Porting Fusion and Biomed Applications using DRMAA API and Globus GridWay

JOINT EGEE AND SEE-GRID SUMMER SCHOOL ON GRID APPLICATION SUPPORT
Budapest, Hungary
June 29, 2007

José Luis Vázquez-Poletti
Distributed Systems Architecture Group
Universidad Complutense de Madrid
Contents

1. A Global Vision
2. GridWay Out There
3. Interacting With GridWay
   3.1. Command Line Interface
   3.2. DRMAA API (C Binding)
4. A Short Demo
1. A Global Vision

**GridWay**

- **Applications**
  - DRMAA
  - Results
  - CLI

- **Grid Scheduling Steps**
  - **open source meta-scheduler**
    - job execution management
    - resource brokering
  - Standard interfaces
  - Globus services
  - end-to-end (e.g. TCP/IP)
  - highly dynamic
  - heterogeneous
  - high fault rate

- **Infrastructure**
  - PBS
  - SGE

- **Grid Middleware**
  - GridWay
  - Globus
  - Standard interfaces
  - Globus services
  - end-to-end (e.g. TCP/IP)
  - highly dynamic
  - heterogeneous
  - high fault rate
1. A Global Vision

**Workload Management**
- Advanced scheduling functionalities and new scheduling policies support
- Failure detection and recovery
- Accounting service
- Simple jobs, job arrays and DAGs

**User Interface**
- Complete support for the OGF DRMAA Standard (C & JAVA)
- DRM-like command line interface

**Integration**
- Deployment of new services is not needed
- Interface with new grid services
- Interoperability with other infrastructures
## GridWay complements gLite

<table>
<thead>
<tr>
<th>FEATURES</th>
<th>BENEFITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRMAA OGF Standard support (C &amp; JAVA)</td>
<td>Application compatibility with DRM systems which implement the Standard: SGE, Torque, ...</td>
</tr>
<tr>
<td>DRM-like Command Line Interface (allows users to submit, kill, migrate, query state and synchronize jobs)</td>
<td>CLI similar to that found in local resource managers</td>
</tr>
<tr>
<td>Lightweight Middleware</td>
<td>Higher Efficiency for certain application profiles</td>
</tr>
<tr>
<td>Site-level Accounting and scheduling policies</td>
<td>Use analysis, profiling utilization and user monitoring</td>
</tr>
<tr>
<td>Minimum installation requisites</td>
<td>Easy deployment and maintenance</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Simultaneous access to different infrastructures</td>
</tr>
</tbody>
</table>
1. A Global Vision
1. A Global Vision

Grid Infrastructure

GridWay daemon

Execution Manager  Transfer Manager  Information Manager  Scheduling Module

Users

EGEE-CE  PBS Cluster

GSI-Blahp  MDS2

BLAHP

PBS Cluster

pre-WS GRAM  MDS2

Condor-C/BLAHP

PBS Cluster

Grid-FTP  MDS2

Storage Element

EGEE-SE

pre-WS GRAM  MDS2

PBS Cluster

Storage Element

EGEE-CE

Grid-FTP  MDS2

EGEE-SE

Future
gLite

gLite 3.0.2

LCG 2
(based on pre-WS GT)
1. A Global Vision

GridWay / LCG-2 RB Comparative
EGEE 1st User Forum (CERN, 1st-3rd March 2006)

- Avg Productivity
- Avg Overhead/job
- Jobs Affected
- Total
- Max Resub/Resch per Job
- Productivity (Every 15')

- Resubmissions/Reschedules
Contents

1. A Global Vision

2. GridWay Out There

3. Interacting With GridWay
   3.1. Command Line Interface
   3.2. DRMAA API (C Binding)

4. A Short Demo
2. GridWay Out There

Some Projects and Infrastructures

- IRISGrid
- Politecnico di Torino
- CABGrid (Centro de Astrobiología)
- C2VO (Universidad de Castilla La Mancha)
- Grid en ESAC (Agencia Espacial Europea)
- CRO-GRID (Croacia)
- Sun Microsystems Solution Center World Grid
- Infraestructura EGEE
- Proyecto BeinGRID
- GridX1 (Canadian Grid for HEP applications)
- Universidade do Porto
- Madras Institute of Technology
- National Center for High-Performance Computing

More at: http://www.gridway.org/ (Success Stories)

