Future of Grid parallel exploitation

Roberto Alfieri - Parma University & INFN – Italy

Outline

MPI support in the current grid middleware (gLite)

MPI and multi-thread support in the forthcoming EMI middleware

- *Wholenodes reservation support (new JDL attributes)*
- *CPU affinity and hybrid programming support (new mpi-start)*
- *Use-cases analysis*

Status of parallel clusters in grid

supporting a preliminary version of the new features

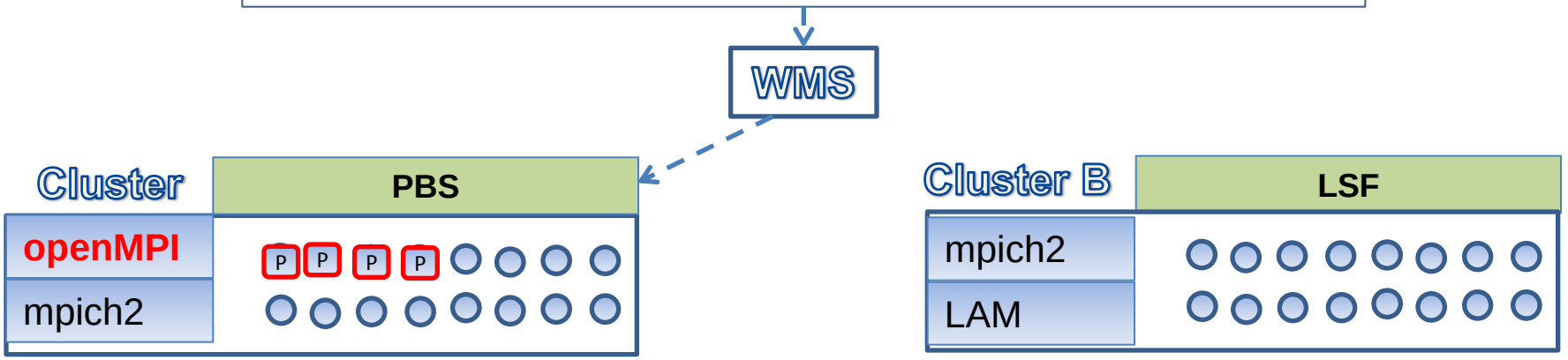
Conclusions

Current MPI support in Grid (gLite middleware)

Requirements	Solution	Description
- Multiple CPU allocation	JDL attribute	CPUnumber=4
- Files distribution among nodes **- Multiple MPI version/flavour Support** **- Get the MPI machine-file from the job manager** **- Pre/Post Execution scripts**	MPI-start	- The user has to specify MPI flavour and pre/post hook scripts - The other information are detected by MPI-start from the cluster environment

JDL file Example

```
CPUnumber = 4;
Executable = "mpi-start-wrapper.sh";
Arguments = "my-mpi-prog OPENMPI";
InputSandbox = "mpi-start-wrapper.sh  mpi-hooks.sh  my-mpi-prog";
....
```

