Interactive European Grid

I2G CrossBroker

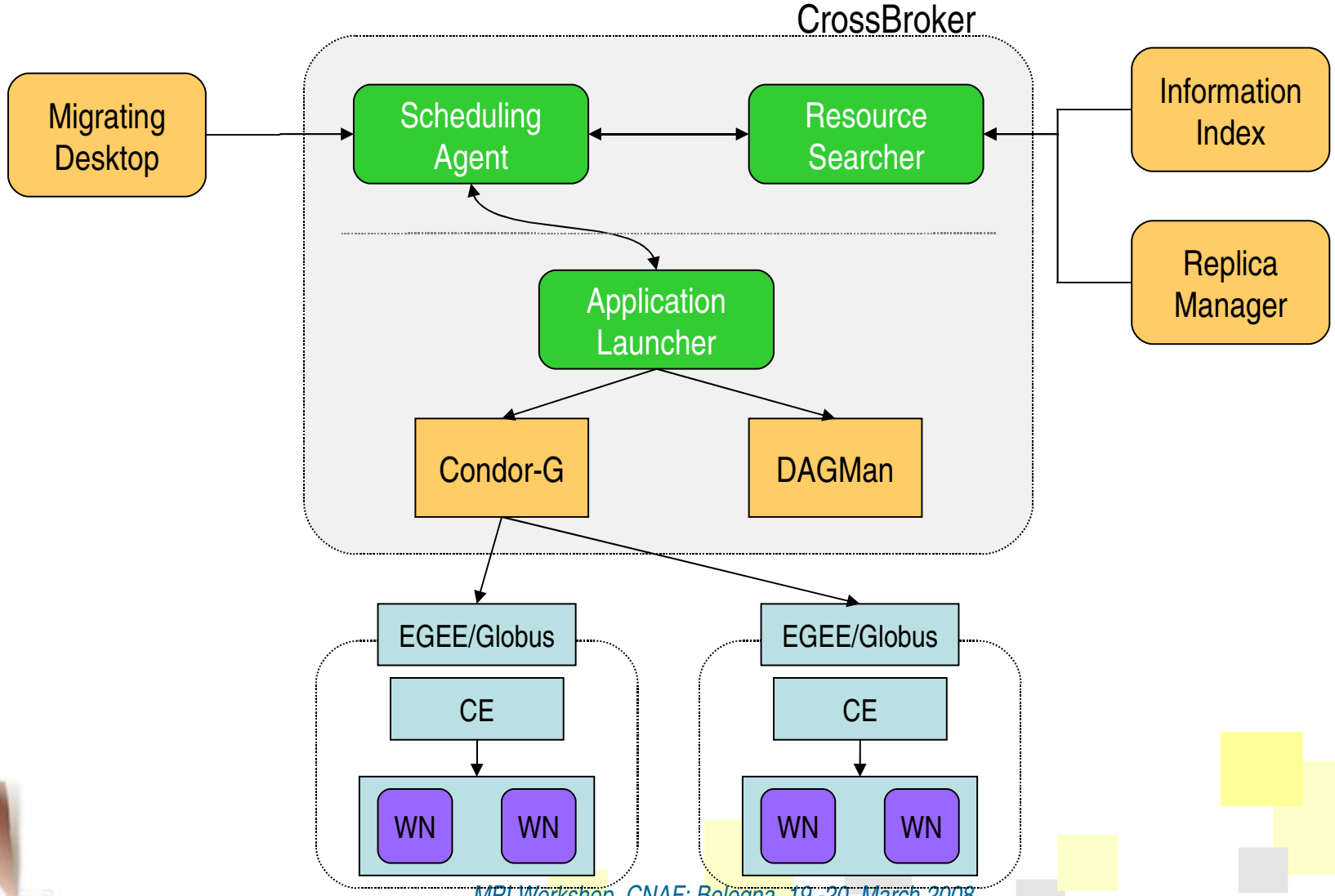
Enol Fernandez

MPI Workshop, Bologna, 19.-20. March 2008

CrossBroker

- CrossBroker does automatic scheduling in Grid Environments
  - Resource discovery
  - Resource selection
  - Job execution
- Jobs not treated by gLite:
  - parallel jobs (MPI)
    - Run in more than one resource, in a coordinated fashion.
  - Interactive jobs
    - The user interacts with the application during its execution

