5.5 GT Resource management

*Grid Resource Allocation Management (GRAM)*

GRAM – module supplying remote execution and job status following

**GT2 GRAM** ([http://www.globus.org/toolkit/docs/2.4/gram/](http://www.globus.org/toolkit/docs/2.4/gram/))

- `globusrun` command

- *Resource Specification Language – RSL*

- `gatekeeper`-daemon

- *Global Access to Secondary Storage – GASS*
  
  **GT2 GASS** ([http://www.globus.org/toolkit/docs/2.4/gass/](http://www.globus.org/toolkit/docs/2.4/gass/))
GRAM overview:
globusrun-command

- (as glite-wms-job-submit)
- request to execute a job on remote machine
- transfer of executables and transfer of output files

Resource Specification Language RSL (JDL)

- language for client job description
- all job resources given in RSL, like
  - executable
  - constraints
    * like needed RAM
gatekeeper

- builds secure communication between client and server
- talks with GRAM client (globusrun) and authenticates job execution rights
- forks after authentication and creates job manager to whom delegates all communication with the client

Job manager

Creates interface to local job managers (load balancers) like

- OpenPBS, LSF, LoadLeveler, SunGE etc.
Job manager functions:

- RSL interpretation
  - segments RSL scripts into commands
- sends commands to local job managers (PBS, LSF etc.)
- callbacks to clients, if needed
- receive status and cancellation requests from client
- sends resulting files and output to client using GASS if needed
Global Access to Secondary Storage – GASS

- A way for result delivery to clients

- GSI secure transfer possibilities

- can be accessed through `globusrun` and `gatekeeper` commands by job manager
Dynamically-Updated Request Online Coallocator (DUROC)

- GT2 DUROC (http://www.globus.org/toolkit/docs/2.4/duroc/)
- Sending jobs to different jobmanagers in different network locations
5.6 GT Information services

**Monitoring and Discovery Service (MDS)**

MDS enables access to static and dynamic information about grid resources.

MDS components:

- **GRIS** - *Grid Resource Information Service*
- **GIIS** – *Grid Index Information Service*
- **Information Provider**
- **MDS client.**

**MDS web-site** ([http://www.globus.org/toolkit/docs/2.4/mds/](http://www.globus.org/toolkit/docs/2.4/mds/))
MDS overview:
Resource information – objects managed by MDS, like

- Infrastructure components
  - for example:
    * name of job manager
    * running job name

- Hardware resources
  - for example:
    * network interface
    * IP address
    * RAM size
GRIS - *Grid Resource Information Service*

- Local resource information manager
- GRIS can register its information with GIIS (but GRIS himself does not accept registration requests)
- GRIS information being updated on request
- Information kept until predefined time (TTL – *time-to-live*) and is deleted in case of no requests
- In case of later requests information collected again from corresponding sources
GIIS – *Grid Index Information Service*

- Database keeping
  - resource information indeces registered by GRIS
  - information about other GIIS-es
- can be considered as information server over all grid
- has hierarchical structure similarly to DNS
- Each GIIS has his own name
  - user can specify GIIS name for information queries
Information Provider

- translates properties, status information etc. to a suitable format for GRIS

- to add one’s resource to the grid, a corresponding information provider needs to be created

MDS client

- Software based on LDAP (Lightweight Directory Access Protocol) client command `ldapsearch`
  
  - Resource lookup command one wants to run first in grid environment

MDS hierarchical structure: Similar mechanism to DNS: lower level GRIS and GIIS register with higher level GIIS-es
5.7 Data management

GridFTP

- performs secure and reliable data transfer between grid nodes
- word GridFTP denotes:
  - protocol
  - server
  - as well as a number of *helper tools*
GidFTP protocol

- data transfer protocol for grid
- based on FTP, but also adds:
  - multithreaded transfer
  - automatic tuning
  - security based on GSI
- Possible to make server-to-server transfers
- Partial file possibility
GridFTP server and client

- `in.ftpd`
- `globus-url-copy`

Usual file movement with GridFTP:
Server-to-server file transfer with GridFTP:
GridFTP tools

- **gsi-ncftp** – used for communication with GridFTP server

- GASS API package, used by GRAM to transfer output files from server to client
6 Globus Toolkit 3