Some Application Porting Areas

- Life-Sciences
- Aerospace
- Fusion Physics
- Computational Chemistry
2. GridWay Out There

**MAssive RAy TRAcings in Fusion Plasmas**

Executable: *Truba*
- 1,8 MB – 9' – 50 Executions
- Input files =~ 70 KB
- Output files =~ 549 KB

**CD-HIT**

To your local cluster
- PBS
- GridWay
- To EGEE

**GridWay**

**cniö**
Contents

1. A Global Vision
2. GridWay Out There
3. Interacting With GridWay
   3.1. Command Line Interface
   3.2. DRMAA API (C Binding)
4. A Short Demo
3.1 Command Line Interface

Main Commands

- **gwps**: Shows job information and state
- **gwhistory**: Shows execution history
- **gwkill**: Sends signals to a job (kill, stop, resume, reschedule)
- **gwsubmit**: Submits a job or array
- **gwwait**: Waits for job's end (any, all, set)
- **gwuser**: User Monitoring
- **gwhost**: Host Monitoring
- **gwacct**: Accounting

Application Model and Lifecycle

[Diagram of job lifecycle and resource management]
# Execution variables
EXECUTABLE = job
ARGUMENTS = ${TASK_ID} ${TOTAL_TASKS} 100000
ENVIRONMENT = LD_LIBRARY_PATH=/usr/local/lib

# Resource selection parameters
REQUIREMENTS = HOSTNAME= "*.dacya.ucm.es"
RANK = CPU_MHZ

# I/O files
INPUT_FILES = my_inputfile
OUTPUT_FILES = my_outputfile

# Standard streams
STDOUT_FILE = stdout_file.${TASK_ID}
STDERR_FILE = stderr_file.${TASK_ID}
... other variables related to checkpointing, fault tolerance, performance,...
3.2 DRMAA API (C Binding)

**DRMAA API**

- **Distributed Resource Management Application API**
  http://www.drmaa.org/
- Open Grid Forum Standard
- Homogeneous interface to different Distributed Resource Managers (DRM):
  - SGE
  - Condor
  - PBS/Torque
- **GridWay**
  - C
  - JAVA
  - Soon!
  - Perl & Ruby
3.2 DRMAA API (C Binding)

Application Profiles with DRMAA

- **Embarrassingly Distributed**

```c
rc = drmaa_init(contact, err);
// Execute initial job and wait for it
rc = drmaa_run_job(job_id, jt, err);
rc = drmaa_wait(job_id, &jstat, timeout, rusage, err);
// Execute n jobs simultaneously and wait
rc = drmaa_run_bulk_jobs(job_ids, jt, 1,
 JOB_NUM, 1, err);
rc = drmaa_synchronize(job_ids, timeout, 1, err);
// Execute final job and wait for it
rc = drmaa_run_job(job_id, jt, err);
rc = drmaa_wait(job_id, &jstat, timeout, rusage, err);
rc = drmaa_exit(err_diag);
```

- **Master-Worker**

```c
rc = drmaa_init(contact, err_diag);
// Execute initial job and wait for it
rc = drmaa_run_job(job_id, jt, err_diag);
rc = drmaa_wait(job_id, &jstat, timeout, rusage, err_diag);
while (exitstatus != 0)
{
    // Execute n Workers concurrently and wait
    rc = drmaa_run_bulk_jobs(job_ids, jt, 1, JOB_NUM, 1, err_diag);
    rc = drmaa_synchronize(job_ids, timeout, 1, err_diag);
// Execute the Master, wait and get exit code
rc = drmaa_run_job(job_id, jt, err_diag);
rc = drmaa_wait(job_id, &jstat, timeout, rusage, err_diag);
rc = drmaa_wexitstatus(&exitstatus, stat, err_diag);
}
rc = drmaa_exit(err_diag);
```
Include the DRMAA library

```c
#include "drmaa.h"
```

Verify the following env variable (.bashrc)

```bash
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$GW_LOCATION/lib/
```

Include the compiling options for DRMAAA

```bash
-gcc ejemplo.c -L $GW_LOCATION/lib
-I $GW_LOCATION/include -lDrmaa -o ejemplo
```
3.2 DRMAA API (C Binding)

**DRMAA Sessions**

- Initialize DRMAA Session
  ```c
  int drmaa_init (const char *contact, char *error_diagnosis, size_t error_diag_len)
  ```

- Finalize DRMAA Session
  ```c
  int drmaa_exit (char *error_diagnosis, size_t error_diag_len)
  ```