Architecture



Architecture

- Scheduling Agent
  - Receives each job and keeps it in a persistent queue
  - Contacts Resource Searcher and gets a list of available resources
  - Selects resources and passes them to Application Launcher
- Resource Searcher
  - Given a job description (JobAd), performs the matchmaking between job needs and available resources.
  - Uses the Condor ClassAd library, originally designed for matches of a single job with a single resource.
  - A set matching has been developed to support matches of a single job to a group of resources.
- Application Launcher
  - Responsible for providing a reliable submission service of parallel applications on the Grid.
  - Responsible for file staging at the remote site (executable and input/output files)
  - Uses the services of Condor-G

Parallel Job Support

- Support for parallel jobs:
  - Open MPI
  - PACX-MPI
  - MPICH-P4
  - MPICH-G2
  - Plain (just the machines)
- Takes into account sites capabilites.
- Ability to define starter scripts/process to start the parallel job
  - MPI-Start is configured automatically and used by default.

Parallel Job Support

- Changes in JDL
  - JOBTYPE:
    - Normal: sequential jobs, just one CPU
    - Parallel: more than one CPU
  - SUBJOBTYPE:
    - openmpi
    - pacx-mpi
    - mpich
    - mpich-g2
    - plain
  - JOBSTARTER (if not defined, mpi-start)
  - JOBSTARTERARGUMENTS

Parallel Job Support

```
Type = "Job";
VirtualOrganisation = "imain";
JobType = "Parallel";
SubJobType = “openmpi";
NodeNumber =  5;
Executable = "test-app";
Arguments = "-v";
InputSandbox = {"test-app", "inputfile"};
OutputSanbox = {"std.out", "std.err"};
StdErr = "std.err“;
StdOutput = "std.out";
Rank = other.GlueHostBenchmarkSI00 ;
Requirements =
   other.GlueCEStateStatus == "Production";
```

Parallel Job Support

```
Type = "Job";
VirtualOrganisation = "imain";
JobType = "Parallel";
SubJobType = "plain";
NodeNumber =  10;
JobStarter = "my-starter.sh"
JobStarterArguments = "-procs 10"
Executable = "test-app";
Arguments = "-v";
InputSandbox = {"test-app", "my-starter.sh"};
OutputSanbox = {"std.out", "std.err"};
StdErr = "std.err";
StdOutput = "std.out";
Rank = other.GlueHostBenchmarkSI00 ;
Requirements =
   other.GlueCEStateStatus == "Production";
```

