Parallel jobs with MPI
Hands-on tutorial

Enol Fernández del Castillo
Instituto de Física de Cantabria
Tutorial Survey

Your feedback would be appreciated - please visit us online at:

https://www.egi.eu/online_surveys/index.html
Introduction

• What is a parallel job?
  – A job that uses more than one CPU simultaneously to solve a problem

• Why go parallel?
  – Save time
  – Solve larger problems
  – Limits of hardware (speed, integration limits, economic limits)
• Shared Memory Model
  – Tasks share a common address space, which they read and write asynchronously.
  – Various mechanisms such as locks / semaphores may be used to control access to the shared memory.
  – No need to specify explicitly the communication of data between tasks. Program development can often be simplified.
• Message Passing Model
  – Tasks that use their own local memory during computation.
  – Multiple tasks can reside on the same physical machine as well across an arbitrary number of machines.
  – Tasks exchange data through communications by sending and receiving messages.
  – Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation.

• MPI (Message Passing Interface) is an API for programming Message Passing parallel applications
entials

• Defines uniform and standard API (vendor neutral) for message passing
• Allows efficient implementation
• Provides C, C++ and Fortran bindings
• Several MPI implementations available:
  – From hardware providers (IBM, HP, SGI….) optimized for their systems
  – Academic implementations
• Most extended implementations:
  – MPICH/MPICH2 (from ANL/MSU), includes support for a wide range of devices (even using globus from communication)
  – Open MPI (join effort from FT-MPI, LA-MPI, LAN/MPI and PACX-MPI developers): modular implementation that allows the use of advanced hardware during runtime
Processes

• Every MPI job consist of \( N \) processes
  – \( 1 \leq N \), \( N = 1 \) is valid

• Each process in a Job could execute a different binary
  – In general it's always the same binary which executes different code path based on the process number

• Processes are included in **groups**
Groups

• A group is an ordered set of processes
• Every process in a group has an unique rank inside the group
  – From 0 to #PROCS -1
  – Processes can have different ranks in different groups
• Groups are defined with MPI Communicators
Hello World (I)

```c
#include <mpi.h>  /* for MPI functions */
#include <stdio.h> /* for printf */

int main(int argc, char *argv[]) {
    int rank = 0, size = 0;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    printf("Hello my rank is %i of %i\n", rank, size);

    MPI_Finalize();
    return 0;
}
```
Hello World (II)

- **MPI_Init**
  - Initialize the MPI system
  - Must be called before any other MPI function can be called

- **MPI_Comm_size**
  - Return the number of processes in the processes group
  - `MPI_COMM_WORLD` is a default group with all processes

- **MPI_Comm_rank**
  - Return the rank of the current process in the group

- **MPI_Finalize**
  - Shutdown the MPI system
  - After this call MPI must not be called
Hello World (III)

• Compile

$ mpicc hello.c -o hello

• Run

$ mpirun -np 5 hello
Hello my rank is 0 of 5
Hello my rank is 3 of 5
Hello my rank is 1 of 5
Hello my rank is 2 of 5
Hello my rank is 4 of 5
```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
    int rank = 0, size = 0;
    int next = 0, prev = 0;
    int data = 0;

    int status;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    prev = (rank + (size-1)) % size;
    next = (rank + 1) % size;

    MPI_Send(&rank, 1, MPI_INT, next, 0, MPI_COMM_WORLD);
    MPI_Recv(&data, 1, MPI_INT, prev, 0, MPI_COMM_WORLD, &status);

    printf("%i received %i\n", rank, data);
    MPI_Finalize();
    return 0;
}
```
Collective Operations

- **Broadcast**
  - P0: A
  - P1: A
  - P2: A
  - P3: A

- **Scatter**
  - P0: A B C D
  - P1: B
  - P2: C
  - P3: D

- **Gather**
  - P0: A
  - P1: B
  - P2: C
  - P3: D

- **All gather**
  - P0: A B C D
  - P1: A B C D
  - P2: A B C D
  - P3: A B C D

- **All to All**
  - P0: A0 A1 A2 A3
  - P1: B0 B1 B2 B3
  - P2: C0 C1 C2 C3
  - P3: D0 D1 D2 D3

