int.eu.grid

Interactive European Grid

MPI Support in Int.Eu.Grid: Open MPI, PACX-MPI, MPI-Start

Sven Stork, Kiril Dichev

HLRS, Stuttgart

MPI Support
Overview

- Open MPI
  - a recent MPI-2 implementation
- PACX-MPI
  - MPI between clusters
- MPI-Start
  - a common layer to start MPI processes in EGEE and I2G

Open MPI
History

- in 2003 the developers of FT-MPI, LA-MPI, LAM/MPI decided to focus their experience and efforts on one MPI implementation instead on several
- in 2004 the real designing and coding started
- HLRS joined Open MPI
- First release on Super Computing 2005

Open MPI
Who is MPI?

- Founders
  - High Performance Computing Center, Stuttgart
  - Indiana University
  - Los Alamos National Laboratory
  - The University of Tennessee
- Current status
  - 14 Members
  - 8 Contributors
  - 1 Partner

Open MPI
Project Goals

- State of the art MPI implementation
  - Full support of the MPI-2 standard
  - Full thread support
  - Avoidance of old legacy code
  - Profit from long experience in MPI implementations
  - Avoiding the “forking” problem
  - Community / 3rd party involvement
  - Production-quality research platform
  - Rapid deployment for new platforms

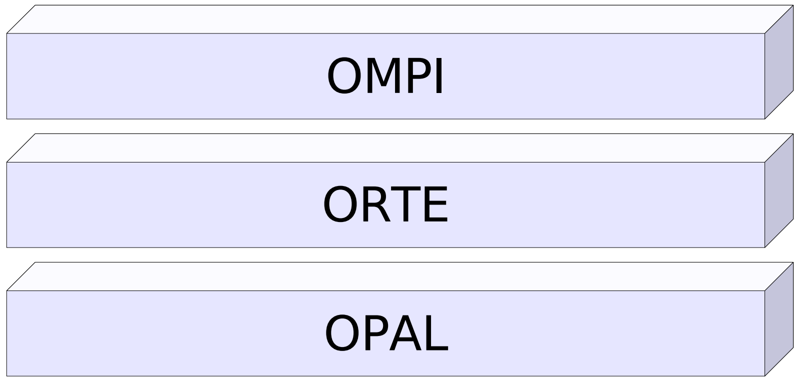
int.eu.grid

Open MPI
Design Goals

- Component architecture
- Design for heterogeneous environments
- Multiple networks (run-time selection)
- Support for automatic error detection / retransmission
- Portable and performant
  - Small cluster
  - “Big iron” hardware
  - “Grid” (everyone has a different definition)
  - ...

