Parallel Computing in EGI

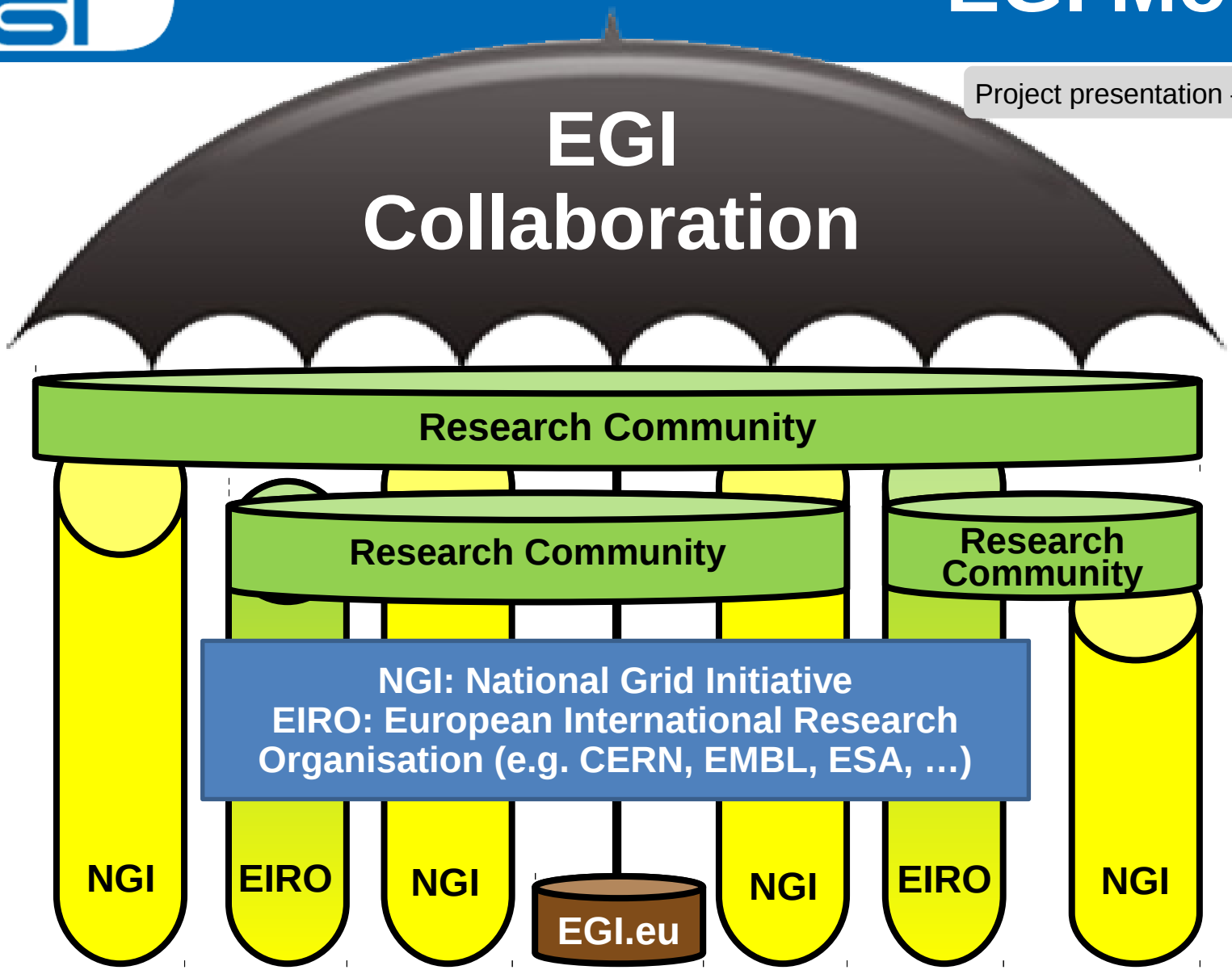
V. Šipková, M. Dobrucký, P. Slížik
Institute of Informatics, Slovak Academy of Sciences
Bratislava, Slovakia

