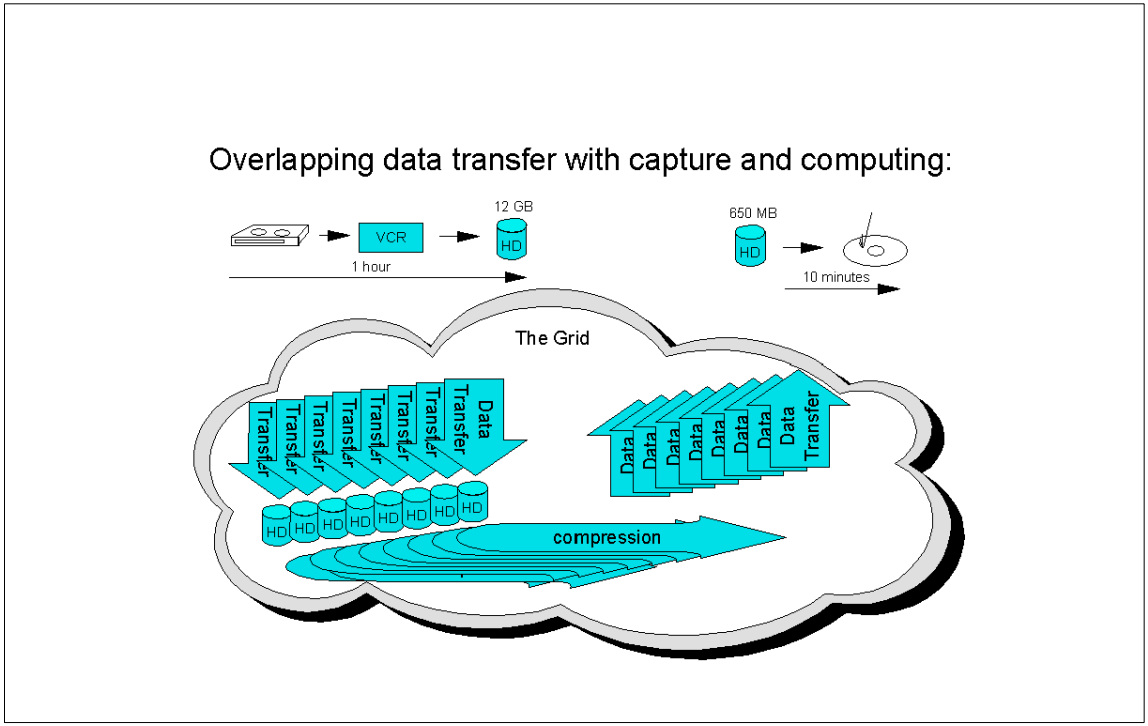

Figure 10-5 Parallel video capture and date transfer

Example 10-10 through Example 10-13 on page 213 are the improved scripts to use for the enhanced demo. A parameter is given to the main.sh script, which says how many minutes you want to record. Since a VCD can only hold about an hour, this number should be 60 or less.

Example 10-10 Place this file in /home/snobol/videocapture.sh on machine zeta

```
#!/bin/sh
#videocapture.sh <maxfilenumberwanted>

#set up capture environment
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
#the following three commands need to be performed by root in advance:
#chmod 666 /dev/raw1394
#modprobe ohci1394
#modprobe raw1394

rm -f videocap0*.*

#start the video capture in the background. 1800 frames = 1 minute
# Note: for RedHat 8.0 with preinstalled dvgrab,
```

```
#       remove the "--index small" parameter
dvgrab --autosplit --frames 1800 --format dv2 --index small videocap &

ndx=`expr $1 + 1`
ndx=`echo 000$ndx | rev | cut -c 1-3 | rev`

#wait for capture to finish
# It does this by killing the capture when the ${ndx}th file is created.
flag=0
while [ $flag -eq 0 ]
do
   sleep 30
   flag=`ls videocap${ndx}.avi 2>/dev/null | wc -l`
done

#kill dvgrab process
for i in `ps -efw | grep dvgrab | cut -c10-14`
do
   kill -9 $i
done

#dont delete the extra file so main.sh can tell they are all captured
```

Example 10-11 Place this file in /home/snobol/videoconversion.sh on zeta

```
#!/bin/sh
#video_conversion.sh  <inputfile.avi> <uniquenumber>
lav2wav $1 | mp2enc -V -o sound${2}.mp2
lav2yuv $1 2>/dev/null | yuvscaler -O VCD 2>/dev/null | mpeg2enc \
                -s -r 16 -o video${2}.m2v 2>/dev/null
mplex -f1 sound${2}.mp2 video${2}.m2v -o $1.mpg
rm -f sound${2}.mp2
rm -f video${2}.m2v
rm -f $1
```

Example 10-12 Place this file in /home/snobol/videocd.sh on zeta

```
#!/bin/sh
#Either run as root or give permissions to use cd writer using:
#   chmod 666 /dev/sg?
#The --driver --device --speed parameters may need to be changed to match your
# cd writer
vcdimager videocap.mpg
cdrdao write --driver generic-mmc --device 0,0,0 --speed 8 --eject videocd.cue
rm -f videocd.cue
rm -f videocd.bin
```

Example 10-13 Place this file in /home/snobol/main.sh on zeta

```
#!/bin/sh
#main.sh <number of minutes to capture>
# Note: a normal VCD will not hold more than 60 minutes

# Make certain certificate is not expired
echo type your user passphrase, if asked by grid-proxy-init
grid-proxy-init

#capture video from DVcamera
#Improvements "Run capture in the background so we can begin converting pieces"
./videocapture.sh $1 &

#set environment variables
target_dir=/home/globususer
curdir=`pwd`

#stage code and video to remote machines

rm -f resources_used.txt
#kdx is the section number being handled
#ndx is equal kdx + 1
kdx=001
ndx=002
while [ $kdx -lt `expr $1 + 1` ]
do
   #wait for capture of current section $kdx to finish
   flag=0
   while [ $flag -eq 0 ]
   do
                flag=`ls videocap${ndx}.avi 2>/dev/null | wc -l`
                if [ $flag -eq 0 ]
                  then sleep 13
                fi
   done

   #Use MDS to find a machine which is more than 95% free cpu time
   #check to see if we got any responses
   available=0
   while [ $available -eq 0 ]
   do
      grid-info-search -x \
         "(&(Mds-Device-Group-name=processors)(Mds-Cpu-Free-1minX100>=95))" \
         Mds-Cpu-Free-1minX100 | grep Mds-Host-hn | cut -d , -f 2 | cut -d = \
         -f 2 > tmp.out
      echo waiting for resource more sufficiently free
                available=`cat tmp.out | wc -l`
                if [ $available -eq 0 ]
```

```
                    then sleep 29
                fi
    done

    target_host=`head -1 tmp.out`
         echo Video section $kdx will be sent to $target_host
    echo $target_host >> resources_used.txt
         # send the script to run at the target host
         globus-url-copy file://${curdir}/videoconversion.sh \
              gsiftp://${target_host}:2811${target_dir}/videoconversion.sh
    # send the video section to the target host
    globus-url-copy file://${curdir}/videocap${kdx}.avi \
              gsiftp://${target_host}:2811${target_dir}/videocap${kdx}.avi
         # get rid of local copy of captured video
         rm -f videocap${kdx}.avi
    # make the script executable
    globus-job-run $target_host /bin/chmod 755 videoconversion.sh

    echo submiting job to $target_host
    globus-job-run $target_host $target_dir/videoconversion.sh \
           videocap${kdx}.avi ${kdx} &

         ndx=`expr $ndx + 1`
         kdx=`expr $kdx + 1`
         # now left pad the numbers with zeros
    kdx=`echo 000$kdx | rev | cut -c 1-3 | rev`
    ndx=`echo 000$ndx | rev | cut -c 1-3 | rev`
done

# delete extra capture file now
rm -r videocap$kdx.avi

echo waiting for all conversions to complete
wait
rm -f videocap.mpg
kdx=001
for target_host in `cat resources_used.txt`
do
    globus-url-copy \
       gsiftp://${target_host}:2811${target_dir}/videocap${kdx}.avi.mpg \
       file://${curdir}/videocap${kdx}.avi.mpg
    cat videocap${kdx}.avi.mpg >> videocap.mpg
         # delete local file
         rm -f videocap${kdx}.avi.mpg
         # delete remote files
         globus-job-run $target_host /bin/rm -f videocap${kdx}.avi.mpg
         globus-job-run $target_host /bin/rm -f videoconversion.sh

         kdx=`expr $kdx + 1`
```

```
        kdx=`echo 000$kdx | rev | cut -c 1-3 | rev`
done

# delete some other temp files
rm -f tmp.out
# this file showsshould which machines were chosen.  rm -f resources_used.txt

# Now create the video cd (VCD)
./videocd.sh
```

The following steps are used to convert 15 minutes of video and make a VCD out of it. Put a blank CD-R in your CD writer. Then get your DV camcorder ready to play, issue the commands below (Example 10-14), and, as you enter your passphrase for the **grid-info-cert** request, press Play on your camcorder.

Example 10-14 Commands to start the video conversion on zeta

```
# As user root (once per boot):
modprobe ohci1394
modprobe raw1394
# As user snobol:
chmod 755 *.sh
./main.sh 15
<passphrase>             <- you probably set it to the snobol password: kjsdnb
```

Other enhancements will become apparent as you use these scripts. The scripts could be enhanced to automatically generate a second VCD when the video is longer than 60 minutes. Additional MDS queries could be made to locate only x86 Linux machines that have the mjpegtools installed, so that these procedures could be used in a larger grid. In fact, it may be possible to set up a small grid in a neighborhood of homes connected via cable modems or other means, permitting all of the participants to use each other's machines to speed up conversion of their old home video tapes to disks. These improvements are left as an exercise to the reader.

Part 5

Implementations

11

Grid examples

In this chapter, we bring five examples of grid computing implementations. The implementations of the examples try to rationalize problem solutions. We provide the examples in the hope that they will be used as a guide and as a scale to use grid computing to solve real life and business problems successfully.

11.1 Five examples

As stated earlier, grid computing is the utilization of computers' available resources across networks. Grid computing technology will assist organizations in gaining immense computing power by utilizing their resources more efficiently. Additionally, grid computing is an advancement in technology using existing assets and technologies.

A typical IT structure consists of a set of business solutions that, combined, create computing problems. This set of problems consists of performance, storage, and memory shortage problems. Grid computing technology solves these problems by distributing tasks to multiple concurrent suitable resources instead of performing them sequentially on a single resource. Grid technology will normalize the peaks and bottlenecks by creating immense computing power and throughput to handle the same workload in much less time.

In the ZetaGrid example, we demonstrate how calculations that are estimated to take 213 years are calculated in less than one year. In the cancer imaging example, we show how a grid is being used in medical research and diagnoses. In the Online Gaming example, we present an innovative, commercial venture. In the Excel example, we demonstrate a simple financial model that can be implemented in a small office or department.

Even though we believe that grid technology could be implemented in all organizations with network infrastructure in place, organizations with the following profile are the best candidates for grid technology:

- Applications take too long to finish.
- Have many servers and looking to consolidate.
- CPU intensive applications.
- Storage intensive applications.
- Organization is geographically distributed.

11.2 Digital cancer imaging

This is an example of grid computing used in the area of digital cancer imaging. It illustrates how data resource sharing can be implemented in a grid to aid in research and potentially save lives.

11.2.1 Needs

A need has developed for a medical records system to capture, manage, and store patient files from any location for fast retrieval and diagnostic evaluations. These files include patient records, clinical history, and medical images, such as tomography (CT or cat scan), magnetic resonance imagery (MRI), ultra sounds, and mammograms. The main benefit for such a system is number of lives saved by early diagnosis and treatment.

In some hospitals, files have become mis-filed or otherwise make unavailable, which has left physicians and radiologists without comparative records for making diagnoses. Patients also sometimes obtained their health care services at other medical facilities, making it difficult to quickly retrieve those records to study the medical history and prior treatments. For the patient, there is the potential for complications leading to disabilities or death. For the hospital or medical practitioner, there is the possibility of litigation for misdiagnosis due to the unavailability of patients' records. As a result of missing records, additional, often costly, diagnostic studies are required.

Rising health care costs are due to the high overhead and administrative costs of maintaining paper and x-ray film file systems. Each year, the average hospital spends $4 million to develop x-ray films, according to some estimates. Millions of dollars can be saved each year by using the medical records grid.

A unique property exists for one form of cancer diagnosis in that a set of standards and protocols exist for mammography imagery. Traditionally, radiologists will scan the films with a magnifying glass to look for micro calcifications. There are then three options:

- Wait six months to see if the mass or calcifications increase, which suggests malignancy
- Biopsy
- Biopsy and removal

It is very important that prior x-rays be taken into account when deciding on an option. To help, x-ray images can be converted to a digitized format and analytical tools used to help diagnose individual patients and quickly isolate abnormalities.

11.2.2 Solution

As illustrated in Figure 11-1, this examples demonstrates the use of a data grid disperses among four hospitals. The grid takes into account the following challenges:

- Data images of 160 MB per mammogram exam results in more than 5.6 PB per year.
- Daily traffic for archiving current exams and comparing past and present exams is a minimum of 28 TB.
- Network bandwidth responses range from high speed access to expert consultation and unscheduled exams down to low speed Internet access.

At each hospital, physicians, radiologists, and other authorized personnel upload images to a data repository that is accessible by the other hospitals. The medical personnel are able to query and retrieve patient records within 90 seconds.

