Grid Computing 7700
Fall 2005
Lecture 10 and 12: Globus V2

Gabrielle Allen
allen@bit.csc.lsu.edu

http://www.cct.lsu.edu/~gallen/
Required Reading

- Globus 4 Primer
Coursework

- Essay: 4 pages
- Describe the motivation and architecture of the Grid Application Toolkit, and debate its advantages and disadvantages, end with a thoughtful conclusion.
Recap ...

- Four core components to Globus Toolkit
  - Resources (GRAM)
    - http://www.globus.org/toolkit/docs/4.0/execution/key/
  - Data (GridFTP, RLS, RFT)
    - http://www.globus.org/toolkit/docs/4.0/data/key/
  - Information (MDS)
    - http://www.globus.org/toolkit/docs/4.0/info/key-index.html
  - Security (GSI)
    - http://www.globus.org/toolkit/docs/4.0/security/key-index.html
GRAM

- **Grid Resource Allocation and Management**
  - Creation and management of remote computations
  - GSI for authentication, authorization, delegation
  - GRAM implementations map requests expressed in a Resource Specification Language (RSL) into commands understood by local schedulers and computers
  - Multiple GRAM implementations exist (with C, Java, Python interfaces)
  - **GT2 implementation**
    - Based on HTTP protocol
    - "gatekeeper" initiates remote computations
    - "jobmanager" manages remote computation
    - GRAM reporter monitors and publishes information
Resource Management Review

- Resource Specification Language (RSL) is used to communicate requirements.
- The Grid Resource Allocation and Management (GRAM) API allows programs to be started on remote resources, despite local heterogeneity.
- A layered architecture allows application-specific resource brokers and co-allocators (e.g. DUROC) to be defined in terms of GRAM services.
GRAM Components

- **Client**: GRAM client API calls to request resource allocation and process creation.
- **Gatekeeper**: GRAM client API calls to request resource allocation and process creation.
- **Globus Security Infrastructure**: GRAM client API calls to get resource info.
- **MDS: Grid Index Info Server**: MDS client API calls to locate resources.
- **MDS: Grid Resource Info Server**: MDS client API calls to get resource info, GRAM client API state change callbacks.
- **Local Resource Manager**: Query current status of resource, Allocate & create processes.
- **Job Manager**: Parse, Request, Monitor & control.
- **Site boundary**: Site boundary.
Resource Specification Language (RSL)

- **Common language for specifying job requests**
  - GRAM service translates this common language into scheduler specific language
- **Specified as multiple attribute value pairs**
  - E.g. `&([executable=“/bin/ls‟])(arguments=“-l”)`
- **GRAM has a defined set of attributes**
RSL Attributes For GRAM

- (executable=string)
- (directory=string)
- (arguments=arg1 arg2 arg3...)
- (environment=(E1 v1)(E2 v2))
- (stdin=string)
- (stdout=string)
- (stderr=string)
- (count=integer)
- (project=string)
- (queue=string)
- (maxTime=integer)
- (maxWallTime=integer)
- (maxCpuTime=integer)
- (maxMemory=integer)
- (minMemory=integer)
- (gramMyjob=value)
  - Value is one of “collective”, “independent”
- (jobType=value)
  - mpi: Run the program using “mpirun -np <count>”
  - single: Only run a single instance of the program, let the program start the other count-1 processes.
  - multiple: Start <count> instances of the program using the appropriate scheduler mechanism
  - condor: Start a <count> Condor processes running in “standard universe”
- (dryRun=true)
  - Do not actually run job
GRAM Defined
RSL Substitutions

- GRAM defines a set of RSL substitutions before processing the job request
- **Machine Information**
  - `GLOBUS_HOST_MANUFACTURER`
  - `GLOBUS_HOST_CPUTYPE`
  - `GLOBUS_HOST_OSNAME`
  - `GLOBUS_HOST_OSVERSION`
- **Paths to Globus**
  - `GLOBUS_LOCATION`
- **Miscellaneous**
  - `HOME`
  - `LOGNAME`
  - `GLOBUS_ID`
GRAM Examples

The globus-job-run client is a sample GRAM client that integrates GASS services for executable staging and standard I/O redirection, using command-line arguments rather than RSL.

```
% globus-job-run pitcairn.mcs.anl.gov /bin/ls
% globus-job-run pitcairn.mcs.anl.gov -s myprog
% globus-job-run pitcairn.mcs.anl.gov \ 
  -s myprog -stdin -s in.txt -stdout -s out.txt
```
GRAM Examples

The globusrun client is a more involved tool that allows complicated RSL expressions.

