

Perspectives on Distributed Computing: March 2008 Quarterly Report

Lisa Childers (childers@mcs.anl.gov)
Lee Liming (liming@mcs.anl.gov)

*Distributed Systems Lab
Mathematics and Computer Science Division
Argonne National Laboratory*

*Computation Institute
University of Chicago*

31 March 2008

Table of Contents

1	EXECUTIVE SUMMARY	4
2	INTRODUCTION.....	5
3	INTERVIEW SUMMARIES: AN INTEGRATED VIEW	6
3.1	USER GOALS	8
3.2	METHODS FOR DEVELOPING PRODUCTS	9
3.3	METHODS FOR PROVIDING INFRASTRUCTURE	10
3.4	METHODS FOR PURSUING SCIENCE GOALS	11
3.5	USER ISSUES	12
3.6	SATISFACTION POINTS.....	13
4	CHARACTERIZING USERS	14
4.1	INTERACTIONS WITH TECHNOLOGY.....	14
4.2	TECHNOLOGY INTERACTION CLUSTERS	15
5	 USER PROFILE: THE HPC SCIENTIST	17
6	 USER PROFILE: THE HPC DOMAIN-SPECIFIC DEVELOPER.....	18
6.1	USER OVERVIEW	19
6.2	USER GOALS	19
6.3	USER ISSUES	21
7	 USER PROFILE: THE GENERAL-PURPOSE HPC INFRASTRUCTURE PROVIDER	24
8	 USER PROFILE: THE GENERAL-PURPOSE HPC TECHNOLOGY DEVELOPER.....	25
9	RECOMMENDATIONS	26
10	CONCLUSION.....	27
APPENDIX A STUDY METHODOLOGY.....		28
APPENDIX B THE INTERVIEWEES		29
APPENDIX C SUMMARY DATA FOR SECTION 3.....		34
C.1	USER GOALS	34
C.2	METHODS FOR DEVELOPING PRODUCTS.....	38
C.3	METHODS FOR PROVIDING INFRASTRUCTURE	46
C.4	METHODS FOR PURSUING SCIENCE GOALS	58
C.5	USER ISSUES.....	67
C.6	SATISFACTION POINTS	84
APPENDIX D SUMMARY DATA FOR SECTIONS 5-8.....		89
D.1	HPC SCIENTIST.....	89
D.2	HPC DOMAIN-SPECIFIC DEVELOPER.....	89
D.3	GENERAL-PURPOSE HPC INFRASTRUCTURE PROVIDER.....	92
D.4	GENERAL-PURPOSE HPC TECHNOLOGY DEVELOPER	92

APPENDIX E	COMPLETE INTERVIEW TRANSCRIPTS	93
E.2	 TROUBLESHOOTING REQUIRES KNOWLEDGE ABOUT SOFTWARE INTERNALS	93
E.3	 THE GRID IS A BLACK BOX TO ME	93
E.4	 THE REASON MY TASKS ARE SO TIME-CONSUMING IS FAILURE	93
E.5	 PERFORMANCE IMPROVED FROM DAYS TO SECONDS	93
E.6	 THE GRID IDEA IS GREAT, BUT THERE ARE BARRIERS TO MAKING IT WORK TODAY	93
E.7	 IF THINGS DON'T WORK YOU NEED AN EXPERT TO FIX THEM	93
E.8	 I AM TRYING TO UNDERSTAND WHERE GRID COMPUTING ADDS VALUE	93
E.9	 GLOBUS ENABLES MORE SCIENCE	93
E.10	 SOLVING PROBLEMS IS EASY ONCE YOU HAVE ALL THE DATA	93
E.11	 RESOURCE USAGE WITHIN OUR VIRTUAL ORGANIZATION IS OPPORTUNISTIC	93
E.12	 THE RIGHT APPROACH IS TO BE HIGHLY COLLABORATIVE WITH DOMAIN SPECIALISTS	93
E.13	 WE PLAY A STRONG BRIDGE ROLE IN CONNECTING PEOPLE WITH TECHNOLOGY	93
E.14	 I START WITH MICROBENCHMARKS AND FOLLOW-UP WITH REAL APPLICATIONS	93
E.16	 WE ASSUME A WORLD WHERE LIGHTPATHS CAN BE SCHEDULED BETWEEN COMPUTERS	93
E.17	 GRAM2 IS KEPT ALIVE BY THE NEED TO INTEROPERATE WITH EUROPEAN EXPERIMENTS	93
E.20	 WE CAN PROVIDE OUR USERS WITH FRESH DATA MORE FREQUENTLY BECAUSE OF THE GRID	93
E.21	 WE WORK TO ENABLE DISCOVERY, ACCESS AND SYNTHESIS OF DISTRIBUTED DATASETS	93
E.26	 OUR GOAL IS TO MAKE IT EASIER TO TROUBLESHOOT GRID APPLICATIONS	93
E.27	 THE END GOAL IS TO AUTOMATICALLY DETECT NETWORK ANOMALIES	93

1 Executive Summary

This is the third quarterly report issued by the User Perspectives study, a subproject of the *Community Driven Improvement of Globus Software (CDIGS)* project. This report is a preview of the final report, which is expected to be released in mid-2008.

The purpose of the User Perspectives study is to document the work-related goals, methods and challenges faced by users of distributed computing technology today. To this end, interviews have been conducted with a variety of technology users in the scientific community. The following material will be included in the final report:

- A method for characterizing users according to their technology interactions
- The identification of at least four types of users from the interview data
- A description of each user type
- Interview transcripts and summaries
- A description of the interview protocol
- Recommendations for further work

This preliminary report contains a subset of the planned content. Those wishing to examine the key findings contained in this report should read *Section 4, Characterizing Users*, which describes the method used to characterize users, and *Section 6, User Profile: The HPC Domain-Specific Developer*, which describes one of the user types found in the data.

Feedback on this report is appreciated; please send comments to childers@mcs.anl.gov.

2 Introduction

The purpose of the User Perspectives project is to document the work-related goals, methods, and challenges facing today’s scientific technology users and to record their perspectives on Globus software and the distributed-computing ecosystem.

In the remainder of the document, we present interview summaries (Section 3), categorize the users (Section 4), and discuss user profiles for the types of users found in the interview data (Sections 5-8). Five appendixes also are included, providing details about the methodology and data used for the main body of the report.

Note: Content listed in gray is not included in this preliminary report but planned for the final report.

Table 1: Current and planned final report content

	To learn about the findings, read:	For further examination, read:
For no detail	- Section 1, Executive Summary	
For minimal detail	- Section 2, Introduction - Section 3, Interview Summaries - Section 4, Characterizing Users - Section 5, The Simulation Scientist - Section 6, The HPC Domain-Specific Developer - Section 7, The General-Purpose HPC Infrastructure Provider - Section 8, The General-Purpose HPC Technology Developer - Section 9, Recommendations - Section10, Close	- Appendix A, Study Methodology - Appendix B, The Interviewees
For greater detail		- Appendix C, Summary Data for Section 3 - Appendix D, Summary Data for Sections 5-8
For greatest detail		- Appendix E, Complete Interview Transcripts

3 Interview Summaries: An Integrated View

The interview summaries provide an integrated view of the work-related goals, methods, issues and satisfaction points reported in the interviews of the study participants. More broadly, they provide examples of how users work in the field today. Note that the summaries are not representative of the community as a whole; experiences not reported in an interview are not represented.

How the summaries were created

We identified interview excerpts in which users describe work relevant to the particular topic. Within these we created broad, second-tier categories by grouping excerpts according to their similarities. A third tier was then created by summarizing the entries observed within each of the second-tier groups. Duplicate entries are listed once.

Sample interpretations

Subsections 3.1-3.6 present a detailed overview of the six top-level topics, respectively. Figures 1-6 show one example from each of these top-level topics.

"Some domain experts run simulations in order to conduct research in computational astrophysics."

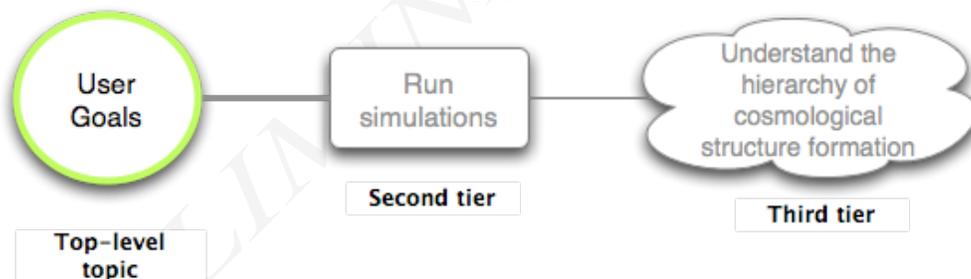


Figure 1

"Some domain experts determine product requirements in cooperation with their peers."



Figure 2

"Some users work with both domain and CS experts to integrate scientific applications."

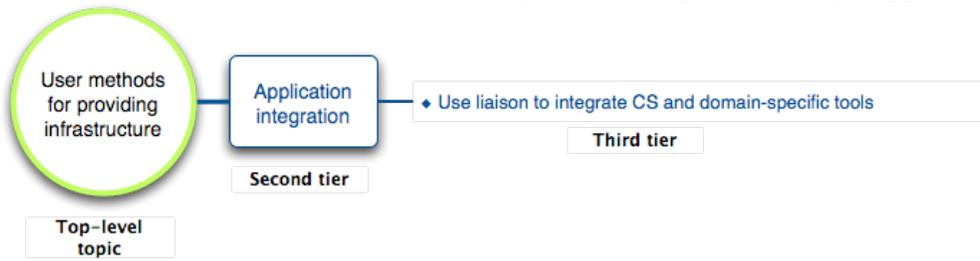


Figure 3

"Some users work to distribute a terabyte of domain data per day."

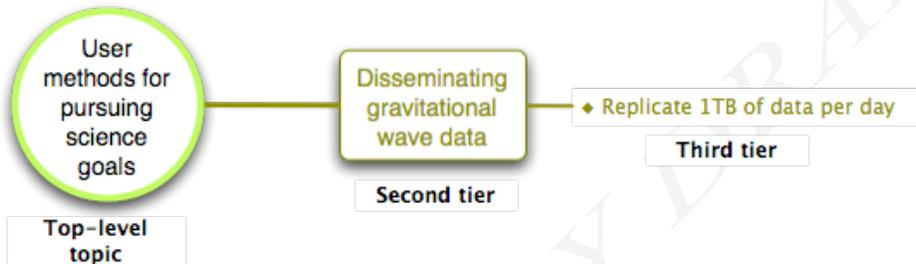


Figure 4

"Some users experience reliability problems due to system failures."

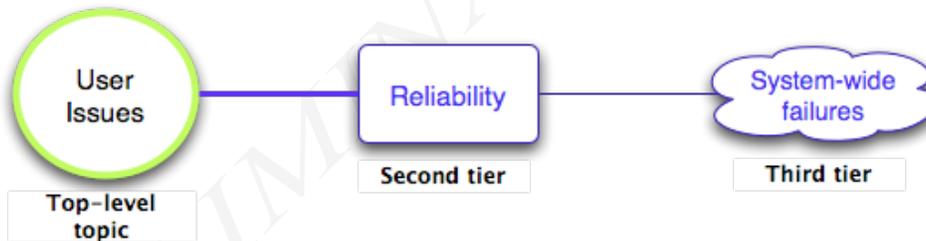


Figure 5

"Some users highly value the transfer rates offered by GridFTP."

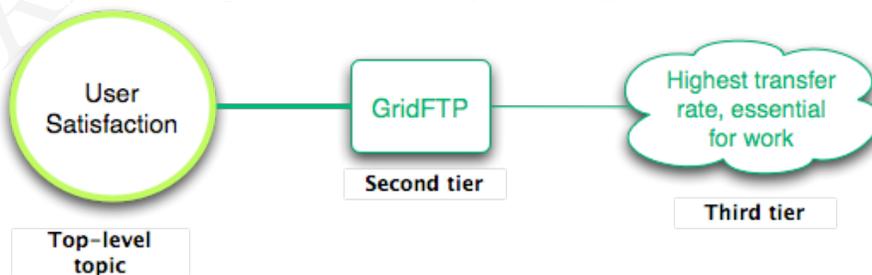


Figure 6

3.1 User Goals

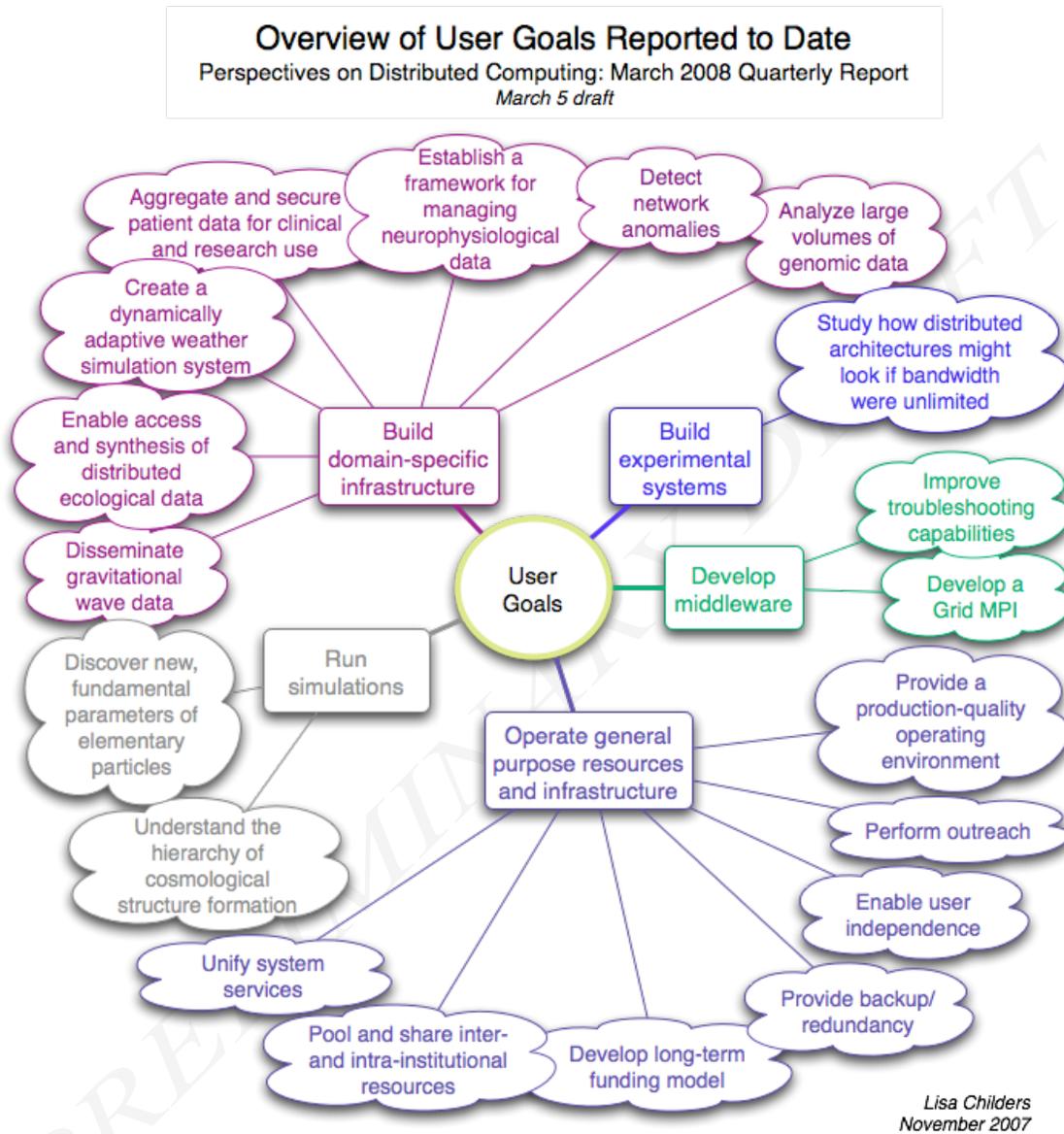


Figure 7: User goals reported to date

[Summary text for this picture will be included in the final report. The observations underlying this summary picture can be found in Appendix C of this report.]

3.2 Methods for Developing Products

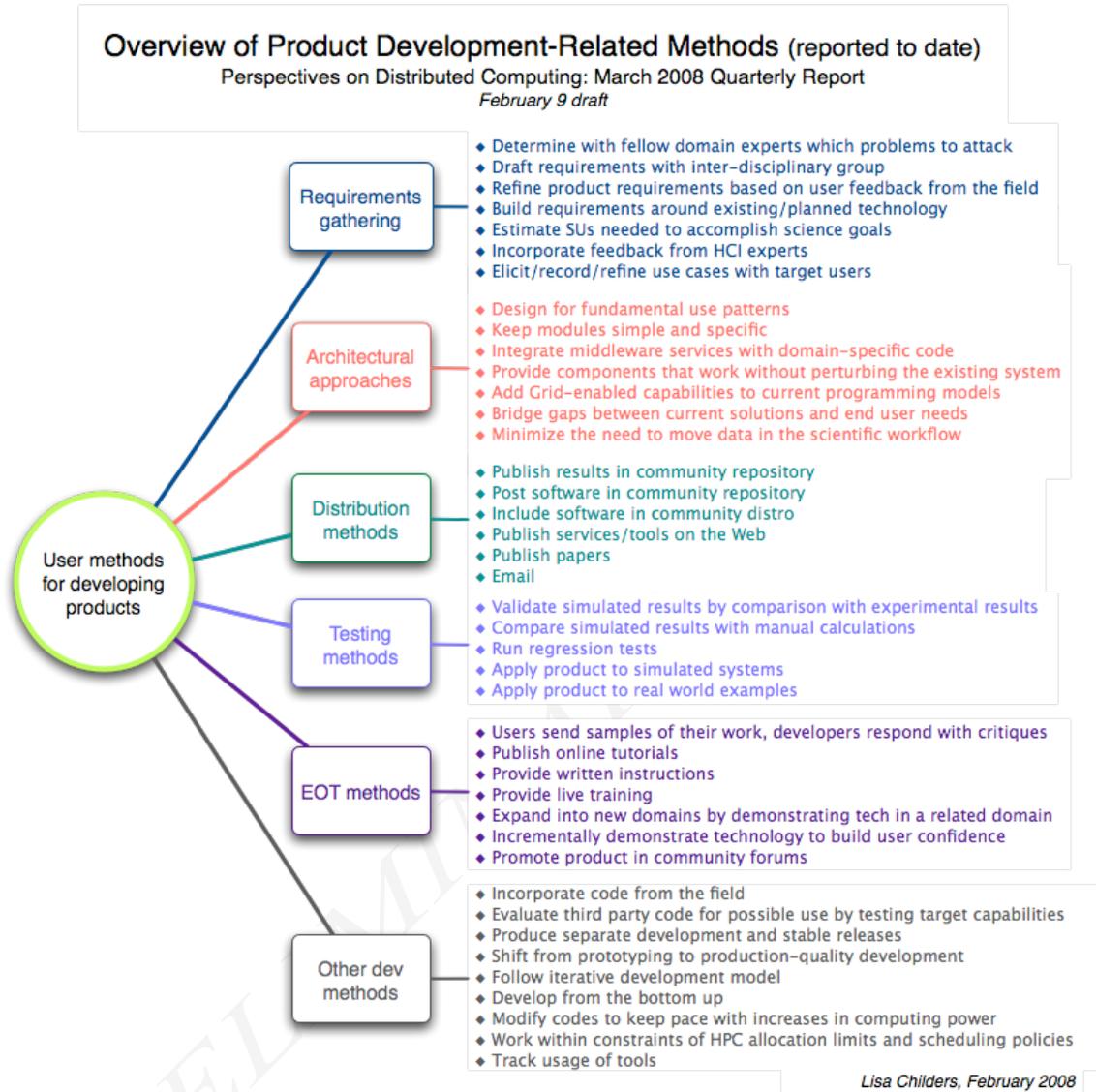


Figure 8: Methods for developing products reported to date

[Summary text for this picture will be included in the final report. The observations underlying this summary picture can be found in Appendix C of this report.]

3.3 Methods for Providing Infrastructure

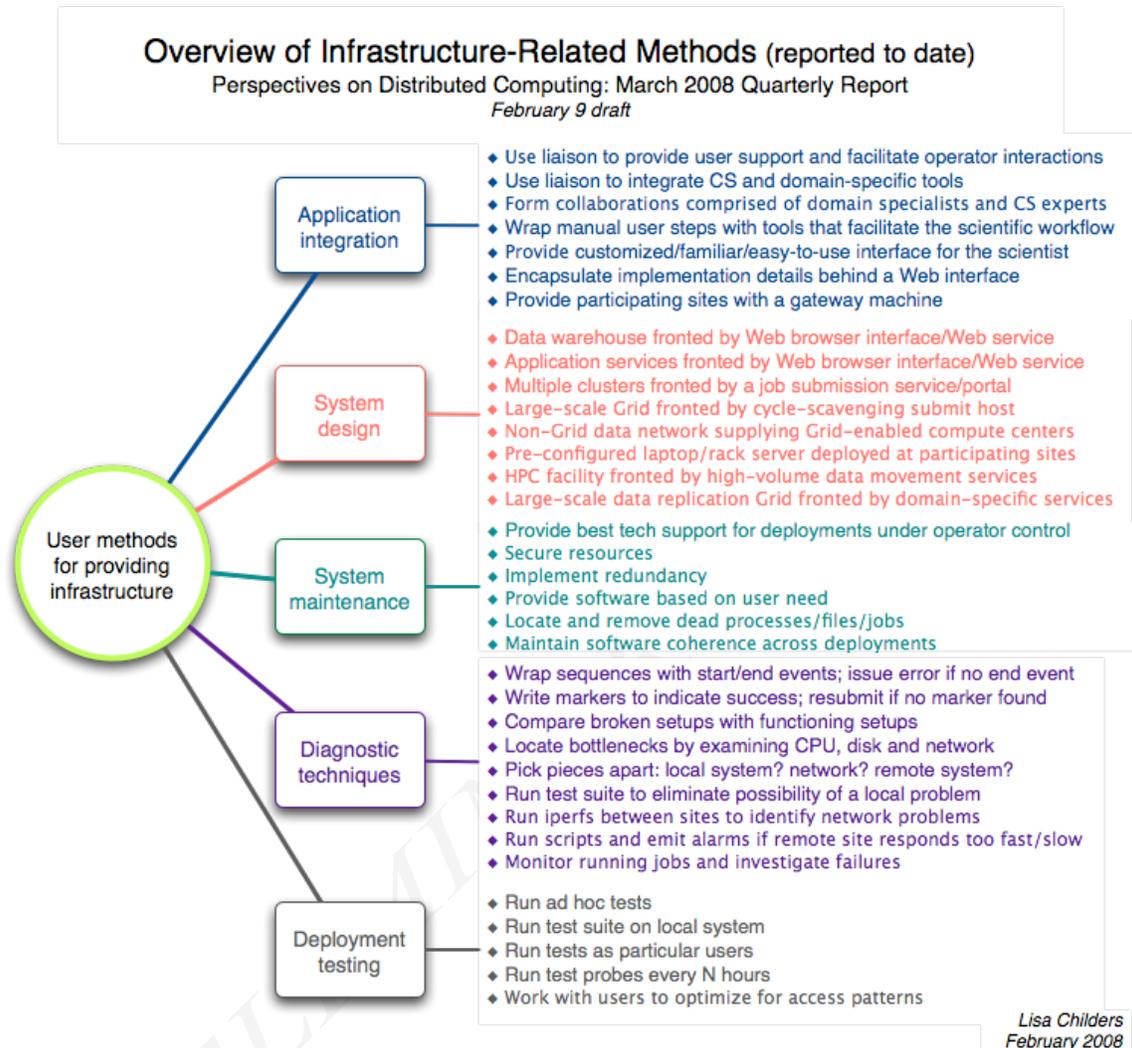


Figure 9: Methods for providing infrastructure reported to date

[Summary text for this picture will be included in the final report. The observations underlying this summary picture can be found in Appendix C of this report.]

3.4 Methods for Pursuing Science Goals

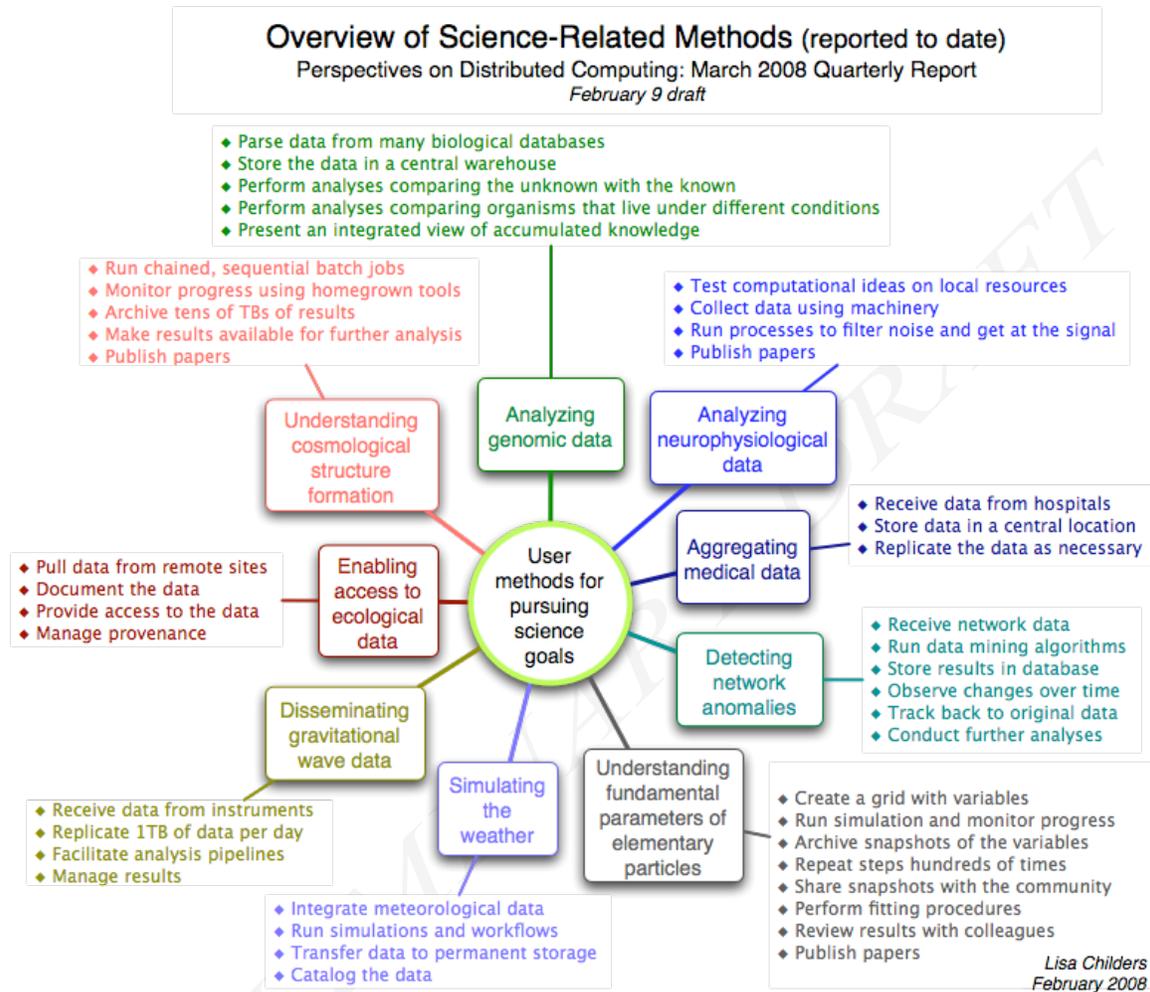


Figure 10: Methods for pursuing science goals reported to date

[Summary text for this picture will be included in the final report. The observations underlying this summary picture can be found in Appendix C of this report.]

3.5 User Issues

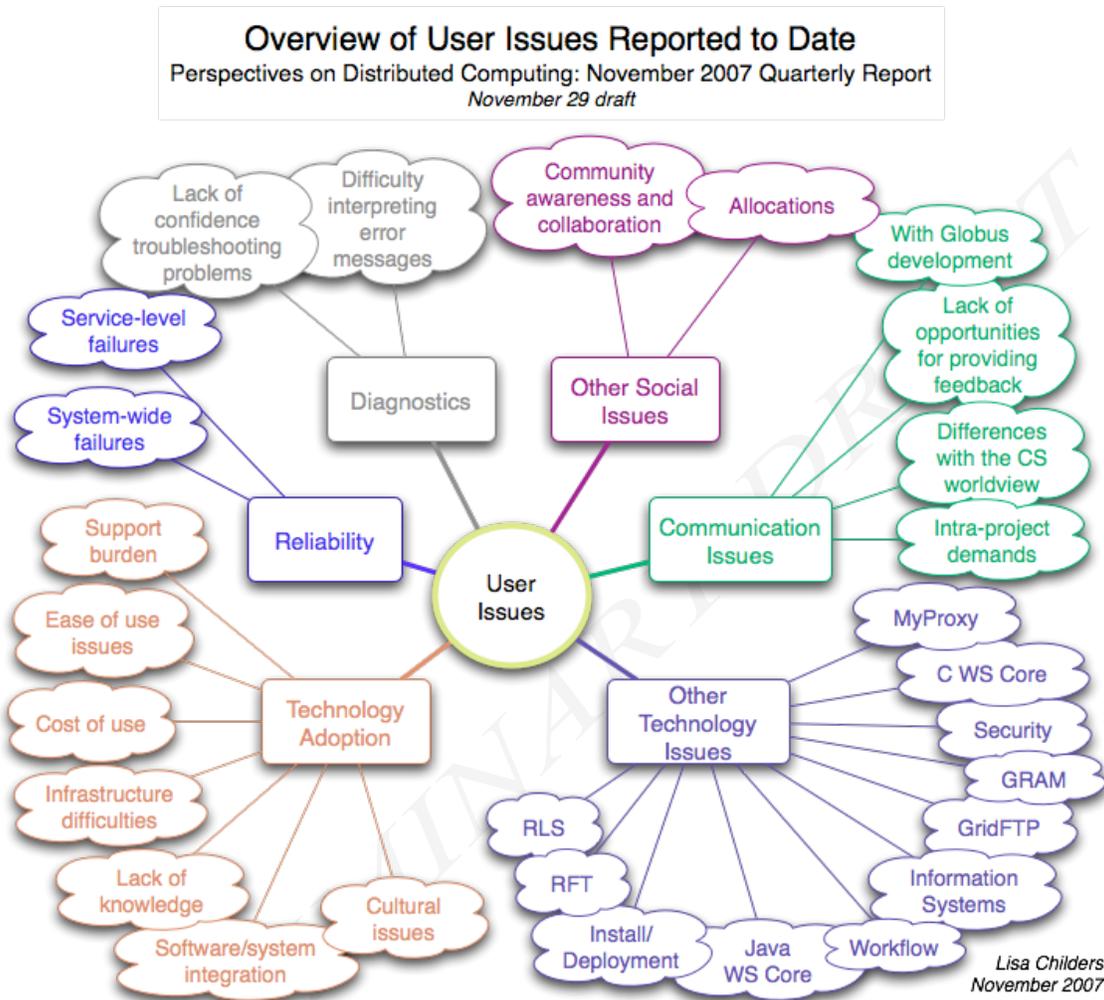


Figure 11: User issues reported to date

[Summary text for this picture will be included in the final report. The observations underlying this summary picture can be found in Appendix C of this report.]