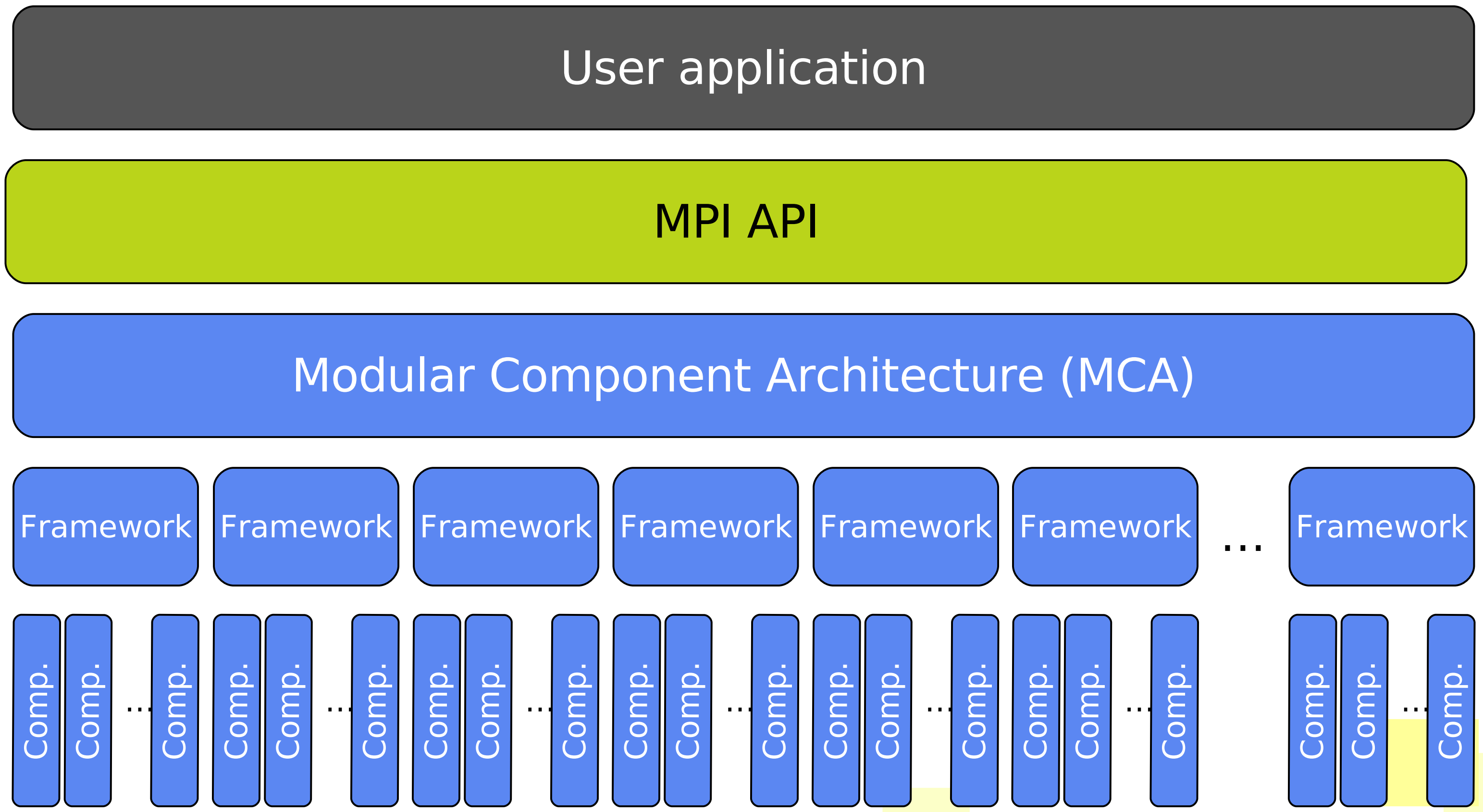
Open MPI
Design Goals

- Open MPI consists of 3 different layers
  - OMPI : The MPI API and supporting logic
  - ORTE : The Open Run-Time Environment (support for different back-end run-time systems)
  - OPAL : The Open Portable Access Layer (utility and "glue" code used by OMPI and ORTE)



Open MPI
Modular Component Architecture

- MCA top level view



Open MPI
MCA Concept

- Each MCA framework
  - will load all available components
  - evaluate/query each component regarding its capability to run in the current environment
  - will initialize the (best) runnable components and unload the others
- Composing the best fitting MPI implementation for the current system 'on the fly'

Open MPI
Example Parameters

- selecting which low level transport to use/ not to use
  - mpiexec -n 4 -mca btl mvapi,sm,self Killer_App
  - mpiexec -n 4 -mca btl ^tcp Killer_App
- manipulating the startup mechanism
  - mpiexec -n 4 -mca pls_rsh_agent /usr/bin/ssh.orig Killer_App
- performance tuning
  - mpiexec -n 4 -mca mpi_leave_pinned 1 Killer_App
  - mpiexec -n 4 -mca btl_tcp_rcvbuf 514288 Killer_App

Open MPI
Startup

- Open MPI supports both **mpirun** and **mpiexec**
- ORTE supports several startup mechanisms natively
  - rsh/ssh based
  - PBS/Torque
  - SGE
- The startup mechanism can be forced/selected via an MCA parameters

