London e-Science Centre

Imperial College London

GridSAM Tutorial

William Lee and Steve McGough

14th September, 2005

open middleware infrastructure institute

GridSAM London e-Science Centre

GridSAM Overview

Grid Job **S**ubmission **a**nd **M**onitoring Service

- What is GridSAM to the end-users?
  - In the OMII 2.0 server distribution
    - A Job Submission & Monitoring Web Service
  - In the OMII 2.0 client distribution
    - Set of command-line tools for job submission, monitoring, control and file transfer.
    - Describe jobs and their file staging requirements in a standard job submission language (JSDL)

GridSAM Overview

Grid Job **S**ubmission **a**nd **M**onitoring Service

- What is GridSAM to the resource owners?
  - A Web Service to expose execution resources
    - Single machine through ***Forking*** or ***SSH***
    - ***Condor Pool***
    - ***Grid Engine 6*** through ***DRMAA***
    - ***Globus 2.4.3*** exposed resources
    - …

GridSAM Overview

Grid Job **S**ubmission **a**nd **M**onitoring Service

- What is GridSAM to Grid developer?
  - Develop plug-ins for GridSAM to
    - expose proprietary resource management system
    - expose other file transfer mechanisms

What's not?

- GridSAM is not
  - a scheduling service
    - It's the role of the underlying launching mechanism
    - It's the role of a super-scheduler that brokers jobs to a set of GridSAM services
  - a provisioning service
    - GridSAM runs what's been told to run
    - GridSAM does not resolve software dependencies and resource requirements

London e-Science Centre

Imperial College London

JSDL Primer

With thanks to:

Andreas Savva, Fujitsu Laboratories

Michel Drescher, Fujitsu Laboratories of Europe

And the JSDL group for some of the material for these slides

JSDL Introduction

- JSDL stands for *Job Submission Description Language*
  - A language for *describing the requirements of computational jobs for submission* to Grids and other systems.
  - In this case GridSAM
- A JSDL *document* describes the job requirements
  - *What to do, not how to do it*
- JSDL *does not* define a submission interface or what the results of a submission look like
  - This is done by GridSAM
  - Or how resources are selected, or …
    - *To come in the future?*

Primary deliverables and status

- A specification for an *abstract* standard JSDL
- Independent of language bindings, including:
  - JSDL attributes; attribute relationships and ranges of attribute values.
  - Currently draft version 1.0
- A normative XML Schema
  - Currently draft version 1.0
- In reality the specification and schema are one document
  - Status: In 90-day public comment, ending Sep 10.
  - Public comment draft
    - http://www.ggf.org/Public_Comment_Docs/Documents/July-2005/draft-ggf-jsdl-spec-021.pdf

JSDL Document (1)

- A JSDL document is an XML document
- It may contain
  - Generic (job) identification information
  - Application description
  - Resource requirements (main focus is computational jobs)
  - Description of required data files
- Out of scope (at least for version 1.0)
  - Scheduling
  - Workflow
  - Security
  - …

JSDL Document (2)

- A JSDL document is a template ...
  - It can be submitted multiple times and can be used to create multiple job instances
- ... so JSDL does not define attributes to describe the state of a running job
  - No start time, end time, submission status, or even JobID
- A JSDL document can be composed with other languages (open content model)
  - For example to express scheduling, security, etc, requirements in more detail

JSDL Document Life Cycle

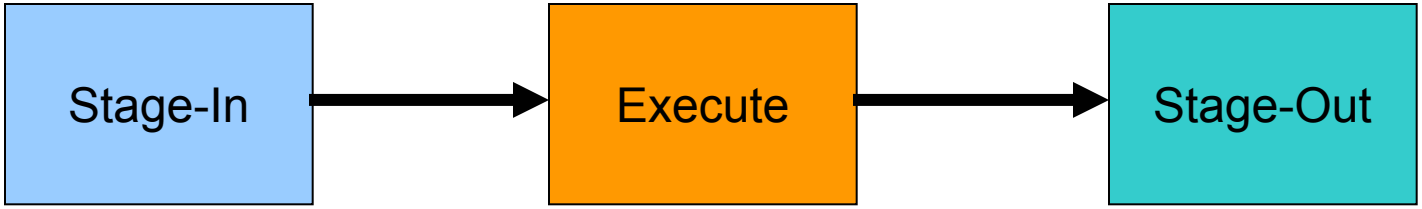
- A JSDL document may be
  - *Abstract*
    - Only the minimum information necessary
      - For example, *application name* and *input files*
    - Runnable at sites that understand this level of description
  - *Refined*
    - More detail provided
      - Target site, number of CPUs, which data source
    - May be refined several times
  - *Tied* to a specific site/system
    - *Incarnated* (Unicore speak); or
    - *Grounded* (Globus speak)

