Introduction to Grid Computing and the Globus Toolkit™

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Outline

- Introduction to Grid Computing
- Some Definitions
- Grid Architecture
- The Programming Problem
- The Globus Toolkit™
  - Introduction, Security, Resource Management, Information Services, Data Management
- Related work
- Futures and Conclusions
The Grid Problem

- Flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resource
  
  From “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”
  
  - Enable communities (“virtual organizations”) to share geographically distributed resources as they pursue common goals **assuming the absence of**...
    - central location,
    - central control,
    - omniscience,
    - existing trust relationships.
Elements of the Problem

- **Resource sharing**
  - Computers, storage, sensors, networks, ...
  - Sharing always conditional: issues of trust, policy, negotiation, payment, ...

- **Coordinated problem solving**
  - Beyond client-server: distributed data analysis, computation, collaboration, ...

- **Dynamic, multi-institutional virtual orgs**
  - Community overlays on classic org structures
  - Large or small, static or dynamic
Why Grids?

- A biochemist exploits 10,000 computers to screen 100,000 compounds in an hour
- 1,000 physicists worldwide pool resources for petaop analyses of petabytes of data
- Civil engineers collaborate to design, execute, & analyze shake table experiments
- Climate scientists visualize, annotate, & analyze terabyte simulation datasets
- An emergency response team couples real time data, weather model, population data
Why Grids? (contd)

- A multidisciplinary analysis in aerospace couples code and data in four companies
- A home user invokes architectural design functions at an application service provider
- An application service provider purchases cycles from compute cycle providers
- Scientists working for a multinational soap company design a new product
- A community group pools members’ PCs to analyze alternative designs for a local road
Online Access to Scientific Instruments

Advanced Photon Source

real-time collection

tomographic reconstruction

wide-area dissemination

archival storage

desktop & VR clients with shared controls

DOE X-ray grand challenge: ANL, USC/ISI, NIST, U.Chicago

October 12, 2001 Intro to Grid Computing and Globus Toolkit™
Data Grids for High Energy Physics

There is a "bunch crossing" every 25 nsecs. There are 100 "triggers" per second. Each triggered event is ~1 MByte in size.

1 TIPS is approximately 25,000 SpecInt95 equivalents

Physicists work on analysis "channels". Each institute will have ~10 physicists working on one or more channels; data for these channels should be cached by the institute server.

Image courtesy Harvey Newman, Caltech

October 12, 2001
Intro to Grid Computing and Globus Toolkit™
Mathematicians Solve NUG30

- Looking for the solution to the NUG30 quadratic assignment problem
- An informal collaboration of mathematicians and computer scientists
- Condor-G delivered 3.46E8 CPU seconds in 7 days (peak 1009 processors) in U.S. and Italy (8 sites)

MetaNEOS: Argonne, Iowa, Northwestern, Wisconsin
Network for Earthquake Engineering Simulation

- NEESgrid: national infrastructure to couple earthquake engineers with experimental facilities, databases, computers, & each other
- On-demand access to experiments, data streams, computing, archives, collaboration

NEESgrid: Argonne, Michigan, NCSA, UIUC, USC
Home Computers
Evaluate AIDS Drugs

- Community =
  - 1000s of home computer users
  - Philanthropic computing vendor (Entropia)
  - Research group (Scripps)

- Common goal =
  advance AIDS research

Free Software for Your PC - By downloading Entropia onto your PC, FightAIDS@Home uses your computer's idle resources to accelerate powerful new anti-HIV drug design research!

FightAIDS@Home is a computational research project conducted by the Olson laboratory at The Scripps Research Institute in La Jolla, California. The project uses Entropia's global Internet computing grid, which runs both commercial and research applications on PCs.

How Your PC Helps - FightAIDS@Home uses your computer to generate and test millions of candidate drug compounds against detailed models of evolving HIV viruses, a feat previously impossible without dozens of multi-million dollar supercomputers. Every PC matters!
Broader Context

- "Grid Computing" has much in common with major industrial thrusts
  - Business-to-business, Peer-to-peer, Application Service Providers, Storage Service Providers, Distributed Computing, Internet Computing...

- Sharing issues not adequately addressed by existing technologies
  - Complicated requirements: “run program X at site Y subject to community policy P, providing access to data at Z according to policy Q”
  - High performance: unique demands of advanced & high-performance systems
Why Now?

- Moore’s law improvements in computing produce highly functional endsystems
- The Internet and burgeoning wired and wireless provide universal connectivity
- Changing modes of working and problem solving emphasize teamwork, computation
- Network exponentials produce dramatic changes in geometry and geography
Network Exponentials

- Network vs. computer performance
  - Computer speed doubles every 18 months
  - Network speed doubles every 9 months
  - Difference = order of magnitude per 5 years

- 1986 to 2000
  - Computers: x 500
  - Networks: x 340,000

- 2001 to 2010
  - Computers: x 60
  - Networks: x 4000

Moore's Law vs. storage improvements vs. optical improvements. Graph from Scientific American (Jan-2001) by Cleo Vilett, source Vined Khoslan, Kleiner, Caufield and Perkins.
The Globus Project™
Making Grid computing a reality

- Close collaboration with real Grid projects in science and industry
- Development and promotion of standard Grid protocols to enable interoperability and shared infrastructure
- Development and promotion of standard Grid software APIs and SDKs to enable portability and code sharing
- The Globus Toolkit™: Open source, reference software base for building grid infrastructure and applications
- Global Grid Forum: Development of standard protocols and APIs for Grid computing
### Selected Major Grid Projects

<table>
<thead>
<tr>
<th>Name</th>
<th>URL &amp; Sponsors</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Grid</td>
<td><a href="http://www.mcs.anl.gov/FL/accessgrid">www.mcs.anl.gov/FL/accessgrid</a>; DOE, NSF</td>
<td>Create &amp; deploy group collaboration systems using commodity technologies</td>
</tr>
<tr>
<td>BlueGrid</td>
<td>IBM</td>
<td>Grid testbed linking IBM laboratories</td>
</tr>
<tr>
<td>DISCOM</td>
<td><a href="http://www.cs.sandia.gov/discom">www.cs.sandia.gov/discom</a></td>
<td>Create operational Grid providing access to resources at three U.S. DOE weapons laboratories</td>
</tr>
<tr>
<td>DOE Science Grid</td>
<td>sciencegrid.org</td>
<td>Create operational Grid providing access to resources &amp; applications at U.S. DOE science laboratories &amp; partner universities</td>
</tr>
<tr>
<td>Earth System Grid (ESG)</td>
<td>earthsystemgrid.org</td>
<td>Delivery and analysis of large climate model datasets for the climate research community</td>
</tr>
<tr>
<td>European Union (EU) DataGrid</td>
<td>eu-datagrid.org</td>
<td>Create &amp; apply an operational grid for applications in high energy physics, environmental science, bioinformatics</td>
</tr>
</tbody>
</table>

*New projects are indicated by the "New" label.*
## Selected Major Grid Projects

<table>
<thead>
<tr>
<th>Name</th>
<th>URL/Sponsor</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>EuroGrid, Grid Interoperability (GRIP)</td>
<td>eurogrid.org</td>
<td>Create tech for remote access to supercomp resources &amp; simulation codes; in GRIP, integrate with Globus Toolkit™</td>
</tr>
<tr>
<td></td>
<td>European Union</td>
<td></td>
</tr>
<tr>
<td>Fusion Collaboratory</td>
<td>fusiongrid.org</td>
<td>Create a national computational collaboratory for fusion research</td>
</tr>
<tr>
<td></td>
<td>DOE Off. Science</td>
<td></td>
</tr>
<tr>
<td>Globus Project™</td>
<td>globus.org</td>
<td>Research on Grid technologies; development and support of Globus Toolkit™; application and deployment</td>
</tr>
<tr>
<td></td>
<td>DARPA, DOE, NSF, NASA, Msoft</td>
<td></td>
</tr>
<tr>
<td>GridLab</td>
<td>gridlab.org</td>
<td>Grid technologies and applications</td>
</tr>
<tr>
<td></td>
<td>European Union</td>
<td></td>
</tr>
<tr>
<td>GridPP</td>
<td>gridpp.ac.uk</td>
<td>Create &amp; apply an operational grid within the U.K. for particle physics research</td>
</tr>
<tr>
<td></td>
<td>U.K. eScience</td>
<td></td>
</tr>
<tr>
<td>Grid Research Integration Dev. &amp; Support Center</td>
<td>grids-center.org</td>
<td>Integration, deployment, support of the NSF Middleware Infrastructure for research &amp; education</td>
</tr>
<tr>
<td></td>
<td>NSF</td>
<td></td>
</tr>
</tbody>
</table>
## Selected Major Grid Projects

<table>
<thead>
<tr>
<th>Name</th>
<th>URL/Sponsor</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid Application Dev. Software</td>
<td>hipersoft.rice.edu/grads; NSF</td>
<td>Research into program development technologies for Grid applications</td>
</tr>
<tr>
<td>Grid Physics Network</td>
<td>griphyn.org NSF</td>
<td>Technology R&amp;D for data analysis in physics expts: ATLAS, CMS, LIGO, SDSS</td>
</tr>
<tr>
<td>Information Power Grid</td>
<td>ipg.nasa.gov NASA</td>
<td>Create and apply a production Grid for aerosciences and other NASA missions</td>
</tr>
<tr>
<td>International Virtual Data Grid Laboratory</td>
<td>ivdgl.org NSF</td>
<td>Create international Data Grid to enable large-scale experimentation on Grid technologies &amp; applications</td>
</tr>
<tr>
<td>Network for Earthquake Eng. Simulation Grid</td>
<td>neesgrid.org NSF</td>
<td>Create and apply a production Grid for earthquake engineering</td>
</tr>
<tr>
<td>Particle Physics Data Grid</td>
<td>ppdg.net DOE Science</td>
<td>Create and apply production Grids for data analysis in high energy and nuclear physics experiments</td>
</tr>
</tbody>
</table>
### Selected Major Grid Projects

