OpenGridForum
Europe

Managing Computational Activities on the Grid

from Specifications to Implementation
(2nd edition)

CAPACITIES

OGF IPR Policies Apply

- “I acknowledge that participation in this meeting is subject to the OGF Intellectual Property Policy.”
- Intellectual Property Notices Note Well: All statements related to the activities of the OGF and addressed to the OGF are subject to all provisions of Appendix B of GFD-C.1, which grants to the OGF and its participants certain licenses and rights in such statements. Such statements include verbal statements in OGF meetings, as well as written and electronic communications made at any time or place, which are addressed to:
    - the OGF plenary session,
    - any OGF working group or portion thereof,
    - the OGF Board of Directors, the GFSG, or any member thereof on behalf of the OGF,
    - the ADCOM, or any member thereof on behalf of the ADCOM,
    - any OGF mailing list, including any group list, or any other list functioning under OGF auspices,
    - the OGF Editor or the document authoring and review process
- Statements made outside of a OGF meeting, mailing list or other function, that are clearly not intended to be input to an OGF activity, group or function, are not subject to these provisions.
- Excerpt from Appendix B of GFD-C.1: ”Where the OGF knows of rights, or claimed rights, the OGF secretariat shall attempt to obtain from the claimant of such rights, a written assurance that upon approval by the GFSG of the relevant OGF document(s), any party will be able to obtain the right to implement, use and distribute the technology or works when implementing, using or distributing technology based upon the specific specification(s) under openly specified, reasonable, non-discriminatory terms. The working group or research group proposing the use of the technology with respect to which the proprietary rights are claimed may assist the OGF secretariat in this effort. The results of this procedure shall not affect advancement of document, except that the GFSG may defer approval where a delay may facilitate the obtaining of such assurances. The results will, however, be recorded by the OGF Secretariat, and made available. The GFSG may also direct that a summary of the results be included in any GFD published containing the specification.”

- OGF Intellectual Property Policies are adapted from the IETF Intellectual Property Policies that support the Internet Standards Process.

Tutorial outline

- Introduction (A. Ciuffoletti)
- JSDL specification (A. Ciuffoletti)
- BES (A. Konstantinov)
- Glue resource description (B. Konya)
- GIN Security profile (M. Riedel)
- Demos...

Introduction to Job Management

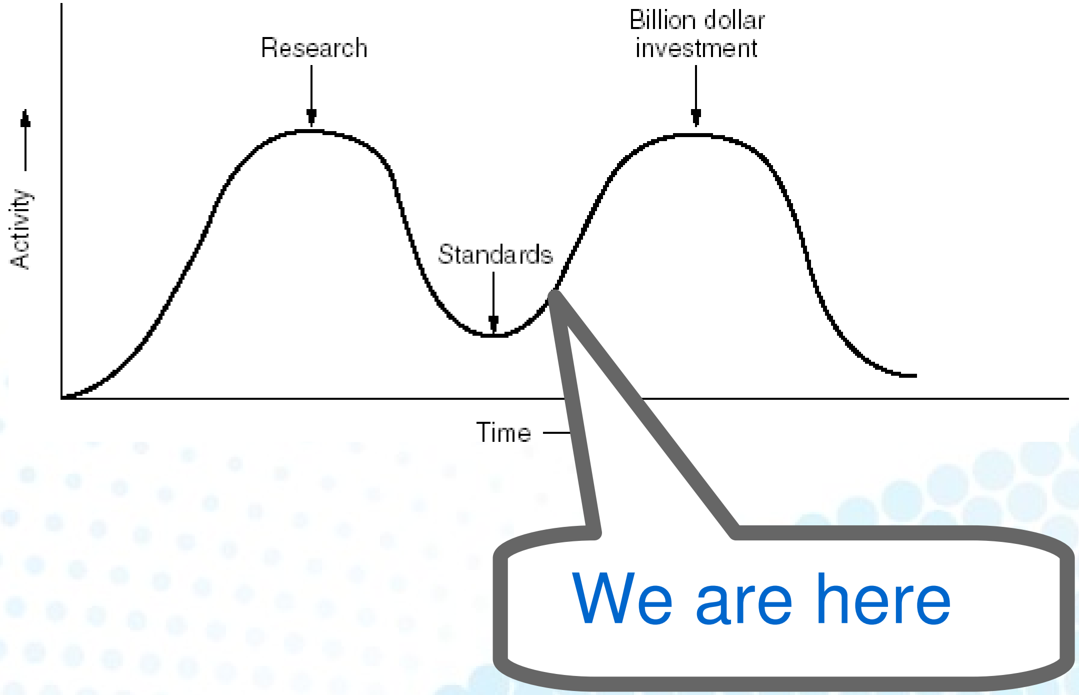
- Job Management is one of the basic functionalities of a Grid
- The user **submits** a (computational) job to an appropriate Execution Service
- The user interacts with the Execution Service to control the **execution** of his job

So, where is the problem?