GridSAM

- *This model is supported/allowed but not required by JSDL*

Data Staging Requirement

- JSDL does not define workflow
  - *But* … data staging is a common requirement for any meaningful job submission
- Assume simple model
  - Stage-in – *Execute* – Stage-Out
- Files required for execution
  - Files are staged-in before the job can start executing
- Files to preserve
  - Files are staged-out after the job finishes execution



JSDL Document Structure Overview

London e-Science Centre

Imperial College London

```
<JobDefinition>
   <JobDescription>
      <JobIdentification ... />?
      <Application ... />?
      <Resources... />?
      <DataStaging ... />*
   </JobDescription>
</JobDefinition>
```

Note:

None	[1..1]
?	[0..1]
*	[0..n]
+	[1..n]

Job Identification Element


```
<JobIdentification>
    <JobName .../>?
    <Description .../>?
    <JobAnnotation .../>*
    <JobProject .../>*
  <xsd:any##other>*
</JobIdentification>?
```

Extensibility point

Example:

```
<jsdl:JobIdentification>
  <jsdl:JobName>
    My Gnuplot invocation
  </jsdl:JobName>
  <jsdl:Description>
    Simple application ...
  </jsdl:Description>

  <tns:AAId>3452325707234
  </tns:AAId>

</jsdl:JobIdentification>
```

Application Element


```
<Application>
   <ApplicationName ... />?
   <ApplicationVersion ... />?
   <Description ... />?
         <xsd:any##other>*
</Application>
```

How do I explicitly define applications?

==> See next slide!

Example:

```
<jsdl:Application>
   <jsdl:ApplicationName>
      gnuplot
   </jsdl:ApplicationName>
   <jsdl:ApplicationVersion>
      5.7
   </jsdl:ApplicationVersion>
   <jsdl:Description>
      Use the gnuplot application v5.7
      regardless where it is installed on
      the target system
   <jsdl:Description>
</jsdl:Application>
```

Application: POSIXApplication extension


```
<POSIXApplication>
  <Executable .../>
  <Argument .../>*
  <Input .../>?
  <Output .../>?
  <Error .../>?
  <WorkingDirectory .../>?
  <Environment .../>*
  ...
</POSIXApplication>
```

POSIXApplication is a normative JSDL extension

Defines standard POSIX elements

- stdin, stdout, stderr
- Working directory
- Command line arguments
- Environment variables
- POSIX limits (not shown here)

Resources Element


```
<Resources>
   <CandidateHosts .../>?
   <FileSystem .../>*
   <ExlusiveExecution .../>?
   <OperatingSystem .../>?
   <CPUArchitecture .../>?
   <IndividualCPUSpeed .../>?
   <IndividualCPUTime .../>?
   <IndividualCPUCount .../>?
   <IndividualNetworkBandwidth .../>?
   <IndividualPhysicalMemory .../>?
   <IndividualVirtualMemory .../>?
   <IndividualDiskSpace .../>?
   <TotalCPUTime .../>?
   <TotalCPUCount .../>?
   <TotalPhysicalMemory .../>?
   <TotalVirtualMemory .../>?
   <TotalDiskSpace .../>?
   <TotalResourceCount .../>?
   <xsd:any##other>*
</Resources>*
```

Example:
One CPU and at least 2 Megabytes of memory

```
<jsdl:Resources>
   <jsdl:CPUCount>
      <Exact> 1.0 <Exact>
   </jsdl:CPUCount>
   <jsdl:PhysicalMemory>
      <LowerBoundedRange>
         2097152.0
      </LowerBoundedRange>
   </jsdl:PhysicalMemory>
</jsdl:Resources>
```

Currently not supported

But soon!

DataStaging Element


```
<DataStaging>
  <FileName ... />
  <FileSystemID ... />?
  <CreationFlag ... />
  <DeleteOnTermination ... />?
  <Source ... />?
  <Target ... />?
</DataStaging>*
```

Example:
Stage in a file (from a URL) and name it “control.txt”. In case it already exists, simply overwrite it. After the job is done, delete this file.

```
<jsdl:DataStaging>
  <jsdl:FileName>
    control.txt
  </jsdl:FileName>
  <jsdl:Source>
    <jsdl:URI>
        http://foo.bar.com/~me/control.txt
    </jsdl:URI>
  </jsdl:Source>
  <jsdl:CreationFlag>
    overwrite
  </jsdl:CreationFlag>
  <jsdl:DeleteOnTermination>
    true
  </jsdl:DeleteOnTermination>
</jsdl:DataStaging>
```

