The GridWay Metascheduler

Constantino Vázquez and Eduardo Huedo

dsa-research.org

Open Grid Forum 22

Boston, February 27, 2008

Contents

- **Introduction**

- **What is GridWay?**

- **Architecture**

- **Components**

- **Scheduling Policies**

- **Examples of Grid Deployments**

- **History**

Introduction

- **Resource selection**: Where do I execute my job ?

- **Resource preparation:** What do I need?

- **Job submission**: How do I submit my job?

- **Job monitoring:** How is my job doing?

- **Job migration:** Is there any better resource?

- **Job termination:** How do I get my output?

Introduction

- **Meta-scheduler:** Job to resource (**other schedulers**) matching (*execution management*).

- **Goal:** Optimize the performance according to a given metric (performance model):
  - Global Throughput
  - Resource usage
  - Application – Stand-alone, HPC, HTC and self-adaptive
  - User usage

- **Grid characteristics**
  - Heterogeneity (job requirements)
  - Dynamism (high fault rate, load, availability, price)
  - Site autonomy

What is GridWay?

The GridWay meta-scheduler is a scheduler virtualization layer on top of basic Globus services (GRAM, MDS & GridFTP)

For the user

– A LRM-like environment for submitting, monitoring, and controlling jobs

For the developer

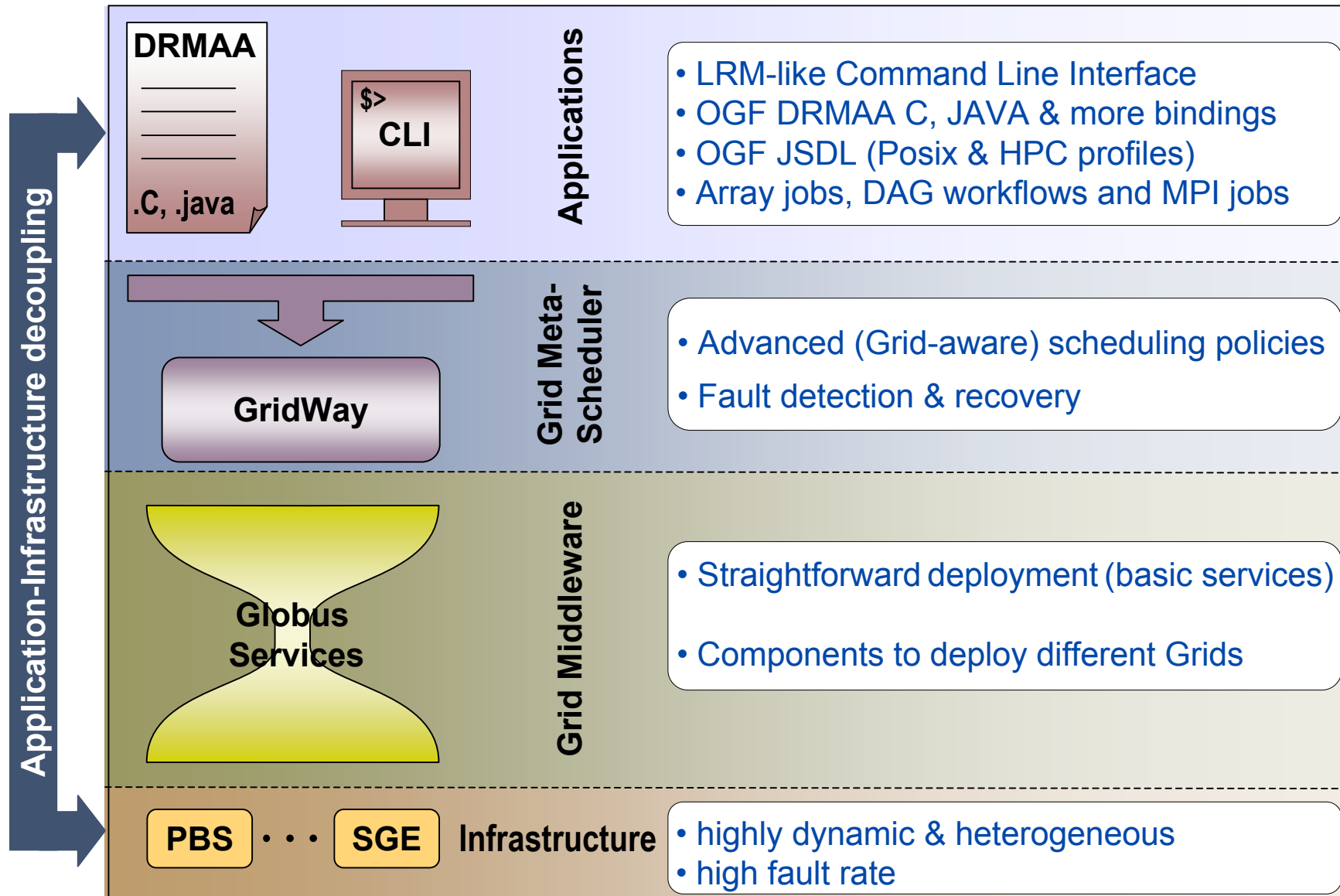– An standard-base development framework for Grid Applications