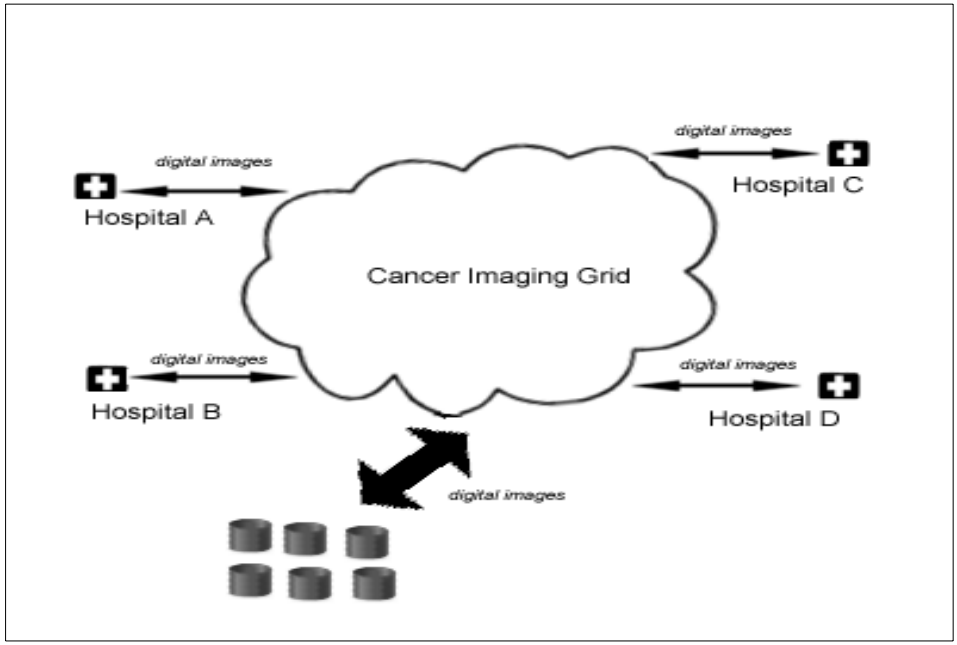

Figure 11-1 Conceptual view of the cancer imaging grid

Data grid

This is a typical example of a data grid, where the application accesses a storage system that is transparent to the hospital. There is also a metadata access service to access and manage information about data stored in the storage system.

The grid has the following characteristics, as illustrated in Figure 11-2:

- Images are loaded at each hospital or end node. There are two servers. One acts as a temporary repository. The other is a link to the Internet or to the next generation of the Internet.
- Each hospital transmits images to a metropolitan hub.
- The metropolitan hubs funnels its images to a high-capacity regional hub for resource pooling.
- The distributed archive emulates one huge archive. Queries to the archive are handled rapidly by a secure, highly available database that indexes and catalogs the data.
- Access at local hospitals is transparent and fast.
- Management software monitors and controls the nodes and provides security and diagnostic information.

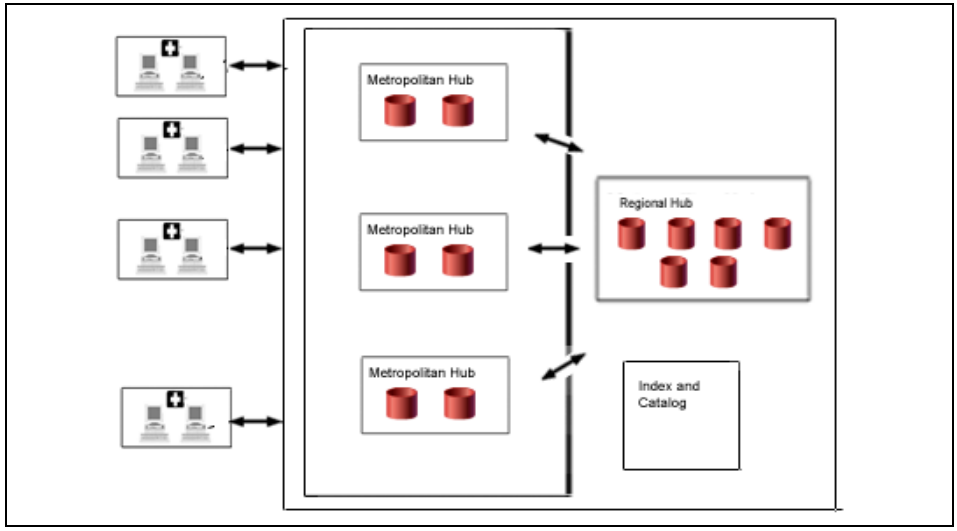

Figure 11-2 Cancer imaging grid topology

Application

Digital images of potential cancers are being shared among four hospitals. The potential exists to scale this grid to include thousands of hospitals and clinics and to share more types of patient data not only for aiding diagnoses but also for research and training. Using the grid, algorithms can uncover patterns to identify abnormal concentrations of cancer in the population. A suite of educational tools will be deployed on the grid to help doctors, medical students, and interns learn more about cancer and related diseases.

11.3 Spreadsheet

This example demonstrates how combining personal computers and applications connected on a local area network with grid computing technology could deliver superior computing power.

11.3.1 Needs

Organizations around the world invested for several years in personal computers and software. Today, we come to find that an average PC is underutilized in all aspects. Most personal computers are idle or used for short amounts of time. Research shows that thousands of computers around the world are used as e-mail clients and Web browsers with small fragmented activities. Personal computers are necessary in all organizations to achieve little tasks, and it is cost effective to buy the latest CPU technology and more memory because of the developing technology and applications requirements. Today, personal computer applications are limited to the size of workspace and processing they can do in a reasonable amount of time. Using grid computing technology, PC users can develop applications that could well exceed the reasonable power of one personal computer. In addition, new applications could be developed because of the developing grid technology and because of the growing processors' power.

11.3.2 Solution

Today, spreadsheet applications are limited to tens of thousands of rows and less than a few hundreds of columns. An application that requires ten times that much space maybe broken into ten or more multiple spreadsheets and distributed to ten or more personal computers utilizing grid technology. Each part of the spreadsheet will calculate on its own node and report results back to the master node for final reporting. In the past, organizations bought expensive computers to establish such tasks.

Computational grid

Redundancy and multiple job submission are native in computational grids. Application integrity can be achieved by submitting the same tasks to more than one node, then results can be compared for accuracy. Also, jobs could be resubmitted if a node does not return result in a reasonable amount of time, which may indicate a node became unavailable after the task was submitted. Applications that requires calculation of a great quantity multiple spreadsheets can take advantage of using multiple nodes performing the calculation.
Figure 11-3 on page 225 illustrates the difference in calculation speed between one PC vs. seven grid nodes. The problem (in this example, it is linear) is to calculate 55 selected cells; the calculation of the 55 cells took about 330 seconds to complete on a single PC vs. 100 seconds on the grid.

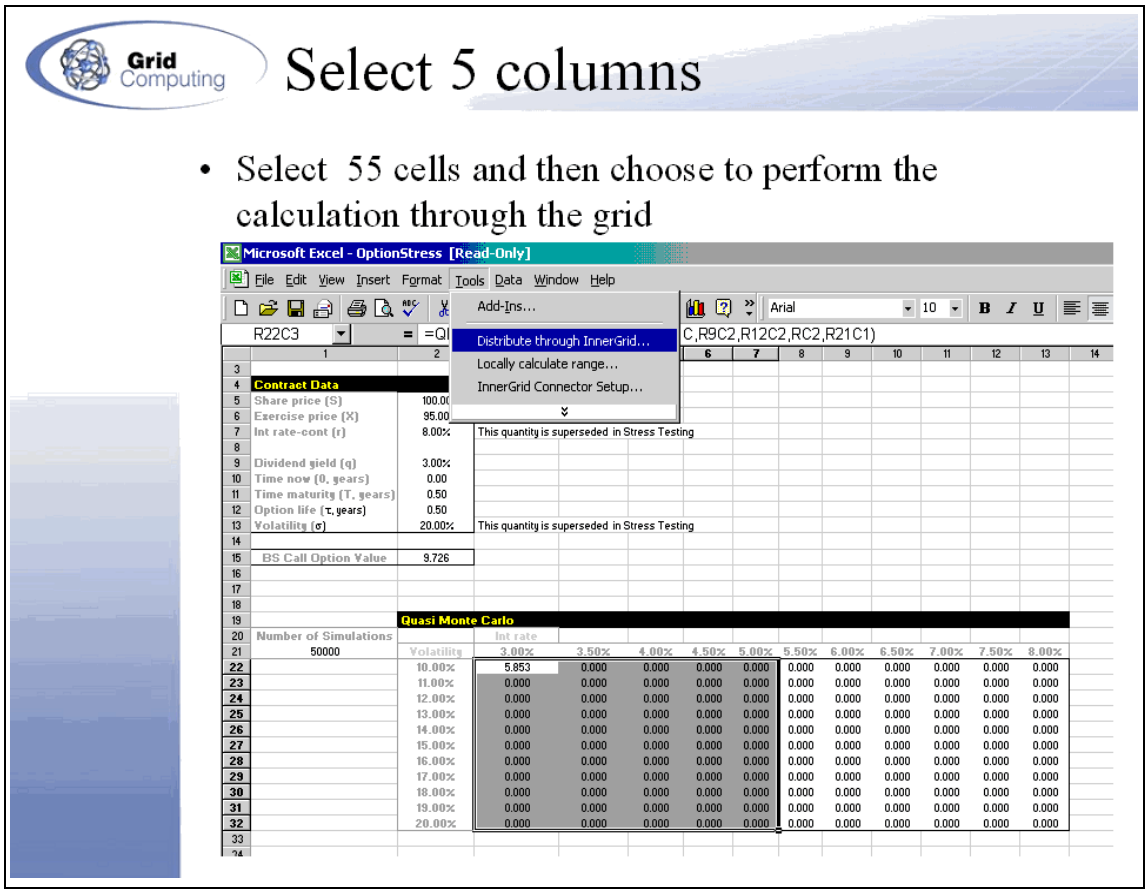

Figure 11-3 One PC vs. seven grid nodes

Application

A financial institute could use any licensed spreadsheet application that is already installed on their local area network stations to compute complex tasks. When a workstations is idle, it can be utilized to carry out the queued small tasks to assist completing a bigger task. This can be done around the clock, during and after business hours, as the distributed work runs in the background in a lower priority unless the computer is entirely free and available.

11.4 ZetaGrid

ZetaGrid is a computational grid that run on participating computers around the world and across the Internet. It was developed in an IBM lab in Germany in

2000 and has been functioning flawlessly. The participants download the client software that is suitable for their operating system, install it, and run it. When the screen saver on the client computer is activated, the ZetaGrid client initiates a request to the grid for a new task. The client's processor then is utilized until the task is completed. Then the client return results to the ZetaGrid servers and acquires a new task, this loop of operation continues until the client's screen saver is interrupted.

11.4.1 Needs

ZetaGrid is worldwide heterogeneous computational grid that tries to verify the Riemann hypothesis, by calculating all the zeroes of the Riemann zeta function.

Additional information about the Riemann zeta function is available from the ZetaGrid Web site at:

http://www.zetagrid.net

The verification of Riemann's hypothesis, formulated in 1859, is considered to be one of modern mathematics most important problems. The last 140 years did not bring its proof, but a considerable number of important mathematical theorems that depend on the hypothesis being true, for example, the fastest known primarily test of Miller.

11.4.2 Solution

This implementation, as illustrated in Figure 11-4 on page 227, involves more than 3000 heterogeneous workstations and has a peak performance rate of about 530 GFLOPS. About 2 billions of zeros for the zeta function are calculated every day.