Open MPI
Requirements

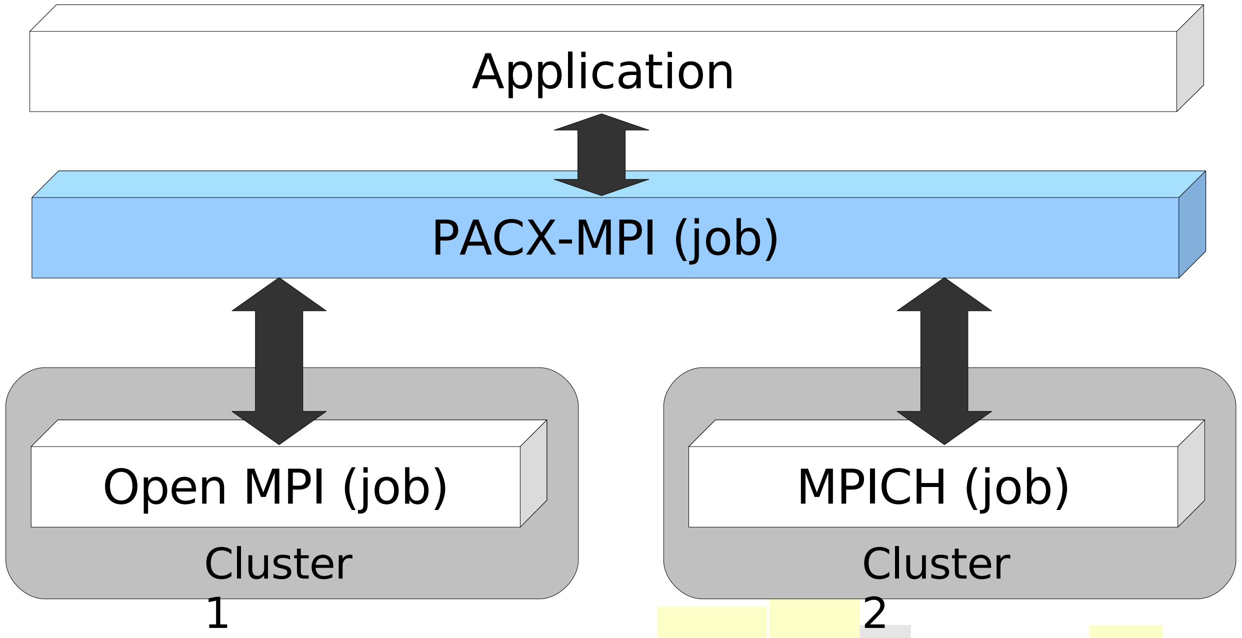
- The orte daemons need an open TCP/IP Port for incoming connections
- Different requirements for the different PLS
- ssh requires login without password (e.g. public keys)
- software installation
  - Open MPI needs to be installed on the
    - WN
    - CE (for PACX-MPI runs)

PACX-MPI
Overview

- A middleware for seamlessly running an MPI-application on a network of parallel computers
- originally developed in 1995 to connect Vector+MPP
- PACX-MPI is an optimized standard-conforming MPI- implementation
- application just needs to be **recompiled**
- PACX-MPI uses locally installed, optimized vendor implementations for cluster inter communication

PACX-MPI
Overview

- PACX-MPI starts an MPI job in each cluster
- PACX-MPI “merges/manages” these MPI jobs internally and emulate transparently a bigger MPI job to the application



PACX-MPI
Usage

- PACX-MPI compiler wrappers
  - pacxcc
  - pacxfc
  - ppacxcc
  - ppacxfc
- compiling with PACX
  - pacxcc -c hello.c
  - pacxcc -o hello hello.o

MPI-Start
Overview

- Motivation
  - Parallel job support in Grids
  - Problems with the current approach
  - Future Parallel Job Support in Grid
- mpi-start
  - Design Goals
  - Architecture
- Usage
- Advanced Usage

