

An Extension Proposal of the JSDL1.0 for Parallel Applications

I. Rodero, F. Guim, J. Corbalan, J. Labarta

{irodero, francesc.guim, julita.corbalan, jesus.labarta}@bsc.es

Barcelona Supercomputing Center
(<http://www.bsc.es>)

8th May 2006



- Extension of the JSDL for Parallel Jobs (Proposal)
 - Current Specification
 - Extension Proposal
 - Document Example
- Barcelona Supercomputing Centre (BSC) Use Case
 - Motivation: eNANOS Project
 - eNANOS execution Framework
 - Examples Description
 - 2 Levels of parallelism (MPI+OpenMP)
 - 1 Level of parallelism (MPI)
 - 2 Levels of parallelism (OpenMP is malleable)



Extension of the JSDL 1.0 for Parallel Jobs (Proposal)



- With the current specification (JSDL v1.0 + POSIX Extension) it is possible to specify some parallelism details for a job:

- TotalCPUCount

-

“This element is a range value specifying the **total number of CPUs required for**

This element is a range value specifying the **number of CPUs for each of the resources** to be allocated to the job submission. If this is not present then it is not defined and the consuming system **MAY** choose any value.

OK

app

:-)

Problems:

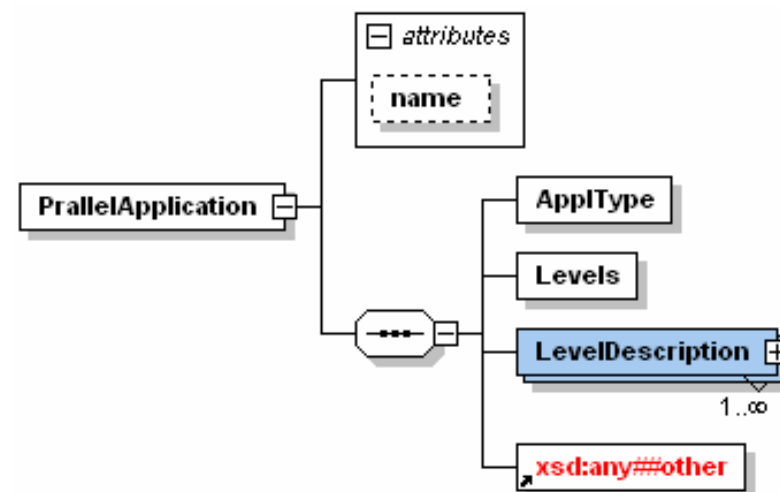
- It is not allowed to specify the programming model
- No multilevel are taken into account (MPI oriented)
- No information regarding the application behavior is included



- Our intention is to specify an extension for parallel applications:
 - General
 - Suitable for all the programming models
 - Suitable for the future approaches
- In this proposal we have chosen the extension by new XML elements.
Should we think about an extension by attributes?
- The namespace prefix used for this schema is “**jsdl-par**”. Since this is only a proposal the normative namespace is not given.

- ParallelApplication Element

```
<ParallelApplication name="xsd:NCName" ?>
  <ApplType... />
  <Levels... />
  <LevelDescription... />+
  <xsd:any##other>*
</ParallelApplication>
```



This element describes the **parallelism details** of an application. It contains the description of the **type of the application**, the **number of levels** of the application, for each level the **topology**. If it is present as a sub-element of the JSDL Application element it **MUST** appear only once. It **MUST NOT** appear for sequential applications.



- ApplType Element

```
<ApplType>  
  jsdl-par:ParallelApplTypeEnumeration  
</ApplType>
```

This element is an
regarding the **prog**
should be interpreted
to use the most su
POE, OpenMP run

Name	Definition
mpi	MPI, message passing
openmp	OpenMP, shared memory model
mpi_openmp	Hybrid programming model
pvm	Paralle Virtual Machine
threads	Threaded application
caf	Co-Array Fortran
upc	Unified Parallel C
hpf	High Performance Fortran
other	Other programming models



- Levels Element

```
<Levels> xsd:positiveInteger </Levels>
```

This element is a positive integer that specifies the **number of parallelism levels** of the application. For instance, a MPI+OpenMP application has 2 levels of parallelism.



