Authors:
Ivan Rodero, BSC
Francesc Guim, BSC
Julita Corbalán, UPC
Jesús Labarta, UPC

29 March 2006

Examples of JSDL 1.0 Documents for Parallel Jobs: BSC Use Case.

Abstract

This document shows some examples of JSDL documents with an extension for parallel jobs used at the Barcelona Supercomputing Center (BSC). These JSDL documents examples have been obtained from a web portal developed in the HPC-Europa JRA2 project framework. As a use case, it is also included a short description of the eNANOS execution environment and the LoadLeveler scripts obtained from the JSDL documents which are executed on an IBM SP2 cluster at BSC.

Contents

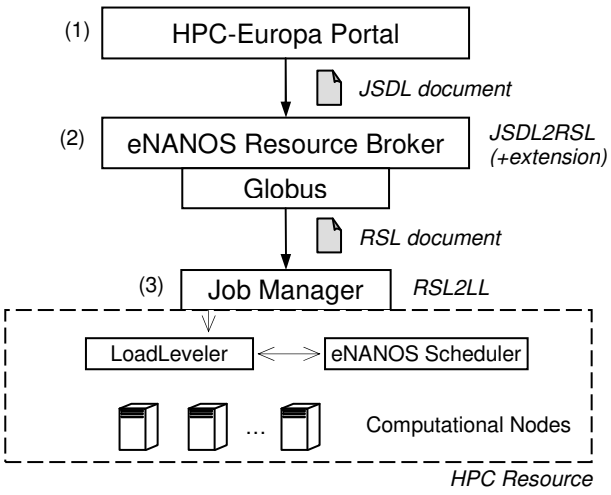
1 Introduction to the eNANOS execution environment at BSC

In our current execution environment, the Grid Resource Broker (eNANOS Broker) receives a JSDL document from the HPC-Europa Portal (1). These are JSDL 1.0 documents with an extension for parallel jobs and look like the examples shown in section 2.1.

Afterwards, 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.

Finally, in the local resource the appropriate job manager (3) transforms the RSL document to a LoadLeveler script. Some examples of LoadLeveler scripts are also shown in section 2.2.

A simple schema of the eNANOS environment is shown in following figure.



2 Examples of description of parallel jobs at BSC

In the following examples we present some JSDL documents with our extension for parallel jobs. We also present the LoadLeveler scripts obtained in the BSC resources when a job is submitted to our execution environment.

2.1 JSDL Examples

Some examples of JSDL documents are shown in this section. 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).

2.1.1 NAS BT (MultiZone)

In this example, the JSDL document describes a job composed of a NAS BT benchmark, MultiZone (MPI+OpenMP) and class A. 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>
        <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>
```

2.1.2 Simple MPI (Solver)

In this example the JSDL document describes a job composed of a simple MPI application. In particular 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>
```

2.1.3 CPMD

In this example the JSDL document describes a multilevel parallel job composed of a CPMD application (MPI+OpenMP). In this case the job requires at least 2 MPI processes with a configuration power of 2 but the second level of parallelism (OpenMP threads) is malleable and MUST be executed with a maximum of 16 threads.

```
<?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>
    <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:LowerBoundedRange>2.0</jsdl:LowerBoundedRange>
            </ns2:Parallelism>
            <ns2:Topology>power2</ns2:Topology>
        </ns2:LevelDescription>
        <ns2:LevelDescription level=”2” malleable=”true”>
            <ns2:Parallelism>
                <jsdl:UpperBoundedRange>16.0</jsdl:UpperBoundedRange>
            </ns2:Parallelism>
        </ns2:LevelDescription>
    </ns2:ParallelApplication>
</JobDefinition>
```

2.2 LoadLeveler Script Examples

In this section the LoadLeveler scripts obtained from the previous JSDL documents are shown. These scripts are generated automatically by the LoadLeveler Job Manager called by the Globus gatekeeper.

Since the 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 the value of the "total_tasks" field is used for specifying the total number of MPI processes but there is no mechanism to specify the number of OpenMP threads.

Therefore, the eNANOS Scheduler manages the second level of parallelism and the details expressed in the JSDL. Provisionally, we use environment variables as mechanism to specify the parallelism details. The eNANOS scheduler interprets the environment variables to assign the nodes or number of CPUs because it has all the required information.

2.2.1 NAS BT (MultiZone)

In the following script we indicate some important parameters with environment variables. Some of them are the number of OpenMP threads (OMP_NUM_THREADS), the ID of the job assigned in the Grid layer, or the malleability characteristic. It is also significant that the name of the standard output and error files are a cache file assigned by the Globus middleware.

Since the LoadLeveler system has no support for multilevel applications, it would take into account only 2 MPI processes ignoring the OpenMP threads. The eNANOS scheduler is responsible of managing the second level of parallelism.

```
#! /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/c304c3b0417c6d05ad764f2f3db3e
0/md5/2e/1cdbe76729af40306461c2447b1e4c/data
# @ error          =
/user1/uni/upc/ac/irodero/.globus/.gass_cache/local/md5/37/c304c3b0417c6d05ad764f2f3db3e
0/md5/e0/db2941aeb23a18167bd2c821a2d7ba/data
# @ account_no     = BSC_Test
# @ class          = short
# @ restart        = yes
# @ requirements   = (LL_Version >= "2.0") && (Adapter == "ethernet")
# @ total_tasks    = 2
# @ node           = 1
# @ environment = COPY_ALL;\
#
X509_USER_PROXY=/user1/uni/upc/ac/irodero/.globus/.gass_cache/local/md5/37/c304c3b0417c6
d05ad764f2f3db3e0/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=1@1143561909633; \
#   OMP_SCHEDULE=static; \
#   THREAD_BOUND=1; \
#   OMP_NUM_THREADS=4; \
#   PAR_MALLEABLE=false
# @ queue
#
/user1/uni/upc/ac/irodero/enanos/benchmarks/ExecNas bt-mz.A 2 4
#
# End of job command file.
```

2.2.2 Simple MPI (Solver)

In this case the semantic provided by the LoadLeveler system is enough for a simple MPI execution because only requires the number of MPI processes.

```
#! /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
# @ node           = 1
# @ environment = COPY_ALL;\
#
X509_USER_PROXY=/user1/uni/upc/ac/irodero/.globus/.gass_cache/local/md5/37/c304c3b0417c6
d05ad764f2f3db3e0/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.
```

2.2.3 CPMD

Since the JSDL document indicates that the number of MPI processes has to be power of 2, the eNANOS scheduler is in charge of the selection of MPI processes. The OMP_NUM_THREADS and PAR_TOPOLOGY variables indicate the maximum number of threads and the topology of the application respectively. These environment variables should be interpreted by the local scheduler to spawn the appropriate number of processes and threads (it depends of the system status).

```
#! /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
# @ node           = 1
# @ environment = COPY_ALL;\
#   MP_EUILIB=ip;\
#   MP_EUIDEVICE=en0;\
#   PP_LIBRARY_PATH=/scratch_tmp/irodero/CPMD-3.9.1/PP_LIB;\
#   OMP_NUM_THREADS=16;\
#   PAR_TOPOLOGY=power2
#@ 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.
```