MPI-Start
Motivation

- Parallel job support in GRID
  - The current Grid middleware has only support for “Normal” jobs and “MPICH” jobs.
  - “Normal” JobType
    - Execution of a sequential program
    - one process allocation.
  - “MPICH” JobType
    - The name is program, only supports MPICH
    - Hard-coded into the WMS/RB
    - The grid middleware produces a wrapper script that executes the binary with (mpich) mpirun

MPI-Start
Motivation

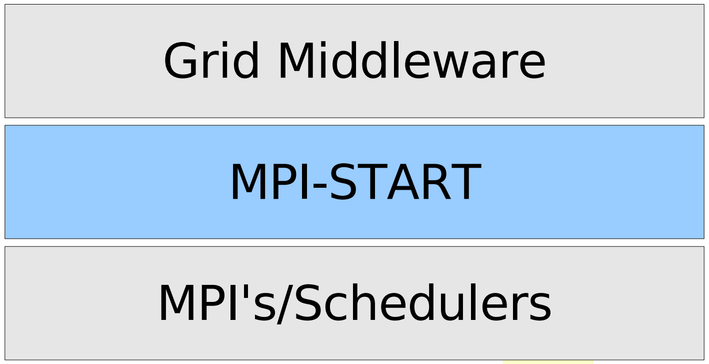
- Problems with hard-coded approach
  - For every new implementation that needs to be supported the middleware needs to be modified
  - The modified middleware needs to be recompiled (compiling about 1-2 hours, setting up a proper build environment about a few days)
  - The modified middleware needs to go through the whole release cycle:
    - Test + Validation
    - Testbed
    - Release
    - takes about 8 month in the EGEE project
  - Combinations of schedulers and MPI implementations might not work
    - how to find the hostfile
    - e.g. format of the SGE machinefile is not supported by mpich2.

MPI-Start
Motivation

- More Grid specific problems
  - 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 ?
  - How to compile MPI program ?
    - How can a physicist working on Windows workstation compile his code for/with an Itanium MPI implementation ?
    - License issues when giving people access to compiler ?

