EGEE-II

MPI WORKING GROUP

~~FIRST~~ REPORT

Document identifier:	EGEE-II-MPI-WG-TEC-v1-0.doc
Date:	13/12/2006~~13/12/2006~~~~02/10/2006~~
Activity:	TCG MPI WG ~~(e.g. JRA2: Quality Assurance)~~
Document status:	**DRAFT**
Document link:	https://edms.cern.ch/document/edmsId/version

~~*[Note: Text in blue italics is included to provide guidance to the author and should be deleted before publishing the document.]*~~

Abstract: This document summarises the recommendations of the MPI working group for improving MPI support on the EGEE grid. It includes an analysis of existing limitations and proposes solutions in the areas of site configuration, middleware, and user submission methodology.

Document Log

Issue	Date	Comment	Author/Partner
0-0	22/08/2006	First draft	Stephen Childs/TCD
0-1	13/12/2006	Incorporating input from MPI workshop	Stephen Childs/TCD

Document Change Record

Issue	Item	Reason for Change

TABLE OF CONTENTS

TABLE OF TABLES

1. INTRODUCTION

1.1 PURPOSE

The MPI working group was set up to investigate and improve the ~~poor~~ support for parallel jobs within the EGEE middleware, with particular reference to the widely-used Message Passing Interface (MPI) standard. While our primary focus is on MPI, our findings should~~are~~ also be relevant to other methods for submitting parallel jobs. ~~(Which other methods?)~~

1.2 DOCUMENT ORGANISATION

The document is organised as follows:

- Problem statement
- Analysis of existing limitations in the EGEE middleware
- Recommended short-term actions
- Pointers for long-term improvements to parallel job support

1.3 APPLICATION AREA

TCG

1.4 REFERENCES

Table 1: Table of references

R 1	**Preparing the LCG for MPI applications, Richard de Jong and Mathijs Koot, https://twiki.cern.ch/twiki/bin/view/LCG/ImplementationOfMpi**
R 2	

1.5 DOCUMENT AMENDMENT PROCEDURE

1.6 TERMINOLOGY

Definitions

JDL	Job Description Language
MPI	Message Passing Interface
WMS	Workload Management System

Glossary

2. SUMMARY

2.1 DEFINITION

An MPI job is one which:

- Requires simultaneous parallel access to multiple compute nodes
- Requires libraries and tools for parallel execution (e.g. MPICH) to be installed and accessible on compute nodes
- Is an EGEE job submitted by the normal means

We focus on single-site MPI on clusters with a shared home filesystem.

2.2 RECOMMENDATIONS

For convenience, the recommendations of this report are summarised here. We recommend that sites supporting MPI are configured to publish the particular flavours and versions of MPI they support in the information system. They should also make environment variables available to jobs to allow MPI libraries and tools to be located. The MPI-start package produced by the int.eu.grid project will be installed to hide some of the complexity of MPI setup from users.

We recommend minor modifications to the middleware that allow multiple CPUs to be requested for normal jobs. The MPICH job type should be deprecated as it hard-codes assumptions about site setup, and does not meet the needs of either site admins or users.

We recommend that users should submit MPI jobs using wrapper scripts that set up their desired environment correctly. Templates will be provided which users can customise as needed.

~~It should be noted that although this working group's official mandate was to suggest methods for improving MPI support, many of the issues are common to other methods for parallel job submission. (Which other methods are you thinking of?)~~

Formatted: Heading 2,A.B.C.,Heading2-bio,Career Exp.,H2,T2,H21,T21,A.B.C.1,Heading2-bio1,Career Exp.1,H22,T22,A.B.C.2,Heading2-bio2,Career Exp.2,H23,T23,A.B.C.3,Heading2-bio3,Career Exp.3,H24,T24,A.B.C.4,Heading2-bio4,Career Exp.4,H25,T25,A.B.C.5,Heading2-bio5,H26