<table>
<thead>
<tr>
<th>Name</th>
<th>URL/Sponsor</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>TeraGrid</td>
<td>teragrid.org</td>
<td>U.S. science infrastructure linking four major resource sites at 40 Gb/s</td>
</tr>
<tr>
<td></td>
<td>NSF</td>
<td></td>
</tr>
<tr>
<td>UK Grid Support Center</td>
<td>grid-support.ac.uk</td>
<td>Support center for Grid projects within the U.K.</td>
</tr>
<tr>
<td></td>
<td>U.K. eScience</td>
<td></td>
</tr>
<tr>
<td>Unicore</td>
<td>BMBFT</td>
<td>Technologies for remote access to supercomputers</td>
</tr>
</tbody>
</table>

Also many technology R&D projects: e.g., Condor, NetSolve, Ninf, NWS

See also www.gridforum.org
The 13.6 TF TeraGrid: Computing at 40 Gb/s
iVDGL: International Virtual Data Grid Laboratory

U.S. PIs: Avery, Foster, Gardner, Newman, Szalay

www.ivdgl.org

October 12, 2001
Intro to Grid Computing and Globus Toolkit™
For More Information

- Globus Project™
  - www.globus.org
- Grid Forum
  - www.gridforum.org
- Book (Morgan Kaufman)
  - www.mkp.com/grids
Some Definitions

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Some Important Definitions

- Resource
- Network protocol
- Network enabled service
- Application Programmer Interface (API)
- Software Development Kit (SDK)
- Syntax

- Not discussed, but important: policies
Resource

- An entity that is to be shared
  - E.g., computers, storage, data, software
- Does **not** have to be a physical entity
  - E.g., Condor pool, distributed file system, ...
- Defined in terms of interfaces, not devices
  - E.g. scheduler such as LSF and PBS define a compute resource
  - Open/close/read/write define access to a distributed file system, e.g. NFS, AFS, DFS
Network Protocol

- A formal description of message formats and a set of rules for message exchange
  - Rules may define sequence of message exchanges
  - Protocol may define state-change in endpoint, e.g., file system state change
- Good protocols designed to do one thing
  - Protocols can be layered
- Examples of protocols
  - IP, TCP, TLS (was SSL), HTTP, Kerberos
Network Enabled Services

- Implementation of a protocol that defines a set of capabilities
  - Protocol defines interaction with service
  - All services require protocols
  - Not all protocols are used to provide services (e.g. IP, TLS)

- Examples: FTP and Web servers

<table>
<thead>
<tr>
<th>FTP Server</th>
<th>Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP Protocol</td>
<td>HTTP Protocol</td>
</tr>
<tr>
<td>Telnet Protocol</td>
<td>TLS Protocol</td>
</tr>
<tr>
<td>TCP Protocol</td>
<td>TCP Protocol</td>
</tr>
<tr>
<td>IP Protocol</td>
<td>IP Protocol</td>
</tr>
</tbody>
</table>
Application Programming Interface

- A specification for a set of routines to facilitate application development
  - Refers to definition, not implementation
  - E.g., there are many implementations of MPI
- Spec often language-specific (or IDL)
  - Routine name, number, order and type of arguments; mapping to language constructs
  - Behavior or function of routine
- Examples
  - GSS API (security), MPI (message passing)
Software Development Kit

- A particular instantiation of an API
- SDK consists of libraries and tools
  - Provides implementation of API specification
- Can have multiple SDKs for an API
- Examples of SDKs
  - MPICH, Motif Widgets
Syntax

- Rules for encoding information, e.g.
  - XML, Condor ClassAds, Globus RSL
  - X.509 certificate format (RFC 2459)
  - Cryptographic Message Syntax (RFC 2630)
- Distinct from protocols
  - One syntax may be used by many protocols (e.g., XML); & useful for other purposes
- Syntaxes may be layered
  - E.g., Condor ClassAds -> XML -> ASCII
  - Important to understand layerings when comparing or evaluating syntaxes
A Protocol can have Multiple APIs

- TCP/IP APIs include BSD sockets, Winsock, System V streams, ...
- The protocol provides interoperability: programs using different APIs can exchange information
- I don’t need to know remote user’s API
An API can have Multiple Protocols

- MPI provides portability: any correct program compiles & runs on a platform
- Does not provide interoperability: all processes must link against same SDK
  - E.g., MPICH and LAM versions of MPI

![Diagram showing different message formats and exchange sequences]
APIs and Protocols are Both Important

- **Standard APIs/SDKs are important**
  - They enable application *portability*
  - But w/o standard protocols, interoperability is hard (every SDK speaks every protocol?)

- **Standard protocols are important**
  - Enable cross-site *interoperability*
  - Enable shared infrastructure
  - But w/o standard APIs/SDKs, application portability is hard (different platforms access protocols in different ways)
Grid Architecture

The Globus Project™

Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Why Discuss Architecture?

- **Descriptive**
  - Provide a common vocabulary for use when describing Grid systems

- **Guidance**
  - Identify key areas in which services are required

- **Prescriptive**
  - Define standard “Intergrid” protocols and APIs to facilitate creation of interoperable Grid systems and portable applications
One View of Requirements

- Identity & authentication
- Authorization & policy
- Resource discovery
- Resource characterization
- Resource allocation
- (Co-)reservation, workflow
- Distributed algorithms
- Remote data access
- High-speed data transfer
- Performance guarantees
- Monitoring

- Adaptation
- Intrusion detection
- Resource management
- Accounting & payment
- Fault management
- System evolution
- Etc.
- Etc.
- ...

Another View: “Three Obstacles to Making Grid Computing Routine”

1) New approaches to problem solving
   - Data Grids, distributed computing, peer-to-peer, collaboration grids, ...

2) Structuring and writing programs
   - Abstractions, tools

3) Enabling resource sharing across distinct institutions
   - Resource discovery, access, reservation, allocation; authentication, authorization, policy; communication; fault detection and notification; ...
Programming & Systems Problems

- The programming problem
  - Facilitate development of sophisticated apps
  - Facilitate code sharing
  - Requires **prog. envs**: APIs, SDKs, tools

- The systems problem
  - Facilitate coordinated use of diverse resources
  - Facilitate infrastructure sharing: e.g., certificate authorities, info services
  - Requires **systems**: protocols, services
  - E.g., port/service/protocol for accessing information, allocating resources
The Systems Problem: Resource Sharing Mechanisms That ...

- Address security and policy concerns of resource owners and users
- Are flexible enough to deal with many resource types and sharing modalities
- Scale to large number of resources, many participants, many program components
- Operate efficiently when dealing with large amounts of data & computation
Aspects of the Systems Problem

1) Need for **interoperability** when different groups want to share resources
   - Diverse components, policies, mechanisms
   - E.g., standard notions of identity, means of communication, resource descriptions

2) Need for **shared infrastructure services** to avoid repeated development, installation
   - E.g., one port/service/protocol for remote access to computing, not one per tool/appln
   - E.g., Certificate Authorities: expensive to run

   - A common need for **protocols & services**
Hence, a Protocol-Oriented View of Grid Architecture, that Emphasises …

- Development of **Grid protocols & services**
  - Protocol-mediated access to remote resources
  - New services: e.g., resource brokering
  - “On the Grid” = speak Intergrid protocols
  - Mostly (extensions to) existing protocols

- Development of **Grid APIs & SDKs**
  - Interfaces to Grid protocols & services
  - Facilitate application development by supplying higher-level abstractions

- The (hugely successful) model is the Internet
Layered Grid Architecture
(By Analogy to Internet Architecture)

“Coordinating multiple resources”: ubiquitous infrastructure services, app-specific distributed services

“Sharing single resources”: negotiating access, controlling use

“Talking to things”: communication (Internet protocols) & security

“Controlling things locally”: Access to, & control of, resources
Protocols, Services, and APIs Occur at Each Level

Applications

Languages/Frameworks

Collective Service APIs and SDKs

Collective Services

Resource APIs and SDKs

Resource Services

Connectivity APIs

Connectivity Protocols

Fabric Layer

Local Access APIs and Protocols
Important Points

- Built on Internet protocols & services
  - Communication, routing, name resolution, etc.
- “Layering” here is conceptual, does not imply constraints on who can call what
  - Protocols/services/APIs/SDKs will, ideally, be largely self-contained
  - Some things are fundamental: e.g., communication and security
  - But, advantageous for higher-level functions to use common lower-level functions
The Hourglass Model

- **Focus on architecture issues**
  - Propose set of core services as basic infrastructure
  - Use to construct high-level, domain-specific solutions

- **Design principles**
  - Keep participation cost low
  - Enable local control
  - Support for adaptation
  - “IP hourglass” model

![Diagram of the Hourglass Model](image)
Where Are We With Architecture?