EGI-UF. Vilnius, Apr 2011.
int main(int argc,char *argv[]) {
    int n, myid, numprocs, i;
    double mypi, pi, h, sum, x;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);

    n = 10000; /* default # of rectangles */

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

    h = 1.0 / (double) n;
    sum = 0.0;
    for (i = myid + 1; i <= n; i += numprocs) {
        x = h * ((double)i - 0.5);
        sum += 4/(1+x*x);
    }
    mypi = h * sum;

    MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

    if (myid == 0) {
        printf("pi is approximately %.16f\n", pi);
    }

    MPI_Finalize();
    return 0;
}

\[ \pi \approx \int_{-1/2}^{1/2} \frac{4}{1 + x^2} \, dx \]
Take into account...

• MPI is no Magic:
  – “I installed XXX-MPI in 3 of my machines but my Photoshop doesn't run faster !?!”

• MPI is not an implementation

• MPI is not the holy grail
  – Some problems don't really fit into the message passing paradigm
  – It's quite easy to write a parallel MPI application that perform worse than the sequential version
  – MPI works fine on shared memory systems, but maybe an OpenMP solution would work better
Some MPI resources

- MPI Tutorials:
  - [https://computing.llnl.gov/tutorials/mpi/](https://computing.llnl.gov/tutorials/mpi/)
Submission/Allocation:
- Definition of job characteristics
- Search and select adequate resources
- Allocate (or co-allocate) resources for job

Execution:
- File distribution
- Batch system interaction
- Parallel job runtime implementation details
JobType = "Normal";
CPUNumber = 4;
Executable = "/bin/hostname";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};

$ glite-wms-job-submit -a -o id job.jdl
...
Your job identifier is:
https://wms01.ncg.ingrid.pt:9000/K3U3wqBwzjGMhdc3PGS_IQ
...
$ glite-wms-job-output -i id --dir $PWD
...
have been successfully retrieved and stored in the directory:
/home/enol/tests/eciencia/parallel/enol_K3U3wqBwzjGMhdc3PGS_IQ

$ cat enol_K3U3wqBwzjGMhdc3PGS_IQ/std.out
worker-node-1-3.local
How to start a MPI job?

• There is no standard way of starting an MPI application
  – No common syntax for mpirun, mpiexec support optional

• The cluster where the MPI job is supposed to run doesn't have a shared file system
  – How to distribute the binary and input files?
  – How to gather the output?

• Different clusters over the Grid are managed by different Local Resource Management Systems (PBS, LSF, SGE,...)
  – Where is the list of machines that the job can use?
  – What is the correct format for this list?

• How to compile MPI program?
  – How can a physicist working on Windows workstation compile his code for/with an Itanium MPI implementation?
MPI-Start

• Specify a unique interface to the upper layer to run a Parallel (MPI) job
• Allow the support of new MPI implementations without modifications in the Grid middleware
• Portable
• Modular and extensible architecture
• Extensive debugging options
MPI-Start architecture
MPI-Start for users

• Single interface for all parallel jobs:
  – No need to learn a different command line option every time a MPI version changes
  – No need to learn how each scheduler manages the hosts

• Control of process mapping:
  – One process per host
  – N processes per host
  – K total processes

• File distribution
  – No need to worry (much) about shared or not filesystems

• Customizable with hooks
  – Compilation
  – input preparation
  – management of output
MPI-Start for admins

• Single interface for all parallel jobs:
  – No need to configure a different Execution Environment / Run Time Environment for each type of job

• Easy to deploy for admins without much experience:
  – Default MPI installations paths for SL5 (current target for EMI) installations detected
  – Yaim module for configuration

• Customizable with hooks for sites with specific requirements
MPI-Start flow

START

Scheduler Plugin

NO

Do we have a scheduler plugin for the current environment?

Ask Scheduler plugin for a machinefile in default format

Execution Plugin

NO

Do we have a plugin for the selected MPI?