Outline

- **EGI**
- **EGI-InSPIRE**
- **EGI middleware**
  - gLite 3.2, EMI 1 (Kebnekaise)
  - Job management services
    - CREAM (Computing Resource Execution and Management)
    - WMS (Workload Management System)
  - Grid jobs
    - Job Description Language (JDL) language
    - MPI jobs, OpenMP jobs

- **EGI (European Grid Initiative)**
  (EGI design study was created within 2007-2009)
  - a partnership between National Grid Initiatives (NGIs), European International Research Organisations (EIROs) and a coordinating body EGI.eu, which is the “glue” enabling coherence between NGIs for the benefit of the international user community
  - National Grid Initiatives are the main actors within EGI; they are autonomous, officially-recognized bodies that ensure the operation of the grid infrastructures in their own country, and an effective representation of the needs of their scientific communities and resource providers

EGI Model

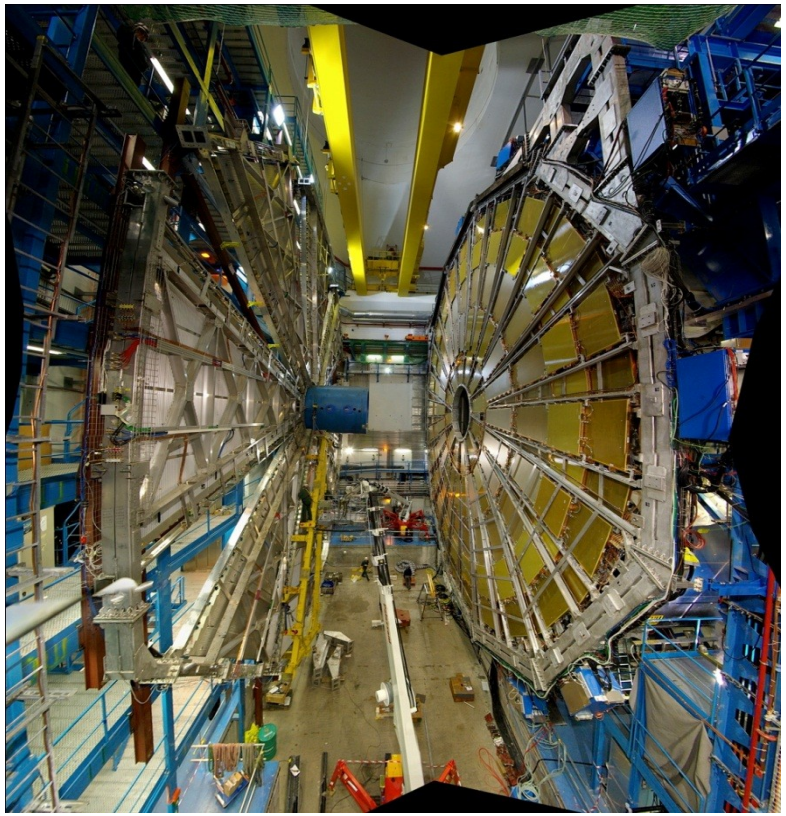

EGI
Collaboration
Research Community
Research Community
Research Community
NGI: National Grid Initiative
EIRO: European International Research Organisation (e.g. CERN, EMBL, ESA, …)
NGI
EIRO
NGI
EGI.eu
NGI
EIRO
NGI

- **EGI.eu**
  - foundation established in February 2010 to create and maintain a pan-European Grid Infrastructure (in collaboration with NGIs and EIROs) to guarantee the long-term development, availability and sustainability of e-infrastructure and grid services for all European research communities and their international partners
  - Its work builds on previous EU-funded projects: LHC/WLCG, DataGrid, EGEE

- **History of EGI**
  - LHC (Large Hadron Collider) at CERN, 1999
    - high-energy physics experiments (ALICE, ATLAS, CMS, LHCB)
  - WLCG (Worldwide LHC Computing Grid), 2002-
    - distributing computing grid to analyse the experimental data produced by the LHC
  - EDG (European DataGrid), 2002-2004
    - computing infrastructure providing intensive computation and analysis of shared large-scale databases; led the research and development of grid technologies
  - EGEE (Enabling Grids for E-sciencE), 2004-2010
    - took over the grid's further development