Formatted: Bullets and Numbering

3. ANALYSIS OF CURRENT SUPPORT

3.1 MPICH JOB TYPE

The current release of the EGEE middleware (gLite 3.0.1 at time of writing) provides the MPICH job type to support~~s~~ MPI jobs ~~by providing the MPICH job type~~. Users wishing to submit MPI(CH) jobs set the JobType variable to "MPICH" rather than "Normal" and set the variable "NodeNumber" to indicate how many nodes they require. This causes the following actions by the middleware:

- The WM API code on the UI checks for the NodeNumber parameter which is mandatory for MPICH jobs.
- The WM API code on the UI adds an expression[1] to the job's ClassAd to ensure that the job can only be sent to sites with MPICH installed and the required number of CPUs available .
- The WMS job adapter checks that the target job manager is supported (only "lsf" and "pbs" are).

[1] `(other.GlueCEInfoTotalCPUs >= NodeNumber) && Member(other.GlueHostApplicationSoftwareRuntimeEnvironment, "MPICH"`

- The WMS job adapter reads the requested number of nodes and sets two variables in the Globus RSL accordingly: "count", and "hostCount".
- The WMS job adapter creates an MPI-specific job wrapper using "MpiLsfJobWrapper" or "MpiPbsJobWrapper" as appropriate. The job wrapper creates a ".mpi" directory, and runs the user's executable using the MPI "mpirun" command, passing the list of hosts allocated by the batch system. ~~(Question: Does it actually use the .mpi directory for anything?)~~

This approach has ~~many~~ significant limitations, and is not the right solution for parallel job submission. The main problems are:

- The range of job managers supported is very limited. In particular the widely-used "lcgpbs" and "torque" job managers are not supported, although in fact they ~~y can be made to~~ work. Because the list of supported job managers is compiled into the WMS code, it is very difficult to modify it. Sites wanting to support MPI currently have two choices: either they configure their site to use one of the supported jobmanagers, or they falsely advertise their jobmanager in the information system. The restrictions imposed by the WMS are unnecessary and should be removed.
- ~~A~~The MPICH job type assumes too much about the configuration of the executing site: for example, even at sites that have installed MPICH tools, many administrators have replaced mpirun ~~has been superceded by~~with mpiexec. Again, site administrators can work around the hard-coded mpirun invocation by replacing it with a local wrapper, but they shouldn't have to.
- The level of abstraction is far too low: MPI is just one means of parallel job submission, and MPICH is just one implementation of MPI. If we were to support all possible methods for parallel job submission in this way, the number of job types would explode.

Formatted: Bullets and Numbering

~~3.2SITE CONFIGURATION~~

~~To ensure that MPI sites are configured consistently and that available functionality is correctly advertised, MPI configuration should be integrated into YAIM and Quattor. Note that this does not mean that YAIM should be able to completely configure a site for MPI, but rather that it should provide for integrating an MPI site with the Grid middleware.~~

~~Work has been done on this by two students from the University of Amsterdam, and their recommendations can be found in [R1]. The following is a summary of their work, and of the discussions in the MPI working group.~~

Formatted: Bullets and Numbering

~~3.2.1ENVIRONMENT VARIABLES~~

~~For each implementation of MPI that is installed on the system the following environment variables should be defined:~~

~~MPI_<MPI flavour>_PATH~~	~~Path to root of library installation~~	~~MPI_MPICH2_PATH="/opt/mpich2-1.0.4"~~
~~MPI_<MPI flavour>_VERSION~~	~~Library version~~	~~MPI_MPICH2_VERSION="1.0.4"~~

~~Currently the following MPI implementations are supported:~~

~~MPICH~~		
~~MPICH2~~		
~~LAM 6 (do we want to support this one?)~~		
~~LAM 7~~		

OPENMPI		

3.2.2INFORMATION SYSTEM

Users must be able to locate sites that support a particular MPI implementation so they can target their jobs to a suitably configured site.

