Parallel computing on the Grid

CCR-InfnGrid Workshop 2011

Roberto Alfieri
Parma University & INFN

Outline

- MPI support in Grid (gLite/EMI)
- Forthcoming “Granularity” attributes
- Parallel Job types (not only MPI..)
- The National parallel cluster for theoretical physics (CSN4cluster)
- Conclusions and future works

Current
MPI support in gLite

It comes from EGEE MPI-WG (2007-08, recommendations: http://www.grid.ie/mpi/wiki/) and is based on **MpiStart** which is a set of scripts that ease the execution of MPI programs by using a unique and stable interface to the middleware.

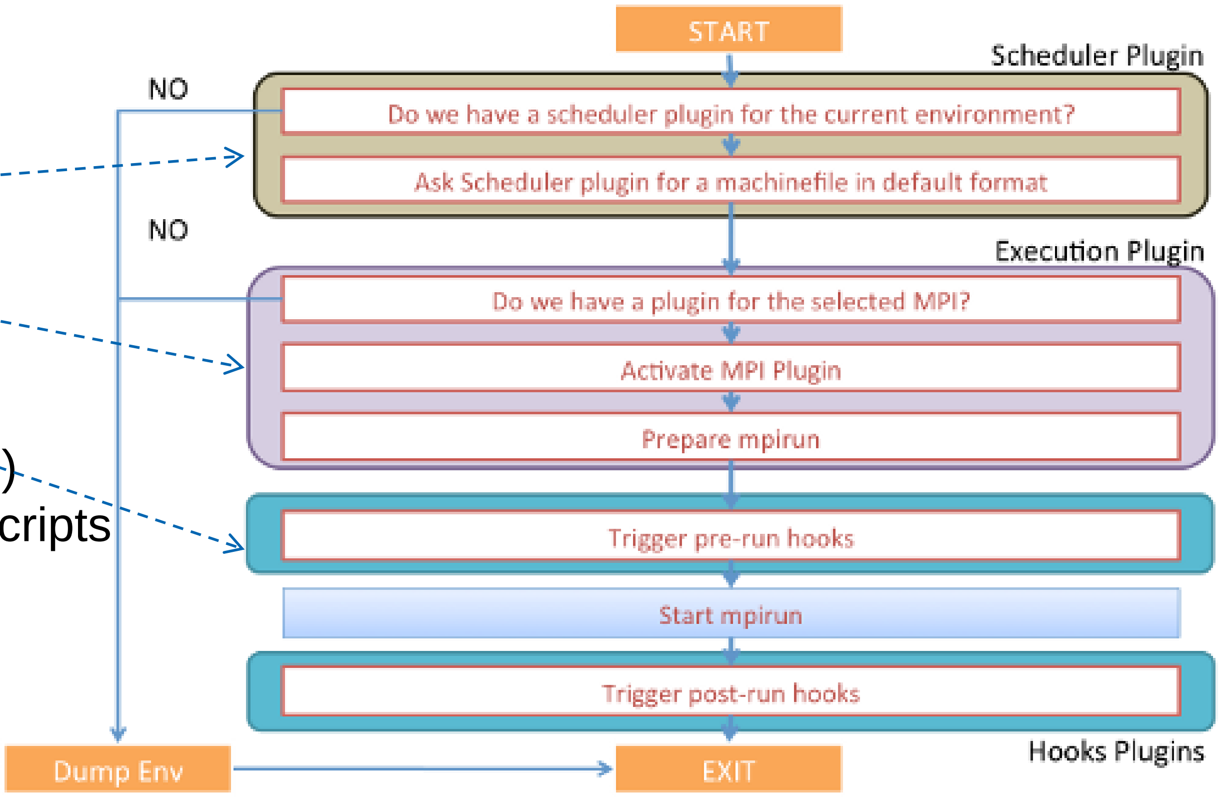
Mpi-start frameworks:

1) **Batch scheduler**
   - PBS, LSF, SGE, ..

2) **MPI implementations**
   - openMPI, MpiCh2, ..