Activate MPI Plugin

Prepare mpirun

Hooks Plugins

Trigger pre-run hooks

Start mpirun

Trigger post-run hooks

Dump Env

EXIT
Using MPI-Start

The script:

```bash
#!/bin/sh
# This is a script to show how mpi-start is called

# Set environment variables needed by mpi-start
export I2G_MPI_APPLICATION=/bin/hostname
export I2G_MPI_APPLICATION_ARGS=
export I2G_MPI_TYPE=openmpi
export I2G_MPI_PRECOMMAND=time
export I2G_MPI_START=/opt/i2g/bin/mpi-start

# Execute mpi-start
$I2G_MPI_START
```

The submission (in SGE):

```bash
$qsub -S /bin/bash -pe mpi 2 ./test2mpistart.sh
```

The StdOut:

```bash
Scientific Linux CERN SLC release 4.5 (Beryllium)
Scientific Linux CERN SLC release 4.5 (Beryllium)
lflip30.lip.pt
lflip31.lip.pt
```

The StdErr:

```bash
Scientific Linux CERN SLC release 4.5 (Beryllium)
Scientific Linux CERN SLC release 4.5 (Beryllium)
real 0m0.731s
user 0m0.021s
sys 0m0.013s
```

- MPI commands are transparent to the user
  - No explicit mpiexec/mpirun instruction
  - Start the script via normal LRMS submission
## Using MPI-Start: variables

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2G_MPI_APPLICATION</td>
<td>The application binary to execute.</td>
</tr>
<tr>
<td>I2G_MPI_APPLICATION_ARGS</td>
<td>The command line parameters for the application</td>
</tr>
<tr>
<td>I2G_MPI_TYPE</td>
<td>The name of the MPI implementation to use.</td>
</tr>
<tr>
<td>I2G_MPI_VERSION</td>
<td>Specifies the version of the MPI implementation specified by I2G_MPI_TYPE. If not specified the default version will be used.</td>
</tr>
<tr>
<td>I2G_MPI_PRE_RUN_HOOK</td>
<td>This variable can be set to a script which must define the pre_run_hook function.</td>
</tr>
<tr>
<td>I2G_MPI_POST_RUN_HOOK</td>
<td>This variable can be set to a script which must define the post_run_hook function.</td>
</tr>
<tr>
<td>VARIABLE</td>
<td>MEANING</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>I2G_MPI_START_VERBOSE</td>
<td>Set to 1 to turn on the additional output.</td>
</tr>
<tr>
<td>I2G_MPI_START_DEBUG</td>
<td>Set to 1 to enable debugging output</td>
</tr>
<tr>
<td>I2G_MPI_START_TRACE</td>
<td>Set to 1 to trace every operation that is performed by mpi-start</td>
</tr>
<tr>
<td>I2G_MPI_APPLICATION_STDIN</td>
<td>Standard input file to use.</td>
</tr>
<tr>
<td>I2G_MPI_APPLICATION_STDOUT</td>
<td>Standard output file to use.</td>
</tr>
<tr>
<td>I2G_MPI_APPLICATION_STDERR</td>
<td>Standard error file to use.</td>
</tr>
<tr>
<td>I2G_MPI_SINGLE_PROCESS</td>
<td>Set it to 1 to start only one process per node.</td>
</tr>
<tr>
<td>I2G_MPI_PER_NODE</td>
<td>Number of processes to start per node.</td>
</tr>
<tr>
<td>I2G_MPI_NP</td>
<td>Total number of processes to start.</td>
</tr>
</tbody>
</table>
From mpi-start 1.0.0 and onwards, you can use command line arguments

```bash
export I2G_MPI_APPLICATION=myapp
export I2G_MPI_APPLICATION_ARGS=arg1 arg2
export I2G_MPI_TYPE=openmpi
export I2G_MPI_PRE_RUN_HOOK=myhook
export I2G_MPI_START=/opt/i2g/bin/mpi-start

$I2G_MPI_START

mpi-start -t openmpi -pre myhook myapp arg1 arg2
```
Using MPI-Start (v1.x)

mpi-start [-h] [-t mpi_type] [-v] [-vv] [-vvv]
   [-pre hook] [-post hook] [-pcmd cmd]
   [-npnode n] [-pnode] [-np n]
   [-i file] [-o file] [-e file]
   [-x VAR[=VALUE]] [-d VAR=VALUE] [--]
   application [...]