Hello World


```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition
   xmlns:jsdl="http://schemas.ggf.org/2005/06/jsdl"
   xmlns:jsdl-posix=http://schemas.ggf.org/jsdl/2005/06/apps/posix>
<jsdl:JobDescription>
   <jsdl:Application>
      <jsdl-posix:POSIXApplication>
         <jsdl-posix:Executable>/bin/echo<jsdl-posix:Executable>
         <jsdl-posix:Argument>hello</jsdl-posix:Argument>
         <jsdl-posix:Argument>world</jsdl-posix:Argument>
      </jsdl-posix:Application>
   </jsdl:Application>
  </jsdl:JobDescription>
</jsdl:JobDefinition>
```

Standard In and Standard Out


```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition
   xmlns:jsdl="http://schemas.ggf.org/2005/06/jsdl"
   xmlns:jsdl-posix=http://schemas.ggf.org/jsdl/2005/06/apps/posix>
<jsdl:JobDescription>
   <jsdl:Application>
      <jsdl-posix:POSIXApplication>
         <jsdl-posix:Executable> /bin/echo  <jsdl-posix:Executable>
         <jsdl-posix:Input>/dev/null</jsdl-posix:Input>
         <jsdl-posix:Output>program.out</jsdl-posix:Output>
         <jsdl-posix:Argument>hello</jsdl-posix:Argument>
         <jsdl-posix:Argument>world</jsdl-posix:Argument>
      </jsdl-posix:Application>
   </jsdl:Application>
</jsdl:JobDescription>
</jsdl:JobDefinition>
```

Staging data Out


```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns:jsdl="http://schemas.ggf.org/2005/06/jsdl"
   xmlns:jsdl-posix=http://schemas.ggf.org/jsdl/2005/06/apps/posix>
<jsdl:JobDescription>
   <jsdl:Application>
      <jsdl-posix:POSIXApplication>
         <jsdl-posix:Executable> /bin/echo  <jsdl-posix:Executable>
         <jsdl-posix:Input>/dev/null</jsdl-posix:Input>
         <jsdl-posix:Output>program.out</jsdl-posix:Output>
         <jsdl-posix:Argument>hello</jsdl-posix:Argument>
         <jsdl-posix:Argument>world</jsdl-posix:Argument>
      </jsdl-posix:Application>
   </jsdl:Application>
   <jsdl:DataStaging>
         <jsdl:FileName>program.out</jsdl:FileName>
         <jsdl:Target>
            <jsdl:URI>http://foo.bar.com/~me/control.txt</jsdl:URI>
         </jsdl:Target>
         <jsdl:CreationFlag>jsdl:overwrite</jsdl:CreationFlag>
   </jsdl:DataStaging>
  </jsdl:JobDescription>
</jsdl:JobDefinition>
```

Complete example of JSDL document

This example contains lots of information

Real exampes are much shorter

Installing and using the GridSAM Client tools

Client Installation

- Pre-requisite
  - Downloaded and unpacked the OMII 2.0 Client distribution
  - OMII 2.0 Client installed and tested properly

- To install, run

```
$> cd mydownload/
$> cd omii-client-2.0.0/managed_programme
$> OMIIManagedProgrammeClientInstall.sh

OMII Managed Programme client installation


Please enter the location of the OMIICLIENT home directory
(default: /home/myuser/OMIICLIENT):
```

Client distribution at a glance

- GridSAM client is installed in
  - <OMII_CLIENT_HOME>/gridsam
- All command-line tools are installed in
  - <OMII_CLIENT_HOME>/gridsam/bin
    - gridsam-submit
    - gridsam-status
    - gridsam-terminate
    - gridsam-ftp-server
    - gridsam-version
    - myproxy
    - gridsam-file-transfer (To appear in the next version)

Task: Testing the client installation

- Check the GridSAM version

```
$> cd ${OMIICLIENT_HOME}/gridsam/bin

HINT: add ${OMIICLIENT_HOME}/gridsam/bin to your PATH environment

$> ./gridsam-version
gridsam-0.1.4 (0.1 beta 4)
$>
```

Task: Submitting and Monitoring Job

- Submit a ‘sleep’ job to the test server

Service Endpoint
(-s)

```
$> gridsam-submit -s \
   “http://dustpuppy.doc.ic.ac.uk:55554/gridsam/services/gridsam?WSDL” \
   ${OMIICLIENT_HOME}/gridsam/data/examples/sleep.jsdl
urn:gridsam:12298601064fed1701064fef0364000a
```

JSDL File

Job ID of submitted job

```
$> gridsam-status -s \
   “http://dustpuppy.doc.ic.ac.uk:55554/gridsam/services/gridsam?WSDL” \
   urn:gridsam:12298601064fed1701064fef0364000a
```