For each installed implementation of MPI, a corresponding GlueHostApplicationSoftwareRunTimeEnvironment variable should be published in the information system as <LIBRARYNAME> <VERSION>. (And probably also one with just <LIBRARYNAME> for those who don't care about the version.) For example, a site with MPICH-1.2.7 installed should publish the following:

```
GlueHostApplicationSoftwareRunTimeEnvironment: MPICH-1.2.7
```

3.2.3LOCAL CONFIGURATION

Sites publishing support for MPI within EGEE must ensure that certain conditions are met by their MPI configuration(s). In particular, that the binaries for a particular flavour of MPI are available at standard locations relative to the defined installation path.

An SFT based on the standard MPI wrapper script could be used to verify that local configuration has been done correctly.

3.3MPI FUNCTIONALITY TESTING

It would be useful for prospective MPI users to have access to functional tests that indicate whether a site advertising MPI support is correctly configured. We have done some work on adding an MPI test to the existing Site Functional Tests (SFT) suite. This proved to be more difficult than anticipated because testing MPI currently requires the use of a different job type. If the recommended changes are made to the WMS then it will be much easier to add an MPI test to the SFT suite.

4. RECOMMENDED ACTIONS

4.1 OVERVIEW

Our recommendations are divided into two sections: firstly, site configuration changes that will be deployed on existing MPI sites and recommended for integration into the standard release, and secondly, minor changes to the middleware.

We first recommend some minor changes to the middleware that would help users to submit parallel jobs.

4.2 (NEED TO ADD A SECTION ABOUT INFORMATION SYSTEM CHANGES: NEED PER-QUEUE PARAMETER WHICH GIVES MAXCPUSPERJOB AND NEED PER-QUEUE "FLAGS". THE FIRST BECAUSE THE TOTAL NUMBER OF CPUS MAY NOT BE AVAILABLE TO MPI JOBS; THE SECOND BECAUSE WE MAY ONLY WANT TO ADVERTISE MPI AVAILABILITY ON SPECIFIC QUEUES AND NOT THE SITE AS A WHOLE.)SITE CONFIGURATION

To ensure that MPI sites are configured consistently and that available functionality is correctly advertised, MPI configuration should be integrated into YAIM and Quattor. Note that this does not mean that YAIM should be able to completely configure a site for MPI, but rather that it should provide for integrating an MPI site with the Grid middleware.

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Work has been done on this by two students from the University of Amsterdam, and their recommendations can be found in [R1]. The following is a summary of their work, which was further clarified during the joint EGEE/int.eu.grid workshop in December 2006.

4.2.1 INFORMATION SYSTEM

Formatted: Bullets and Numbering

Users must be able to locate sites that support a particular MPI implementation so they can target their jobs to a suitably-configured site.

For each installed implementation (flavour) of MPI, a corresponding GlueHostApplicationSoftwareRunTimeEnvironment variable should be published in the information system as <FLAVOUR>-<VERSION> (And also one with just <FLAVOUR> for those who don't care about the version.) For example, a site with MPICH-1.2.7 installed should publish the following:

```
GlueHostApplicationSoftwareRunTimeEnvironment: MPICH
GlueHostApplicationSoftwareRunTimeEnvironment: MPICH-1.2.7
```

Formatted: Font: (Default) Courier New, 10 pt

Sites should also advertise the availability of special interconnects if they are available. This is in the format MPI-<Interconnect>. Examples of interconnect might be Ethernet, Infiniband, Myrinet, SCI. So:

```
GlueHostApplicationSoftwareRunTimeEnvironment: MPI-Infiniband
```

The default compiler suite is assumed to be gcc. Sites with special compilers (e.g. Intel compilers) may advertise this by appending a string representing the compiler to the Glue variable published. For example:

```
GlueHostApplicationSoftwareRunTimeEnvironment: OPENMPI-1.0.2-ICC
```

Formatted: Font: (Default) Courier New, 10 pt

Formatted: Font: (Default) Courier New, 10 pt

It should be noted that the definition of compilers will not be fully supported initially as there are still issues to resolve. We intend to do this by working with sites that have special compilers available, and their associated user communities.