% globusrun -r pitcairn.mcs.anl.gov -f myjob.rsl
% globusrun -r pitcairn.mcs.anl.gov \
  '(&(executable=myprog)'

globus_gram_client

- **globus_gram_client_job_request()**
  - Submit a job to a remote resource
  - **Input:**
    - Resource manager contact string
    - RSL specifying the job to be run
    - Callback contact string, for notification
  - **Output:**
    - Job contact string

- **globus_gram_client_job_status()**
  - Check the status of the job
    - **UNSUBMITTED, PENDING, ACTIVE, FAILED, DONE, SUSPENDED**
  - Can also get job status through callbacks
    - **globus_gram_client_callback_{allow,disallow,check}()**

- **globus_gram_client_job_cancel()**
  - Cancel/kill a pending or active job
DUROC Review

- globusrun will co-allocate specific multi-requests
  - Uses a Globus component called the Dynamically Updated Request Online Co-allocator (DUROC)

```plaintext
+( & (resourceManagerContact= "flash.isi.edu:2119/jobmanagerlsf:/O=Grid/.../CN=host/flash.isi.edu")
  (count=1)
  (label="subjob A")
  (executable= my_app1)
 )
( & (resourceManagerContact="sp139.sdsc.edu:2119:/O=Grid/.../CN=host/sp097.sdsc.edu")
  (count=2)
  (label="subjob B")
  (executable=my_app2)
)```
Grid Information System

Information for ...

- Operation of Grid
  - Monitoring and testing Grid

- Deployment of applications
  - What resources are available to me? (Resource discovery)
  - What is the state of the grid? (Resource selection)
  - How to optimize resource use? (Application configuration and adaptation)
What are the Problems

- How to obtain needed information? (automatic and accurate)
- Information is always old
  - Resources change state
  - Takes time to retrieve information
  - Need to provide quality metrics
- Grid is distributed
  - Global state is very complex
  - Scalability, efficiency and overhead
- Component failure
- Security
- Many different usage scenarios
  - Heterogeneous policy, different information organizations, etc.
Virtual Organizations
Grid Information

- **Compute Resource Specific**
  - Name of resource, IP address, site name, location, firewalls, names of administrators, scheduled downtimes
  - Machine type (SMP, ccNUMA, number of processors, interconnects)
  - Processor types and characteristics (vendor, OS, cache, clockspeed)
  - Software installations (software version, location, license)
  - Jobs (queue names and properties, current running and queued jobs)
Grid Information

- **Network Specific**
  - Network type (peak speed, physical characteristics)
  - Network properties (bandwidth, jitter, latency, QoS)
  - Scheduled downtimes

- **Storage Resource Specific**
  - File system locations
  - File system properties
  - Current space
Capabilities

- Queryable across a network
- Supports virtual organizations
- Complex queries (search for all linux machines with at least 1GB memory and MPI-LAM installed)
- Authentication and authorization.
- Multiple information providers
- Extensible information schemas
- Efficient return of information
- Extensible to large numbers of resources
- Up-to-date information!!
- Queryable in multiple ways (clients, web, APIs)
Example Information Server

- **One of my favorite information**
  - [http://www.imdb.com](http://www.imdb.com)
  - Information about films (movies) updated by viewers

- **History**
  - 1990: shell scripts created by Col Needham used to search FAQs posted to the newsgroup rec.arts.movies
  - 1993: centralized e-mail interface for querying database
  - 1994: interface was extended to allow the submission of information. Then moved to a Web-based interface.
  - 1996: incorporated in the UK to form Internet Movie Database Ltd. with banner ads added to the web site
  - 1998: bought by Amazon.com, the current owner

- Now used by other applications
Globus MDS

- Monitoring and Discovery Service
  - Set of information service components for publishing and discovering information
  - Single standard interface and scheme to information services in a virtual organization

- MDS can aggregate information from multiple sites each with multiple resources

- Information about each resource is provided by an information provider
Globus MDS

- Handles static (OS type) and dynamic (current load) data
- Access to data can be restricted with GSI (Grid Security Infrastructure) credentials and authorization features