Grid Computing

- **LHC** (Large Hadron Collider)

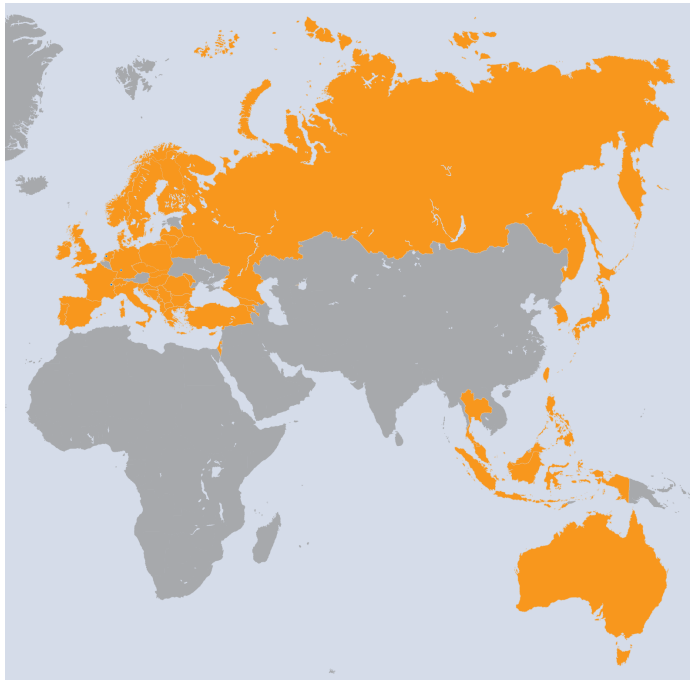


- **WLCG** (Worldwide LHC Computing Grid)
  - computing resources, data storage capacity, sensors, physics software, visualisation tools, etc.
  - more than 8000 physicists actively access and analyse the data in near real-time
    (15 Petabytes of data produced annually)

Grid Computing

- **EGEE**
  - the largest collaborative production grid infrastructure in the world for e-science supporting a wide range of research disciplines
  - three successive 2-year phases
  - at its close in 2010:
    - 300 research centres providing about 200 000 CPU cores, several Petabytes of storage, software tools and services
    - 13 million jobs running per month
    - thousands of scientists federated in over 200 VOs
    - grid middleware gLite - the crucial component which binds the resources into a single infrastructure providing access for the project's user communities

- **EGI-InSPIRE**
  Integrated Sustainable Pan-European Infrastructure for Researchers in Europe
  - Mission: to coordinate the transition from EGEE to EGI by supporting grids of HPC and HTC resources, and to provide support for the user communities
  - 4 year project, started in May 2010
  - Partners (50, from 40 countries)
    - EGI.eu, 38 NGIs, 2 EIROs, Asia Pacific (9 partners)
  - Funding from the EC: 25M €
  - Project cost: 72M €
  - Total effort: ~330M € / 9261 PMs



EGI Software

- EGI does not develop the software deployed in the grid infrastructure, all upgrades and new programs are produced by independent technology providers
  - EMI (European Middleware Initiative)
    - a deliver of a consolidated set of grid middleware components (as part of Unified Middleware Distribution UMD)
  - IGE (Initiative for Globus in Europe)
    - a service provider for the grid middleware Globus
  - SAGA (Simple API for Grid Applications)
    - tools and frameworks for building distributed applications
  - StratusLab
    - project to develop open-source cloud distribution that allows to exploit an infrastructure as a Service cloud