MPI-Start
Goals

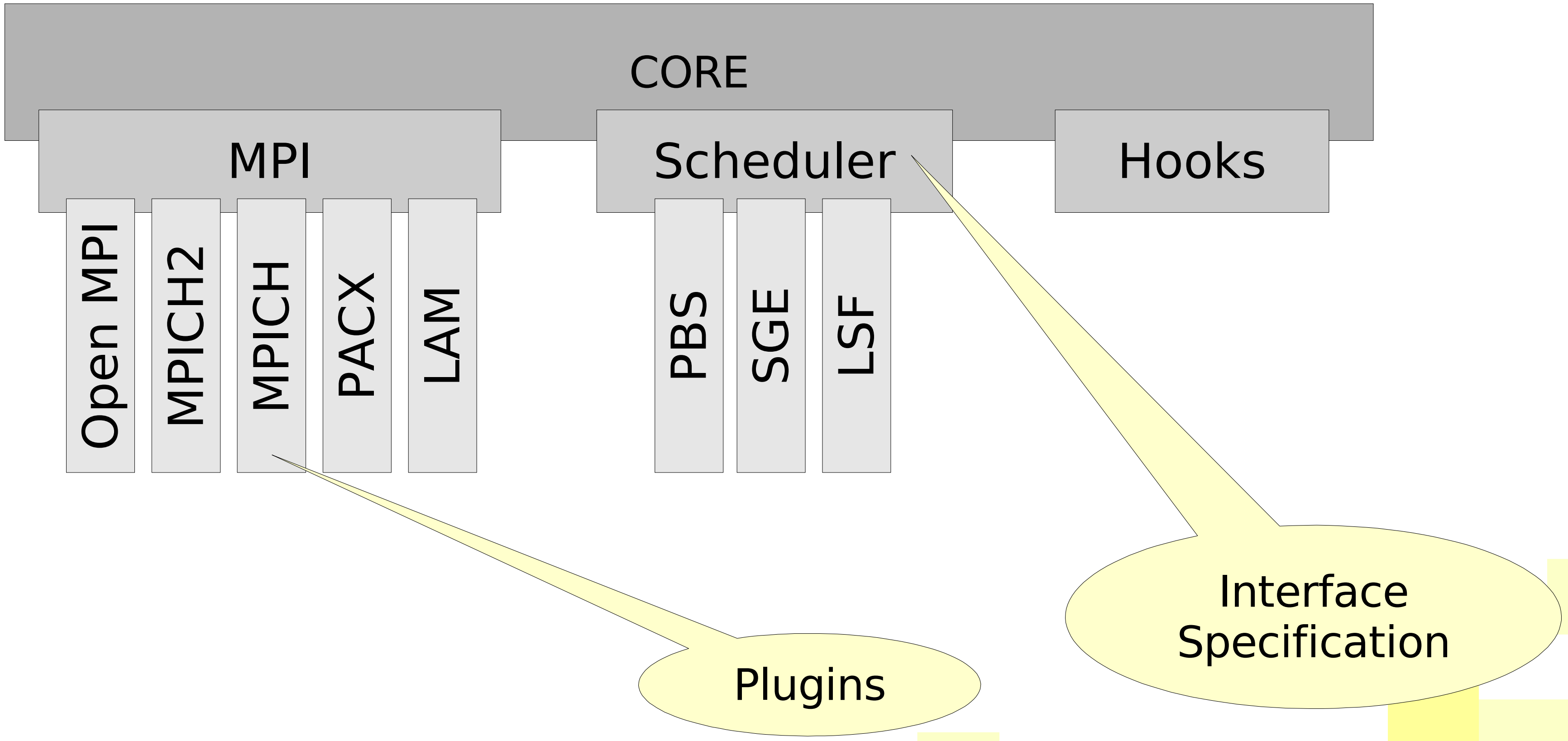
- Goals of mpi-start
  - Defines a unique interface to the upper layer for MPI jobs
  - Support of a new MPI implementation doesn't require any change in the Grid middleware
  - Support of “simple” file distribution
  - Provide some support for the user to help manage his data.