For the sysadmin

– A policy-driven job scheduler

– User-side Grid Accounting

For the Grid architect / solution provider

– A modular component to use different infrastructures

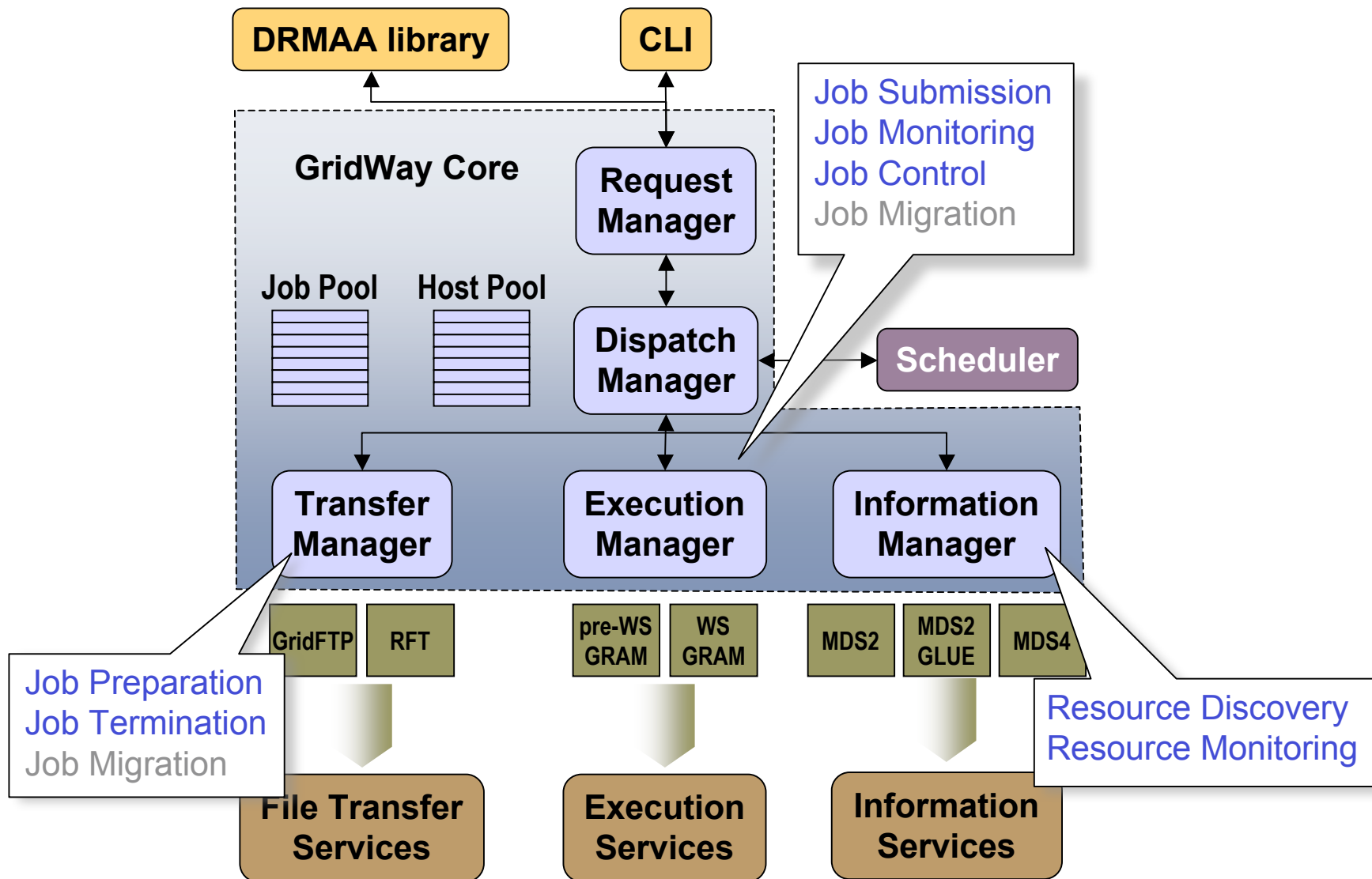– A key component to deploy different Grids (enterprise, partner, utility…)