4.2.2 ENVIRONMENT VARIABLES

Formatted: Bullets and Numbering

For each implementation of MPI that is installed on the system the following environment variables should be defined on worker nodes and made available to jobs:

MPI_<MPI flavour>_PATH	Path to root of library installation	MPI_MPICH2_PATH="/opt/mpich2-1.0.4"
MPI_<MPI flavour>_VERSION	Library version	MPI_MPICH2_VERSION="1.0.4"
MPI_<MPI flavour>_COMPILER	Compiler used to build library	MPI_MPICH2_COMPILER="gcc"

Currently the following MPI implementations are supported: MPICH, MPICH2, LAM, OPENMPI.

Formatted: Normal

We have also discussed extensions to allow for cases where multiple versions of a particular flavour are available, but further work is needed here before

4.2.3 LOCAL CONFIGURATION

Formatted: Bullets and Numbering

Sites publishing support for MPI within EGEE should ensure that certain conditions are met by their MPI configuration(s). In particular, while the MPICH job-type is still in use, mpirun should be replaced with a dummy script that simply executes the program passed to it.

Packages have been created for standard MPI libraries and tools and are available at http://quattorsrv.lal.in2p3.fr/packages/mpi/. These packages will be added to ETICS so that they can be made generally available. The i2g-mpi-start package (see below) will also need to be installed on worker nodes.

4.2.4 TESTING

An SAM test based on the standard MPI wrapper script could be used to verify that local configuration has been done correctly. Some work has been done on this but further investigation is needed before a solution can be deployed.

4.24.3 MIDDLEWARE CHANGES

As previously stated, we believe that the "MPICH" job is the wrong level of abstraction for parallel job support. Any approach that attempts to ~~perform~~ implement ~~all~~ configuration in the RB ~~necessary~~ is too inflexible to work with the variety of setups present at EGEE sites. Rather than attempting to write a suite of job types with different wrappers, we recommend that **the "Normal" job type should be extended to allow the use of multiple nodes per job. The MPICH job type should be retained initially for compatibility but should be deprecated and removed once upgraded middleware has been widely deployed.**

Essentially this approach leaves it up to the user to wrap their executable in such a way that it is run in parallel in the cluster it eventually executes on; t~~. This is not a long-term approach, as it users should not have to know how a parallel job should be run on a part~~his provides great flexibility. There is some cost in terms of increased complexity for the user, but this should not be a significant problem if appropriate documentation and examples are provided. ~~icular cluster. Eventually some scheme will have to be agreed to name parallel environments, which are then selected by the user and configured by site administrators. For further details see Section 5.~~

This change is low-impact and will make parallel job submission much easier, without any negative effects on existing usage of either the "Normal" or "MPICH" job types. These changes have been implemented in a gLite 3.0.1 LCG RB and UI (WMS software release 2.1.74) and have been tested in production within Grid-Ireland. **A similar change must be made to the gLite WMS** services. We now describe~~tail~~ the ~~minor~~ code changes that are necessary (code patches are in the appendix).

~~These changes have been implemented in a gLite 3.0.1 LCG RB and UI (WMS software release 2.1.74) and have been tested by Grid-Ireland. A similar change must be made to the WMS-flavored services.~~

~~(Should also mention that passing parameters to the batch system is vital to increase the number of sites supporting MPI. If not available, the co-scheduling of MPI and non-MPI jobs is very inefficient and discourages support of MPI.)~~

4.2.14.3.1 WMS changes

The job adapter code must be changed to allow the "NodeNumber" parameter to be used with any job type. This means moving the code for adding "count" and "hostCount" variables out of the MPICH-specific section, and placing it where "Normal" jobs are processed. A check for the existence of the nodenumber variable also needs to be added. If the "NodeNumber" parameter is set, then a requirement is added so that only sites whose have more than "NodeNumber" CPUs ~~free~~ will match against the job. ~~(We should probably be consistent with the terminology we want to support. At the moment we allocate CPUs and not nodes. I would suggest we start with that and then try to add CPUs and nodes and some point in the (distant) future.)~~