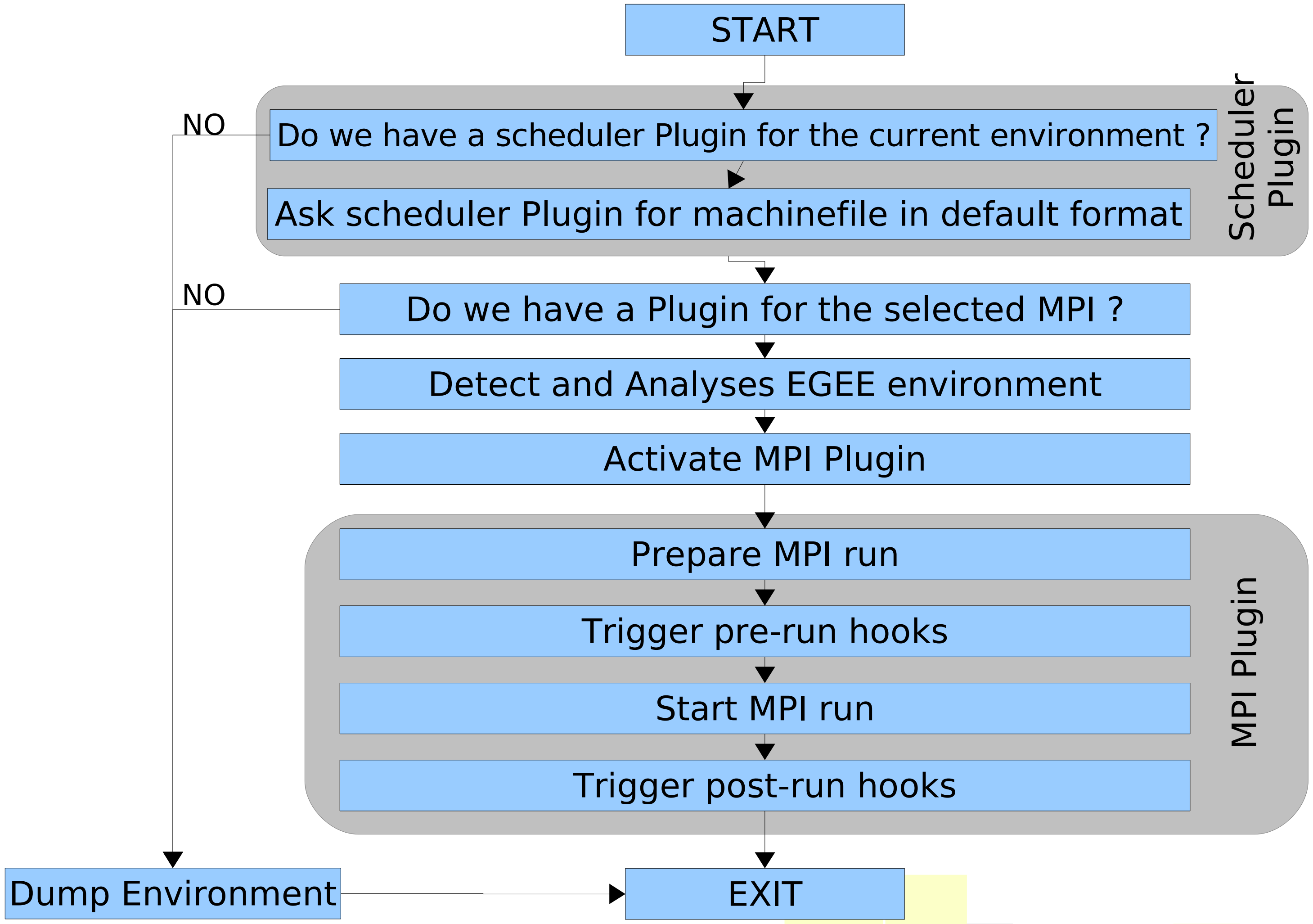
MPI-Start
Design Goals

- Design Goals
  - Portable
    - The program must be able to run under any supported operating system
  - Modular and extensible architecture
    - Plugin/Component architecture
  - Relocatable
    - The program must be independent of absolute path, to adapt to different site configurations.
    - Remote “*injection*” of mpi-start along with the job
  - Very good “remote” debugging features

MPI-Start
Architecture



MPI-Start
Flow


START
NO
Do we have a scheduler Plugin for the current environment ?
Ask scheduler Plugin for machinefile in default format
Scheduler Plugin
NO
Do we have a Plugin for the selected MPI ?
Detect and Analyses EGEE environment
Activate MPI Plugin
Prepare MPI run
Trigger pre-run hooks
Start MPI run
Trigger post-run hooks
MPI Plugin
Dump Environment
EXIT

MPI-Start
Interface

- Interface **Intra** Cluster MPI
  - I2G_MPI_APPLICATION
    - This variable describes the executable.
  - I2G_MPI_APPLICATION_ARGS
    - This variable contains the parameters of the executable.
  - I2G_MPI_TYPE
    - Specifies the MPI implementation to use (e.g openmpi, ...).
  - I2G_MPI_VERSION
    - Specifies which version of the the MPI implementation to use. If not defined the default version will be used.

MPI-Start
Interface

- Interface **Intra** Cluster MPI
  - I2G_MPI_PRECOMMAND
    - Specifies a command that is prepended to the mpirun (e.g. time).
  - I2G_MPI_PRE_RUN_HOOK
    - This variable can point to a shell script that must contain a “pre_run_hook” function. This function will be called before the parallel application is started.
  - I2G_MPI_POST_RUN_HOOK
    - Like I2G_MPI_PRE_RUN_HOOK, but the script must define a “post_run_hook” that is called after the parallel application finished.