- Different Grid Systems provide different interfaces for job management
- Such interfaces provide mostly the same functionalities, but with slightly different semantics and/or syntax
  - Different syntax used to **describe** a job
  - Different **execution** models

The two elephants

(from Tanenbaum's “Computer Networks”)



We are here

The role of a standard

- The importance of a standard:
  - protects investments
  - enhances competition and technical progress
  - focusses innovation on relevant aspects
  - avoids technological barriers
- A standard is a long term investment
  - research provides a long range perspective

The Job Submission case

- A Grid provides an interface to users that want to submit a job
- We need a standard since:
  - The user wants to submit tomorrow the job description he writes today (protect investments)
  - The user wants to be able to switch to another Grid if the current one is not good for her
  - The designer doesn't want to care about the syntax to describe jobs

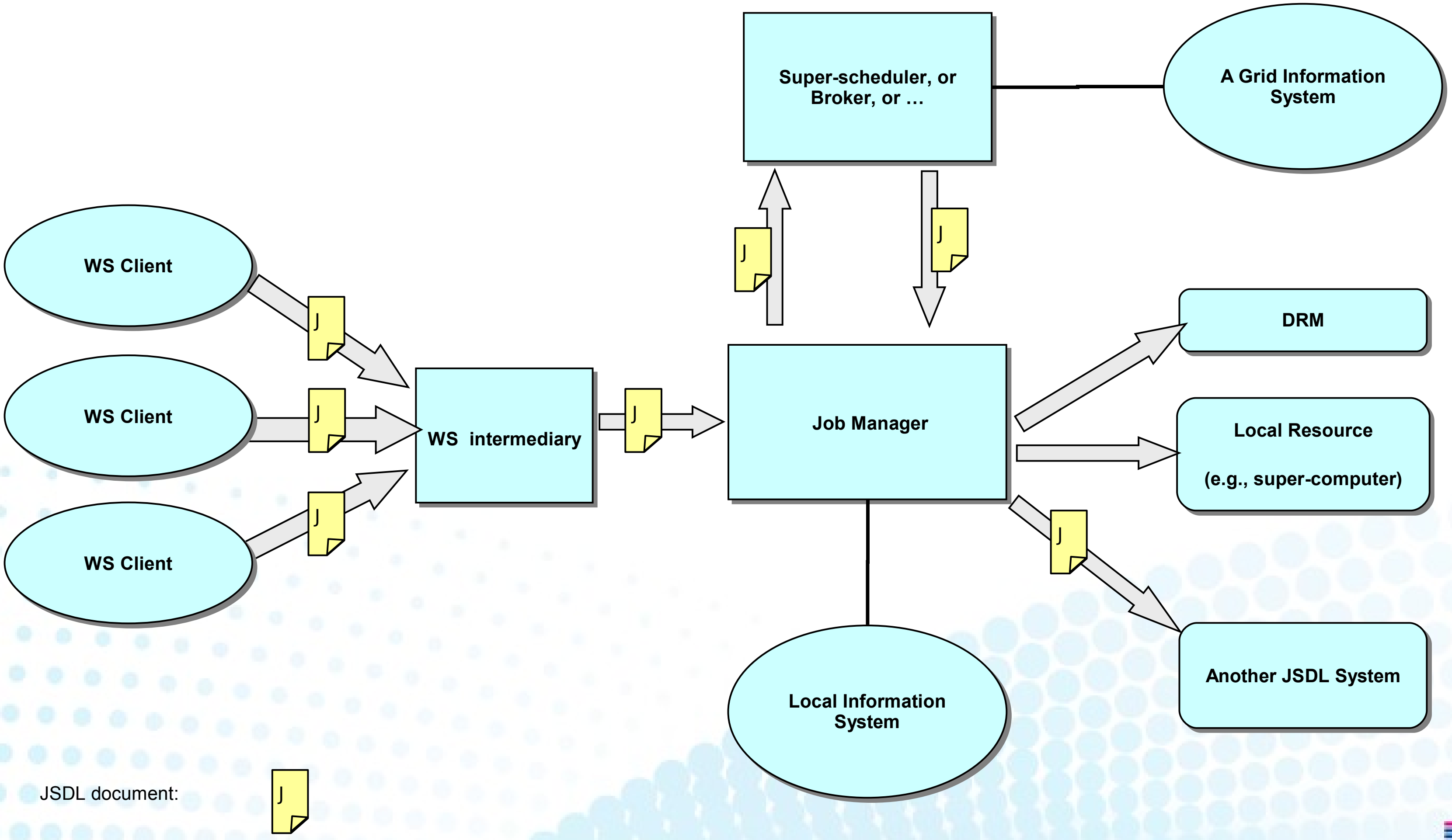
... and also ...