EGI Middleware

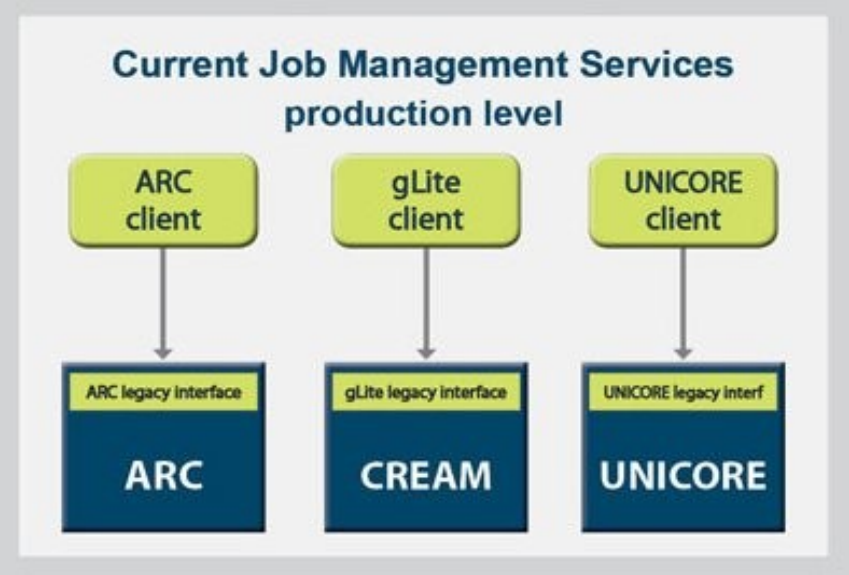
- **EGI midlleware**
  - a specific software product placed between the infrastructure and user applications, which enables sharing grid resources
    - HPC clusters, disk storage, various instruments, data archives or digital libraries, software packages, etc.
  - gLite 3.2 - used in the first period of EGI
  - EMI 1 (Kebnekaise) - released in May 2011
    - a unified, standardized software for distributed computing infrastructure
    - the reference platform: Scientific Linux 5/64 bit
  - EMI 2 (Matterhorn, 2012), EMI 3 (Monte Bianco, 2013)

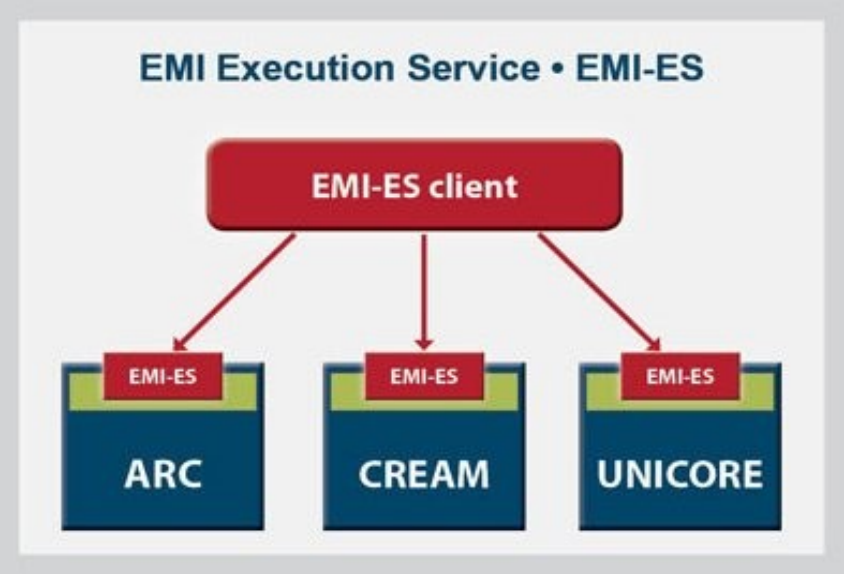
EGI Middleware

- **EMI 1** (Kebnekaise)
  - a complete and consolidated set of components from gLite, ARC, dCache, and UNICORE
  - the first release is focused primarily on:
    - laying the foundations for the distribution
    - increasing the level of integration among the original middleware stacks
    - improving the compatibility with mainstream operating systems' guidelines
    - extending the compliance with existing standards
  - EMI 1 introduces a number of changes and new functionalities in areas of Security, Computing, Data, and Infrastructure

- **Compute area**
  - includes middleware services and corresponding client components involved in the processing and management of user requests concerning the execution of a computational task
  - Services cover
    - the interaction with Local Resource Management Services
    - the provision of a common interface to the computational resources of a site (so-called Computing Element)
    - the availability of high-level meta-scheduling, workflow execution and task tracking functionality

- **Compute area – major highlights**
  - CREAM (Computing Resource Execution and Management) service has become an integrated part of the EMI1 execution service