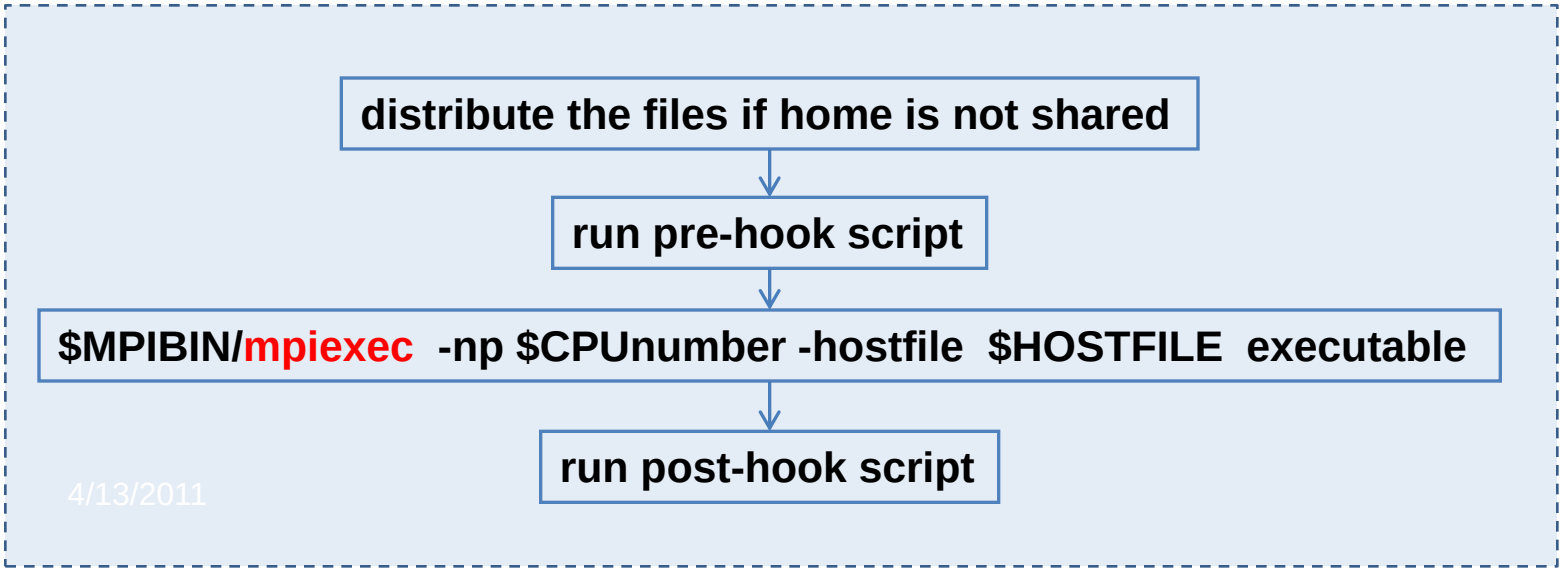


MPI-start

MPI-start (originally developed by HLRS – Stuttgart) is a set of scripts that ease the execution of MPI programs by using a unique and flexible interface to the resources.

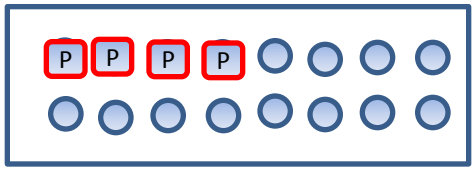
The adoption of **MPI-start** in gLite comes from the EGEE MPI-WG recommendations *http://www.grid.ie/mpi/wiki/ (2007-2008)*

MPI-start is a wrapper for mpiexec:

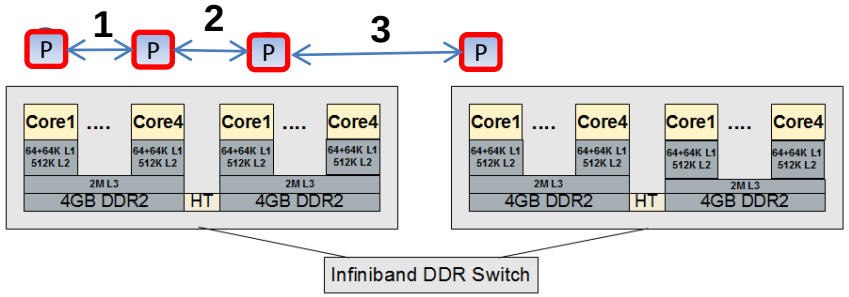


Communication types in modern Worker Nodes

Cluster model in the current Grid middleware



In modern clusters the processors are multicore, the memory access is **NUMA** and we have at least 3 communication types.



Measured network performance (on CSN4cluster, using NetPIPE):

	Comm Type	Comm. device	Latency	MAX Bandw.
1	Intra-socket	SHared Mem - L3 Cache or DDR	640 ns (DDR)	14 GBytes/s
2	Intra-node	SHared Mem - NUMA (HT or QPI)	820 ns	12 GBytes/s
3	Extra-node	Infiniband	3300 ns	11 GBytes/s

The Grid middleware should support new features for parallel clusters:

-- **multi-threaded parallelism** (to exploit the SHM comm. model in multi-core architectures)