Parallel job starter

optional arguments:
   -h           show this help message and exit
   -V           show mpi-start version
   -t type      use the mpi type (sets I2G_MPI_TYPE)
   -v           verbose
   -vvv          full trace
   -pre hook    use specified pre run hook script
   -post hook   use specified post run hook script
   -pcmd cmd    use specified pre command
   -npnode n    set number of processes per node
   -pnode       start only one process per node
   -np n        set total number of processes
   -c file      use file for sourcing mpi-start variables
   -i file      use file for standard input
   -o file      use file for standard output
   -e file      use file for standard error
   -x VAR[=VALUE] export the environment variable VAR,
                 optionally define value
   -d VAR=VALUE define mpi-start variable VAR with specified VALUE
   --           separator for application and arguments
PRACTICAL SESSION
Get the examples

• `wget http://devel.ifca.es/~enol/mpi-examples.tgz`
• `tar xzf mpi-examples.tgz`
Sites publish in their tags:
• MPI-START (if they have mpi-start installed)
• Then for each MPI flavour supported:
  – <MPI flavour>
  – <MPI flavour>-<MPI version>
  – <MPI flavour>-<MPI version>-<Compiler>
• And interconnect info (optional and mostly absent):
  – MPI-<interconnect>
  – Interconnects: Ethernet, Infiniband, SCI, Myrinet
Finding where to run (II)

$ lcg-info --vo ops.vo.ibergrid.eu --list-ce --query 'Tag=MPICH2'
  - CE: ce02.ific.uv.es:8443/cream-pbs-infbandShort

$ lcg-info --vo biomed --list-ce --query 'Tag=OPENMPI-1.4.4'
  - CE: ce01.kallisto.hellasgrid.gr:2119/jobmanager-pbs-biomed
  - CE: cream01.kallisto.hellasgrid.gr:8443/cream-pbs-biomed
  - CE: egeece01.ifca.es:2119/jobmanager-sge-biomed
  - CE: egeece02.ifca.es:2119/jobmanager-sge-biomed
  - CE: egeece03.ifca.es:2119/jobmanager-sge-biomed
  - CE: gridce01.ifca.es:8443/cream-sge-biomed
  - CE: gridce02.ifca.es:8443/cream-sge-biomed
  - CE: ngiescream.i3m.upv.es:8443/cream-pbs-biomed

$ lcg-info --vo biomed --list-ce --query 'Tag=*MPI*'
  - sed | wc 100
$ lcg-info --vo biomed --list-ce --query 'Tag=MPI-INFINIBAND'
  - CE: ce.reef.man.poznan.pl:2119/jobmanager-pbs-biomed
  - CE: ce002.ipp.acad.bg:2119/jobmanager-pbs-biomed
  - CE: cr1.ipp.acad.bg:8443/cream-pbs-biomed
  - CE: creamce.reef.man.poznan.pl:8443/cream-pbs-biomed

$ lcg-info --vo biomed --list-ce --query 'Tag=OPENMPI,Tag=MPI-START,Tag=MPI-ETHERNET'
  - CE: glite.univ.kiev.ua:2119/jobmanager-pbs-grid
  - CE: glite.univ.kiev.ua:8443/cream-pbs-grid
  - CE: grid-lab-ce.ii.edu.mk:2119/jobmanager-pbs-biomed
Finding where to run (IV)

• Use the JDL requirements to match to sites supporting your needs:

```plaintext
Requirements = Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment) && Member("OPENMPI", other.GlueHostApplicationSoftwareRunTimeEnvironment);
```
Site Monitoring

• Basic MPI probes are available
  – Submit job asking for 2 CPUs
  – For each MPI flavour announced by the CE
    • Compile mpi application
    • Run application
  – Check that the allocated slots is exactly 2

• Check your NGI or VO monitoring for results and availability
### Service State Information