3.6 Satisfaction Points

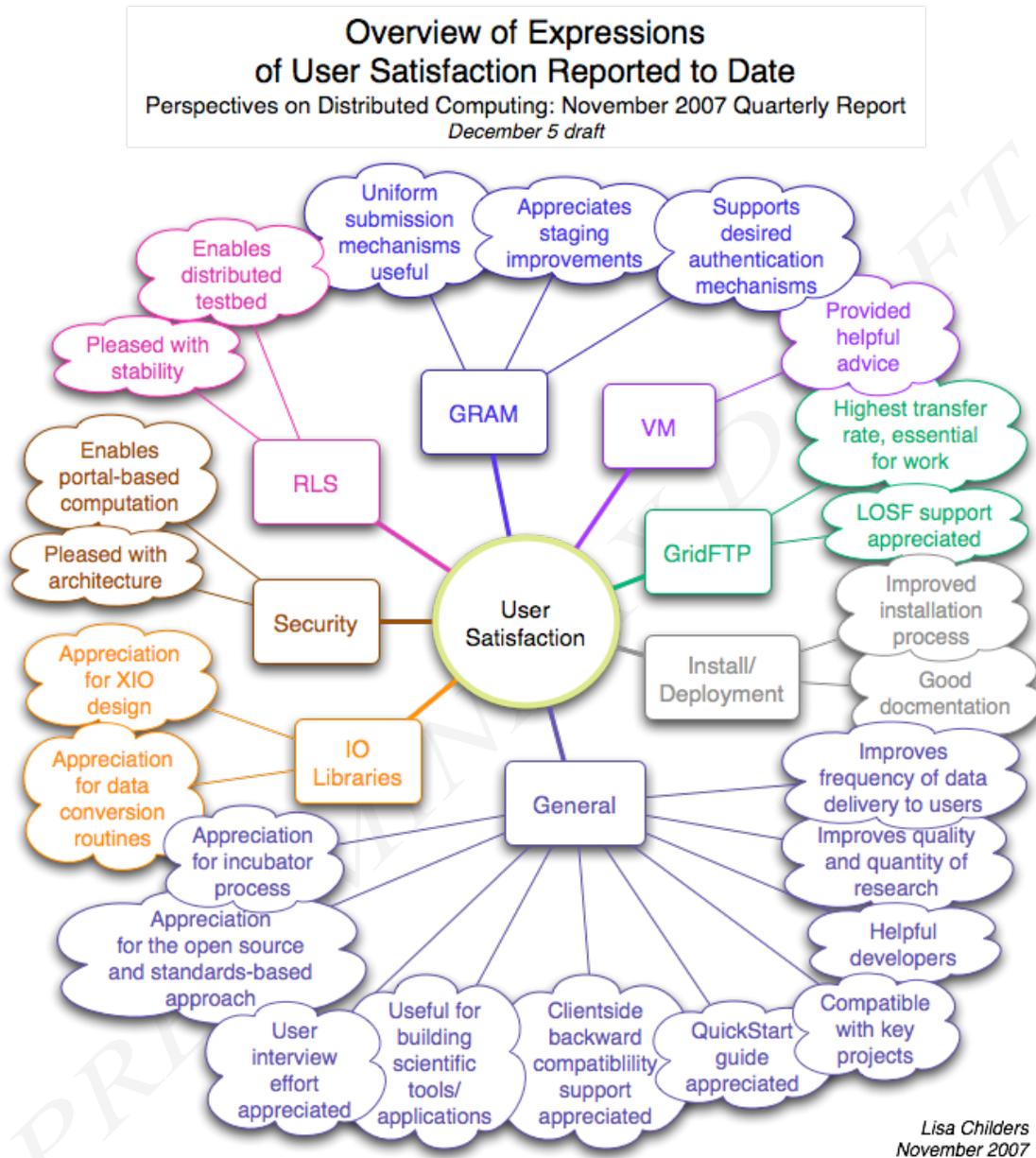


Figure 12: Expressions of satisfaction reported to date

[Summary text for this picture will be included in the final report. The observations underlying this summary picture can be found in Appendix C of this report.]

4 Characterizing Users

To further examine the distributed computing user experience, we first classify users according to their interactions with technology.

4.1 Interactions with Technology

We create six technology interaction categories by crossing three broad interaction types (“develop”, “integrate” and “use”) with two broad technology categories (“domain-specific technology” and “general-purpose HPC technology”). The following definitions are used:

- **Domain-specific technology** includes domain-specific portals, applications, tools, libraries and/or deployed systems.
- **General-purpose HPC technology** includes general-purpose distributed computing clients, services and tools, and/or general-purpose HPC deployments.
- People **develop** technology if they are involved in the creation of new tools, services and/or machine deployments.
- People **integrate** technology if they integrate existing technology into a larger system.
- People **use** technology if they make direct use of existing tools, services and/or machine deployments in support of their work.

Table 2 shows the breakdown of technology interactions as described in the user interviews. (Note that the sequence and timing of interactions are not represented. Also, technology interactions not reported during an interview do not appear in the table.)

Table 2: Technology interactions reported to date

	Domain-Specific Technology			General-Purpose HPC Technology		
	Develop	Integrate	Use	Develop	Integrate	Use
User2	X	X			X	X
User3	X	X			X	X
User4	X		X			X
User5	X	X			X	X
User6	X		X			X
User7		X		X	X	X
User8	X		X			X
User9	X	X			X	X
User10		X		X	X	X
User11		X		X	X	X
User12	X	X			X	X
User13		X		X	X	X
User14				X	X	X
User16		X		X	X	X
User17		X		X	X	X
User20	X	X			X	X
User21	X	X			X	X
User26				X	X	X
User27	X	X			X	X

The table shows that all users interviewed interact with technologies in at least three ways. Nearly all users report the need to integrate technology in order to accomplish their goals, and most report the need to integrate both domain-specific and general-purpose technologies.

4.2 Technology Interaction Clusters

From the data we identify four unique technology interaction patterns, suggesting the existence of four distinct user types, as shown in Table 3.

Table 3: Four user types and their interactions with technology

	Domain-Specific Technology			General-Purpose HPC Technology		
	Develop	Integrate	Use	Develop	Integrate	Use
HPC Scientist	X		X			X
HPC Domain-Specific Developer	X	X			X	X
General-Purpose HPC Infrastructure Provider		X		X	X	X
General-Purpose HPC Technology Developer				X	X	X

Detailed profiles of each of these four user types will be presented in Sections 5-8 of the final report. This preliminary report contains a draft profile of the HPC domain-specific developer (Section 6).

The profiles will provide an evidence-based view of key aspects of the distributed computing user's experience. Arguably, additional user types (not captured by the interviews) may exist in the community. By identifying these first user types, we hope to lay the foundation for further study, and to help improve user support, education and technology development activities.

As the following list shows, a user's self-described job type appears to be a poor indicator of that user's actual technology interactions:

- *People describing themselves as a "Project Lead"*
 - general-purpose HPC infrastructure provider, User16
 - HPC domain-specific developer, User20
 - general-purpose HPC technology developer, User26
- *People describing themselves as a "Scientist"*
 - HPC domain-specific developer, User12
 - general-purpose HPC infrastructure provider, User13
- *People describing themselves as a "Developer/Scientist"*
 - HPC scientist, User4
 - HPC domain-specific developer, User5

5 ▼ User Profile: The HPC Scientist

[To be included in the final report]

PRELIMINARY DRAFT

6 User Profile: The HPC Domain-Specific Developer

Eight of the nineteen users listed in Table 2 fall under the category of HPC domain-specific developer. Table 4 shows a detailed view of the technology interactions mentioned in the interviews of the eight users.

Table 4: HPC domain-specific developer technology interactions reported to date

	Domain-Specific Technology			General-Purpose HPC Technology		
	Develop	Integrate	Use	Develop	Integrate	Use
User2	meteorology portal, workflows	meteorological data, tools and legacy codes			GRAM, GridFTP, GSI, MyProxy, RFT, RLS, MPI, MPICH-G2, TeraGrid	Java CoG kit, PURSE, GridSphere, GPEL, GFac, WebMDS, perl, python, jython, shell scripts, Java, Fortran
User3	analytical framework for neuroscientists	brain image data, Monte Carlo simulations, tools			R, 250 CPU cluster, 4TB storage, TeraGrid	Swift workflow framework, database, perl, shell script, awk, ced
User5	Patient-Centric Authorization Model, Grid Interface Service, Grid Book	medical picture databases, medical images, radiology workstations, DICOM library			GSI, MyProxy, GridFTP, RLS, GridShib, Shibboleth, MDS4	Java WS Core, Java CoG kit, GridShib for GT, Java
User9	infrastructure for analyzing gravitational waves	metadata service, interferometer data, analysis pipelines			GSI, GSI-OpenSSH, MyProxy, RLS, GridFTP, Condor, tape archives, community HPC centers	Python Globus, Simple CA, Java WS Core, C WS Core, python, Java, C
User12	access to electrophysiological and neurophysiological data	magnetic resonance imaging and electroencephalography data, visualization and analytical tools			many processors, large filesystems, data archives	Matlab, R, SPSS, Java, Java++, C++
User20	system for high-throughput analysis of genomes and metabolic reconstructions, algorithms, workflow plans, bioinformatics portal	biology information repositories, biological data, bioinformatics tools			Globus, Condor, national HPC centers, institutional HPC facilities	perl, VDL, FTP, GridFTP
User21	ecological data warehouse, Ecological Metadata Language-compatible data exploration tools	data collection sites linked by community network, ecological data			10s of terabytes datastore, quad core blade servers	LDAP server, Java, PHP, perl, HTTP
User27	Mechanisms for providing and exchanging network data	network data capture feeds, data mining algorithms and tools, files of network data			R, Globus, institutional HPC facility, national HPC centers	Python, perl, C, C++, database, shared filesystem

6.1 User Overview

The HPC domain-specific developer designs and builds systems composed of both domain-specific and general-purpose HPC technology. The user is familiar with many domain and HPC technology concepts, though he may not be an expert in both areas. High-level work includes understanding domain-specific requirements and translating them to a distributed computing context. Detailed work entails integrating existing technology and developing new infrastructure to support the scientific inquiry. System integration is facilitated through the use of general-purpose technologies (as shown in the far right column of Table 4):

- Client-side APIs (used to interact with pre-existing remote services)
- Service development kits (used to build and host domain-specific services)
- Workflow tools
- Scripting languages

HPC domain-specific developers help bridge the worlds of high-performance computing and domain science. These users can be technology trailblazers, having the potential to transform the way the science is conducted in their domain.

6.2 User Goals

The HPC domain-specific developers interviewed for this study describe four broad categories of work goals relating to science, domain-specific computation, domain-specific data and infrastructure development.

6.2.1 Science Goals

Nearly all of the HPC domain-specific developers refer to a specific science goal during their interviews. The science goals are highly domain-dependent:

- support research in the recovery from stroke
- understand the physiological and metabolic processes of various genes
- automatically detect network anomalies
- enable interesting epidemiological questions to be answered
- detect gravitational waves and conduct gravitational wave astronomy
- build a dynamically adaptive weather simulation system
- ensure radiology imaging workflow is flowing

Note that “facilitating the publication of peer-reviewed papers” is reported as a success measure of more than one HPC domain-specific developer, suggesting the user maintains close ties to his domain. The HPC domain-specific developer plays a crucial role in the integration of general-purpose technologies (e.g., a file transfer service or community authorization service) into the domain context.

6.2.2 Computation Goals

The computational goals of the HPC domain-specific developers interviewed fall into two groups. Some pursue high-end computation goals in order to **speed data analyses**. Specific examples include speeding the processing of brain data and minimizing the time spent analyzing huge volumes of genomic data. The second type of computation-related goal is **providing easy access to high-end computation**. Examples include enabling users to run sophisticated meteorological models on high-end resources, and making it easier for data miners to interactively run their algorithms. Note that a given HPC domain-specific developer may pursue none, one or both types of these computational goals.

When addressing a problem like the need to speed the analysis of domain data, the HPC domain-specific developer must understand how concepts like targeted job types¹, mutual authentication² or delegation apply. Even after distributed computing concepts are understood, additional time and expertise are needed to work out the specifics of the conceptual approach (e.g., working out the implementation details for creating a domain-specific virtual organization). The HPC domain-specific developer plays a key role:

- **Translating domain goals to technological concepts:** developing an overall conceptual approach by merging domain-specific goals with distributed computing concepts
- **Translating concepts to practice:** designing and building a concrete technical solution based on an overall conceptual approach

6.2.3 Data Goals

The HPC domain-specific developers describe three types of data-related goals in the interviews. One of these is **providing access to domain data**. Examples include providing access to ecological data that is distributed and heterogenous. Another user works to provide access to results of computationally-intensive analyses. A second type of data-related goal is **aggregating domain data**. One user interviewed works to aggregate distributed datasets as a way of enabling new synthetic products to be created. Another user works to aggregate patient information so the same data can be shared by multiple healthcare providers. The third type of data-related goal mentioned in the interviews is **moving data securely**.

We note that only one domain-specific goal mentioned thus far (moving data securely) is directly addressed by a single Globus service. Other domain-specific goals require more elaborate technical solutions involving multiple components.

We also note that the HPC domain-specific developer is generally concerned with the management, organization and movement of data, as opposed to actually using domain data as part of a scientific inquiry.

¹ <http://www.globus.org/toolkit/docs/4.0/execution/key/>

² <http://www.globus.org/toolkit/docs/4.0/security/key-index.html>

6.2.4 Infrastructure Goals

In addition to domain-specific goals, HPC domain-specific developers reported three types of infrastructure goals. One is the need to **integrate technologies**. Examples of this type include the following:

- Integrating legacy Fortran codes and applications
- Integrating compute systems with atmospheric science models and instruments
- Leveraging tools available in the community
- Accommodating domain device incompatibilities

The “Integrate” columns in Table 4 provide examples of the technologies integrated by the HPC domain-specific developer.

The second type of infrastructure goal is to **build a user interface**. Some HPC domain-specific developers build interfaces to shield their users from complexity or unfamiliar interfaces. Specific examples include building “dashboards” for collaborators, domain-specific portals and other web-based applications.

The third type of infrastructure goal observed in the interview data is to **build a coherent architecture and design**. Examples include the desire to implement a service-oriented architecture, build WSRF-compliant services and develop a systemwide security model. Some users also pursue specific design goals relating to scalability and reliability.

6.3 User Issues

The technology challenges facing HPC domain-specific developers are many and varied. In this section we discuss the types of issues facing these users, as reported in their interviews.

Some HPC domain-specific developers bring novel techniques to their domain by building systems that leverage general-purpose HPC technology. Such users can encounter resistance when introducing their approaches, triggering a **need to overcome sociological barriers**.

We note that this class of HPC domain-specific developer serves not only as a user of general-purpose HPC technology, but as an advocate for it. Users who assume technology support responsibilities as a side effect of their advocacy may in turn rely heavily on general-purpose HPC technology developers for support.

Many HPC domain-specific developers **face a learning curve to transforming science goals into technical solutions**, especially those trained in disciplines other than computer science. Service-level (not to mention multiservice) tutorial material is scarce, and domain-specific case studies are not documented in detail. This situation, combined with the lack of national technology support mechanisms aimed at helping science users, means that many HPC domain-specific developers face this learning curve on their own.

HPC domain-specific developers often **need to address component-level integration issues**. Impediments to integration at the component level can include the following:

- Technology incompatibilities (bad interactions with existing tools, third party libraries, machine architectures, etc.)
- Missing features (insufficient logging controls, lack of security hooks, etc.)
- implementation difficulties
 - Integration requires extensive application-level changes
 - APIs implemented in an unfamiliar language
 - The user interface for the technology is the wrong type (i.e., programatic interface available, but target users don't code)
 - Advanced configurations of the technology are not well understood
 - Error messages of the technology are misleading or vague, making it difficult to implement automatic responses to them

HPC domain-specific developers can **face system-wide integration issues**, which some see as obstacles related to security. One new user observed that it's not easy to integrate non-Globus code into the Grid security fabric. One cannot just drop the code into a deployment but must learn new APIs and a new programming model, eventually modifying their code. It is also painful and complicated to generate certificates for each user and distribute them across resources and is perceived as a high cost to pay for enabling capabilities such as file copying and job submission.

The **need to address system stability issues** is a key focus of some HPC domain-specific developers. One user worries that failures in his system's information services undermine its stability. The system he is building depends on the information in order to function properly, and when the information providers go down, work stops.

Another user, out of a desire for stability, consumes only production releases of general-purpose HPC technology. Yet it is not easy for general-purpose technology developers to provide these users with stable software. Domain-specific data and interaction patterns are difficult for a general-purpose HPC technology developer to emulate. Moreover, in the field the stability of the underlying infrastructure also comes into play.

Another issue facing HPC domain-specific developers is **the difficulty in troubleshooting runtime problems**. The user can encounter undocumented error codes and misleading error messages. One user observes that troubleshooting requires an understanding of the internals of the software. This would seem an impractical requirement, given the user is driven by domain-specific goals and juggling a multiplicity of technology interactions.

Some HPC domain-specific developers work on projects with complete control over the resources they use. Others work on projects that use community or national resources. HPC domain-specific developers who use shared resources must **work within the constraints of deployed infrastructure**. Users reported three different types of problems related to resource sharing.

One type of problem arises when a desired capability or service is not offered on the resource. An example is the user who reports not having access to a GridFTP client and hence uses scp to move data. We note that in some cases the user may have access, but not know where to find the service or how to become authorized to use it.

A second type of problem involves bad interactions with a deployment because of system configuration. For example, if several GridFTP servers used by a project sit behind differently configured firewalls, the HPC domain-specific developer may need to do extra work to ensure successful transfers. Interactions among issues can compound problems. For instance, problems may not be resolved efficiently if the error messages returned by wrongly-configured services are misleading or difficult to interpret.

A third type of problem involves prerequisite software for remote user applications. An HPC domain-specific developer with control over his own resources can install software prereqs in shared space for his users. However, if the HPC domain-specific developer does not control the resources, he cannot assume project-specific prereqs will be available. In this case he must figure out how to get the prereqs running in “user space” on the shared resource and must develop new operational procedures (modifying invocation scripts, user documentation, etc.) to adapt to the changes. This issue raises the barrier to entry for running applications on remote resources.

The final issue uncovered in the interviews with HPC domain-specific developers is their **need to respond to technology changes**. Once he has a working system, the user can be greatly impacted by feature changes and compatibility breakages in the general-purpose HPC technologies. Yet the HPC domain-specific developer may lack the time or technical grounding to understand the implications of published workplans for general-purpose technologies. Effectively engaging these users in the product planning process may require explaining workplans in terms of their impact to this user’s domain-specific goals.

7 ● User Profile: The General-Purpose HPC Infrastructure Provider

[To be included in the final report]

PRELIMINARY DRAFT

8 User Profile: The General-Purpose HPC Technology Developer

[To be included in the final report]

PRELIMINARY DRAFT

9 Recommendations

[To be included in the final report]

PRELIMINARY DRAFT

10 Conclusion

[To be included in the final report]

PRELIMINARY DRAFT

Appendix A Study Methodology

[To be included in the final report]

Overview

Protocol description

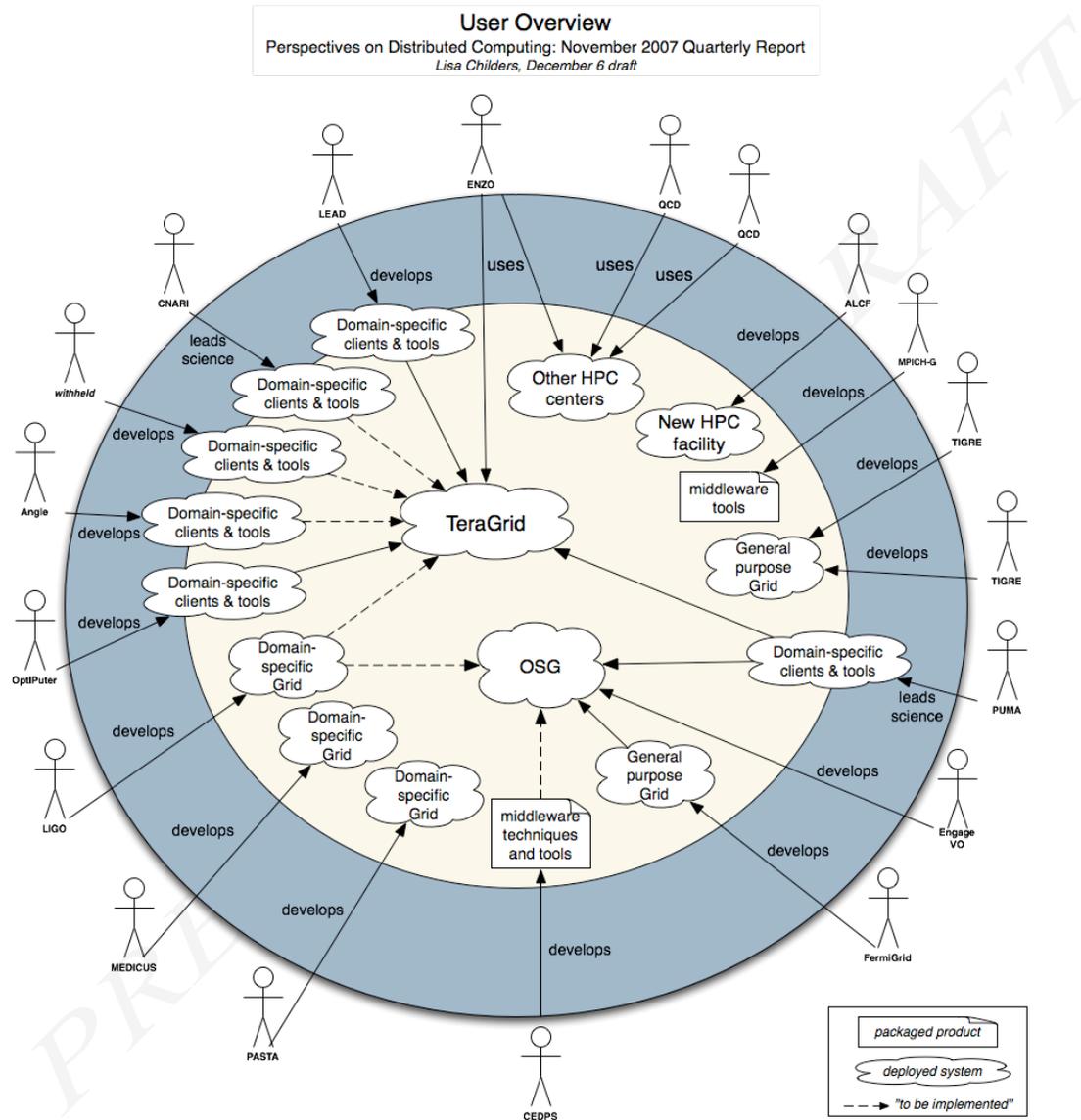
Interview questions

Limitations of the study

PRELIMINARY DRAFT

Appendix B The Interviewees

The observations in this report are based on transcripts produced to date. Figure 13 shows an overview of these users. A high-level description of each user follows the figure.



System Administrator/Research Programmer

September 2007 interview

Featured interview quote: **“The end goal is to automatically detect network anomalies.”**

User27 develops infrastructure enabling the identification of anomalous network behavior. *User27* supports domain scientists – developers of data mining algorithms – by providing easy and efficient access to the data. Currently the infrastructure is running on a local cluster, but plans include offloading processing to distributed computers. *User27* provides a system developer’s view of today’s concerns in porting application code to the Grid.

Project Lead

September 2007 interview

Featured interview quote: **“Our goal is to make it easier to troubleshoot Grid applications.”**

User26 creates techniques and tools to make it easier to troubleshoot Grid middleware applications. The current focus of the project is working with middleware vendors to adopt a common logging format. Future work includes creating a central logging facility and analysis of the data. *User26* not only discusses methods for troubleshooting distributed systems, but also for interacting with potential adopters of technology.

Lead Scientist/System Architect/System Administrator/Developer

September 2007 interview

Featured interview quote: **“We work to enable discovery, access and synthesis of distributed datasets.”**

User21 is building a central data warehouse for storage and synthesis of ecological data. The data are drawn from over twenty sites distributed across the U.S., Antarctica, Tahiti, and Puerto Rico. Future use scenarios of the system include enabling access to distributed computation in order to run simulations on the warehoused data. An overriding goal is to broaden the scope of the science conducted, which is currently at site scale but will be moving to national scale and perhaps global scale. Other project foci include the topics of metadata, provenance, and security.

Project Lead

September 2007 interview

Featured interview quote: **“We can provide our users with fresh data more frequently because of the Grid.”**

User20 is the domain expert driving development of an application that enables biologists to analyze large volumes of genomic data. To facilitate analyses, *User20’s* team warehouses genomes from over twenty biological databases. The application under development integrates these data with a suite of domain-specific tools and computational resources, providing biologists with an interactive system for high-throughput analysis of genomes and metabolic reconstructions. *User20* and his team are a good example of how distributed computing can help science to scale.

Computer Professional/Assistant Group Leader

September 2007 interview

Featured interview quote: **“GRAM2 is kept alive by the need to interoperate with European experiments.”**

User17 develops and maintains a general-purpose institutional Grid to pool and share system resources and to unify several system-level services. *User17* provides a seasoned, practical perspective of several key issues facing today’s middleware consumers – from both the client side and the server side. Key topics for *User17* include how to deal with technology failures, manage deployments comprising thousands of nodes, and maintain interoperability with both domestic and international Grids.

Project Lead

September 2007 interview

Featured interview quote: **“We assume a world where lightpaths can be scheduled between computers.”**

User16 works on an experimental systems project designed to understand how distributed system architectures might look in an environment where bandwidth is unlimited. Working with TeraGrid infrastructure, project members use tens of gigabits of bandwidth in their experiments. *User16* outlines an approach for both investigating and evaluating experimental architectures.

Professor of Computer Science/Principal Investigator

August 2007 interview

Featured interview quote: **“I start with benchmarks and follow up with real applications.”**

User14 develops a middleware parallel computing tool specifically designed to help solve problems that do not fit on a single computational resource. This work leverages the services and libraries that already exist in the distributed computing ecosystem. As such, *User14* describes a method and associated challenges in working with both domain experts and middleware providers.

Scientist

August 2007 interview

Featured interview quote: **“We play a strong bridge role in connecting people with technology.”**

User13 builds general-purpose Grid infrastructure in support of biosciences and medicine, energy exploration, and air quality modeling for the state of Texas. A significant portion of *User13*'s time is spent working with members of the statewide scientific community, helping to bring their applications to the Grid. In addition to insights related to community engagement, *User13* outlines some current issues in security deployment.

Scientist

July 2007 interview

Featured interview quote: **“The right approach is to be highly collaborative with domain specialists.”**

User12 is a domain expert whose team uses database technology to store neurophysiological data for use by scientists. The project team includes Grid experts, who work to leverage computational resources for processing data. *User12*'s application area involves research in the recovery from stroke.

System Designer/Developer

July 2007 interview

Featured interview quote: **“Resource usage within our Virtual Organization is opportunistic.”**

User11 builds infrastructure to lower the barrier to entry for the Open Science Grid; a complementary goal is to bring new users onto OSG. *User11*'s project does not own any OSG resources: it provides users access to resources left unused by the major OSG-based experiments. Much of *User11*'s work involves detecting and fixing problems with OSG infrastructure before users encounter them. *User11* provides a glimpse of both the challenges and opportunities afforded by today's distributed computing tools.

Storage Engineer/Project Lead

July 2007 interview

Featured interview quote: **“Solving problems is easy once you have all the data.”**

User10 is building the storage and I/O systems for a new leadership-class computing facility, due to come online in late 2007. Many of *User10*'s future users run their applications at HPC centers that exist today. The new facility will offer scientists opportunities to scale their work by increasing the size of a mesh or the number of molecules simulated, for example. *User10* describes issues associated with providing data services that will operate at unprecedented scale.

System Architect/Scientist

July 2007 interview

Featured interview quote: “Globus enables more science.”

User9 builds infrastructure for distributing gravitational wave data detected from astrophysical sources to analysts around the world. Having created a data replication Grid that is robust and reliable, the *User9* team is now working to reduce the time needed to configure, maintain, and manage this Grid. Other work includes enabling production data analyses to run across a greater number of resources, with plans to federate into Grids such as OSG or TeraGrid. *User9* provides a view on the challenges scientists face keeping track of data: not only coming off the sensors, but also tracking results from Grid-facilitated analyses.

Professor

June 2007 interview

Featured interview quote: “I am trying to understand where Grid computing adds value.”

User8 works in the area of lattice QCD, a numerical approach to quantum field theory. *User8* runs teraflop year-scale calculations, producing hundreds of files, which must be archived for use as input for later job runs. Curious about whether Globus could help with data movement, *User8* shares valuable insights regarding the challenges of using Globus as an HPC facility user.

System Architect

June 2007 interview

Featured interview quote: “If things don’t work, you need an expert to fix them.”

User7 is building a Grid enabling the state of Texas to provide scientific applications for research and education. A secondary goal of the effort is to help develop the business of the state, in terms of providing greater access to resources and knowledge. *User7’s* team provides high-level support, helping users translate their applications to the Grid. *User7* describes the team’s approach for building the Grid and outlines some of the problems infrastructure providers face in building these systems.

Scientist/Developer/Project Lead

June 2007 interview

Featured interview quote: “The Grid idea is great, but there are barriers to making it work today.”