3) **Workflow control**
   - Files distribution (if needed)
   - user's pre/post execution scripts

EMI maintainer web page:
https://devel.ifca.es/mpi-start



START
Scheduler Plugin
NO
Do we have a scheduler plugin for the current environment?
Ask Scheduler plugin for a machinefile in default format
NO
Execution Plugin
Do we have a plugin for the selected MPI?
Activate MPI Plugin
Prepare mpirun
Trigger pre-run hooks
Start mpirun
Trigger post-run hooks
Dump Env
EXIT
Hooks Plugins

MPI-start use example

#mpi-start-wrapper.jdl (example)

```
JobType="Normal";
CPUNumber=4;
Executable="mpi-start-wrapper.sh";
Arguments=“my-mpi-prog OPENMPI";
StdOutput="std.out";
StdError="std.err";
InputSandbox={"my-mpi-prog.c","mpi-start-wrapper.sh","mpi-hooks.sh"};
```

#mpi-start-wrapper.sh (simplified version)

```
export I2G_MPI_APPLICATION=‘pwd‘/$1
export I2G_MPI_TYPE=$2
export I2G_MPI_PRE_RUN_HOOK=mpi-hooks.sh
export I2G_MPI_POST_RUN_HOOK=mpi-hooks.sh
/opt/i2g/bin/mpi-start   #Invoke mpi-start
```

#mpi-hooks.sh (example)

```
pre_run_hook()
{ mpicc -o ${I2G_MPI_APPLICATION} ${I2G_MPI_APPLICATION}.c }

post_run_hook()
{ mail user@domain -s "${I2G_MPI_APPLICATION} completed" < std.out }
```

Forthcoming

MPI and multi-thread support in EMI

Despite the work of the MPI-WG, MPI in grid was scarcely used in 2009.

In 2009 EGEE-III designated a **new MPI-WG**. Purpose:
- Investigate why Mpi isn't used and provide new enforced recommendations
- Provide a solution for the support of the upcoming multicore architectures

Recommendation document released in 06/2010:
http://www.grid.ie/mpi/wiki/WorkingGroup

Main recommendations:

-- MPI-start is confirmed as submission method
-- multiple MPI flavours distribution and support
-- Shared file-system and Ssh password-less among WNs
-- Functionality tests (SAM) , documentation and training
New JDL attributes («granularity») are introduced to support multicore architectures

"Granularity" attributes

Semantics and syntax

Attribute	Meaning
CPUNumber=P	Total number of required CPUs
SMPGranularity=C	Minimum number of cores per node
HostNumber=N	Total number of required nodes
WholeNodes=true	Reserve the whole node (all cores)

New JDL attributes

```
CPUNumber = 64;         # 32 nodes, with 2 CPUs per node
SMPGranularity = 2;     # (SMPsize >=2 )

CPUNumber  = 16;        # 2 nodes, with 8 CPUs per node
HostNumber = 2;         # (SMPsize >=8 )

WholeNodes=true;        # 2 whole nodes with SMPsize>=8
HostNumber=2;
SMPGranularity=8;

WholeNodes=true;        # 1 whole node with SMPsize>=8
SMPGranularity=8;       # (default HostNumber=1)
```

Examples

"Granularity" attributes

Preliminary patch

The New JDL attributes proposed by the WG **aren't implemented in gLite yet**

- CE support is coming with Cream-CE 1.7 (EMI-1 release)

A **preliminary patch for Cream-CE** has been developed and tested in collaboration with the gLite middleware developers (October 2010)

It comes with a **different syntax but the same semantics**.

Examples:

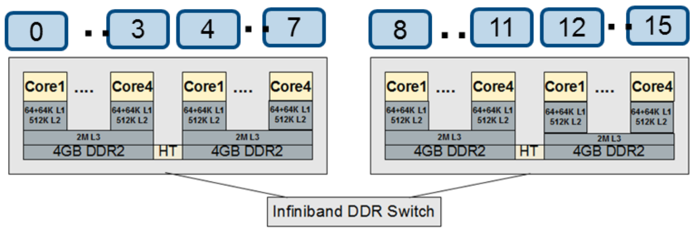
```
CeRequirements = “wholenodes=\”true\”  && hostnumber==2”; # 2 whole nodes

CPUNumber      = 16;                          # 8 nodes with 2 CPUs per node
CeRequirements = “SMPGranularity==2”
```

The patch is now **installed and working at INFN Pisa (LSF)** and **Parma (PBS)**