- LevelDescription Element

```
<LevelDescription level="xsd:positiveInteger" malleable="xsd:boolean"?>  
  <Parallelism... />?  
  <Topology... />?  
  <xsd:a... />?  
</LevelDescription>
```

This element describes a parallel application. It has one or more child elements. Attributes:

modalable applications can be defined implicitly just by the “unrestricted” topology and disabling the malleable attribute. Example:

- level**—integer value that the application uses to describe the number of processes.
- malleable**—boolean attribute that indicates if the application is modalable.

```
<ParallelApplication>  
  <ApplType>mpi</ApplType>  
  <Levels>1</Levels>  
  <LevelDescription level="1">  
    <Parallelism>  
      <jsd1:LowerBoundedRange>4.0</jsdl:LowerBoundedRange>  
    </Parallelism>  
    <Topology>unrestricted</Topology>  
  </LevelDescription>  
</ParallelApplication>
```



- Parallelism Element

```
<Parallelism>  
  jsdl:RangeValue_Type  
</Parallelism>?
```

This is a **range value** that describes the **number of processes or threads** that are required by the parallel application. The topology element only has sense if the value of this element is not an exact number. In this case it can be supposed that the application is moldable but the malleability is explicitly indicated by the malleable attribute of the LevelDescription element.



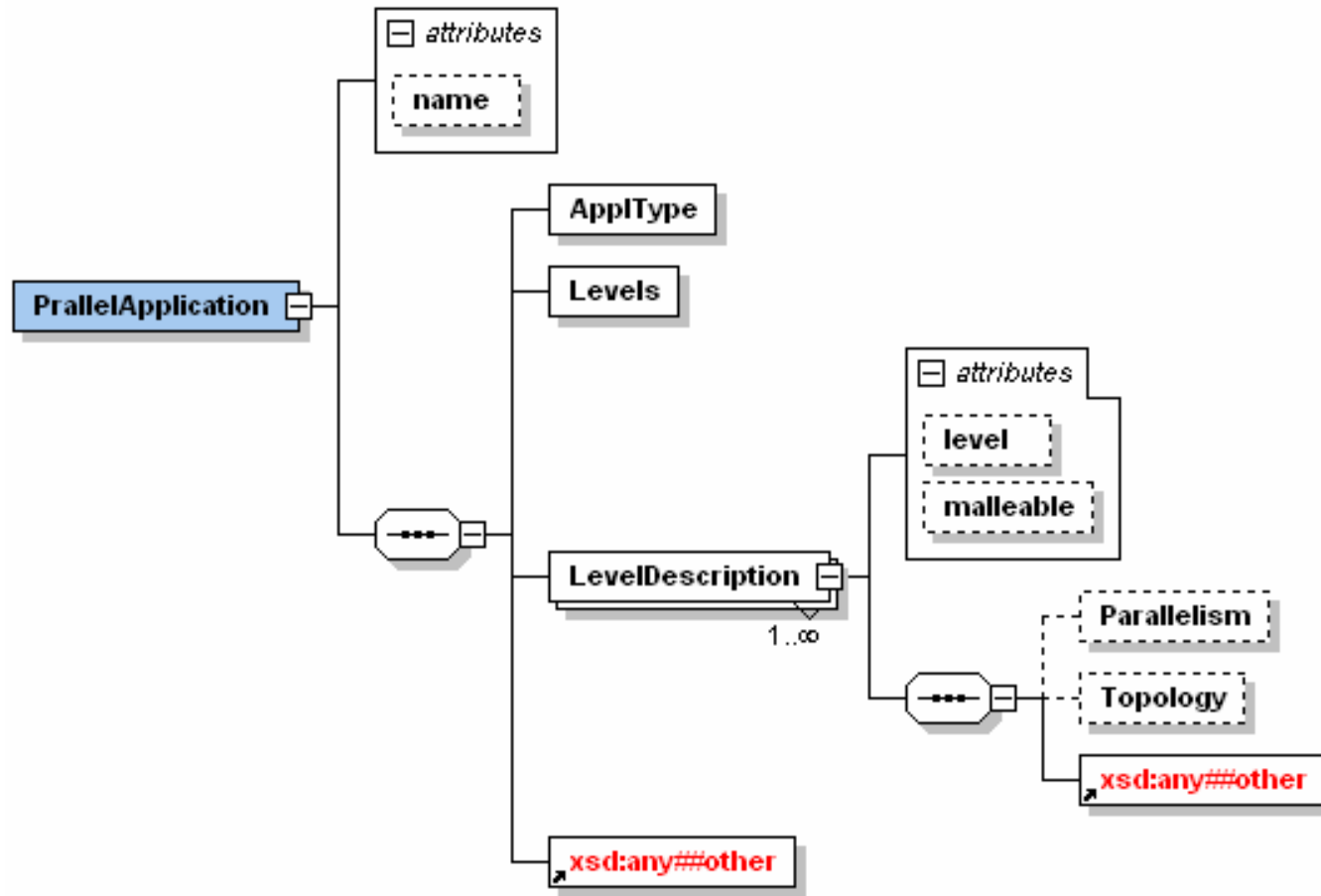
- Topology Element

```
<Topology>  
  jsdl-par:ParallelApplTopologyEnumeration  
</Topology>
```

