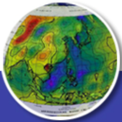
Parallel applications with gLite

Dr. Marco Fargetta

Marco.Fargetta@ct.infn.it

INFN, Italy

ACGRID-II school



Outline

MPI Overview

The MPI

- MPI (Message Passing Interface) is a library to easily create parallel applications
    - Providing a distributed memory environment for the running processes
- Two versions available for the developer
    - MPI-1 and MPI-2
    - Both used in current applications
- Each version has different implementations
    - Some implementation are hardware related
        - E.g. InfiniBand networks require MVAPICH v. 1 or 2 libraries

Goals of the MPI standards

- To provide source-code portability
- To allow efficient implementation across a range of architectures
- A great deal of functionalities
    - the user need not cope with communication failures
    - all processes can share a common workspace and/or exchange data based on `send()` and `receive()` routines

Write an MPI Application

- Any MPI application respect the following schema
  - Include the mpi header `mpi.h`
  - Start the MPI context with `MPI_Init()`
  - Insert your business code using the MPI primitive to exchange messages
    - `MPI_Recv()` and `MPI_Send()`
  - Close the MPI context with `MPI_Finalize()`
- Standard output is available only in the main node

Execute the application

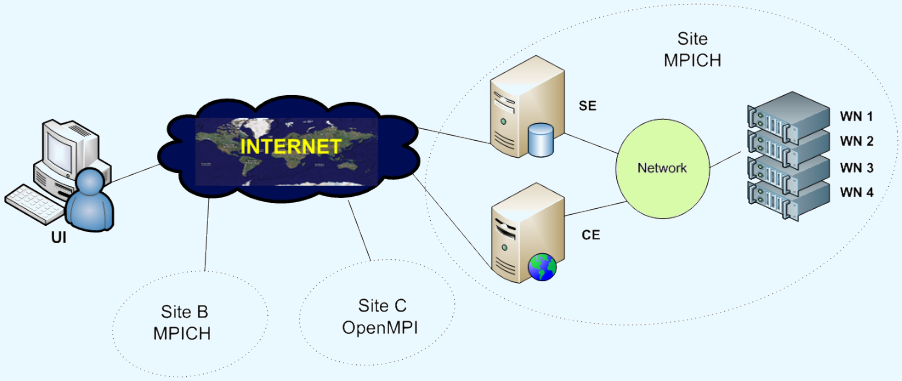
- MPI applications need to be compiled with a special compiler to include the parallelism support
  - Compiler name `mpicc`
  - Provided with the MPI implementation
- An additional application is provided to start the execution
  - The application is responsible to start the execution in all the requested nodes
  - A list of nodes for the execution should be provided

Grid and Parallel Application

gLite Grid Model

- The initial model for gLite was a super-scheduler for batch applications
  - The reference applications were almost sequential
  - Different executions could run in parallel on Grid
- MPI applications are limited to intra cluster execution
  - Using batch queues it is impossible to start all the instances together
  - The network latency among sites will reduce the performances
    - roughly move from microseconds to milliseconds when the communication go outside the cluster

Resources for parallelism



MPI execution in gLite

The MPI library

- The site has to provide the MPI library needed for your application
  - The application needs exactly the same library used in compile time
- Each site has to publish the list of library available between the tags

Check the Tags

```
$ lcg-info --vo gilda --list-ce --attrs 'Tag'
```

Check the CE supporting the library

```
$ lcg-info --vo gilda --list-ce --query 'Tag=<library name>'
```

Using the resources

- To use the resources for MPI the JDL should include

The MPI library requirements

```
Requirements =  Member("MPI-START",
        other.GlueHostApplicationSoftwareRunTimeEnvironment)
  && Member("MPICH2",
        other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

The amount of resources needed

```
CPUNumber = <Number of requested cores>;
```

Run the application: `MPI-START`

- The application needs the MPI commands to start
  - You have to provide a script running the command opportunely
- The binary code should be in all the machine
- The path to the correct library has to be provided
- Fortunately, EGEE provides a set of script performing all the set-up: **MPI-START scripts**

Run the application: `MPI-START`

- MPI-START should be installed on the site
  - You have to include the Tag `MPI-START` among the requirements
- To run the script an additional wrapper script is required by the user
  - A template is provided

JDL for running the wrapper

```
CPUNumber       = <Number of CPU cores>;
Executable      = "<wrapper file>" ;
Arguments       = "<application>" "<MPI Library>";
InputSandbox    = {"<wrapper file>","<hooks script>","<application>"};
```

The hooks

- Sometime users need to do some activity before or after running the application
  - EGEE documentation suggests to compile the application on the WN before the execution (**:-o**)
- The wrapper file has to define two variables for the hook
  - `I2G_MPI_PRE_RUN_HOOK` and `I2G_MPI_POST_RUN_HOOK`
- The hook script should define two functions
  - `pre_run_hook` run before the application
  - `post_run_hook` run after the application

Look at scripts

- You'll find the scripts in the agenda

Job without `MPI-START`

- Several sites do not support `MPI-START` but you can still submit MPI applications
- There are two options:
  1. `MPI-START` is a set of script. You can send with your job and manually run
  2. Create a script running your job
     - The location of library has to be retrieved by the script
     - The hosts list is provided by the scheduler but the format will depend on the version
     - You have to find a way to transfer files among the hosts

Questions!

Exercise

- Use the scripts to submit a parallel application in `glite-tutor`
- There are example applications in `/opt/mpich-1.2.7p1/examples/`

References

- EGEE Mpi guide
  http://egee-uig.web.cern.ch/egee-uig/production_pages/MPIJobs.html
- EGEE MPI WG
  http://egee-intranet.web.cern.ch/egee-intranet/NA1/TCG/wgs/mpi.htm
- MPI-START Documentation
  http://www.hlrs.de/organization/av/amt/research/mpi-start/mpi-start-documentation/
- Site config for MPI
  https://twiki.cern.ch/twiki/bin/view/EGEE/MpiTools