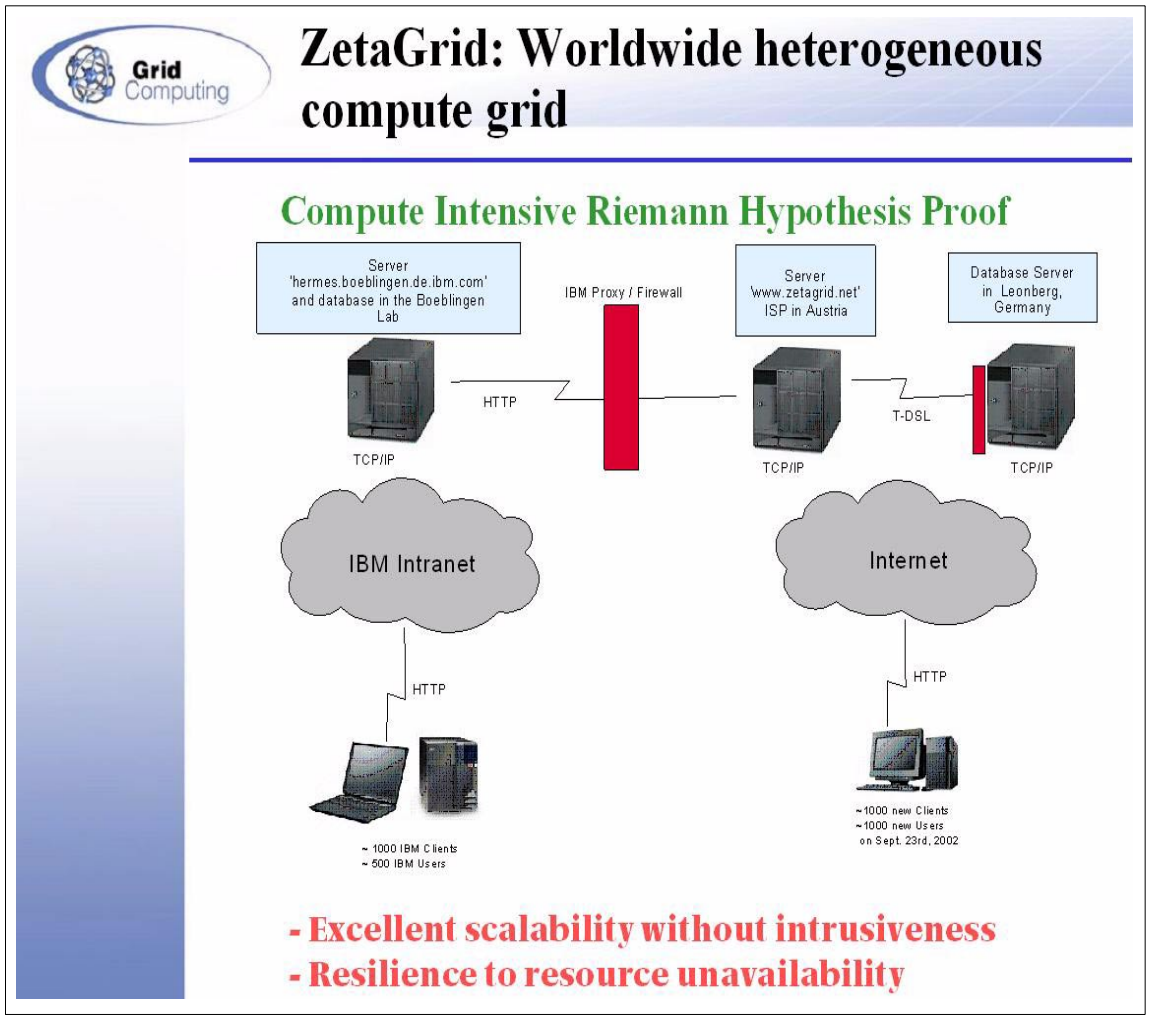

Figure 11-4 ZetaGrid architecture

Scavenging

CPU scavenging grids will be growing in the research field. Resources schedule themselves in and out of the grid, their availability is variable, and there is no fixed expectation on time to return results. The task may be interrupted unexpectedly by participants and resources maybe powered down randomly. This type of grid is mostly used in life science applications that may not have a requirement to finish in a certain timetable.

In this ZetaGrid example, CPU cycles are donated from Internet users that are interested in proofing the theory while other users are doing it for rewards. When

the screen saver on the client computer is activated, indicating that the system is idle, then the ZetaGrid client initiates, as illustrated in Figure 11-5.

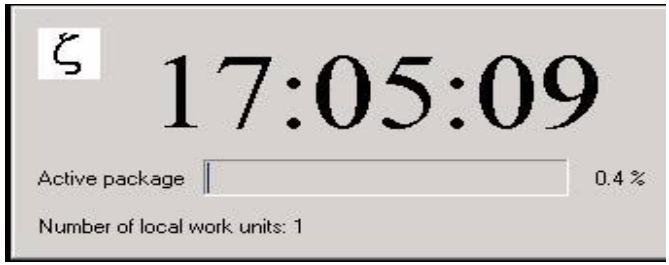

Figure 11-5 ZetaGrid client as a screen saver on Windows-based computers

Application

The ZetaGrid software is open source and platform independent. The source can be used for any CPU intensive application that can be split into separate sub-processes. ZetaGrid client support various platforms like AIX, Linux, and Microsoft Windows. The source can be downloaded at the following Web site:

http://www.zetagrid.net/zeta/downloads.html

Multiple platforms

Heterogeneity is necessary in this type of grid, not only on an operating system level but also on the hardware architecture level. This is to increase the number of participants and decrease the amount of time to accomplish results.
Table 11-1 presents which operating systems participates today (November 2002) in ZetaGrid.

Table 11-1 Operating systems for ZetaGrid

Operating system	Processor	
AIX	ppc	230
Linux	i386	427
Linux	s390	3
Linux	x86	99
Mac OS X	ppc	1
SunOS	sparc	12
Windows 2000	x86	1448
Windows 95	x86	15

Operating system	Processor	
Windows 98	x86	150
Windows Me	x86	47
Windows NT	x86	346
Windows XP	x86	462

11.5 Simulation

In the aerospace and the automotive industries, many manufacturers run simulations. Simulation is not a replacement for real experience, but it gives a great deal of information before sending the product for testing; this not only economical but also logical.

11.5.1 Needs

Simulation and visualization of a simulation requires high computing power. Industries are using grid computing technology to perform their simulations and calculate design problems that were impossible before due to the cost of processing. Calculations that took months are done in days in a grid environment.

11.5.2 Solution

Applications for design and simulation are known to demand high computing power when it comes to advanced design and testing phases. Computer cycles that are necessary to drive simulation in high resolution mode will far exceed the abilities of one single computer, regardless of it how many processors a single computer may have installed.

In the automotive industry, grid computing technology has assisted many car manufacturers using different simulation and design application to achieve compute times and build scenarios that were time consuming to build before. This gave car manufacturing companies the advantage of building better products and simulation problems before they are distributed, unknowingly, to the consumer.

Grid computing and application

Grid computing technology and compute-intensive applications, such as finite element analysis, computational fluid dynamics, and DOE (design-of-experiment) simulations, gave the automotive industry the advantage

of producing efficient and performing products in a way that would take greater time and expense to achieve. Grid allows you to perform thousands more "what-if" analyses up-front, so they could increase quality and see how minute changes in the third-gear ratio impact a car's overall performance. Also, applications like MCAD (Mechanical Computer Aided Design) demand that kind of CPU power and can take advantage of a computational grid.

Web interface

Using the same custom-built Web interface that front-ended their previous cluster software, engineers can now send their jobs to the grid, which automatically distributes those jobs to the machines that are available at that moment.

In the automotive industry, we can assume a computational grid configured with hundreds computers, as seen in Figure 11-6. Imagine the power of all those computers supplying a pool of available resources to provide computing on demand.

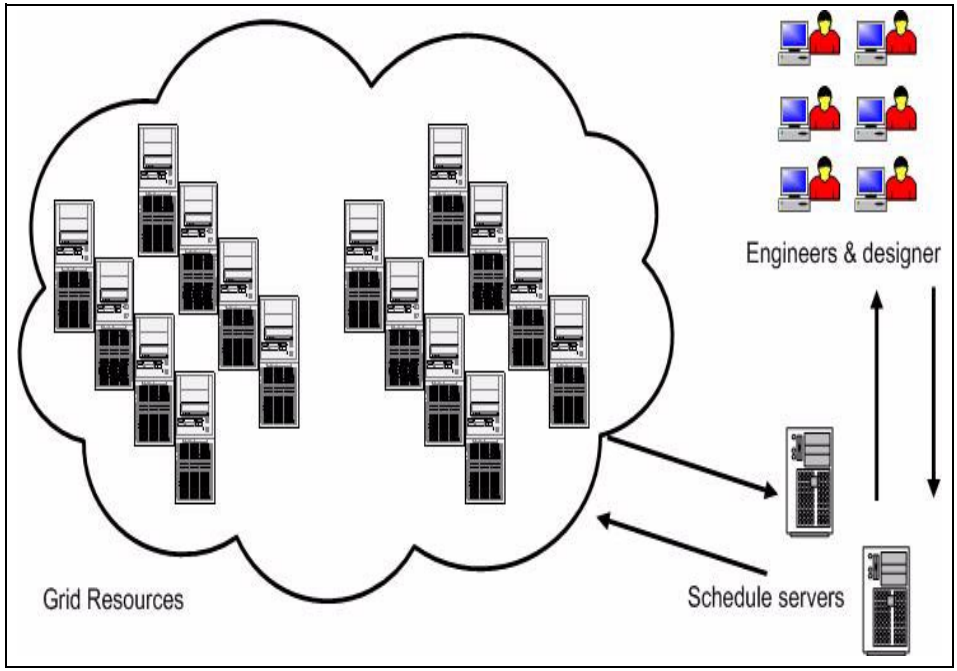

Figure 11-6 Power computing on demand

11.6 Online gaming

This example demonstrates how grid computing can be used in the entertainment industry, such as for online gaming. It illustrates how processing resources can be used on an as-needed basis as the demand for power grows.

11.6.1 Needs

The popularity of multi player online games has grown as users buy subscriptions to play online games with other users from around the world. The challenge with traditional multi player games is the limitation placed on the number of concurrent players per server. The following scenario occurs:

- A player buys a game and a subscription to play online with others.
- There are thousands other players who buy the same game and want to play online.
- The server handling the game is capable of handling a limited number of players.

If the power capacity is overcome, the solution has been to acquire additional servers and run additional copies of the gaming engine, but this must be done considering the application capability, that is, the capability of the application runs in a parallel environment.

11.6.2 Solution

Online gaming is an example of a computational grid that utilizes resource aggregation on demand from third parties. Instead of investing a considerable amount in new hardware, the gaming company can take the approach of leasing computing power from service providers, such as IBM, who have data centers of servers with connectivity to the Internet.

Starting with the limitations on the number of simultaneous users per server, the grid overcomes this by providing resources on demand. Instead of a single server hosting a number of players, the grid itself becomes the host. The challenge is to design gaming software to take advantage of this feature of one large virtual gaming world. As demand for a particular game increases, an additional server can be integrated into the online gaming grid to handle the processing requests.

The nature of the grid also provides for transferring resources on demand from one node to another node. It is transparent which server on the grid is responding. When a server needs maintenance, upgrades, or replacement, the

resources it was handling are taken over by another server, This has the effect that a game player has no downtime.

Application considerations

Considering that the application has parallel characteristics and low-intensive communication, a computational grid is perfectly suitable to host the solution.

Expanding this scenario for other type of video games leads to other considerations:

- More servers are needed to support more games, but what if a couple of the games are not popular? As the servers can be allocated on demand, this allows a better usage distribution.
- What if a game's popularity grows rapidly? How soon can another server be added? On grid, a server can be added transparently, without interruptions
- During a maintenance or upgrade to a server, game play for the users on that server does not need to stop, since the servers can fade in and out when necessary.

With these considerations, there is a motivation for gaming companies to come up with a cost-effective infrastructure where servers can be added or replaced without interrupting users as they play in a true multi-player environment.

Use of data centers

As illustrated in Figure 11-7 on page 233, the online gaming grid can use a network of data centers, each of which contains multiple servers. The game developers access the grid to load and test their games, objects, rules, and parameters. Once tested, game specific configuration information is transferred to the data center or grid service providers. The end user or game player purchases a gaming client and the fun begins. As the demand changes for the game play, the resources are shifted to handle the load.

Just as most companies do not generate their own electricity, there are some companies who do not own their computing infrastructure. Online gaming is an example of thinking of a grid as a utility. Companies cannot afford to gamble on investing in servers, networks, and storage. Grid service providers supply a growing need for computing power without the headaches of hardware maintenance and administration.

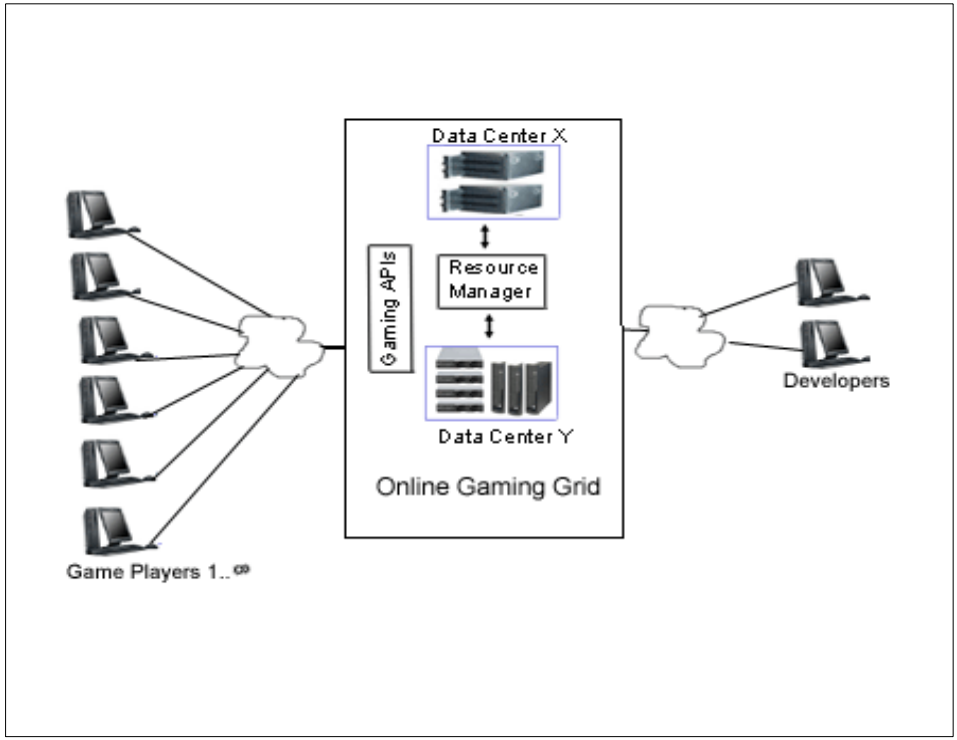

Figure 11-7 Online gaming grid

12

Available demos

This chapter provides descriptions of some available demonstrations created by IBM to supply practical grid knowledge to our customer and support marketing, sales, and technical sales enablement.

All presented demonstrations were available by the time this redbook was written, most of them at the IBM Design Center for e-business on demand in Montpelier, France. Also, when available, we provide Web site links to those demonstrations.

For more information on how to access these and other available demonstrations, contact your local IBM representative.