<table>
<thead>
<tr>
<th>Current Status:</th>
<th><strong>OK</strong> (for 0d 11h 8m 42s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Information:</td>
<td>worker-node-3-2.local: OK</td>
</tr>
<tr>
<td></td>
<td>Note: If MPI-START tag was found, all supported MPI flavours by node will be tested using MPI-START. If at least one MPI supported flavour fails - overall MPI status will be 'ERROR'</td>
</tr>
<tr>
<td>Date:</td>
<td>Tue Apr 5 15:31:25 UTC 2011</td>
</tr>
<tr>
<td>DN:</td>
<td>/DC=es/DC=irisgrid/C=esga/CN=javier-lopez/CN=proxy/CN=proxy/CN=proxy/CN=proxy/CN=proxy/CN=proxy/CN=proxy/CN=proxy/CN=proxy</td>
</tr>
<tr>
<td>WN:</td>
<td>worker-node-3-2.local</td>
</tr>
<tr>
<td>TopBDII found:</td>
<td>totoBDII01.ncgr.ingrid.pt:2170</td>
</tr>
<tr>
<td>Searching for published MPI tags: MPI-START, MPICH, MPICH2, OPENMPI</td>
<td></td>
</tr>
<tr>
<td>ldapsearch -xLLL -h totoBDII01.ncgr.ingrid.pt:2170 -b o=grid '(&amp;objectClass=GlueHostApplicationSoftware)(GlueSubClusterUniqueID='be.cp.dl.uminho.pt')' GlueHostApplicationSoftwareRunTimeEnvironment</td>
<td></td>
</tr>
<tr>
<td>MPI tags were found at ce.cp.dl.uminho.pt node:</td>
<td></td>
</tr>
<tr>
<td>MPI-START</td>
<td></td>
</tr>
<tr>
<td>MPICH</td>
<td></td>
</tr>
<tr>
<td>OPENMPI</td>
<td></td>
</tr>
<tr>
<td>OPENMPI-1.2.9</td>
<td></td>
</tr>
<tr>
<td>MPI environment was found:</td>
<td></td>
</tr>
<tr>
<td>MPI_SSH_HOST_BASED_AUTH=-no</td>
<td></td>
</tr>
<tr>
<td>MPI_OPENMPI_PATH=/opt/openmpi-1.2.9</td>
<td></td>
</tr>
<tr>
<td>MPI_OPENMPI_VERSION=1.2.9</td>
<td></td>
</tr>
<tr>
<td>MPI_MPICH_VERSION=1.2.7p1</td>
<td></td>
</tr>
<tr>
<td>i2G_MPI_START=/opt/i2g/bin/mpisrta</td>
<td></td>
</tr>
<tr>
<td>MPI_MPICH_MPIEXEC=/usr/bin/mpixec</td>
<td></td>
</tr>
<tr>
<td>MPI_MPICH_PATH=/opt/mpich-1.2.7p1/</td>
<td></td>
</tr>
<tr>
<td>MPI tag: MPI-START was found</td>
<td></td>
</tr>
<tr>
<td>Searching for i2G_MPI_START...</td>
<td></td>
</tr>
<tr>
<td>i2G_MPI_START=/opt/i2g/bin/mpisrta</td>
<td></td>
</tr>
</tbody>
</table>
JobType = "Normal";
nodeNumber = 4;
Executable = "starter.sh";
Arguments = "hello.sh OPENMPI";
InputSandbox = {"starter.sh", "hello.sh"};
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
Requirements =
Member("MPI-START",
other.GlueHostApplicationSoftwareRunTimeEnvironment)
&&
Member("OPENMPI",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
#!/bin/bash

MY_EXECUTABLE=$1
MPI_FLAVOR=$2

# Convert flavor to lowercase for passing to mpi-start.
MPI_FLAVOR_LOWER=`echo $MPI_FLAVOR | tr '[[:upper:]]' '[[:lower:]]'`

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

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

# Touch the executable. It exist must for the shared file system check.
# If it does not, then mpi-start may try to distribute the executable
# when it shouldn't.
touch $MY_EXECUTABLE
chmod +x $MY_EXECUTABLE