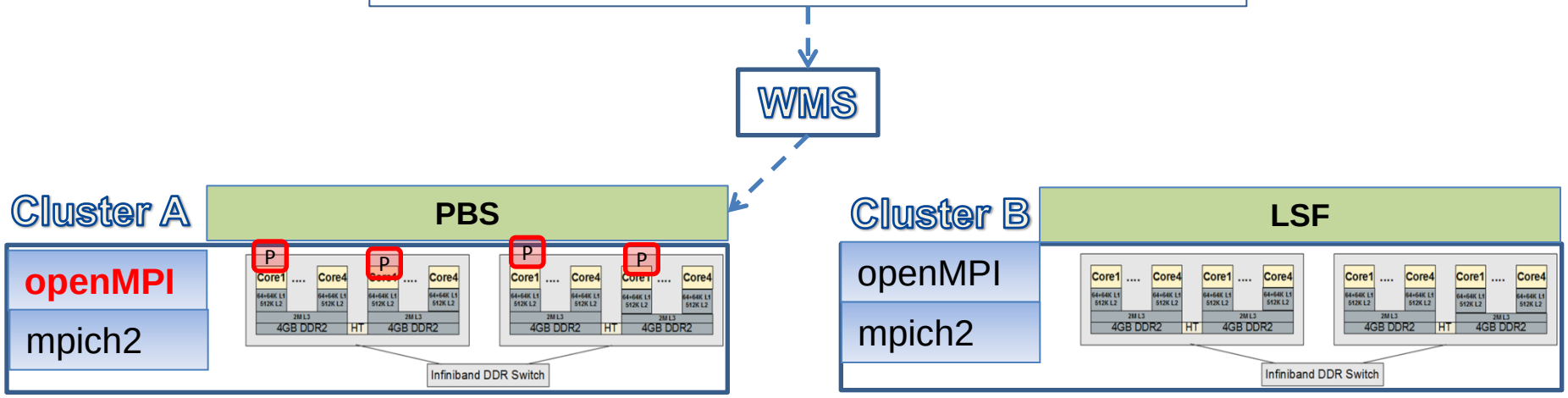
-- **CPU/memory affinity** (ability to bind a process to a specific core or memory bank, needed to exploit the cache effect and avoid the NUMA bottleneck)

Forthcoming parallelism support (EMI middleware)

New Requirements	Solution
- Multiple CPU allocation with granularity selection supporting multi-threaded applications	New JDL attributes
- CPU / memory affinity control - openMP support (Hybrid programming)	Enhanced mpi-start

JDL file example

```
#CPUnumber=4;
Wholenodes=true;
HostNumber=2;
SMPGranularity=8;
Executable= "mpi-start";
Arguments="-t openmpi -psoket -- my-prog";
InputSandbox="my-prog";
```



New JDL attributes

In 2009 EGEE designated a new MPI-WG.
Purpose: to provide a solution for the support of the upcoming multicore architectures.
The recommendation document, http://www.grid.ie/mpi/wiki/WorkingGroup released in 06/2010, proposes the following new attributes:

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

Examples

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

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)
```

Note:
CPUnumber and Wholenodes are alternative to each other

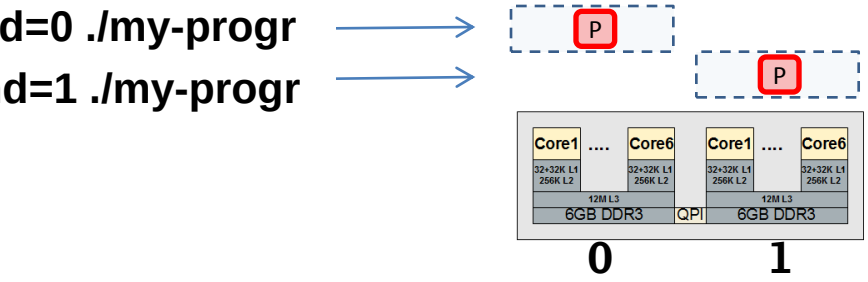
CPU Affinity control

AFFINITY LIBRARIES

CPU/memory Affinity can be controlled through a specific library such as "**Numactl"**, or the "**Sched Affinity**" provided by GNU.

Examples: **numactl --cpubind=0 ./my-progr**

numactl --cpubind=1 ./my-progr



AFFINITY CONTROL IN MPI

Affinity is supported by the principal **MPI** implementations.

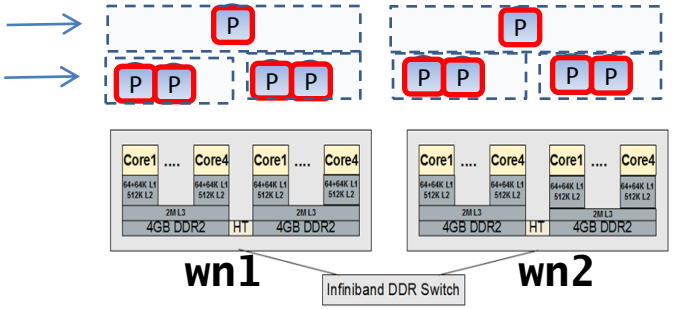
OpenMPI provides a set of command line options such as:

--pernode, --npersocket, --npernode, --rankfile , etc.

Examples:

mpiexec --pernode -host wn1,wn2 my-mpi-prog

mpiexec --npersocket 2 -host wn1,wn2 my-mpi-prog



New MPI-start

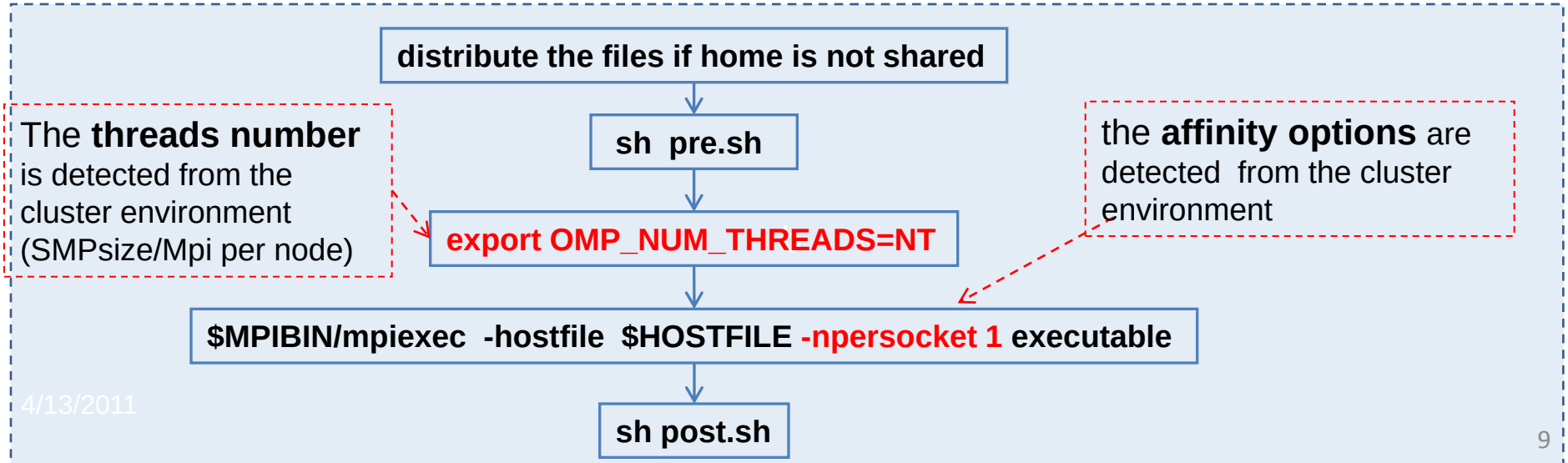
The new Mpi-start tool included in EMI is maintained by Enol Fernandez (https://devel.ifca.es/mpi-start).

It is backward compatible, but supports a new syntax with command line options (still under development)

Example:

mpi-start -t openmpi -pre pre.sh -post post.sh -psocket -- executable";

Main Options	meaning
-t type	Select the Mpi flavor
-pre HOOK -post HOOK	pre/post run hook script
Affinity/OMP Options	
-pcore|-psocket|-pnode	Start 1 process per core|socket|node
-npnode N	Start N processes per node
-npsocket N	Start N processes per socket

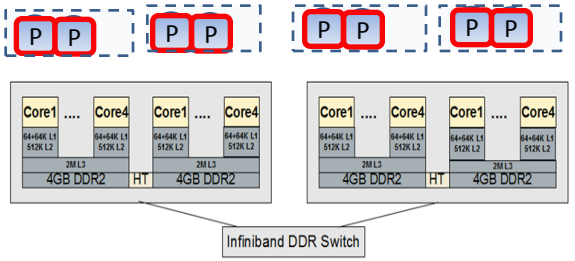


Use case: 2GB per MPI process

APPLICATION NEEDS	8 MPI processes, 2GB per MPI process
RESOURCES	2 sockets per node, 4 cores per socket , 1 GB per core

starter.jdl