12.1 Excel spreadsheet

As explained in 11.3, “Spreadsheet” on page 224, the Excel demonstration shows how easy it is to integrate any application in a grid and the performance gain that can be achieved. The purpose of the spreadsheet is to perform a financial risk assessment using Monte Carlo simulations, well known to people working in the financial sector, as seen in Figure 12-1. With a larger number of iterations, the results will be more precise and the calculation longer. The demonstration is a distributed spreadsheet computation. The purpose is to distribute the calculation through the grid and then calculate the performance increase achieved at the end of the calculation.

3										
4	Contract Data									
5	Share Price (S)	100.00								
6	Exercise price (X)	95.00								
7	Int rate-cont ®	8.00%								
8										
9	Dividend yield (q)	3.00%								
10	Time now	0.00								
11	Time maturity	0.50								
12	Option life	0.50								
13	Volatility	20.00%								
14										
15	BS Call Option Value	9.726								
16										
17										
18										
19		Quasi Monte Carlo								
20	Number of Simulations		Int Rate							
21	50000	Volatility	3.00%	3.50%	4.00%	4.50%	5.00%	5.50%	6.00%	6.50%
22		10.00%	5.853	0.000	0.000	0.000	0.000	0.000	0.000	0.000
23		11.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24		12.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
25		13.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
26		14.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
27		15.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
28		16.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
29		17.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
30		18.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
31		19.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
32		20.00%	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Figure 12-1 Monte Carlo simulation

As shown in Figure 12-2 on page 237, the calculation can be monitored. It is possible to see the active nodes and the tasks that are being performed.

Name	Version	Group	Task	Microtask	Action
GRIDNODE15_IGClient1_0	2.2	maya	OptionStress.xls_9.100.195.87_500102187	4	Details View logs Remove Reset
INJECTOR1_igclient_0	2.2	maya	OptionStress.xls_9.100.195.87_500102187	2	Details View logs Remove Reset
INJECTOR2_igclient_0	2.2	maya	OptionStress.xls_9.100.195.87_500102187	6	Details View logs Remove Reset
INJECTOR3_igclient_0	2.2	maya	OptionStress.xls_9.100.195.87_500102187	3	Details View logs Remove Reset
INJECTOR4_igclient_0	2.2	maya	OptionStress.xls_9.100.195.87_500102187	5	Details View logs Remove Reset

Figure 12-2 Monitoring active nodes and tasks

12.2 ZetaGrid

The ZetaGrid is a worldwide heterogeneous computation grid that tries to verify the Riemann hypothesis, as explained in 11.4, “ZetaGrid” on page 225. However, this demonstration shows that other computational intensive applications could be deployed using the same technology. In this instance, IBM Global Services designed, implemented, and deployed the ZetaGrid and Java clients.

The ZetaGrid demonstration shows how scalable and heterogeneous a grid infrastructure can be. In this instance, clients are available for Linux, AIX, Windows, and Solaris. It also shows the performance gain you can achieve using grid technology over the Internet. Computation on the client is unintrusive, as it is active only while a screen saver is enabled. The Zeta software is a customized client in the form of a screen saver or a command line version. It is active only during the idle time of the computer and can be deactivated at any time without

delay. Easy to configure, as illustrated in Figure 12-3, connectivity to the server via HTTP is only required to download work units and upload results.

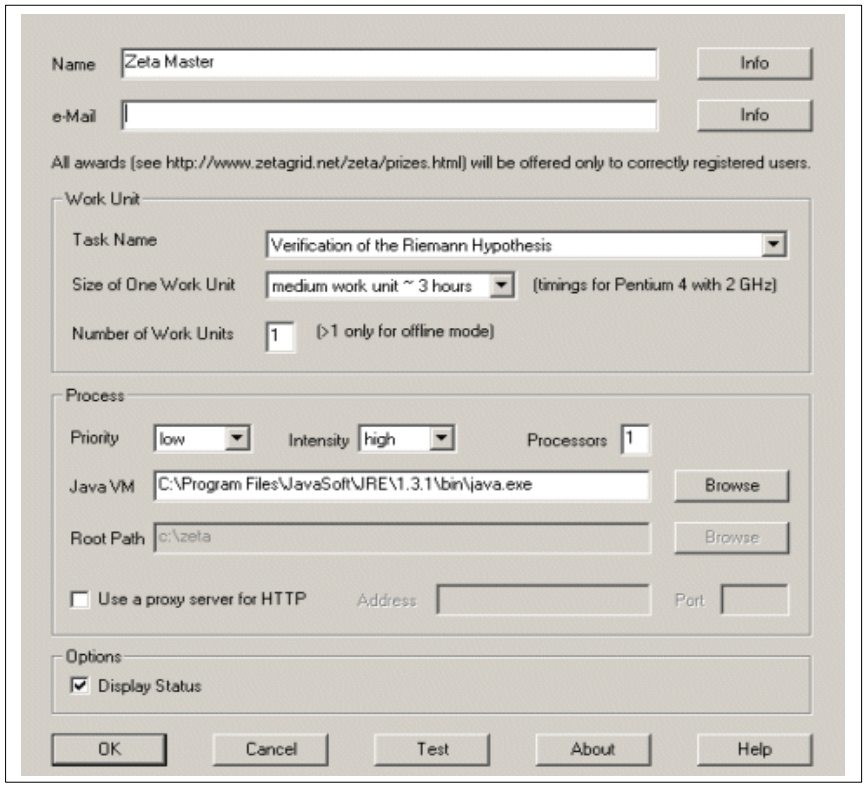

Figure 12-3 Configuring the Zeta client

All file transfers use a secure mechanism to protect privacy and prevent tampering with the data.

For more information and to download the Zeta client, see:

http://www.zetagrid.net

12.3 RNA folding

The RNA (ribonucleic acid) folding demonstration is a distributed computation on a heterogeneous grid. It shows good integration through an application server and how multiple sites can be accessed for additional power.

Proteins and other biological macromolecules assume specific higher order tertiary structures that are prerequisites for their function. The common

assumption is the RNA folds from a primary to a secondary structure, which then folds to a 3-dimensional tertiary structure. This demonstration shows the effect of temperature on different structures and produces a representation of the new molecule, as seen in Figure 12-4.

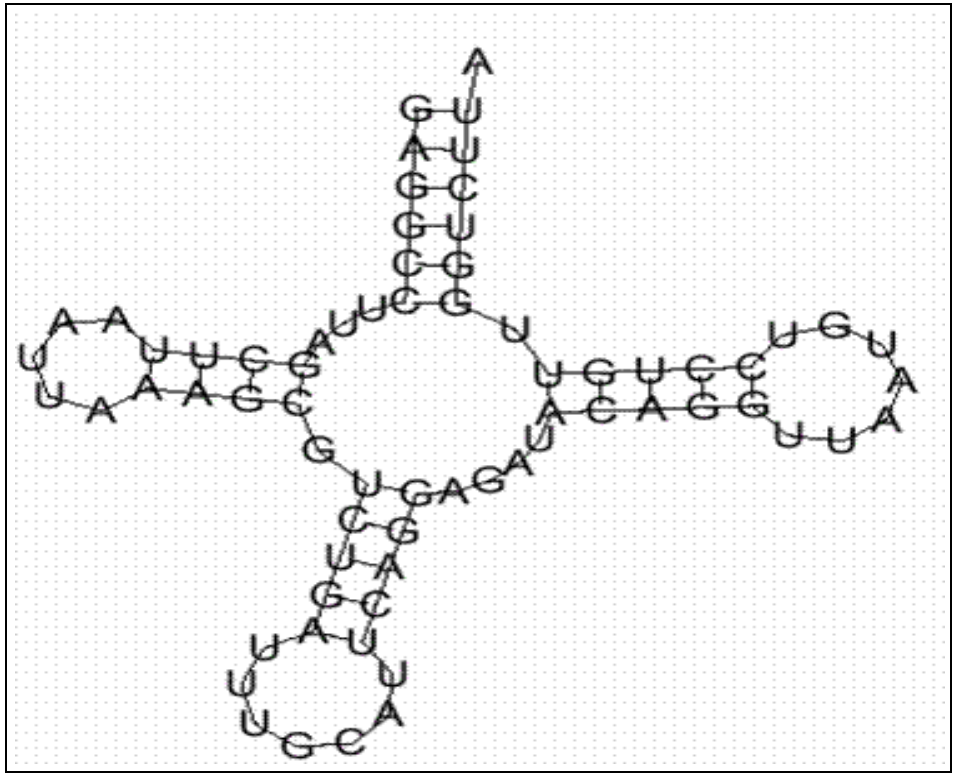

Figure 12-4 A new molecule

As illustrated in Figure 12-5 on page 240, the end user runs the demonstration via a Web browser to enter the parameters for the test. After it begins, the status of the grid may be monitored to see which nodes on the grid are actively working on the problem. At the end, the results are retrieved from the application server and sent to the end user workstation.

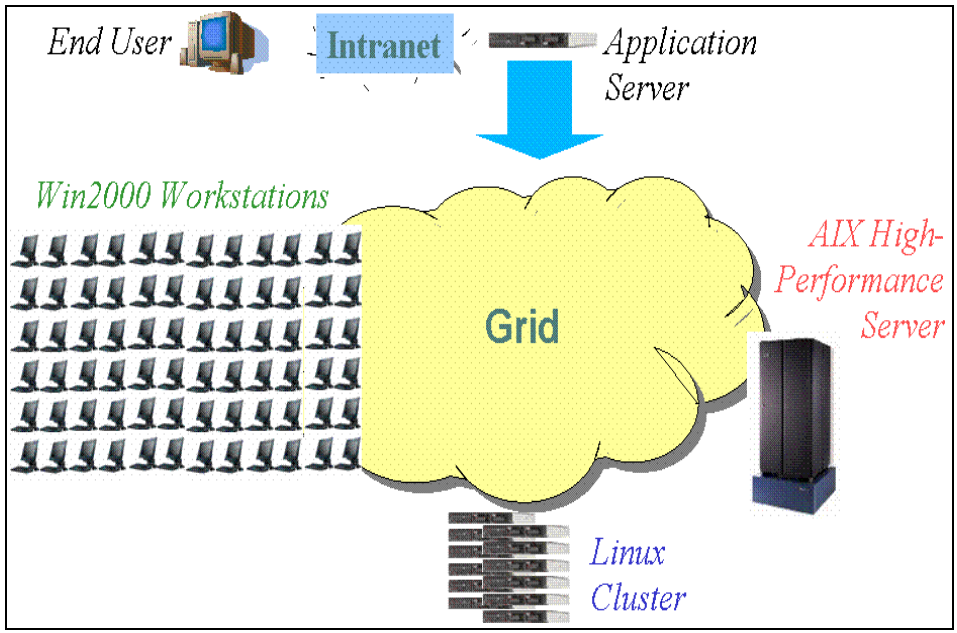

Figure 12-5 RNA folding over temperature variations

12.4 Video production

Digital frame rendering, converting graphics from a file into a continuous, visual form, is a highly computing intensive application. Maya software by Alias|wavefront is used for creating 3-D animation and digital effects. This demonstration shows how a grid can be used to process frames of video with Maya. It takes approximately 29 frames to create a single second of video (see Figure 12-6 on page 241). This implies that several thousands of frames need to be rendered for a video of any size for more than a few seconds. The more titles, effects, and filters that are used, the more processing power and time will be needed to create a finished video. This computing intensive application can use a grid by distributing individual frames to computers with free resources to complete the task. This is similar to the video conversion demo presented in Chapter 10, “Demo: Application” on page 197.

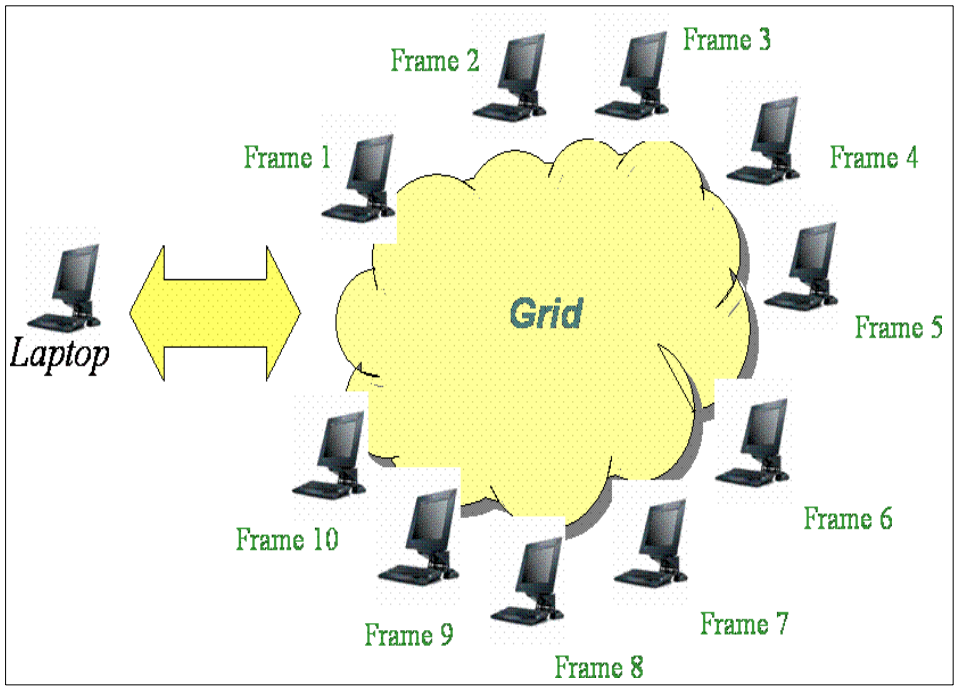

Figure 12-6 Digital frame rendering with Maya

12.5 Electronic Design Automation

The Electronic Design Automation (EDA) is a cooperative design application on a multi-cluster environment. Its demonstration focuses on hiding the complexity of grid computing. The user access is managed at a Web portal level from which access to the functionality provided by the grid is also controlled. Each user has access to an individual dataset per simulation. The Web portal uses the Linux PAM (Pluggable Authentication Mechanisms), which allows the administrator to select security mechanisms to use, such as LDAP, Kerberos, or a local file.