This element
application
the number

Name	Definition
unrestricted	The LRMS can select any number of processes/threads for the application between the range specified in level description.
power2	The number of processes/threads for the application MUST be power of two (e.g. 3 power 2 = 9).
explicit	The number of processes/threads is explicitly specified.

Complete Schema





```
<ParallelApplication>
  <ApplType>mpi_ope
  <Levels>2</Levels>
  <LevelDescription level="1" maxCpus="16" maxOpenMPThreads="16" topology="unrestricted">
    <Parallelism>
      <jsdl:LowerBoundedRange>4.0</jsdl:LowerBoundedRange>
    </Parallelism>
  </LevelDescription>
  <LevelDescription level="2" maxCpus="16" maxOpenMPThreads="16" topology="unrestricted">
    <jsdl:LowerBoundedRange>4.0</jsdl:LowerBoundedRange>
    <jsdl:UpperBoundedRange>16.0</jsdl:UpperBoundedRange>
  </LevelDescription>
</ParallelApplication>
```

The OpenMP threads are power of 2
(if no enough CPUs are available, it should
spawn only 8, 4, 2 or 1 threads)

MPI+OpenMP application

At least 4 MPI processes
(not limited)

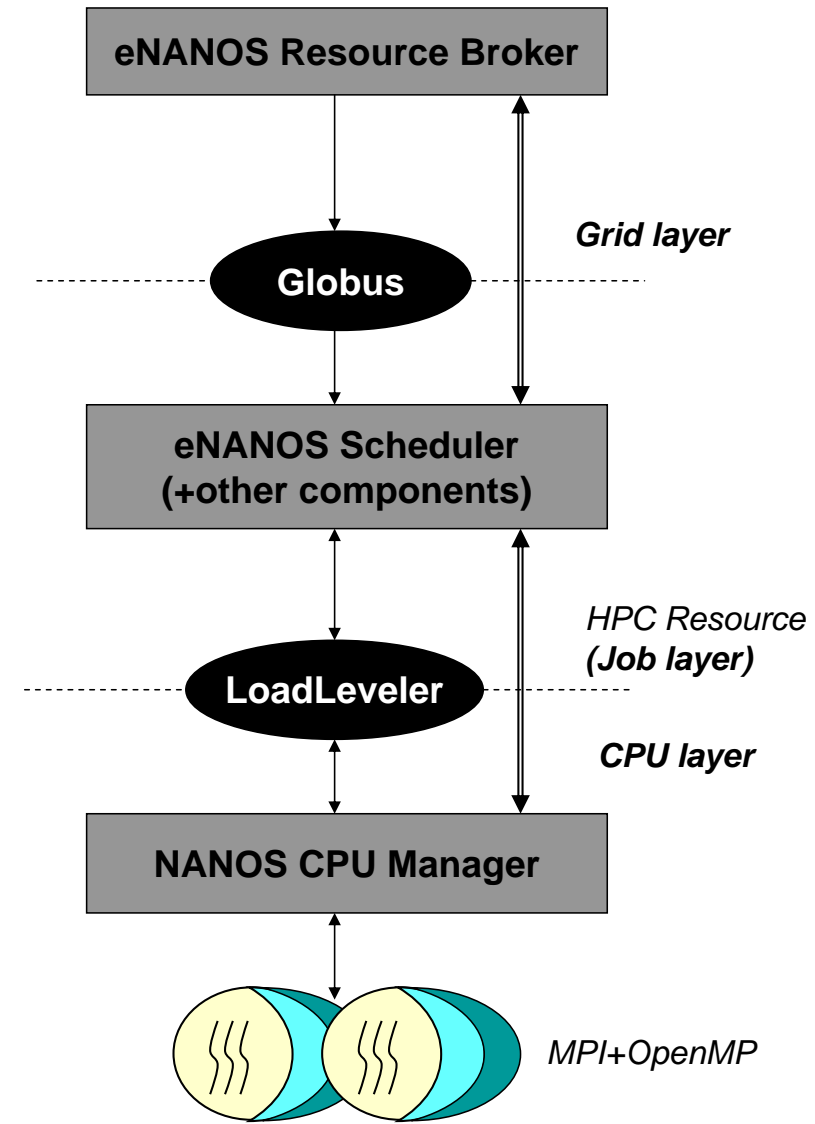
OpenMP threads can be modified
dinamically (malleable)