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
Higher Level Services

- Can query MDS for information (e.g. web browser interface, command line clients, MDS API)
  - List all registered sites
  - List all resources at a given site
  - Provide OS type and number of nodes for a particular machine
  - List all machines running AIX with over 4 nodes
  - Provide hostname of machine with lowest current load
Local Resource Monitoring

- Publishes information to MDS
  - Cluster monitoring: e.g. Ganglia
  - Queue information: GRAM Reporter
  - Network information: NWS
  - Other local monitoring systems may require writing MDS interfaces
MDS Information

- Depends on what information provider wants to provide
- **Static host information**
  - Operating system version, process architecture, number of processors, vendor, location, total disk space
- **Dynamic host information**
  - Load average, queue entries, uptime, available disk space
- **Core information providers which come with MDS**
  provide a given set of static and dynamic information
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 server
  - LDAP as inquiry protocol
MDS Architecture

- LDAP server provides common interface
- MDS uses LDAP protocol to query information
- GRIS: Grid Resource Information Service
  - Speaks LDAP protocol and provides information about a particular resource
- GIIS: Grid Index Information Service
  - GRIS’s register with a GIIS
  - GIIS can be queried for collective-level information
GIS Architecture

Customized Aggregate Directories

Users

Enquiry Protocol

Registration Protocol

Standard Resource Description Services
GRIS

- Grid Resource Information Service
- Front end: OpenLDAP server (protocol processing, authentication, result filtering)
- Back end: specific information providers
  - IPs added by specifying type of information provided and routines implementing GRIS API
- Default: port 2135
- Can be configured to register itself with aggregate directory services (GIIS)
GRIS

- Incoming request is authenticated and parsed
- Determine appropriate local information provider
- Is there up-to-date data cached (time-to-live specified per provider)
- Query dispatched to local information provider (using internal API)
- Results returned
GIIS

- Grid Index Information Service
- Aggregate directory with hierarchical structure
- Front end: OpenLDAP server (protocol processing, authentication, result filtering)
- Accepts registration requests from child GRIS/GIIS instances
- Single command to GIIS can obtain information from multiple GRISs
MDS Deployment

GridLab VO

EGridVO

GIIS

LSU

GRISes
LDAP

- Lightweight Directory Access Protocol
  - open multiplatform standard for accessing directory services
  - Based on X.500 (Directory Access Protocol, DAP) (also open but too complex and not adapted to TCP/IP)
  - Defines protocol for exchanging directory service commands between client/server
  - API for adding LDAP functionality to applications
  - LDAP SDKs implementing this API are available (OpenLDAP is used by MDS)
  - Exists in fast moving standards body IETF
LDAP

- **Main functions addressed**
  - Naming of directory entries
  - Structure of directory information
  - Client access to directory information
  - Distributed storage/access (referrals)
  - 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
How is Data Stored In MDS

- **MDS directory structure follows the LDAP model**
  - Directory information tree (DIT) hierarchy
  - Object class definitions

- **Directory Information Tree (LDAP)**
  - Hierarchical view of all directory data
  - Tree-based search system for data, subtrees can be distributed or replicated
  - Directory contents are object classes and entries
  - Object classes: what kind of information
  - Entries: group related information
  - Objects uniquely named by position in the tree
How is Data Stored in MDS

- Every node in tree is an “entry” (Directory Service Entry (CSE))
- Entries contain records to describe real and abstract computing objects (users, computers, disks, applications)
- Content of a record is a pairs of attributes and values
- Attributes each have a “type” and a “value” (type is created by associating an object class)
MDS Directory Structure

- For computational grids the root of the tree is “o=Grid” (o=organization)