- No “official” standards exist
- But:
  - Globus Toolkit™ has emerged as the de facto standard for several important Connectivity, Resource, and Collective protocols
  - GGF has an architecture working group
  - Technical specifications are being developed for architecture elements: e.g., security, data, resource management, information
  - Internet drafts submitted in security area
Fabric Layer
Protocols & Services

- Just what you would expect: the diverse mix of resources that may be shared
  - Individual computers, Condor pools, file systems, archives, metadata catalogs, networks, sensors, etc., etc.
- Few constraints on low-level technology: connectivity and resource level protocols form the “neck in the hourglass”
- Defined by interfaces not physical characteristics
Connectivity Layer
Protocols & Services

- Communication
  - Internet protocols: IP, DNS, routing, etc.

- Security: Grid Security Infrastructure (GSI)
  - Uniform authentication, authorization, and message protection mechanisms in multi-institutional setting
  - Single sign-on, delegation, identity mapping
  - Public key technology, SSL, X.509, GSS-API
  - Supporting infrastructure: Certificate Authorities, certificate & key management, ...

GSI: www.gridforum.org/security
Resource Layer
Protocols & Services

- **Grid Resource Allocation Mgmt (GRAM)**
  - Remote allocation, reservation, monitoring, control of compute resources
- **GridFTP protocol (FTP extensions)**
  - High-performance data access & transport
- **Grid Resource Information Service (GRIS)**
  - Access to structure & state information
- **Network reservation, monitoring, control**
- **All built on connectivity layer: GSI & IP**

GridFTP: [www.gridforum.org](http://www.gridforum.org)
GRAM, GRIS: [www.globus.org](http://www.globus.org)
Collective Layer
Protocols & Services

- Index servers aka metadirectory services
  - Custom views on dynamic resource collections assembled by a community
- Resource brokers (e.g., Condor Matchmaker)
  - Resource discovery and allocation
- Replica catalogs
- Replication services
- Co-reservation and co-allocation services
- Workflow management services
- Etc.

Condor: www.cs.wisc.edu/condor
Example: High-Throughput Computing System

App
- High Throughput Computing System

Collective (App)
- Dynamic checkpoint, job management, failover, staging

Collective (Generic)
- Brokering, certificate authorities

Resource
- Access to data, access to computers, access to network performance data

Connect
- Communication, service discovery (DNS), authentication, authorization, delegation

Fabric
- Storage systems, schedulers
Example:
Data Grid Architecture

| Collective | Coherency control, replica selection, task management, virtual data catalog, virtual data code catalog, ...
| Collective | Replica catalog, replica management, co-allocation, certificate authorities, metadata catalogs,
| Resource   | Access to data, access to computers, access to network performance data, ...
| Connect    | Communication, service discovery (DNS), authentication, authorization, delegation
| Fabric     | Storage systems, clusters, networks, network caches, ...

Discipline-Specific Data Grid Application
The Programming Problem

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
The Programming Problem

- But how do I develop robust, secure, long-lived, well-performing applications for dynamic, heterogeneous Grids?
- I need, presumably:
  - Abstractions and models to add to speed/robustness/etc. of development
  - Tools to ease application development and diagnose common problems
  - Code/tool sharing to allow reuse of code components developed by others
Grid Programming Technologies

- "Grid applications" are incredibly diverse (data, collaboration, computing, sensors, ...)
  - Seems unlikely there is one solution
- Most applications have been written "from scratch," with or without Grid services
- Application-specific libraries have been shown to provide significant benefits
- No new language, programming model, etc., has yet emerged that transforms things
  - But certainly still quite possible
Examples of Grid Programming Technologies

- MPICH-G2: Grid-enabled message passing
- CoG Kits, GridPort: Portal construction, based on N-tier architectures
- GDMP, Data Grid Tools, SRB: replica management, collection management
- Condor-G: workflow management
- Legion: object models for Grid computing
- Cactus: Grid-aware numerical solver framework
  - Note tremendous variety, application focus
MPICH-G2: A Grid-Enabled MPI

- A complete implementation of the Message Passing Interface (MPI) for heterogeneous, wide area environments
  - Based on the Argonne MPICH implementation of MPI (Gropp and Lusk)
- Requires services for authentication, resource allocation, executable staging, output, etc.
- Programs run in wide area without change
- See also: MetaMPI, PACX, STAMPI, MAGPIE

www.globus.org/mpi
Cactus
(Allen, Dramlitsch, Seidel, Shalf, Radke)

- Modular, portable framework for parallel, multidimensional simulations
- Construct codes by linking
  - Small core (flesh): mgmt services
  - Selected modules (thorns): Numerical methods, grids & domain decomp, visualization and steering, etc.
- Custom linking/configuration tools
- Developed for astrophysics, but not astrophysics-specific

www.cactuscode.org
High-Throughput Computing and Condor

- High-throughput computing
  - CPU cycles/day (week, month, year?) under non-ideal circumstances
  - “How many times can I run simulation X in a month using all available machines?”

- Condor converts collections of distributively owned workstations and dedicated clusters into a distributed high-throughput computing facility

- Emphasis on policy management and reliability

www.cs.wisc.org/condor
Object-Based Approaches

- **Grid-enabled CORBA**
  - NASA Lewis, Rutgers, ANL, others
  - CORBA wrappers for Grid protocols
  - Some initial successes

- **Legion**
  - U.Virginia
  - Object models for Grid components (e.g., “vault”=storage, “host”=computer)
Portals

- N-tier architectures enabling thin clients, with middle tiers using Grid functions
  - Thin clients = web browsers
  - Middle tier = e.g. Java Server Pages, with Java CoG Kit, GPDK, GridPort utilities
  - Bottom tier = various Grid resources
- Numerous applications and projects, e.g.
  - Unicore, Gateway, Discover, Mississippi Computational Web Portal, NPACI Grid Port, Lattice Portal, Nimrod-G, Cactus, NASA IPG Launchpad, Grid Resource Broker, ...
Common Toolkit Underneath

- Each of these programming environments should not have to implement the protocols and services from scratch!
- Rather, want to share common code that...
  - Implements core functionality
    - SDKs that can be used to construct a large variety of services and clients
    - Standard services that can be easily deployed
  - Is robust, well-architected, self-consistent
  - Is open source, with broad input
- Which leads us to the Globus Toolkit™...
The Globus Toolkit™: Introduction

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Globus Toolkit™

- A software toolkit addressing key technical problems in the development of Grid enabled tools, services, and applications
  - Offer a modular “bag of technologies”
  - Enable *incremental* development of grid-enabled tools and applications
  - Implement standard Grid protocols and APIs
  - Make available under liberal open source license
General Approach

• Define Grid protocols & APIs
  – Protocol-mediated access to remote resources
  – Integrate and extend existing standards
  – “On the Grid” = speak “Intergrid” protocols

• Develop a reference implementation
  – Open source Globus Toolkit
  – Client and server SDKs, services, tools, etc.

• Grid-enable wide variety of tools
  – Globus Toolkit, FTP, SSH, Condor, SRB, MPI, ...

• Learn through deployment and applications
Four Key Protocols

- The Globus Toolkit™ centers around four key protocols
  - Connectivity layer:
    > Security: Grid Security Infrastructure (GSI)
  - Resource layer:
    > Resource Management: Grid Resource Allocation Management (GRAM)
    > Information Services: Grid Resource Information Protocol (GRIP)
    > Data Transfer: Grid File Transfer Protocol (GridFTP)
Three Types of API/SDK

1) Portability and convenience API/SDKs
2) API/SDKs implementing the four key Connectivity and Resource layer protocols
3) Collective layer API/SDKs

- This tutorial focuses primarily on the functionality available in #2 and #3
- Developer tutorial included in depth API discussions of all three
Portability and Convenience API

- **globus_common**
  - Module activation/deactivation
  - Threads, mutual exclusion, conditions
  - Callback/event driver
  - Libc wrappers
  - Convenience modules (list, hash, etc).
Connectivity APIs

- **globus_io**
  - TCP, UDP, IP multicast, and file I/O
  - Integrates GSI security
  - Asynchronous and synchronous interfaces
  - Attribute based control of behavior

- **Nexus (Deprecated)**
  - Higher level, active message style comms
  - Built on globus_io, but without security

- **MPICH-G2**
  - High level, MPI (send/receive) interface
  - Built on globus_io and native MPI
The Globus Toolkit™: Security Services

The Globus Project™

Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Security Terminology

- Authentication: Establishing identity
- Authorization: Establishing rights
- Message protection
  - Message integrity
  - Message confidentiality
- Non-repudiation
- Digital signature
- Accounting
- Certificate Authority (CA)
GSI in Action

“Create Processes at A and B that Communicate & Access Files at C”

Single sign-on via “grid-id” & generation of proxy cred.
Or: retrieval of proxy cred. from online repository

Remote process creation requests*

Authorize Map to local id
Create process
Generate credentials

Communication*

Remote file access request*

* With mutual authentication
Why Grid Security is Hard

- Resources being used may be valuable & the problems being solved sensitive
- Resources are often located in distinct administrative domains
  - Each resource has own policies & procedures
- Set of resources used by a single computation may be large, dynamic, and unpredictable
  - Not just client/server, requires delegation
- It must be broadly available & applicable
  - Standard, well-tested, well-understood protocols; integrated with wide variety of tools
Grid Security Requirements

User View
1) Easy to use
2) Single sign-on
3) Run applications
   ftp, ssh, MPI, Condor, Web, ...
4) User based trust model
5) Proxies/agents (delegation)