6.1 OGSA (*Open Grid Services Architecture*)

OGSA – preliminary specification for distributed systems integration achieved with the help of following main components:

- interfaces
- behaviours
- resource models
- bindings

etc., which altogether forms OGSA Platform.
Why is OGSA needed?
Short answer: in the same reasons like architecture in building industry, due to:

• all sizes of grids
• consisting of different substances
• used for diverse purposes
• having different throughput and volumes
• needing different security levels
• needing diverse levels and types of infrastructure
Some OGSA terms:

- Identity establishment and authentication negotiation
- Policy expression and negotiation
- Service discovery, monitoring, and management
- Service level agreement negotiation and monitoring
- Virtual organization membership management and communication
- Hierarchical service collection
- Data resource integration into computations
- Distributed resource management across heterogeneous platforms
- Seamless Quality of Service (QoS) delivery
- Common base for autonomic management solutions
• Common infrastructure building blocks

• Open and published Interfaces

• Integration with standards: SOAP, XML, etc.

• Seamless integration with existing IT resources

OGSA architecture layers can be implemented:

• in different software products and components

• by different developers

  – (organisations, software companies, open communities throughout the world)
6.2 OGSA Programming Model (PM)

Developing new service in OGSA-system one has to consider the grid-service programming model

OGSA PM lists certain features important to distributed/grid systems (Some obligatory ones, some appearing only with a subset of services):

- Factory
- Registry
- Discovery
- Life cycle
- Query service data
- Notification
- Reliable invocation
Only contact between service and its user is the user interface

- Service user interfaces are described in WSDL (*Web Services Description Language*)

- For OGSI a few propositions for enhancing WSDL standard

- See e.g. World Wide Web Consortium (W3C) website ([http://www.w3.org/2002/ws/](http://www.w3.org/2002/ws/)) about WSDL
OGSA + OGSI = Grid Services

Grid services - web services with certain properties (interfaces, behaviour) determining how clients interact with Grid services.

What is the difference between OGSA and OGSI?

or: what is the difference between architecture and infrastructure?

Architecture like OGSA defines grid service, infrastructure as OGSI implements it with special tools to create a real grid service

OGSA and OGSI relation:
OGSI - like web service, but not completely?

In web service specification (at least in preliminary one):

- service does not have state (memory)

- lacks transiency

  What is transiency?

  - data available to one client available also to all others (also for modification)
  
  - services will continue their existence also after all clients have finished their work
Life cycle of a service following OGSI standard:

OGSI clients communicate with factories which build up a set of services.
Each service instance has unique Grid Service Handle (GSH).
Client wishing to access a service, resolves
GSH → GSR (Grid Service Reference), which he uses to contact the service in factory.
Client can close a service in the end with a special command or can trust garbage collection built into OGSI depending on service life cycle.
6.4 Web services

Web services are basis for grid services (based on OGSI and OGSA)

Web services – distributed systems technology enabling creation of applications based on client/server model

Web services are:

• independent of platform and language

• use open and known protocols, like HTTP

CORBA (Common Object Request Broker Architecture) and EJB (Enterprise JavaBeans) – meant for tightly coupled clients-server application creation

Web services – clients need not know in advance about the services before contacting the server
Web services (in general):

- are not meant to remember values from one call to another
- no transiency property (given above)

Web services use for communication:

- SOAP (*Simple Object Access Protocol*) and
- XML (*Extensible Markup Language*) grammar

HTTP main web service protocol. About XML, SOAP and HTTP see [World Wide Web Consortium (W3C) website](http://www.w3.org/)
Web service client-server model:
6.5 Web service invocation

1. Client intends to find a web service accessing UDDI (Universal Description, Discovery, and Integration) registry

2. UDDI answers with one service address in URI (Uniform Resource Identifier) form. **Example of URI:**
   
   http://webservices.example-site.com/application/desired-service
   
   (Similarity with URL, as web services usually in web containers).

3. Client knows now the service address but not yet how to invoke it (will enquire about it)

4. Server returns a document in WSDL language, describing thoroughly interface of given service
5. Client knows now how to envoke the service. Envoking can be achieved with the help of different protocols. SOAP (*Simple Object Access Protocol*) – most used – enables information exchange in decentralised distributed environments. Messages sent using XML.