- Each entry in the tree can be referred to by a Distinguished Name (DN), which is usually the first attribute of an entry
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
Searching an MDS Server

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

- Default grid-info-search options
  -h mds.globus.org  \(MDS\) server
  -p 389           \(MDS\) port
  -b “o=Grid”     \(search\) start point
  -T 30           \(LDAP\) query timeout
  -s sub          \(scope\) = subtree
  \(alternatives:\)
    \(base\) : lookup this entry
    \(one\) : lookup immediate children
Filtering

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

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

- Distributed access to distributed data, focusing on reading (large) datasets and creating new datasets
  - Large: terabytes to petabytes
  - Datasets:
    - Simulation data (e.g. Cactus Numerical Relativity)
    - Experimental/observational data (e.g. CERN, LIGO, NVO)
    - Information (e.g. Library of Congress)
    - Data bases

- Data Grid not a good name, because also closely involves computational resources
Data Movement

- Move data between storage systems or between programs or between storage systems and programs
- Data movement is the foundation of just about everything on the Grid
- Efficiency very important (large files, WANs)
- Data may be filtered before transfer (prefetch analysis ... “virtual data”)
- Reliable file transfer: maintain state on operations to retry failed operations
GridFTP

- Designed (Globus) as fundamental data access and data transport service
  - Uniform interface to different storage systems (e.g. hierarchical, disk, storage brokers)
  - Incompatible data access protocols by these partition data on the Grid
  - Provides extensions to FTP
Common Data Transfer Mechanism

- FTP (File Transfer Protocol) is attractive because
  - Widely implemented and well-understood IETF standard protocol
  - Well defined architecture for extensions, with dynamic discovery of extensions
  - Supports transfers between client and server, and third party transfers between two servers.
FTP

- File Transfer Protocol
  - http://www.ietf.org/rfc/rfc0959
  - provides the basic elements of file sharing between hosts. FTP uses TCP to create a virtual connection for control information and then creates a separate TCP connection for data transfers. The control connection uses an image of the TELNET protocol to exchange commands and messages between hosts.
GridFTP

- Extension of FTP standard (by Globus team)
  - Automatic negotiation of TCP buffer/window sizes
  - Parallel data transfer
  - Third party control of data transfer
  - Partial file transfer
  - Security (GSSAPI authentication with optional integrity/privacy)
  - Support for reliable file transfer (fault recovery methods, restart of failed transfers)
GridFTP

- Can be used both for access data, and to move data.
- For accessing data, server-side processing allows the inclusion of user-written code that can process data prior to transmission.
Third Party Transfer

Data transfer:
- GridFTP Server to GridFTP Server

Control:
- Client to GridFTP Server

Status:
- GridFTP Server to Client
Reliable File Transfer

State of transfer

Handle returned for monitoring

Request

RFT Client

RFT Factory

RFT Instance

RFT instance started
Replica Location Service: RLS

- Keep track of copies (replicas) of files
  - Examples?
- Files are “registered” with RLS registry, users or services can query for the location of files.
- RLS can be distributed (protect from component failure)
- Logical file name: unique identifier for the contents of a file
- Physical file name: location of a copy of the file on a physical storage device
- One logical file name can point to multiple physical file locations
- Can also associate attributes (eg filesize)
Coursework

- Work through to completion the exercises we started in the lab class on Monday.
- Any problems email the class mail list (note: credit is given both for sending and replying to mails)
- Google answers many questions!
- Hand in:
  - Stdout from running Cactus remotely using both GRAM and GAT (using simple example)
  - Write a small GAT application which moves the executable and then runs it (start from the simple example above) [use C, C++ or Python]
- Note: the web page will not be up to date until tomorrow or Friday ... working with Archit on this.