Figure 12-7 on page 242 illustrates the cooperative design of the application. Partners or vendors of the company work on different subcomponents of chips. They need to run simulations, as shown in Figure 12-8 on page 242, to test what they have developed under various conditions, such as voltage and temperature. The vendors may be competitors of each other, so it is important that each user has its own data and can only see its own results, as shown in Figure 12-9 on page 243.

The demonstration shows the capabilities of EnginFrame from NICE (http://www.nice-italy.com), as a computing and application portal. The grid also uses the LSF product from Platform Computing. Three IT centers are simulated by three clusters of Linux, AIX 4.3.3, and AIX 5.1.

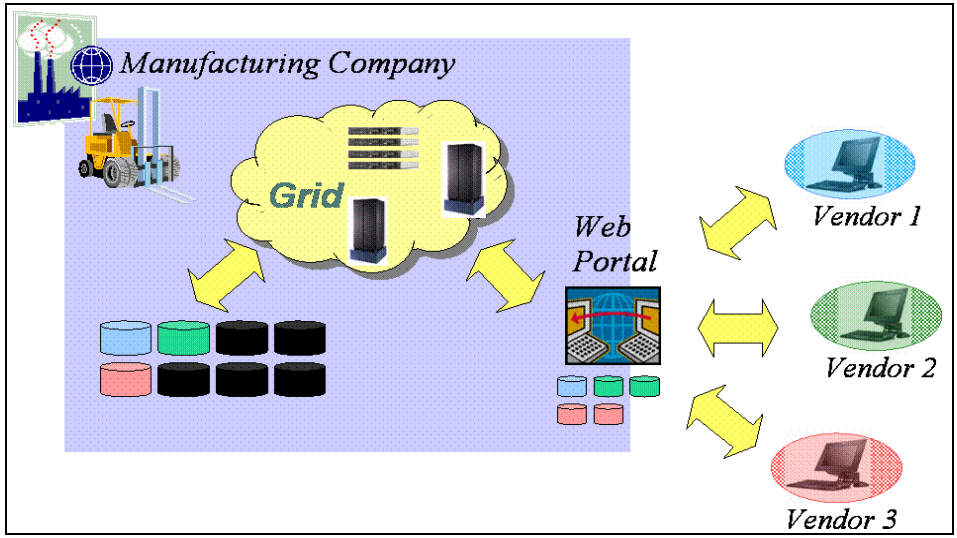

Figure 12-7 Electronic Design Automation

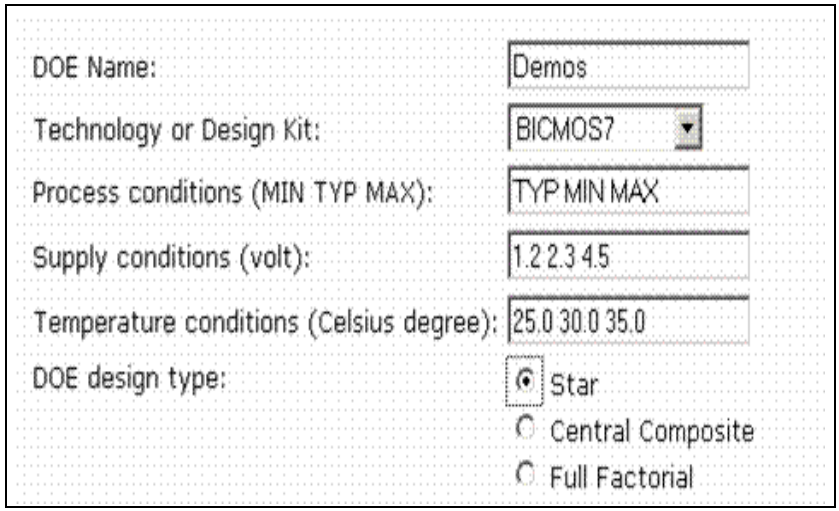

Figure 12-8 Sample parameters for EDA simulation

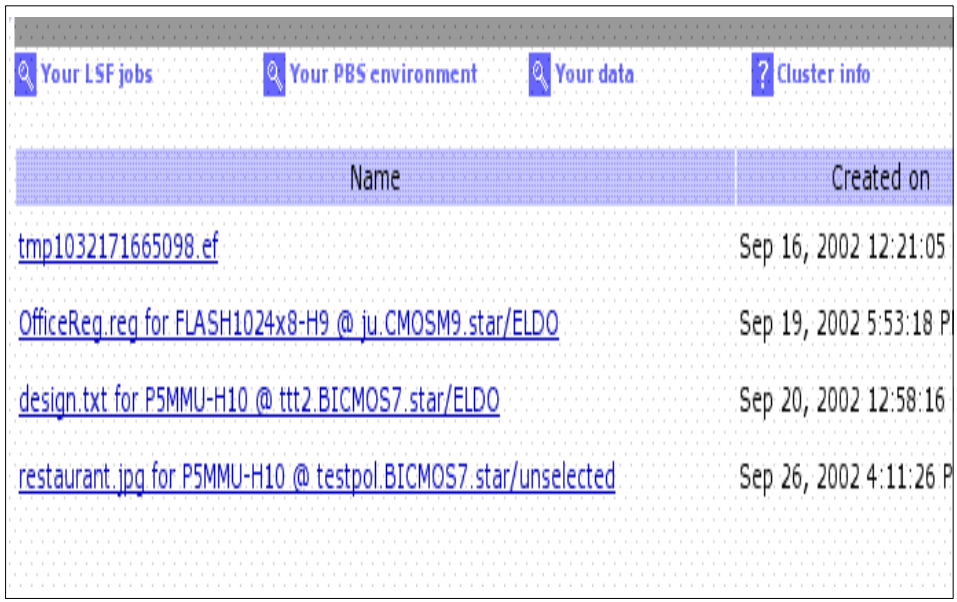

Figure 12-9 Retrieve results of simulation

12.6 Friendly Enterprises Grid - DNA String

The Friendly Enterprises Grid demonstration features a grid based Life Sciences application using Web services, as presented in Figure 12-10 on page 244. It implements a typical routine task in bio-science: A rather small query object (a DNA string, such as ACGGTAAG...) is compared against selected distributed sets of huge databases containing known gene data (again, long strings of characters, such as ACGAGTAAG...) to identify most similar sequences. Results are collected, merged and presented to the user for further scientific analysis, as shown in Figure 12-11 on page 245.

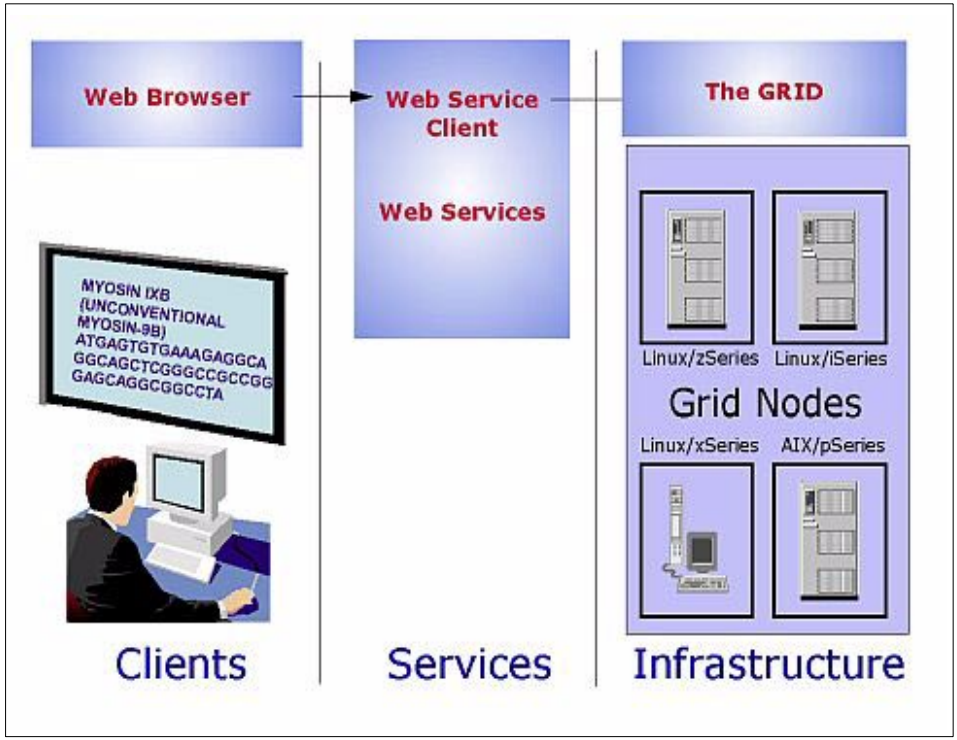

Figure 12-10 Friendly Enterprises Grid

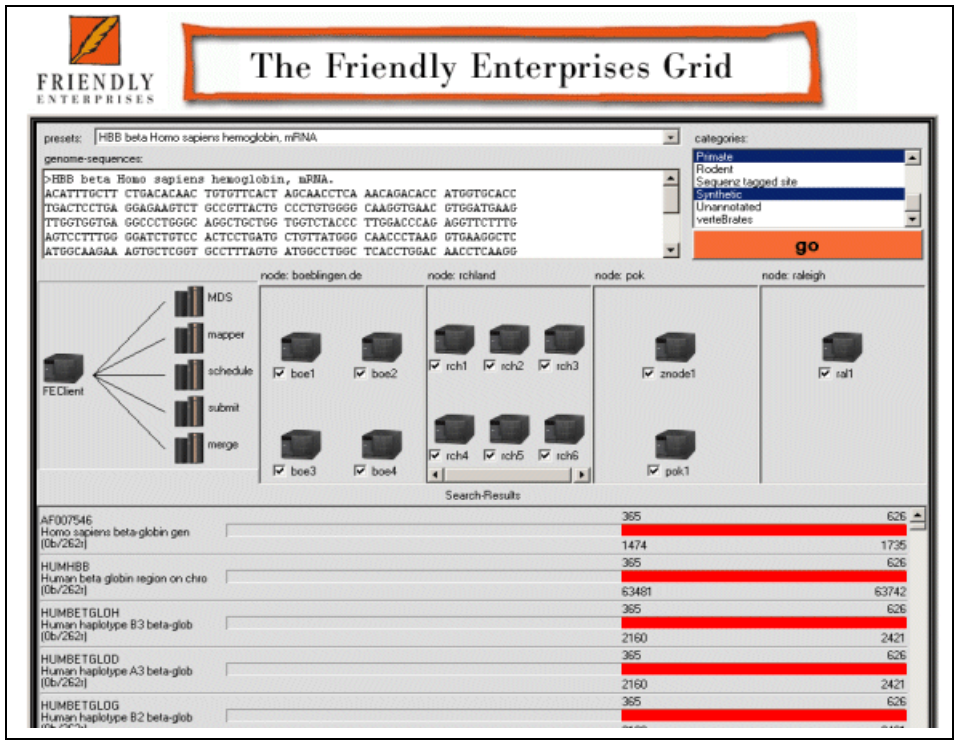

Figure 12-11 Friendly Enterprises Grid user interface

Emphasis in this demo, built in Boeblingen Lab in Germany, is on the grid and Web services aspects of the implementation, not on the Life Sciences scientific aspects, that is, although representing a rather special scientific environment, the demo is very well suited to exhibiting fundamental structures of evolving Open Grid Services Architecture (OGSA) environments and the natural synergy of Web services and grid infrastructures.

- The end user enters his request through a Web browser interface to the system where it is transformed into a request for Web services running on an application server environment tier.
- Based on information about the underlying grid of compute nodes supplied by the Globus Toolkit based infrastructure, the application server then can select nodes and dispatch jobs appropriately through additional Web services. Finally, after the completion of all jobs, other Web services merge results into a consolidated XML result file to be viewed in the end-user's Web browser.
- The underlying grid infrastructure is an heterogeneous set of compute nodes, and is currently a set of nodes (Linux/zSeries - AIX/pSeries -Linux/xSeries) in Boeblingen, nodes in Rochester (Linux/iSeries and AIX/pSeries), nodes in

Austin (Linux/pSeries and AIX/pSeries) and nodes in Poughkeepsie (Linux/zSeries and Linux/xSeries), subject to future local and remote changes/updates as appropriate.

Part 6

OGSA

13

Open Grid Services Architecture

This chapter introduces some of the motivations behind Open Grid Services Architecture (OGSA), the technology driving it, and problems addressed by the Grid Service Specification proposed by the Global Grid Forum (GGF) - Open Grid Services Infrastructure Working Group

The ideas explained here are taken primarily from four papers, *Web Services Conceptual Architecture (WSCA 1.0)*, by Kreger, which introduces Web Services concepts and terminology, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, by Foster, et al, which introduces Grid concepts and terminology, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, by Foster, et al, which introduces the Open Grid Services concepts and terminology, and the *Grid Service Specification*, by Tuecke, et al, which details the first steps towards OGSA (the Grid Service). In addition, information has been collected from different sources and they are mentioned by citation. More information on obtaining these sources can be found in “Related publications” on page 271.