Task: Submitting and Monitoring Job


```
Job Progress: pending -> staging-in -> staged-in -> active -> executed -> staging-out -> staged-out -> done

--- pending - 2005-09-13 16:01:47.0 ---
job is being scheduled
--- staging-in - 2005-09-13 16:01:47.0 ---
staging files...
--- staged-in - 2005-09-13 16:01:47.0 ---
no file needs to be staged in
--- active - 2005-09-13 16:01:47.0 ---
'/bin/sleep 5' is being forked
--- executed - 2005-09-13 16:01:52.0 ---
'/bin/sleep 5' completed with exit code 0
--- staging-out - 2005-09-13 16:01:52.0 ---
staging files out...
--- staged-out - 2005-09-13 16:01:52.0 ---
no file needs to be staged out
--- done - 2005-09-13 16:01:52.0 ---
Job completed

--------------
Job Properties
--------------
urn:gridsam:exitcode=0
$>
```

Job events

Additional properties associated with the job

Task: Submitting and Monitoring Job

- Retrieving XML status output

```
$> gridsam-status -x -s \
   http://dustpuppy.doc.ic.ac.uk:55554/gridsam/services/gridsam?WSDL \
   urn:g:12298601d1701064fef0364000a
```

Show XML output
(-x)

```
<g:JobStatus xmlns:g="http://www.icenigrid.org/service/gridsam" >
  <g:Stage >
    <g:State>pending</g:State>
    <g:Description>job is being scheduled</g:Description>
    <g:Time>2005-09-13T16:01:47+01:00</g:Time>
  </g:Stage>
  <g:Stage >
    <g:State>staging-in</g:State>
    <g:Description>staging files...</g:Description>
    <g:Time>2005-09-13T16:01:47+01:00</g:Time>
  </g:Stage>
  <g:Stage >
    <g:State>staged-in</g:State>
    <g:Description>no file needs to be staged in</g:Description>
    <g:Time>2005-09-13T16:01:47+01:00</g:Time>
  </g:Stage>
  <g:Stage >
    <g:State>active</g:State>
    <g:Description>'/bin/sleep 5' is being forked</g:Description>
    <g:Time>2005-09-13T16:01:47+01:00</g:Time>
  </g:Stage>
  <g:Property name="urn:gridsam:exitcode">
    <![CDATA[0]]>
  </g:Property>
</JobStatus>
```

Shortcuts

- Tedius to remember and type the service endpoint
  - Refers to ***service by name*** by storing a list of commonly used service endpoints in ~/.gridsam/services.properties

In ${HOME}/.gridsam/services.properties

```
TestService=http://dustpuppy.doc.ic.ac.uk:55554/gridsam/services/gridsam?WSDL
MyOtherService=http://other:8080/gridsam/services/gridsam?WSDL
```

```
$> gridsam-submit -sn TestService myjob.jsdl

$> gridsam-status -sn TestService urn:gridsam:129f924942e214b89c21532
```

Service name
(-sn)

What about input/output?

- Use JSDL DataStaging elements

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl">
   <JobDescription>
      <Application>
         <POSIXApplication xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix">
                  <Executable>/bin/echo</Executable>
            <Argument>hello world</Argument>
            <Output>stdout.txt</Output>
         </POSIXApplication>
            ion>
            aStaging>
            leName>stdout.txt</FileName>
            Flag>overwrite</CreationFlag>
                  <URI>
                        ftp://myftpserver/directory/file.txt
                  </URI>
               <Target>
            </DataStaging>
         </Application>
      </JobDescription>
   </JobDefinition>
```

Indicate the 'virtual' file name to write the standard output

"Target" to indicate the file should be staged out. "Source" is to stage in.

Define how the 'virtual' file should be staged in or out

The URI describes the location of the file to be staged in from or staged out to.

Data Staging

- Supported Data Staging URI Schemes
  - http(s)://<username>:<password>@host:port/path/file (Read-only)
  - ftp://<username>:<password>@host:port/path/file
  - sftp://<username>:<password>@host:port/path/file
  - webdav(s)://<username>:<password>@host:port/path/file
  - gsiftp://host:port/path/file

FTP File Transfer

- GridSAM bundles an unsecured FTP server for testing purpose or small-scale usage

```
$> gridsam-ftp-server -p 19245 -d /directory/to/make/public
2005-09-13 16:49:55,318 WARN [GridSAMFTPServer](main:) /directory/to/make/public is
exposed through FTP ftp://anonymous@146.169.6.129:19245/
325 WARN [GridSAMFTPServer](main:) Please
implication of using anonymous FTP for file
root.dir = /tmp/
g.data = /home/wwhl/.gridsam/ftp-540408:106
FtpServer.server.config.server.host = 146.169.6.129
FtpServer.server.config.port = 19245
Started FTP
```

Port to use for providing FTP service ***(-p)***

Directory to be exposed through FTP ***(-d)***

Job Termination

- Job termination in GridSAM is ***'harsh'*** and ***asynchronous***
  - File staging is not performed when the job is terminated
  - A running job can be terminated by `gridsam-terminate`

```
$> gridsam-terminate -sn TestService urn:gridsam:129f924942e214b89c21532
$>
```

Installing the GridSAM Service

Server Installation


```
$> cd mydownload/
$> cd omii-server-2.0.0/
$> ./OMIIstackInstall.pl

Welcome to the OMII_2 unified installer.
This installer is designed to take you through the installation of
'base', 'extension', 'services', the 'cauchy' application and
any included Managed Programme components.