A distributed world!

- Grid environments may involve the interaction of a number of different types of job management systems
  - The same resource may have a different name, or measurement unit
  - The description of a job may be transformed by intermediaries
  - Basically the semantic area is the same (cpu, storage), but variants should be coped with.

Job Submission Scenario

Source: *JSDL Specification*, version 1.0



The Job Execution case

- Grid environments may give distinct ways to interact with a job
  - Some job state may exist only on some platform (e.g. “waiting for a resource”)
  - User/Job interactions may cross intermediaries
  - Some common denominator exists, but not very expressive.

Standardization issues

- Protocol Specifications
  - defines the messages that pass between clients and services
  - the functional states that characterize the agents
  - how messages have a certain effect (i.e., state transition)

Standardization issues

- Application Programming Interfaces
  - define programming language interfaces that expose service functionalities within a client program
  - generally focused on users (clients) of services, rather than service providers

Standardization issues

- Information schemas
  - define the information that is passed between clients and services, or between peer services
  - provide a single vocabulary used to describe resources and activities

OGF for Interoperation

- GiN (2005-…):

  “The scope of the group is to pursue interoperation on 6-8 month horizons using solutions for which there are working implementations available, wherever possible using standards.”

  – Focus on defined objectives

  – Many demos showing effective interoperability:

  http://forge.gridforum.org/sf/projects/gin

OGF for Interoperation

- PGI (2008-...):

  “The objective of this working group is to formulate a well-defined set of profiles ... for job and data management ... with a Grid security and information model that addresses the needs of production grid infrastructures. ”

  – A sprout of GiN

  – Focus on Job Submission

- Our speakers...

Introducing Speakers

- Balazs Konya
  - Chair of the PGI Working Group of OGF
  - Will speak about resource description
- Morris Riedel
  - Founded the GIN Community of OGF
  - Chair of the PGI Working Group of OGF
  - Will speak about security aspects
- Augusto Ciuffoletti
  - Long term research in distributed computing (Univ. of Pisa - INFN-CNAF)

The OGSA/OGF Solution

- Job Submission Description Language (JSDL)
  - an XML notation for describing computational jobs and their submission requirements
- Basic Execution Service (BES)
  - an XML notation for describing the execution dynamics
  - Uses JSDL to describe jobs requirements
- Extensions for both

JSDL Scope

- What JSDL ***is***
  - JSDL is a language for describing the requirements of individual jobs.
- What JSDL ***is not***
  - Does not address the life-cycle of a job
  - Does not address job management
  - Does not consider interactions or dependencies among jobs (workflow)
- Other languages/protocols required (Unix style)

JSDL Syntax (XML)

- A JSDL document is described using XML
- XML is a general purpose language to describe mark-up languages
- An XML document is a tree of **elements**, identified with **tags**

```
<element_name attribute_name="attribute_value">
    Element Content
</element_name>
```

- **Attributes** add a flavour to an element

JSDL Syntax (Schema)

- A JSDL Document is a XML document that matches the JSDL Schema
- A Schema is a set of rules to which an XML document must conform
- A Schema is an XML document
- A Schema introduces types for values to simplify processing of a document

XML (Examples)