"Granularity" attributes

Parallel job types

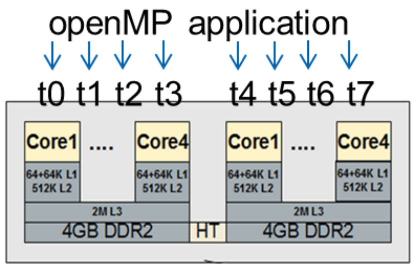
- Pure MPI jobs



- Multi-thread jobs

-- require a single "**Wholenodes**" with C cores

-- start C threads

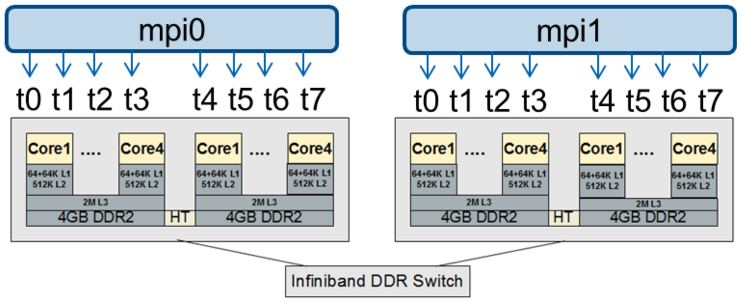


- Hybrid MPI-openMP jobs

-- require N **"Wholenodes"** with C cores each

-- start 1 MPI ranks per node

-- each MPI rank will spawn C threads

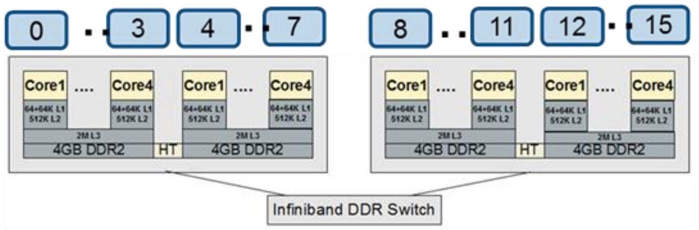


MPI jobs

MPI-start is the submission method supported by gLite/EMI.

This example executes 16 MPI ranks (2 whole nodes):

```
# mpi-start-wrapper.jdl
# New EMI syntax
WholeNodes=true;
Hostnumber=2;
SMPgranularity=8;
Executable = "mpi-start-wrapper.sh";
Arguments = “my-mpi  OPENMPI";
InputSandbox = {"mpi-start-wrapper.sh","mpi-hooks.sh",“my-mpi.c"};
#
# Temporary syntax (gLite with Granularity patch)
# CeRequirements = "wholenodes==\"true\" && hostnumber==2";
# Requirements =(other.GlueCEInfoHostName == "gridce3.pi.infn.it");
```

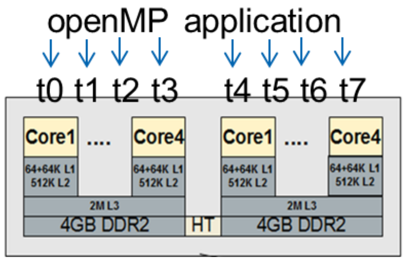


multi-thread jobs

Wholenodes attribute allows the submission of **multi-thread jobs** (and the **exclusive use of the node memory**)

This example executes 8 openMP threads on a whole node:

```
# my-openmp.jdl
#
# New EMI syntax
WholeNodes=true;
SMPgranularity=8;
Executable = “my-openmp.sh";
InputSandbox = {“my-openmp.sh",“my-openmp.c"};
#
# Temporary syntax (gLite with “granularity” patch)
# CeRequirements = "wholenodes==\"true\"  && hostnumber==1";
# Requirements =(other.GlueCEInfoHostName == "gridce3.pi.infn.it");
```



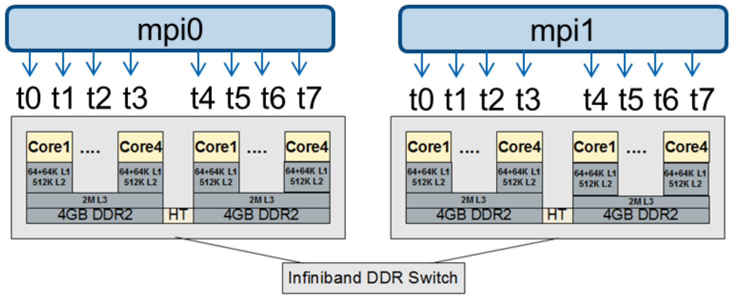
EMI syntax

Hybrid MPI-openMP jobs

Hybrid MPI-openMP programming will be supported by mpi-start in EMI-1 release.