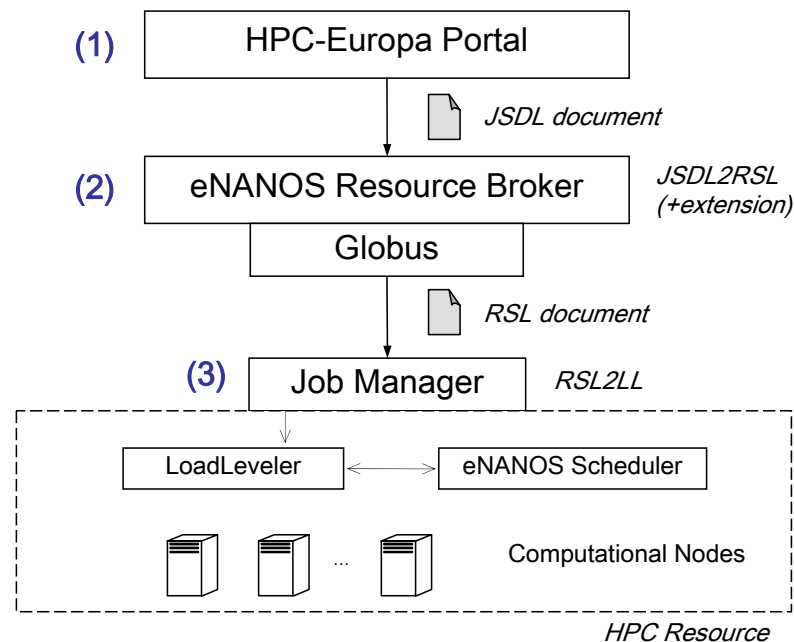
Barcelona Supercomputing Center (BSC) Use Case

Motivation: eNANOS Project

- **HPC**-Applications
- Hybrid **MPI+OpenMP** programming model (including pure MPI and OpenMP)
- HPC-Grids. Composed of HPC resources
- Clusters of SMP's & CC-NUMA architectures with a medium to high number of processors
- **Heterogeneous** resources
- **Efficient** execution on the Local Resources
- **Coordination** between the Grid and Local Levels
- Experience obtained from the **HPC-Europa** project



1. The Grid Broker receives a JSDL 1.0 document from the HPC-Europa Portal.
2. The JSDL document is converted to RSL (2) because the Grid broker is built on top of the Globus infrastructure. Not all the information expressed in a JSDL document can be covered in an RSL document, so we have to use some environment variables as a simple mechanism to solve this problem
3. In the local resource, the appropriate job manager (3) transforms the RSL document to a LoadLeveler script.





In the following examples there are shown two files:

- JSDL documents. Since the current implementation of our execution environment does not support all the functionality proposed in the extension for parallel jobs, some of the examples are obtained from real jobs (NAS BT and Simple MPI) and some others are only described as a new use case or proposal (CPMD).
- LoadLeveler scripts obtained from the previous JSDL documents. These scripts are generated automatically by the LoadLeveler JobManager called by the Globus gatekeeper.
 - LoadLeveler system is not able to manage multilevel parallel applications.
 - Some of the semantic provided by the extended JSDL for parallel jobs is lost.
 - Usually “total_tasks” field is used for specifying the total number of MPI processes
 - There is no mechanism to specify the number of OpenMP threads in current LL
 - The eNANOS Scheduler manages the second level of parallelism and the details expressed in the JSDL.
 - Currently we use environment variables as mechanism to specify the parallelism details (provisional). eNANOS scheduler interprets the environment variables.