User6 is an experienced lattice gauge theorist working to understand the interactions of quarks and gluons and applying that understanding to the discovery of new, fundamental parameters of elementary particles. Looking at Globus for help in managing lattice files, *User6* provides important feedback on the barriers facing new users.

Developer/Scientist

June 2007 interview

Featured interview quote: “Performance improved from days to seconds.”

User5 is in the fourth year of building infrastructure enabling radiologists in North America to share medical data in research and clinical settings. The *User5* team is creating higher-level capabilities by integrating domain-specific code with the programmatic interfaces of Globus services. The project is one of the few open source medical applications in existence today, and its successes are serving as a catalyst for change in the field of medical technology.

Developer/Scientist

June 2007 interview

Featured interview quote: “The reason my tasks are so time-consuming is failure.”

User4 maintains a simulation code that runs at the largest computing scale currently available. The focus is on the HPC aspects of the simulation as opposed to the science: fixing algorithmic deficiencies to make the code run at the next largest scale. The application is demanding, not only in terms of CPU usage, but also in terms of producing prodigious amounts of data. Because the application pushes so many system boundaries, *User4* encounters problems not experienced by the “thousands of users chipping away at small jobs.”

Scientist/Developer

June 2007 interview

Featured interview quote: “The Grid is a black box to me.”

User3 is building a framework that enables neuroscientists to store, analyze, and share data. The user is not a Grid computing expert but collaborates with people who are. He depends on these experts to translate his domain-specific needs to the Grid. The group is working toward using TeraGrid resources to scale processing of high-volume domain-specific data.

Science and Technology Liaison/Portal Developer

May 2007 interview

Featured interview quote: “Troubleshooting currently requires knowledge about software internals/”

User2 works with both atmospheric and computer scientists to implement meteorological use cases in a portal environment. *User2* simultaneously assumes the role of a CS user and scientific application developer, working on integration and testing of distributed computing tools and clients. The Globus services used are maintained by the TeraGrid. Familiar with Grid computing concepts, *User2* provides a seasoned, practical view from the client side.

PRELIMINARY DRAFT

Appendix C Summary Data for Section 3

C.1 User Goals

C.1.1 Building domain-specific infrastructure

Disseminating gravitational wave data

- The main goal of the project is to detect gravitational waves and to conduct gravitational wave astronomy. In other words, to learn about our universe through the use of gravitational waves. Diving down a bit, my real project is the DataGrid. And the purpose of the DataGrid is to enable as much science as possible to be conducted using the project data. The data has to be analyzed. It's very computation and data intensive. And the main goal of the project is to build infrastructure (tools, middleware, end-user tools, services and systems) that enable scientists to efficiently analyze the data and conduct their research.
- I'd like to try to get production data analysis running on sites that are external to the DataGrid. So federate into other Grids, such as Open Science Grid and TeraGrid.
- I'm also working on getting some of the data analysis pipelines or workflows to run in more sophisticated ways across multiple computing sites. Right now our users tend to pick a site and go run there. And while they use some Grid tools to do some of the data finding, they don't typically use Grid tools to leverage more resources than are available at a single site. So another of my metrics for the coming year is to try to enable production data analyses that run across multiple DataGrid sites in a continuous way.
- Moving forward, we want to build WSRF-compliant services on top of both Globus Toolkit 4.0 Java and C WS Core.
- We plan to put up really nice dashboards that our collaborators can look at and see the current state of replication throughout our DataGrid.
- The goal I'm really looking for is to have a mean time between failures of three months. That's what I'm targeting from the data replication side.

Weather simulation system

- From the computer science part, we've been trying to make all these legacy Fortran codes and the legacy applications run in a Service Oriented Architecture. And providing users direct access to advanced computational resources from a portal-based environment. So users don't need to learn about how to use a particular cluster or job manager.
- What we are trying to do is integrate computation systems with atmospheric science models and instruments. From the science perspective, we are trying to look for more ways of running the huge computational science models at high resolutions. That's been a big challenge. So running a weather forecast at storm scale and at tornado scale is something we've been trying to do.
- The science goal is building a dynamically adaptive weather simulation system. The engineering goal is to build a Service-Oriented Architecture in support of the science goal. So we are building the Service-Oriented Architecture and Grid middleware ourselves while trying to leverage as much as possible the tools already available in the community.

Framework for neuroscientific data

- Establishing a framework where Neuroscientists, especially people who are involved in brain imaging, can store, analyze and share their data in an effective manner.
- Our project is aimed at using database technology to store data that is electrophysiological or neurophysiological in nature. The data consist of hundreds or thousands of timepoints in a time series. Each timepoint consists of a large amount of data itself, such as a brain image or an electrophysiological recording from different sites on the brain. Using database technology to store the data is a way of allowing much more useful access to the data. Also through interactions with Grid computing experts at the University of Chicago, we found that it's a much better way to

interface with distributed and high-powered computing devices in order to process the data. This is all done in the context of research in the recovery from stroke, which is a disease of brain vessels that is very devastating.

Analyze genomic data

- The main goal of the project is the analysis of large volumes of genomic data. When genomes are sequenced, they are represented as strings of letters, which signify different nucleotides. Once a genome is sequenced you want to know what functions the genes perform and what physiological and metabolic processes the genes are involved in. So starting from just the alphabet soup of the sequence, by the end of the analysis you know:
 - how many genes this organism has
 - what they do
 - how this organism lives (because we're reconstructing double helix properties)
 - does it have any pathogenic or non-pathogenic factors
 - what does it transport in the cell
 - and what does it produceSo pretty much by the end of the analysis, not through experiments but by using pure bioinformatic methods, the biologists know quite a bit about the organism already.
- To do the analysis we take the genomes from a national biology information repository, which is called the National Center for Biotechnological Information [NCBI]. We then analyze a genomic sequence with an array of bioinformatics tools so it will be easier for the researchers to answer the questions that they usually have.

Aggregating and securing medical data

- The main goals are to efficiently and compliantly communicate medical images in a secure fashion so that patient privacy is guaranteed, using existing security mechanisms and other standards-based technology provided by the Globus Toolkit. An example of the vision is that a patient comes into the hospital, and this patient's record is not only existent in the hospital, but also available on the Grid so that other healthcare providers can access the data and also add to it. So that wherever you go as a patient Grid can basically follow you, aggregating all the information that exists for you at various health providers, and collecting them at the point of care. The challenge is to provide a technical solution that can scale to allow a large number of healthcare providers to interact and share images.
- We also work on the security model to make sure the privacy protection is there. This will actually be very important to further explore the medical field. When you have a patient-doctor relationship, you as the patient sign consent that only the doctor who is treating you or the staff of that facility is allowed to see your medical charts. So now Grid enables us to communicate all these medical information wherever we want.
- So one motivating factor is to develop a security model that allows the same doctor-patient relationship being translated onto the Grid, allowing data access, but only if the patient authorizes it. So we are developing what we call a Patient-Centric Authorization Model as a way to approach this. What we want to accomplish in in the project is to show that there's technology available today, such as the very strong security infrastructure in the Globus Toolkit, which can be used in an intelligent way to build a security model for patient authorization and privacy for health data.
- Specific goals within that project are to ensure that the radiology imaging workflow is flowing. That means that there are very different modalities out there that we must take care of, and there's a lot of incompatibilities between the very different devices. Because of that, one current work focus is to make sure that the project can handle all these different vendor-specific imaging devices.
- Now that you have all these images collected, what are the cool questions? Some cool questions, if you have many, many deployments are, "Give me all the computer tomography images of the twenty-year old males who have lung cancer." These are totally relevant questions to ask in the medical domain, but we cannot do that today because the data are not aggregated. The critical thing here is that if you build a Grid which is connected to clinical data you can come up with very

interesting epidemiological questions. "Give me all the cases of a specific disease in a specific area of the country" - and then you can see a big picture.

Building a warehouse for ecological data

- The primary project involves an architecture we call PASTA, which stands for Provenance Aware Synthesis Tracking Architecture. As part of the system there are 26 sites, which are spatially distributed across the continental United States, two in Antarctica, one in Tahiti and one in Puerto Rico. Each of the sites is collecting scientific data. The goal of the architecture is to pull data from them in a seamless way, based on both the metadata records and open access to the actual data file. These data are brought into a centralized data warehouse to enable data discovery, data access and synthesis of distributed datasets within the network.
- Once the data are actually centralized we develop interfaces to enable users to explore the data. I think one term is "exploratory". So we're developing web-based applications that include discovery interfaces, plotting routines, different types of data download mechanisms, and allowing end users to integrate different datasets so they can generate more or less synthetic products on the fly.
- Part of our goal is to make this next leap from science that takes place at the site, to science that takes place at the network level. This would be a national, if not global scale. So the anticipation is once these datasets become available to end users, that simulation and modeling will begin taking place. That's probably on the horizon within the next 1-5 years.

Detect network anomalies

- The end goal is to automatically detect network anomalies. As a first step toward that goal, we want to be able to interactively analyze the data to gain better understanding of it. This will allow us to devise better algorithms that will automatically do that for us. Such anomalies could include:
 - a user transferring large amounts of data
 - the presence of a probe
 - or some kind of an attackAny kind of anomaly, but we're looking at the problem from a behavioral point of view, as opposed to mining actual content.

C.1.2 Running simulations

Understanding cosmological structure formation

- As far as the cosmological part of it goes, we're trying to understand the hierarchy of structure formation on various scales, from the larger scales and the universe, down to galactic scales. This is a tremendous number of orders of magnitude in physical scale - in space and time.
- We hope to be able to account for observations, to determine the cosmological parameters of the universe we're living in. To provide theoretical underpinnings for observations from things like the Hubble space telescope, or the James Webb space telescope, or any of those major projects.
- The big challenge for us is in three-dimensional radiative transfer, which is how light basically interacts with a fluid medium. Our code is being extended right now to incorporate these frontier pieces of physics, that up to now we haven't had enough computer power to include. So, there is a finite list of things we'd want to add. I wouldn't want to be too strict about this point, but we'll probably add most of the physical things we want to add in the next two to three years. And then it will be mainly a question of just how much computer power can we get our hands on. Our goal will be to be able to run problems that continue to match the most powerful resources available in the US, whether they're DOE or NSF.

Elementary particle simulations

- The main goals are to understand Quantum Chromodynamics (QCD), which is one component of the Standard Model of elementary particle physics. This includes:
 - calculating the masses of particles that interact strongly; those particles are made out of quarks and gluons,
 - calculating their decay properties,
 - and we're interested in studying QCD at very high temperatures, and possibly with non-zero chemical potential.
- The goals are to understand the interactions of quarks and gluons, and applying that understanding to the discovery of new, fundamental parameters of elementary particles.

C.1.3 Operating general-purpose resources and infrastructure

- To expose mass storage in a shared way.
- Making the infrastructure all highly redundant so we don't have any one single point of failure.
- To provide a unified authorization and authentication structure.
- To create a structure within our institution so that six or seven interest groups with big pots of computing can share each other's resources. This is by far the bigger focus: to share the resources amongst ourselves, as opposed to sharing resources with people outside our institution.
- To have a unified systems support structure.
- One goal is to provide a unified point of access for inbound Grid jobs, in order to share our institutional resources with the Open Science Grid.
- To demonstrate applicability of Grid technologies to a wide variety of economically interesting and intellectually useful activities in the state.
- To take Grid technology deeper into the academic infrastructure than it has gone so far. Our particular project is targeted at the State of Texas, but we're also working with the Open Science Grid, SURA (the Southeastern Universities Research Association) and several regional organizations.
- The main goals of the initial phase are to
 - have an operational system
 - have an initial set of users (scientists, researchers and educators) using it
 - and to begin to form the collaborations and relationships to keep it going longer term
- The project was composed as a demonstration project in the sense that we have to demonstrate the capabilities in these areas. But we're just now transitioning into creating the conditions for a production-scale Grid.
- To bring new users on to the Open Science Grid.
- To provide stable computing facilities for people to do cutting-edge science. We are not about doing science; we are a resource/infrastructure provider. So our primary goal is to keep the equipment up-and-running, have as stable an environment (i.e., not changing, as few crashes as possible.)
- We're charged with bringing up Grid applications in three targeted application areas: biosciences and medicine, energy exploration, and air quality modeling. These areas were chosen as examples of the application of Grid technologies to economically useful and interesting topics.
- To bring in industrial partners and to help the business of the state as well, in terms of access to more resources and access to new knowledge and collaborations.

C.1.4 Developing general-purpose HPC technology

- The goal is to make it easier to troubleshoot Grid middleware and Grid applications, where troubleshooting doesn't just include failures but also includes performance-related issues. The nature of Grids makes it quite difficult to figure out the source of failures, and much of the underlying middleware lacks the right hooks to make it easy. So the goal of this research project is to figure out what is missing and try to get it added. We are trying to get many pieces of Grid middleware to use a common logging format and to log the right stuff. The hope is for OSG to report a noticeable drop in number of failed jobs. Also to report a decrease in the amount of time it takes to track down problems.

- We are creating a tool called MPICH-G2, which is a Grid MPI, that can be used to solve computational problems that cannot otherwise be solved. Every time that we have solved a problem that no one was able to do before, we are very happy because MPICH-G2 is enabling technology. That's our goal. And we keep pushing that envelope farther and farther out - as far as we can. I also have a personal interest in trying to make this stuff work as a way of improving usability.

C.1.5 Building experimental systems

- The focus of the project is to study what happens to distributed computing in an environment where there is no bandwidth limitation. It assumes a world in which people can schedule dedicated lightpaths between distributed computers, on demand, in the same way that they would schedule supercomputer clusters. We want to understand how those assumptions change application architectures, as well as change application users' perceptions of high-speed networking:
 - What are the middleware capabilities that need to be developed that are still missing?
 - What will the endpoints look like that connect to these high performance services? Will they be desktop computers? Will they be browsers?

C.2 Methods for Developing Products

C.2.1 Requirements gathering

- In the past I have been guilty of not doing enough design and not spending enough time trying to think clearly about what the usage scenarios are, and how the infrastructure should really work. We were so pressed to have anything working that we just threw a lot of stuff out there. We would try to iterate quickly but we'd end up usually tying ourselves in knots and going in directions that we couldn't support. So we're trying to be more deliberate about designing and thinking ahead, without going overboard, in terms of how the pieces fit together. I'd say that's going to be a larger component of what we do now. Going out and talking to the different middleware providers to understand what their roadmaps are, so we can try to get an insight into what things are going to look like one year, two years - even three years from now.
- Requirements were gathered mostly from sysadmins, I think, because they're the ones who are often are in charge of troubleshooting. We certainly involved some applications developers, some middleware developers... a little bit of everything. But probably the majority were sysadmin types.
- The science people come up with a use case. Then we drill down to figure out what components are needed to enable them to do their weather forecast. For example, we might determine that we need to search for X data, and then run Y computational model. So we take a data search tool and look at it to see if it needs to be modified for the use case. Then we apply it to a computation and run it on TeraGrid, for example, run it with Globus and see if we are getting the feedback. We then show it to the user and see if he's happy with the status and the monitoring and what he's getting back. If not, we'll go back and work with the developers to improve the user-friendliness of the tools.
- The way our collaborations work - this wasn't by design, it just turned out this way - we'll sit in the room with the mathematicians, or the physicists, or the civil engineers. These domain experts will describe their problem to us in layman's terms, e.g.: "We are interested in simulating the flow of blood through the human arterial system. The interesting part to us is where the arteries split. Etc." Then they will drill down a little bit more and talk about the application they've already written to solve problems. So far, they've always had MPI applications; it will probably be this way for the foreseeable future. As I listen to the MPI application description and I gain an understanding of the problem and their approach, my experience prompts me to ask them specific questions. Like, "Are you using MPI's collective operations here? And are using the asynchronous or the non-blocking point-to-point Send/Receives here?" Because I know where the choke points are when you're running Grid applications. Also, "Why do you need more than one machine to solve this problem? What's your limit?" So at the end of the day, what happens is that I begin to understand their problem and the scale at which they're trying to solve it. I also get an idea of how they're trying to solve the problem through their application. They walk away with knowledge of the

- potential choke points when you're trying to run on a Grid. What has always happened is that we find places both in the application and in MPICH-G2 that need to be modified in order to make the application run well on a Grid. So the application gets a modification that's good for the application. But what's exciting for me is that MPICH-G2 gets modifications, which not only helps the specific application, but also tends to be of general benefit to all applications. So that's a real big win for us because it's this type of feedback that helps MPICH-G2 to advance.
- The way we start is a few of us will be involved in thinking about the next physics problem we want to attack. And that will involve a lot of discussions before we commit to doing something - to design what it is that we want to do from a physics point of view.
 - We assembled an inter-disciplinary group in a room (Grid experts, medical informatics experts, software engineering people) and drafted a set of requirements. We identified the need for data transport, security, DICOM protocol translation into Grid protocols, and so on. After we had our requirements, the second step was to identify what was already available. And that was a really interesting step for me, because I learned that much of the basic functionality, such as data transport, is already included in the Globus Toolkit.
 - We designed our database schema by discussing what types of events or types of queries the students will utilize to select the datasets. Based on their particular needs, the database schema was defined and indices created to support the types of queries that will be taking place. Then we start populating it with the data. So the students help determine what queries the database needs to support.
 - We get some requirements from people in the community who've installed our system. Both through mailing lists, but also from workshops or events that we've run, such as iGrid. Then there are meetings sponsored by organizations like the GLIF (Global Lambda Integrated Facility). So it's really a tightly knit community where once they start installing the software, they don't go away.
 - We started working with experts from the School of Information at the University of Michigan who have done a lot of prior research in group work. These are the people involved heavily with the Access Grid. They worked with us and we taught them how to build tiled displays and how to use our software. Then they worked with their research community within the University of Michigan to deploy some of these displays. The experts also began observing how people worked with the prototype at the University of Michigan, and they started giving us feedback.
 - What the applications people know is that there is a new type of hardware that's coming out into the world. They know the new hardware is characterized by many, many CPUs per board - much more than there used to be. And nobody knows how to harness that problem - not the application guys, nor the guys like me who write the middleware code. Addressing the problem will require an ongoing conversation. It's not going to be the application folks coming to middleware folks like me and saying this is what we want. And it's not going to be the middleware folks like me walking up to the application guys and saying, "This is what we're giving you, now change your application." It will be a dialog. In that dialog, both sides will come up with ideas and will find a place to meet in the middle. It's not going to be just between my group and the rest of the world. It will be all the middleware people (and there's a lot of us) working with all the applications. And everyone's going to be talking to everyone up and down and side to side.
 - If any modification is needed we put them in our bugzilla and consider them when we are developing and further enhancing the software.
 - The process for our project begins in January or so when we sit around and decide what it is that we want to do next year (in terms of physics.) Then we try to work out exactly what that will cost in terms of CPU hours and so on, and which computers can be used.
 - the way we work is we assess the landscape, and we see what we can do with what's out there. So as an example of that, eight years ago we saw MPICH and Globus and thought, "We can do something interesting here." Subsequently things like GridFTP, Reliable Blast UDP, UDT... fast data transport technology came up and we said, "Hmm. These are good things. What can we do with these?" Recently, these threads packages are coming up, we're looking at these and we're saying "Hmm. What can we do with these?" So it's not the case that need forces the exploration of a tool. The way it has been so far is as we become aware of tools, we think about how we can use them.

C.2.2 Architectural approaches

- Either you take what's there and try to put these things together to integrate them, or you say, "Ok, I don't like what's available. I will re-invent the wheel with the flavor I need." And depending on which paths you go, you have different obstacles. Obviously we chose the integration approach, and so any problems we have would be related to the existing software. But in the case of our technology we did pretty well, because we only need to have some very basic functions and the Globus Toolkit provides that.
- I'd also like to set up some WSRF customized services that would address very specific project workflows, so that the users are actually talking to this customized service. Then we can use the service to hide from the user some of the details about these other tools. So we can have a project "face" that we present to the users without exposing them to the more generic tools like Pegasus, Condor-C, GRAM, etc. So we'll be building an infrastructure layer on top of the middleware so we can tie things up nicely for our users.
- I'm a strong believer in microbenchmarks. People come at it different ways. Some people do an implementation and then use the application to see how things perform better. And eventually, we do that too. But our method is to drill all the way down to the basest case, and see what we can do there. A solution at the application level, which is the end game, has no chance of seeing the light of day if you don't have a solid solution at the base. If you try and solve it from the top down - from the application layer down - oftentimes things get in the way or are distracting. So I prefer to use a bottom-up approach.
- Our project portal sits on top of Globus middleware, such as the CoG kit, GRAM and GridFTP. So:
 - the users authenticate with the portal (the portal uses the MyProxy-based PURSe credential repository)
 - then the portal as such authenticates with the Grid services using GSI-based authentication mechanisms.

Our portal is a one-stop place to access all the project services. The portal is built using service oriented architecture principles: we have application services and other persistent services, which in turn interact with the actual grid services on the TeraGrid. So as such, the portal is a lightweight portal framework.

- There are very specific issues in implementations of MPI that only surface when you're trying to couple two or more machines together, and that's where we spend all of our time thinking. We have such an implementation; the current release is called MPICH-G2. With this implementation, we seek scientists or engineers with HPC applications to collaborate with. It's not a strict requirement, but almost all of our collaborators already have an MPI application that they work with. They could write one from scratch - we just haven't encountered one yet. The two requirements for us to engage with these people are:
 - 1) they must be domain experts in the application area
 - 2) the problem addressed by the application is one that cannot be solved on any single computational resource.

So if you can run just fine on a single machine, well, then you should continue to run just fine on your single machine and good luck to you there. But if you can't - if your problem is bigger and literally can't fit on the largest machine you can get your hands on, then that's a good reason to come to us. We'll work with you to try and port your MPI application to our MPI implementation and deal with the issues.

- We look at any prior approaches that might be similar in concept, and think how that approach fits the way end users actually work (as opposed to just solving a technical problem.) And then if we determine there is still a gap, we think about what kind of architecture would also meet the requirements of the end user.
- When we think about base interactions, we're in the same position as people who are trying to develop like an instruction set for a brand-new computer. What should the instruction set be? What are the users likely to do? Well, they're probably going to want to move information from the memory to a register, so we're going to need some memory-to-register commands. They're probably going to want to add things and subtract things and divide things, so we'll add some of those constructions. We do the same thing. What are the basic level of operations that are very

likely to occur on those nodes? So the two sides of the same coin are the ones I just described. The model we think they're going to use is they will run a single MPI process on one node, and then use threads within the single MPI process to get things executing on different CPUs on that board. But eventually, we're going to need to collapse or coalesce the information and ship it off to the neighbor next door or across the way. And vice versa: when we get something from our MPI neighbor running on the computer next door, we have to efficiently distribute that. So that's an example guess for a low level, primitive operations or type of things that these applications will do.

- With respect to the engineering aspects of our work, a fundamental principle of our implementation is that we try to keep every module as focused and specific as possible. So if there's any redundancy with something that already exists we try to avoid that. So for instance, our DICOM Grid Interface Service is not doing anything other than receiving DICOM images, compressing and caching the series, and then sending it out with GridFTP. At that point we don't want to know what's in GridFTP - we just use the public interface. And the same way with the security model. We try to use the vanilla service and not add anything fancy.
- That's exactly how our system was designed. Not designed to be exposing the PACS as a resource, but to be an additional component that adds to the PACS system by providing offsite storage of the images. Once you have the images offsite you can use them actively. So you decouple the local hospital workflows from the Grid-based workflows. Obviously the thing that you have is the redundancy of the images: one instance of the image exists in the PACS system and the other exists outside on the Grid. That's how you want it, because if the PACS system dies you can recover it from the Grid. So this is a good buy-in concept, and is the only way to bring in radiology imaging, as well as all medical data in general. This is because of the fact that Grid operations cannot interfere with the local hospital activities.
- The extent to which we care about data is all wrapped in the MPI standard. Our concerns and our goal are to make it as easy as possible for the application to ship the data from point A to point B. Not to get it off of a disk - that's someone else's job. But once it's memory-resident in the application and they want to send it from one MPI process to another, we work to make that as easy as possible for them. So we tell them to call an MPI function, MPI Send for example, and we will take it from there and move the data as fast as possible. Furthermore, if you're moving it from one machine to the other where the architectures are incompatible (like from a big endian machine to a little endian machine) we'll assume the responsibility of doing the data transformation for them.
- We are planning to do more development of the analysis codes so that they can run concurrently with the simulation, because this model simply won't work at petascale. It will not be physically possible anymore to move the data. It may not even be physically possible to store the data, even temporarily for more than a few days given the rate at which it would be produced.

C.2.2 Distribution methods

- A typical way, obviously, when a project is finished you publish the results in a scientific journal. Before the project is finished when you're in the process of understanding your data and you need to share data, there are two ways:
 - 1) You send people a file and maybe the workflow script that generated the file, or some hints about what you're doing to this datafile and how you're analyzing it. This is what we used to do about a year or two ago.
 - 2) The newer way that is tied to the project is basically storing the results of certain stages of the workflow in databases which are accessible over the internet. Then accessing the database with SQL queries.
- All the simulations are made available on the portal for people to use.
- Our code has been open for a long time; it's called the MILC code. And so we have an ensemble of applications that people can use to analyze our configurations, or to create configurations on their own. It has been freely available for years. And people will develop a new application, and there may be a time lag between when the code for that application is produced and it goes into the suite of applications that we make freely available to the world.

- Our software is distributed by the MPICH group. They have an enormous release mechanism and mailing lists and Web pages and we just ride their coattails. We are distributed as one of the modules that they release.
- So we have a number of homegrown open source tools that we make publicly available. Almost all the tools are being supported through the <http://ecoinformatics.org> website; most of this work will be contributed back to ecoinformatics.org.
- The code and documentation for the software we're developing is completely available on the web.
- The configuration files are huge and people don't tend to collect them. So a collection of 500 four gigabyte configuration files is not something you want to bring to your computer because it's only useful to have it where the analysis is done. So these tend to stay on these central, large computers where we have backup/tape systems, large disks and multiple CPUs for performing analyses on the snapshots.
- There is also a newer system called the International Lattice Data Grid (ILDG), which is supposed to be a searchable database of the configurations people have made available to the international community. And I believe we have some ensembles there. But since I'm the producer of the configurations, I spend less time thinking about what's available that way than how we store them. So I archive some of them, and then we archive files at NCSA and at SDSC. They're organized in a hierarchical fashion. So you should be able to see what's available with a little bit of searching around, if you have access to the mass storage system. So internally you can do that; externally you'd have to rely on the Gauge Connection or NERSC. Or email: "Hey! Do you have this available?"
- We document our algorithms and workflows in papers - mostly papers. And also they are open source, so people can download them.
- We have a freely available code database called the MILC code served up via the web. There's also other software infrastructure provided by the US Lattice QCD Infrastructure project, which we usually call the SciDAC project because that's who funds it and we know we're always talking about Lattice Gauge Theory. So if you were to go to <http://www.lqcd.org/> you would probably find that type of software infrastructure involved. There are a bunch of libraries which people can use as the basis of code. So MILC is kind of a higher-level application, and there's a lower-level of code that goes by the name of QLA, QMP, QDP, and a few other things all of which start with "Q". So there is a big effort to provide community software within the field of Lattice Gauge Theory. And there are at least two other groups outside of MILC who are providing application-level code.
- We have an open CVS server.
- People know we have certain ensembles of configurations available. A number of them are made available through something called the Gauge Connection [<http://qcd.nersc.gov>], which is a service of NERSC. So we will deposit configurations that we are ready to share with the world at this NERSC repository.
- The other deployment effort we've been working on is with UCSD. They have embedded this capability into the Rocks distribution. Part of the reason for this is we want to minimize the handholding required for user installs. Rocks has been credited with the ability to manage clusters. So we decided that it would be valuable to put our software into the Rocks distribution. The idea was that people could just buy a cluster, build a cluster, then plug in Rocks and have it rock
- We archive our critical results. Such results include the lattice files - any observables we generate with expensive computation. These files are then stored in mass storage somewhere on tape. So if someone needs to verify or check a result that we've obtained, we can go back to the tape, look at it and repeat some of the calculations. So that's documentation of what we've done. Then of course we publish our results, which is the dissemination of the results of our calculations. So we publish papers that are a distillation of our results, without the intermediate steps

C.2.2 Testing methods

- I'm not comfortable presenting results unless I actually:
 - start with the microbenchmarks
 - present those in the paper
 - and follow it up with a real application that uses the same techniques

I like to demonstrate that the theory we believe in not only bears fruit by empirically observing that it's good at the microbenchmark level, but it also that it works all the way up the top. Because there's some room between the microbenchmarks and the application, and things can go wrong that we don't foresee. So when I read papers that don't have an application, the question that pops into my head is, "This is a fine start, but how do you know that this is going to translate all the way up?" So I have not yet published anything that doesn't include the endgame as well.

- Testing the code: to make sure that after upgrading it still produces the same results on the same problems; to confirm that we haven't broken anything. There's some amount of repetition there because we test it on many different platforms.
- There are a couple of different levels of testing: One is validating the code to ensure that it is computing what we think it's computing. We have ways of setting parameters so that the codes calculate things that we can calculate by hand. We have alternative approaches that we can compare, where we calculate with one algorithm and then calculate with another. There are a whole variety of different techniques that we use to make sure that we're calculating what we think we're calculating. We adjust parameters to what the effects are of some of the approximations that we make, to make sure that things are properly converged. So there are quite a variety of ways to check that the code is doing what it is supposed to. The other level of testing is validating the theory itself, and the approximations we've made, by comparing our results with experimental results. As far as validating the theory and the approximations inherent in the approach (such as setting the coarseness of the mesh on we're calculating), we calculate quantities that can be measured experimentally. Sometimes the experimental results have already been obtained. Lately the results have yet to be obtained, but will be very soon. So we're actually making predictions that are being confirmed. So far we've been doing pretty well.
- We're putting up an OSG integration testbed site so we can submit fake jobs to it (and maybe even real jobs to it) to get some experience with some real logs. It's very, very, very close to being up. We haven't quite got to this stage yet, but I assume the Globus developers will point us at the right CVS branch for the logging code, and we'll do a checkout and a build.
- before applying the tools to the real models we apply them to simulated systems to test things like scalability.