**Job Template Creation**

- Job Template Allocation
  ```c
  int drmaa_allocate_job_template (drmaa_job_template_t **jt, char *error_diagnosis, size_t error_diag_len)
  ```

- Scalar Attribute
  ```c
  int drmaa_set_attribute (drmaa_job_template_t *jt, const char *name,
                          const char *value, char *error_diagnosis, size_t error_diag_len)
  ```

- Vector Attribute (i.e. string with executable arguments)
  ```c
  int drmaa_set_vector_attribute (drmaa_job_template_t *jt, const char *name,
                                  const char *value[], char *error_diagnosis, size_t error_diag_len)
  ```
3.2 DRMAA API (C Binding)

Job Submission

Submit a Job

```c
int drmaa_run_job (char *job_id, size_t job_id_len,
                   drmaa_job_template_t *jt, char *error_diagnosis, size_t error_diag_len)
```

Wait for the Job

```c
int drmaa_wait (const char *job_id, char *job_id_out, size_t job_id_out_len, int *stat, signed long timeout,
                 drmaa_attr_values_t **usage, char *error_diagnosis, size_t error_diag_len)
```

Get the exit code

```c
int drmaa_wexitstatus (int *exit_status, int stat, char *error_diagnosis, size_t error_diag_len)
```

Get remote resource usage stats

```c
int drmaa_get_next_attr_name (drmaa_attr_names_t *values, char *value, size_t value_len)
```

Remove Job Template

```c
int drmaa_delete_job_template (drmaa_job_template_t *jt, char *error_diagnosis, size_t error_diag_len)
```
3.2 DRMAA API (C Binding)

**Job Status and Control**

Get Job status

```c
int drmaa_job_ps (const char *job_id, int *remote_ps, char *error_diagnosis, size_t error_diag_len)
```

Wait Job ending

```c
int drmaa_synchronize (const char *job_ids[], signed long timeout,
                    int dispose, char *error_diagnosis, size_t error_diag_len)
```

Send control signals to Job

```c
int drmaa_control (const char *jobid, int action, char *error_diagnosis, size_t error_diag_len)
```

**Job Arrays**

Submit Job array

```c
int drmaa_run_bulk_jobs (drmaa_job_ids_t **jobids, drmaa_job_template_t *jt, int start,
                         int end, int incr, char *error_diagnosis, size_t error_diag_len)
```

Get next Job id

```c
int drmaa_get_next_job_id (drmaa_job_ids_t *values, char *value, size_t value_len)
```
Contents

1. A Global Vision
2. GridWay Out There
3. Interacting With GridWay
   3.1. Command Line Interface
   3.2. DRMAA API (C Binding)
4. A Short Demo
For More Information

http://www.GridWay.org/

Metascheduler

Metascheduling Technologies for the Grid

Welcome to GridWay

The GridWay Metascheduler enables large-scale, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different LRMs (Local Resource Management) systems, such as PBS, SGE, LSF, Conda... within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid). GridWay is a Globus project, adhering to Globus philosophy and guidelines for collaborative development and co-releasing code and support contributions from individuals and corporations around the world.

Why GridWay?

There exist a number of commercial and open source workload management and scheduling systems available today, each one suitable for different underlying computer infrastructures and execution profiles. GridWay stands out from other metascheduling systems because it has been specifically designed to work on top of Globus services, offering the highest functionality, quality of service and reliability on this kind of infrastructures, namely:

- For project and infrastructure directors. GridWay is an open-source community project, adhering to Globus philosophy and guidelines for collaborative development.
- For system integrators. GridWay is highly modular, allowing adaptation to different grid infrastructures, and supports several OGF standards.
- For system managers. GridWay gives a scheduling framework similar to that found on local LRMs systems, supporting resource accounting and the definition of state-of-the-art scheduling policies.
- For application developers. GridWay implements the OGF standard DRMAA API (C and JAVA bindings), ensuring compatibility of applications with LRMs systems that implement the standard, such as SGE, Condor, Torque...
- For end users. GridWay provides a LRMA-like CLI for submitting, monitoring, synchronising and controlling jobs, that could be described using the OGF standard JSDL.

With GridWay, a Grid infrastructure can be exploited and managed in the same way as a local computing cluster. We invite you to check its Metascheduling Functionality Checklist and its benefits for end users and system administrators.
Thank you for your attention!