4.2.24.3.2 WMS API changes

Formatted: Hyperlink, Font: (Default) Courier New
Formatted: Default Paragraph Font
Field Code Changed
Formatted: Heading 3,l3,H3,Level 2 Heading,Level 2,h2,h3,1.2.3.,T3,H31,T31,l31,Level 2 Heading1,Level 21,h21,h31,1.2.3.1,H32,T32,l32,Level 2 Heading2,Level 22,h22,h32,1.2.3.2,H33,T33,l33,Level 2 Heading3,Level 23,h23,h33,1.2.3.3,H34,T34,l34,Level 2 Heading4,h24
Formatted: Bullets and Numbering
Formatted: Bullets and Numbering
Formatted: Font: Bold
Formatted: Font: Bold
Formatted: Font: Bold
Formatted: Bullets and Numbering
Formatted: Bullets and Numbering

Changes must be made to the WMS API code used on the UI to allow the "NodeNumber" variable to be set for normal jobs. Again, this is a very simple change. The existing code checks that if one of "NodeNumber" or "MPICH" is set, the other must be; this check can now be removed.

5. MPI JOB SUBMISSION METHODOLOGY

As described above, we want to move MPI setup out of the WMS to provide a flexible solution that can be adapted to changing configurations. In principle, this means more work for the user, but we hope that the overall user experience can be improved by providing well-documented use cases.

In order to hide some of the complexity, we plan to use MPI-start, a script developed within the int.eu.grid project to provide a layer of abstraction between the user and the various different implementations of MPI. This script detects the configuration of a site at runtime and runs the user's executable using the requested version of MPI.

From the user's point of view:

- The user creates ~~create~~as ~~a~~ wrapper script ~~t~~ that will first call mpistart to set up the environment, then perform any setup operations they need (e.g. compilation of their binary, retrieval of input data, etc.), and finally run the binary using mpistart. (~~the desired parallel environment this could be as simple as "mpiexec $EXECUTABLE";~~ A sample wrapper script will be made available via the UIG and users can then customise this as necessary.)
- The user creates a JDL with the following characteristics:
  - JobType "MPICH" to allow multiple CPUs to be requested (this will go away if the recommended changes are made to the middleware).
  - The "NodeNumber" parameter is set to the number of CPUs they require.
  - The MPI flavour they require is added to the software requirements to ensure that only sites supporting this are matched.
  - The flavour of MPI they need is passed as an argument to the wrapper script.
  - ~~An example JDLCurrently they would probably need to manually add "MPICH" to their software requirements as it's the only way of requesting MPI support. If they are using another parallel environment, they should add a string appropriate for that instead.~~

~~As the job is of type "Normal", the check for job manager type would not be invoked, and the job should match against any site that supports MPI.~~ and wrapper script are available in Appendix 3. These are for illustrative purposes – canonical examples will be made available via the UIG.

6. LONG-TERM IMPROVEMENTS IN SUPPORT FOR PARALLEL JOBS

~~**NOT SURE IF THIS LEVEL OF STUFF SHOULD BE IN; MAYBE WE SHOULD HOLD IT BACK FOR FUTURE DISCUSSIONS. WE SHOULD PROBABLY MAKE A POINT OF THE NEED FOR PASSING JDL REQUIREMENTS THROUGH – IT'S NOT JUST A QUESTION OF EFFICIENCY BUT ALSO OF FUNCTIONALITY IN THE MPI CASE.**~~

~~**PERHAPS SOME DISCUSSION OF THE PROS AND CONS OF MOVING SOME OF THE WRAPPER SCRIPT FUNCTIONALITY INTO AN "MPI" JOBTYPE? I DON'T THINK THIS WOULD BE DISASTROUS ESPECIALLY IF THE SCRIPT WAS SOURCED FROM THE FILESYSTEM RATHER THAN WRITTEN INTO THE C++ CODE.**~~