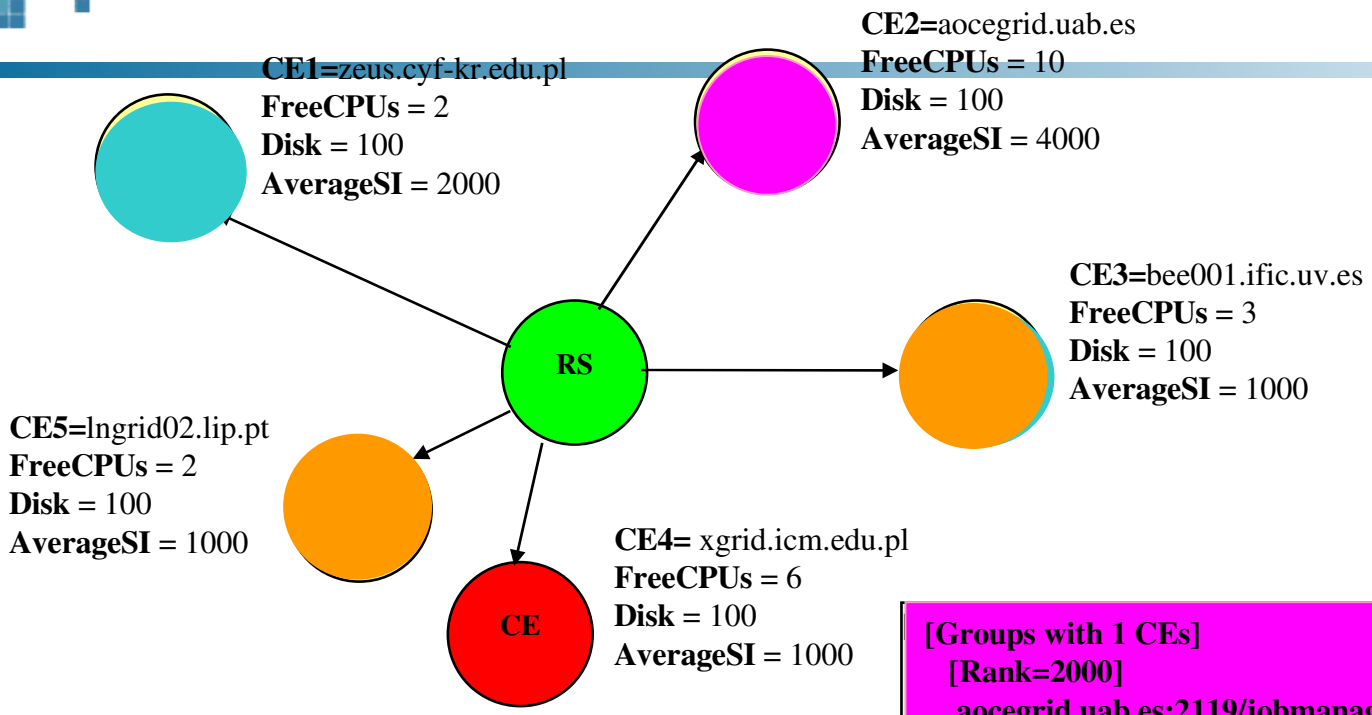
Parallel Job Support

```
Type = "Job";
VirtualOrganisation = "imain";
JobType = "Parallel";
SubJobType = "pacx-mpi";
NodeNumber =  5;
Executable = "test-app";
Arguments = "-v";
InputSandbox = {"test-app", "inputfile"};
OutputSanbox = {"std.out", "std.err"};
StdErr = "std.err";
StdOutput = "std.out";
Rank = other.GlueHostBenchmarkSI00 ;
Requirements =
   other.GlueCEStateStatus == "Production";
```

MPI Across Sites

- CrossBroker search and selects sets of resources for the jobs
- There is no guarantee that all tasks of the same job will start at the same time
  - 1st choice: select only sites with free resources. The job will run immediately. Unfortunately, free resources are not always available
  - 2nd choice: allocate a resource temporarily and wait until all other tasks show up. Timeshare the resource with a backfilling policy to avoid resource idleness

MPI Across Sites



[Groups with 1 CEs]
[Rank=2000]
aocegrid.uab.es:2119/jobmanager-pbs-workq
freeCPUs = 10

[Groups with 2 CEs]
[Rank=1500]
zeus.cyf-kr.edu.pl:2119/jobmanager-pbs-workq
freeCPUs = 2
bee001.ific.uv.es:2119/jobmanager-pbs-workq
freeCPUs = 3

Rank=1000]
lngrid02.lip.pt/jobmanager-pbs-workq
freeCPUs = 2
bee001.ific.uv.es:2119/jobmanager-pbs-workq
freeCPUs = 3

Inter-cluster with multiple requirements

```
Type = "Job";
VirtualOrganisation = "imain";
JobType = "Parallel";
SubJobType = "pacx-mpi"
Executable = "test-app";
Rank = other.GlueHostBenchmarkSI00 ;
InputSandbox = {"test-app", "inputfile"};
SubJobs = {
   [
    NodeNumber = 5;
    Requirements = Member("OpenGL", other.GlueHostApplicationRTE);

  ],
   [
  NodeNumber = 12;
  Requirements = other.GlueCEStateStatus== "Production";
   ]
}
```

Interactive Job Support

- Scheduling priority
  - Interactive jobs are sent to sites with available machines
  - If there are not available machines, use time sharing (glidein)
- Support for interactivity in all kinds of jobs
  - sequential and all the MPI flavors
- CrossBroker injects interactive agents that enable communication between user and job
  - Transparent to the user
  - Full integration with glogin & gvid

Interactive Job Support

- Changes in JDL
  - INTERACTIVE: true/false. Indicates that the job is interactive and the broker should treat it with higher proirity
  - INTERACTIVEAGENT
  - INTERACTIVEAGENTARGUMENTS
    - These attributes specify the command (and its arguments) used to communicate with the user.