- This is the definition of an element in the Schema:

```
<xsd:element name="Recipient" type="xsd:string">
</xsd:element>
```

or

```
<xsd:element name="Recipient" type="xsd:string" />
```

- This is the corresponding element in the Document

```
<Recipient>
    Mario Rossi
</Recipient>
```

XML (New types)

- We can build new types out of primitives ones (numbers, strings etc.)
- We use two basic kinds of types constructors to describe JSDL:
  - **Restriction**: a subset of a primitive type
  - **Sequence**: a sequence of elements of certain types
- The definition of a new type defines also **attributes** for elements of the defined type

JSDL (JobDescription)

```
<xsd:complexType name="JobDescription_Type">
    <xsd:sequence>
        <xsd:element ref="jsdl:JobIdentification"
          minOccurs="0"/>
        <xsd:element ref="jsdl:Application" minOccurs="0"/>
        <xsd:element ref="jsdl:Resources" minOccurs="0"/>
        <xsd:element ref="jsdl:DataStaging" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other"
      processContents="lax"/>
</xsd:complexType>
```

Comments

- Contains elements that describe relevant aspects of a job
- All of them are optional
- Can be extended at will with elements and attributes (from given schemas)

JSDL (JobIdentification)

```
<xsd:complexType name="JobIdentification_Type">
    <xsd:sequence>
        <xsd:element ref="jsdl:JobName" minOccurs="0"/>
        <xsd:element ref="jsdl:Description"
            minOccurs="0"/>
        <xsd:element ref="jsdl:JobAnnotation" minOccurs="0
            maxOccurs="unbounded"/>
        <xsd:element ref="jsdl:JobProject" minOccurs="0"
            maxOccurs="unbounded"/>
        <xsd:any namespace="##other" processContents="lax"
            minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    <xsd:anyAttribute namespace="##other"
            processContents="lax"/>
</xsd:complexType>
```

JSDL (Application)

```
<xsd:complexType name="Application_Type">
    <xsd:sequence>
        <xsd:element ref="jsdl:ApplicationName"
          minOccurs="0"/>
        <xsd:element ref="jsdl:ApplicationVersion"
          minOccurs="0"/>
        <xsd:element ref="jsdl:Description" minOccurs="0"/>
        <xsd:any namespace="##other" processContents="lax"
          minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other"
      processContents="lax"/>
</xsd:complexType>
```

Too simple?

There is a normative extension for POSIX apps

JSDL (Resources)

```
<xsd:complexType name="Resources_Type">
    <xsd:sequence>
        <xsd:element ref="jsdl:CandidateHosts" minOccurs="0"/>
        <xsd:element ref="jsdl:FileSystem" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element ref="jsdl:OperatingSystem" minOccurs="0"/>
        <xsd:element ref="jsdl:CPUArchitecture" minOccurs="0"/>
        <xsd:element ref="jsdl:IndividualCPUTime" minOccurs="0"/
          >
        <xsd:element ref="jsdl:TotalCPUTime" minOccurs="0"/>
        <xsd:element ref="jsdl:TotalCPUCount" minOccurs="0"/>
        <xsd:element ref="jsdl:IndividualNetworkBandwidth"
          minOccurs="0"/>


... etc. ...
```

Comments

- There is a way to explicitely indicate named hosts
- FileSystem element may be quite specific (mountpoint) or generic (type)
- No check is done as for reasonableness
- Network requirements may be specified