6. Web service sends back SOAP-message in XML-grammar
1. Where can I find a Web Service that does X?
   - UDDI Registry

2. Server A is capable of doing X!
   - UDDI

3. How exactly should I invoke you?
   - SOAP

4. Take a look at this:
   - WSDL

5. Request operation X
   - SOAP

6. Result of operation X
Web service development – usually no need to worry about SOAP and WSDL protocols:

- special stubs for protocol generation:
6.7 Grid services

Grid services based on web services

- OGSI defined additional mechanisms for grid service
  - creation
  - management

**Naming**

Services addressed with URI-s, called **GSH (Grid Service Handle)** in grid case

- GSH: points to a grid service;
  - HTTP/S URL based GSH scheme

- **GSR (Grid Service Reference)** specifies, how to communicate with grid service
  - SOAP on top of HTTP/S, such that GSR-s are in WSDL format
Service Data Elements – SDE

**SDE** - structured dataset connected to a *service instance*

- Clients, servers can:
  - access SDE
  - make additions to SDE
  - make changes to SDE

- Each service instance has
  - a standard set of SDE-s
  - a set of SDE-s particular to the current instance

There exist call-back service or *Notification*
6.8 Notification

- Notifications can be sent in case of:
  - state changes in SDE-s
  - after the end of service associated with a SDE

- A Grid Service can be configured to be a notification source,

- certain clients can be notification sinks (or subscribers)
Notification

1. Client subscription request
2. Subscription expression
3. Service subscription Instance
4. Notification Source
5. Notification message
6. Notification Sink

Service Instance that sends notification
Service Instance that receives notification
subscriber
6.9 Service life-cycle

Factory – mechanism for invoking grid service instances
6.10 GT3 organisation

- GT3 core – gray
6.11 GT3 Hosting environments

- server-environment, (OS), where GT3 runs, 4 possibilities:

1. GT3 Embedded hosting environment
2. Stand-alone hosting environment
3. Servlet (Java Engine) hosting environment
4. GT3 in EJB (Enterprise JavaBeans) environment
6.12 GT3 system level services

GT3 core system level services:

- Administration service – allowing to “ping” and close GT3 environment

- Logging service – log-filter management for system monitoring

- Management service – allow to follow system performance/load, to [de]activate services
6.13 GT3 base services

Not part of *GT3 core* but need to be installed

- **job management** – submitting jobs to the grid and for following job status/progress; client command:
  
  *managed-job-globusrun* – envokes MMJFS (*Master Managed Job Factory Service*) for job submission. [GRAM (MMJFS)](http://www.globus.org/toolkit/docs/3.0/gram/)

- Index services
  
  client command: *ogsi-find-service-data*

[**Globus Information Services**](http://www.globus.org/toolkit/docs/3.2/infosvcs/ws/key/index.html)

- File transfer services
  
  *Reliable File Transfer (RFT) Service (multiRFT)*
6.14 Comparison of GT2 and GT3 main components

<table>
<thead>
<tr>
<th>GT2 components</th>
<th>GT3 services</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAM (gatekeeper)</td>
<td>MMJFS</td>
</tr>
<tr>
<td>MDS GIIS</td>
<td>Index services</td>
</tr>
<tr>
<td>MDS GRIS</td>
<td>SDE in MMJFS</td>
</tr>
<tr>
<td>GridFTP server</td>
<td>GridFTP server</td>
</tr>
<tr>
<td>GRAM Reporter</td>
<td>SDE in Managed Job Service</td>
</tr>
<tr>
<td>globus-url-copy</td>
<td>Reliable File Transfer Service</td>
</tr>
</tbody>
</table>
Comparing GT2, GT3 and GT4

Globus Toolkit 2

- GT2 (ver.2.4, 2002)
- GRAM, MDS, GridFTP, GSI

(http://www.globus.org)
Comparing GT2, GT3 and GT4

Globus Toolkit 3

GT3 (ver.3.2, mid-2004): redesign
– based on OGSA (Open Grid Service Architecture) and OGSI (Open Grid Services Infrastructure) specifications
– The term “Grid services” introduced (OGSI being abandoned).

(http://www.globus.org)