Do you wish to set up the postgres database, in preparation for installing OMII_2.
Or do you wish to install the entire OMII_2 stack having already set up the database?
Or finally do you wish to install the Managed Programme components upon an existing OMII_2 stack?

1) Database setup only
2) Entire stack, excluding database setup
3) Managed Programme components
> 3
```

Server distribution at a glance

- GridSAM service is installed as a web application in the OMII tomcat servlet container
  - <OMII_HOME>/jakarta-tomcat-5.0.25/webapps/gridsam
- All GridSAM configuration and data files are stored in
  - <OMII_HOME>/jakarta-tomcat-5.0.25/webapps/gridsam/WEB-INF
    - server-config.wsdd - ***Axis Web Service configuration***
    - classes/database.xml - ***Database configuration***
    - classes/jobmanager.xml - ***GridSAM core engine***
    - classes/crypto.properties - ***WS-Security configuration***
    - data/* - ***Hypersonic Database Data***
- Out-of-the-box configuration
  - Uses Hypersonic SQL as the embedded database engine.
  - Uses the ***Forking*** plugin to launch job on the local machine.
  - Requires no additional configuration after installation

Starting & Stopping the Service

- GridSAM is made available when the OMII container is started
  - $> ${OMII_HOME}/jakarta-tomcat-5.0.25/bin/start_base.sh
- The GridSAM service will stop answering requests and persist remaining job stages when the OMII container is being shutdown
  - $> ${OMII_HOME}/jakarta-tomcat-5.0.25/bin/stop_base.sh
- To determine whether GridSAM is started properly
  - Connect to http://<omiihost>:<omiiport>/gridsam with a browser
  - Browse the log file and look for the message
  - $> tail -f {OMII_HOME}/jakarta-tomcat-5.0.25/logs/gridsam.log

```
2005-08-31 12:10:25,036 INFO [JobManagerConfigurator] GridSAM machinery initialising...
2005-08-31 12:10:25,290 INFO [ResourceRegistry] loading module description from classpath jobmanager.xml
2005-08-31 12:10:38,194 INFO [JobManagerConfigurator] GridSAM machinery initialised
```

An Extended Example

- Define and run a “Povray” rendering job
  - Povray executable is staged-in from
    - http://www.doc.ic.ac.uk/~wwhl/povray.bin
  - Input scene file is staged-in from
    - http://www.doc.ic.ac.uk/~wwhl/blob.pov
  - Output scene file is staged-out to an FTP server running locally
  - Povray command-line syntax
  - `povray +Ooutput-file-nname +Wwidth-of-image +Hheight-of-image input-scene-file`
    - $> povray +Ooutput.png +H240 +W320 scene.pov

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl">
  <JobDescription>
    <Application>
      <POSIXApplication xmlns="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix">
                <Executable>bin/povray</Executable>
        <Argument>+Oblob.png</Argument>
        <Argument>blob.pov</Argument>
      </POSIXApplication>
    </Application>
    <DataStaging>
              <FileName>bin/povray</FileName>
       <CreationFlag>overwrite</CreationFlag>
       <Source>
          <URI>http://www.doc.ic.ac.uk/~wwhl/povray.bin</URI>
       </Source>
      </DataStaging>
      <DataStaging>
        <FileName>blob.pov</FileName>
        <CreationFlag>overwrite</CreationFlag>
        <Source>
          <URI>http://www.doc.ic.ac.uk/~wwhl/blob.pov</URI>
        </Source>
      </DataStaging>
      <DataStaging>
        <FileName>blog.png</FileName>
        <CreationFlag>overwrite</CreationFlag>
        <Target>
          <URI>ftp://myhost:19245/output.pov</URI>
        </Target>
      </DataStaging>
  </JobDescription>
</JobDefinition>
```