13.1 Overview and directions

It should be noted that OGSA is work in progress, the future of grid computing if you like. Many of the problems that need to be solved are only just being tackled and those that are being tackled, such as the OGSI Working Group within GGF, are continually being revised (Grid Service Specification).

The GGF Open Grid Services Architecture Working Group is looking into producing a document to outline a road map for OGSA Services development. They aim to highlight the services that need to exist to have a fully functional grid environment and to suggest, at a high level, how these services should inter-relate. By highlighting the highest priority services, they will encourage the creation of other working groups within GGF to develop these services, for example, the OGSA Security Working Group.

OGSA by itself will not give you grid computing. It is an architecture and, as such, requires an implementation, a container perhaps, in which an OGSA Grid Service can exist. The OGSI working group exists to address the basics for the OGSA architecture: the Grid Service. We can think of Grid Service in a similar way to the Linux kernel; it is the core and the future for grid computing, although by itself it does not provide anything useful. It does provide building blocks for powerful grid middleware and applications, just as Linux provides the power for GNU tools and the popular Linux distributions and applications.

OGSA has the commitment of many vendors within the grid community, including Entropia, Unicore, United Devices, Platform, Globus, and IBM. Indeed, there is currently a tech-preview OGSA implementation available at the Globus Web site (`http://www.globus.org/ogsa`) and Globus Toolkit 3 is scheduled for beta release in April 2003. Globus is working to ensure backwards compatibility of Toolkit 3 with Toolkit 2.

Note: OGSA is becoming much greater than just the Grid Service Specification; working groups within GGF are already looking into Security, but there are hopes that the Grid Service Specification will near completion in 2003

13.2 Motivations for OGSA

OGSA has three main predecessors:

- Globus Toolkit
- Autonomic computing initiative
- Emerging Web Services standards

Globus Toolkit

Globus Toolkit Version 2 has been considered by many as the *de facto* standard for the implementation of grids, providing many key technologies, such as directory service for resource discovery, authentication, and job scheduling. The toolkit grew from a realization that to make a grid work for scientists and engineers required a significant middleware infrastructure. Such middleware protects the application writer from the underlying complexity of operating systems, and allows users to exploit the power of distributed systems without having to know the details of the systems being used.

Autonomic computing

The IBM autonomic computing initiative has much in common with Globus in that it strives to unify a user's view of a heterogeneous distributed system through the provision of common facilities. It has long been recognized that middleware requires a set of functionality, such as logging, security, failover, clustering, heartbeat monitoring, first failure data capture, trace, and so on. across a variety of platforms and technologies. Application servers, databases, or messaging engines all require similar functionality, but are often implemented in different ways by different middleware. For this reason, IBM embarked in this project to provide a common set of core infrastructure functions across all platforms.

Emerging Web Services standards

Web Services are one of the hottest topics in IT at the present time. Through a significant cooperative venture by some of the biggest players in the industry (including IBM and Microsoft) working through a number of standards organizations, a common set of technologies have emerged. These technologies allow for the registration, discovery, and use of distributed services.

13.2.1 Today's focus

The focus of grid computing had been on shared resources, where these resources are typical shared among large multi-national groups of institutes with heterogeneous environments. The Globus Toolkit has focused primarily on the academic community, particularly High Performance Computing, with a view to improve usage of a particular shared resource. This enables the end user to become increasingly less concerned with resources and more concerned with applications and making use of portals to carry out analysis of results ("services").

So, it is the application developers who are concerned with resources, but why should even they be concerned? The autonomic computing initiative was aimed at virtualizing these resources to application developers and IT staff. They, just like the end users, want to view the resources as services they can make use of and not expose the underlying heterogeneity that may exist.

Web Services traditionally allow companies to provide services to other companies and allow for the discovery of these services. However, the concept of a virtual organization within the grid blurs the boundaries of two companies, so Web Services are a viable approach to grid computing as well as B2B.

Grid resources and protocols are going to the Web Services approach of services and functionality. This moves grid technology towards an environment which enables QoS (Quality of Service) to play an important part, issues such as peak utilization, priorities, security, reliability, disaster recovery etc.

13.3 Basis for OGSA

In Chapter 2, “Application considerations” on page 39 and Chapter 4, “Design” on page 81, we have discussed requirements and characteristics of grid computing, topics basically related to the operation and management of computational resources, but homogeneous in certain sense.

There are many definitions of what is architecture. Take, for example, that of the IEEE Standard for Architectural Description of Software-Intensive Systems (IEEE P1471/D5.3):

“The fundamental organization of a system embodied by its components, their relationships to each other and to the environment and the principles guiding its design and evolution.”

To make use of the resource availability in a grid, we need to know the following:

- Are the resources that I need available somewhere?
- What are the characteristics of such resources?
- Do they satisfy my requirements?
- Where are they?
- How can I reach them?
- Can I trust the data storage and transport mechanisms?
- How can I get access to those resources?
- How much do I have to pay?

An answer to these questions, positive or not, implies that a series of inter-operable components have been arranged based on OGSA.

The OGSA foundation is based on the Globus Toolkit combined with Web Services.

13.3.1 The Globus Toolkit

The Globus Toolkit is the *de facto* standard for building grid infrastructures and applications. Details about Globus Toolkit can be found in Chapter 7, “Components” on page 131 and 5.2, “IBM Grid Toolbox (Globus)” on page 106.

Note: Version 3 of the Globus Toolkit is planned to be OGSA compliant.

13.3.2 Web Services

Web Services is an interface that describes a collection of operations that are network-accessible through standard XML messaging. Web Services is intended to facilitate the conversation or communication between computer programs. It is said that what the Web did for program-to-user communication, Web Services will do for program-to-program communication.

Most of the information described here is derived from the IBM paper *Web Services Conceptual Architecture (WSCA 1.0)* by Heather Kreger.

A Web Service is an interface that describes a collection of operations that are network-accessible through standardized XML messaging. It is a framework for the building and deployment of Web Services applications.

Some Web Services functionality exists today in some products, such as:

- IBM XML and Web Services Development Environment
- IBM Web Services Toolkit
- IBM WebSphere Application Server
- Microsoft.Nets
- JBoss
- BEA Systems

Web Services uses a program-to-program communications model built on existing standards, such as Hyper Text Transmission Protocol (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Description Discovery, and Integration (UDDI).

Web Services defines techniques for describing:

- Software components to be accessed
- Methods for accessing those components
- Discovery methods that enable the identification of service providers with the understanding that services are neutral entities, that is, a programming language, data, a system software, a computing device, and so on.

The Web Services model

As we see in Figure 13-1, the Web Services model has three roles: *Service Requestor*, *Service Provider*, and *Service Registry.*

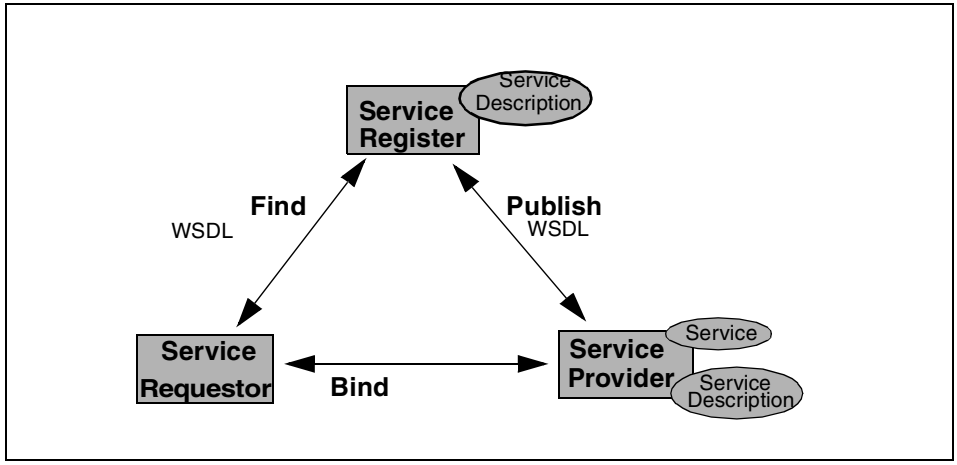

Figure 13-1 Roles in the Web Services model

Service Requestor

This is the business entity asking for a resource or service. From an architectural viewpoint, this is the application that is looking and invoking or initiating an interaction with the service. The Service Requestor role can be played by a browser in the hands of a person or by another service provider.

Service Provider

This is the resource owner or service owner. From an architectural viewpoint, this is the environment that hosts access to the service.

Service Registry

This is where requestors can find information about available resources or services. This is a searchable registry in which service providers publish their service descriptions and the availability of their services. Service Requestors find services and obtain binding information from the service descriptions for static binding or dynamic binding. A Service Requestor can bind directly to a Service Provider if they already have the binding information, then the Service Registry can be considered optional.

Operations in a Web Service architecture

As can be seen in Figure 13-1 on page 254, three different actions or operations can be taken that are related to a resource: *Publish*, *Find* and *Bind*.

- Publish

  To be accessible, a service description needs to be known (published).

- Find

  The entity looking for a specific type of resource must find the resource with the necessary characteristics.

- Bind

  Eventually, the service can be invoked and bound to be used.

The Web Services architecture defines two important concepts that are called artifacts. They are:

- Service

  This is the implementation of the Web Service interface.

- Service Description

  This is the details of the interface and the implementation of the service. This includes its data types, operations, binding information and network location. The service description might be published directly to a service requestor or to a service registry.

Their use can be a real execution if it is a computer program or just the use of its characteristics in the design phase. That is what Web Service architecture calls the two different phases for the service requestor; at design time, to get the service's interface description for program development, and at a runtime, to get the service's binding and location description for invocation.

Standards

OGSA proposes the heavy use of three standards: *Simple Object Access Protocol* (SOAP), *Web Service Description Language* (WSDL), and *Web Service Inspection* (WSI).

SOAP

This is a means for providing messaging between a Service Requestor and a Service Provider. It is a simple enveloping mechanism for XML payloads and defines a Remote Procedure Call (RPC) mechanism and conventions. SOAP payloads can be carried on HTTP, FTP, Java Messaging Service (JMS), and so on.

WSDL

This is a XML document for describing Web Services as a set of endpoints on messages containing either document-oriented (messaging) or RPC payloads.

WSI

This is a simple XML language, with related conventions for locating service descriptions published by a service provider. A WSI language (WSIL) document can contain a a collection of service descriptions and links to other source of service descriptions. It is a service description and, normally, is a URL to a WSDL document, but occasionally can be a URL to another WSI document. With WS-Inspection, a service provider creates a WSIL document and makes the document network accessible.

The Web Services stack

To perform the three operations (Publish, Find, and Bind) in an interoperable manner, there must be a Web Services stack that embraces standards at each level, as shown in Figure 13-2 on page 257.

The conceptual stack is built in a layered way. Each layer represents an action, an operation, or merely a document. The text on the left of Figure 13-2 on page 257 represents several standard technologies used by each layer. To the right are the requirements that must be addressed at every layer of the stack.

The foundation of the Web Services stack is the *network.* Web Services must be network accessible to be invoked by a service requestor; understand that the service has been published by the service provider and registered accordingly or communicated its existence directly to the service requestor.

HTTP is the *de facto* standard network protocol for Internet-available services. However, other Internet protocols can be supported, including SMTP and FTP.

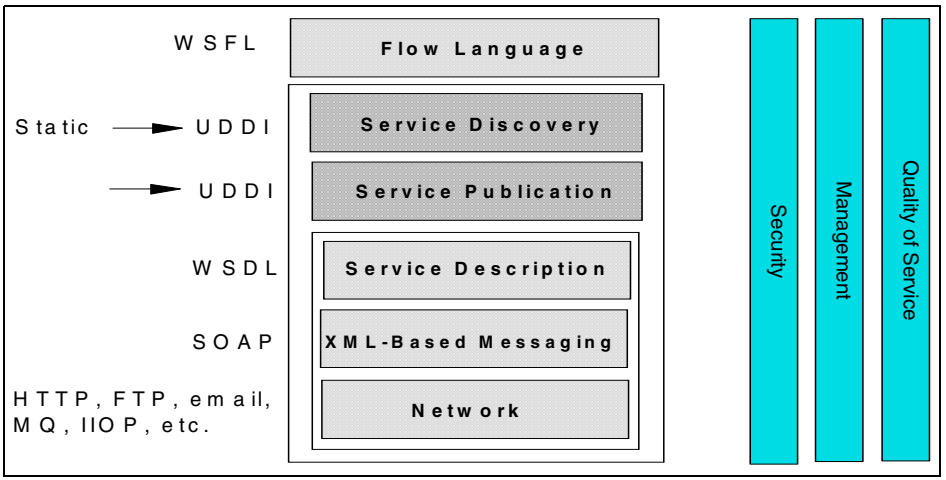

Figure 13-2 Web Services conceptual stack

The next layer, *XML-based messaging*, represents the use of XML as the basis for the messaging protocol. SOAP is a widely used example of XML messaging protocol due to its lack of complexity (HTTP POST with XML can be enveloped as payload). Services interact with one another by exchanging messages. The *Service Description* layer is actually a stack of description documents using WSDL. Any action that makes a WSDL document available to a service requestor qualifies as *Service Publication*. If the service provider sent the document (description) directly to the service requestor, it is said to be a *direct publication*. A Web Service cannot be discovered if it has not been published. This is done in the *Service Discovery* layer. Any mechanism that allows the service requestor to gain access to the service description and make it available to the application at runtime qualifies as service discovery. Finally, on top of the Web Service stack, we might find a Web Service flow model, the *Flow Language*, describing the feasible ways for composing several Web services together.