```
Executable = “starter.sh";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {“starter.sh"};
OutputSandbox = {"std.err","std.out"};
WholeNodes = "true"
HostNumber = “2"
SMPGranularity = "8"
```



starter.sh

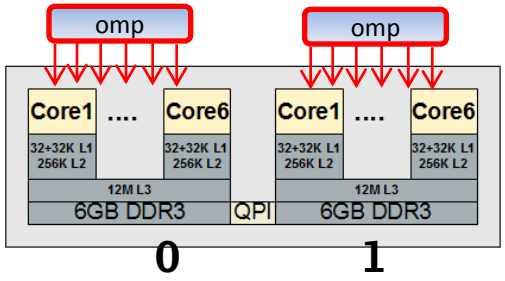
```
#!/bin/bash
lcg-cp -v  lfn:/grid/theophys/my-prog  file://$(pwd)/my-mpi-prog
lcg-cp -v  lfn:/grid/theophys/file.in      file://$(pwd)/file.in
mpi-start -npsocket  2 --t openmpi  --  my-mpi-prog < file.in > file.out
lcg-cr  -v  -l lfn:/grid/theophys/file.out  file://$(pwd)/file.out
```

Use case: 1 openMP program per socket

APPLICATION NEEDS	1 openMP program per socket, 1 thread per socket core
RESOURCES	2 sockets per node, 6 cores per socket

starter.jdl

```
Executable = “starter.sh";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {“starter.sh“,”my-omp-appl”};
OutputSandbox = {"std.err","std.out"};
WholeNodes = "true"
HostNumber = “1"
SMPGranularity = “12"
```



starter.sh

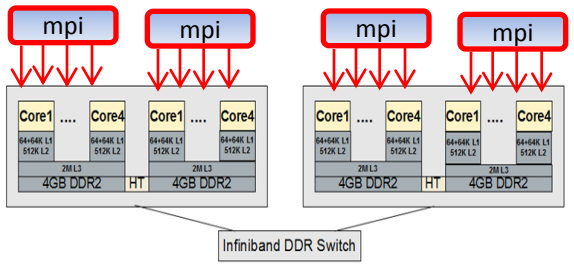
```
#!/bin/bash

OMP_NUM_THREADS=6  numactl --cpubind=0  ./my-omp-appl < file0.in > file0.out &
OMP_NUM_THREADS=6  numactl --cpubind=1  ./my-omp-appl < file1.in > file1.out &
```

Use case:1 MPI per socket, 1 thread per core

APPLICATION NEEDS	4 hybrid processes, 1 MPI per socket, 1 thread per socket core
RESOURCES	2 socket per node, 4 cores per socket

starter.jdl

```
Executable = “starter.sh";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {“starter.sh"};
OutputSandbox = {"std.err","std.out"};
WholeNodes = "true"
HostNumber = “2"
SMPGranularity = "8"
```



starter.sh