Resource Owner View
1) Specify local access control
2) Auditing, accounting, etc.
3) Integration w/ local system
   Kerberos, AFS, license mgr.
4) Protection from compromised resources

Developer View
API/SDK with authentication, flexible message protection,
flexible communication, delegation, ...
   Direct calls to various security functions (e.g. GSS-API)
   Or security integrated into higher-level SDKs:
   E.g. GlobusIO, Condor-G, MPICH-G2, HDF5, etc.
Candidate Standards

- **Kerberos 5**
  - Fails to meet requirements:
    > Integration with various local security solutions
    > User based trust model

- **Transport Layer Security (TLS/SSL)**
  - Fails to meet requirements:
    > Single sign-on
    > Delegation
Grid Security Infrastructure (GSI)

- Extensions to standard protocols & APIs
  - Standards: SSL/TLS, X.509 & CA, GSS-API
  - Extensions for single sign-on and delegation
- Globus Toolkit reference implementation of GSI
  - SSLeay/OpenSSL + GSS-API + SSO/delegation
  - Tools and services to interface to local security
    > Simple ACLs; SSLK5/PKINIT for access to K5, AFS; …
    > Tools for credential management
      > Login, logout, etc.
      > Smartcards
      > MyProxy: Web portal login and delegation
      > K5cert: Automatic X.509 certificate creation
Review of Public Key Cryptography

- **Asymmetric keys**
  - A `private` key is used to encrypt data.
  - A `public` key can decrypt data encrypted with the private key.

- **An X.509 certificate includes...**
  - Someone’s subject name (user ID)
  - Their public key
  - A “signature” from a Certificate Authority (CA) that:
    - Proves that the certificate came from the CA.
    - Vouches for the subject name
    - Vouches for the binding of the public key to the subject
Public Key Based Authentication

- User sends certificate over the wire.
- Other end sends user a challenge string.
- User encodes the challenge string with private key
  - Possession of private key means you can authenticate as subject in certificate
- Public key is used to decode the challenge.
  - If you can decode it, you know the subject
- Treat your private key carefully!!
  - Private key is stored only in well-guarded places, and only in encrypted form
X.509 Proxy Certificate

- Defines how a short term, restricted credential can be created from a normal, long-term X.509 credential
  - A “proxy certificate” is a special type of X.509 certificate that is signed by the normal end entity cert, or by another proxy
  - Supports single sign-on & delegation through “impersonation”
  - Currently an IETF draft
User Proxies

- Minimize exposure of user’s private key
- A temporary, X.509 proxy credential for use by our computations
  - We call this a user proxy certificate
  - Allows process to act on behalf of user
  - User-signed user proxy cert stored in local file
  - Created via “grid-proxy-init” command
- Proxy’s private key is not encrypted
  - Rely on file system security, proxy certificate file must be readable only by the owner
Delegation

- Remote creation of a user proxy
- Results in a new private key and X.509 proxy certificate, signed by the original key
- Allows remote process to act on behalf of the user
- Avoids sending passwords or private keys across the network
Globus Security APIs

- **Generic Security Service (GSS) API**
  - IETF standard
  - Provides functions for authentication, delegation, message protection
  - Decoupled from any particular communication method

- But GSS-API is somewhat complicated, so we also provide the easier-to-use `globus_gss_assist` API.

- GSI-enabled SASL is also provided
Results

- GSI adopted by 100s of sites, 1000s of users
  - Globus CA has issued >3000 certs (user & host), >1500 currently active; other CAs active
- Rollouts are currently underway all over:
  - NSF Teragrid, NASA Information Power Grid, DOE Science Grid, European Data Grid, etc.
- Integrated in research & commercial apps
  - GrADS testbed, Earth Systems Grid, European Data Grid, GriPhyN, NEESgrid, etc.
- Standardization begun in Global Grid Forum, IETF
GSI Applications

- Globus Toolkit™ uses GSI for authentication
- Many Grid tools, directly or indirectly, e.g.
  - Condor-G, SRB, MPICH-G2, Cactus, GDMP, ...
- Commercial and open source tools, e.g.
  - ssh, ftp, cvs, OpenLDAP, OpenAFS
  - SecureCRT (Win32 ssh client)
- And since we use standard X.509 certificates, they can also be used for
  - Web access, LDAP server access, etc.
Ongoing and Future GSI Work

- Protection against compromised resources
  - Restricted delegation, smartcards
- Standardization
- Scalability in numbers of users & resources
  - Credential management
  - Online credential repositories ("MyProxy")
  - Account management
- Authorization
  - Policy languages
  - Community authorization
Restricted Proxies

- **Q:** How to restrict rights of delegated proxy to a subset of those associated with the issuer?
- **A:** Embed restriction policy in proxy cert
  - Policy is evaluated by resource upon proxy use
  - Reduces rights available to the proxy to a subset of those held by the user
- **But how to avoid policy language wars?**
  - Proxy cert just contains a container for a policy specification, without defining the language
    > Container = OID + blob
  - Can evolve policy languages over time
Delegation Tracing

- Often want to know through what entities a proxy certificate has been delegated
  - Audit (retrace footsteps)
  - Authorization (deny from bad entities)
- Solved by adding information to the signed proxy certificate about each entity to which a proxy is delegated.
  - Does NOT guarantee proper use of proxy
  - Just tells you which entities were purposely involved in a delegation
Proxy Certificate Standards Work

- "Internet Public Key Infrastructure X.509 Proxy Certificate Profile"
  - draft-ietf-pkix-proxy-01.txt
    > Draft being considered by IETF PKIX working group, and by GGF GSI working group
  - Defines proxy certificate format, including restricted rights and delegation tracing

- Demonstrated a prototype of restricted proxies at HPDC (August 2001) as part of CAS demo
Delegation Protocol Work

- "TLS Delegation Protocol"
  - draft-ietf-tls-delegation-01.txt
    - Draft being considered by IETF TLS working group, and by GGF GSI working group
    - Defines how to remotely delegate an X.509 Proxy Certificate using extensions to the TLS (SSL) protocol
- But, may change approach here
  - Instead of embedding into TLS, carry on top of TLS
  - This is the current approach in Globus Toolkit
GSS-API Extensions Work

• 4 years of GSS-API experience, while on the whole quite positive, has shed light on various deficiencies of GSS-API

• “GSS-API Extensions”
  - draft-ggf-gss-extensions-04.txt
    > Draft being considered by GGF GSI working group. Not yet submitted to IETF.
  - Defines extensions to the GSS-API to better support Grid security
GSS-API Extensions

- **Credential export/import**
  - Allows delegated credentials to be externalized
  - Used for checkpointing a service

- **Delegation at any time, in either direction**
  - More rich options on use of delegation

- **Restricted delegation handling**
  - Add proxy restrictions to delegated cred
  - Inspect auth cert for restrictions

- **Allow better mapping of GSS to TLS**
  - Support TLS framing of messages
Community Authorization Service

- Question: How does a large community grant its users access to a large set of resources?
  - Should minimize burden on both the users and resource providers

- Community Authorization Service (CAS)
  - Community negotiates access to resources
  - Resource outsources fine-grain authorization to CAS
  - Resource only knows about “CAS user” credential
    - CAS handles user registration, group membership...
  - User who wants access to resource asks CAS for a capability credential
    - Restricted proxy of the “CAS user” cred., checked by resource
Community Authorization (Prototype shown August 2001)

1. CAS request, with resource names and operations
2. CAS reply, with capability and resource CA info
3. Resource request, authenticated with capability
4. Resource reply
Community Authorization Service

- CAS provides user community with information needed to authenticate resources
  - Sent with capability credential, used on connection with resource
  - Resource identity (DN), CA
- This allows new resources/users (and their CAs) to be made available to a community through the CAS without action on the other user’s/resource’s part
Authorization API

- Service providers need to perform authorization policy evaluation on:
  - Local policies
  - Policies contained in restricted proxies
- We are working on 2 API layers:
  - Low level GAA-API implementation for evaluation of policies
  - High level, very simple authorization API that can easily be embedded into services
- Still in early prototyping stage
Passport Online CA & MyProxy

- Requiring users to manage their own certs and keys is annoying and error prone
- A solution: Leverage Passport global authentication to obtain a proxy credential
  - Passport provides
    > Globally unique user name (email address)
    > Method of verifying ownership of the name (authentication)
    > Re-issuance (e.g. forgotten password)
  - Passport credentials can be presented to an online CA or credential repository
    > Creates and issues new (restricted) proxy certificate to the user on demand
Other Future Security Work

- **Ease-of-use**
  - Improved error message, online CA, etc.
- **Improved online credential repositories**
  - See MyProxy paper at HPDC
- **Support for multiple user credentials**
- **Multi-factor authentication**
- **Subordinate certificate authorities for domains**
  - Ease issuance of host certs for domains
- **Independent Data Unit Support**
Security Summary

- GSI successfully addresses wide variety of Grid security issues
- Broad acceptance, deployment, integration with tools
- Standardization on-going in IETF & GGF
- Ongoing R&D to address next set of issues
- For more information:
  - www.globus.org/research/papers.html
    > “A Security Architecture for Computational Grids”
    > “Design and Deployment of a National-Scale Authentication Infrastructure”
  - www.gridforum.org/security
The Globus Toolkit™: Resource Management Services