Example 1: NAS BT-MZ (multilevel)



- The JSDL document describes a job composed of a NAS BT-MZ benchmark (MPI+OpenMP) and class A. This job has 2 levels of parallelism.
 - It is requested 2 MPI processes and 4 OpenMP threads per process, and the standard error and output are redirected to a specific machine (pcmas).

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/10/jsdl">
  <JobDescription>
    <JobIdentification>
      <Description>Execution of a NAS MultiZone class A</Description>
      <JobProject>BSC_Test</JobProject>
    </JobIdentification>
    <Application>
      <ns1:POSIXApplication xmlns:ns1="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix">
        <ns1:Executable filesystemName="__user1_uni_upc_ac_irodero_enanos_benchmarks_">ExecNas</ns1:Executable>
        <ns1:Argument>bt-mz.A</ns1:Argument>
        <ns1:Argument>2</ns1:Argument>
        <ns1:Argument>4</ns1:Argument>
        <ns1:Output>BT.A.OUT</ns1:Output>
        <ns1:Error>BT.A.ERR</ns1:Error>
        <ns1:Environment name="OMP_SCHEDULE">static</ns1:Environment>
        <ns1:Environment name="THREAD_BOUND">1</ns1:Environment>
      </ns1:POSIXApplication>
    </Application>
    <Resources>
      <CandidateHosts>
        <HostName>kadesh8.cepba.upc.edu</HostName>
      </CandidateHosts>
      <FileSystem name="__user1_uni_upc_ac_irodero_enanos_benchmarks_">
        <MountPoint>/user1/uni/upc/ac/irodero/enanos/benchmarks</MountPoint>
      </FileSystem>
    </Resources>
  </JobDescription>
</JobDefinition>
```

(...)

Example 1: NAS BT-MZ (multilevel)



(...)

```
<DataStaging>
  <FileName>BT.A.ERR</FileName>
  <CreationFlag>append</CreationFlag>
  <DeleteOnTermination>>false</DeleteOnTermination>
  <Target>
    <URI>gsiftp://pcmas.ac.upc.es/home/irodero/tests/BT.A.ERR</URI>
  </Target>
</DataStaging>
<DataStaging>
  <FileName>BT.A.OUT</FileName>
  <CreationFlag>append</CreationFlag>
  <DeleteOnTermination>>false</DeleteOnTermination>
  <Target>
    <URI>gsiftp://pcmas.ac.upc.es/home/irodero/tests/BT.A.OUT</URI>
  </Target>
</DataStaging>
</JobDescription>
<ns2:ParallelApplication xmlns:ns2="http://schemas.ggf.org/jsdl/2006/03/jsdl-par">
  <ns2:ApplType>mpi_openmp</ns2:ApplType>
  <ns2:Levels>2</ns2:Levels>
  <ns2:LevelDescription level="1">
    <ns2:Parallelism>
      <jsdl:exact>2.0</jsdl:exact>
    </ns2:Parallelism>
  </ns2:LevelDescription>
  <ns2:LevelDescription level="2">
    <ns2:Parallelism>
      <jsdl:exact>4.0</jsdl:exact>
    </ns2:Parallelism>
  </ns2:LevelDescription>
</ns2:ParallelApplication>
</JobDefinition>
```

Example 1: NAS BT-MZ (multilevel)



- LoadLeveler Script

- We have reused existent ways to specify parallelism (i.e. OMP_NUM_THREADS)