C.2.2 Education/outreach/training methods

- As far as introducing the database to new students, I meet with them and explain the schema to them. I explain what kind of things we can officially support, what kind of queries can be run very rapidly without overloading the server, what to avoid, etc. We go over the documentation of how the database was created and structured. As far as file storage, there's really nothing to document. The clusters and events that are stored in the database point directly to a file system, a path, and a file representing that set of data. So once students pull it out of the database, the students can access it directly. The captured data files have a specific format, so I explain to students how to work the program that extracts the raw data. The raw data files are converted with this program into an easy-to-use, comma separated value format. This is very approachable to students for parsing, working with statistical tools like R, reading into a spreadsheet or into their own program for processing.
- We don't directly help people instrument their code. We point them at our document and they change their own code. It's usually a somewhat iterative process so far. People will send us their new sample logs and we'll critique them, telling them, "No, you need to tweak this, this, and this." And sometimes, "You're forgetting about this, this, and this." If you put a 20-page document in front of programmers they just skim it, and then they go off and do it. So that seems to be the best way to do it, as opposed to us trying to figure out their code, or expecting them to read our document closely and get it right the first time. We say, "Here, go try to do this. We'll tell you what you got right and what you got wrong." It's just usually two or three iterations, and then it looks okay. At least until we actually deploy it and then discover, "Oh, we forgot about this thing." There certainly will be stuff that we won't know is missing until we get it deployed in real systems.
- We don't run the simulations. They always do, with us sitting side-by-side (either literally or metaphorically.) So the way it works is we'll meet with them, we'll talk about how things are, we'll

- get them hooked up with MPICH-G2 so they can run a simple application, then we'll decide what works has to be done both in their application and in MPICH-G2. We both go off and work on things, and then we re-synch, "I'm done. Are you done? Yeah, I'm done. Okay. Go ahead and run the application and let's see how the results look." And then we move from there.
- We participate in the TeraGrid Gateway and other calls, and tell other guys what's available. And if anyone is interested, or if anyone is reading our papers and coming to us, we completely help them use those software and toolkits.
 - If a new partnership were to form, I would provide them with a tool to run on a computer that's capable of observing traffic entering and leaving the partner's network. I provide them with the software and give them instructions on how to run it. Then they run it, and everything gets sent to us, and we do our thing.
 - Once we create the tool and deploy it in the portal system, we create a screencast tutorial [the project uses Wink software to create their screencasts, <http://www.debugmode.com/wink/>] to demonstrate how to interact with the tool. Then we put a link to the tutorial on the our portal's help pages; by taking the tutorial the user can figure out the steps he needs to follow to use the new tool.
 - The approach we've taken is kind of the soft approach of learning by doing. For the sites, if they see "oh, this is working", or the security model is gaining trust, then you build confidence. After building confidence you can go to the next step.
 - You have to find a balance and ways to accommodate the hospital's view. The way to do that is to convince them that these are standards-based security methods, there are large deployments, there are government, industry and academia examples of using the technology. And then finally you convince people to give it a try, and see if it's possible to open ports, to enable that communication.

C.2.2 Other development methods

- As far as acquiring new capabilities: sometimes we write them, and sometimes community members contribute. Like for example we have a group in Korea who wanted to put in support for HD video streaming, and so they wrote something to do that. Then another group in Amsterdam wanted HD streaming with a different piece of hardware, and they contributed by writing a module for that.
- Because astrophysicists have the advantage of knowing our governing equations, unlike fields like biology where they're not really sure what they're looking for. So we can always adapt very quickly to the largest computer hardware available. And we don't see any end in sight yet, really, in terms of our computational requirements. If you could provide us a usable computer today that was a million times faster, we would just simply use it.
- Hopefully once it is pushed out in the next Globus release and people start getting some experience with this type of logging they'll say, "Wow, this is really helpful. This is really useful." And the idea will catch on. We figure we won't get it right the first time - there will be stuff that's missing. It will be a somewhat iterative process to get the exact right logging information in there. To help with that we're putting up an OSG node ourselves here at my home institution as part of what OSG calls their integration test bed. We hope to deploy a version of Globus with the new logging style on our OSG site and get some experience with the log files before the official Globus release.
- In the past we were allowed more flexibility because everything was so new and shiny. People understood that we were solving problems that no one had solved before. So we had a little extra slack and were given extra forgiveness for building things that weren't maybe as stable as one would like. But we've sort of outgrown the toddler stage now, and we're looked at as adolescents. We had better get our act together and start building production quality software that can be stood up for months at a time, scales well, has lots of documentation and is neatly wrapped and easy to install.
- Our method for defining best logging practices has been to put together a document based on years of our own experience plus talking to a lot of people out there. We went to OSG meetings and EGEE meetings and talked to many people to get feedback. We presented the document at OGF, and people seemed to think it's a good idea.

- The University of Michigan people constantly gave us feedback on user interaction issues. So as a result we changed a variety of interfaces, made the system as a whole more stable, which was important :). We were feeding updates back to them, in essentially real-time. They were getting feedback in real-time and they were one of the fastest absorbers of our updates. We started setting up an RSS feed, so every time there was a new software update people who were interested could get the latest version.
- The hardware requirements constantly change. Equipment gets cheaper and new stuff arrives, and you think, "Oh, this could be cool." But the test deployments don't change as rapidly as the software. I think the software is about six months to a year ahead of the deployed versions. It's only in rare cases that people grab the latest version, like the folks in Michigan who want to immediately try stuff out in the community. It's nice for us because they function essentially as a lab for us. So we have both a development and a stable software release stream. The stable releases are more regular, like every six months or every year. For the research version, they tend to be irregular. Like anytime there's some major improvement that we'd like to get out, we make that available. Folks can just go to the subversion server and grab the latest one if they wanted to. We don't expect the average Joe Scientist to ask their system administrator to go and build it. We expect them to go to the Rocks version.
- These days anytime before you go off and develop something new you always Google to see if somebody else has done it first.
- We also track usage of the tools with monitoring software, and use bugzilla for tracking bugs and feature additions.
- We apply for computer time at the centers, and the storage resources are made available to people who have accounts there. Based on the time that was granted to the project, on a yearly basis the centers allocate some percentage of their available compute resources to us. To use the allocation one typically submits a job, and it is put into a queue and scheduled according to the local policies of the center. It will run whenever the resource is available.
- We develop our own software here. We try to reuse tools that are available, that are functional and we can leverage. Unequivocally we try to leverage open source software because funding is hard to come by. We can't spend a lot of money on commercial software apps and they tend to be very expensive.
- We use a modified version of the Rational Unified Process. Doing all the paperwork that associated with the whole iterative to development cycle - all the documentation and stuff like that. Although it's very useful it's also very time consuming and boring. Team members use these documents for directing their work. People outside of the project use it as proof that we're actually doing something. So it's a good documentation trail.
- We tend to prototype things. I might want to look at the details of RFT, for example, and see if it really has the queuing structures or behavior that I need. Or if I am worried about scaling, I can easily deploy an RFT and I can slam it, and see where it falls down for me under the types of transfers I expect it to manage.
- The interest of our project is not to take over the whole domain, but rather to serve as a seed. Demonstrating what can be done with radiology data will hopefully show what can be done for pathology data and for neurology data, and other entities. We don't have expertise in those fields, but would like people to engage with us and bring their expertise to the table. And that's how we look at the project in general. This is not a typical medical technology development model, but we would like to have contributors engaging in this open-source community effort, bringing their expertise to further build out the system.
- When I started here, we were doing problems about 4,000 times smaller than what we can do now. And every factor of two or so in the scale of the computing system from where they were running (~64 CPUs) up to present (4,000 CPUs) has exposed some new algorithmic deficiency which I've had to fix. So we work by extending what we do. We work by growing up the scale of the problem, which is constrained by the scale of the computer we have access to (and funding, of course.)
- When we're studying network performance, for example, we will rarely, if ever, work with the application code directly. We'll work with microbenchmarking ping-ponging applications and things like that, which are completely contrived applications. You can consider them like lab equipment in a laboratory, like a test. So that's our method, always working from the bottom up.

C.3 Methods for Providing Infrastructure

C.3.1 Application integration

- A big part of the work is spent working with OSG, other VOs and resource owners. We verify sites, and as issues come up we interact with the OSG ticketing system. We act as a middleman between OSG and our users to make sure that they don't have to deal with all the day-to-day issues that come up.
- As technology has improved over the last twenty-five or thirty years, it has become clear that people develop software that enables:
 - the automatic division of jobs into sub-parts (so that it does not have to be done manually)
 - the allocation among lots and lots and lots of processors (so you don't have processors waiting for each other to complete partial tasks before completing what they have to do)This has made it possible to take voluminous sets of data (brain images from MRI scanners or electrographic time series from electroencephalography) and divide them up into little pieces so that the processing can be done faster. This becomes very important and valuable when you have procedures that are highly iterative, and where the different components can be performed without a significant amount of interactions. And so we're taking advantage of that through Grid computing. I didn't realize that this would be available to us now. But through the collaborations with the Grid experts at the University of Chicago, it's become clear that we can serve as a test case for certain kinds of computing that are both interesting to our collaborators and make some of our tasks much more soluble. Particularly those tasks that are recursive, iterative, and other tasks on datasets with spatial components that are non-interacting.
- Building different workflows for different categories of users. This is part of my role as a workflow developer. So I do that a lot - deployments and workflows.
- For most of the runs we're doing, if outputs are created then I'm done. I just give them to the user and they tell me if it was a good run or not.
- I sometimes might have to modify the extraction program to add additional fields they're interested in that, while present in the PCAP in some form, need to be computed (such as a running average.) So I'll add support for these features in the utility programs that the students use to extract the actual fields that they're interested in. Handling this class of changes made up the bulk of my "ease of use" work when the project started.
- I take these tools and integrate them with the applications. I do integration and testing of all the different services and tools. So the CS people develop the tools and I apply them. For the CS people I am the user, and for the science people I act as a developer and answer their questions on the CS people's behalf.
- Most of the models that are running in the VO are simulations, but it's not that often that I get to work on the simulation code itself. Most of the time to me it just looks like an executable with a set of inputs and outputs. I work with it on the level of a process that I need to support, although I need to understand the model enough to know what the inputs are, and what the outputs should look like.
- One user I worked with recently, for example, had five hundred thousand input files that needed processing. So a lot of the work there involved setting things up and making sure the jobs were picking the right inputs. In other cases you have pre-processing of inputs, such as with our weather models. Dealing with that is a little messier, where you actually have to understand a little bit more about the model. Making sure the inputs are correct by, for example verifying that they fall into the correct time period.
- Science work is really often about workflows. You have a whole sequence of operations you want to do to the data - it isn't just one program. Maybe in fact it involves analysis plus simulation, or maybe it's simulation followed by analysis with a tool that would also be applicable to analysis of real data, or maybe it's pure simulation. But quite often there are a number of steps that need to be carried out. Scientists build elaborate, and I have to say fairly rickety, methods of doing these things, sometimes involving a lot of hand operations on the terminal to do their data processing. So we always have to make a judgment call how much of the workflow to try to put into tools designed for workflows.

- So I provide a tool that extracts this data from a PCAP file (which is the capture format the data comes in to us.) As new fields need to be extracted or computed based on the PCAP data, I update the tool and provide the new output format to the students.
- Sometimes simply connecting the researcher with an existing virtual organization can be a big benefit. Most of the time the scientists and researchers we talk with don't know anything about Virtual Organizations. And quite often their work doesn't really map well into that paradigm. So we play a strong bridge role in connecting people with technologies in both directions.
- The entire process of how:
 - the file is captured
 - the file is stored in the capture format
 - the file is sent to us
 - the file is archived on our system
 - the file is preprocessed
 - the references to the new pieces of data are stored in the database
 The entire process is basically abstracted with a few utility programs the students use to extract the contents.
- The old model of engagement was to go to the scientists to find out what they need and then implementing it. We find that approach to be insufficient by far in the modern interconnected world. We spend a lot of our time investigating, studying the needs derived from interviews and work with those people. But then we also go out into the broader context and find out how the needs are met at other locations and in other cyberinfrastructures. Many times we come back to the researcher and have a suggestion or tool that they were unaware of, or simply lacked the ability to implement on their own. So we find that this bridge role is very important in both directions: both for the researcher and for us to learn which way to go. In a nutshell we find that in order to establish the best practices for a given field of study (say, for atmospheric sciences) it is insufficient to interact only with the researcher. You actually have to interact with other providers of tools for the infrastructure. Find out who the major providers of frameworks and middleware tools are and talk to them.
- The promise of Grid computing is access to shared resources, but there is a big gap between here and there, and we work to fill that gap. We didn't set out to do that. We really set out to fulfill the charge to demonstrate applications in these areas that were chosen by our steering committee. But to do that we found we had to go through the process of repeatedly visiting scientists, interviewing them (much as you're interviewing me now), finding out their needs. But then not stopping there: actually going out and talking to other supercomputing centers, other Grid projects and saying, "Hey - how do you serve the needs of this subfield?" And then we go back to the scientists or researcher with our findings. Many times not only are they very interested in participating, they also find themselves involved in something they felt they never could have done on their own.
- The whole idea is that we approach users who have smaller projects and smaller teams. They usually have very little IT or computer science knowledge. They have knowledge enough to write the model, or to implement their idea, but that's pretty much where things end. So when we start working with these users we have to make things self-explanatory. They can't be exposed to a lot of errors. Things have to be pretty much self-contained and running. After a couple of weeks, the goal is for them is to be able to do their own runs without emailing us questions.
- We are trying to enable them to run them on a number of machines in our system, one at a time at this point. We act as both deployers of the Grid software and then working with these user groups to get their codes running in different machines and to help them use this Grid software to get their stuff done. So part of our work is providing systems support. The other part is a higher-level user support. This higher-level support involves working with the users to help them translate their applications to the Grid context.
- We distribute a compact stack of tools based on the Web services-only implementation of the Globus Toolkit. It is based on the Virtual Data Toolkit (VDT) - a very small subset of it compared to the Open Science Grid. It is supplemented with small-scale tools that we provide. So it's a very compact stack. It can go in, certainly in an afternoon if you've never done it before, and in fifteen minutes if you have done it before. It has a client stack. And we're supplementing that now with a non-VDT based simple Java client that lets people interact with the project metascheduler.

- We have a Web page where the user logs in and can select a type of data mining operation and subset of the data they want to mine. I'm currently working on facilitating that. I'm also facilitating the launching of these data mining tools on multiple nodes, so that Web user queries will get results back in, hopefully, seconds. There are a lot of details of how to launch these jobs and how to move the files to the compute nodes; that's what I'm currently involved with.
- We have a project-wide scheduler. That's where a lot of our work is going now: getting people's jobs to the point where their workflow models are tuned up to handle their workflows. Much of our work right now is in that stage of it. We have a working application, we need it to run simply and reliably on multiple resources, and we're trying to leverage our project-wide metascheduler for that purpose.
- We have ten developers, which is a lot if you think about it. So we spend our programming time making the infrastructure work so that the science workflow proceeds fairly easily:
 - engage the scientist
 - taking that scientist's work as an example
 - then adapting our portal or submission mechanisms so the example can be carried out easily
 It isn't a perfect approach. Like almost all science-driven work you end up solving the set of problems that are in front of you. Everyone wants the general-purpose tool, but that's not so easy to produce.
- We have to spend a lot of time talking with them to find out their needs. They aren't really completely sure of the full set of cyberinfrastructure tools they need to accomplish their goals. Sometimes we can connect them with best practices in their field by simply having them talk with colleagues. Other times we do a little building for them. So with some effort we can get them - and maybe the whole group of people - running on the Grid. Maybe we even get them to form something that will develop along the lines of a Virtual Organization.
- We sometimes will write a little bit of code for them. We always strongly emphasize that they have to do their own research, but we're here to do what we can to help them in a connective role. One of the easiest engagements for us is when the scientists have a running application but they don't know how to run it on the Grid.
- When we first start meeting a new user, that's time consuming because we're on different pages. Trying to get them up to speed on new technology they've never seen before, and at the same time you're trying to learn about their model or simulation. So you're on two very different pages and you're trying to get closer to each other. So that's time consuming. But after about a week or so you get to a point where things start going forward and it's very rewarding directly after that.
- Access to grid resource, job submission, setting up the jobs, staging out the jobs and wrapping them up back to our machines is all something that we don't have to deal with. That is something that is basically mediated by an application developed at the CI, which is called the Swift framework. We use that as our go-between for everything between us and the Grid. We're kind of blind to it. They will locate the resources for us, and I think they will also do a lot of the authentication and the certificates - basically the overhead that's involved with Grid security.
- As a user, I interact with the data over the Web. So everything that we are doing, all of our applications are on the Web. So if I want to do scientific work, per se, (not the preparation for our tools or anything related to the tools and systems development) then it's just over the web.
- As a user I am organizing the bug:
 - does it have some particular pathogenic factor?
 - if it does, what is the best way to design the antimicrobial drug to kill it?
 This kind of analysis is mostly done over the Web using our applications through the browser. Most of the bioinformatics applications are actually accessible over the Web. Some of the applications can be installed locally, but bioinformatics users prefer to use something that is Web accessible for analysis of the data of interest.
- If you were to expose a PACS system as a resource on the Grid, you would interfere with image workflow at the radiology department. Let's say hypothetically that you expose the PACS server as a resource and all of a sudden there were 1,000 hits on the server: that would interfere with performance of the radiology department. That would cut back on the workflow, that would cut back on the revenue. So this is a total no no to expose PACS as a resource. And then there's also some legal issues associated with that, because you cannot tamper with medical equipment. There are some very fine lines that you cannot cross.

- In deploying the two grids for the Children's Oncology Group and Neuroblastoma Cancer Foundation we found that hospital staff are, on the whole, completely unfamiliar with Grid technology. One way to deal with this is to provide a pre-configured Grid deployment for them. So what we do in our project is we provide them with a gateway machine with all Grid components pre-installed by us, send it to them, and they only have to network it.
- In order to build good software systems for any particular field, one has to gain a significant amount of domain knowledge in that field. If you don't acquire the domain knowledge, you are not going to be able to do as good a job as if you do. The best software we use is written by people who have tried to achieve the kinds of goals that we investigators are trying to achieve. They improved upon existing software that isn't as good. Computer scientists and software engineers develop better code than the scientists themselves. The scientists themselves develop useful code that is less efficient and doesn't make use of valuable computer science techniques like distributed computing. When computer scientists actually try to use existing software to do the tasks they're writing new software for, they develop software that is highly domain-relevant. I don't know whether acquiring this domain knowledge is something that needs to be done by the team itself, or whether they need to have close collaborations. I think the collaboration that we have with the University of Chicago Grid experts may be very valuable in that respect.
- We push data to and fro or pull it to and fro. And help our users do that. And then we're trying to work on data collections, which are organizing a bit more in terms of location and metadata describing what the data are. So it's still early, but we're trying to work with one user group to make air quality and weather data available. How do you organize it in such a way that someone else can make sense of it? The "metadata" is for that. That could include a simple directory structure, it could include tags already in the data - I'm not totally clear on that one at this point.
- We're trying to automate everything. The analytical pipelines are very complex and that's why we need to express them in a meta language. We were doing it without a meta language for a while, but under no conditions will anybody return back to those because it's too unreliable, too time consuming and wastes resources
- When the radiologists issue a query they do so from their own workstation. They don't even see Grid. They don't even do single sign-on because the security is built in to the DICOM Grid Interface Service. So the service behaves like the hospital interface, in terms of query trees and storing images. So they don't even see the Grid, and that's one very critical requirement. They don't want to have yet another workstation to query from. They want to use their own workstation, which they're accustomed to, which will read the images all the day.

C.3.2 System design

- Access to the data warehouse is via a Web browser or a Web service, unless somebody requests a tar dump to put it on some other type of physical medium.
- As you integrate new information, you can look at the data from a new angle and it can lead to discovery. Also, if you apply new algorithms to the data you are able to look at the data in a new way. That can lead to discovery. Different groups are developing good algorithms, but they reside in different places. You can install everything in-house, but it would be much better if you could use Web services and actually access remote services directly. In this case you would have a network of information services and information of services. Users are also distributed between different locations. Pretty much everything in biology is becoming distributed because the projects are getting so large that it's impossible to keep everything in one place.
- At this point the groups performing the simulation control access. To some degree it's via unix account permissions, but we have one group that is doing simulations for a broader community. So they're taking in requests through a web browser interface and then running the simulations in a portal account. It varies a bit, but essentially for each project they decide how they want to do it, as long as it fits within the policies of the resource owners.
- Each of the sites is collecting scientific data. The goal of the architecture is to pull data from them in a seamless way, based on both the metadata records and open access to the actual data file. These data are brought into a centralized data warehouse
- From the top-level page of the portal you can click on the link marked "Documentation" and be taken to a page containing a link marked "Client Software Stack". Also from the top-level page,

- near the "Documentation" link there is a link marked "Administrators." This will take you to a page with a link to the "Server Software Stack." It's a one-page set of instructions; we've developed a tutorial based around this. It implements a small subset of the Virtual Data Toolkit. It's basically Globus Toolkit 4, GSI-OpenSSH, a couple of file transfer tools, Condor-G and MyProxy (for handling Grid credentials and doing job submission.) We are working now with the GRMS client, which is even a lighter-weight layer for the client piece. The goal is to drop a pure Java client onto someone's (MS Windows, MAC or Linux) desktop and have them able to submit to our common scheduler without installing the Globus Toolkit. Our CA is accredited, so it comes with the VDT when you install their CA certificates package. So we don't have to do any special steps to get authentication working.
- How do you locate available resources for use in your work? Via the ReSS plus our own additions to that. Then they're advertised and our local Condor does matchmaking between the users jobs and the remote resources. Our additions include:
 - 1) site verification from our point of view, and
 - 2) the additional site requirements users have for their jobs (what software is installed, what does the network connectivity look like, etc.) ReSS shows a fairly generic view of resources, so we add a little bit more detail to it.
 - If the GridFTP team can make the ssh-based transfers work, we might go with that. That's new, it's not as well tested so it's less stable, and it can't do delegation, so there is a concern about third party transfers. There are security issues with the ssh option. So what we may end up telling our users is, "There are two ways you can get data in and out of this place:
 - 1) You can scp it. Then you don't need to do anything. You can log on right now and scp.
 - 2) If you want better performance than that, GridFTP is the tool we have. And in order to remain secure, you must have a GSI certificate to use it."
 - In an earlier phase of the project my involvement included writing the software that captures network events and sends them to a central location to be archived. They're archived on shared storage so multiple nodes can access it. The information about the archived events is stored in a database.
 - Our DataGrid has a number of fairly large Linux clusters. Caltech has roughly one thousand dual core nodes, which means roughly two thousand CPUs. Then each of the sites has around three to four hundred dual core nodes, totaling approximately two thousand cores. UWM has about six hundred dual core nodes. Penn State has about three hundred single core nodes. In Europe, there's a Linux cluster at Cardiff that has about one hundred fifty nodes, there's a cluster at University of Birmingham that's a similar size. There's a four-hundred node cluster at Albert Einstein University in Germany. There's a very large cluster planned at University of Hannover that will measure in the many thousands of cores. So we have a fair number of compute resources, which are almost all Condor-based. Our data storage resources are all over the map. Caltech's tape archive is something like fifteen petabytes; they also have fifteen terabytes of disk that acts as a cache for the tape storage. That's where all the project data ends up - archived to tape. All the other sites have different types of spinning storage. So at UWM we have twenty-five or thirty boxes, each one with ten or twenty terabytes, for storing data. And all the other sites have storage that scales down from there. I think the small sites in Britain only have three or four terabytes of storage; most have thirty to one hundred terabytes.
 - Most other VOs are either based on an experiment or they are a resource provider, and we are neither. We don't own any resources on OSG, we don't have any single project or experiment. Resource usage within the Engage VO is opportunistic. We are getting leftover cycles on OSG that big experiments haven't used up. Our users know this as well. We don't make any promises on how many CPUs our users will be able to get or anything like that. Resource availability can vary a lot, but our users are just happy to get anything.
 - Our methods require us to maintain a large database backend filled with data we've integrated from various databases, so our current approach is a warehouse model. We parse information from over 20 biological databases and we store it in a central warehouse. This approach allows us to present the user with an integrated view of the accumulated knowledge for a particular genome or piece of data
 - Right now it's a single point system. In other words there's no distributed processing, there's no distributed resources. Once the data is in the warehouse it's all localized, so there's no need for a

- certificate. It's basically a one-time authentication. When a person comes to the website we will use tokens like cookies or something like that to actually track session use.
- Our system is primarily a compute Grid right now. A defect in the original project design is that it did not explicitly address storage. We found that storage and data transfer are in strong demand to implement our targeted application goals. So we've done something for now, and we'll probably go back to our parent organization and recommend that we do more. By "something" I mean that we've implemented some data transfer mechanisms for particular subfields (like atmospheric modeling) that move data around, sometimes by non-Grid methods. At my home institution we've also implemented some Grid-aware data transfer tools, and we're looking to encourage other sites within our project to adopt them.
 - The Engage VO is not really a tech support organization, though we are helping users to get going. The idea is to provide an infrastructure that is easy enough that they don't need much tech support. We provide:
 - a matchmaking system
 - a submit host
 - example scripts and OSG wrapper scripts that they can adapt for their needs
 But once they're running the idea is that they should be self-sustained and we should not need to help them much.
 - The data is transferred from the remote sites to the warehouse as a batch process. An analogy would be that we run a cron job that goes out and polls the datasets. If the metadata indicates that a new dataset is available, then we'll pull it in. The transfer happens with Java servlets over the HTTP protocol.
 - The design of our site gateway basically emulates the Condor job manager that comes with Globus. So a big part of building the site gateway is writing something that accepts jobs like Condor does and then forwards them by means of Condor to our local clusters. So the gateway doesn't actually execute the job itself; it takes the GRAM2 job and resubmits the job to one of the sub clusters. That's the basic technology. Myself and another person in my group wrote most of the current implementation.
 - The only resource owned by the Engagement VO is a single machine called the submit host. It has GridFTP, Condor, etc. on it. Somebody else owns the remote compute resources, so we have no control over them. So our submit host acts like a gateway to OSG resources. Our user jobs are mainly submitted via Condor-G on our submit host to the GRAM2 servers hosted by the various OSG resource owners. As far as which schedulers are used by the remote resource owners, I don't know and I don't really care. That varies from site-to-site, and I don't really care what's happening behind there.
 - The personal metadata catalog is built as part of the project at IU. It is one of the tools integrated as part of my work. I work with both data and orchestration groups, so we interact very closely. We also use some Unidata-based tools from the University Corporation for Atmospheric Research (UCAR) in Boulder Colorado. The UCAR folks are also a core data group in our collaboration. Basically Unidata produces a streaming tool that we use for radar data called Local Data Manager (LDM). Also we use a catalogue service from them called the THREDDS (Thematic Realtime Environmental Distributed Data Services) catalogue, which serves data via the OPeNDAP protocol. So there are some tools coming outside of IU too. Our portal is the one stop place to access all of these tools.
 - Though we will largely not be doing Grid work - operations will mostly be confined to our own facility - we will be using GridFTP to move things in and out. We are going to need to worry about X.509 credentials and what we're going to do about that. A lot of our users probably won't have X.509 certificates. And there's no guarantee that the other end is going to have them. It depends. If there is an existing GridFTP installation it most likely will have them, and we'll just have to worry about the users on our end. If it's a new project that hasn't done any work on the Grid in the past, they may be setting up GridFTP purely so they can move things in and out of ALCF. We're not sure they're going to want to deal with all the overhead of managing a GSI certificate authority. They could use DOE's [DOE hosts its own certificate authority] but still there are problems getting it installed. So we have to address that. I know the GridFTP team has done some work on SSH-based authentication, so we're going to take a look at that. We'll also have to look at what it will take to handle GSI certificates. So that's how we protect access to the user data.

How user data are shared is up to the scientists. We don't control that, of course. They do it through all their normal mechanisms - through their collaborations, through papers being published, some of it goes into publicly-available databases (such as the gene sequences), etc.

- We have what we call the Grid Book, which is our Grid deployment installation. It is a pre-configured Linux and Globus Toolkit laptop, which we send out to each of the participating institutions. And that serves as the imaging gateway there. Some centers that have a larger imaging workflow will have what we call the Grid Gateway, which is a rack server which does the image transport and communication.
- We learn of data that should be integrated by being experts in our field. We know what information we need, and what is available. But sometimes we will have no idea about new cool databases that show up if they are not well advertised. So we integrate information mostly from the largest, most popular sources.
- We will talk to people and say, "Look. You don't need to run this in your basement, subjecting your grad students to all that fan noise. You can put your machines in our machine room, or better yet buy into the next cluster we're going to buy. We'll give you your fraction of the resources on a guaranteed basis, and you contribute all your idle cycles to the common pool." And of course, you know the rest. That's the Grid model. So we implement it by getting people to buy in - actually to send us money to help buy the next cluster. To make that attractive to them, we in the IT division will typically chip in also just to get more cycles. So everybody actually does get more than they put in, which always is a good trick. To guarantee their time we have queue structures, which give them priority over their fraction of the resources.
- When engaging the air quality modelers, and we found that they were a highly unique and diverse lot, each with their own science application. But they all needed certain types of model output. And those that cared about real-time modeling needed that to be very current. We also found that in this field they had implemented a method of distribution which could get you the data with a fairly straightforward tool to install. What was unclear was how to map that to making the sets of data available for Grid processing. If you want to be able to opportunistically take advantage of cycles that are available all over the state, you don't know ahead of time where to move the data. So we implemented a method that's running at three places now and we hope to bring it up on a fourth soon. Our approach allows us to have a non-Grid data distribution method to Grid-enabled storage. That way when you run your simulation job or analysis job and you want to consume a certain dataset, you can pretty much count on it being available in a timely fashion at each of our Grid resources. So this is a case where we ended up using a hybrid approach of Grid and non-Grid methods.
- [prompt asking who defines the structures of the site-specific (raw) data] That is really up to the site. This question points to one of the community, or social, issues we address. Instead of forcing or mandating that all sites map to a specific standard or schema, we allow them to adopt their own. Our requirement is that their data be well described in the metadata documents.
- In the very basic sense of the architecture, data is loaded from the remote site is replicated in the same structure and format (other than the fact that it is stored in a relational database.) Once that - we'll call that raw data, if you will - once that raw data is actually localized within the data warehouse there can be a number of workflow steps applied to it. So you get these additional products, which we call derived products. The derived products can cascade on one another also.
- Primarily human interfaces [as opposed to programmatic interfaces]. A Web browser interface that allows people to
 - explore the warehouse
 - find data of interest
 - plot it in a dynamic fashionAnd then if the data look interesting they're able to download it and pursue their research interests.
- Relational databases are not the best way to model our data. We don't really use the relational aspects of it. What we really want are fast index hashes. So what I've asked the RLS developers to think about is abstracting RLS so that it can support other plug-in backends, just like the GridFTP supports other data storage interfaces (DSIs). I would like RLS to support different DSIs. It should have the relational database as the default, but also provide the option of using other methods of representing user data and its mappings between logical and physical filenames. Then what we would do would be to write our own backend based on a hash table approach. Because I

really like the RLS API and I like the model. I'm very happy with it as a service at that layer. What I want to get away from is the relational database backend because I don't think it's going to scale for us going forward five years from now

- we have a customized metadata service, and that's the service that I'm going to move to WSRF in the next two months. So it's a completely customized metadata service. Right now it uses MySQL on the backend, but I hope to change that. It has the ability for clients to obtain metadata information from the service. Generally each site has a copy of the service and it obtains metadata from other sites. We have this notion of an authority site. So if a particular set of files is generated at some site, it has to be published in the metadata catalogues. Wherever that happens, we consider them to be the authority site. And other sites, if they need that metadata, they can get that from that authority site. So that's one way we share the metadata about the data. Then there's the data itself: how do users and applications find out where the data is within our DataGrid. So we rely very, very, very heavily on RLS. I think it's true that we run the largest RLS network in the world. When RLS goes down, you better believe we know it. And we have to jump into action. Fortunately it very rarely goes down now. We're quite pleased. And then for actually pushing data around the Grid, GridFTP is the technology we use.
- we have a web portal that shows machines you might be able to access. And the user must go to the machine owners and try to negotiate access, seeing how much they can get in terms of cycles or storage space or whatever. Then after you actually have access to the resource, the process becomes more dynamic.