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$MY_EXECUTABLE
export I2G_MPI_TYPE=$MPI_FLAVOR_LOWER
#export I2G_MPI_APPLICATION_ARGS="./ 0 0 1"
# optional hooks
export I2G_MPI_PRE_RUN_HOOK=hooks.sh
export I2G_MPI_POST_RUN_HOOK=hooks.sh

# If these are set then you will get more debugging information.
#export I2G_MPI_START_VERBOSE=1
#export I2G_MPI_START_TRACE=1
#export I2G_MPI_START_DEBUG=1

# Invoke mpi-start.
$I2G_MPI_START
#!/bin/bash

echo "hello world from `hostname`"
$ glite-wms-job-list-match -a job.jdl
Connecting to the service https://wms01.egee.cesga.es:7443/glite_wms_wmproxy_server

==========================================================================
COMPUTING ELEMENT IDs LIST
The following CE(s) matching your job requirements have been found:

  *CEId*
- gridce01.ifca.es:2119/jobmanager-sge-ngifor
- ngiesce.i3m.upv.es:2119/jobmanager-pbs-ngies
- ce2.egee.cesga.es:2119/jobmanager-lcgsge-GRID_ngifor

==========================================================================
$ glite-wms-job-submit -a job.jdl

Connecting to the service https://wms01.egee.cesga.es:7443/glite_wms_wmproxy_server

====================== glite-wms-job-submit Success =======================

The job has been successfully submitted to the WMProxy
Your job identifier is:

https://wms01.egee.cesga.es:9000/q5BP68lwwUH4-YFzrgHBhQ

====================== glite-wms-job-status Success =======================

BOOKKEEPING INFORMATION:

Status info for the Job : https://wms01.egee.cesga.es:9000/q5BP68lwwUH4-YFzrgHBhQ
Current Status: Running
Status Reason: Job successfully submitted to Globus
Destination: gridce01.ifca.es:2119/jobmanager-sge-ngifor
Submitted: Wed Apr 6 10:12:19 2011 CEST

==========================================================================
$ glite-wms-job-output https://wms01.egee.cesga.es:9000/q5BP681wwUH4-YFzrgHBhQ

Connecting to the service https://wms01.egee.cesga.es:7443/glite_wms_wmproxy_server

================================================================================
JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
https://wms01.egee.cesga.es:9000/q5BP681wwUH4-YFzrgHBhQ
have been successfully retrieved and stored in the directory:
/gpfs/csic_projects/grid/tmp/jobOutput/enol_q5BP681wwUH4-YFzrgHBhQ