Advance Service Configuration

Deployment Scenarios

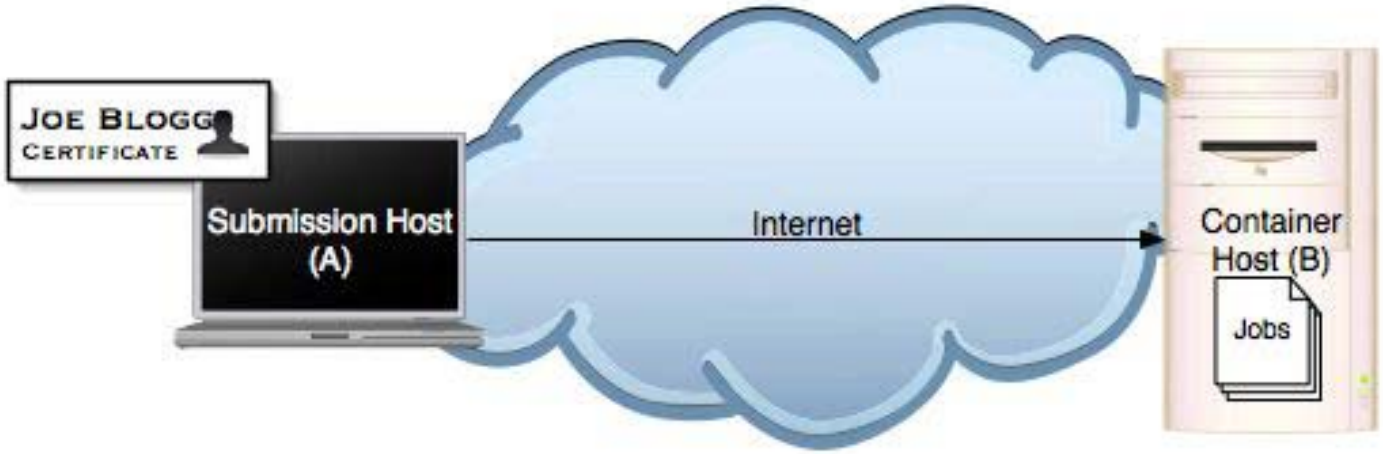
- General configuration for GridSAM core engine
  - ${OMII_HOME}/webapps/gridsam/WEB-INF/classes/jobmanager.xml
  - Templates for various deployment scenarios are available in ${OMII_HOME}/webapps/gridsam/WEB-INF/classes/jobmanager*.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<module id="jobmanager.fork" version="1.0.0">

  <!-- dependent modules -->
  <sub-module descriptor="org/icenigrid/gridsam/resource/config/common.xml"/>
  <sub-module descriptor="org/icenigrid/gridsam/resource/config/embedded.xml"/>
  <sub-module descriptor="org/icenigrid/gridsam/resource/config/fork.xml"/>
  <sub-module descriptor="org/icenigrid/gridsam/resource/config/shell.xml"/>
  <sub-module descriptor="database.xml"/>