This example requires 2 MPI ranks with 8 openMP threads each:

```
# mpi-start-wrapper.jdl
WholeNodes=true;
Hostnumber=2;
SMPgranularity=8;
Executable = "mpi-start-wrapper.sh";
Arguments = “my-hybrid OPENMPI";
...
```



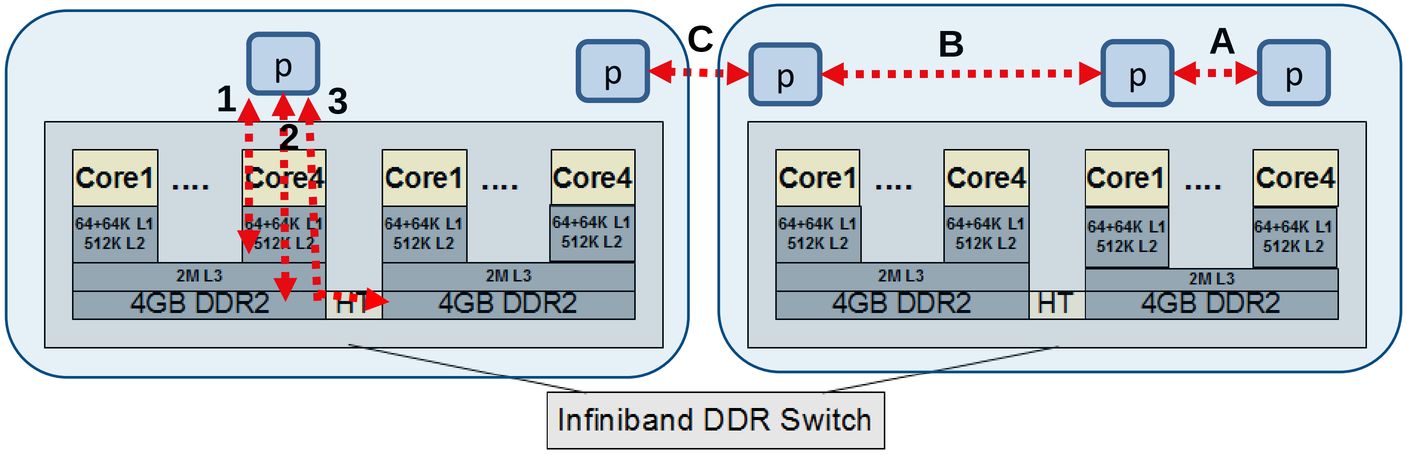
To enable the support of openMP in mpi-start set `MPI_USE_OMP=1`

The env. variable **OMP_NUM_THREADS** will be set by mpi-start on each host with the number of slots allocated per host, in order to start the right number of threads.

memory architecture

In modern multicore processors the memory architecture is **NUMA**

- Cpu/memory **affinity** is the ability to bind a process to a specific CPU/memory bank -



Memory theoretical peak performance:

	Memory Type	Latency	Bandw.
1	L3 cache	≈35 ns	
2	RAM	≈ 50 ns	≈30 GB/s
3	Numa (HT or QPI)	≈ 90 ns	≈10 GB/s

Communication perf. (using NetPIPE on CSN4cluster):

	Comm. Type	Latency	Bandw.
A	Shm (intra-socket)	640 ns	14 GBytes/s
B	Shm (inter-socket)	820 ns	12 GBytes/s
C	infiniband	3300 ns	11 GBytes/s

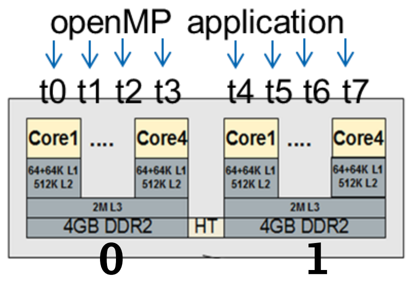
Affinity Control: Numactl library

In general CPU/memory Affinity can be controlled at command line level or code level, using a specific library such as **"Numactl"**.

```
numactl --cpubind=0 --membind=0 ./my_progr  #Command line example
```

In case of **multi-thread job** the affinity control is performed at code level. Example:

```
main() {
#pragma omp parallel
  {
  if (omp_get_thread_num()/4)
        numa_run_on_node(1);    // run on socket 1
   else numa_run_on_node(0);    // run on socket 0
  ...
  }
}
```



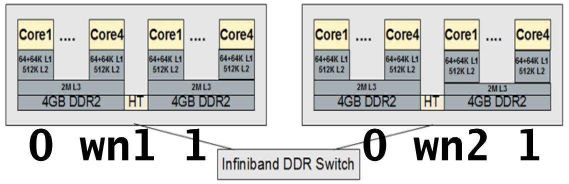
Affinity Control with MPI jobs

CPU affinity is supported by the principal MPI implementations.

OpenMPI supports CPU affinity at command line level by means of the Rankfile.

Example:

```
mpirun --rankfile rank.txt ./my_mpi_appl
```



A beta version of mpi-start supporting the affinity has been developed and is currently under test.