================================================================================

$ cat /gpfs/csic_projects/grid/tmp/jobOutput/enol_q5BP681wwUH4-YFzrgHBhQ/*
hello world from gcsic142wn
hello world from gcsic142wn
hello world from gcsic142wn
hello world from gcsic142wn
parallel-job.jdl new syntax

```
JobType = "Normal";
nodeNumber = 4;
Executable = "/usr/bin/mpi-start";
Arguments = "-t openmpi hello.sh";
InputSandbox = {"hello.sh"};
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
Requirements = Member("MPI-START",
other.GlueHostApplicationSoftwareRunTimeEnvironment) 
&& Member("OPENMPI",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

No need for starter wrapper
• Errors happen frequently, we need a way to figure out what went wrong
• Use the MPI-START debugging variables:
  – Basic information:
    • I2G_MPI_START_VERBOSE=1
  – Debug information:
    • I2G_MPI_START_DEBUG=1
  – Full trace of the execution (normally too much, but useful for mpi-start developers)
    • I2G_MPI_START_TRACE=1
• Set them in the starter.sh or in the JDL:
  Environment = {“I2G_MPI_START_VERBOSE=1”,…}
Debugging output

UID = ngifor080
HOST = gcsic177wn
DATE = Tue Jul 6 12:07:15 CEST 2010
VERSION = 0.0.66

mpi-start [DEBUG ]: dump configuration
mpi-start [DEBUG ]: => I2G_MPI_APPLICATION=hello.sh
mpi-start [DEBUG ]: => I2G_MPI_APPLICATION_ARGS=
mpi-start [DEBUG ]: => I2G_MPI_TYPE=openmpi

[...]
mpi-start [INFO ]: search for scheduler
mpi-start [DEBUG ]: source /opt/i2g/bin/../../../etc/mpi-start/lsf.scheduler
mpi-start [DEBUG ]: checking for scheduler support : sge
mpi-start [DEBUG ]: checking for $PE_HOSTFILE
mpi-start [INFO ]: activate support for sge
mpi-start [DEBUG ]: convert PE_HOSTFILE into standard format
mpi-start [DEBUG ]: dump machinefile:
mpi-start [DEBUG ]: => gcsic177wn.ifca.es
mpi-start [DEBUG ]: => gcsic177wn.ifca.es
mpi-start [DEBUG ]: => gcsic177wn.ifca.es
mpi-start [DEBUG ]: => gcsic177wn.ifca.es
mpi-start [DEBUG ]: starting with 4 processes.
A real MPI job

1. Compile your application
   
   $ mpicc cpi.c -o cpi

2. Change your JDL:
   
   InputSandbox = {"starter.sh", "cpi"};
   Arguments = "cpi OPENMPI";

3. Submit
$ glite-wms-job-output https://wms01.ncg.ingrid.pt:9000/3hXhJD3eRRgiSLPqW3vSKg

Connecting to the service https://wms01.ncg.ingrid.pt:7443/glite_wms_wmproxy_server

================================================================================
JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
https://wms01.ncg.ingrid.pt:9000/3hXhJD3eRRgiSLPqW3vSKg
have been successfully retrieved and stored in the directory:
/home/enol/tests/eciencia/cpi/enol_3hXhJD3eRRgiSLPqW3vSKg

================================================================================

$ cat /home/enol/tests/eciencia/cpi/enol_3hXhJD3eRRgiSLPqW3vSKg/*
Process 3 on gcsic093wn: n=1
Process 2 on gcsic091wn: n=1
Process 1 on gcsic091wn: n=1
Process 0 on gcsic026wn: n=1
Using 16384 intervals
pi is approximately 3.1415926539002341, Error is 0.0000000003104410
wall clock time = 0.003501
Remote compilation

• What if your binary does not match the site environment?
• Remote compilation is needed
• Mpi-start hooks can help here!
Compilation using hooks

#!/bin/sh

pre_run_hook () {

  # Compile the program.
  echo "Compiling ${I2G_MPI_APPLICATION}"

  # Actually compile the program.
  cmd="mpicc ${MPI_MPICC_OPTS} -o ${I2G_MPI_APPLICATION} ${I2G_MPI_APPLICATION}.c"
  $cmd
  if [ ! $? -eq 0 ]; then
    echo "Error compiling program. Exiting..."
    return 1
  fi

  # Everything's OK.
  echo "Successfully compiled ${I2G_MPI_APPLICATION}"

  return 0
}

Compilation using hooks