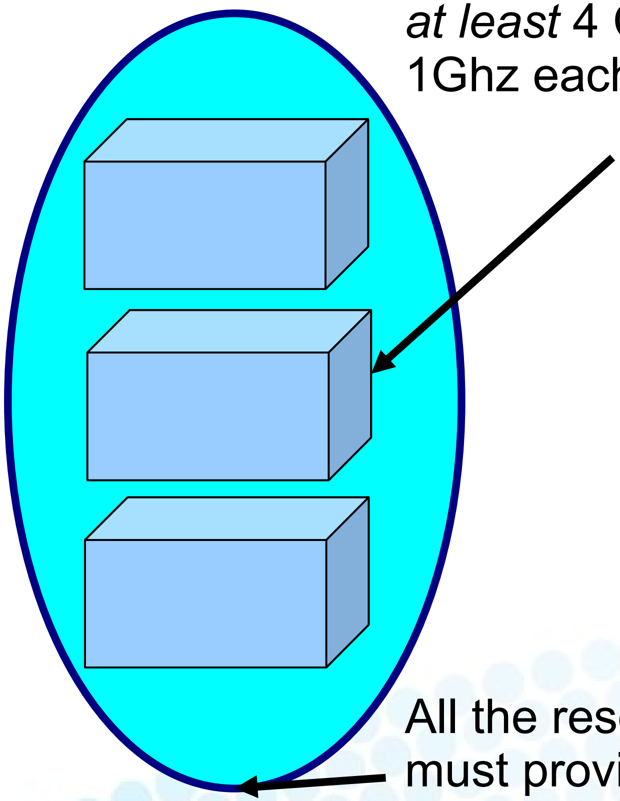
FileSystem element (example)

```
<jsdl:FileSystem name=”HOME”>
   <jsdl:Description>Ali’s home</jsdl:Description>
   <jsdl:MountPoint>/home/ali</jsdl:MountPoint>
   <jsdl:DiskSpace>
      <jsdl:LowerBoundedRange>
         1073741824.0
      </jsdl:LowerBoundedRange>
   </jsdl:DiskSpace>
   <jsdl:FileSystemType>normal</jsdl:FileSystemType>
</jsdl:FileSystem>
```

There is a way to explicitly indicate named hosts

Example (Resources)

```
<jsdl:Resources>
    <jsdl:IndividualCPUSpeed>
        <Exact> 1E9 </Exact>
    </jsdl:IndividualCPUSpeed>
    <jsdl:IndividualCPUCount>
        <jsdl:LowerBound>
            4.0
        </jsdl:LowerBound>
    </jsdl:IndividualCPUCount>
    <jsdl:TotalCPUCount>
        <jsdl:UpperBound>
            16.0
        </jsdl:UpperBound>
    </jsdl:TotalCPUCount>
</jsdl:Resources>
```



Each resource has *at least* 4 CPUs at 1Ghz each

All the resources must provide *at most* 16 CPUs

Inconsistent requirements

```
<jsdl:Resources>
    <jsdl:TotalResourceCount>
        <jsdl:UpperBound> 8.0 </jsdl:UpperBound>
    </jsdl:TotalResourceCount>
    <jsdl:IndividualCPUCount>
        <jsdl:Exact> 1.0 </jsdl:Exact>
    </jsdl:IndividualCPUCount>
    <jsdl:TotalCPUCount>
        <jsdl:Exact> 16.0 </jsdl:Exact>
    </jsdl:TotalCPUCount>
</jsdl:Resources>
```

- *At most* 8 resources...
- ...each one providing *exactly* 1 CPU...
- ...with a *total* CPU count of *exactly* 16

More about Resource description…

- Resources can be described using a different language (RRL)
- The GLUE experience:
  - started as an agreement in the frame of an International cooperation (DATATAG)
  - Later developed as an OGF standard (OGF-GLUE-wg)
  - More during the second part of the tutorial…

JSDL (DataStaging)

```
<xsd:complexType name="DataStaging_Type">
    <xsd:sequence>
        <xsd:element ref="jsdl:FileName"/>
        <xsd:element ref="jsdl:FilesystemName" minOccurs="0"/
          >
        <xsd:element ref="jsdl:CreationFlag"/>
        <xsd:element ref="jsdl:DeleteOnTermination"
          minOccurs="0"/>
        <xsd:element ref="jsdl:Source" minOccurs="0"/>
        <xsd:element ref="jsdl:Target" minOccurs="0"/>
    ... etc ...
```

Comments

- This is used to describe data files that have to moved
- Permissions are not considered (different OS – different permissions)
- Basic flags
- Unordered
- ...but expandable

Example (Stage in)

```
<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>
```

Example (from a filesystem)

```
<jsdl:FileSystem name=”HOME”>... </jsdl:FileSystem>
...
<jsdl:DataStaging>
   <jsdl:FileName> control.txt </jsdl:FileName>
   <jsdl:FileSystemName> HOME </jsdl:FileSystemName>
   <jsdl:Source>
      <jsdl:URI> http://site.it/~me/control.txt </jsdl:URI>
   </jsdl:Source>
   <jsdl:CreationFlag> overwrite </jsdl:CreationFlag>
   <jsdl:DeleteOnTermination>
      true
   </jsdl:DeleteOnTermination>
</jsdl:DataStaging>
```