```
#!/bin/bash
lcg-cp -v  lfn:/grid/theophys/my-prog  file://$(pwd)/my-hybrid-prog
lcg-cp -v  lfn:/grid/theophys/file.in      file://$(pwd)/file.in
mpi-start  -psocket  -t openmpi --  my-hybrid-prog < file.in > file.out
lcg-cr  -v  -l lfn:/grid/theophys/file.out  file://$(pwd)/file.out
```

Parallel clusters in Grid

In collaboration with the developers of **Cream-CE** (M. Sgaravatto team, summer 2010) and **MPI-start** (E. Fernandez, still on going) a **preliminary support to the new MPI features** has been released and installed at 2 INFN sites:

INFN-Pisa : CSN4Cluster is the centralized facility for parallel and serial computations for the theoretical physics community (Gruppo IV)

- Resources: 128x8 cores (10 TFlops peak perf.), Infiniband, LSF, openMPI
- Official inauguration : 13/04/2011
- Wiki site: http://wiki.infn.it/cn/csn4/calcolo/csn4cluster/home

INFN-Parma : 8x8 cores, PBS, openMPI

- 2 Virtual Organizations (**theophys** and **comput-er.it**) are currently running parallel applications (lattice field theory, relativistic astrophysics, molecular dynamics, electronic-structure calculations) on the clusters

Parallel queues access and organization

Resource access:

Parallel queues are special Grid resources.
To **prevent the access from generic serial jobs** we have introduced a special Role (role=parallel) which is granted to parallel job users.

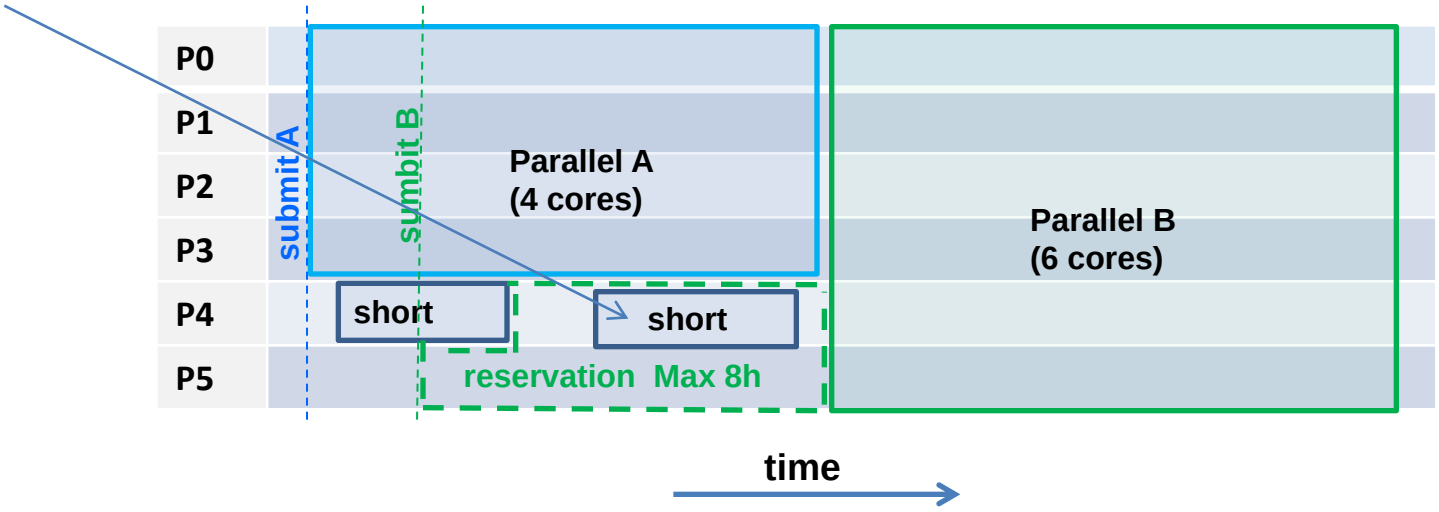
voms-proxy-init –voms theophys:/theophys/Role=parallel

Queues organization:

At Pisa site 2 queues have been created on the same pool of WNs:

- **parallel** : parallel job only, runtime 72h, **Reservation time** max 8h on the free job slots
- **short** : short jobs, runtime 4h (shorter than the reservation time)

- The "short" queue allows the exploitation of cores when they are unused by parallel jobs
- **Backfill** scheduling technique enables short Jobs to use slots reserved for parallel Jobs



Conclusions and future work

At present we have parallel clusters (both LSF and PBS) in Grid working with a preliminary support of the “wholenodes” attributes and a preliminary version of the new MPI-start tool.

EMI will provide in the **near future** an extended and stable support for parallel jobs (pure MPI, multi-threaded and hybrid).

A **collaboration among interested VOs** (so far “theophys” and “comput-er.it”) is on going aiming at the definition of common basis for parallel clusters **configuration, usage and future developments in Grid.**

Thank you
for your attention!