Possible mpi-start syntax example:

```
MPI_USE_OMP=1
MPI_USE_AFFINITY=1
#pcore      #1 MPI process per core
pnode       #1 MPI process per node
#psocket    #1 MPI process per socket
```

Rank_pcore

```
rank  0=wn1 slot=0:0
rank  1=wn1 slot=0:1
rank  2=wn1 slot=0:2
rank  3=wn1 slot=0:3
rank  4=wn1 slot=1:0
rank  5=wn1 slot=1:1
rank  6=wn1 slot=1:2
rank  7=wn1 slot=1:3
rank  8=wn2 slot=0:0
rank  9=wn2 slot=0:1
rank 10=wn2 slot=0:2
rank 11=wn2 slot=0:3
rank 12=wn2 slot=1:0
rank 13=wn2 slot=1:1
rank 14=wn2 slot=1:2
rank 15=wn2 slot=1:3
```

Rank_psocket

```
rank  0=wn1 slot=0:0-3
rank  1=wn1 slot=1:0-3
rank  2=wn2 slot=0:0-3
rank  3=wn2 slot=1:0-3
```

Rank_pnode

```
rank  0=wn1 slot=0-7
rank  1=wn2 slot=0-7
```

CSN4cluster

CSN4Cluster is the centralized facility for parallel and serial computations reserved for the theoretical physics community (Gruppo IV).

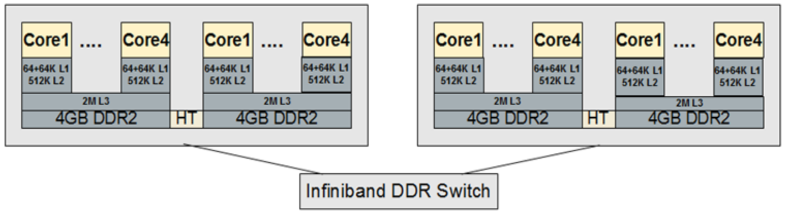
The cluster is installed, configured and maintained by the **INFN-Pisa staff.**

Computing:

128 WNs Opteron 2x4 cores, SL5/x86_64, openMPI

1024 total cores, 10TFlops peak perf

1 CE gridce3.pi.infn.it : Cream-CE, LSF



High Speed Network:

Infiniband DDR

Storage:

Shared home among WNs, GPFS/Infiniband

1 SE gridsrm.pi.infn.it : Storm, GPFS/Infiniband

Details: http://wiki.infn.it/cn/csn4/calcolo/csn4cluster/home

queues organization

Access method: via Grid only

The direct job submission to the CE

```
Requirements = (other.GlueCEInfoHostName == "gridce3.pi.infn.it")
```

gives access to 2 queues:

- **theompi** : parallel job only, runtime 72h, reservation time 8h , Role=parallel is required

  ```
  voms-proxy-init –voms theophys:/theophys/Role=parallel
  ```

- **theoshort**: short jobs, runtime 4h, Role=parallel will not be specified

  ```
  voms-proxy-init –voms theophys
  ```

The “short” queue allows the exploitation of cores when they are unused by parallel jobs using two scheduling techniques:
- **Slots reservation** ensures a lock for parallel jobs on the free job slots
- Backfill allows short Jobs to use slots reserved for parallel Jobs

Parallel jobs on PBS clusters

The **Maui Job Scheduler** currently distributed in gLite (maui-3.2.6p1-x86_64) doesn't work properly when the job requires multiple processors (-l nodes=x:ppn=y)

A working Maui release (maui-3.3.1-1.sl5.x86_64) is installed and under test at Parma site. The correct Maui release will be hopefully included soon in the EMI distribution.

To access the Parma parallel queue **`cream-ce.pr.infn.it:8443/cream-pbs-parallel`** the user must include **`Role=parallel`** in the VOMS proxy:

```
voms-proxy-init –voms <vo-name>:/<vo-name>/Role=parallel
```

This is the same access policy used to access the CSN4cluster parallel queue

Conclusions

At present we have parallel clusters (both LSF and PBS) in Grid working with a preliminary support of the “granularity” attributes.

Open Issues:

- “Granularity” support as first level JDL attributes -> EMI-1
- openMP support in MPI-start -> EMI-1
- CPU affinity support in MPI-start -> EMI work in progress
- Maui problem with parallel jobs -> EMI work in progress

EMI will provide in the **near future** an extended and stable support for parallel jobs.

An **cooperation among interested VOs** is desirable in order to define common basis for parallel clusters **configuration, policy and usage.**

Thank you

for your attention!