C.3.3 System maintenance

- Also we're working on getting MPI authenticated jobs in place. We haven't yet decided our approach for this. We're considering two major alternatives:
 - Is it cheaper to just buy extra hardware and put up a private NAT over which to do rsh among the MPI jobs like everybody else does?
 - Or should we send a Kerberos along with the job and do a Kerberos authenticated thing?This is one key thing at my institution: every single worker node is on the public Internet. There are no NATs and very limited firewalls. The way we've survived up until now has been requiring Kerberos 5 authentication on everything we do. But the Grid obviously doesn't use Kerberos 5 authentication. So you have to figure out some way to let these things talk amongst themselves. Will it be some kind of a VPN that's GSI authenticated? Those are out there - they exist. Should we put up a real private network? Or perhaps we should just send Kerberos authentication around like we used to do two years ago before we started doing Grid stuff.
- I need to worry about security because I have to let people move data in and out of the facility. Users want performance that is better than scp, and GridFTP is the only solution that I'm aware of that can meet that performance need. Therefore I'm locked into whatever security mechanism GridFTP uses. If you could provide me with an alternative mechanism that has all the same benefits and stability of GSI, yet was easier to install, I'd jump on it in a heartbeat. The big thing that GSI gives you that no other solution does is delegation. GridFTP particularly needs delegation because it does third party transfers. The way third party data transfer works is:
 - a client talks to two servers
 - the client tells one of those servers: hey start listening on a port
 - then somebody - you have no idea who in the hell it is - is going to connect to the client and start sending commands. You can see the danger, right? And that's why in plain FTP servers many administrators disable third party transfers. Third party GSI-enabled GridFTP transfers work as follows:
 - the client connects to each server and delegates a self-signed credential
 - so a proxy is created on each server with the client's user credentials
 - further communications during the transfer require the client and server credentials to be identical, or error conditions will be raised [prompt asking about scp] In my experience, for most data movement scenarios you want security only on the transfer commands. You want to make sure the right person is telling the right thing to move. Most of the scientific community does not care if the data being transferred is encrypted. But scp encrypts everything. That is one of the reasons scp has such horrible performance. It just opens up an ssh connection and ships everything

over the connection after encrypting it. With GridFTP you're encrypting the control channel and integrity-protecting it to make sure nobody is tampering with it, but by default the data channel is only authenticated. The data channel is not encrypted or integrity-protected by default. You can turn those options on if you want to - for instance medical people do - but the vast majority of scientific users don't need to. scp also does not support third party transfers. Third party transfers are necessary because they allow me to move data at tremendous rates from a client on my laptop. For instance, say I have a very powerful GridFTP server at the data source and a very powerful server at the target destination. Without third party transfers I would have to route it all through my laptop (or whatever the client has to be running on.) Now, the counter argument is, "Yeah, but I could just log in over there." Well, what if you don't have an account over there? As an admin, there may be a GridFTP server that I am willing to give you permission to run on, but I may not be willing to let you log in to my machine. So the third party model allows you to invoke work remotely.

- I'm not going to accept just any certificate - only those that have come from an IGTF-accredited CA. This is the basis of many large-scale Grid projects such as caGrid, OSG, PRAGMA, the European Grid projects, etc. So this is an area that has emerged out of community practice to solve the problem of distribution of trust anchors.
- My third major focus and highest priority in the current fiscal year is making it all redundant and having failover capacity.
- Our approach for managing a deployment as big as ours is to have only a few approved client installations that receive full support. We promise they will work, and we promise to answer questions after the user works with them. People are welcome to install clients on their own desktop if they want to, but they will get better support on the machines where we installed the client and can log in and watch their job from start to finish.
- The biggest hurdle that we overcame to get to where we are now is by throwing hardware at the problem and getting a BlueArc NAS server, which is a very, very high capacity NFS appliance. Before that, NFS was crashing more than monthly, triggered by the kind of NFS activity that GRAM2 does. We still crash every once in a while - maybe once every other month or so - but nowhere near as bad. [prompt asking if they understand what's triggering the failures] We have some ideas. In short, GRAM2 is doing hard links across NFS, and either the NFS client-of-the-day or the NFS server-of-the-day is not always reliable enough to implement that right. With the BlueArc in place the crashes have to be caused by the NFS client because there is no OS on the BlueArc server at all. These things are fixable if you have time to customize your kernel and work on it a month or two, but we don't have time.
- The first part of the redundancy approach we have the hardware for and are in test mode already. We are targeting our authentication services, such as VOMS, GUMS and a service we created called SAS, the Site Authorization Service. For redundancy we're setting up two physical machines, each with four virtual servers. We're using Xen to do the virtual server. We talked to the leader of the Globus Virtual Workspaces service and received some advice in the early stages that helped us figure out what to do there. It was very helpful. So the idea is no matter whether they're virtual or real servers, we're going to activate all of them. So we'll hide a MySQL server behind there. We tested all of the pieces and we're just now in the integration stage of putting it up and trying to see if it all works together. The next stage is doing failover for the Globus gatekeeper and the Globus GRAM4 gatekeeper. That problem's tougher, as it can only be an active/passive thing. You would have one gatekeeper that's active all the time, and another OS instance that could read that filesystem and be ready to take over should the first one fail.
- We issue credentials through the Texas Advanced Computing Center CA to anyone in our project who needs one. We also have a Virtual Organization Registration Management Service (VOMRS). This is a front-end to a Virtual Organization Management Server, used to generate gridmap files, or to control access with more sophisticated tools (such as the GUMS and PRIMA tools used by Open Science Grid.) Some of our sites are Open Science Grid sites, others are not.
- We use Cobalt because it has Blue Gene-specific stuff in it. I think a Cobalt-GRAM adapter has been written but we do not use it. We're not against running GRAM, but the reason to run it would be if it is requested by a user. Like our user also runs jobs someplace else using a different scheduler, so their scripts assume GRAM. We'll install and run GRAM the day a user shows up

and says, "I use GRAM. That's the interface I've written all my code around." But now because all of our users are Blue Gene users, they're all running Cobalt or Loadleveler.

- We've not yet found it necessary to create a separate VO for each application. We do, on the other hand, map accounts uniquely so that we don't have to worry about people all landing in the same account. The philosophy we take, which I think is very common with Grid applications, is that we control the authentication of a person. We make sure they have an established credential. But it's up to the local resource owner to map them. We simply ask them to map these people uniquely. So in principle it would be possible for a resource provider to map everybody to a generic Grid account. We discourage that. We ask them to make a separate entry for every DN if they are going to use simple gridmap files. We do have several sites (including my own) that have far more sophisticated infrastructures where this mapping is done dynamically, persistently and automatically.
- tasks include:
 - hunting for dead processes
 - cleaning out dead files
 - ensuring all of software on the compute nodes is in synch
 - cleaning dead jobs out of the batch system
 - distributing authorization userids everywhere

Those are certainly things that take up a lot of my time right now

- we knew early on that there would be a high value, at least for the State of Texas, in having a central certificate authority. When we approached our participants we found that one of them (the Texas Advanced Computing Center) was also a TeraGrid partner. They already had a certificate authority that was already being distributed by one subset (the Virtual Data Toolkit), though it wasn't yet accredited. So it seemed to us the best path was to take that as a starting point. Go forward in the context of TAGPMA. Do all the necessary work to make that certificate authority high quality and accepted. By doing that we would then be able to offer all of our participants in the State of Texas a Grid credential that would be accepted worldwide. So this seemed to be a very efficient path for us. We are actually surprised that other projects have not followed that path. Having an accredited CA really is a great barrier dropper for any subsequent partnerships and collaborations you want to make with other organizations.
- We try to find potential software systems that will satisfy our needs, we try to deploy them and see how they work, and then we work with the people who write the software to try to improve it in ways that we need. Our search for potential software begins with first thinking about what we need. Our needs are determined:
 - from our project milestones
 - what we learn as we start building things
 - by helping our users get done what they want to get done

So we'll decide that we need something to do X. And then a lot of times, since a number of us already know the community, we'll have a couple candidate packages we already know. Then usually some quick googling or asking a couple contacts would find any other ones we might not know about. So say we're looking for metascheduling. We'll know of a couple software packages that do that. We'll poke around a bit more, we find a couple others, and then go from there.

C.3.4 Diagnostic techniques

- If people wrap everything with a start and an end event one can look for missing end events. For a given operation, you might know that it normally takes one second and worst case it takes 20 seconds. If 60 seconds have passed and you still haven't gotten the end event, you can generate an error event. We have a simple tool that does that. We're also starting to play around with some more sophisticated performance anomaly detection techniques, such as tracking the running averages of GridFTP performance. We just presented a paper talking about some of those ideas at that Grid conference last week. The idea is to store all this stuff in a database so you can keep track of baseline performance. If your current performance deviates too much from the baseline, you can use things like the MDS trigger service to notice that and generate some sort of alarm. It's still at the conceptual phase right now. The devil is in the details in terms of baselines. You can say this program should always take X amount of time, or this program on this architecture should

always take Y amount of time, or this network connection... just how specific you need to be for a baseline to be valid is definitely an open question.

- Let's say that a filesystem goes away while the code is running. To detect that type of failure is not that easy. This is because different error codes are returned, and some schedulers will say the job was successful while others will say that it was not successful. So one technique we use for dealing with this is we write markers to the job output. The job wrapper will run a code and if it is not successful (if it finished prematurely or something like that) the marker will never be written to the output. After then run the job output is parsed on our side to assess whether or not it succeeded. If not, the job will be resubmitted.
- Most of the times it is easy for me to figure out what is going wrong once it is detected. Once you start zeroing in on an issue its easy. But to know that there's an issue - I think that's the problem. The method I use to detect problems is mainly just getting a feel for the run and seeing how it's progressing. If there's a site that doesn't have any long-running jobs, for example, that's probably an indication that something is bad and jobs are dying. I have a few scripts that help me with detecting those problems. They will notify me if it looks like the site is responding too fast or too slow. In general it's about knowing the jobs and seeing how the sites are behaving. So when a new OSG site pops up that we've never run on before, we send only a couple jobs at a time to the site for a week or so. If after a week the site is running the jobs without problems, we'll slowly increase the maximum number of jobs placed there. This approach works pretty well, actually. A lot of sites want us to be on mailing lists to receive announcements about downtime. At this point we don't even care about receiving those messages. It makes no difference to us whether it's planned downtime or if it's a failure.
- Sometimes the type of problem is obvious. If they're trying to move their data in then we know at least it is a GridFTP problem. And then I have a list of pretty standard things I can start doing:
 - run Iperfs between sites to see if it is the network
 - run all the test suites on our end to make sure we haven't hosed somethingIf I can get on the other end (sometimes I can, sometimes I can't):
 - run disk benchmarks to find out if the remote storage system is a problemYou just start picking the pieces apart. Is the problem in the link between the ends? No. Is our end working right? Yes. Then you start poking at the other end to see what we can find out. Sometimes you find that something was misconfigured, or just congestion on the network, or sometimes you find new flakiness. New software interactions, new Linux operating systems, etc. To use a realworld example: the base SLES 10 kernel is 2.6.16; this kernel has wicked bad TCP problems. But if you can get above that (we're running 2.6.20) they've fixed them. So you stumble across things like that. But so much depends on where the problem lies, and sometimes isolating the problem.
- When we hit a bottleneck we try to figure out where it is: the CPU, the disk, or the network - it's usually one of those three. Then we look and see if anyone else has an installation as big as ours. We look to see what they are doing and then try to do the same thing they did, step by step. We work until either we resolve the particular scalability issue, or we throw more hardware at it.
- we fix the squeaky wheel when we see jobs in the gateway are getting held because of something. We monitor and when jobs begin to fail, we look to identify the cause. One common problem is when new users make assumptions about how Condor ought to work that we didn't anticipate. In some cases this uncovers problems in our code, which we diagnose and then fix.
- we take an incremental approach to tackling issues. Has anybody else done it this big? If so, how did they try to address it? If no one else has done it this big, then we also ask ourselves if we should be trying to do it that big.

C.3.5 Deployment testing

- Each resource on OSG reports into a central advertising component, called the Resource Selection Service (ReSS). The advertisements include which Virtual Organizations (VOs) are supported. So we poll the ReSS (every ten minutes or so) and build a list of all the sites advertising support for our VO. Then we submit a couple of different jobs every six hours to each site, to probe and test them. So that's the first line of defense. If a site has an authentication problem for example, they will obviously fail our test and won't get advertised in our local metascheduler. That type of

- testing takes care of the sites that are obviously broken. Other times we break sites, for example, by submitting too many jobs. And those problems are a little bit harder to find. Those are the ones that we see, "Ok, we have a hundred jobs running, and now all the new jobs are failing for some reason." Maybe we filled up a filesystem or something like that. So sometimes we won't find out about this type of problem until six hours later when the next verification run comes around.
- I am the only person doing the testing/site verification work for the VO. But there is a distinction between the work I do for this VO and the global OSG infrastructure testing effort. What I'm doing is testing from our users point of view, which is different than the larger effort. For example, during the OSG operations center testing, they run under a certain user account. Now maybe the OSG test user encounters no authentication problems, but when running as a user in my VO, authentication problems do appear. And the same thing with filesystems permissions and all that stuff. So we call what we do "testing from the Engagement VO's point of view."
 - I just work within a six-hour (or maybe a day-long) time period, where it's just a moving window. As I mentioned earlier, when the tests are successful they spit out some extra classads - some extra little pieces of information. I take the real classad from ReSS, I add my little extra things to it, and then insert it into my local Condor process. That's how the resources are advertised within the Engage VO. When the next verification test comes around in six hours, previous advertisements are just thrown away. So I just keep a current view.
 - I'll interpret that to mean how do we determine if a chunk of software is doing what we need it to do. In our case we just deploy it and try it out. And by "try it out" I mean that the developers will try it out themselves to see if it actually works, the documentation accurately represents the way the software works, and then we usually have a user or two give it a shot as well. Our testing at this point is ad hoc - we don't develop a big test plan, or unit tests or anything. We just kind of essentially work through the different things we want to use it for.
 - My approach is to do a lot of testing: site verification and probing things to make sure I run into problems before my users do. When a user submits a job and a failure occurs, things are managed in a way that the job will be resubmitted to another site. I then go to the logs at the failing site and look to see what's going on. So my job involves a lot of detective work, I would say. Finding out details about issues. I try to diagnose problems before the users even know that they have them. For example, if a user submits a big job at a site where I see a lot of other jobs failing, I can look at the logs and say, "Ok, that site has a problem with its filesystem." Then I can reduce the number of jobs dispatched to that site, and the user's jobs will not be submitted there anymore. If an error is detected then the site will fall off the list of available resources by itself. The workflow will be successful either way because the job will be resubmitted somewhere else. But time will be wasted submitting to the broken site - I try to minimize that. Sometimes a site looks ok for a while before it begins failing; that's what we have to watch for. So a big part of my work is writing tests and scripts to make sure I can detect problems early.
 - Our testing framework is homegrown with scripts. For our acceptance tests we are testing using Karajan, the CoG kit workflow tool. And if that works out well for the acceptance tests we will use that as a harness to run all of our test suites. "Acceptance tests" are part of our standard test suite; they are used to determine whether or not to pay vendors for new equipment.
 - We have a series of test jobs, so when problems arise we first run our tests and make sure that our tests pass. If our tests fail that helps us identify what's broken. We have test suites that are under development. Some of them exist in a simple form, but we are improving those. These are test suites that basically help us ensure that as we do things we don't take a step backwards. For instance if we decide to make a step up on a Linux kernel, sometimes the Linux kernel TCP performance is not as good as the previous kernel. And these test suites help us identify that we took a step backward, and whether we either need to do something about that or accept the fact that other benefits outweigh this loss. So performing test suites and standard experiments is one of our methods, but it is actually a smaller part of our work. Much of our time is spent focusing on how our users work. Sometimes they come to us and say, "This isn't good enough." Then working with them to see what we can do to improve it. A lot of their problems are code-specific, because optimization depends on access patterns:
 - large contiguous reads
 - small reads that are regularly strided
 - completely random IO

We optimize those differently. Some examples of things we run inside our test suite:

- Iperfs for network
- MPP test suite from MPICH, which is a parallel IO test suite
- IO Bench, for standard disk benchmarking
- the John May MPI-IO test suite, which is another parallel IO test suite
- we run some standard filesystem correctness test suites that are out there in the open source world (such as FX-Linux).
- We run the tests any time we make a change. If we go down for maintenance, or if we upgrade the kernel, we always run the test suite after that. It's kind of our equivalent to release candidate testing. That's our way of making sure we haven't broken things. In our case "broke" not only means not running, but also seriously degraded performance. There's also the flipside: we made some change intending to improve performance, and testing tells us whether we succeeded or not. We don't analyze trends on a regular schedule because they tend to be very intense. We will periodically kick them off if we think there's a problem, or if it's just been awhile.
- [prompt asking how interviewee verifies a site] There's one job going to the fork job manager and one going to the scheduler. The jobs probe certain things that we know might be an issue. For example they make sure that the data and application directories exist and are write-able by us, and that they're not full. We also test for prerequisite software, and whether or not it has outbound network connectivity. Not all of the error conditions raised by the tests are fatal. If we can't write to the data directory, that's a fatal error. But for instance if we can write to the data directory but we don't have outbound access, that's still fine for some jobs. We pass information like that back, and use it later when matching jobs against that site. A user job can express a requirement for outbound network connectivity, so a site lacking that would not be considered as a potential resource for that run.
- [prompt asking how interviewee decides which tests and probes need to be written] That's just by knowing what things have been failing in the past. We have a pretty solid set of tests now. They're not very complicated. We may be testing like twenty or thirty little things. There's nothing that's like a huge test; most of them are very simple, like, "Can I touch a file in this directory?" Things like that. Development of the tests is ongoing. If I have a site that always fails X test, I will go back and see why that is. I need to determine if my test is broken, or if it's something I should report to the OSG operations center. Another thing that can happen: I start seeing that jobs failing at a certain site, I'll say, "If the job is correct yet it is failing, then obviously we're not doing our testing well enough." And we add a new test to make sure that doesn't happen again.

C.4 Methods for Pursuing Science Goals

C.4.1 Analyzing genomic data

- To do the analysis we take the genomes from a national biology information repository, which is called the National Center for Biotechnological Information [NCBI]. We then analyze a genomic sequence with an array of bioinformatics tools so it will be easier for the researchers to answer the questions that they usually have. Some of our users are interested in the whole organism, genome and the properties and the properties of the organisms. But some of the users are interested in particular properties. There are a huge variety of questions that people can ask our application. Mostly it's designed for comparative and evolutionary analysis of the biological data.
- We are trying to understand biological systems through comparative analysis because biology is a reverse engineering science. We have no idea how the systems are built or who built them. Some people think it is God, some people think it is a spaceship with some living creature. Since we don't know how they are built, our method is to compare what is known to what is unknown, and then transfer knowledge from the known to the unknown. Organisms can also be compared, such as genomes that live under different conditions. For example, one might compare microbes that live in boiling vents in temperatures over 100 degrees centigrade with organisms that live at room temperature. The differences can be used to help understand what allows them to live under such harsh conditions. This technique also involves comparative analysis. To do this type of analysis

requires comparing large amounts of data against large amounts of data, which is very CPU-intensive. That's why we need a lot of CPU power.

- We do pairwise comparison of a database, which has six million sequences. So can you imagine how many computations it is? And every analysis run of one sequence produces a substantial output. So it is a data- and CPU-intensive process. Analyzing six million by six million sequences will require on the order of 10^{12} jobs. So it's a huge number of jobs. These comparisons are performed on the sequences stored in the warehouse. Every job applies a popular algorithm in bioinformatics, called BLAST [Basic Local Alignment Search Tool]. It takes one sequence at a time and compares it in a pairwise fashion against six million other sequences in the database to find similarities.
- When genomes are sequenced, they are represented as strings of letters, which signify different nucleotides. Once a genome is sequenced you want to know what functions the genes perform and what physiological and metabolic processes the genes are involved in. So starting from just the alphabet soup of the sequence, by the end of the analysis you know:
 - how many genes this organism has
 - what they do
 - how this organism lives (because we're reconstructing double helix properties)
 - does it have any pathogenic or non-pathogenic factors
 - what does it transport in the cell
 - and what does it produce

So pretty much by the end of the analysis, not through experiments but by using pure bioinformatic methods, the biologists know quite a bit about the organism already.

- we add value to the genomic sequences. We do this by performing comparative analyses using an array of bioinformatics tools. We employ a number of algorithms used in bioinformatics, and we accumulate the results, adding it to our database. We then make the added information available to the user too. So the information becomes more valuable:
 - not only do we provide access to genetic sequences
 - not only do we update it with the information from 20 other databases
 - we also add value through the use of bioinformatics tools.

Analysis of those sequences using bioinformatics tools is something we need to do on the grid. The volume of information is very, very large and some of the bioinformatics algorithms are very, very computationally intensive. If you analyze the sequences that are available now in GenBank on one CPU it will take you approximately 12,000 days. But our Grid-based computational gateway GADU can provide access to 2,000 CPUs, perhaps more.

C.4.2 Analyzing neurophysiological data

- In the distributed computing aspect of what we're doing, we sometimes don't have access to as much of the distributed resource as we need. Prior to making use of expensive and/or harder to access distributed resources we will try out our computational ideas on individual computers or on smaller subsets of computing nodes, before we move things to larger sets of nodes.
- On the broadest level we have an algorithm that looks at the output of the simulations for certain properties. We build a distribution that characterizes that output from all our simulations. So the output of 10,000 simulations will be a distribution of a certain parameter we're interested in. On the basis of this summary distribution that we construct, we make inferences from our data.
- This is a basic principle in the current project:
 - we store our data in relational databases
 - when we analyze the data, we don't read them from files
 - we read them by issuing SQL queries, retrieving the data that we need from whatever databaseTypically for each research project we construct the database that will contain as many tables as we need. So it's a central part of how we work during data analysis.
- We assess some properties of our data. We generate fake datasets where data are actually samples from a normal distribution. And then we try to see to what extent the data we have differ from random datasets. A concrete example: We collect data about the brain, and we have 100,000 units in which we measure the brain - tiny parcels of information. So we try to identify groups of parcels that are active together in the brain; such a grouping would be called an active region. So if

you hear language, an active region would be some lobe in the brain. Because we sample so many units of resolution - let's say, in our example, 100,000 - even if you were just by chance sampling from such a enormous space, you would get some units that are active in close proximity to each other, that are in fact false positives. So we generate random datasets to understand what properties such false positives might have. That is, we generate false datasets to know what cluster sizes of active units one might get purely by chance. And once we learn about that from the simulations we go to our own data and see what's not likely to be due to chance, as compared with the simulation.

- We collect the data using machinery. We collect observations, for instance, on how people understand a given word. I say a word "dog, dog, cat, cat" and see how the brain reacts to them. The fMRI identifies reactions in the brain in a non-invasive way. We have many observations, so we have general scientific procedures that extract the signal from the noise. Our data tells us which parts of the brain react to words like "dog" and "cat". It's basically analyzing digital data stored in files. We take them, we average them, we run processes that help us filter noise and we try to get at the signal.
- We generate the inputs to the simulation in different ways. Sometimes we will just sample data randomly from a normal distribution, and do this many times in Monte Carlo simulations. So not much interaction there. In other cases, each simulation will be defined as some permutation of the original data, within a general statistical context of permutation-based testing. That is, we generate a permutation using some algorithm from the original data and analyze that permuted dataset (just like we do with our real data). We do that many, many times to see if our real data differs from permutations that one can generate out of it. So generating a permutation is a somewhat more complex matter than just generating a noise from distribution.
- We make observations and try to make inferences from those observations. Everything that a scientist does has to do with the data. We record from the human brain using magnetic resonance imaging [MRI] or electroencephalography. We store the MRI data, which I'm most familiar with, as four-dimensional arrays with X, Y, Z spatial coordinates and T for time. We store the data on large filesystems. Data are archived on CDs, DVDs and tape, and are retained until the media disintegrate. We use visualization tools, we run analyses on the data, and we write about the results so other scientists can learn from us.
- We use open source software that takes the brain images we collect at the fMRI scanner, and build files out of them that we manipulate using open source software. We then apply a series of transformations to the data until we get results that we can draw conclusions from. So we set up workflows with input and output running on unix machines.

C.4.3 Aggregating medical data

- A lot of cases, if you look at them on an individual level... let's take cancer as an example: there are specific cancers that you may get one or two cases per year at one facility. More cases are needed in order to really understand the disease, so data must be drawn from multiple centers. That's what we're doing right now with our system. So you end with maybe 60 or 100 cases per year because data are collected from 20 or 30 hospitals. But the really interesting question is "How can we get the 20% of the population that has this disease?" This is why I think Grid technology is relevant within the medical domain. Because it provides the technical underpinning for asking these kind of questions.
- As far as finding resources local to the Grid, you have the images stored on a resource and there's reference to them in a metacatalog. And the metacatalog holds part of the medical information of different types: patient level, study level, series level, and image level. For instance, there's a unique serial number for every image that we use for finding specific images on the Grid. (That means we don't need to create unique identifiers because they come with the data.) Patient-level data includes non-PHI data like age of the patient (not birthdate, which is PHI.) In order to find these resources, you use a metacatalog, and the metacatalog carries the identifier for these images, and then you go to the replica location service and identify the physical representation of the data. And then you find one or more urls where the data is located and then you can get them using GridFTP as the data transport.

- For the research case where the patient is part of a clinical trial, the participating health provider is required to send all data relating to the trial to the trial authority, which in our scenario resides on the Grid. So what happens is that the Grid Book is registered to the PACS system, and the PACS system pushes the images to the Grid Book. The Grid Book cannot query the PACS system. Such a query would be illegal in terms of HIPAA compliance because all of the hospital's clinical cases are in the PACS system, not just the trial participant's case. So what happens is the PACS system pushes the images to the Grid, and then the images become available. What we can do then is store the data on the Grid: you can replicate the data there, you can make the data available to any other instance in the Grid. But you must obtain the images first. There's no way legally to interfere with the clinical PACS.
- We get our images from the hospitals' Picture Archiving Communications System (PACS). This is a standard data storage system within the radiology domain. There are different vendor implementations for that, and all these PACS systems communicate over the DICOM protocol. We get these medical images from these PACS databases into our Gateway and from the Gateway we push the images into the Grid. And then on the Grid we can do replication with the replication management services. And to be able to find the images we also index them using the OGSA-DAI service.

C.4.4 Detecting network anomalies

- All data are captured on the networks we monitor, then data mining algorithms highlight events of interest. We have relationships with institutions that are interested in the same line of research, and they provide us with capture feeds of their network activities. So we receive our partner's data plus our own network's data. So the data from those networks is stored and mined. The database has references to all the raw data files. We can quickly query based on location, time period and the network load. So we can find out these very general network properties from the database. Then in the background as these files come in, we're doing cluster analysis on features derived from that data. The clustering results are also stored in the database. So we cluster on a set of features, and these cluster coordinates and some statistics about these clusters are also stored in the database. You can then observe the changes in the clustering results between different sites and between unions of different sites. Based on these how these clusters evolve (the changing number of clusters, the sizes of the clusters and other parameters) the clusters themselves can be mined for patterns. Particular files can be picked out of storage and additionally mined because of interest in either the network conditions at the time of capture, or the clustering results. At the top level if we've identified an event that is linked to several clusters we have a way to track back to the original data (the original data being those data files that came in from the originating sites.) So once we know when the event happened, we can look at the original files and maybe derive further understanding of what happened by running a different algorithm on the same file.
- Most students have some background in databases, so they're able to query the data they're looking for. Once they retrieve a subset of the available data from the database, they can pull the files representing those events and start data mining. The students have a number of nodes available to them to do the data mining. With the shared storage they can just directly access the file and start their computation. So, basically their job boils down to:
 - starting some number of processes on a number of nodes,
 - querying the database,
 - accessing the right files from the right nodes on shared storage,
 - and running their algorithm on the data.
- Our area of research is networking and data mining. The particular project I'm involved with mines network data in order to identify anomalous behavior of the network and traffic moving through the network. We're trying to use distributed technologies to offload some of the processing that the user interactively might request, as well as to offload processing of large amounts of captured network behaviors, mining them for specific events.

- So right now students just get those fields and do their data mining thing, which involves deciding:
 - which data to mine,
 - how to score it,
 - how to cluster it,
 - what parameters to use for it,
 - how to look for similar behavior in the database once they observe something
 And based on the file they are processing, they can try querying the database to see if similar behavior took place at an earlier time. If so, they can access the files representing the earlier behavior and compare it to the current one.
- The end goal is to automatically detect network anomalies. As a first step toward that goal, we want to be able to interactively analyze the data to gain better understanding of it. This will allow us to devise better algorithms that will automatically do that for us. Such anomalies could include:
 - a user transferring large amounts of data
 - the presence of a probe
 - or some kind of an attack
 Any kind of anomaly, but we're looking at the problem from a behavioral point of view, as opposed to mining actual content.