Interactive Job Support

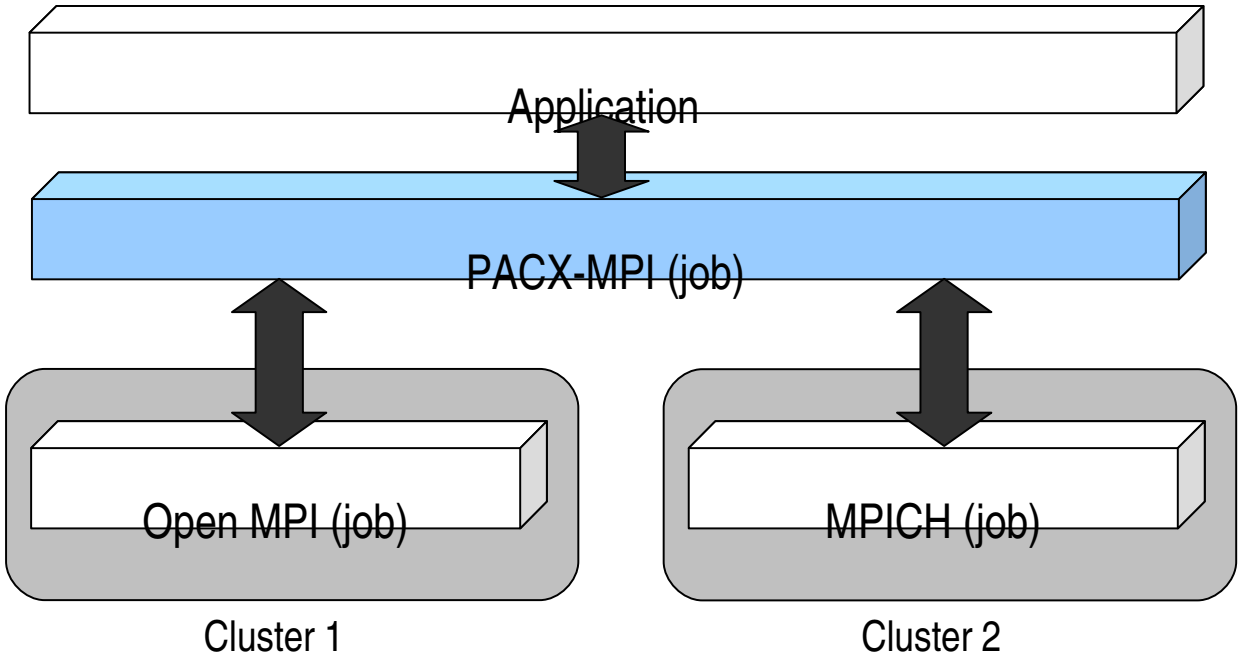
```
Type = "Job";
VirtualOrganisation = "imain";
JobType = "Parallel";
SubJobType = "openmpi";
NodeNumber =  11;
Interactive = TRUE;
InteractiveAgent = "glogin";
InteractiveAgentArguments = "-r -p 195.168.105.65:23433";
Executable = "test-app";
InputSandbox = {"test-app", "inputfile"};
OutputSanbox = {"std.out", "std.err"};
StdErr = "std.err";
StdOutput = "std.out";
Rank = other.GlueHostBenchmarkSI00 ;
Requirements =
   other.GlueCEStateStatus == "Production";
```

Other features

- Originally forked from LCG-RB, current development version merges code from glite WMS
  - We do not (yet) support CREAM
  - Some incompatible job features due to changes in JDL
- Intelligent job retrial
  - disables submission to failing sites temporarily
- Fast notification of job status
  - better interaction with the application
- gLite interoperability
  - accepts jobs from gLite's UI
  - able to submit jobs to gLite resources (LCG-CE and gLite CE)