- **Compute area – major highlights** (cont.)
  - the full support for CLUSTER service in CREAM
  - the integration of ARGUS authorisation in CREAM
  - the initial implementation of common MPI methods across the different compute services
    - the support for the user defined fine-grained mapping of processes to physical resources
    - new command-line options for MPI-Start
    - the basic support for OpenMP
    - the support for SLURM&Condor schedulers

Job Management

- **Job management services**
  - are concerned with the acceptance, scheduling, monitoring and execution of remote computations (called jobs)

- **Computing Element (CE)**
  - represents a set of computational resources localized at a site (i.e. a cluster); CE consists of:
    - Cluster - a collection of Worker Nodes (WNs) where jobs are executed
    - Local Resource Management System (LRMS) (batch system)
    - Grid Gate (GG) - acts as a generic interface to the cluster; the role of GG is to accept jobs and dispatch them for execution on WNs through the LRMS

Job Management

- **Grid Gate (GG)**
  - is implemented as the CREAM based computing element (CREAM-CE) - a lightweight service, which is responsible for performing all the job management operations
- **CREAM service**
  - accepts job submissions and other requests through the Workload Management System, or through a generic client which allows to submit jobs directly to a CREAM-CE

Job Management

- **Workload Management System (WMS)**
  - a global grid resource broker - a software service of the gLite which takes care of distributing and managing tasks across computing and storage resources available in the grid
  - WMProxy - main service providing access to the WMS
  - Both services CREAM and WMProxy implement similar functionalities and expose a web services interface, which the user can interact with by means of the command line interface

Grid Operations

- **WMS/CREAM client commands**
  enable to perform the following operations:
  - delegation of proxy credentials to speed up subsequent operations; renewal of delegations
  - match-making: to display the list of CEs which match the requirements of the job
  - submission of jobs for execution
  - monitoring the status of submitted jobs
  - cancellation of jobs at any point in their life cycle
  - suspension/resumption of running jobs
  - retrieval of the output of finished jobs and logging information of the job execution
  - getting information about jobs, services, and CEs

Client Commands

WMProxy	CREAM
glite-wms-delegate-proxy	glite-ce-delegate-proxy
glite-wms-job-submit	glite-ce-job-submit
glite-wms-job-status	glite-ce-job-status
glite-wms-job-cancel	glite-ce-job-cancel
glite-wms-job-output	glite-ce-job-output
glite-wms-job-perusal	glite-ce-job-suspend
glite-wms-job-logging-info	glite-ce-job-resume
glite-wms-job-list-match	glite-ce-job-purge
glite-wms-job-info	glite-ce-proxy-renew
	glite-ce-job-list
	glite-ce-service-info

Grid Jobs

- **Grid job**
  - consists of a remote computation, and optionally, a file transfer and management operations related to the computation
- **Job description**
  - jobs to be submitted to the grid must be described using the Job Description Language (JDL)
    - JDL is a flexible, extensible, high-level language based on the Condor Classified Advertisement which enables to describe jobs and aggregates of jobs with arbitrary dependency relations, and to express any requirements and constraints on the CE, WNs, SE and installed software

Grid Jobs

- **Application Type**
  JDL supports the following types of applications:
  - **Job** - a simple job
    - Normal: a simple batch-job (executable, script)
    - Parametric: multiple jobs with one parametrized description (a set of identical jobs operating on different data)
  - **DAG** (Directed Acyclic Graph) - a set of dependent jobs
    - the input/output/execution of one or more jobs may depend on one or more other jobs
  - **Collection** - multiple independent jobs with a common description

Grid Jobs

- **Application Type** (cont.)
  - WMS and CREAM services support different sets of JDL attributes ↔ different types of jobs
  - Simple jobs of type Normal - are supported by both the WMS and CREAM
  - Compound jobs: Parametric, DAG, Collection - are supported only by the WMS

Grid Jobs

- **Parallel applications**
  - MPI and multi-threaded OpenMP tasks are classified in JDL as “Simple” jobs of type “Normal”, but they need special handling
- **MPI** (Message Passing Interface)
  - has become a standard programming model for distributed memory architectures (e.g. clusters)
- **OpenMP** (Open Multi-Processing) Interface
  - provides a parallel programming model for shared memory architectures (e.g. multicore CPUs)