C.4.5 Understanding fundamental parameters of elementary particles

- As far as documenting the measurements, we have one or several files for each of the snapshots, and then someone will average that up and do some kind of fitting procedure to extract a physically useful result. So you either make a table of what the fitted values are (and there may be different cases.) You may have to do some other fits, at a second level of fitting. At that point we're talking to each other and producing graphs that we hand around for people to look at. So generally a few people are involved in the actual fitting. Then the people who aren't doing the actual fits will examine the results of the fit and the confidence levels, and the plots, etc. And eventually it goes into a paper.
- Generally a project that we agree we want to accomplish is assigned to a center. And someone is assigned to do the running, and that person tries to use up our allocation. We have discussions if something's not going quickly enough. In these cases we might move it to another center, or ask for a dedicated queue, or tell someone to let another person get more time. This level of coordination happens within the MILC collaboration, as well as other collaborations I'm involved in.
- I interact with data by creating it, archiving it, moving it around the country, and analyzing it. I spend a lot of time making sure files got moved.
- Running thousands of jobs to create the configurations, and running many hundreds of jobs on each ensemble of configurations to do the measurement routines. And then to move stuff around from one center to another. That kind of stuff gets very repetitive. And as I say, we do our best within our tools to automate that. For instance, I have several jobs in the queue, and as soon as one finishes it submits the next one, but I have to check to see whether anything crashed and whether anything actually ran. Similarly I have a babysitting script that can see whether any new file that needs to be archived has been produced, and then move the new file. This process would work automatically in a system where a password isn't required to be typed.
- So we are trying to calculate the masses of these particles and compare them with experimental masses that have been observed. Also we try to calculate some masses that have not yet been observed
- The method involves solving a quantum field theory using numerical simulation. And the quantum field theory we're solving is the well-accepted theory of how quarks and gluons interact with each other. In order to solve it we have to do massive computations because we're simulating a theory in three dimensions of space and one dimension of time. We represent the space and time with grid points, which we call a lattice. We then solve for the interactions of quarks and gluons described as fields on the lattice. So the computations become more and more refined the closer we can put those lattice points together. We refine the mesh size, and we get better and better approximation of what we hope is reality. The computations are quite demanding because the

interactions are rather complicated. So we manage to saturate any large machine at this moment in achieving the degree of accuracy that we want.

- The repetitive tasks are somewhat time-consuming. And then the parts that are more fun are doing the fitting later, and trying to understand out what's going on. Just running the jobs is not all that exciting, but it needs to be done. I tend to call it blue-collar physics. Then we also have to write new codes, and that can be time-consuming. Creating the codes and doing the debugging.
- Using this theory Quantum Chromodynamics, we take the continuous space-time of nature and approximate it as a grid of points in space and time. And then we have our variables either live at the grid points or, in the case of a very important variable called a gauge field, on the links joining the grid points. We can then formulate this theory as a finite theory with a finite number of grid points and links. This gives us a finite number of variables, so we can put it on a computer system. Then we examine the system using various numerical techniques, the most important of which is something called "hybrid molecular dynamics." So we use an approach like that to create typical snapshots of the fields that describe the system. The different snapshots correspond to the quantum fluctuations in nature. Then we average over the snapshots that we save, in order to calculate different physical processes. We have to do a lot of sparse matrix inversions. And this molecular dynamics involves frequent inversions. Thus it takes a lot of time to do the calculations.
- We are doing computations in lattice gauge theory at six or seven national centers. These computations involve, among other things, reasonably large files (the files range from hundreds of megabytes to ten gigabytes.) We archive these files, we then pull them out of the archive and use them to study various physics questions. We call those "lattice files". Some of the analysis that we do also generates large files, some of which we like to store, some of which we throw away as soon as we generate them. So we operate between many different sites, and also between those sites and our home institutions.
- We have campaigns that run for months on end. But each of the jobs within a campaign may run for a few hours. We examine the results. We can also do measurements on the lattice that gets produced. But we're not interacting in real-time with any of our calculations, except the most primitive way, which is running a tail on the logfile to see where things are. We check that the job is still producing sensible results and not in some peculiar state because something happened to the machine.
- We have the log files that I mentioned [monitor w/tail].
- We have tools for producing histories of these simulations. So we produce simple graphs that are easy to view on an X Window display. We use these tools for monitoring the progress of the simulations. Once the simulation is finished we go through an analysis process. Normally we bring the log files to our home workstation. This is the primary analysis. We run scripts on it: curve fitting and whatever else we have to do. But that's all done on a single processor. The other result would be a set of lattice files. And those become input into perhaps a subsequent calculation, which is examining and measuring some physically interesting quantity on those lattices. So then it's another campaign with a series of lattice files, and the result always in the end is a bunch of log files containing the quantities that we wanted to measure. Those require statistical analysis and interpretation. But all that is done on a home workstation.
- We take snapshots of the system as we evolve it. We have this grid with variables defined on it. And we archive snapshots of the variables in our system. So typically I might run, for three hours, say, at the Pittsburgh Computing Center and create the next configuration. I would store it on the disk. (Actually, at Pittsburgh the run would be one hour and twenty minutes.) I have five of these evolutions in a job. And at the end of the job I have one 4.3 Gigabyte file that I will archive on the tape system. And so we save hundreds of these configurations. Not all are that size - many are smaller. This represents our latest calculations, which generally get bigger with time. These archived files aren't results yet. With this snapshot of the variables we then look at different physical processes using a different computer program. But once the files are archived, and we share these files with the rest of the lattice gauge theorists in the world, people are able to look at different physical quantities on that grid of variables. Then what we do is average over an ensemble of snapshots; in our case an ensemble would be made up of five- or six-hundred snapshots. Once the averaging is done, we have an average that is much smaller than the totality of the data. And then we do fitting of this, for instance to get masses of particles. So generally things go in two stages. One stage is generating the configurations. The second stage, which can be done

anytime later, is writing and running a program to measure some physical quantity. There are many possible physical quantities one could look at.

- When it's in progress, our jobs will normally run for several hours, depending on the job policy of the center. We don't tend to interact ever with a job that is running, except possibly for machine diagnostic purposes. But each of the jobs ends with some new increment, and then we can look at the result - look at the log file for that increment - and make adjustments in the parameters if we need to.

C.4.6 Simulating the weather

- Much of the input data we are integrating is community data, so anyone can search upon it and use it. Most of the data is available publicly via anonymous FTP. The data resulting from computations are catalogued in personal metadata catalogs. The user has the option to share it, but by default it is private. So the user must go explicitly to publish the data to the outside world, otherwise it is secure.
- So we induce those data, then the user can query the data in a spatial and temporal way. An example query might be: I need the data for a particular region. So these data have spatial boundaries and are valid for only a certain time limit. So we take these temporal and spatial data searches and apply them to our workflows. So we
 - transfer those data
 - we address it to a certain location
 - we catalog it
 - we transform the data
 - after the computation is complete the data are transferred into permanent storage
 - and we catalog the location.
- Our internal scientists need to go through an account approval process. Then we let them run large-scale simulations and real-time computations. We also grant limited anonymous access to the resources. There are limits because some of the simulations and production workflows can be very compute-intensive and time-intensive. So for example if they're doing a weather forecast, we only allow a prediction run covering the next twelve hours for a very small region or at a very coarse resolution. We know roughly how many resources are needed for the simulations. We leave allocations to the user on a semi-honorary basis, asking them to run once in a day. We get monitoring emails to understand what users are doing so if we need to we immediately disable their accounts. So far no one has abused the system so we've never had to disable an account.

C.4.7 Disseminating gravitational wave data

- Because of the kind of experiment LIGO is, all of our files are time-oriented. Every single file has a beginning timestamp and an ending timestamp. And timestamps index very, very, very well. And they hash very, very, very well. So whenever anyone asks a question, "I need a piece of data," what they're really asking is, "I need a piece of data beginning at X time and ending at Y time." And once you know the time range, then all the other metadata is secondary. If we've got five years of data, that's a lot of data files and time. But if you're selecting around an hour, you've already picked out a small subset of the data. Then drilling down with the rest of the metadata attributes is easy.
- I'm the architect, and I design the system. I have two developers who write code around it for me. I have between seven and ten administrators who administer RLS across our data replication network. And then we have one hundred fifty users that interact with the system to figure out where data is.
- Right now our users tend to pick a site and go run there. And while they use some Grid tools to do some of the data finding, they don't typically use Grid tools to leverage more resources than are available at a single site. So another of my metrics for the coming year is to try to enable production data analyses that run across multiple DataGrid sites in a continuous way.
- The rough numbers we always use are a terabyte per day that we have to replicate to nominally seven sites around the world. We still use those numbers, but it's actually much more complicated. It used to be just the raw data products, or the interferometer data, but now as we've given users the ability to replicate and manipulate the data there are all kinds of boutique, customized datasets

being generated. So instead of having data coming out of three interferometers, in some sense we have data coming out of twenty-five instruments. It's just that some of those instruments are computer codes doing different types of things.

- The project DataGrid it's all about the data. We're replicating the data as much as possible to all the sites. Since the replication happens outside of the workflows, then you really want the workflows to schedule around data location. After data location then you need information related to the details of the site:
 - where the storage is located
 - type of storage, such as whether or not it is an NFS-shared disk
 - local directories that should be used for certain workflows

Then there are application-level details at each site. Our applications tend to rely on certain types of environment variables being set. It would be nice to understand whether or not they're all set at all the sites. If we had a way to monitor that, trigger on it and be available for use by the infrastructure's decision-making processes - that would be helpful. Those are more second-order effects. Basically everything else can be accounted for after the data location; that's the big one.

- The number of data products is really exploding. And there's so much to track about all the data. And that's even outside the realm of the science. I'm not even a scientist analyzing the data, and I still have problems tracking it. So that's a major pain point for us. And there's not only metadata about the data, there's the metadata about the workflows, and by "workflows" I mean "analysis pipelines". That's a huge issue. We now have about one hundred and twenty users who do analysis. So these are the people who actually type commands and run codes. They generate a huge amount of data themselves. Post-processing the data, keeping track of it all, and being able to understand where a result came from (the issues of provenance) are all tremendous issues for us.

C.4.8 Enabling access to ecological data

- As kind of the side effect of that whole process though is doing things like managing provenance of the datasets: the provenance from how the original data was collected to how is it derived or integrated into the secondary or tertiary products (in addition to the whole issue of citation of authorship and things like that.) So these side effects are not the primary goals of the project but they're certainly valid secondary goals.
- Right now the ecological community, and specifically our collaboration, has pretty much standardized on a metadata standard called the Ecological Metadata Language [EML]. It's an XML-based language that allows somebody to document a dataset. We're building tools around the standard so that we can:
 - read those types of metadata documents
 - understand the types of datasets that are being described
 - access the datasets
 - and load them into, say, another relational database.
- So we have these two distinct lines of work
 - talking with the remote sites, trying to understand their data, and agreeing upon how to get it from them
 - maintaining the data warehouse

It's a very modular process. There are a number of different parts of the entire workflow process that can be implemented and enabled independently. So right now we're focusing on the data warehouse and the exploitation routines. Basically on the interface to the derived datasets. About a year ago we worked on the component that would actually read the metadata document (the specification for the data table), and load that into a relational database. We haven't yet integrated those two parts yet. There is still work to be done with pulling data from the remote sites. Many issues we're starting to address with the sites are related to quality assurance: both in terms of the quality of the metadata and the quality of the actual data in the tables. We're also working to ensure that there's good correspondence between how data is described within the metadata and how the data is structured within whatever medium it's being stored.

- The issue of ownership becomes a very gray area as you produce these new products. With a product containing a provenance path back to the original owner, one could say that the person still owns that data. But if I perform an invertible process where I can't get back to the original

data, then technically I think we become the surrogate owner of that data. Another example is in the commercial satellite business, and how they consider their data proprietary. In their case it doesn't matter. Many companies consider any derived product from their original product still under license of their business. Let's say I have a derived dataset that goes through a number of stepwise sequences and I create a new product. If I can still go backward and produce the raw product then there's still some level of ownership from the original data owner within my derived product. There's still a component of somebody else's work and toil in my derived product. However if I create something based on a derived product, there is no way I can distinguish what percentage of the value can be attributed to the original owner. I think eventually as time goes on if the product morphs enough, eventually the original attribution somehow fades away. That attribution list can become quite large if you're not careful.

- We're investigating different techniques and informatics in terms of
 - how to access data
 - how to document data well enough (using metadata standards) to be able to read in a data table, for example.

This is still a difficult problem: understanding the different semantics and syntaxes of these diverse datasets.

C.4.9 Understanding cosmological structure formation

- Ah, the output. It consists of a time series of three-dimensional physical fields and particle positions. So for example, the current simulation I'm running at NERSC consists of a 2000^3 non-adaptive grid with eight billion dark matter particles in it. We're tracking about ten or so physical three-dimensional fields (things like density, temperature) and also the positions of those eight billion particles and their velocities. So each dump at any instance in time is about 700 gigabytes (700G) of HDF5 files. We also use the same mechanism for checkpoint and restart. The rate at which those 700G files are produced depends on the phase of the calculation. On a 2,000-processor machine the average dump rate is approximately once every one or two wallclock hours. Then all that data has to be moved from where it is generated to where it can be analyzed and archived. Many centers, for example, have excellent MPP computers, but then fail to provide any suitable computers for doing the analysis.
- Let's say your batch job ran successfully. And now you have several terabytes of output sitting on the high-performance disk, and you've got to do something with it, which in this case means save it to a permanent storage system. Now, is the permanent storage system across the country, or is it local? That adds another wrinkle. However you do it, you first have to aggregate the results into large chunks. This is important either for transferring it effectively across a long-haul network, or it's also mandatory if you're going to archive it on anything that involves a tape medium. I learned that lesson early on, very expensively: if you ever intend to get anything out of an archive there had better be just a few very large chunks rather than a lot of very small chunks.
- One of the things we do to make sure the simulation is progressing correctly is use our own tools to make three 3D volumes of specific fields, or projections through 3D objects. We then make graphical images usually using IDL software running on one of our p690 machines here. The offline analysis of the whole thing - extracting all the science - can take months to years after a simulation is complete. Specialized tools developed by individual researchers may be used in the analysis phase; these tools are beyond the scope of simulation itself. We don't know the details of how people in Germany are examining the results a simulation output. We have no idea.
- These are generally batch-oriented processes that have to be run in many chained, sequential jobs. The data output is enormous - typically on the order of a hundred terabytes. We archive almost everything that we output. At my home institution I personally own something in excess of four hundred terabytes of project data in the archive at present.
- We have accounts with allocations of service units coming from several different sources. For example, we won a NERSC INCITE award, and that is currently running on computers at NERSC and Berkeley. We are in a collaborative project with Lawrence Livermore National Laboratory, and we run simulations on machines there. That system is not classified exactly, but it is restricted access. Then within NSF we have accounts on four or five or six machines that are spread across the centers: Texas, NCSA, Pittsburgh and San Diego. Generally speaking we simply log into to

these machines using secure shell from our laptops or machines at our home institutions. We log in individually and manage the simulations. I hope I've already made it clear that any one simulation may have to be run as twenty or more sequential batch runs. So there's a lot of waiting. Which is why we each individually use different systems across the NSF that were chosen for political expediency, and also as in some cases for their unique computational characteristics.

- Within the US there's an extended group of at least a dozen of us who work either with developing the simulation or using it in specific sub-domains of astrophysics and cosmology. We work together to publish papers and make academic progress. We share the results freely amongst ourselves; most of our results are archived here at my home institution, and they're freely available to anyone who wants them.

C.5 User Issues

C.5.1 Reliability

Systemwide

- As far as mitigating the effects of system failures for this frontline work where you're basically using an entire computer system at a site: one idea is to move away from the batch-queuing model. Move to a model that is closer to a physical experiment, as if you're using for instance, an astronomical telescope. In other words, it would be much more beneficial to us to be able to run for a long time, but to book that runtime at some point in the future. And to have systems staff on call when the reserve timeslot begins, to fix anything that occurs. `_doc: up4.txt`
- And at 60,000 processors (or whatever it's going to be) I suspect that computation at that scale will not be possible using the current approach to batch production. How on earth would you assure a user that when their timeslot came up that every single component of the system was functional? And how long would it stay in that state, given that the mean time between failures is proportional to the component count?
- Having jobs crash, often due to a node going down. That's the most frequent reason that a job fails to complete. Having to fix things up.
- I'm running in this 2,000-4,000 CPU range at the moment. And within the next year we expect that to go up to at least the 32,000 CPU range, if not a factor of two more than that. The unreliability that I see in filesystems, even in batch-process launching systems, disks, monitoring tools - you name it. Nothing really works reliably at the 2,000- or 4,000-processor level today. I am extremely doubtful about it working at a level ten times greater than that.
- I've heard from both OSG and from the LHC Grid and from a number of other Grid projects. They all give roughly the same answer: somewhere around 25 percent of their remote job submissions fail. This is a shockingly high number. In general they don't know why the jobs fail - they can only guess why. The top reasons cited for failure are basic authentication problems. You know - the user might not be in the right gridmap file. There are also disk-related issues such as running out of disk space during the act of staging in some input file, or they don't have the right permissions, etc. But then there are a whole lot of other failures that fall into the unknown category
- It seems like always somehow NFS is involved in some very sticky way when we're dealing with GRAM2 and it's not a pleasant experience. The biggest hurdle that we overcame to get to where we are now is by throwing hardware at the problem and getting a BlueArc NAS server, which is a very, very high capacity NFS appliance. Before that, NFS was crashing more than monthly, triggered by the kind of NFS activity that GRAM2 does. We still crash every once in a while - maybe once every other month or so - but nowhere near as bad. [prompt asking if they understand what's triggering the failures] We have some ideas. In short, GRAM2 is doing hard links across NFS, and either the NFS client-of-the-day or the NFS server-of-the-day is not always reliable enough to implement that right. `_doc: up17.txt`
- Stability for me, in the context of PVFS, means running without failures. Servers don't hang. User jobs run to completion. So right now PVFS hangs and jobs have to stop because they can't write data. We've got to get to the point that something figures that out, a backup comes into play, and the job can continue. `_doc: up10.txt`

- The difficulty for us is not that things break, the difficulty is in detecting that something's broken. I may not even know who owns the site - it's just a black box to me - and something went wrong. Now I have to figure out what went wrong. So I do a little bit of probing, and then either tell the remote site what went wrong, or fix my stuff. Most of the times it is easy for me to figure out what is going wrong once it is detected. _doc: up11.txt
- There's many ways a job can fail, obviously. One of the ways is that the job will be successfully submitted to a site, something will run, but not successfully. Let's say that a filesystem goes away while the code is running. To detect that type of failure is not that easy. This is because different error codes are returned, and some schedulers will say the job was successful while others will say that it was not successful. _doc: up11.txt
- When everything is working, it's great. So a lot of what we need is more fault tolerance and improvements in the way errors and exceptions are handled. The errors can be due to hardware or middleware at any level. _doc: up2.txt
- from a workflow point of view it's dealing with failures. If you can move 99 files with a batch script and they all got there safely, it takes you as much human time to deal w/the one that didn't as moving the 99 that did. So human intervention to deal with the failures is the expensive time. _doc: up4.txt
- it would be good to really understand what kind of failures Grids like OSG experience today. Most of what I know is somewhat anecdotal. Getting a picture of this is a hard problem. I don't know that they know. TeraGrid is the same way. It would be nice if there were somebody tracking and documenting failures in an organized way. _doc: up26.txt
- it's much more common that you suffer a failure in the first few seconds, than it is the last few seconds. If any node, for example can't see the parallel filesystem, that's fatal to a user job but it might be something that can be fixed quite quickly by a sysadmin. But if you're running in batch, you wait days (if not weeks) till your batch job starts, it fails instantly, and you have to go through the whole thing again. So the operation of these things needs to be made reliable in both physical and human terms. You've got to have systems support to overcome that kind of problem in real-time. _doc: up4.txt
- we have so many things to keep track of we're losing the ability to keep track of it. We're building tools that need the information to function; they need the information to leverage the infrastructure. Tools need the information to help the users get the work done. But the systems that provide the information are breaking underneath the load. Then all these tools and infrastructure become unworkable and work stops _doc: up9.txt

Service level

- not all the tools work as well as you would hope in terms of doing what they say they'll do, or having bugs, or "oh we didn't think of this yet". So a lot of maturity issues with some of the tools we try to use. _doc: up7.txt
- GRAM2 has been more stable and reliable than GRAM4. That is the only reason I prefer GRAM2 over GRAM4. I need at least 70% success rate to consider a service stable. Ideally we want it to be much higher, but with GRAM4 we are seeing a much lower success rate. I certainly don't want to blame everything on GRAM4. We've seen hardware failures on the cluster side. But I would say GRAM should improve the way it responds to hardware and network failures. _doc: up2.txt
- I personally would like to have many of the basic services, like GridFTP and GRAM, be more reliable before I see more features coming out.
- It should be a stable service to our user community. A reliable deliverer of services. This probably means we shouldn't use the development versions of the software tools. We probably should use only the production versions. If the production versions are compatible it will be much easier on the application developers.
- The ability to pull things together so easily now using a web interface. But at the same time this opens up problems, because it really allows anybody without a formal background in software development to develop these applications. The concern is similar to my pet peeve with Visual Basic: the resulting code seems very fragile and you have to be very careful how you use it. Things break, or the maintenance of those types of applications become a nightmare at times.
- The goal I'm really looking for is to have a mean time between failures of three months.

- Give us the hooks to make it redundant if you don't do it yourself.
- Let's say you decide that within the software you're developing you want to rely on this particular open source software that seems really cool. You need some assurance that this piece of software will have longevity before jumping into it. Or you may decide that you are better off writing it from scratch, which you really want to avoid if at all possible.

C.5.2 Diagnostics

Error Messages

- When these problems are happening, for instance when hardware fails, the middleware we rely on gives cryptic error messages which we cannot read and parse automatically so that we can adapt to it. As I mentioned before the GridFTP "login incorrect" error does not provide us with sufficient information. In other contexts the source of login problems typically are on the client side, but in the GridFTP case it is often a server side problem. So what I would wish is when middleware cannot determine the particular error (and it's reasonable that it cannot determine everything), I would rather it propagate the original error message. Send it up the middleware layers of the architecture, instead of misinterpreting something and issuing a misleading error message.
- Another type of information that is lacking is documentation about errors. For example with GRAM, all we get is an error code and there is not enough documentation explaining the error. We then have to google to find out how other users handled the problem. Sometimes we even need to go as far as to dig into the GRAM source code to determine under what conditions the error code is sent. There is some documentation, but not at a level enough that we can use it.
- Sometimes we get weird errors that don't really reflect what's going on. Other than that it's fine. Weird errors like, "Error code 17" that supposedly means one thing, but is most commonly due to something else. For example, it says, "could not create job description file" when the real reason is the user doesn't exist.
- We can't use a tool that produces a three-kilobyte file of error messages when something bad happens. Something happens and then all the processes start sending messages saying, "I can't do this. I can't do this. Etc." We can't manage that. That's too much information, which is just as useless to us as no information.
- if we see certain errors right up front we cannot directly take that and send it to, for instance, the TeraGrid helpdesk. I have to do at least an hour of digging. Because if I send directly an error message to the helpdesk they will reply, "This is something to do with your client side. There is something wrong." So I dig deeper and deeper and go through my usual tests, and see, "Oh this service is down. Ok here is what's happening."
- their error messages are very cryptic. The most common error we get is "login incorrect", but it has nothing to do with an incorrect login. It's something like a hardware problem, or there's a node goes down in a striped GridFTP server, or the allocation is out of the limit, or some scheduler is paused. For all these conditions we get the same "login incorrect" message.

Troubleshooting

- How you go about troubleshooting when things don't work. Example: So I do a globus-url-copy from one center to another and I get an error message saying "End of file encountered". And the file at the other end is of zero length. Now what do I do? Right now, I send an email to the administrator asking, "What does this mean? Why didn't it work? It worked six months ago."
- Solving problems is easy once you have all the data in front of you. It's getting the data and knowing what data to get that's the hard part. Networks are notorious for this, right? They're black boxes. Very rarely are you lucky enough to have access to somebody who can actually find out operational status on routers and the like. So you have to infer what's happening by using things like Iperf, netperf, pipechar, etc.
- We're worried about things like firewalls and security policy getting in our way. I don't know if security policy is a technology obstacle or not, but it could be considered one. Part of the problem with logs is that there is potentially sensitive information in there, and if you strip out the potentially sensitive information you often lose the ability to do troubleshooting.

- When anything goes wrong with your certificates, site certificates - anything like that: it's completely beyond the scope of anything a user can do. And usually it's beyond the scope of what the computer center personnel can deal with as well. It usually means that you're just crippled for a couple of days until the one guru at site X can actually figure out why what used to work no longer does.
- When new software tools become available it would be useful to know:
 - what the new features are
 - whether the features are compatible with previous versions
 - where the incompatibilities might impact the other parts of the system
 Because in this case we'll know what to troubleshoot.
- When you write multi-threaded code, bad things can happen. Deadlock. Another condition that also appears in serial code, but is perhaps more pronounced in multi-threaded code: accidental memory overwrites. A tool to handle that at large scale would be tremendously useful. By large-scale I mean hundreds of distributed processes - even thousands.
- Finding out what to do next or troubleshooting is not something I am capable of doing. Not at this point, without going through a learning process, which I didn't have time to do.
- The bigger missing documentation is in the troubleshooting area, where something is happening and we need to find out how to deal with it. Troubleshooting type of documentation is not only for me, but for system administrators - they struggle without this. Because whenever something happens we immediately post it to help@teragrid.org, and the system administrators try to figure things out. All the troubleshooting right now requires knowledge about the internals of the software. So only experienced people can troubleshoot right now. So if expertise is missing on the admin side, the issue keeps spinning for three or four days.

C.5.3 Communication

Globus developers

- The other thing that is a little bit difficult: sometimes I think there's a reliance on the email lists for archiving information. And that's great, because sometimes the details really only exist in an email list and you want to be able to find them. But it can be hard. There are so many email lists I'm trying to monitor.
- It would be helpful if some of the campaign details again were in a more centralized place. And information that was exchanged through the email lists that's pertinent to the roadmap or the campaigns could end up in this other place.
- It would help us to know the assumptions that Globus developers are making on the various files. I'm referring to what are they doing with locking and where the state of the gatekeeper is living (for both Web services and pre-Web services.) I know the broad strokes, but we'll need to know a lot more detail when we do the redundancy work. We'll be emulating the service, and we need to know as much as much of nitty-gritty implementation details as we can. Right now we find these things out by trial and error. [prompt asking for more information on what it means to "emulate a service"] Take our work with our job forwarding as an example: What we did is we took a file that lives down in a Globus library, condor.pm, and rewrote it to do what we wanted it to do. We're basically emulating the pre-Web services "2119/jobmanager-condor" interface [fragment of the conventional network address for remote Condor jobs]. If you send a job to that on our Grid, it's not really Condor underneath the covers, it is our own proprietary system.
- Sometimes when trying to work with various Globus developers, I get the feeling that there's nobody really in charge. Everybody seems to say, "Well, I don't know. Is that more important than this? Is that more important than that?" And the developers are very hesitant to commit to anything without talking to somebody else first. One week it'll sound like it will be a priority, and the next week the work will get bumped. From the outside perspective there doesn't seem to be a lot of cohesive direction and vision. It seems like a lot of firefighting and jumping around :). All the Globus developers we've worked with have been great to work with. But every single time you ask, "Hey, can you add this?" They'll say, "Well, sure, but I've gotta find out if this is more

important than that." I always get that response. I'm talking about tasks that take somewhere between a half day and two days.

- The Globus team has gotten better at this, but there are still times where the team appears to be self-focused or focused inward. This doesn't apply across the whole team. But some folks seem to be focused on infrastructure for infrastructure's sake, as opposed to infrastructure for other people to build on. There are still some pockets of that occasionally. But that's certainly not the rule. As I think about it, the teams I've interacted closely with - the RLS and GridFTP teams - I can say that's the opposite. They tend to be very supportive in terms of reaching out, asking for use case scenarios and requirements, and being responsive to input.
- We're trying to be more deliberate about designing and thinking ahead, without going overboard, in terms of how the pieces fit together. I'd say that's going to be a larger component of what we do now. Going out and talking to the different middleware providers to understand what their roadmaps are, so we can try to get an insight into what things are going to look like one year, two years - even three years from now.

General feedback opportunities

- The way the centers work all the information comes down and there's no feedback, this conversation notwithstanding, from the poor users at the end of this who are forced to use poorly designed and inadequately supported computers. And they suffer terribly in loss of scientific productivity dealing with the endless failures at every level of these systems.
- There is no feedback from the users to the center management or to the NSF, in terms of the cost in human resources in using these systems. The current round of the NSF program is a perfect example: this obsession with buying a petaflop computer for political reasons, presumably to brag about it internationally or something. With:
 - No input whatsoever from the userbase
 - No clear understanding of how it possibly could be used
 - No input from the end-users as to its architecture, its characteristics, or what it will support.