Example (stage in/out)

```
<jsdl:DataStaging>
    <jsdl:FileName> state.txt </jsdl:FileName>
    <jsdl:Source>
        <jsdl:URI> http://node1/~me/state.txt </jsdl:URI>
    </jsdl:Source>
    <jsdl:Target>
        <jsdl:URI> http://node2/~me/state.txt </jsdl:URI>
    </jsdl:Target>
    ...
</jsdl:DataStaging>
```

Attribute extensions

```
<jsdl:DataStaging o:order="1">
    <jsdl:FileName> a.dat </jsdl:FileName>
    <jsdl:CreationFlag> overwrite </jsdl:CreationFlag>
    <jsdl:Source>
        <jsdl:URI> http://site.com/a-file </jsdl:URI>
    </jsdl:Source>
</jsdl:DataStaging>
<jsdl:DataStaging o:order="2">
    <jsdl:FileName> b.dat </jsdl:FileName>
    <jsdl:CreationFlag> overwrite </jsdl:CreationFlag>
    <jsdl:Source>
        <jsdl:URI> http://site.com/b-file</jsdl:URI>
    </jsdl:Source>
</jsdl:DataStaging>
```

Element extensions

- Similarly for elements

```
<jsdl:Resources>
    <jsdl:TotalCPUCount>
        <jsdl:Exact> 1.0 </jsdl:Exact>
    </jsdl:TotalCPUCount>
    <jsdl:TotalDiskSpace>
        <jsdl:LowerBoundedRange> 1E6 </jsdl:LowerBoundedRange>
    </jsdl:TotalDiskSpace>
    <res:Reservation
      xmlns:res="http://www.example.org/reservation">
        <res:Ticket>h933fsolenri900wnmd90mm34</res:Ticket>
    </res:Reservation>
</jsdl:Resources>
```

JSDL (references)

- Job Submission Description Language
- Most recent specification is Version 1.0
  - Version 1.1 being worked out
  - Version 1.0 available at http://www.gridforum.org/documents/GFD.56.pdf
- JSDL Working Group https://forge.gridforum.org/projects/jsdl-wg/

Normative extension: POSIX Application

- Defines a schema for applications executed on a POSIX compliant system.
  - It contains Executable, Argument, Input, Output, Error, WorkingDirectory, Environment, various POSIX limits elements as well as User and Group names.
  - If it is present as a sub-element of the JSDL Application element it MUST appear only once.

The Schema (partial)

```
<xsd:complexType name="POSIXApplication_Type">
   <xsd:sequence>
      <xsd:element ref="jsdl-posix:Executable" minOccurs="0"/>
      <xsd:element ref="jsdl-posix:Argument" minOccurs="0”
        maxOccurs="unbounded"/>
      <xsd:element ref="jsdl-posix:Input" minOccurs="0"/>
      <xsd:element ref="jsdl-posix:Output" minOccurs="0"/>
      <xsd:element ref="jsdl-posix:WorkingDirectory"
        minOccurs="0"/>
      <xsd:element ref="jsdl-posix:Environment” minOccurs="0"
        maxOccurs="unbounded"/>
      <xsd:element ref="jsdl-posix:CPUTimeLimit" minOccurs="0"/
        >
      <xsd:element ref="jsdl-posix:UserName" minOccurs="0"/>
      ... and more ...
   </xsd:sequence>
```

JSDL Example / 1

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns="http://www.example.org/"
        xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
        xmlns:jsdl-posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <jsdl:JobDescription>
        <jsdl:JobIdentification>
            ...
        </jsdl:JobIdentification>
        <jsdl:Application>
            <jsdl:ApplicationName>gnuplot</jsdl:ApplicationName>
            <jsdl-posix:POSIXApplication>
                <jsdl-posix:Executable>
                    /usr/local/bin/gnuplot
                </jsdl-posix:Executable>
                <jsdl-posix:Argument>control.txt</jsdl-posix:Argument>
                <jsdl-posix:Input>input.dat</jsdl-posix:Input>
                <jsdl-posix:Output>output1.png</jsdl-posix:Output>
            </jsdl-posix:POSIXApplication>
        </jsdl:Application>
        <jsdl:Resources>
            <jsdl:IndividualPhysicalMemory>
                <jsdl:LowerBoundedRange>2097152.0</jsdl:LowerBoundedRange>
            </jsdl:IndividualPhysicalMemory>
            <jsdl:TotalCPUCount>
                <jsdl:Exact>1.0</jsdl:Exact>
            </jsdl:TotalCPUCount>
        </jsdl:Resources>                            to be continued...
```

JSDL Example / 2