Formatted: Normal

Formatted: Bullets and Numbering

Formatted: Bulleted + Level: 2 + Aligned at: 0.75" + Tab after: 1" + Indent at: 1"

Formatted: Bullets and Numbering

~~(I'm not opposed to having this section in the report. However, I think that we need an introductory paragraph which says that we follow the simple changes above, look at the actual usage of the system, and then decide the best direction for simplifying and extending the use of MPI. The text below is one example. What I want to avoid is having JRA1 say that we should give them all of the use cases and they'll do all of the changes in one go. This, as we've seen, leads to disaster.)~~

~~The EGEE Grid middleware needs to provide better support for parallel jobs. This is becoming more and more important as new application communities get involved with the Grid. The following is one proposal (largely due to David Golden of the Dublin Institute for Advanced Studies):~~

~~There should be another JobType, "Parallel", intended for running a single parallel job of N parallel processes. This job type should allow the parameters "NodeNumber" or perhaps better "ProcCount" as "Node" more usually refers to a single physical machine which may have multiple CPUs.~~

~~Somewhat like Sun Grid Engine, it should also have a ParallelEnvironment parameter encapsulating the expected parallel job conventions, e.g. "MPI2-BSF90" meaning that the job's "Executable" is a script written to expect to compile something with an "mpif90" (etc.) [BSF90 => bootstrap fortran 90] it finds in the environment path, then run the resulting executable with the "mpiexec2" it finds in the environment path, with an mpiexec conforming to the MPI-2 mpiexec command line parameter syntax.~~

~~I'd specify that a toplevel .jdl parameter RealTimeDuration a.k.a. WallTime was at least allowed too. It is highly unusual for it _not_ to be a locally required parameter on systems large enough to support nontrivial parallel jobs. For parallel processes, one can typically assume that the max cpu time required is always going to be ProcCount * RealTimeDuration, so it's effectively redundant in the minds of most end-users to have to specify cpu time, though for shared memory supercomputers as opposed to job gets exclusive node lock clusters it might be nice to allow independent specification as it becomes a more important distinction there.~~

~~If a parameter "Count" is supplied, it should probably be to kick off multiple similar parallel jobs. As this combines the features of the existing "Multiple" job type and the proposed "Parallel" job type, it should be yet another job type "MultiParallel").~~

~~Most parallel job batch systems other than SGE don't currently have a native concept of ParallelEnvironment, but this is a feature that should be added in the future. It would be possible to add this to Torque at least. In any case, if the system doesn't have a native idea of ParallelEnvironment, it could advertise that it only supports a single Parallel Environment.~~

All the recommendations made so far are low-impact, yet will make it much easier to use MPI on the EGEE infrastructure. However, there are still a number of issues that will need to be investigated in the future to further improve the user experience.

- All requirements submitted by the user must be made available at the Compute Element at job run-time. This would greatly increase the sites' ability to schedule MPI and non-MPI jobs together and would also make it possible for job-specific setup to be performed.
- The information system should be extended to allow a per-queue parameter which gives MaxCPUsPerJob and a facility per-queue "flags". The first because the total number of CPUs may not be available to MPI jobs; the second because we may only want to advertise MPI availability on specific queues and not the site as a whole
- More controllable allocation of CPUs so users can request that CPUs are allocated on the same physical machine. This requires further investigation as interactions with the various LRMS are not fully understood.

Formatted: Bullets and Numbering

Formatted: Bulleted + Level: 1 + Aligned at: 0.25" + Tab after: 0.5" + Indent at: 0.5"

- Based on feedback from users who use the job submission scheme outlined above, the possibility of incorporating the MPI start setup code into a WMS job type should be examined. This could be an effective way of hiding complexity from users, particularly if combined with pre- and post-processing hooks as envisaged in the gLite WMS.
- Cross-site MPI is a difficult problem to solve but is demanded by some applications. We intend to monitor work done on this within int.eu.grid. We hope that future solutions they develop in this area would be suitable for integration with EGEE middleware.