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
The Challenge

- Enabling secure, controlled remote access to heterogeneous computational resources and management of remote computation
  - Authentication and authorization
  - Resource discovery & characterization
  - Reservation and allocation
  - Computation monitoring and control
- Addressed by new protocols & services
  - GRAM protocol as a basic building block
  - Resource brokering & co-allocation services
  - GSI for security, MDS for discovery
Resource Management

- The Grid Resource Allocation Management (GRAM) protocol and client API allows programs to be started on remote resources, despite local heterogeneity.
- Resource Specification Language (RSL) is used to communicate requirements.
- A layered architecture allows application-specific resource brokers and co-allocators to be defined in terms of GRAM services.
  - Integrated with Condor, PBS, MPICH-G2, ...
Resource Management Architecture

- Application
  - RSL
  - Ground RSL
- Broker
  - RSL specialization
  - Queries & Info
- Information Service
- Co-allocator
  - Simple ground RSL
- Local resource managers
  - GRAM
  - LSF
  - GRAM
  - Condor
  - GRAM
  - NQE
Resource Specification Language

- Common notation for exchange of information between components
  - Syntax similar to MDS/LDAP filters
- RSL provides two types of information:
  - Resource requirements: Machine type, number of nodes, memory, etc.
  - Job configuration: Directory, executable, args, environment
- Globus Toolkit provides an API/SDK for manipulating RSL
RSL Syntax

- **Elementary form: parenthesis clauses**
  - `(attribute op value [ value ... ] )`

- **Operators Supported:**
  - `<, <=, =, >=, > , !=`

- **Some supported attributes:**
  - executable, arguments, environment, stdin, stdout, stderr, resourceManagerContact, resourceManagerName

- **Unknown attributes are passed through**
  - May be handled by subsequent tools
Constraints: “&”

- For example:
  
  \[ & (\text{count} \geq 5) (\text{count} \leq 10) \\
  (\text{max\_time}=240) (\text{memory} \geq 64) \\
  (\text{executable}=\text{myprog}) \]

- “Create 5-10 instances of \text{myprog}, each on a machine with at least 64 MB memory that is available to me for 4 hours”
Disjunction: “|”

- For example:
  & (executable=myprog)
  ( | (&(count=5)(memory>=64))
   (&(count=10)(memory>=32)))
- Create 5 instances of myprog on a machine that has at least 64MB of memory, or 10 instances on a machine with at least 32MB of memory
GRAM Protocol

- **GRAM-1**: Simple HTTP-based RPC
  - Job request
    > Returns a “job contact”: Opaque string that can be passed between clients, for access to job
  - Job cancel, status, signal
  - Event notification (callbacks) for state changes
    > Pending, active, done, failed, suspended

- **GRAM-1.5 (U Wisconsin contribution)**
  - Add reliability improvements
    > Once-and-only-once submission
    > Recoverable job manager service
    > Reliable termination detection

- **GRAM-2**: Moving to Web Services (SOAP)...
Globus Toolkit Implementation

- **Gatekeeper**
  - Single point of entry
  - Authenticates user, maps to local security environment, runs service
  - In essence, a “secure inetd”

- **Job manager**
  - A gatekeeper service
  - Layers on top of local resource management system (e.g., PBS, LSF, etc.)
  - Handles remote interaction with the job
GRAM Components

Client

MDS client API calls to locate resources

MDS client API calls to get resource info

GRAM client API calls to request resource allocation and process creation.

Grid Security Infrastructure

MDS: Grid Index Info Server

Local Resource Manager

MDS: Grid Resource Info Server

Job Manager

Request

Allocate & create processes

RSL Library

Process

Monitor & control

Parse

Create

Gatekeeper

Query current status of resource

Site boundary

MDS client API state change callbacks
Co-allocation

• Simultaneous allocation of a resource set
  - Handled via optimistic co-allocation based on free nodes or queue prediction
  - In the future, advance reservations will also be supported (already in prototype)

• Globus APIs/SDKs support the co-allocation of specific multi-requests
  - Uses a Globus component called the Dynamically Updated Request Online Co-allocator (DUROC)
Multirequest: “+”

- A multirequest allows us to specify multiple resource needs, for example
  
  + (& (count=5)(memory>=64) (executable=p1))
  (&(network=atm) (executable=p2))

  - Execute 5 instances of p1 on a machine with at least 64M of memory
  - Execute p2 on a machine with an ATM connection

- Multirequests are central to co-allocation
A Co-allocation Multirequest

\[ + (\ (resourceManagerContact=\
    \ "flash.isi.edu:754:/C=US/.../CN=flash.isi.edu-fork")\
    (count=1)\
    (label="subjob A")\
    (executable=my_app1)\
  )\
\]

\[ (\ (resourceManagerContact=\
    \ "sp139.sdsc.edu:8711:/C=US/.../CN=sp097.sdsc.edu-lsf")\
    (count=2)\
    (label="subjob B")\
    (executable=my_app2)\
  )\]

Different resource managers

Different executables

Different counts
Job Submission Interfaces

- Globus Toolkit includes several command line programs for job submission
  - `globus-job-run`: Interactive jobs
  - `globus-job-submit`: Batch/offline jobs
  - `globusrun`: Flexible scripting infrastructure

- Others are building better interfaces
  - General purpose
    - Condor-G, PBS, GRD, Hotpage, etc
  - Application specific
    - ECCE’, Cactus, Web portals
globus-job-run

- For running of interactive jobs
- Additional functionality beyond rsh
  - Ex: Run 2 process job with executable staging
    `globus-job-run -: host -np 2 -s myprog arg1 arg2`
  - Ex: Run 5 processes across 2 hosts
    `globus-job-run \
    -: host1 -np 2 -s myprog.linux arg1 \
    -: host2 -np 3 -s myprog.aix arg2`
  - For list of arguments run:
    `globus-job-run -help`
globus-job-submit

- For running of batch/offline jobs
  - `globus-job-submit` Submit job
    > Same interface as `globus-job-run`
    > Returns immediately
  - `globus-job-status` Check job status
  - `globus-job-cancel` Cancel job
  - `globus-job-get-output` Get job stdout/err
  - `globus-job-clean` Cleanup after job
globusrun

- **Flexible job submission for scripting**
  - Uses an RSL string to specify job request
  - Contains an embedded globus-gass-server
    > Defines GASS URL prefix in RSL substitution variable:
      (stdout=$(GLOBUSRUN_GASS_URL)/stdout)
  - Supports both interactive and offline jobs

- **Complex to use**
  - Must write RSL by hand
  - Must understand its esoteric features
  - Generally you should use globus-job-* commands instead
Resource Management APIs

- The `globus_gram_client` API provides access to all of the core job submission and management capabilities, including callback capabilities for monitoring job status.
- The `globus_rsl` API provides convenience functions for manipulating and constructing RSL strings.
- The `globus_gram_myjob` allows multi-process jobs to self-organize and to communicate with each other.
- The `globus_duroc_control` and `globus_duroc_runtime` APIs provide access to multirequest (co-allocation) capabilities.
Advance Reservation and Other Generalizations

- General-purpose Architecture for Reservation and Allocation (GARA)
  - 2nd generation resource management services
- Broadens GRAM on two axes
  - Generalize to support various resource types
    > CPU, storage, network, devices, etc.
  - Advance reservation of resources, in addition to allocation
- Currently a research prototype
GARA: The Big Picture
Resource Management Futures: GRAM-2 (planned for 2002)

- Advance reservations
  - As prototyped in GARA in previous 2 years
- Multiple resource types
  - Manage anything: storage, networks, etc., etc.
- Recoverable requests, timeout, etc.
- Better lifetime management
- Policy evaluation points for restricted proxies
- Use of Web Services (WSDL, SOAP)

Karl Czajkowski, Steve Tuecke, others
The Globus Toolkit™: Information Services

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Grid Information Services

- System information is critical to operation of the grid and construction of applications
  - What resources are available?
    > Resource discovery
  - What is the “state” of the grid?
    > Resource selection
  - How to optimize resource use
    > Application configuration and adaptation?

- We need a general information infrastructure to answer these questions
Examples of Useful Information

- **Characteristics of a compute resource**
  - IP address, software available, system administrator, networks connected to, OS version, load

- **Characteristics of a network**
  - Bandwidth and latency, protocols, logical topology

- **Characteristics of the Globus infrastructure**
  - Hosts, resource managers
Grid Information: Facts of Life

- Information is always old
  - Time of flight, changing system state
  - Need to provide quality metrics
- Distributed state hard to obtain
  - Complexity of global snapshot
- Component will fail
- Scalability and overhead
- Many different usage scenarios
  - Heterogeneous policy, different information organizations, etc.
Grid Information Service

- Provide access to static and dynamic information regarding system components
- A basis for configuration and adaptation in heterogeneous, dynamic environments
- Requirements and characteristics
  - Uniform, flexible access to information
  - Scalable, efficient access to dynamic data
  - Access to multiple information sources
  - Decentralized maintenance
The GIS Problem: Many Information Sources, Many Views
What is a Virtual Organization?

- Facilitates the workflow of a group of users across multiple domains who share (some of) their resources to solve particular classes of problems
- Collates and presents information about these resources in a uniform view
Two Classes Of Information Servers

- **Resource Description Services**
  - Supplies information about a specific resource (e.g. Globus 1.1.3 GRIS).