Figure 13-3 on page 258 shows XML messaging using SOAP and we can see the following:

1. A service requestor's application creates a SOAP message invoking a Web service operation provided by a service provider.
2. The network infrastructure delivers the message to the service provider's SOAP runtime (for example, a SOAP server).
3. The response SOAP message is presented to the SOAP runtime with the service requestor as the destination.

4. The response message is receive by the networking infrastructure on the service requestor's node.

This example uses the request/response transmission that is very common in most distributed environments.

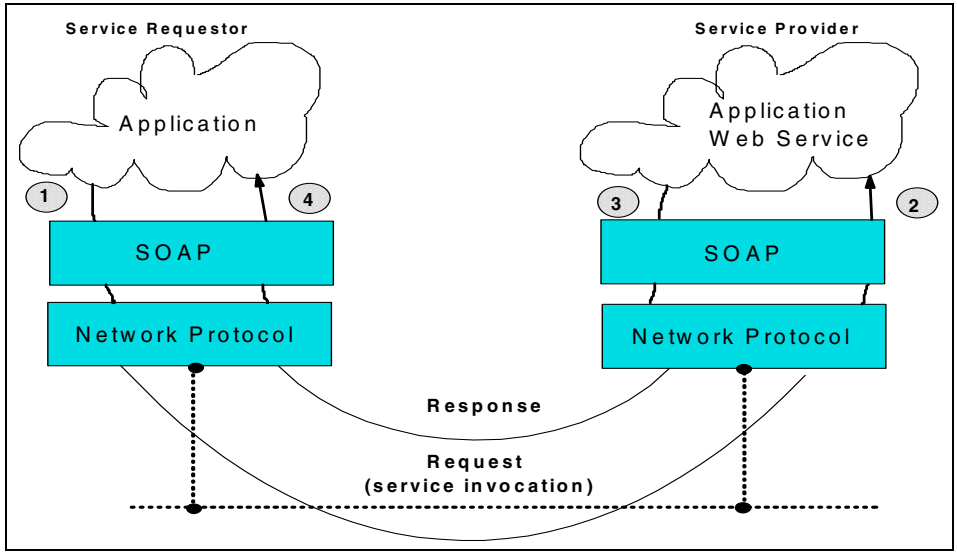

Figure 13-3 XML-based messaging using SOAP

13.3.3 Grid security

Grid security has been described in Chapter 3, "Security" on page 51. However, we would like to note that the concepts and functions are also requirements for OGSA. The minimal functional requirements to provide security for the service and resources are:

- Authentication

  Verifying the validity of a claimed individual and identifying who he or she is.

- Access Control

  Assurance that each user or computer that uses the service is permitted to do what he or she asks for.

- Data Confidentiality

  Assurance that sensitive information must not be revealed to parties that it was not meant for.

- Data Integrity

  Assurance that the data is not altered or destroyed in an unauthorized manner.

- Key Management

  The secure generation, distribution, authentication and storage of keys used in cryptography.

The Grid Security Infrastructure (GSI) and a Public Key Infrastructure (PKI) provide the technical framework (including protocols, services, and standards) to currently support grid computing with the five security requirements mentioned above.

Within the Web Services community, standards are emerging, such as WS-Security, to address these security requirements, but it is as yet unclear how the merging of existing technology with grid computing and the emerging technology within Web Services will combine.

Up to now, we have described standards and experience on which OGSA is based. In the next paragraphs, we will cover OGSA in more detail and how it is defined using the standards and concepts previously discussed.

13.4 OGSA in detail

A strong interface definition is one of the fundamentals of good software engineering. This is because interfaces can be used to achieve consistent definition of function while hiding the implementation of that function from the invoking application. Scale this behavior up to a distributed heterogeneous system and it is easy to see why a good interface definition is important. The Web Services Description Language (WSDL) provides a mechanism to define service interfaces in XML. These descriptions set out the structure and sequence of exchanges between the invoker and the service.

The exchanges between an invoker and a service must be carried over some transport mechanism using a protocol known at both end points. WSDL allows the same service to support multiple protocol bindings for a single interface. This capability contributes greatly to support of heterogeneous distributed systems. Not only can the binding be, for example, SOAP over JMS or HTTP, but a service can support multiple bindings for each offering, for example, different qualities of service or authentication mechanisms. All the variants of the service are described by the WSDL definition of that service.

OGSA brings the grid and Web service communities together to address the problem of services across a distributed, heterogeneous, dynamic, and virtual

organization. OGSA can be considered as taking Web Services into an environment in which state becomes important – we are exposing services that can be very dynamic or transient in nature, and these services may wish to maintain state for their lifetime or to allow re-use of the service in that state. The core of the OGSA architecture is the grid service. Grid services may be computational resources, storage resources, programs, or databases. Taking the Web services model as the example, grid services map very well to the concepts of registration, discovery, and use. The two critical aspects for users in such a Service Orientated Architecture (SOA) are definition of the service interfaces and identification of the protocol(s) that can be used to invoke a given service.

Nothing in the grid service specification dictates how grid services are written, what operating systems they run on, what languages they use, or the programming model they conform to. Grid services are equally applicable to native operating system processes written in C, C++, Fortran, or Java, and to container or component-based hosting environments, such as J2EE, WebSphere, .NET, and SunOne. The basic difference between the two approaches is the extent to which the writer of the service needs to worry about interaction with the underlying operating system. Native operating system process based services require more work to implement. However, the component-based model hides implementation from the invoking application or service so that low-level considerations, such as whether the service is managing its own run-time environment, are hidden from the invoker.

Interaction between services takes place by messages, and the existence of the internal state makes it important that a message is only delivered once or not at all. This becomes even more important in the context of an unreliable distributed system.

According to official Globus Project documentation, there is a commitment to develop a high-quality open source OGSA implementation. For this year and the forthcoming 2003, the Globus Project will evolve the current Globus Toolkit 2.2 towards an OGSA-compliant Globus Toolkit 3.0, working with its partners to address backward compatibility and transitional issues. A fact sheet detailing the Globus Toolkit 3.0 plans is available at:

http://www.globus.org/toolkit/gt3-factsheet.html

A basic premise of OGSA is that any computational resource like processor cycles, storage, memory, databases, and so on, can be understood as a service. Furthermore these services exhibit certain properties and needs that can be described as a core set of interfaces (or operations). The first Working Group to address OGSA within the Global Grid Forum is the Open Grid Services Infrastructure. The Grid Service Specification they are producing aims to specify this core set of behaviors as operations within several PortTypes, which can be composed with other Port Types to create a service.

Grid Services extend the Web Services concept by laying out a set of well-defined interfaces that address discovery, dynamic service creation, lifetime management, notification, and manageability and a set of conventions for naming and upgrade ability. These interfaces and conventions are vital for allowing reliable interoperability between services and with invoking applications. WSDL refers these interface as *portTypes*.

These portTypes include:

- GridService portType

  To allow the discovery of data relating to the service and enable lifetime management of the service

- Notification-Source portType

  To allow the sending of notification messages

- Notification-Sink portType

  To allow the receiving of notification messages

- Notification-Subscription portType

  To allow a NotificationSource portType to subscribe to a set of notifications for a period of time

- Registration portType

  To allow a service instance to register/unregister to enable/disable discovery of the service instance

- Factory portType

  To allow the creation of service instances

- HandleResolver portType

  To allow a grid service handle to be converted into a Grid Service Reference, necessary for binding to the service

The GridService portType is required for all services. The other portTypes are optional.

13.4.1 Needs in a grid process

There are several needs for building a robust grid environment. The following list describe some of them and how OSGA is addressing those needs.

- Dynamic Service Creation

  Virtual organization participants want more than persistent static services, where each interaction can be considered completely separate. They want to

be able to create service instances to handle the management and interactions associated with the state of particular activities.

To allow a dynamic service creation, OGSA uses the concept of a service factory (Factory portType), something we can use to create an instance of the service we desire. Transience is not a problem; since the factory is a persistent service, we can always create another instance at any point in the future.

- Dynamic Service Management

  We need a mechanism to identify a service instance once created. We may wish to return to that same instance to examine its state or it may be building up results from many other asynchronous services.

  OGSA addresses this need with the HandleResolver portType, Grid Service References (GSR), and Grid Service Handles (GSH). Once the service instance has been created, how can we allow anyone to connect to it and read/update its state? GSH provides a unique way of naming all OGSA services, but it does not enable us to use the service instance; we need the GSR to do this. We must have a way to convert a GSH into a GSR before we can actually use the service instance. For this task, OGSA uses a HandleResolver portType that takes a GSH and returns a GSR, which contains protocol and instance specific data to allow anyone to connect to the service instance.

  How do we find the HandleResolver to convert our GSH? OGSA indicates that the GSH is structured in such a way as to contain the home handle resolver for our service instance. Other HandleResolver may know about our service instance, but the home HandleResolver will always know about our service instance.

  This has implications for the factory, since we now need the factory to create a GSH and GSR for us as well as the service instance. The factory can look after choosing the home HandleResolver and ensure the home HandleResolver can map the GSH to the current GSR.

- Upgrade ability and compatibility

  A mechanism is required to upgrade services over time, and we might like to know if the new upgraded service is still compatible with the older version. We cannot guarantee that this service upgrade will take place at a convenient time. A client may be interacting with a service while the upgrade takes place; ideally, the client is never aware the upgrade has happened. However, if the service as a result of the upgrade, now supports different protocols or additional functions, the client may wish to know. A infrastructure should be provided to allow clients to become aware of these changes.

  OGSA addresses this need by providing the Grid Service portType. GSR does not solve the upgrade ability problem, since changes effectively

invalidates the GSR. We do not know who has the old GSR, so we cannot tell them what has changed.

GSH is effectively the GSR without protocol or instance specific information; this is unique for our service instance and will not change, even after upgrading our service instance. So if we use the GSH to tell everyone about our service instance, then this will still remain valid after an upgrade.

GSR will have a lifetime associated with it. This is an indication of how long the GSR is expected to be valid, which saves us the need to convert the GSH into a GSR. Of course, we may discover the GSR is invalid after all and still have to convert a GSH into a GSR, but until this point, we do not have the overhead of this conversion before connecting to the service instance.

- Lifetime management

  If each service instance is to be transient or dynamic in nature, then we must have some way of removing it when it is no longer required. Within a grid environment, an actual delete message may never reach the service instance due to unreliable transport within a distributed environment.

  OGSA addresses the lifetime issue by providing a “soft state registration” within the GridService portType. Upon creating a service instance, a lifetime is associated with it, which means the service instance will terminate itself at this point in time. However, anyone with correct permissions to connect to this service instance can extend its lifetime. If you know you need the service instance to exist for the next 24 hours, you would ensure its lifetime is set appropriately. So if any communication problems happen within the grid, the service instance will not exist forever and will destroy itself once its lifetime expires.

- Registration/Discovery

  A standard representation of information about a service and mechanism to enable someone to discover all alike services within a registry is necessary. This enables choices between service instances to be made based on things such as QoS, functions of the service, and perhaps attributes/state of a service instance.

  OGSA addresses this issue by providing “service data” elements within GridService portType and Registration portType. It indicates a registry should exist that contains service data and descriptions to allow the selection of a service instance. GSH is returned, which must be used to obtain the GSR before the service instance can be used.

- Notification

  A collection of distributed services must be able to notify each other of asynchronously changing states. There should be a basic standard mechanism to allow the subscription for notification from a service and delivery of these notifications.

OGSA addresses this notification issue by using the Notification-Source portType, Notification-Sink portType and Notification-Subscription portType. OGSA defines a source and access to support notification. You can give your notification to a service instance's notification source or use the subscription mechanism to give some indication of the notifications you am interested in.

You could even give the access for another service and not receive the notifications directly, and you may wish to ensure the source has a lifetime as long as you are interested in the notifications.

13.4.2 Conclusions

What does all this mean to people with a need to build applications? At one level, they will be able to exploit and compose grid services to build their applications rather than implementing these functions from scratch. They will find grid services by examining registries, and communicate with them using standard, well defined interfaces and conventions. It will allow exploitation of facilities and services that they do not own, but they can afford to occasionally use.

At another level, it will provide work for programmers to grid enable services in their organizations. By using Web Services, it allows easy migration and integration with the already existing Web Services environments.

The fact that the drive to define and develop OGSA is a broad-based collaborative effort rather than a proprietary solution and that it is based on emerging standards for Web Services should mean that, like the Internet, it will actually work.

Part 7

Appendixes

Glossary

Adaptive Algorithm. An algorithm that can *learn* and change its behavior by comparing the results of its actions with the goals that it is designed to achieve.

AFS. The **A**ndrew **F**ile **S**ystem is a mountable networked file system.

Autonomic computing. An approach to self-managed computing systems with a minimum of human interference. The term derives from the body's autonomic nervous system, which controls key functions without conscious awareness or involvement.

CA. A **C**ertificate **A**uthority is (**1**) an instance or external institute to issue authority certificates to identify the certificate holder to use certain services; (**2**) in e-commerce, an organization that issues certificates, authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who no longer exist.

CERN. The **C**onseil **E**uropeen pour la **R**echerche **N**ucleaire is a European organization for nuclear research.

DFS. A **D**istributed **F**iles **S**ystem is a type of mountable networked file system.

GASS. **G**lobal **A**ccess to **S**econdary **S**torage is used for file staging and cache management.