7. APPENDIX 1

```
Index: JobAdapter.cpp
===================================================================
RCS file: /local/reps/lcgware/workload/planning/jobadapter/JobAdapter.cpp,v
retrieving revision 1.61
diff -u -r1.61 JobAdapter.cpp
--- JobAdapter.cpp        7 Sep 2005 21:16:36 -0000        1.61
+++ JobAdapter.cpp        17 Aug 2006 08:49:17 -0000
@@ -325,6 +325,18 @@
   string ljobtype(jobtype);
   transform(ljobtype.begin(), ljobtype.end(), ljobtype.begin(),
::tolower);

+    /* Mandatory */
+    /* node number is mandatory for the mpich job */
+    int    nodenumber = requestad::get_node_number(*m_ad);
+    if (nodenumber > 0 ) {
+    string nn(boost::lexical_cast<string>(nodenumber));
+
+    globusrsl.append("(count=");
+    globusrsl.append(nn);
+    globusrsl.append(")(hostCount=");
+    globusrsl.append(nn);
+    globusrsl.append(")");
+    }
   if (ljobtype == "mpich") {
     /* Mandatory */
     /* lrms type is mandatory for the mpich job */
@@ -339,16 +351,6 @@
                                            helper_id);
     }

-    /* Mandatory */
-    /* node number is mandatory for the mpich job */
-    int    nodenumber = requestad::get_node_number(*m_ad);
-    string nn(boost::lexical_cast<string>(nodenumber));
-
-    globusrsl.append("(count=");
-    globusrsl.append(nn);
-    globusrsl.append(")(hostCount=");
-    globusrsl.append(nn);
-    globusrsl.append(")");

     if (llrmstype == "lsf") {
       jw = new jobwrapper::MpiLsfJobWrapper(executable);
@@ -358,7 +360,6 @@
     } else {
```

```
       // not possible;
     }
-    jw->nodes(nodenumber);

   } else if (ljobtype == "interactive") {
     if (find_if(env.begin(), env.end(),
@@ -394,6 +395,9 @@
     jw = new jobwrapper::JobWrapper(executable);
   }

+   if (nodenumber > 0 && jw) {
+    jw->nodes(nodenumber);
+  }
   /* Define the VO for the wrapper script, if available */
   string vo(requestad::get_virtual_organisation(*m_ad));
   if (!vo.empty()) {
```

8. APPENDIX 2

```
Index: JobAd.cpp
===================================================================
RCS file: /local/reps/lcgware/workload/common/requestad/JobAd.cpp,v
retrieving revision 1.79
diff -u -r1.79 JobAd.cpp
--- JobAd.cpp        10 Nov 2004 09:51:58 -0000        1.79
+++ JobAd.cpp        17 Aug 2006 08:49:59 -0000
@@ -735,8 +735,8 @@
       - ShPort -> interacive
       - JobSteps-> checkponintable
     */
-        if ( hasAttribute(JDL::NODENUMB) && !hasAttribute ( JDL::JOBTYPE,
JDL_JOBTYPE_MPICH) )
-                throw   AdSemanticGroupException(__FILE__ , __LINE__
,METHOD,WL_JDLMANDATORY,"Nodenumber, MPICH jobs") ;
+/*        if ( hasAttribute(JDL::NODENUMB) && !hasAttribute (
JDL::JOBTYPE,
JDL_JOBTYPE_MPICH) )
+                throw   AdSemanticGroupException(__FILE__ , __LINE__
,METHOD,WL_JDLMANDATORY,"Nodenumber, MPICH jobs") ;*/
         if ( hasAttribute(JDL::SHPORT) && !hasAttribute ( JDL::JOBTYPE,
JDL_JOBTYPE_INTERACTIVE ) )
                 throw   AdSemanticGroupException(__FILE__ , __LINE__
,METHOD,WL_JDLMANDATORY,"ShadowPort, INTERACTIVE jobs") ;
         if ( hasAttribute(JDL::CHKPT_STEPS)  )
@@ -749,7 +749,7 @@
     if (hasAttribute ( JDL::JOBTYPE, JDL_JOBTYPE_MPICH) ){
         // TBD todavia faltan parentesis al cabo y al final (logica
booleana)
         if (!hasAttribute(JDL::NODENUMB) )
-                throw AdSemanticGroupException (__FILE__ , __LINE__
,METHOD,WL_JDLMANDATORY, "Nodenumber, MPICH jobs") ;
+                throw AdSemanticGroupException (__FILE__ , __LINE__
,METHOD,WL_JDLMANDATORY, "NodeNumber, MPICH jobs") ;
         // Modify Requirements:
         string buffer  = "";
         char NodeNum [1024] ;
```