- **Aggregate Directory Services**
  - Supplies collection of information which was gathered from multiple GRIS servers (e.g. Globus 1.1.3 GIIS).
  - Customized naming and indexing
Information Protocols

- **Grid Resource Registration Protocol**
  - Support information/resource discovery
  - Designed to support machine/network failure

- **Grid Resource Inquiry Protocol**
  - Query resource description server for information
  - Query aggregate server for information
  - LDAP V3.0 in Globus 1.1.3
GIS Architecture

Customized Aggregate Directories

Users

Enquiry Protocol

A

Registration Protocol

A

Standard Resource Description Services

R

R

R

R
Metacomputing Directory Service

- Use LDAP as Inquiry
- Access information in a distributed directory
  - Directory represented by collection of LDAP servers
  - Each server optimized for particular function
- Directory can be updated by:
  - Information providers and tools
  - Applications (i.e., users)
  - Backend tools which generate info on demand
- Information dynamically available to tools and applications
Two Classes Of MDS Servers

- **Grid Resource Information Service (GRIS)**
  - Supplies information about a specific resource
  - Configurable to support multiple information providers
  - LDAP as inquiry protocol

- **Grid Index Information Service (GIIS)**
  - Supplies collection of information which was gathered from multiple GRIS servers
  - Supports efficient queries against information which is spread across multiple GRIS servers
  - LDAP as inquiry protocol
LDAP Details

- **Lightweight Directory Access Protocol**
  - IETF Standard
  - Stripped down version of X.500 DAP protocol
  - Supports distributed storage/access (referrals)
  - Supports authentication and access control

- **Defines:**
  - Network protocol for accessing directory contents
  - Information model defining form of information
  - Namespace defining how information is referenced and organized
MDS Components

- LDAP 3.0 Protocol Engine
  - Based on OpenLDAP with custom backend
  - Integrated caching
- Information providers
  - Delivers resource information to backend
- APIs for accessing & updating MDS contents
  - C, Java, PERL (LDAP API, JNDI)
- Various tools for manipulating MDS contents
  - Command line tools, Shell scripts & GUIs
Grid Resource Information Service

- **Server which runs on each resource**
  - Given the resource DNS name, you can find the GRIS server (well known port = 2135)

- **Provides resource specific information**
  - Much of this information may be dynamic
    - Load, process information, storage information, etc.
    - GRIS gathers this information on demand

- **“White pages” lookup of resource information**
  - Ex: How much memory does machine have?

- **“Yellow pages” lookup of resource options**
  - Ex: Which queues on machine allows large jobs?
Grid Index Information Service

- **GIIS describes a class of servers**
  - Gathers information from multiple GRIS servers
  - Each GIIS is optimized for particular queries
    > Ex1: Which Alliance machines are >16 process SGIs?
    > Ex2: Which Alliance storage servers have >100Mbps bandwidth to host X?
  - Akin to web search engines

- **Organization GIIS**
  - The Globus Toolkit ships with one GIIS
  - Caches GRIS info with long update frequency
    > Useful for queries across an organization that rely on relatively static information (Ex1 above)

- **Can be merged into GRIS**
Finding a GRIS and Server Registration

- A GRIS or GIIS server can be configured to (de-) register itself during startup/shutdown
  - Targets specified in configuration file
- Softstate registration protocol
  - Good behavior in case of failure
- Allows for federations of information servers
  - E.g. Argonne GRIS can register with both Alliance and DOE GIIS servers
Logical MDS Deployment

Grads

GIIS

ISI

GRISes

Gusto
MDS Commands

- LDAP defines a set of standard commands
  ldapsearch, etc.
- We also define MDS-specific commands
  - grid-info-search, grid-info-host-search
- APIs are defined for C, Java, etc.
  - C: OpenLDAP client API
    > ldap_search_s(), ...
  - Java: JNDI
Information Services API

- RFC 1823 defines an IETF draft standard client API for accessing LDAP databases
  - Connect to server
  - Pose query which returns data structures contains sets of object classes and attributes
  - Functions to walk these data structures

- Globus does not provide an LDAP API. We recommend the use of OpenLDAP, an open source implementation of RFC 1823.
Searching an LDAP Directory

grid-info-search [options] filter [attributes]

- Default grid-info-search options
  -h mds.globus.org  \(\text{MDS server}\)
  -p 389  \(\text{MDS port}\)
  -b "o=Grid"  \(\text{search start point}\)
  -T 30  \(\text{LDAP query timeout}\)
  -s sub  \(\text{scope} = \text{subtree}\)

  alternatives:
  - base  : lookup this entry
  - one  : lookup immediate children
Searching a GRIS Server

grid-info-host-search [options] filter [attributes]

- Exactly like grid-info-search, except defaults:
  -h localhost  
  -p 2135

- Example:
  grid-info-host-search -h pitcairn "dn=*" dn
Filtering

- Filters allow selection of object based on relational operators (=, ~=, <=, >=)
  - grid-info-search “cputype=*

- Compound filters can be construct with Boolean operations: (&, |, !)
  - grid-info-search “(&(cputype=*)(cpuload1<=1.0))”
  - grid-info-search “(&(hn~=sdsc.edu)(latency<=10))”

- Hints:
  - white space is significant
  - use -L for LDIF format

required
Example: Filtering

% grid-info-host-search -L "(objectclass=GlobusSoftware)"

dn: sw=Globus, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid
objectclass: GlobusSoftware
releasemajor: 1
releaseminor: 1
releasepatch: 3
releasebeta: 11
lastupdate: Sun Apr 30 19:28:19 GMT 2000
objectname: sw=Globus, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid
Example: Attribute Selection

% grid-info-host- search -L "(objectclass=*)" dn hn

- Returns the distinguished name (dn) and hostname (hn) of all objects

```
dn: sw=Globus, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid
```

```
dn: hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid
hn: pitcairn.mcs.anl.gov
```

```
dn: service=jobmanager, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid
hn: pitcairn.mcs.anl.gov
```

```
dn: queue=default, service=jobmanager, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid
```

- Objects without hn fields are still listed
- DNs are always listed
Example: Discovering CPU Load

- Retrieve CPU load fields of compute resources

```bash
% grid-info-search -L "(objectclass=GlobusComputeResource)" \ 
  dn cpuload1 cpuload5 cpuload15

dn: hn=lemon.mcs.anl.gov, ou=MCS, o=Argonne National Laboratory, 
  o=Globus, c=US
  cpuload1: 0.48
  cpuload5: 0.20
  cpuload15: 0.03

dn: hn=tuva.mcs.anl.gov, ou=MCS, o=Argonne National Laboratory, 
  o=Globus, c=US
  cpuload1: 3.11
  cpuload5: 2.64
  cpuload15: 2.57
```
The Globus Toolkit™: Data Management Services

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Data Grid Problem

- “Enable a geographically distributed community [of thousands] to pool their resources in order to perform sophisticated, computationally intensive analyses on Petabytes of data”

- Note that this problem:
  - Is common to many areas of science
  - Overlaps strongly with other Grid problems
Major Data Grid Projects

- **Earth System Grid (DOE Office of Science)**
  - DG technologies, climate applications
- **European Data Grid (EU)**
  - DG technologies & deployment in EU
- **GriPhyN (NSF ITR)**
  - Investigation of “Virtual Data” concept
- **Particle Physics Data Grid (DOE Science)**
  - DG applications for HENP experiments
There is a "bunch crossing" every 25 nsecs. There are 100 "triggers" per second. Each triggered event is ~1 MByte in size.

1 TIPS is approximately 25,000 SpecInt95 equivalents.

Physicists work on analysis "channels". Each institute will have ~10 physicists working on one or more channels; data for these channels should be cached by the institute server.
Data Intensive Issues Include ...

- Harness [potentially large numbers of] data, storage, network resources located in distinct administrative domains
- Respect local and global policies governing what can be used for what
- Schedule resources efficiently, again subject to local and global constraints
- Achieve high performance, with respect to both speed and reliability
- Catalog software and virtual data
Data Intensive Computing and Grids

- The term “Data Grid” is often used
  - Unfortunate as it implies a distinct infrastructure, which it isn’t; but easy to say
- Data-intensive computing shares numerous requirements with collaboration, instrumentation, computation, ...
  - Security, resource mgt, info services, etc.
- Important to exploit commonalities as very unlikely that multiple infrastructures can be maintained
- Fortunately this seems easy to do!
Examples of Desired Data Grid Functionality

- High-speed, reliable access to remote data
- Automated discovery of “best” copy of data
- Manage replication to improve performance
- Co-schedule compute, storage, network
- “Transparency” wrt delivered performance
- Enforce access control on data
- Allow representation of “global” resource allocation policies
A Model Architecture for Data Grids

Metadata Catalog

Application

Replica Catalog

Replica Selection

MDS

NWS

Disk Array

Disk Cache

Tape Library

Replica Location 1

Replica Location 2

Replica Location 3

Logical Collection and Logical File Name

Attribute Specification

Selected Replica

Multiple Locations

Performance Information & Predictions

GridFTP Control Channel
Globus Toolkit Components

Two major Data Grid components:

1. Data Transport and Access
   - Common protocol
     - Secure, efficient, flexible, extensible data movement
   - Family of tools supporting this protocol

2. Replica Management Architecture
   - Simple scheme for managing:
     - multiple copies of files
     - collections of files
Motivation for a Common Data Access Protocol

• Existing distributed data storage systems
  - DPSS, HPSS: focus on high-performance access, utilize parallel data transfer, striping
  - DFS: focus on high-volume usage, dataset replication, local caching
  - SRB: connects heterogeneous data collections, uniform client interface, metadata queries