Differences in worldview

- Another challenge is working with the domain science community and trying to understand their needs. Trying not only to advance their science, but also to advance your own. Because one of the problems we face as computer scientists is that we are seen as the technicians for the domain scientists. The advancement of computer science is seen as secondary, as opposed to something that could be an equal partnership. Establishing this type of relationship takes a bit of education on both sides actually.
- Because it's old doesn't mean it should be ignored. Ninety percent of the science codes in a recent Oak Ridge survey were found to be written in Fortran, for example. No one in the computer science community can be bothered to help a Fortran programmer anymore. They probably don't even know Fortran. But in science it's still tremendously important, and C is the next one behind that. We're not going to switch languages. I'm sorry to say that the DARPA HPCS language initiative is pie-in-the-sky.
- Generally I find it hard to imagine that the people who do these security services have ever done any large-scale computing project. For instance, I'm moving files from Pittsburgh to NCSA, so every time a job finishes (and these jobs run for over a year, one after another) I have to transfer a file. So the notion of typing in a password and doing that by hand is very annoying, compared to having it happen with some sort of automatic system.
- In some areas of science people use packages a lot and they're used to the idea of typing in some sort of GUI and hitting "go" and getting some sort of answer. We're about as far away from that as you can possibly get. It doesn't even make sense to me to consider such a thing. When the computation time is measured in months, any kind of traditional view of that just doesn't work. For example this one simulation that I'm running I started working on it in the last week of November and it's now June. It's not over yet. These things don't fit well with the Computer Science idea of running myriad little processes. _doc: up4.txt

- Most of the services seem to do what I have been otherwise able to do for a decade or more (such as moving files with scp or ftp.) So I'm trying to understand where the value is added. Maybe UberFTP is able to move my file about twice as fast... I haven't yet tried it between Pittsburgh and NCSA.
- The people who are doing this ought to have some experience using these systems for large-scale projects. My feeling, perhaps out of ignorance, is that most of these tools that I've seen that are called Grid tools reproduce services we could do before. They seem to have complicated names and complicated protocols. And, regarding the security, the tools are not designed to run jobs that will run for months and months and months without too much user interaction.
- They can't distinguish between their domain science and your infrastructure: Middleware, Grid logins, infrastructure, clusters, their particular science application. Maybe they've even been given a science application from someone they're working with. They can't distinguish among them - it's all "The Computer" to them.
- When I go to a new computer at the Pittsburgh Supercomputing Center I find they do an excellent job of organizing the information that I need to know in order to use that computer. In contrast, the Grid-related documentation for things like getting a certificate is organized into very small chunks. They weren't organized in the steps I wanted. As I mentioned, there are lots of different acronyms. The documentation seems to have lots of different paths because there are so many different ways of doing things. As a user I want one way that is going to work, and work easily.
- For many middleware developers, they often think that's the only thing left to do after you install their middleware. We have to gently point out that application porting is a mere starting point that occupies a small fraction of our time, because it is in fact so easy. It is by no means the end of the story. If we just have someone who has a computationally intensive application that needs a lot of CPU hours, then we hook him up with our Grid, TeraGrid or OSG, and we're done. But very often there's a great deal of social interaction to be done. There's a great deal of organization building.

Intraproject

- I'm a sysadmin so this may not be true of everyone, but I find meetings, collaborating with outsiders, getting everybody up to speed, exchanging docs to be very, very time consuming. So ideally I can have everything in my lab and have my students stop by my office to answer their questions, write something on the whiteboard, and have them start running it immediately. That saves me a lot of time. I'm comparing this to bringing different groups together, having weekly or biweekly meetings, exchanging our little limited views of each other's work, and trying to make sense of how we are going to put things together. That seems like a big, big time drain.
- I, like my a lot of my colleagues, spend entirely too much time in meetings, on telecons and answering email. And not nearly enough time being able to just sit down and solve the problems I need to solve. When you're in these collaborations there's just so many people you have to coordinate with and it just takes so much time; it can literally eat up a third of my time. It's not all bad, but there are days I wish I could just lock myself in my office and do nothing but write code, because I'd get a lot done.
- In a project as distributed as OSG is, you spend a lot of time in meetings and writing emails, just communicating issues back and forth. So I think that's my main problem. If you want to be a part of something you have to invest a lot of time. But the big problem is how distributed the project is. There are so many sites, there are so many projects and experiments and they all have slightly different agendas. So something that might be important to you, nobody else might care about (or the other way around.) Or you're getting pushback from people on something that doesn't make sense to you.
- when you start coordinating with groups like OSG it's pretty complicated, because they're such a big organization with so many different conference calls. It's hard to figure out. Also just trying to keep up with them in e-mail lists - some of these lists get hundreds of messages a week.

C.5.4 Technology adoption

Support burden

- I solve the engineering problems, and I do everything else that's needed to support this lab. I'm very short on time. If I have a choice between using local CPUs to do work, or Globus CPUs controlled by somebody else to do the same work, I'll use my local CPUs. I know it'll take me an order of magnitude less time to set up something on my nodes to give the numbers that need to be computed, than it would take for me to schedule the use of the Globus nodes on the TeraGrid (for instance). The setup work I'm referring to includes account setup for the students that need to access the resources, scheduling time to run our code on them, and installing prerequisite software. To use the remote resources I need to either:
 - work with the admin of the remote sites to install things, or
 - devise instructions for the students on how to compile the software in their home directories so they can be ran that way.Or I can just use my nodes and just get it over with much, much quicker.
- If we can get a lot of users going on our clusters with minimal interaction with user support staff, we should be able to do as well when getting them on the Grid.
- Monitoring our infrastructure decreases my productivity. We have a big academic course in the spring for example, where my research goals and development tasks have to be ignored because the burden of supporting the tools is great. What I mean by supporting the tools is trying to see what's happening on which resource.

Ease of use

- It's not like they can go to the website, download a Microsoft installer package, do a double click and the software installs and you can start it. It's not working that way, and it will probably never be that way because you have to have credentials being created.
- The complexity and reliability of the tools we have to work with is a key problem for us. If you add up all the tools you don't get a very good user environment out of them. It just still seems to be too hard for our users. So for example, there's only been one person I've worked with so far that can really just figure out the stuff on his own. But he's really an exceptional kind of person this way.
- The management of credentials by users directly is too difficult. Our user base is not sophisticated enough to manage their credentials directly, so we are moving away from that approach. We'll be beginning to rely on MyProxy and similar types of credential repositories so users don't have to manage their PKI credentials themselves.
- the VDT is very helpful as far as getting things deployed much easier than in the past. But then after that, trying to get users working with those deployed tools is still a problem, and takes a lot of our time to help users. So ease-of-use is still a big problem.

Cost of use

- I would not consider installing it myself. I don't like the overhead. When anything goes wrong with your certificates, site certificates - anything like that: it's completely beyond the scope of anything a user can do.
- GridFTP carries all the baggage of Globus with it, but it's the only component we're interested in. Really it's just an FTP program - why on earth do we have to bother with all the certificates and all the stuff that goes with it? All we want is point-to-point transfer to be fast and reliable.
- If I could be convinced that a non-GSI version of GridFTP was stable and secure, I'd use it. I hate GSI. It's very good at what it does, but it is a pain. When I was involved in GridFTP development, GridFTP didn't have problems - GSI had problems. Once I could get people past the GSI issues and get it all configured, GridFTP just runs.
- One time I had an expired Grid certificate, and at NCSA it was quite easy to generate a new Grid certificate, but only because I had taken a (paper) file folder from my office, which normally I would not have with me, that has my default password (because I'm traveling right now.) So if I

hadn't decided at the last minute to take this file I would not have been able to get a new certificate.

- Running a CA, deciding whom you trust - that's all a large pain. A very large pain. For example, you have to get a CA certified by TAGPMA and buy special hardware. And to be blunt: after all this is done, as a user we don't gain much of anything. No additional capabilities - you can access the same machines as you could before. You know, it's a big hassle for some potential benefits, like delegation, having your own agents out there to do things for you. There is some potential there, but it just hasn't come to pass that we've needed it.
- If we were to start using all your RPCs, your Secure MPI, and the secure-wrapped standard libraries that have been modified for Globus use. If we were to use all that plus GridFTP, sharing a common authentication infrastructure, then the burden of all these additional layers and centralized key distribution would seem worthwhile. But if we have eight nodes, and we just want to launch our scripts remotely, that seems like a lot of work.
- The time burdens associated with setting up the application-specific environment on the remote machine is a big challenge. The University of Chicago Grid experts handled the Globus-specific setup, so I can't comment on challenges associated with that.

Infrastructure difficulties

- I find it very difficult to figure out how to register these certificates at different sites, because I have a different user name at NCSA and at Pittsburgh and at SDSC. So first of all I found it difficult to find the right place to start looking for documentation about how to get my certificate registered at a new site. I found it easy to google and figure out how to get a certificate. But then to get the Distinguished Name registered and hooked up to each individual account took me a long time to find the right place to start looking for documentation to do that. And then once I found the documentation, some instructions said to use gx-map - other places said to use gx-request. In almost every case, neither one was on my path, and I had to hunt for probably thirty minutes before finding it so I could actually use it.
- I find that I often don't have the right commands by default in my path.
- know that some of the large bioinformatics applications, like in CAMERA [Community Cyberinfrastructure for Advanced Marine Microbial Ecology Research and Analysis] projects, they have amounts of data that are 100 times larger than what we deal with, and they have no idea how they will deal with that. We were trying to download at least small chunks of their data and it was taking hours and hours. Sometimes we can use GridFTP, and sometimes you can't. [prompt asking why using GridFTP is sometimes not an option] Because in some cases we don't have a GridFTP client available to us.
- Once I get a transfer to Oak Ridge running right it's pretty much fixed, except for congestion on the network (like somebody kicking off a big job at the same time I did). That is, until the next time they do a kernel upgrade and blow away all the modifications you made.
- One of the big things keeping GRAM2 alive is interoperability with European experiments. They are not going to GRAM4, as far as I know.
- Security is out of your control and require bizarre and completely Byzantine communication between centers. For example, try using an NSF certificate at a DOE site. Dead on arrival.
- There are no obviously robust methods that we use to help us get around this. Now, I believe there's a thing called Reliable File Transfer (RFT) that might help. But again, we don't just use NSF TeraGrid, we use DOE centers as well, and it's not clear to me that they would implement anything like that.
- Typically it's difficult to go to a regular Globus site and just run our code because we rely on so many external libraries and tools that are not part of the Globus standard install. Sometimes it's possible to request these things to be installed at the site, sometimes not. If it's not, a lot of these tools can be compiled and installed in a home directory and run from there, as opposed to assuming the system has them.
- UberFTP: the major challenge I faced is the first time I tried to transfer a file, it only transferred one tenth of it and then it stopped. And in general it's not always clear who to ask for help because it's always a transfer between two different sites. So you have to get both sides involved, which

- can sometimes be difficult. Sometimes they don't communicate with each other - they'll only communicate with me. They may have different help systems.
- What I find annoying is there are so many authentication schemes. For instance in our DOE SciDAC-funded project we have computers at Jefferson Laboratory, Brookhaven National Laboratory and Fermilab. Everyone uses a different security system
 - When it comes to GridFTP and moving things in and out? We only control one end of the transfer. We can make sure the machines on our end are beefy enough and are configured correctly and tuned right. But if the guy is trying to transfer the other end off his laptop, we'll only go as fast as his laptop. Data transfer is an interesting problem in that respect. It's a two-ended problem. If you're trying to schedule the transfer, it requires co-scheduling - you must schedule resources at both ends. You don't have control over your own destiny: you can control your end and you can coach the other end. But if they don't have the hardware there's nothing you can do. And that actually gets quite frustrating. While I know that rationally they understand it, all the user knows is he's not getting what he wants.
 - If you email them about GRAM2 they will be very responsive, but if you email them about GRAM4, their response is only best effort. I think it is this way right now because GRAM2 is the production version for OSG. If GRAM2 is failing for them, the site is considered to be failing. If GRAM4 is failing it is not that big of a deal for most people.
 - It's not been solvable in a general way. Metadata very quickly becomes very application specific. And most scientists have perhaps not as a good a system as they would like, but they have a system of some kind that they already use for tracking metadata. So we don't provide a standard system service that could be called a metadata service.
 - Keep GRAM2 around in addition to GRAM4, especially in the Open Science Grid at large. The Open Science Grid doesn't have an information system to reasonably send GRAM4 stuff around. Our whole information system right now is tied to GRAM2.
 - the file transfer I was doing from NCSA to Fermilab is going to a special tape archive at Fermilab that's managed by dCache. And so to make this work I have to use an SRM-copy. And the SRM-copy was failing. And the reason it was failing is that Fermilab has to set up certain map files to make that work, and those are not being properly maintained. Maybe it's that not enough people are doing this kind of thing, so these things are not being maintained to the point that it makes it easy to rely on whether or not it's going to work. So then finally just fall back to the old FTP again, and that works, sort of. But in order to get the files onto the tape archive at Fermilab the scp has to go through two stages. You have to move files from a disk to a disk, then you have to move them from the disk to the tape. So that's a painful process and doubles the amount of work
 - There are the hardware problems: dealing with the sluggishness on some of the networks like the ESNET, which we've had some problems with recently. The file transmission rates are painfully slow, errors occur and then we have to retransmit.
 - Users of my Grid have to negotiate separately with each site they want to use. So that means they need to contact the person at each site who has the power to authorize them. This is hard at a lot of the sites because they're set up to serve their local campus users. It's easier for TACC because being a TeraGrid site, TACC can say, "Sure we'll give you a little starter account, and go through TeraGrid to get more cycles."

Lack of knowledge

- I think what really needs to happen is to give training courses to hospital IT- on what is Grid technology in general and what is a concrete implementation of it. That would really help because then we wouldn't have to repeat these discussions with every single institution when we do a deployment. But my gut feeling is that it may be a little too early because this whole field still has to mature. Not only in the Grid domain, but especially the interaction with the clinical hospital domain and the overall healthcare enterprise.
- I would also find more tutorial-like information quite useful. For example I read the whole Globus Toolkit 4: Programming Java Services book and I practiced a lot of examples in there. This is kind of helpful and we would like more examples. Like when we are writing clients to a GRAM service we look for more tutorials or even CoG help in some sense. So more tutorials would be helpful.

- It is difficult to figure out where to find the right documentation. Once I do find the documentation it's very hard to understand - it's full of acronyms, and refers to unfamiliar and unnatural concepts.
- It seems to me that in order to get Grid solutions you have to be pretty tech savvy. Getting the certificates, doing the job submission, doing the DAG of the workflows on Condor, managing the security: all of that seems to be an enormous barrier for actually getting jobs done on the Grid. It didn't seem to me like there is any mechanism on the TeraGrid or OSG to sit down and work out how to solve problems together.
- People write great developer guides. And that's great for somebody who is a developer. But what about the rest of us? What the hell does this thing do? Even with GridFTP we've tried, but people sometimes just don't get the big picture of GridFTP. Concepts as simple as, "What's a client? What's a server? What's a third party transfer?"
- The learning curve for some of the software is quite hard. It takes me five months to train people to use some of our software. That's a long time. But I'm not sure it's related to the software/technology. I think it is that some of the concepts are difficult.
- They say, "Oh good. You're here. You can do this now." And you have to say, "No! That's not what we're here for. We're happy to help, but you have to keep doing your science." The group is populated by a mix:
 - people who are just getting started
 - people who haven't figured it out yet
 - people who really don't know how to use the tools
 - the senior professor who is now going to ask you from this day on how to log on to his email (I kid you not!)

So more than half of the time when we try to work with people who ostensibly are researchers and scientists in their field. They say, "You're so much more qualified than I am to do this that it's hopeless for me to do more.
- We're working to convince the radiologists that this is a good paradigm that is beneficial to them. And the radiologists are not technically at a level where they understand what the underpinning is, but that's not really necessary. So the approach we've taken is kind of the soft approach of learning by doing. For the sites, if they see "oh, this is working", or the security model is gaining trust, then you build confidence. After building confidence you can go to the next step.
- We've certainly hit challenges with the road to Web services: writing Java submission scripts that can be submitted through Web services. The XML - that's the place we sit down with the user. We'll just do it with them because many people look at XML and, well, it doesn't fit their worldview. But it's utterly trivial to sit and work with them, saying, "This is how you specify the batch queue."
- With Globus there's lots of information that you need to get the Grid certificates:
 - * knowing which one is the best one to get
 - * knowing how you use those certificates to authenticate
 - * if you've gotten one from somewhere, how you get to another place and get authenticated there
- If someone were trying to do this who didn't know the area that well, it would be kind of tough. Since for example, compare it to something like Linux. You can take one of a number of distros and then kind of do all your shopping there. There's a bit of that going on in the Grid area, but I think not everything needed is covered in them yet. So for example metascheduling: we couldn't go to a distro like VDT and pluck it out of there. It's not in there yet. It's not mature enough. There hasn't been a consensus yet on what is the best one. So this is where it's a good thing that a number of us on the project know the area. So we know the providers, contact them and work with them directly.
- The lack of technologically trained neuroscientists. I've been trying to hire a post-doctoral fellow who's a neuroscientist to do computer modeling for two years. So one technologic obstacle is training more computationally sophisticated biologists.
- The technology challenge that I face is the learning curve involved in using different software.
- There are still not enough people in the world, as far as I'm concerned, that have real in-depth Globus knowledge. And certainly they're hard to find and hire. So we train people up here, to the best that we can. But it's still hard to say to someone: I'm thinking about putting RFT into this service, but I need to understand where it's going to break. I need you to stand up an RFT, and throw larger and larger requests at it until it breaks." Or "I want you to throw larger and larger

requests at it and tell me the load and memory footprint on the machine." I could do that with my staff, but I usually end up getting it kick-started and spending a little more management time than is optimal. That is not a comment on my staff because they're all good, hardworking, smart people. They just don't have some of the expertise, especially with of the Web services stuff now in Globus Toolkit 4.

- There was a case in which I was creating configurations at Pittsburgh and doing analysis at IU. This is a perfect example for distributed computing: where you run a job at one place and you put the output somewhere else and run some subsequent step of the job. I could do all that just fine with ssh, using the network, queuing a job at IU. And nobody ever picked up the challenge of trying to do that with Grid commands. All I needed was a little background script running at Pittsburgh that I could understand quite well.

System integration

- "Keep it simple" would be the only real advice I would have. You know? The KISS principle. Users these days have got an unbelievable amount of extra work to do compared to the supercomputing programs twenty years ago, when all you needed was a Fortran compiler and a Cray XMP and you were absolutely the best in the world. The complexity of it now is so great that I see it breaking down. It isn't worth our time to consider adding more sophistication, because if we've got any spare time or any spare brain cells, adding sophistication within our code in terms of things that are domain-specific to us.
- An important consideration is which tools are extensible and allow me to build on top of them. This is in contrast to tools that try to provide a complete solution that force me to rip and replace stuff. I try to stay away from the rip-and-replace tools because we just can't do that. Such tools offer solutions, but require me to give up other stuff that I'm already doing in order to use them.
- Educating my home institution about the Grid infrastructure itself. I spend a fair amount of my time making sure that things we need to implement to make Grids work aren't going to get tripped over by the folks who do security. This is, of course, a very common theme. We spend a lot of our time making sure there's adequate communication there so that nobody shuts down my Grid servers because they don't have username/passwords expiring every ninety days.
- I can look at a class of tools that do certain things - purporting to do something, for example "manage data transfers" or "manage workflows". I then try to decide if the tool offers bright shiny new functionality, but will at the same time be unstable and I won't be able to rely on it.
- I have mixed feelings about Globus as it is. It forces the user to implement their code in some very specific ways. So there's a certain mindset that you have to work with. You cannot just take your code and just pop it in there if you really want to take advantage of Globus. Otherwise, you're just putting your own code on Globus using your own socket code, disregarding Globus security, and you're just using Globus to schedule things and gain access to machines. So unless you do it the Globus way, you're not really utilizing it.
- I look at is the interfaces. I'm going to usually have to put some glue in place to pull these pieces together in reasonable ways, and how easy is it going to be for me to do that gluing? Do I have an API that's only supported in one language? If it's not my first choice for the project, I'll need to extend outside of our area of expertise. Or is it something that's technology or API agnostic and I can easily just write whatever I want.
- If there will be some continuity between the releases that would be helpful. Ideally we would not need to rebuild everything in our system to accommodate the new changes. It would be really good to somehow lighten the burden of transitions to new releases.
- It's wonderful when new things are developed, but every time there is a new tool available, it means that in awhile we will need to rewrite the whole system to accommodate the new release. And this can be a problem. I understand that new technologies are being developed and that's why they are getting better and better. But it's a little hard on the application developers when new versions are not compatible with the other parts of the system.
- Most of our applications actually don't code against any APIs. They need to have the environment and security managed for them. So I can't go to a project data analyst and say, "I need you to link with this library so that your tools will interact properly with the security." They expect the

- infrastructure to operate at a level either above or below that, depending on how you characterize it. They just want to run their job, and they want everything to be handled for them.
- There are a lot of different bioinformatics tools and currently we are mostly installing them locally to run them. Sometimes we are submitting it on the network, but probably we just need somehow a more developed system in bioinformatics for Web services. So it should be probably Web-service based, the whole infrastructure. Note that we don't have any distributed algorithms. All of the data and all of the parallelization we are doing is embarrassingly parallel.
 - To me, Globus is a set of daemons and infrastructure that:
 - provides a unified security mechanism with cryptography, key exchange, and authentication on each service using a common set of keys
 - provides a uniform remote procedure call interface
 - provides some file transfer protocols using multiple underlying network protocols
 - has some scheduling capabilities (I guess limited to per node scheduling)
 - contains a set of standard libraries of tools that one can rely on being available on Globus nodes

In order to start doing something outside of the Globus-provided services but staying within the Globus security network, one has to learn additional APIs and how to code things up. So it seems like there's a high startup cost to use Globus. To me it's cheaper to put 20 CPUs behind a firewall and a private network with no way in except through some gateway node that's well secured. I can then just run whatever I want behind that firewall using the most approachable, easiest to use toolkits with the least overhead and with least restrictions on how we code things up. I know there are many people who truly want to distribute their processing across multiple data centers. To them security will be more important. But as long as our project will fit within our local cluster that we can handle ourselves in the back of our lab, it's too much additional work to do it the Globus way.
 - We use a lot of libraries and toolkits like R, which typically are not included in a standard Globus install. So we can't rely on them being on other Globus clusters. So we're shipping our own precompiled code that maybe has R installed in the home directory. We do things like that to get our code running, but in the process we're missing the point of Globus.
 - What kind of logging does it have? Is this a tool I'll be able to drill down easily when I think there's something going wrong? Can I turn up the logging levels so I can really get a picture of what's going on?
 - From the security realm: there are a number of solutions based on proprietary tools. Some people are interested in those because they seem to offer to the users a better experience. I use the term "seem" because I'm not convinced that they actually offer a better experience for the user. But the problem with these integrated solutions is then on the backend there's no choice about how to hook them in to other systems and services.
 - The attempts from Globus-related teams (I don't think these are Globus Toolkit proper) to provide tools and infrastructure to help with metadata and provenance have not scaled. And especially in terms of provenance, they've required too many application level changes. The approach was, "Just do everything this way then you'll get the provenance information." But there's no way to "just do it this way". That's not the way my users can be approached. They are going to do their science. The science is going to lead, and all the other stuff has to be tacked on.
 - The things I need to do from a syntax perspective are completely different between GRAM2 and GRAM4, and require a rewrite of my stuff. And the RSL versus the XML is completely different. All that stuff is completely different. But functionally, no.

Cultural issues

- Sometimes there are terribly, terribly intrinsic issues to deal with. For example in the petroleum engineering field we find they have extremely powerful, well-developed expensive codes that the providers are happy to give you almost free academic licenses for. Really shocking how open they are with their code - you can download it almost like you would a piece of shareware. Extensively developed code. But then if you turn around to a particular researcher and say, "Ok, let's put this on the Grid." You find that they stop like a mule at a door because they won't let go of their data. It's the data that's important in that field. They are highly proprietary, having to do with detailed

field measurements of oil-bearing strata. They are absolutely unwilling to let that part from what their perception of what a secure space is. So we have to spend a lot of our time working with them to assure them about data security and implementing tools to make sure that they always feel in control

- The same people who are probably logging on with cleartext passwords to POP email accounts react with great skepticism when you approach them with an absolutely locked down X.509-secured, strong cryptography solution for controlling access to their data.
- There's tremendous chaos in the identity management area. Everybody thinks they're in charge of identity management. Everybody! It's like when I first started teaching, I went home and told my wife, "Everyone thinks they're my boss: students, the dean, my funding agency." The problem is that none of them are wrong. Certainly your university thinks they're in control of all of the computer identities associated with you. The Virtual Organizations that you work with all want control. EDUCAUSE and Internet2 think they've got a good scheme. TeraGrid has its own thing and they want to be able to decide who in your university can log on to their resource and they're not interested in your opinion about it.

C.5.5 Other technology issues

C WS core

- C WS Core: The examples and documentation. I know they're working on that and it will get better. But right now there's not as much documentation and not enough examples.

GRAM

- A lot of times a cluster user will modify their .profile [file holding unix environment settings] to set their environment for their jobs. They want those values to be used for the job via GRAM4, but they aren't. In contrast, if they submit the job directly to the scheduler those values will be picked up.
- Both GRAM2 and GRAM4 are lacking in the same thing, and that's the ability to do co-scheduling. That's the biggest problem for us. Both GRAM2 and GRAM4 are great for saying I need 10 nodes on that machine, and I want you to run this application when you get them. And I don't want to worry about specific scheduler syntax. I don't care. I'm just going to specify the job in XML and let GRAM talk to the scheduler for me. GRAM is great at that, negotiating to put you in the queue, notifying you when the job is running, etc. That's perfect. But we don't run jobs like that. None of our MPICH-G2 jobs run like that, meaning on a single machine. All of our MPICH-G2 jobs necessarily run on two or more machines. It is imperative that the jobs are co-scheduled: that each job is launched is launched near the same time. It does us no good, in fact in some cases it does us harm, if one job actually gets through the queue executes on machine A, and then two hours later the second job gets through the queue and begins running on machine B. It doesn't do us any good. We need to make sure that they both get through the queue and hit the nodes at the same time.
- Documentation could always be easier to find. In the effort of deprecating GRAM2, the Globus documentation has been made very hard to find. At least it was the last time I looked a few months ago - I haven't even checked recently.
- GRAM4 is a huge resource hog. It takes 700 megabytes of memory to sit there and do nothing.
- The biggest concern for GRAM4, however, is the GRAM container goes into hibernation or stops for awhile without any log messages. And it just comes back by itself after a few hours.
- The challenge that we have to solve eventually is try to figure out what the GRAM4 analogue of the GRAM2 forwarder will look like. How are we going to implement in GRAM4 what we've done for GRAM2 for our Grid:
 - Will we just put GRAM4 in front and keep GRAM2 in the back?
 - Will we try to do a GRAM4-to-GRAM4 thing?
- The other thing is Globus' nasty habit of (at least one time in three, and sometimes more) deleting the file you would like to see before you can get at it. This is with regard to debugging GRAM2.

- There are also issues that we have with GRAM, be it 2 or 4, with regard to job auditing. It always takes investigation into at least three log files to get a full picture of what has happened with a job. Not all of the authentication information is in the right place always, etc. There could be more information.
- We just went through an issue that turned out not to be a GRAM4 issue, but a Condor-G issue. It took us two or three weeks to debug that and identify the problem. It turned out that:
 - Some authentications and authorizations didn't play nice together.
 - Also Condor-G was making calls when it ought not to (or not making calls when it should).
 So one GRAM4-related challenge would be working with the external callouts that are common in OSG.
- A lot of the RSL attributes that are defined, if you can find them on the webpage, are not implemented in the backend scripts. So, for example, we just added some memory support into ours here. Given that our nodes are multi-core, we need to allow our users to say, "I need this many processes with this much memory per core." So that results in us putting one process per core per node, or perhaps one process per two cores per node, depending on the amount of memory they need. So that wasn't a big deal, but it was something we had to add in recently. The LSF.pm scripts did not include support for taking the min memory XML-based RSL attribute and turning it into the right LSF line in the submit script.
- Given our stakeholders it's unlikely that we'll be rid of GRAM2 any time in the LHC era. I expect we'll have to keep it going for at least five years, maybe more.
- There is an issue when the GRAM4 state thing is mounted on a shared file system. This could really put a crimp in what we were trying to do with our high availability.

GridFTP

- An engineering guide written for sysadmins (or people about to install a GridFTP server.) There should be a document that walks you through the thought process:
 - how big does the machine need to be?
 - how big do the drives need to be? how fast?
 - what should the network connectivity look like?
 - should I run a striped server? should I not run a striped server?
 - should I run GSI?
- One big problem we have is with the firewalls, with active/passive settings. We need to have different combinations of active/passive settings depending on the hosts. For instance, for some host-pairs we need to make the source active and the destination passive, but for others we need to make both active. So it's been really crazy and we've had to do all sorts of hacks to switch settings.
- I really don't want to make a big strong pitch for GUI-based tools, but certainly in the area of data transfer that would make our life easier. So if I could get a hold of the developers of CGFTP and say, "Make this real or make this go away." I'd do that.
- I tried to install the GridFTP client myself but it failed on Solaris, and then I gave up because I could use it from Fermilab. I didn't try to track it down further, but when I was trying to install it, it looked like it was trying to pull half of the internet onto my workstation. Part of the problem was, I think, that I ran out of disk space.
- The interface to GridFTP is a bit clunky - we would like something to be as simple as scp. So I gather that the TeraGrid project has done a fairly good job encapsulating some of the knowledge you need into tools such as tgcp, but I get the impression that some of those things aren't really maintained so well.
- The lack of a GUI-based client for GridFTP is a barrier to some of our users. We've tried this CGFTP thing that's coming out of China Grid in some highly incomplete state. That satisfied a couple of our users. Some of our users like GUIs, and they don't like using the commandline to move things around.

Information systems

- I essentially need to figure out:
 - Can I build my code here?

- Will my problem fit on this machine?
- What is the operating system?
- What is the software that's already been installed?
- Related but different: Is my prerequisite software installed?
- The number of CPUs
- The number of nodes
- The amount of memory
- The disk quotas
- The scratch disk space
- It would be absolutely great if there were some information system - actually I guess it's probably not for Globus, because it's probably domain-specific knowledge. The information system would enable finding the services, finding the information, and somehow linking it in a simple way. In this case it could be distributed services and distributed data
- MDS4 Index: It breaks, it's slow, and it's overly complex, in terms of the model. What I mean by that is:
 - XML and XPath is more than is needed 98% of the time
 - Java makes it quite heavyweight for small things
 - The last I heard they were running in memory instead of out of a disk-based database, which hogs a lot of memory.
- So the number of data products we are responsible for is growing quickly. Therefore the number of files is growing quickly. And for us the big issue is not so much the raw data size, because in a sense it's still a terabyte a day. But now instead of being divided over a couple thousand files a day, now it's over tens of thousands of files per day. And they're all different sizes. We have to track so much information about the data now. And so, as has been the case for the past couple of years, we're getting killed by the metadata.
- There are many monitoring tools out there such as INCA-based services, and the TeraGrid user portal has some of them and the WebMDS is supposed to have monitoring information. But based on our practical experience we see that the frequency of those tools monitoring GRAM and GridFTP is not accurate. So we end up having our own tools to test in real-time whether or not GRAM and GridFTP are up and running. If they are down we immediately black-list that host until someone manually goes and brings them back up. Until they are back up we submit to a different resource. The reason we need to test in real-time is we have seen many examples where the monitoring tools aren't showing whether a service is really available. If you ping GRAM or GridFTP it works fine, but if you actually try to do some functional thing (like transfer a file or submit a job) it fails.
- If I can find all my bank history in a split second, why can't I find a machine that meets my requirements in less than ten seconds

Install/deployment

- A lot of the more advanced configurations and uses of Globus seem to be not as well documented. So for me, that means I'm generating each key by hand for each user, and distributing the signatures to each node to allow the user to log in. It seems very painful and very complicated. And for what I was tasked to do (enable users to copy files and launch jobs remotely) it seems like a lot of work.
- A significant amount of user problems related to Globus or associated Grid stuff is simply that the client is configured incorrectly.
- Another technology-related obstacle I encounter is the issue of coherence of a given set of software. It is not possible to implement just one piece. Even all of the Globus Toolkit clearly is one piece. So the technology obstacles are ones of keeping the different components into a compatible state.
- In the two-and-a-half years of my project Grid there has not been a time when we've had the latest software versions installed on all of our machines. We are still not up-to-date.