```
        <jsdl:DataStaging>
            <jsdl:FileName>control.txt</jsdl:FileName>
            <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
            <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
            <jsdl:Source>
                <jsdl:URI>http://foo.bar.com/~me/control.txt</jsdl:URI>
            </jsdl:Source>
        </jsdl:DataStaging>
        <jsdl:DataStaging>
            <jsdl:FileName>input.dat</jsdl:FileName>
            <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
            <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
            <jsdl:Source>
                <jsdl:URI>http://foo.bar.com/~me/input.dat</jsdl:URI>
            </jsdl:Source>
        </jsdl:DataStaging>
        <jsdl:DataStaging>
            <jsdl:FileName>output1.png</jsdl:FileName>
            <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
            <jsdl:DeleteOnTermination>true</jsdl:DeleteOnTermination>
            <jsdl:Target>
                <jsdl:URI>rsync://spoolmachine/userdir</jsdl:URI>
            </jsdl:Target>
        </jsdl:DataStaging>
    </jsdl:JobDescription>
</jsdl:JobDefinition>
```

JSDL Extensions

- HPC Basic Profile (HPC-BP) v1.0 (GFD.114)
  - http://www.ogf.org/documents/GFD.114.pdf
- JSDL HPC Profile Application Extension, Version 1.0 (GDF.111)
  - http://www.ogf.org/documents/GFD.111.pdf
- JSDL SPMD Application Extension, Version 1.0 (GFD.115)
  - http://www.ogf.org/documents/GFD.115.pdf
- HPC File Staging Profile (currently in Public Comment)
  - http://www.ogf.org/Public_Comment_Docs/Documents/2008-02/HPC%20File%20Staging%20Profile.pdf

HPC Basic Profile

- Describes how a particular set of specifications are composed in order to solve a basic use case of High Performance Computing (HPC) systems.
- The “use case” is defined as follows:

“a high-throughput compute cluster that is managed by a batch job scheduler and that is used only from within an organization”

(from OGF document GFD-I-100)

HPC Basic Profile

- The Profile establishes a link between:
  - JSDL v1.0, Job Submission and
  - BES, Job Runtime model
- It describes the basic capabilities needed to define a given system as compliant
- It is a fundamental step towards interoperability
- Requires an extension to the basic JSDL

HPC Profile Application Extension

- Defines an extension to JSDL 1.0 for describing HPC applications in a “basic use case” perspective
- Essentially a subset of the JSDL POSIXApplication
- Removes some of the features that present barriers to interoperability (with Windows)
- It is needed to conform to the Basic Profile

HPC Profile Application Extension

- The “pseudo-schema”:

```
<HPCProfileApplication name="xsd:NCName"?>
    <Executable ... />?
    <Argument ... />*
    <Input ... />?
    <Output ... />?
    <Error ... />?
    <WorkingDirectory ... />?
    <Environment ... />*
    <UserName ... />?
</HPCProfileApplication>
```

Note: no “limits”, no “groupName” w.r.t. POSIX

SPMD Application Extension

- Describes an SPMD (single program multiple data) parallel application and its requirements
- Based on the (normative) POSIXApplication extension
- A parallel application may execute on a number of different hosts
- The WorkingDirectory may be on a shared filesystem.

SPMD Application Extension

- The “pseudo-schema:

```
<HPCProfileApplication name="xsd:NCName"?>
      <Executable ... />?
      <Argument ... />*
      <Input ... />?
      <Output ... />?
      <Error ... />?
      <WorkingDirectory ... />?
      <Environment ... />*
      <UserName ... />?
      <NumberOfProcesses .../>?
      <ProcessesPerHost ... />?
      <ThreadsPerProcess ... />?
      <SPMDVariation .../>
</HPCProfileApplication>
```

SPMD (notes)

- SPMDApplication elements describe the parallel application; they do not specify resource requirements
- Simply put: cannot infer resources from application parameters
- SPMDVariation indicates the flavor of SPMD support (MPI etc.) to use

Full Copyright Notice