Grid Jobs

- **MPI jobs**
  - an MPI application must be initialised by means of an executable script, for instance, using the MPI-Start interface
- **MPI-Start**
  - is an abstraction layer that offers a unique interface to the grid middleware
  - it can be controlled via environment variables or command line switches

Grid Jobs

- **MPI-Start**
  - enables to start MPI programs with various execution environment implementations (MPICH, MPICH2, Open MPI, LAM-MPI, PACX-MPI) and batch schedulers (PBS/Torque, SGE, LSF)
  - provides a hooks framework which enables the simple file distribution for sites without a shared file system, and for user applications to perform any pre-processing (e.g. compilation, data fetching) and post-processing (e.g. storage of application results, clean-up)
  - supports the fine-grained mapping of MPI processes to physical resources

Grid Jobs

- **OpenMP jobs**
  - at present the submission of OpenMP jobs is supported only by the CREAM clients
  - as the CREAM service has become an integrated part of the EMI 1 execution service, CREAM JDL is constantly varying and extending in order to better address new requirements and scenarios

Conclusion

- **Summary**
  - Current computing systems allow applying of many parallel solutions at the same time to achieve the maximum performance and efficiency gains
  - The EGEE project was concentrated first of all on providing a computing infrastructure and on running distributed applications, the support for parallel applications was rather poor
  - EMI 1 Kebnekaise delivers a significant number of new features
    - the new CREAM JDL attributes and the support for fine-grained mapping of MPI processes and OpenMP threads to physical resources extend the variety of application models, and improve the resource utilization as well

Grid Jobs

- **Example: Simple job**

```
Type = "Job";
JobType = "Normal";
Executable = "start_test.sh";
Arguments = "test.exe input.dat output.dat";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = { "start-test.sh", "test.exe", "input.dat" };
OutputSandbox = { "std.out", "std.err", "output.dat" };
Retrycount = 0;
ShallowRetryCount = 3;
```

Grid Jobs

- **Example: MPI job**

```
Type = "Job";
JobType = "Normal";
CpuNumber = 8;
Executable = "start_mpitest.sh";
Arguments = "mpitest.exe input.dat";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = { "start-mpitest.sh", "mpitest.exe", "input.dat" };
OutputSandbox = { "std.out", "std.err", "mpistd.out", "output.dat" };
Requirements = other.GlueCEInfoTotalCPUs >= 8
   && Member("MPI-START",
        other.GlueHostApplicationSoftwareRunTimeEnvironment)
   && Member("OPENMPI",
        other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

Grid Jobs

- **Example: OpenMP job**

```
Type = "Job";
JobType = "Normal";
HostNumber = 1;
WholeNodes = True;
SMPGranularity = 4;
Environment = { "OMP_NUM_THREADS=4" };
Executable = "omptest.exe";
Arguments = "input.dat output.dat";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = { "omptest.exe", "input.dat" };
OutputSandbox = { "std.out", "std.err", "output.dat" };
Requirements =
    other.GlueHostArchitectureSMPSize >= SMPGranularity;
```

Grid Jobs

- **Example: Parametric job**

```
Type = "Job";
JobType = "Parametric";
Executable = "partest.exe";
Arguments = "input_PARAM_.dat output_PARAM_.dat";
Parameters = 10;
ParameterStart = 0;
ParameterStep = 1;
StdOutput = "std_PARAM_.out";
StdError = "std_PARAM_.err";
InputSandbox = { "partest.exe", "input_PARAM_.dat" };
OutputSandbox = { "std_PARAM_.out", "std_PARAM_.err",
                  "output_PARAM_.dat" };
```

Grid Jobs

- **Example: DAG job**

```
Type = "DAG";
InputSandbox = { "input.dat" };
max_running_nodes = 2;
nodes = [
   nodeA = [ description = [
      JobType = "Normal";
      Executable = "testA.exe";
      InputSandbox = { "testA.exe", rootInputSandbox[0] };
       . . .
   ]; ];
   nodeB = [ description = [ . . . ]; ];
   dependences = { nodeA, nodeB };
];
```

Thank you for your attention!