GGF. The **G**lobal **G**rid **F**orum was founded in 2001 when the merger of regional grid organizations created a single worldwide one

GIIS. **G**rid **I**ndex **I**nformation **S**ervice is the database that contains indexes of resource information registered by the GRIS and other GIISs. It can be seen as a grid-wide information server. GIIS has a hierarchical mechanism, like DNS, and each GIIS has its own name. This means client users can specify the name of a GIIS node to search for information.

Globus. A collaborative project centered at Argonne National Laboratory that is focused on enabling the application of grid concepts to computing.

GPFS. **G**eneral **P**arallel **F**ile **S**ystem is a type of mountable networked file systems

GRAM. The **G**rid **R**esource **A**llocation and **M**anagement API is a means allowing programs to be started on remote resources.

Grid Computing. A type of distributed computing in which a wide-ranging network connects multiple computers whose resources can then be shared by all end users; includes what is often called *peer-to-peer* computing.

GridFTP. The **Grid F**ile **T**ransfer **P**rogram provides high-performance and reliant data transfer.

GRIS. The **G**rid **R**esource **I**nformation **S**ervice is the repository of local resource information derived from information providers. GRIS is able to register its information with a GIIS, but GRIS itself does not receive registration requests. The local information maintained by GRIS is updated when requested, and cached for a period of time known as the time-to-live (TTL). If no request for the information is received by GRIS, the information will time out and be deleted. If a later request for the information is received, GRIS will call the relevant information provider(s) to retrieve the latest information.

GSI. The **G**rid **S**ecurity **I**nfrastructure contains components to secure your grid network.

IEEE. The **I**nstitute of **E**lectrical and **E**lectronics Engineers is a professional society accredited by the American National Standards Institute (ANSI) to issue standards for the electronics industry.

IESG. The **I**nternet **E**ngineering **S**teering **G**roup is the executive committee of the Internet Engineering Task Force (IETF). The IESG reviews and oversees the work produced by individual IETF working groups and charters all new working groups.

IETF. The **I**nternet **E**ngineering **T**ask **F**orce is the task force of the Internet Architecture Board (IAB) that is responsible for solving the short-term engineering needs of the Internet. The IETF consists of numerous working groups, each focused on a particular problem. Internet standards are typically developed or reviewed by individual working groups before they can become standards.

ISV. An **I**ndependent **S**oftware **V**endor is an enterprise offering software solutions.

ITS. **I**ntegrated **T**echnology **S**ervices is a sub division of IBM Global Services providing technical support services for IBM and non-IBM platforms and products.

J2EE. **J**ava **2** **E**nterprise **E**dition is the second version of the platform definition for advanced Java applications built on Enterprise Java Beans (EJB).

LDAP. The **L**ightweight **D**irectory **A**ccess **P**rotocol builds on TCP/IP to define a query-response protocol for querying the state of remote databases.

MPI. The **M**essage **P**assing **I**nterface is an application interface allowing formatted messages to enter and leave an application.

NCSA. The **N**ational **C**enter for **S**upercomputing **A**pplications is a US government institution for supercomputing.

NFS. A **N**etworked **F**ile **S**ystem is a mountable networked file system.

NTP. A **N**etwork **T**ime **P**rotocol server keeps the time on the network.

OASIS. The **O**rganization for the **A**dvancement of **S**tructure **I**nformation **S**tandards is an industry consortium promoting XML, TCP/IP, and UDDI.

ODN. An **O**pen **D**ata **N**etwork is capable of carrying all kinds of services for all kind of users from all kind of providers, for example, telephone or computer grids.

OGSA. The **O**pen **G**rid **S**ervices **A**rchitecture is a standard setting the base for communication in grids across virtual organizations. OGSA marries open standards and grid computing protocols with Web Services, bringing together the ability to share computing resources with the ability to provide application interoperability over the Internet.

QoS. **Q**uality **o**f **S**ervice is a term used in a Service Level Agreement (SLA) denoting a guaranteed level of performance (for example, response times less than 1 second).

RA. A **R**egistrant **A**uthority works in conjunction with the Certificate Authority (CA) approving or rejecting requests for certificates of public keys.

Secure Interface. The physical layer connection between a gateway and a secure network.

Secure Network. A set of nodes that are controlled by a single administrative party.

SLA. A **S**ervice **L**evel **A**greement is a contract in which a service provider agrees to deliver a minimum level of service.

SOAP. **S**imple **O**bject **A**ccess **P**rotocol is a means of sending messages between a service requester and provider.

SSL/TLS. **S**ecure **S**ocket **L**ayer **/ T**ransport **L**ayer **S**ecurity is a security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security.

TLS. **T**ransport **L**ayer **S**ecurity is a protocol that provides privacy and data integrity between two communicating applications, layered on top of TCP/IP.

UDDI. **U**niversal **D**escription **D**iscovery and **I**ntegration is a standard to use Web Services across various systems, including .NET and J2EE-based solutions.

URL. A **U**niversal **R**esource **L**ocator is the address that translates to a TCP/IP address in order to access resources in the Internet.

W3C. The **W**orld **W**ide **W**eb **C**onsortium is an independent work group formed in 1994 that provides standards like XML.

Web Services. A way of providing computational capabilities using standard Internet protocols and architectural elements. For example, a database Web Service would use Web browser interactions to retrieve and update data located remotely. Web Services use UDDI to make their presence known.

WSA. The **W**eb **S**ervices **A**rchitecture is a standardized approach based on SOAP and WSDL to develop Web Service solutions.

WSDL. **W**eb **S**ervices **D**escription **L**anguage is a mechanism for representing the services that a provider has and is making available and the specifics for accessing that service.

WSFL. The **W**eb **S**ervices **F**low **L**anguage addresses flows and dependencies between services.

WSIL. **W**eb **S**ervices **I**nspection **L**anguage is a means for discovering the WSDL-described services that a provider has made available.

XML. e**X**tensible **M**arkup **L**anguage is a data representation method.

Virtual Organization. A virtual entity whose users and servers are geographically apart but share their resources collectively as a larger grid. The users of the grid can be organized dynamically into a number of virtual organizations, each with different policy requirements.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 276.

- *Building a Linux HPC Cluster with xCAT*, SG24-6623
- *Deploying a Public Key Infrastructure,* SG24-5512
- *Linux Clustering with CSM and GPFS*, SG24-6601
- *Understanding LDAP,* SG24-4986

Other resources

These publications are also relevant as further information sources:

- *Fundamentals of Grid Computing,* REDP3613
- Foster, et al, *The Grid: Blueprint for a New Computing Infrastructure,* Morgan Kaufmann, 1999, ISBN 1558604758
- *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, found at:

  http://www.globus.org/research/papers/anatomy.pdf
- *A Brief Introduction to Grid Technology*, found at:

  http://www.bo.infn.it/alice/introgrd/introgrd/
- *Computational Grids*, found at:

  http://www.globus.org/research/papers/chapter2.pdf
- *The Globus Project: A Status Report*, found at:

  ftp://ftp.globus.org/pub/globus/papers/globus-hcw98.pdf
- *GridFTP Update January 2002*, found at:

  http://www.globus.org/datagrid/deliverables/GridFTP-Overview-200201.pdf
- *Grid Service Specification*, found at:

  http://www.gridforum.org/ogsi-wg/drafts/GS_Spec_draft03_2002-07-17.pdf

- Internet Draft GridFTP: Protocol Extensions to FTP for the Grid, found at:

  http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf

- Internet Draft Internet X.509 Public Key Infrastructure Proxy Certificate Profile, found at:

  http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-03.txt

- *MDS 2.2 User's Guide*, found at:

  http://www.globus.org/mds/mdsusersguide.pdf

- *The Open Grid Services Architecture and Data Grids*, found at:

  http://aspen.ucs.indiana.edu/CCPEwebresource/c600gridkunszt/c600FINALGridServices_DataGridv3.pdf

- *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, found at:

  http://www.globus.org/research/papers/ogsa.pdf

- *A Resource Management Architecture for Metacomputing Systems*, found at:

  ftp://ftp.globus.org/pub/globus/papers/gram97.pdf

- RFC 3280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, found at:

  http://www.ietf.org/rfc/rfc3280.txt

- *Web Services Conceptual Architecture (WSCA 1.0)*, found at:

  http://www.ibm.com/software/solutions/webservices/pdf/WSCA.pdf

- *Web Service Description Language (WSDL) 1.1*, found at:

  http://www.w3.org/TR/wsdl

Referenced Web sites

These Web sites are also relevant as further information sources:

- Argonne National Laboratory

  http://www.anl.gov/

- Avaki Corporation

  http://www.avaki.com/

- Cactus

  http://www.cactuscode.org/

- CDRDAO

  http://cdrdao.sourceforge.net/index.html

- Commodity Grid Kits

  http://www-unix.globus.org/cog
- Condor

  http://www.cs.wisc.edu/condor/
- Database Access and Integration Services

  http://www.gridforum.org/6_DATA/dais.htm
- The DataGrid Project

  http://eu-datagrid.web.cern.ch/eu-datagrid/
- DataSynapse, Inc

  http://www.datasynapse.com/
- Distributed Systems Laboratory - Argonne National Laboratory

  http://www-fp.mcs.anl.gov/dsl/
- Enginframe

  http://www.nice-italy.com
- Entropia, Inc.

  http://www.entropia.com/
- globus_gsincftp-0.2 .tar file

  http://www-unix.globus.org/ftppub/gt2/2.0/contrib/globus_gsincftp-0.2.tar.gz
- Global Grid Forum

  http://www.gridforum.org/
  http://www.ggf.org/
- Globus API documentation

  http://www-unix.globus.org/api/c-globus-2.2
- Globus bin bundles

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/bundles/bin
- Globus contributor's src files

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/contrib/src
- Globus GRAM job manager src and bin bundles

  ftp://ftp.globus.org/pub/gt2/2.2/2.2.2/extra/gram_job_manager
- The Globus Project

  http://www.globus.org/

- The Globus Project Bugzilla Database

  http://bugzilla.globus.org/
- Globus Project collaborators

  http://www.globus.org/about/collaborators.html
- Globus Resource Allocation Manager (GRAM) 1.6

  http://www.globus.org/gram
- The Globus Resource Specification Language RSL v1.0

  http://www-fp.globus.org/gram/rsl_spec1.html
- Globus Simple CA directory

  ftp://ftp.globus.org/pub/gsi/simple_ca
- Globus src bundles

  ftp://ftp.globus.org/pub/gt2/2.2/2.2-latest/bundles/src
- Globus Toolkit

  http://www.globus.org/toolkit
- Globus Toolkit 2.2 Advisories

  http://www.globus.org/gt2.2/advisories.html
- Globus Toolkit 2.2 Download Page

  http://www.globus.org/gt2.2/download.html
- Globus Toolkit 2.2 Platform Notes

  http://www.globus.org/gt2.2/platform.html
- Globus Toolkit Public License

  http://www.globus.org/toolkit/download/license.html
- GNU VCDImager

  http://www.vcdimager.org
- Google Internet search engine

  http://www.google.com
- Grid Security Infrastructure (GSI)

  http://www.globus.org/security
- IBM Grid computing

  http://www.ibm.com/grid/
- The Internet Engineering Task Force

  http://www.ietf.org/

- Linux Digital Video

  http://www.schirmacher.de/arne/dvgrab/index_e
- MPICH-G2

  http://www3.niu.edu/mpi
- NASA Information Power Grid

  http://www.nas.nasa.gov/About/IPG/ipg.html
- OASIS

  http://www.oasis-open.org/
- OGSA

  http://www.globus.org/ogsa
- OpenSSL Project

  http://www.openssl.org
- Platform Computing

  http://www.platform.com/
- Project: GNU/Linux 1394 AV/C Library: Summary

  http://sourceforge.net/projects/libavc1394
- Project: Libraw1394: Summary

  http://sourceforge.net/projects/libraw1394
- Project: The MJPEG/Linux square: Summary

  http://sourceforge.net/projects/mjpeg
- Project: Quasar DV Codec: Summary

  http://sourceforge.net/projects/libdv
- San Diego Supercomputer Center

  http://www.sdsc.edu/
- Secure Shell Charter

  http://www.ietf.org/html.charters/secsh-charter.html
- Status and Plans for Globus Toolkit 3.0

  http://www.globus.org/toolkit/gt3-factsheet.html
- TeraGrid

  http://www.teragrid.org/
- United Devices

  http://www.ud.com

- University of Chicago

  http://www.uchicago.edu/
- University of Southern California Information Sciences Institute

  http://www.isi.edu/
- World Wide Web Consortium (W3C)

  http://www.w3.org/
- ZetaGrid

  http://www.zetagrid.net
- ZetaGrid downloads

  http://www.zetagrid.net/zeta/downloads.html

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

D

E

F

G

S

T

U

V

W

X

Z

Redbooks

Introduction to Grid Computing with Globus

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Introduction to Grid Computing with Globus

Fundamentals and concepts

Using the Globus Toolkit

Grid demos and OGSA

This IBM Redbook is intended to give readers interested in the technical aspects of grid computing a hands-on introduction using the Globus Toolkit. This will include a discussion of the first basics of grid computing, applications to be run on the grid, and various grid products and architectures that are currently available.

This redbook is a good starting point for learning about grid computing, and gives you a good foundation before learning more about the future of grid computing, OGSA, e-business, and IBM's vision of the on-demand era.

This redbook covers the following topics:

- Grid computing fundamentals
- Architecture and security considerations
- Introduction of OGSA (Open Grid Services Architecture)
- Description of the components of the Globus Toolkit
- Globus Toolkit Version 2.2 implementation

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6895-00 ISBN 0738427969