```
#!/bin/sh
# Job command file created by GRAM/JobManager/loadleveler.pm
# @ job_type      = parallel
# @ initialdir    = /user1/uni/upc/ac/irodero
# @ input         = /dev/null
# @ output        = /user1/uni/upc/ac/irodero/.globus/.gass_cache/local/md5/37/c304c3b0417c6d05ad764f2f3db3e0/md5/2e/1c
#                 dbef76729af40306461c2447b1e4c/data
# @ error         = /user1/uni/upc/ac/irodero/.globus/.gass_cache/local/md5/37/c304c3b0417c6d05ad764f2f3db3e0/md5/e0/db2941
#                 aeb23a18167 bd2c821a2d7ba/data
# @ account_no    = BSC_Test
# @ class         = short
# @ restart       = yes
# @ requirements  = (LL_Version >= "2.0") && (Adapter == "ethernet")
# @ total_tasks  = 2 -----> MPI processes
# @ node         = 1
# @ environment   = COPY_ALL;¥
# X509_USER_PROXY=/user1/uni/upc/ac/irodero/.globus/.gass_cache/local/md5/37/c304c3b0417c6d05ad764f2f3db3e0/md5/b7/c5e
# a252c9f577a52da2db277d3058b/data; ¥
# GLOBUS_LOCATION=/aplic/GLOBUS/2.4; ¥
# GLOBUS_GRAM_JOB_CONTACT=https://kadesh8.cepba.upc.edu:37895/33752/1143561915/; ¥
# GLOBUS_GRAM_MYJOB_CONTACT=URLx-nexus://kadesh8.cepba.upc.edu:37896/; ¥
# HOME=/user1/uni/upc/ac/irodero; ¥
# LOGNAME=irodero; ¥
# GRID_ID_ENV=1@1143561909633; ¥
# OMP_SCHEDULE=static; ¥
# OMP_NUM_THREADS=4;¥ -----> OpenMP threads
# THREAD_BOUND=1; ¥
# PAR_MALLEABLE=false -----> Is not malleable
# @ queue
#
/user1/uni/upc/ac/irodero/enanos/benchmarks/ExecNas bt-mz.A 2 4
#
# End of job command file.
```

MPI processes

OpenMP threads

Is not malleable

Example 2: Simple MPI (Solver)



- The JSDL document describes a job composed of a simple MPI application. In particular is a typical solver that should be executed with 16 MPI processes.

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns="http://schemas.ggf.org/jsdl/2005/10/jsdl">
  <JobDescription>
    <JobIdentification>
      <Description>Execution of a simple MPI-based Solver</Description>
      <JobProject>BSC_Test</JobProject>
    </JobIdentification>
    <Application>
      <ns1:POSIXApplication xmlns:ns1="http://schemas.ggf.org/jsdl/2005/06/jsdl-posix">
        <ns1:Executable filesystemName="__user1_uni_upc_ac_irodero_enanos__solver_">Solver</ns1:Executable>
        <ns1:Output>/user1/uni/upc/ac/irodero/enanos/solver/solver.out
        </ns1:Output>
        <ns1:Error>/user1/uni/upc/ac/irodero/enanos/solver/solver.out</ns1:Error>
      </ns1:POSIXApplication>
    </Application>
    <Resources>
      <CandidateHosts>
        <HostName>kadesh8.cepba.upc.edu</HostName>
      </CandidateHosts>
      <FileSystem name="__user1_uni_upc_ac_irodero_enanos_solver_">
        <MountPoint>/user1/uni/upc/ac/irodero/enanos/benchmarks/solver</MountPoint>
      </FileSystem>
    </Resources>
  </JobDescription>
  <ns2:ParallelApplication xmlns:ns2="http://schemas.ggf.org/jsdl/2006/03/jsdl-par">
    <ns2:ApplType>mpi</ns2:ApplType>
    <ns2:Levels>1</ns2:Levels>
    <ns2:LevelDescription level="1">
      <ns2:Parallelism>
        <jsdl:exact>16.0</jsdl:exact>
      </ns2:Parallelism>
    </ns2:LevelDescription>
  </ns2:ParallelApplication>
</JobDefinition>
```

Example 2: Simple MPI (Solver)



- LoadLeveler Script
 - This is a case in which the LoadLeveler System can manage the job by itself because it only has one level of parallelism and it is a rigid application. The added information is only for our execution framework.

```
#!/bin/sh
# Job command file created by GRAM/JobManager/loadleveler.pm
# @ job_type      = parallel
# @ initialdir    = /user1/uni/upc/ac/irodero/enanos/solver
# @ input         = /dev/null
# @ output        = /user1/uni/upc/ac/irodero/enanos/solver/solver.out
# @ error         = /user1/uni/upc/ac/irodero/enanos/solver/solver.err
# @ class         = short
# @ restart       = yes
# @ total_tasks   = 16 -----> MPI processes
# @ node          = 1
# @ environment   = COPY_ALL;¥
# X509_USER_PROXY=/user1/uni/upc/ac/irodero/.globus/.gass_cache/local/md5/37/c304c3b0417c6d05ad764f2f3db3e0/md5/b7/
c5ea252c9f577a52da2db277d3058b/data; ¥
# GLOBUS_LOCATION=/aplic/GLOBUS/2.4; ¥
# GLOBUS_GRAM_JOB_CONTACT=https://kadesh8.cepba.upc.edu:37895/33752/1143561915/; ¥
# GLOBUS_GRAM_MYJOB_CONTACT=URLx-nexus://kadesh8.cepba.upc.edu:37896/; ¥
# HOME=/user1/uni/upc/ac/irodero; ¥
# LOGNAME=irodero; ¥
# GRID_ID_ENV=7@1143561974629; ¥
# @ queue
#
/user1/uni/upc/ac/irodero/enanos/solver/Solver
#
# End of job command file.
```