- So then when we go to add pieces like our metascheduler, if that falls out of synchronization with some features of the Globus Toolkit it can cause us problems. Nobody owns these problems. We have to solve them because they're our set of choices of what to include.
- The challenge of maintaining a very big and very complicated software stack on more than 3,000 machines is very difficult. The solutions we have in place now for managing this are not adequate. I send the instructions to the sysadmins and they say, "What? This is crazy." and I say, "Yeah, I know, but it's all we've got right now." So getting a very complicated software stack distributed and running on all these machines is difficult. But this is mostly not a Globus problem.
- It is turning into a situation where you can't even use a distributed file system to get the software out there. There are more and more requirements, and more and more stuff has to be pushed out locally to every single compute node. Of course the Grid was sold in a totally different way when it first came out. It was supposed to just live on your batch system host and you wouldn't have to worry about it. The nodes wouldn't have to know about the Grid. In practice on the OSG this is not the case. The OSG stack for every single worker node these days includes all the Globus clients, such as globus-url-copy and the Web service equivalent. It includes Grid security certificates and certificate of authority files, which are used for authenticated file transfers. And then there are many more things the OSG has on top of Globus, the latest of which is gLExec, which is used for pilot [technology from gLite] jobs. Several of the big virtual organizations have this technology. You might have one guy sitting at FermiLab sending out Condor Glideins all the way across the Grid, and pilot uses gLExec to determine the appropriate userid the jobs should run under. In the big picture gLExec pushes responsibility for authentication and authorization to every single compute node. The software stack to support this is very complicated.
- Version consistency, standardization - that's clearly the name of the game here. The pace of change of some of the software is dizzying.
- What I'm kind of looking for personally, is like when I install my favorite linux distro (or cygwin on windows, or fink on the mac). I can go and pick out what I want and it almost always works. You get a menu, you pick, it installs it and everything works.

Java WS core

- I think the Globus MySQL instructions and the way the database is connected should be revised. You have to install a specific version of the driver. Why is that? These kinds of things are a little bit nagging. And some of the drivers you can't even get anymore because they're outdated.
- The major problem with the Java container is that the database connectivity just sucks. In all the compilations (I started with 4.0.0 and then all the subsequent versions until the latest) the ODBC drivers always give me problems. Compiling this container can be a very simple thing, but it can also be a very painful, depending on whether or not you need these database drivers in there
- There's also the issue of the guaranteed delivery of notifications. Let's say we have a sensor that needs to aggregate some data. So every now and then it pops up and says, "Ok, here's my data for the past hour" and sends the data to a service. At the same time it would check for any pending updates from the service, so it would process notifications sent by the service while the sensor was offline.
- More dynamic IP address handling is needed. You know, how the container handles the network coming and going needs work. I think the container's notifications could be a good fit for us, but we need something that works better in that environment. So the use case I'm dealing with (in a different project) is where there's a sensor somewhere connected by a GPRS cell phone. The sensor gets different IP addresses every time it connects. It's just up for a few minutes and then goes down again. The current notification framework doesn't really work well in that dynamic scenario
- The common use of PostgreSQL in the toolkit should be revised. I think MySQL is more common than Postgres

MyProxy

- I don't know where the documentation is, and if it doesn't work the first time I have no idea what to do.

RFT

- RFT is very good when you set all the optimal settings. On the default setting the performance is very bad compared to GridFTP. But if you tweak all the parameters we get the optimizations. So we need to find out and learn some external tools to provide these optimization values. For example we need to look more into MDS and see what are the optimal configurations to set between two hosts.

RLS

- And as it turns out, relational databases are not the best way to model our data. We don't really use the relational aspects of it. What we really want are fast index hashes. So what I've asked the RLS developers to think about is abstracting RLS so that it can support other plug-in backends, just like the GridFTP supports other data storage interfaces (DSIs). I would like RLS to support different DSIs. It should have the relational database as the default, but also provide the option of using other methods of representing user data and its mappings between logical and physical filenames. Then what we would do would be to write our own backend based on a hash table approach. Because I really like the RLS API and I like the model. I'm very happy with it as a service at that layer. What I want to get away from is the relational database backend because I don't think it's going to scale for us going forward five years from now
- We tried to deploy RLS on a 64-bit machine, and during our critical production mode it did not scale beyond a certain limit. So very low scalability in 64-bit mode. We told the RLS developers and they identified some problems in the C globus IO libraries. They gave us some fixes and there is still an open bug report about it. In general the scalability issue has haunted us a lot and so we've had to find workarounds - on 64-bit machines. Things are fine on 32-bit machines.

Security

- It seems that if I were to have 70 Globus nodes under my control (which I have not reached yet) but if I had 70, I can foresee difficulties associated with centralized account management. Key management for Globus seems very complicated. Until I go beyond 20 or 30 nodes it's been suggested to me just to keep the keys locally on all the machines and not try to centralize everything - it's much easier that way. Some heavy Globus users have suggested this to me.
- the standard Globus instructions basically lead the new user into an exercise of SimpleCA and building their own X.509 capabilities. These instructions are essentially useless in the context of any large-scale deployment where you actually have to trust each other and you need to build a foundation for trust.

Workflow

- Our Grid gateway uses VDL. We haven't transferred all of our domain-specific applications to VDL. Some time ago there was no recursion, but I think the issue may be addressed in Swift.
- VDL is good, but the problem with VDL was that it wasn't very stable. But then currently we are running pretty well with it. So I don't know what the future of it will be because I know that now it is called Swift

C.5.6 Other social issues

Allocations

- One of the biggest challenges is getting a large enough amount of computer time to do the calculations that we would really like to be able to do but just can't accomplish right now.
- the usual funding issues where you're writing grants and waiting six months to find out if you did or didn't get it. Especially for solicitations that have less than a ten percent success rate - that is pretty counter-productive.

Community awareness and collaboration

- For us to do the research we want to do we need to have Globus and (ideally) Condor and some other software using the new format logs. We also need to get OSG to deploy the central log file collection stuff. Some of these things are not super-high priority items for everyone involved. So it can take a lot of phone calls and prodding. Everybody agrees it's a good idea. It's just not always on the top of everybody's priority stack.
- One challenge is that there is so much development happening now, in so many directions, by so many people, that it's becoming harder and harder to keep track of all the developments. Trying not to reinvent things and trying to leverage what's already there is a constant challenge.
- People have a tendency to write and rewrite monitoring applications as if forgetting all the enormous amount of work done on this topic by people before them.
- When you need changes or modifications or improvements it's not under your control. You do not directly control the developer resources to get those changes done. And so you have to go back and ask them, and you don't really have much to offer in return other than the greater good of what you're trying to do. We've been doing well operating under these constraints, but it hasn't been easy all of the time. There have been times when we've been told no, flat-out, "No, we aren't going to do it; I don't care how much you need it." And then other times when we've been told, "Yeah, but it will take a while." And in some cases, it takes a long while. Also there are other times when you get it right away. No one's out to get you, but they all have their own agendas. Your requests need to be fit into larger priorities, and sometimes the requests are not given as high a priority as you would like. It's not that they're trying to hurt you; it's just that they have other bells to answer. That's hard. There's no way around it; I don't know how else to put it. It can be a problem at times.

C.6 Satisfaction Points

C.6.1 GRAM

- GRAM is great. In one fell swoop it allows me to specify jobs in a single language, and hides all the details of every local job manager (which nobody wants to learn.) Plus, it has the entire security infrastructure built-in. That's a hard, hard problem to solve.
- GRAM4: because the staging support is better. It's a pretty big improvement over GRAM2 in that sense, because you can do smarter staging (like the whole filelist). That maps much better to our Condor description files. And in general the architecture is better.
- The Rendezvous Service was written to provide all-to-all exchange of information in the Web services context. We were able to whittle it down to a reasonable API. And it was even a great exercise because I showed up with a need that the GRAM developers didn't foresee. We talked with them and nailed down the requirements, and it was done. This was about a year or two ago. It was great.
- We do use GRAM4 - we've started to use GRAM4. We also use a new thing, which I can't tell you exactly where it sits. It is called the Rendezvous Service. It may be its own service; it may sit inside of GRAM4 - I don't really know. But it was critical for us as we moved from pre-Web services Globus to Web services Globus.
- The GRAM interface is really cool and provides us with one common interface for all the job managers. This is important for us since for our requirements we need to use multiple clusters and

- each has its different job managers. GRAM enables us to use the uniform mechanisms for both job submission and monitoring. It also supports the authentication mechanism that we like.
- We didn't see any particular need to support pre-Web services. And so as an experiment we tried just GRAM4. It seemed to have advantages in terms of being stateful and allowing us to interact more closely with our potential services. Certainly from the point of view of just job submission it was relatively trivial to adopt.
 - We use GRAM4 for our job submission. That's where we started two years ago. We've had a little bit of struggle along the way - of course it's been improved. We really have never regretted that decision. We will support GRAM2 job submissions to our batch-oriented resources on request, but we've never had a request that couldn't be satisfied by teaching the person how to submit via GRAM4.

C.6.2 GridFTP

- GridFTP: Our major challenge has been the LSOFF, but it looks to be addressed and we're very excited about leveraging that functionality. Other than that - jeez it's completely reliable and moves tons of data. What else could I want?
- It is the way that exposes the highest rate of network transfer from filesystem to filesystem across a transcontinental distance. So for example if I have to move output from Pittsburgh to San Diego, GridFTP buys me a factor of two or three over bbFTP, which I could look after myself. And that two - three is essential.

C.6.3 VM

- For redundancy we're setting up two physical machines, each with four virtual servers. We're using Xen to do the virtual server. We talked to the leader of the Globus Virtual Workspaces service and received some advice in the early stages that helped us figure out what to do there. It was very helpful.

C.6.4 Install/deployment

- The Globus installation has improved a lot. I've been installing Globus from the first version to today. The installation is very good, the documentation - the problems we encounter during the installation - they've been documented very, very well.
- The challenges that are associated with using Globus (many people feel it's complex and hard to implement) were greatly lessened by our decision to adopt the VDT-based installs. We've worked very closely with the VDT team, including with security updates (a couple of which actually made it back to the Globus repository.)

C.6.5 RLS

- So we rely very, very, very heavily on RLS. I think it's true that we run the largest RLS network in the world. When RLS goes down, you better believe we know it. And we have to jump into action. Fortunately it very rarely goes down now. We're quite pleased.
- Our project testbed can be distributed across multiple locations because of the replicas. The replicas also allow us to choose the fastest available compute server for our computations

C.6.6 Security

- The Grid security infrastructure is one of the things we completely rely on and are building our tools on. For example all these remote job submissions and remote file transfers are completely based on GSI. So we absolutely love GSI-based authentication. This is something that has been really helpful and paved the way for the portal-based computation.
- We'd be lost without X.509 authentication. That just solves a whole raft of problems.
- With Grid technology with the security model, you can do quite a better job electronically: you can do an audit, verify certificates, verify attributes, etc. These mechanisms are way better than what clinical practice is right now, because many of the documents today are in writing, stored in physical files, reports end up in the trash, etc. So there are many places in the current system

where private information is exposed to the outside. This is one way that I think Grid technology can help, because it has a very good security model.

- You just walk into a sysadmin's office these days and say, "We're using GSI - the Globus security infrastructure." and they get it. They know that it's been looked at by many eyes (maybe even their own). They trust it, so you don't have to convince them it is secure.
- If you have an IGTF-accredited CA that's enough, because other large-scale projects throughout the world get these sets of trust anchors. So they know whether or not to trust the credentials of your CA, and on what basis. They know that you have, for example, been in-person identity-proofed by someone in the chain. They also know the CA is run in a method that does not allow a graduate student to walk in and issue their own certificates. So since these things are widely distributed and commonly accepted, it's very easy to start a virtual organization. We always make the point that authentication is not authorization, but it's a starting point. They can then know the quality of the Grid credentials coming in and use that as a basis for signing membership to the Virtual Organization. It becomes barrier-lowering because I can accept a certificate issued in Czechoslovakia, for instance.
- The Globus GSI and everything that's built around that ecosystem now works really well for us. We can hook into it in so many different ways. We can set up services that manage the delegation for the users. The only thing the users have to do is enter the system once using something like MyProxy. Everything else is handled for them. That works really well.

C.6.7 IO libraries

- IO was very good, but XIO is even better because it allows us to build protocol stacks. The protocol stack design will make it much, much easier for us to introduce new technologies in the future. So when it was all Globus IO-based we had to go through the exercise of pushing GridFTP into MPICH-G2. It didn't kill us, but it wasn't easy. We had to munge the code pretty heavily to shove that stuff in. And we had to go through the same exercise when we had to push UDT into MPICH-G2. With Globus XIO, all of this can be very neatly wrapped into an XIO module that either I or someone else writes. That's the recipe for rapid prototyping. I won't have to mess with the MPICH-G2 code at all anymore. I'll just write an XIO module, and one line of code to activate the module in MPICH-G2, and it's done. And you get it for free. That's a big step forward for us. That's a big help.
- The Globus data conversion library is indispensable. We need it. If it were to go away, we'd have to write it from scratch ourselves. MPICH-G2 is responsible for doing the data conversion between big endian and little endian machines, for example. The MPI application is not going to give a darn about that. I need to care about that. It's an ugly job that no application should have to write. One library should write it once and then provide it. We provide it in MPICH-G2 because the Globus developers wrote it.

C.6.8 General

- We went to a scientist who was highly computationally bound working on a small set of machines and created a portal environment to encapsulate his workflow. In the two week demo the scientist had access to far more compute cycles than he had been able accumulate to date. He said he got publish-able work out of it. ... The scientist has since gone on to get research funding to buy clusters, and then contribute those clusters to our project.
- Globus is certainly the dominant technology. It is compatible with a lot of the larger projects we want to interact with.
- Client side backward compatibility is important. When a new version comes out I should not have to rewrite my software. This has been a major concern in the past, but has been much better lately. If the clients can talk to the services in the same way and get the same functionality, that would be good.
- Globus rules. It's gonna save the world. I'm not kidding. We rely quite heavily on Globus. Globus does all process management, the start-up, the security. In Globus we're using IO for all inter-machine communication. Globus is the one software that we use across all applications.
- I am attracted to the WSRF framework. I think that, especially for someone like myself who's not a computer science person, it allows me to quickly and easily leverage things like resource

- properties and the publication of resource properties. The lifetime management, the subscription and notification - all the nice things. It allows me to leverage them quickly and much more easily than I could do if I had to do all that stuff by myself. After all I am a physicist and I'm dangerous when I'm writing code. The more code other people write for me, the better
- I certainly appreciate all their efforts. The Globus Toolkit has really succeeded in what I think is one of its primary missions: enabling more science. Without a doubt, Globus has made more science possible. Period.
 - I really appreciate the overall effort. For Grid the whole paradigm can only thrive is if there's an open source and standards-based implementation, and the Globus Toolkit is delivering exactly that. See one problem in the medical domain is that the internals of every equipment vendor, both software and hardware are proprietary. There are some standards on the interface side, but internally it's all proprietary. I think that the whole concept of service-oriented architecture presented in the Globus Toolkit and the Grid paradigm can have a major impact on how medicine is being addressed from a technology side.
 - I think it's very good that this survey of users is being done. And I think it's something that should have been done five or more years ago.
 - I want to say that I appreciate having people who build the software actually look at how people are using them. So this interview process is useful.
 - I'd say within the last six to ten months we've been seeing more Globus developers showing up on some of the OSG calls and being there as a resource, if needed. I know where to find them if I need them.
 - It's really not a lot of work to customize the Globus tools to fit their own users' use cases. I see TeraGrid doing that, and I think that's great.
 - The Grid has made a qualitative difference in what we can do. For example, to analyze the data in preparation for a new release of our data products: If you want to do it on the cluster sometimes it's very difficult to get nodes. And on forty nodes the analysis will run for weeks. But on the grid we can immediately do it and it will take so much less time. We can provide our users with fresh data more frequently because of the Grid.
 - Keep doing it. The work is incredibly valuable, and I just don't think we're at a point where we can stop. It's like we're building a car and we haven't put the engine in, so we can't start it yet.
 - When the time is right we're certainly going to start utilizing those services and those applications. Exactly how and when and other details have yet to be decided. But yeah, go for it.
 - Thanks. But don't stop.
 - Some people are perfectly ready for Grid technologies – for access to highly distributed computational resources. Those who say, "I've been waiting for you to come along." – they have an application that needs cycles or needs to move data or somehow needs a cluster. Dealing with these type of folks is so easy. We enjoy it so much. We give them credentials, we adapt their application. Ours is Web services only, so we wrap it in Web services submission script. Maybe we even build them a small service. We drop the tech in and they're happy
 - The formalization of Web services I think is an incredible technology to allow machines to communicate with machines in a very standard and structured way. Though there are still questions on what to use in the Web services area: Do I use SOAP? Do I use REST?
 - Honestly those types of remote procedure calls have been around for a long time in the Unix environment using sockets, but they were unique to those platforms. I think the convergence now with the standards is really making things fly.
 - There's no other technology out there that provides compute resources, data management resources and security at the level that the Globus Toolkit does. Period.
 - We really like Web services because it lets us build tools that are better suited to scientific workflows. They really are services - the steps that we need to accomplish. That could be better if we had user-pluggable services. We're not there yet.
 - We're working in the field of radiation modeling for cancer therapy, and there's a proton accelerator here in the state that has been built by the M. D. Anderson Cancer Center. This large-scale \$120 million facility has huge modeling needs. We thought it would be a very hard problem to move the medical data around. But we found that there are tools in the caGrid software stack that are not only well-suited, they're actually explicitly written for the purpose of moving medical image data around with high security using Grid tools.

- So that is an area where we thought we'd need to do a lot of development. Instead we found a complete working infrastructure that we just didn't know about
- We've recently become an incubator project for our netlogger work. There have been no hassles other than trying to convince LBL lawyers that Apache and free BSD licenses are effectively the same thing - which is LBL's problem, not Globus's problem. I've been pretty impressed with the whole incubator process. I like the fact that you get a Wiki, a bug tracker, and a CVS repository. And if you need something configured it seems to happen pretty quickly. The lead dev.globus infrastructure person seems really good. I was impressed with the whole incubator startup process and how smoothly it all went.
- When computer scientists actually try to use existing software to do the tasks they're writing new software for, they develop software that is highly domain-relevant. I don't know whether acquiring this domain knowledge is something that needs to be done by the team itself, or whether they need to have close collaborations. I think the collaboration that we have with the University of Chicago Grid experts may be very valuable in that respect. As neuroscientists we've had our frustrations, and those frustrations are being solved by some of these new approaches. Actually, that's not fair to say. They're not being solved, but we're working towards solutions. We'll see in five or eight or ten years whether we've really had a good effect. I agree with the University of Chicago Grid experts' approach, which is to be highly collaborative with the domain specialists. I applaud that and think it's the right way to go.
- With the Globus team I generally feel like when I ask questions they're quite responsive.
- Based on our profession interactions with other large-scale centers, we can quite often implement a framework that would have been beyond the reach of a researcher left to his own devices. That is very satisfying. It isn't really a Grid topic per se, but it intersects a lot with the whole idea of shared resources.
- Leveraging large software projects like MPICH and Globus (and even to a lesser degree GridFTP or UDP or UDT) is great because it's a tremendous leg up. You leverage it. There's a bunch of code that's there, and you apply a little bit of work, and you get a tremendous benefit from it without having to do all of the work.
- Much of the basic functionality, such as data transport, is already included in the Globus Toolkit. And there was not a lot of effort required to make Grid technology work for the medical domain. And that was very neat because you don't have to reinvent the wheel.
- Thank you so much because somehow the use of the grid and the use of the Globus really, really, really made such a huge difference in what we are doing. We can do so much more science after using the grid. So just my deepest, deepest, deepest and sincere thank you.
- The Quickguide to installing the container is very good, very straightforward and clear. Even for somebody who is a beginner, this is a straightforward document
- The toolkit model fits me exceptionally well. I'm really looking for tools and infrastructure that allow me to build on top of them. So they have to be extensible, they have to have nice APIs and hooks that I can layer my own stuff on top.
- We chose it because it seemed to be the dominant technology for Grid services. And we were interested in going the Web services direction. We haven't found a reason to revisit that decision.
- With the Grid technology we deliver the images from any clinical trial center into the radiologist's own review workstation. That's capable now with Grid technology. And that's a big reward for us to see that really happening. And the radiologists are very pleased with this. This actually engages more radiologists in clinical trials than before. So we can actually improve the quality and quantity of research being done

Appendix D Summary Data for Sections 5-8

D.1 HPC Scientist

D.2 HPC Domain-Specific Developer

D.2.1 Goals

- science-related goals
 - understand the physiological and metabolic processes of various genes
 - automatically detect network anomalies
 - enable interesting epidemiological questions to be answered
 - ensure radiology imaging workflow is flowing
 - detect gravitational waves and conduct gravitational wave astronomy
 - support research in the recovery from stroke
 - build a dynamically adaptive weather simulation system
- computation-related goals
 - interact with distributed computers to process data
 - run computational science models at high resolutions
 - analyze large volumes of genomic data
 - enable data analysis to refine anomaly-detection algorithms
 - get analysis pipelines running on multiple compute sites
- data-related goals
 - provide useful access to domain data, 100s timepoints in a series, each a large data recording from the brain,
 - enable data discovery, data access and synthesis of distributed datasets in the network
 - allow users to integrate different datasets so they can dynamically generate synthetic products
 - primary project involves a provenance aware synthesis tracking architecture
 - data warehouse fed by 26 spatially distributed sites collecting scientific data
 - architecture goal is pull data from the sites in a seamless way, based on metadata records and open access to the file
 - framework for neuroscientists can store, analyze, share, data in an effective manner
 - efficiently and compliantly move medical images in a secured fashion
 - allow communication of images on demand
 - patient comes to hospital, data can be accessed elsewhere; wherever you go all information is aggregated and available to health providers
 - make data available after it comes off the instruments to analysis sites
 - enable as much science as possible to be conducted with the data
 - enable scientists to efficiently use the data sites around the work to do analysis
 - dashboards for collaborators to see the current state of replication
 - once datasets become available that simulation and modeling with take place
 - see a bigger picture by making more data available
- general-purpose technology-related goals
 - integrate legacy fortran codes and apps
 - service oriented architecture
 - provide access to advanced computational resources
 - portal so users don't need to learn about cluster or lrm
 - integrate compute systems with atmospheric science models and instruments
 - build soa and grid middleware ourselves while trying to leverage tools available in the community

- once data are centralized we develop interfaces to enable users to explore the data
- developing web-based applications that include discovery interfaces, plotting routines, different data download mechanisms
- develop security model to ensure privacy; key to further exploring applications to the medical domain
- guarantee privacy
- use globus toolkit
- build wsrp-compliant services
- support user modalities and device incompatibilities
- provide scalable technical solution to allow a number of users to interact and share images
- demonstrate that technology available today like the globus toolkit can build a model for patient authz and privacy for data
- leap from science that takes place at the site to science that takes place at the national/global network scale
- get analyses running on sites external to grid
- mean time of 3 months btw failure for data replication side
- build infrastructure (tools, middleware, end-user tools, services and system) enabling scientists to efficiently analyze data and conduct research
- people-related goals
 - enable real-life modality to be supported by technology, providing patient with same controls irl
 - to make it easier for researchers to answer questions

D.2.3 Issues

1. Overcome sociological barriers within the domain

adoption, culture: sociological barriers to sharing medical data need to be overcome
 adoption, knowledge: domain experts do not understand the technical underpinning, so difficult to communicate benefits

2. Learn how to translate goals into a technical solution

adoption, knowledge: high learning curve for concepts
 adoption, knowledge: lack of technologically trained domain specialists
 adoption, knowledge: need more tutorial examples like in the GT4 programming java services book, such as examples on how to write clients to a gram service
 adoption, knowledge: the learning curve involved in using different software is a challenge
 adoption, knowledge: need to be tech savvy in order to construct grid solutions; no mechanism on TeraGrid or OSG to work out how to solve problems together
 adoption, knowledge: dearth of globus expertise in the community: difficult to find and hire them. so we need to train them, which requires management time

3. Component-level integration issues

adoption, integration: I stay away from rip-and-replace tools; can't use tools that force me to give up stuff I'm already doing
 adoption, integration: to integrate a technology I look at the interfaces; how easy I can glue the things together; right language? or do I need to go outside area of expertise
 adoption, integration: application people don't code against APIs; need environment with security managed for them; they won't link with a library, but expect the infrastructure to operate at a level below that
 adoption, integration: what kind of logging does it have? can I turn up logging levels so I can get a picture of what's going on?
 adoption, integration: need backend interfaces for security services to hook them in to other systems and services

adoption, integration: we can't make too many application level changes to integrate metadata and provenance features. the approach of "just do it this way" will not work with his scientists. they will do their science and all the other stuff has to be tacked on

adoption, overhead: if we have only a few nodes and want to launch scripts on them, doing it the globus way (with additional software layers and centralized key distribution) seems like a lot of work

diagnostics: error messages issued by globus services as a result of infrastructure failure sometimes do not contain sufficient information to automatically parse the messages and adapt the behavior of our system

technology: advanced configurations of globus services are not well-documented

technology: getting services correctly configured can be a challenge (runtime plus install time)

technology: must install specific version of a database driver; some drivers you can't get anymore because they're outdated

technology: 3rd party prereqs (like postgresSQL) can be frustrating; user likes MySQL

technology: machine architecture problems can be frustrating; e.g., 64-bit machines vs 32-bit

4. System-level integration issues

adoption, integration: high startup cost; cheaper to put 20 cpus behind a firewall with no way in except through a secure gateway node; then can run whatever I want using the easiest to use toolkits with the least overhead and least restrictions on how we code things up

adoption, integration: globus forces user to implement code in specific ways; can't take our own code and pop it in and just pick up security; in order to do something outside of the globus-provided services but staying within the security network, one needs to learn additional apis and the programming model

adoption, integration: as long as our project fits within the local cluster it is too much additional work to do it the globus way

adoption, overhead: generating each key by hand for each user, and distributing the signatures to each node to allow the user to log in; seems painful and complicated just to enable users to copy files and launch remote jobs

communications: don't need RPC calls or handshakes to be encrypted because I run on my own secure network

technology: credential mgmt by users too difficult, so relying on myproxy

adoption, overhead: I can foresee difficulties associated with centralized account mgmt if I were maintaining 70 Globus machines; key management seems very complicated

5. Provide a stable system for supporting the scientific workflow

reliability, service: gram should improve its response to hardware and network failure

reliability, service: we need services that enable us to provide our user community with stable services; so we need the production versions of the software (as opposed to development)

reliability, service: concern about user-written code that is fragile and difficult to maintain

reliability, system: more fault tolerance

reliability, system: these systems we're building depend on the information in order to function, but the information providers are breaking under the load; when they break work stops

reliability, system: we have so many things to keep track of we're losing the ability to track it

technology: data is being spread out over a couple thousand of files per day that we need to keep track of; we're getting killed by the metadata

reliability, system: monitoring tools don't always show whether remote services are really available or not

technology: gram4 goes into hibernation

adoption, integration: evaluate tools for whether or not they'll be stable and I can't rely on them

6. Troubleshooting

diagnostics: undocumented error codes; there is some documentation but not usable

diagnostics: troubleshooting right now requires knowledge about the internals of the software, so if expertise is missing on the remote site things don't get resolved quickly

diagnostics: error messages issued by globus can be misleading, such as saying "login incorrect" which might suggest a client problem but is usually a serverside problem

7. Encounter limitations of deployed infrastructure

adoption, infrastructure difficulties: can't move data quickly because gridftp client not available

adoption, infrastructure difficulties: can't go to typical globus machine (such as on the TeraGrid) and just run our code because we rely on external libraries that are not included in the default install
adoption, integration: a lot of libs and tools like R are not included in a standard globus install; so we ship our own precompiled code that refers to R installed in the home directory; but then we're missing the point of globus
adoption, overhead: time burdens associated with setting up the application-specific environment on remote machines is a big challenge
adoption, support burden: so much less time to use a local resource that's controlled by me compared with using a remote resource controlled by someone else like TeraGrid due to setup costs (account setup, scheduling on the resource, installing prerequisite software). Must either work with the admin of the remote sites to install things or devise instructions and setup for installing in their local directories.
technology: getting two gridftp servers to talk

8. Respond to changes in the technologies they integrate

adoption, integration: would rather not rebuild everything in system to accommodate changes between releases

adoption, integration: incompatibility is difficult for application developers

communications: difficulty monitoring so many project & technology email lists; difficult to keep track of globus workplans

D.3 General-Purpose HPC Infrastructure Provider

D.4 General-Purpose HPC Technology Developer

PRELIMINARY DRAFT

Appendix E Complete Interview Transcripts

[Full transcripts will be included in the final report]

- E.2  Troubleshooting requires knowledge about software internals
- E.3  The Grid is a black box to me
- E.4  The reason my tasks are so time-consuming is failure
- E.5  Performance improved from days to seconds
- E.6  The Grid idea is great, but there are barriers to making it work today
- E.7  If things don't work you need an expert to fix them
- E.8  I am trying to understand where Grid Computing adds value
- E.9  Globus enables more science
- E.10  Solving problems is easy once you have all the data
- E.11  Resource usage within our Virtual Organization is opportunistic
- E.12  The right approach is to be highly collaborative with domain specialists
- E.13  We play a strong bridge role in connecting people with technology
- E.14  I start with microbenchmarks and follow-up with real applications
- E.16  We assume a world where lightpaths can be scheduled between computers
- E.17  GRAM2 is kept alive by the need to interoperate with European experiments
- E.20  We can provide our users with fresh data more frequently because of the Grid
- E.21  We work to enable discovery, access and synthesis of distributed datasets
- E.26  Our goal is to make it easier to troubleshoot Grid applications
- E.27  The end goal is to automatically detect network anomalies