PACX-MPI

Design


Application
PACX-MPI (job)
Open MPI (job)
MPICH (job)
Cluster 1
Cluster 2

PACX-MPI Usage

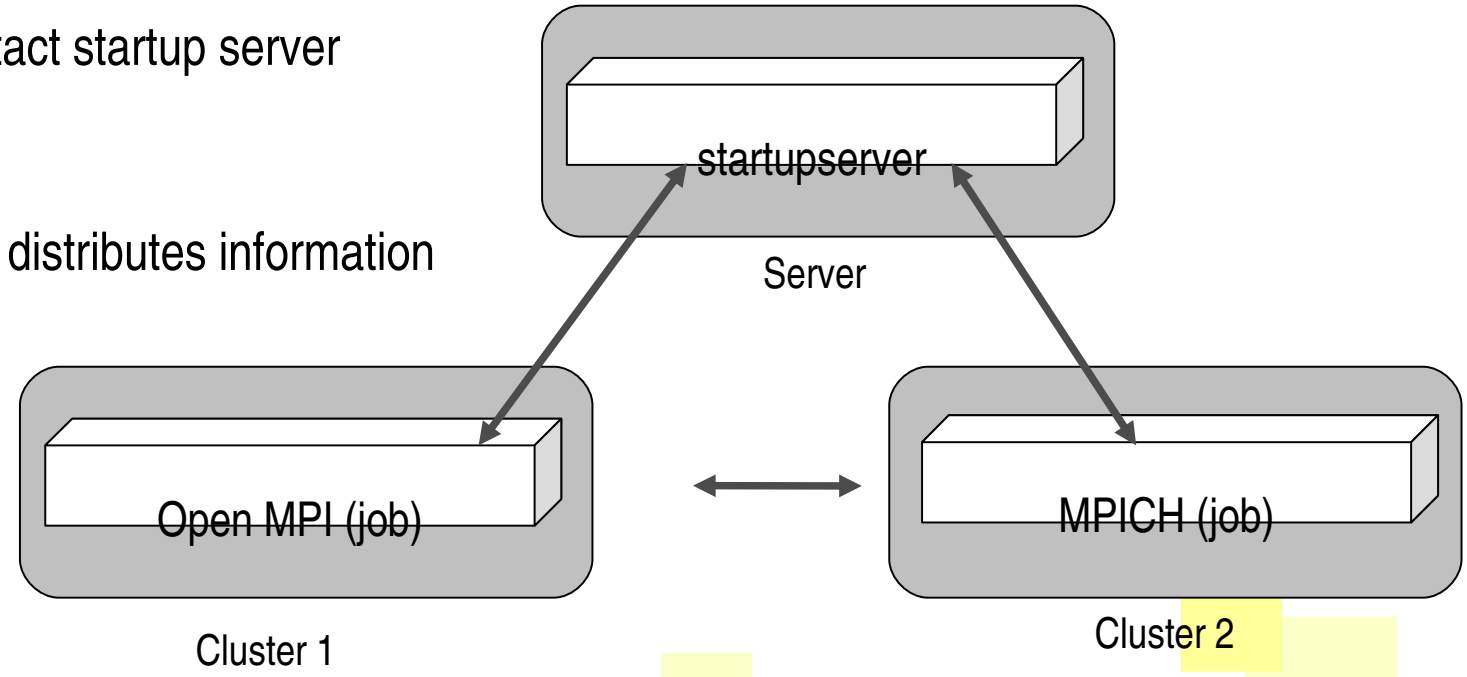
- PACX works with any MPI 1.2-conformant MPI implementation
- It uses PMPI interface
- Just recompilation of sources required:
  - pacxcc -o hello hello.c

How it works

- The processes of an application are mapped:
  - Global_process(N+2) : = Local_process(N) for 0..N-1 processes
  - Globsl_process(0) := outgoiing daemon
  - Global_process(1):= incoming daemon

PACX MPI – startup

- Startup server
- Daemons contact startup server
- Startup server distributes information
- Communicate



int.eu.grid

PACX-MPI usage

- The 2 additional daemons are run per local MPI application and relay inter-site communication
- Inter-site communication uses TCP/IP, intra-site uses vendor MPI implementation

Inter-site MPI With No WMS Support