MPI processes

Example 3: CPMD (malleable)



- The JSDL document describes a multilevel parallel job composed of a CPMD application (MPI+OpenMP).
- In this case the job requires 2 MPI processes but the second level of parallelism (OpenMP threads) is expected to be power of 2 with a maximum of 16 threads. The second level of parallelism is malleable as well.

```
<?xml version="1.0" encoding="UTF-8"?>
<JobDefinition xmlns="http://schemas.ggf.org/jSDL/2005/10/jSDL">
  <JobDescription>
    <JobIdentification>
      <Description>Execution of a CPMD application</Description>
      <JobProject>BSC_Test</JobProject>
    </JobIdentification>
    <Application>
      <ns1:POSIXApplication xmlns:ns1="http://schemas.ggf.org/jSDL/2005/06/jSDL-posix">
        <ns1:Executable>/scratch_tmp/irodero/CPMD-3.9.1/cpmd.x</ns1:Executable>
        <ns1:Argument>/scratch_tmp/irodero/CPMD-3.9.1/inputs/small.inp </ns1:Argument>
        <ns1:Output>cpmd.4.pwr4.out</ns1:Output>
        <ns1:Error>cpmd.4.pwr4.err</ns1:Error>
        <ns1:Environment name="PP_LIBRARY_PATH">/scratch_tmp/irodero/CPMD-3.9.1/PP_LIB</ns1:Environment>
        <ns1:Environment name="OMP_SCHEDULE">static</ns1:Environment>
      </ns1:POSIXApplication>
    </Application>
    <Resources>
      <CandidateHosts>
        <HostName>kadesh8.cepba.upc.edu</HostName>
      </CandidateHosts>
    </Resources>
  </JobDescription>
  (...)
</JobDefinition>
```

Example 3: CPMD (malleable)



(...)

```
<ns2:ParallelApplication xmlns:ns2="http://schemas.ggf.org/jsdl/2006/03/jsdl-par">
  <ns2:ApplType>mpi_openmp</ns2:ApplType>
  <ns2:Levels>2</ns2:Levels>
  <ns2:LevelDescription level="1">
    <ns2:Parallelism>
      <jsdl:exact>2.0</jsdl:exact>
    </ns2:Parallelism>
  </ns2:LevelDescription>
  <ns2:LevelDescription level="2" malleable="true">
    <ns2:Parallelism>
      <jsdl:UpperBoundedRange>16.0</jsdl:UpperBoundedRange>
    </ns2:Parallelism>
    <ns2:Topology>power2</ns2:Topology>
  </ns2:LevelDescription>
</ns2:ParallelApplication>
</JobDefinition>
```

Example 3: CPMD (malleable)



- LoadLeveler Script
 - It is a multilevel application
 - The OpenMP level is malleable (PAR_MALLEABLE=true)

```
#!/bin/sh
# Job command file created by GRAM/JobManager/loadleveler.pm
# @ job_type          = parallel
# @ initialdir        = /scratch_tmp/irodero/cpmd
# @ input              = /dev/null
# @ output             = cpmd.4.pwr4.out
# @ error              = cpmd.4.pwr4.err
# @ class              = short
# @ restart            = yes
# @ total_tasks        = 2 -----> MPI processes
# @ node                = 1
# @ environment        = COPY_ALL;¥
#   MP_EUILIB=ip;¥
#   MP_EUIDEVICE=en0;¥
#   PP_LIBRARY_PATH=/scratch_tmp/irodero/CPMD-3.9.1/PP_LIB;¥
#   GRID_ID_ENV=24@1143531900624;¥
#   OMP_NUM_THREADS=16;¥ -----> MPI processes
#   PAR_TOPOLOGY=power2;¥
#   PAR_MALLEABLE=true -----> OpenMP is malleable
# @ queue
#
/scratch_tmp/irodero/CPMD-3.9.1/cpmd.x /scratch_tmp/irodero/CPMD-3.9.1/inputs/small.inp
#
# End of job command file.
```