  <!-- override the factory defaults here -->
  <contribution configuration-id="hivemind.ApplicationDefaults">
    <!--
    The spooling directory for shell-based job submission
    -->
    <default symbol="spool.directory" value="/tmp"/>
  </contribution>
</module>
```

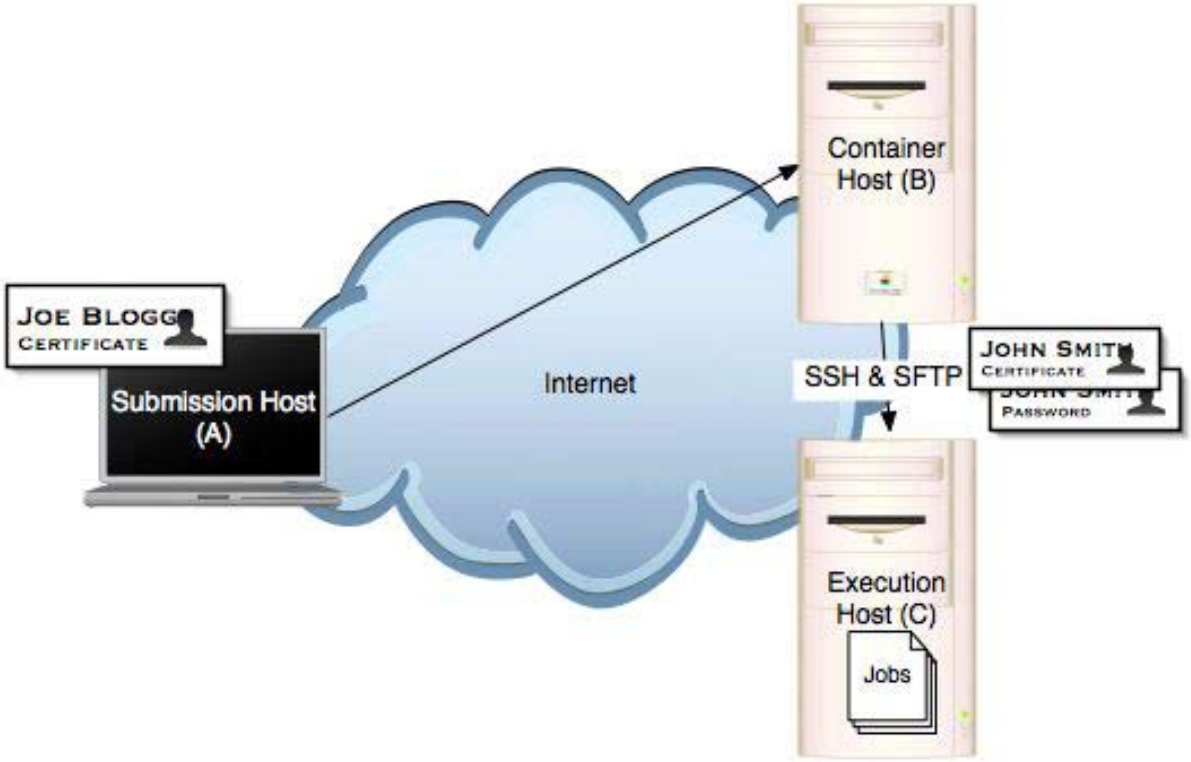
Includes pre-defined sub-module that makes up the runtime engine

For each sub-module, there might be configurable values that can be modified by administrator.

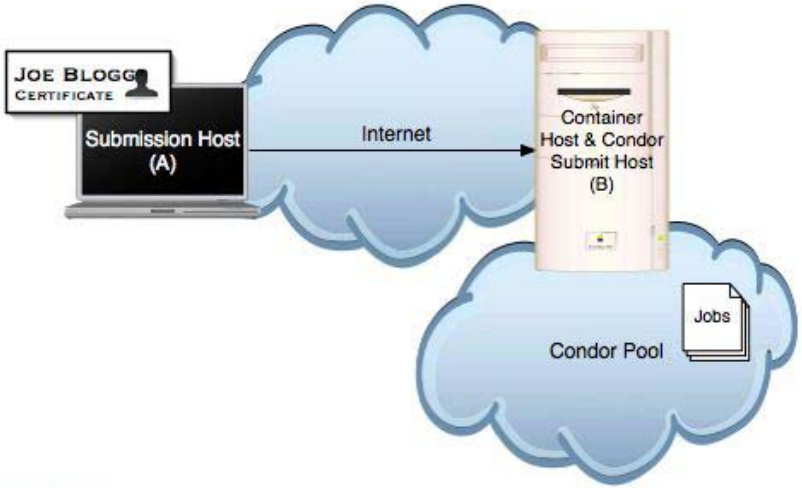
Scenario: Forking