• Problems
  - Incompatible (and proprietary) protocols
    > Each require custom client
    > Partitions available data sets and storage devices
  - Each protocol has subset of desired functionality
A Common, Secure, Efficient Data Access Protocol

- Common, *extensible* transfer protocol
  - Common protocol means all can interoperate
- Decouple low-level data transfer mechanisms from the storage service
- Advantages:
  - New, specialized storage systems are automatically compatible with existing systems
  - Existing systems have richer data transfer functionality
- Interface to many storage systems
  - HPSS, DPSS, file systems
  - Plan for SRB integration
Access/Transport Protocol Requirements

- Suite of communication libraries and related tools that support
  - GSI, Kerberos security
  - Third-party transfers
  - Parameter set/negotiate
  - Partial file access
  - Reliability/restart
  - Large file support
  - Data channel reuse
  - Integrated instrumentation
  - Loggin/audit trail
  - Parallel transfers
  - Striping (cf DPSS)
  - Policy-based access control
  - Server-side computation
  - Proxies (firewall, load bal)

- All based on a standard, widely deployed protocol
And The Protocol Is ... GridFTP

- **Why FTP?**
  - Ubiquity enables interoperation with many commodity tools
  - Already supports many desired features, easily extended to support others
  - Well understood and supported

- **We use the term GridFTP to refer to**
  - Transfer protocol which meets requirements
  - Family of tools which implement the protocol

- **Note GridFTP > FTP**

- **Note that despite name, GridFTP is not restricted to file transfer!**
GridFTP: Basic Approach

- FTP protocol is defined by several IETF RFCs
- Start with most commonly used subset
  - Standard FTP: get/put etc., 3rd-party transfer
- Implement standard but often unused features
  - GSS binding, extended directory listing, simple restart
- Extend in various ways, while preserving interoperability with existing servers
  - Striped/parallel data channels, partial file, automatic & manual TCP buffer setting, progress monitoring, extended restart
GridFTP Protocol Specifications

- **Existing standards**
  - RFC 949: File Transfer Protocol
  - RFC 2228: FTP Security Extensions
  - RFC 2389: Feature Negotiation for the File Transfer Protocol
  - Draft: FTP Extensions

- **New drafts**
  - GridFTP: Protocol Extensions to FTP for the Grid
  > Grid Forum Data Working Group
GridFTP vs. WebDAV

- **WebDAV** extends http for remote data access
  - Combines control and data over single channel
- **FTP** splits control and data
  - Supports multiple, user selectable data channel protocols
- **Advantage to split channels**
  - Third party transfers handled cleanly
  - Can (cleanly) define new data channel protocols
    > E.g. parallel/striped transfer, automatic TCP buffer/window negotiation, non-TCP based protocols, etc.
  - Amenable to high-performance proxies
    > E.g. For firewalls, load balancing, etc.
The GridFTP Family of Tools

- **Patches to existing FTP code**
  - GSI-enabled versions of existing FTP client and server, for high-quality production code

- **Custom-developed libraries**
  - Implement full GridFTP protocol, targeting custom use, high-performance

- **Custom-developed tools**
  - Servers and clients with specialized functionality and performance
Family of Tools: Patches to Existing Code

- Patches to standard FTP clients and servers
  - gsi-ncftp: Widely used client
  - gsi-wuftp: Widely used server
  - GSI modified HPSS pftpd
  - GSI modified Unitree ftpd
- Provides high-quality, production ready, FTP clients and servers
- Integration with common mass storage systems
- Some do not support the full GridFTP protocol
Family of Tools: Custom Developed Libraries

- Custom developed libraries
  - globus_ftp_control: Low level FTP driver
    > Client & server protocol and connection management
  - globus_ftp_client: Simple, reliable FTP client
    > Plugins for restart, logging, etc.
  - globus_gass_copy: Simple URL-to-URL copy library, supporting (gsi-)ftp, http(s), file URLs

- Implement full GridFTP protocol
- Various levels of libraries, allowing implementation of custom clients and servers
- Tuned for high performance on WAN
Family of Tools: Custom Developed Programs

- Simple production client
  - globus-url-copy: Simple URL-to-URL copy

- Experimental FTP servers
  - Striped FTP server (ala.DPSS): MPI-IO backend
  - Multi-threaded FTP server with parallel channels
  - Firewall FTP proxy: Securely and efficiently allow transfers through firewalls
  - Load balancing FTP proxy: Large data centers

- Experimental FTP clients
  - POSIX file interface
globus\_ftp\_client Plug-ins

- globus\_ftp\_client is simple API/SDK:
  - get, put, 3\textsuperscript{rd} party transfer, cd, mkdir, etc.
  - All data is to/from memory buffers
    > Optimized to avoid any data copies
  - Plug-in interface
    > Interface to one or more plug-ins:
    - Callouts for all interesting protocol events
    - Callins to restart a transfer
    > Can support:
    - Monitor performance
    - Monitor for failure
    - Automatic retry: Customized for various approaches
GridFTP at SC’2000: Long-Running Dallas-Chicago Transfer

SciNet Power Failure

Other demos starting up (Congestion)

Parallelism Increases (Demos)

DNS Problems

Transition between files (not zero due to averaging)

Backbone problems on the SC Floor
(Prototype)
Striped GridFTP Server

GridFTP Control Channel

GridFTP server master

GridFTP Control Channel

mpirun

Control socket

To Client or Another Striped GridFTP Server

GridFTP Data Channels

GridFTP Server Parallel Backend

Control Plug-in

Control Plug-in

Control Plug-in

Control Plug-in

MPI (Comm_World)

MPI (Sub-Comm)

MPI-IO

Parallel File System (e.g. PVFS, PFS, etc.)
Striped GridFTP Plug-in Interface

- **Given a RETR or STOR request:**
  - Control calls plug-in to determine which nodes should participate in the request
  - Control creates an MPI sub-comm for nodes
  - Control calls plug-in to perform the transfer
    > Includes request info, communicator, `globus_ftp_control_handle_t`
  - Plug-in does I/O to backend
    > MPI-IO, PVFS, Unix I/O, Raw I/O, etc.
  - Plug-in uses `globus_ftp_control_data_*()` functions to send/receive data on GridFTP data channels
Stripped GridFTP Performance

- At SC’00, used first prototype:
  - Transfer between Dallas and LBNL
  - 8 node Linux clusters on each end
  - OC-48, 2.5Gb/s link (NTON)
  - Peaks over 1.5Gb/s
    > Limited by disk bandwidth on end-points
  - 5 second peaks over 1Gb/s
  - Sustained 530Mb/s for 1 hr (238GB transfer)
    > Had not yet implemented large files or data channel reuse.
    > 2GB file took <20 seconds. New data channel sockets connected for each transfer.
    > Explains difference between sustained and peak.
Replica Management

- Maintain a mapping between logical names for files and collections and one or more physical locations
- Important for many applications
  - Example: CERN HLT data
    - Multiple petabytes of data per year
    - Copy of everything at CERN (Tier 0)
    - Subsets at national centers (Tier 1)
    - Smaller regional centers (Tier 2)
    - Individual researchers will have copies
Our Approach to Replica Management

- Identify replica cataloging and reliable replication as two fundamental services
  - Layer on other Grid services: GSI, transport, information service
  - Use LDAP as catalog format and protocol, for consistency
  - Use as a building block for other tools

- Advantage
  - These services can be used in a wide variety of situations
Replica Manager Components

- **Replica catalog definition**
  - LDAP object classes for representing logical-to-physical mappings in an LDAP catalog

- **Low-level replica_catalog API**
  - globus_replica_catalog library
  - Manipulates replica catalog: add, delete, etc.

- **High-level reliable replication API**
  - globus_replica_manager library
  - Combines calls to file transfer operations and calls to low-level API functions: create, destroy, etc.
Replica Catalog Structure: A Climate Modeling Example

Replica Catalog

Logical Collection
C02 measurements 1998
Filename: Jan 1998
Filename: Feb 1998
...
Filename: Mar 1998
Filename: Jun 1998
Filename: Oct 1998
Protocol: gsiftp
UrlConstructor: gsiftp://jupiter.isi.edu/nfs/v6/climate

Logical Collection
C02 measurements 1999

Location
jupiter.isi.edu
Filename: Jan 1998
Filename: Feb 1998
...
Filename: Mar 1998
Filename: Jun 1998
Filename: Oct 1998
Protocol: gsiftp
UrlConstructor: gsiftp://jupiter.isi.edu/nfs/v6/climate

Location
sprite.llnl.gov
Filename: Jan 1998
...
Filename: Dec 1998
Protocol: ftp
UrlConstructor: ftp://sprite.llnl.gov/pub/pcmdi

Logical File Parent

Logical File
Jan 1998
Size: 1468762

Logical File
Feb 1998
Replica Catalog Services as Building Blocks: Examples

- Combine with information service to build **replica selection** services
  - E.g. “find best replica” using performance info from NWS and MDS
  - Use of LDAP as common protocol for info and replica services makes this easier

- Combine with application managers to build **data distribution** services
  - E.g., build new replicas in response to frequent accesses
Relationship to Metadata Catalogs

- Metadata services describe data contents
  - Have defined a simple set of object classes
- Must support a variety of metadata catalogs
  - MCAT being one important example
  - Others include LDAP catalogs, HDF
- Community metadata catalogs
  - Agree on set of attributes
  - Produce names needed by replica catalog:
    - Logical collection name
    - Logical file name
Replica Catalog Directions

