Introduction to Grid Computing

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
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
Some Definitions
Some Important Definitions

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

FTP Server:
- FTP Protocol
- Telnet Protocol
- TCP Protocol
- IP Protocol

Web Server:
- HTTP Protocol
- TLS Protocol
- TCP Protocol
- IP Protocol
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
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 APIs and protocols](image-url)
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
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; ...
The programming problem
- Facilitate development of sophisticated apps
- Facilitate code sharing
- Requires programming environments
  > APIs, SDKs, tools

The systems problem
- Facilitate coordinated use of diverse resources
- Facilitate infrastructure sharing
  > e.g., certificate authorities, information services
- Requires systems
  > protocols, services
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 Emphasizes ...

- 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
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

- **Applications**
  - Diverse global services
  - Core services
  - Local OS
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/gsi](http://www.gridforum.org/security/gsi)
GT2 Resource Layer Protocols & Services

- Grid Resource Allocation Management (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
- Others emerging: Catalog access, code repository access, accounting, etc.
- All built on connectivity layer: GSI & IP
**GT2 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](http://www.cs.wisc.edu/condor)
The Programming Problem
Common Toolkit Underneath

- Each programming environment 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™...
Introduction to the Globus Toolkit™
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
Key Protocols

- The Globus Toolkit™ centers around four key protocols
  - Connectivity layer:
    > Security: Grid Security Infrastructure (GSI)
  - Resource layer:
    > Resource Management
    > Information Services
    > Data Transfer

- Also key collective layer protocols
  - Info Services, Replica Management, etc.
Grid Security Infrastructure (GSI)

- Globus Toolkit implements GSI protocols and APIs, to address Grid security needs
- GSI protocols extends standard public key protocols
  - Standards: X.509 & SSL/TLS
  - Extensions: X.509 Proxy Certificates & Delegation
- GSI extends standard GSS-API
Resource Management

- The Grid Resource Allocation Management (GRAM) protocol and client API allows programs to be started and managed 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, ...
Information Services

- GT2 – MDS (GRIS/GIIS)
  - Based on LDAP protocol
- GT3 – Service Data Elements
  - From the OGSI spec
Data Access & Transfer

- **GridFTP**: extended version of popular FTP protocol for Grid data access and transfer
  - Secure, efficient, reliable, flexible, extensible, parallel, concurrent, e.g.:
    - Third-party data transfers, partial file transfers
    - Parallelism, striping (e.g., on PVFS)
    - Reliable, recoverable data transfers

- **Reference implementations**
  - Existing clients and servers: wuftp, globus-url-copy
  - Flexible, extensible libraries in Globus Toolkit
Summary

- The Grid problem: Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations
- Grid architecture emphasizes systems problem
  - Protocols & services, to facilitate interoperability and shared infrastructure services
- Globus Toolkit™: APIs, SDKs, and tools which implement Grid protocols & services
  - Provides basic software infrastructure for suite of tools addressing the programming problem