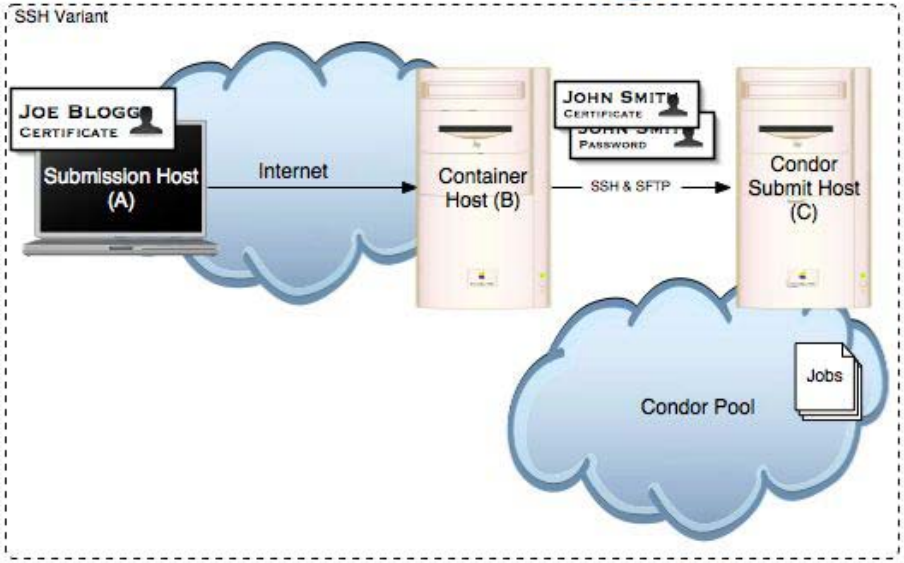


Scenario: Secure Shell (SSH)

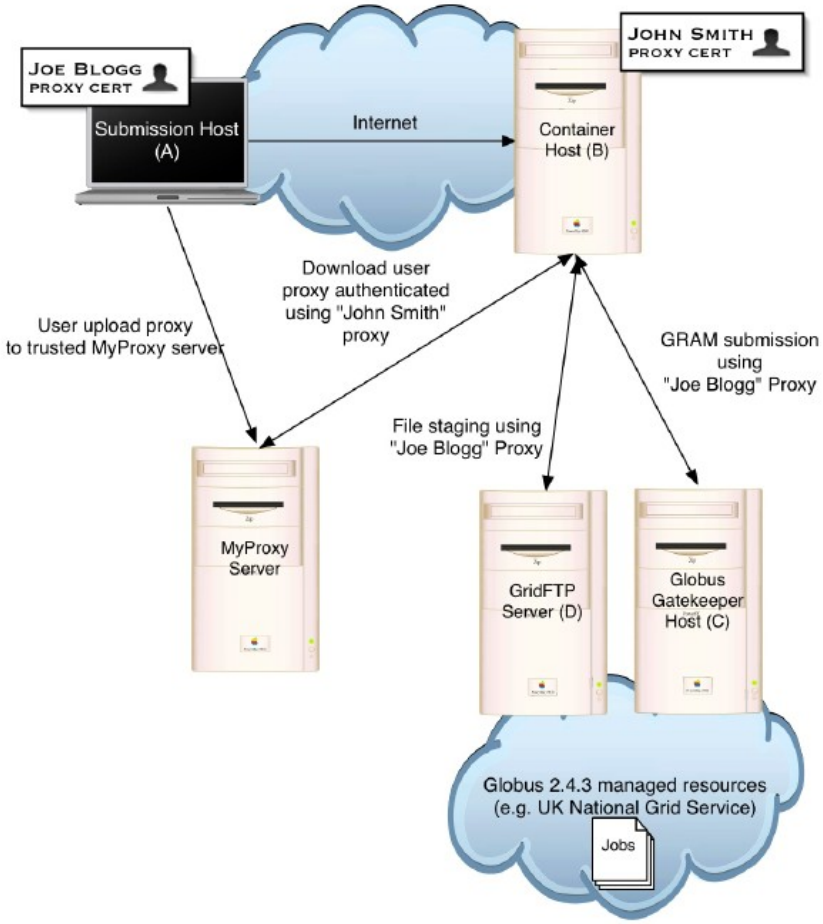


Scenario: Condor Pool

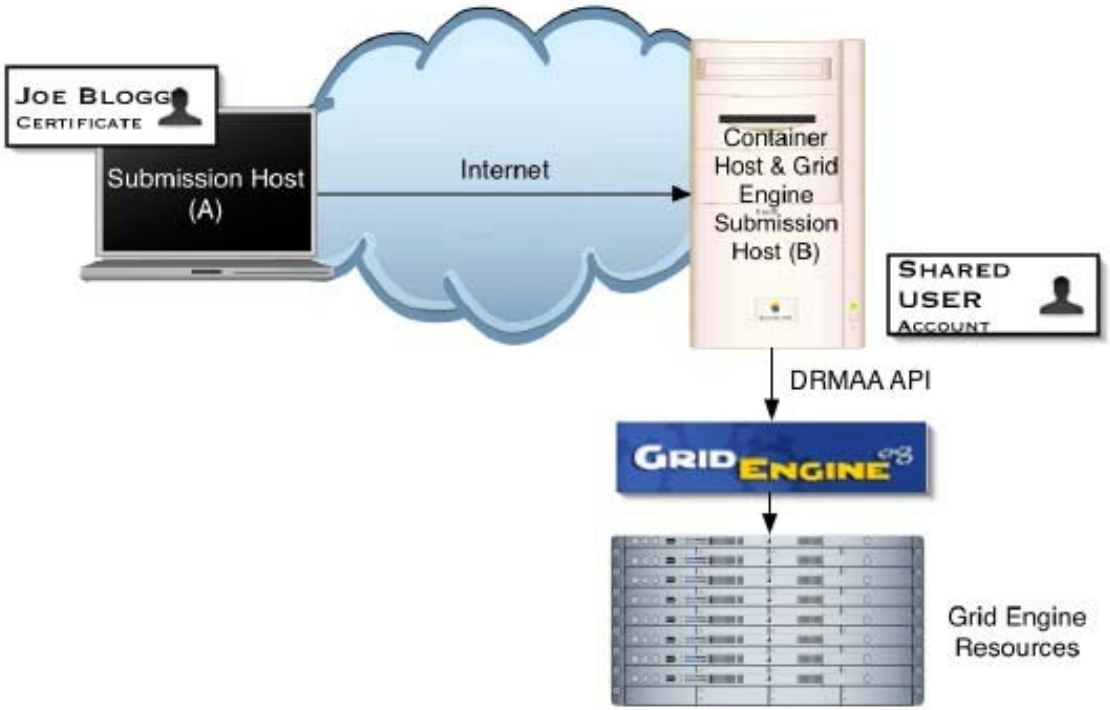




Scenario: Globus 2.4.3



Scenario: Grid Engine 6



Runtime Monitoring

- GridSAM supports the Java Management Extension (JMX)
  - Gather runtime statistics
  - Dynamically change configuration
  - Use standard JMX compliant client (e.g. Jconsole) to manage a running GridSAM service.

What next?

- Visit the GridSAM website
  - http://www.lesc.imperial.ac.uk/gridsam