- Many data grid applications do not require tight consistency semantics
  - At any given time, you may not be able to discover all copies
  - When a new copy is made, it may not be immediately recognized as available
- Allows for much more scalable design
  - Distributed catalogs: local catalogs which maintain their own LFN -> PFN mapping
  - Soft-state updates as basis for building various configurations of global catalogs
Data Transfer APIs

- The `globus_ftp_control` API provides access to low-level GridFTP control and data channel operations.
- The `globus_ftp_client` API provides typical GridFTP client operations.
- The `globus_gass_copy` API provides the ability to start and manage multiple data transfers using GridFTP, HTTP, local file, and memory operations.
  - The `globus-url-copy` program is a thin wrapper around this API.
Replica Management APIs

- The `globus_replica_catalog` API provides basic Replica Catalog operations.
- The `globus_replica_management` API (under development) combines GridFTP and the Replica Catalog to manage replicated datasets.
A Word on GASS

- The Globus Toolkit provides services for file and executable staging and I/O redirection that work well with GRAM. This is known as Globus Access to Secondary Storage (GASS).
- GASS uses GSI-enabled HTTP as the protocol for data transfer, and a caching algorithm for copying data when necessary.
- The `globus_gass`, `globus_gass_transfer`, and `globus_gass_cache` APIs provide programmer access to these capabilities, which are already integrated with the GRAM job submission tools.
Future Directions

- Continued enhancement & standardization of protocol
  - Globus Toolkit libraries provide reference implementation
- Continue building on libraries
  - Striped server w/ server side processing
  - Reliable replica/copy management service
  - Proxies for firewalls & load balancing
- Work with more application communities
Grid Physics Network (GriPhyN)

Enabling R&D for advanced data grid systems, focusing in particular on Virtual Data concept

- Virtual Data Tools
- Request Planning and Scheduling Tools
- Request Execution Management Tools

Interactive User Tools

- Production Team
- Other Users
- Individual Investigator

Resource Management Services

Security and Policy Services

Other Grid Services

Raw data source

Distributed resources (code, storage, computers, and network)

Transforms

ATLAS
CMS
LIGO
SDSS
The Virtual Data Concept

“[a virtual data grid enables] the definition and delivery of a potentially unlimited virtual space of data products derived from other data. In this virtual space, requests can be satisfied via direct retrieval of materialized products and/or computation, with local and global resource management, policy, and security constraints determining the strategy used.”
Virtual Data in Action

- Data request may
  - Access local data
  - Compute locally
  - Compute remotely
  - Access remote data
- Scheduling subject to local & global policies
- Local autonomy
Related Work:
Condor-G

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
What Is Condor-G?

- Enhanced version of Condor that uses Globus Toolkit™ to manage Grid jobs
- Two Parts
  - Globus Universe
  - GlideIn
- Excellent example of applying the general purpose Globus Toolkit to solve a particular problem (I.e. high-throughput computing) on the Grid
Condor

- High-throughput scheduler
- Non-dedicated resources
- Job checkpoint and migration
- Remote system calls
Globus Toolkit

- Grid infrastructure software
- Tools that simplify working across multiple institutions:
  - Authentication (GSI)
  - Scheduling (GRAM, DUROC)
  - File transfer (GASS, GridFTP)
  - Resource description (GRIS/GIIS)
Why Use Condor-G

• Condor
  – Designed to run jobs within a single administrative domain

• Globus Toolkit
  – Designed to run jobs across many administrative domains

• Condor-G
  – Combine the strengths of both
Globus Universe

- Advantages of using Condor-G to manage your Grid jobs
  - Full-featured queuing service
  - Credential Management
  - Fault-tolerance
Full-Featured Queue

- Persistent queue
- Many queue-manipulation tools
- Set up job dependencies (DAGman)
- E-mail notification of events
- Log files
Credential Management

- Authentication in Globus Toolkit is done with limited-lifetime X509 proxies
- Proxy may expire before jobs finish executing
- Condor-G can put jobs on hold and e-mail user to refresh proxy
- Condor-G can forward new proxy to execution sites
Fault Tolerance

- **Local Crash**
  - Queue state stored on disk
  - Reconnect to execute machines

- **Network Failure**
  - Wait until connectivity returns
  - Reconnect to execute machines
Fault Tolerance

- Remote Crash – job still in queue
  - Job state stored on disk
  - Start new jobmanager to monitor job
- Remote Crash – job lost
  - Resubmit job
GRAM-1.5 Changes

- Changes to improve recoverability from faults, to better support Condor-G
  - U Wisconsin contributed these changes
- Added Features
  - Jobmanager checkpoint & restart
  - Two-Phase commit during job submission
- GRAM-1.5 protocol (Globus Toolkit v2.0) is backward compatible with GRAM-1 (Globus Toolkit v1.x)
How It Works

Condor-G

- Schedd

Grid Resource

- LSF
How It Works

Condor-G

Grid Resource

Schedd

LSF

600 Grid jobs
How It Works

Condor-G

- Schedd
- GridManager

Grid Resource

- LSF

600 Grid jobs
How It Works

Condor-G

Schedd

GridManager

Grid Resource

JobManager

LSF

600 Grid jobs
How It Works

Condor-G

Schedd

GridManager

Grid Resource

JobManager

LSF

User Job

600 Grid jobs

October 12, 2001 Intro to Grid Computing and Globus Toolkit™ 201
Globus Universe

- Disadvantages
  - No matchmaking or dynamic scheduling of jobs
  - No job checkpoint or migration
  - No remote system calls
Solution: GlideIn

- Use the Globus Universe to run the Condor daemons on Grid resources
- When the resources run these GlideIn jobs, they will join your personal Condor pool
- Submit your jobs as Condor jobs and they will be matched and run on the Grid resources
How It Works

Condor-G

Schedd

Collector

Grid Resource

LSF

600 Condor jobs
How It Works

Condor-G

Schedd

Collector

glide-ins

Grid Resource

LSF

600 Condor jobs
How It Works

**Condor-G**

- Schedd
- GridManager
- Collector

**Grid Resource**

- LSF

---

600 Condor jobs

GLIDE-INS
How It Works

Condor-G

- Schedd
- GridManager
- Collector

Grid Resource

- JobManager
- LSF

600 Condor jobs

glide-ins
How It Works

Condor-G

- Schedd
- GridManager
- Collector

Grid Resource

- JobManager
- LSF
- Startd

600 Condor jobs

glide-ins

the globus project
www.globus.org
How It Works

Condo-G

Schedd

GridManager

Collector

glide-ins

Grid Resource

JobManager

LSF

Startd

600 Condor jobs
How It Works

Condor-G

- Schedd
- GridManager
- Collector

Grid Resource

- JobManager
- LSF
- Startd
- User Job

600 Condor jobs

glide-ins
GlideIn Concerns

- **What if a Grid resource kills my GlideIn?**
  - That resource will disappear from your pool and you jobs will be rescheduled on other machines

- **What if all my jobs are done before a GlideIn runs?**
  - If the glided-in Condor daemons are not matched with a job in 10 minutes, they terminate
GlideIn Concerns

- Can others in the Condor pool run jobs on my GlideIn resources?
  - No
- Where do I get binaries for other platforms?
  - Repository with binaries for all platforms at UW
  - You can set up your own local repository
Who’s Uses Condor-G?

- MetaNEOS (NUG-30)
- NCSA (GridGaussian)
- INFN (DataGrid)
- CMS
Current Status

- Production version out for several months
  - Runs jobs using Globus GRAM or DUROC
  - Stages executable and standard I/O using Globus GASS
  - Detects and uses refreshed proxies automatically
- GRAM changes will be part of Globus Toolkit v2.0
Future Work

- **GridManager**
  - Stage user jobs’ data files
- **Automatic GlideIn**
  - Condor creates GlideIn jobs when more resources are needed
- **Matchmaking in Globus Universe**
  - Use Globus GRIS to create ClassAds for Grid resources and match them to job ClassAds
- **Support for MPICH-G**
Futures & Conclusions

The Globus Project™
Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org
Problem Evolution

- Past-present: $O(10^2)$ high-end systems; Mb/s networks; centralized (or entirely local) control
  - I-WAY (1995): 17 sites, week-long; 155 Mb/s
  - GUSTO (1998): 80 sites, long-term experiment
  - NASA IPG, NSF NTG: $O(10)$ sites, production

- Present: $O(10^4-10^6)$ data systems, computers; Gb/s networks; scaling, decentralized control
  - Scalable resource discovery; restricted delegation; community policy; Data Grid: 100s of sites, $O(10^4)$ computers; complex policies

- Future: $O(10^6-10^9)$ data, sensors, computers; Tb/s networks; highly flexible policy, control
The Future: All Software is Network-Centric

- We don’t build or buy “computers” anymore, we borrow or lease required resources
  - When I walk into a room, need to solve a problem, need to communicate
- A “computer” is a dynamically, often collaboratively constructed collection of processors, data sources, sensors, networks
  - Similar observations apply for software
And Thus ...

- Reduced barriers to access mean that we do much more computing, and more interesting computing, than today => Many more components (& services); massive parallelism

- All resources are owned by others => Sharing (for fun or profit) is fundamental; trust, policy, negotiation, payment

- All computing is performed on unfamiliar systems => Dynamic behaviors, discovery, adaptivity, failure
Summary

- The Grid problem: Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations
- Grid architecture: Emphasize protocol and service definition to enable interoperability and resource sharing
- Globus Toolkit™ a source of protocol and API definitions, reference implementations