```plaintext
JobType = "Normal";
nodeNumber = 4;
Executable = "starter.sh";
Arguments = "cpi OPENMPI";
InputSandbox = {"starter.sh", "cpi.c", "hooks.sh"};
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
Requirements = Member("MPI-START",
other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& Member("OPENMPI",
other.GlueHostApplicationSoftwareRunTimeEnvironment);
Environment = {"I2G_MPI_PRE_RUN_HOOK=hooks.sh"};
```
Compilation using hooks

[...]
mpi-start [DEBUG  ]: Try to run pre hooks at hooks.sh
mpi-start [DEBUG  ]: call pre_run hook
-<START PRE-RUN HOOK>---------------------------------------------------------------------
Compiling cpi
Successfully compiled cpi
-<STOP PRE-RUN HOOK>---------------------------------------------------------------------
[...]
=[START]===============================================================================
[...]
pi is approximately 3.1415926539002341, Error is 0.00000000003104410
wall clock time = 0.008393
=[FINISHED]===============================================================================

EGI UF. Vilnius, Apr 2011.
Input Preprocessing

- How to perform *serial* input processing before the actual MPI execution?
- MPI-Start hooks may be used!
  - grompp: reads a molecular topology file, checks the validity of the file, expands the topology from a molecular description to an atomic description
  - mdrun: main computational chemistry engine within GROMACS
#!/bin/sh

pre_run_hook() {
  echo "pre_run_hook called"

  # Here comes the pre-mpirun actions of gromacs
  grompp -f grompp.mdp -c conf.gro -p topol.top -o topol.tpr

  # Add to the application arguments the number of processes that we are using
  I2G_MPI_APPLICATION_ARGS="-np $I2G_MPI_NP $I2G_MPI_APPLICATION_ARGS"
  return 0
}

Arguments = "mdrun OPENMPI";
InputSandbox = {"starter.sh", "hooks.sh", "conf.gro", "topol.top", "grompp.mdp"};
Environment = {"I2G_MPI_PRE_RUN_HOOK=hooks.sh",
               "I2G_MPI_APPLICATION_ARGS=-s topol.tpr"};

... 

MPI_Comm_rank(MPI_COMM_WORLD,&myid);

... 

char filename[64];
snprintf(filename, 64, "host_%d.txt", myid);
FILE *out;
out = fopen(filename, "w+");
if (!out)
    fprintf(stderr,"Unable to open %s!\n", filename);
else
    fprintf(out, "Partial pi result: %.16f\n", mypi);

...
#!/bin/sh

export OUTPUT_PATTERN="host_*.txt"
export OUTPUT_ARCHIVE=myout.tgz

# the first parameter is the name of a host in the
copy_from_remote_node() {
    if [[ $1 == `hostname` || $1 == 'hostname -f' || $1 == "localhost" ]]; then
        echo "skip local host"
        return 1
    fi
    CMD="scp -r $1:"$PWD/$OUTPUT_PATTERN" ."
    $CMD
}

post_run_hook () {
    echo "post_run_hook called"

    if [ "x$MPI_START_SHARED_FS" == "x0" -a "x$MPI_SHARED_HOME" != "xyes" ] ; then
        echo "gather output from remote hosts"
        mpi_start_foreach_host copy_from_remote_node
    fi

    echo "pack the data"
    tar cvzf $OUTPUT_ARCHIVE $OUTPUT_PATTERN
    return 0
}
Getting output using hooks

```
JobType          = "Normal";
nodeNumber       = 4;
Executable       = "starter.sh";
Arguments        = "cpi2 OPENMPI";
InputSandbox     = {"starter.sh", "cpi2.c", "hooks.sh"};
StdOutput        = "std.out";
StdError         = "std.err";
OutputSandbox    = {"std.out","std.err"};
Requirements     = Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment) && Member("OPENMPI", other.GlueHostApplicationSoftwareRunTimeEnvironment);
Environment      = {“I2G_MPI_PRE_RUN_HOOK=hooks.sh”, “I2G_MPI_PRE_RUN_HOOK=hooks.sh”};
```
[...]  
mpi-start [DEBUG ]: Try to run post hooks at hooks.sh  
mpi-start [DEBUG ]: call post-run hook  
-<START POST-RUN HOOK>------------------------------------------------------------------------  
post_run_hook called  
pack the data  
host_0.txt  
host_1.txt  
host_2.txt  
host_3.txt  
-<STOP POST-RUN HOOK>------------------------------------------------------------------------
Hybrid MPI + OpenMP

• MPI-Start 1.x has basic support for OpenMP
• Just need to define `MPI_USE_OMP=1` and MPI-Start will define the correct environment variables for OpenMP.
• You should also start one process per host (each process will spawn the number of OpenMP threads)
export MPI_USE_OMP=1
export I2G_MPI_SINGLE_PROCESS=1

$I2G_MPI_START

mpi-start -t openmpi -d MPI_USE_OMP=1 -pnode app
More information

- **MPI-Start**: [http://devel.ifca.es/mpi-start](http://devel.ifca.es/mpi-start)
  - Documentation: [http://grid.ifca.es/wiki/Middleware/MpiStart/](http://grid.ifca.es/wiki/Middleware/MpiStart/)

- **EGI MPI-Start installation and configuration**:
  - [https://wiki.egi.eu/wiki/MAN03_MPI-Start_Installation_and_Configuration](https://wiki.egi.eu/wiki/MAN03_MPI-Start_Installation_and_Configuration)