9. APPENDIX 3

An example JDL file for submitting MPI jobs via mpi-start and the agreed conventions is:

```
JobType = "MPICH";
NodeNumber = 2;
Executable = "mpi-start-wrapper.sh";
Arguments = "mpi-test mpich2 2";
StdOutput = "mpi-test.out";
StdError = "mpi-test.err";
InputSandbox = {"mpi-start-wrapper.sh","mpi-test.c"};
OutputSandbox = {"mpi-test.err","mpi-test.out"};
Requirements = RegExp("grid.*.lal.in2p3.fr.*sdj$",other.GlueCEUniqueID);
#Requirements = RegExp(".*egee.fr.cgg.com.*",other.GlueCEUniqueID);
```

The wrapper script to use is:

```
#!/bin/bash

export I2G_MPI_START_PATH=/opt/i2g

# Pull in the arguments.
EXE=$1
MPI_FLAVOR=$2
NUM_PROCS=$3

# Convert flavor to uppercase for environmental variables.
MPI_FLAVOR_UPPER=`echo $MPI_FLAVOR | tr '[:lower:]' '[:upper:]'`

# Pull out the correct paths for the requested flavor.
eval MPI_PATH=`printenv MPI_${MPI_FLAVOR_UPPER}_PATH`

# Add the MPI paths for the selected flavor.
PATH=$MPI_PATH/bin:$PATH
LD_LIBRARY_PATH=$MPI_PATH/lib:$LD_LIBRARY_PATH

# Compile the program.
echo "compiling program ${EXE}"
mpicc -o $EXE $EXE.c
if [ ! $? -eq 0 ]; then
        echo "Error compiling program.  Exiting..."
        exit 1
fi

# Must make the executable an absolute file name.
EXE=`pwd`/$EXE

# Should be set by the sysadmin in final configuration.
export I2G_MPI_START=$I2G_MPI_START_PATH/bin/mpi-start

# Ensure the prefix is correctly set.  Don't rely on the defaults.
eval I2G_${MPI_FLAVOR_UPPER}_PREFIX=$MPI_PATH
export I2G_${MPI_FLAVOR_UPPER}_PREFIX

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$EXE
export I2G_MPI_APPLICATION_ARGS=
export I2G_MPI_NP=$NUM_PROCS
```

Formatted: Heading 1,H1,H11,H12,H13,H14,H15,H16,H17,H18,H19,H110,H111,H112,H113,H114,H115,H116,H117,H118,H119,H120,H121,H122,H123,H124,H125,H126,H127,H128,H129,H1110,H131,H141,H151,H161,H171,H181,H191,H1101,H1111,H1121,H1131,H1141,H1151,H1161,H1171,H1181,H1191

Formatted: Bullets and Numbering

Formatted: Font: 11 pt

Formatted: Polish

```
export I2G_MPI_TYPE=$MPI_FLAVOR
export I2G_MPI_START_VERBOSE=1
export I2G_MPI_START_DEBUG=1

# Invoke mpi-start.
$I2G_MPI_START
```

Formatted: HTML Preformatted