MPI-Start
Interface

- Interface **Inter** Cluster MPI
  - I2G_MPI_FLAVOUR
    - Specifies which local sub MPI implementation to use.
  - I2G_MPI_JOB_NUMBER
    - In the case of a multi cluster MPI job this variable indicate the sub-job id.
  - I2G_MPI_STARTUP_INFO
    - Special synchr. informations for a inter cluster MPI job.
  - I2G_MPI_RELAY
    - Specifies the host via which the MPI traffic will be routed

MPI-Start

Status

- Current progress:
  - Support of MPI in a single cluster
  - Support of different MPI implementations simultaneously
  - Remove all MPI implementation specific features from the middleware
    - Int.EU.Grid : Still provide for each MPI implementation a special JobType, but this information is only passed through to a generic wrapper script. The RB automatically selects proper sites.
  - Have an abstraction layer that simplifies/abstracts from the low-level handling
  - in EGEE:
    - MPI Job == Normal Job that is allowed to allocate more than 1 process. The user has to:
      - take care of the complexity of the MPI implementations.
      - specify the requirements for a suitable site.

MPI-Start
Examples

- Example of start script

```
#!/bin/sh
# IMPORTANT : This example script execute a
#             non-mpi  program
#
export I2G_MPI_APPLICATION=/bin/hostname
export I2G_MPI_APPLICATION_ARGS=
export I2G_MPI_NP=2
export I2G_MPI_TYPE=openmpi
export I2G_MPI_FLAVOUR=openmpi
export I2G_MPI_JOB_NUMBER=0
export I2G_MPI_STARTUP_INFO=
export I2G_MPI_PRECOMMAND=
export I2G_MPI_RELAY=

$I2G_MPI_START
```

must be set to the absolute path of mpi-start

MPI-Start
Debugging

- Debugging Support
  - The debugging support is controllable via environment variables
  - The default is **not** to produce any additional output
  - I2G_MPI_START_VERBOSE
    - If set to 1 only very basic information are produced
  - I2G_MPI_START_DEBUG
    - If set to 1 information about the internal flow are outputed
  - I2G_MPI_START_TRACE
    - If set to 1 that “set -x” is enabled at the beginning.

MPI-Start

Debugging

- Debugging output example (I2G_MPI_START_VERBOSE=1)

```
*****************************************************************
UID      =  iman003
HOST     =  iwra54.fzk.de
DATE     =  Mon Dec 18 16:25:31 CET 2006
VERSION =  0.0.26
*****************************************************************
mpi-start [INFO   ]: search for scheduler
mpi-start [INFO   ]: activate support for pbs
mpi-start [INFO   ]: activate support for openmpi
mpi-start [INFO   ]: call backend MPI implementation
mpi-start [INFO   ]: start program with mpirun
=[START]=========================================================

<<OUTPUT>>

=[FINISHED]======================================================
```

MPI-Start
Usage
IMB (aka Pallas)

- JDL in the I2G framework
  - MPI-Start usage is transparent
  - I2G middleware takes care of the details

```
Executable       = "IMB-MPI1";
Arguments        = "barrier";
JobType          = "Parallel";
SubJobType       = “openmpi”;
NodeNumber       = 4;
StdOutput        = "std.out";
StdError         = "std.err";
OutputSandbox    = {"std.out","std.err"};
InputSandbox     = {"IMB-MPI1"};
```

MPI-Start
Usage
IMB (aka Pallas)

- stdout.txt

```
#---------------------------------------------------
#    Intel (R) MPI Benchmark Suite V2.3, MPI-1 part
#---------------------------------------------------
# Date         : Mon Aug 13 13:05:03 2007
# Machine      : i686# System      : Linux
# Release      : 2.4.21-47.0.1.EL.cernsmp
# Version      : #1 SMP Thu Oct 19 16:35:52 CEST 2006

...

#---------------------------------------------------
# Benchmarking Barrier
# #processes = 2
# ( 2 additional processes waiting in MPI_Barrier)
#---------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
         1000        11.43        11.43        11.43

#---------------------------------------------------
# Benchmarking Barrier
# #processes = 4
#---------------------------------------------------
 #repetitions  t_min[usec]  t_max[usec]  t_avg[usec]
         1000       187.78       187.88       187.83
```

MPI-Start
Hooks

- Hooks are divided into:
  - pre-run-hooks (integrated: file distribution)
  - post-run-hooks (integrated: cleanup of files)
- Any user-defined pre-run-hooks and post-run-hooks easily added
- Useful examples:
  - pre-run-hooks - compilation of the binary
  - post-run-hooks – fetch some runtime data into storage element

MPI-Start
Example of User-Defined Hooks

- o3.jdl

```
JobType                = "Parallel";
SubJobType             = “openmpi”;
NodeNumber             = 8;
VirtualOrganisation = "imain";
Executable             = "o3sg_8";
StdOutput              = "std.out";
StdError               = "std.err";
InputSandbox           = {"o3sg_8","o3_hooks.sh","input.8"};
OutputSandbox          = {"std.out","std.err"};
Environment    = {"I2G_MPI_PRE_RUN_HOOK=./o3_hooks.sh",
                  "I2G_MPI_POST_RUN_HOOK=./o3_hooks.sh"};
```

MPI-Start

Example of User-Defined Hooks

- o3_hooks.sh (1/2)

```
#!/bin/sh
export OUTPUT_PATTERN=L8
export OUTPUT_ARCHIVE=output.tar.gz
export OUTPUT_HOST=iwrse2.fzk.de
export OUTPUT_SE=lfn:/grid/imain/sven
export OUTPUT_VO=imain

pre_run_hook () {
}

copy_from_remote_node() {

    if [[ $1 == `hostname` || $1 == 'hostname -f' || $1 == "localhost" ]]; then
        echo "skip local host"
        return 1
    fi

    # pack data
    CMD="scp -r $1:\"$PWD/$OUTPUT_PATTERN\" ."
    echo $CMD
    $CMD
}

...
```

MPI-Start

Example of User-Defined Hooks

- o3_hooks.sh (2/2)

```
...

post_run_hook () {
    echo "post_run_hook called"

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

    ls -al

    echo "pack the data"
    tar cvzf $OUTPUT_ARCHIVE $OUTPUT_PATTERN
    echo "upload the data"
    lcg-cr --vo $OUTPUT_VO -d $OUTPUT_HOST -l $OUTPUT_SE/$OUTPUT_ARCHIVE
file://$PWD/$OUTPUT_ARCHIVE

    return 0
}
```

MPI-Start
Advanced Features

- Advanced Features
  - Interactivity
  - Remote Injection
  - “Remote Debugging”

MPI-Start
Remote Injection

- mpi-start can be run without being installed locally
  - just unpack mpi-start
  - setup environment
  - run mpi-start and let it do the rest

MPI-Start
Remote Injection

- remote_injection.jdl

```
Executable          = "my-starter.sh";
JobType             = "Parallel";
SubJobType          = "Plain";
NodeNumber          = 4;
StdOutput           = "std.out";
StdError            = "std.err";
OutputSandbox       = {"std.out","std.err"};
InputSandbox        = {"my-mpi-start.tar.gz",
                       "my-starter.sh"}
```

just allocate nodes

tarball with mpi-start

MPI-Start
Future plans

- Future Parallel Job Support in Grid
  - Short/Medium Term:
    - have a separate framework for the file distribution
    - write a framework for recognition of environments (I2G/EGEE/none)
    - have a setup process to use the environments framework
  - Long term
    - Have support for the MPI jobs between multiple clusters.
    - Have support for other parallel programming models (e.g. shared memory/OpenMP).
      - Problem with allocation of N processes on the same physical node.