Architecture

the globus alliance
www.globus.org

DRMAA
.C, .java

$> **CLI**

Applications

- LRM-like Command Line Interface
- OGF DRMAA C, JAVA & more bindings
- OGF JSDL (Posix & HPC profiles)
- Array jobs, DAG workflows and MPI jobs

GridWay

Grid Meta-Scheduler

- Advanced (Grid-aware) scheduling policies
- Fault detection & recovery

Globus Services

Grid Middleware

- Straightforward deployment (basic services)
- Components to deploy different Grids

PBS · · · **SGE** **Infrastructure**

- highly dynamic & heterogeneous
- high fault rate

Application-Infrastructure decoupling

Components

Scheduling Policies

Resource Policies
- Rank Expressions
- Fixed Priority
- User Usage History
- Failure Rate

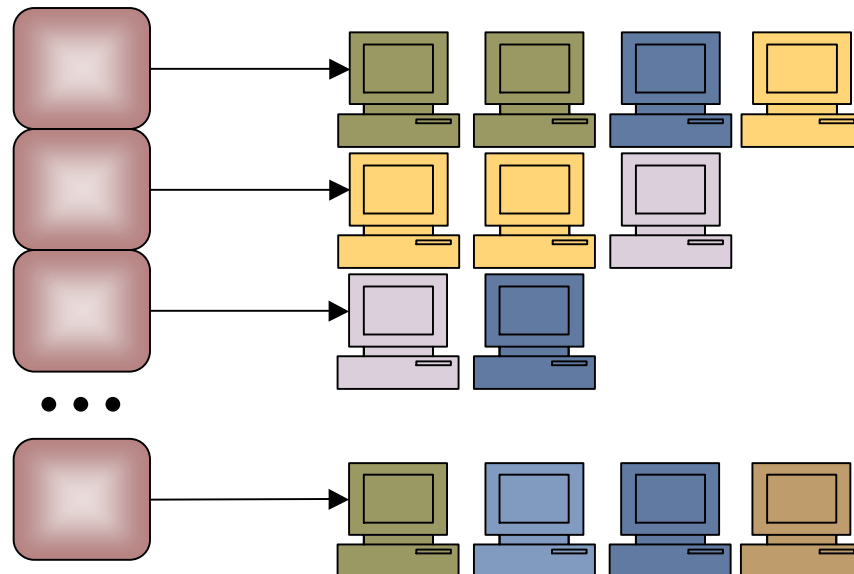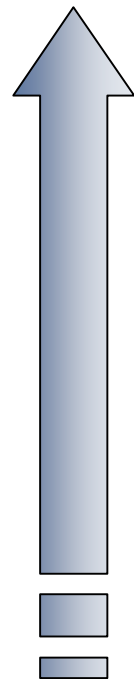**Grid Scheduling = Job + Resource Policies**

Pending Jobs

Matching Resources for each job (user)

Job Policies

- Fixed Priority
- Urgent Jobs
- User Share
- Deadline
- Waiting Time

Installation

1. ## Installing GridWay standalone

   - Uncompress tarball -> gw-<version>.tar.gz
   - ./configure
     - There are many options -- check them out in the manual
   - make
   - make install

2. ## Enabling GridWay in Globus

   - ./configure --enable-gridway

More information in the Installation & Configuration Guide
www.gridway.org/documentation/guides.php

Configuration

- ## $GW_LOCATION/etc/gwd.conf
  - Configuration options for the GridWay daemon (GWD)

- ## $GW_LOCATION/etc/sched.conf
  - Configuration options for GridWay built-in scheduling policies

- ## $GW_LOCATION/etc/job_template.default
  - Default values for job template

- ## $GW_LOCATION/etc/gwrc
  - Default environment variables for MADs

More information in the Installation & Configuration Guide
www.gridway.org/documentation/guides.php

Enterprise Grids

Characteristics

- – "Small" scale infrastructures (campus/enterprise) with one meta-scheduler instance
- – Resources within the same administration domain that may be running different LRMS and be geographically distributed
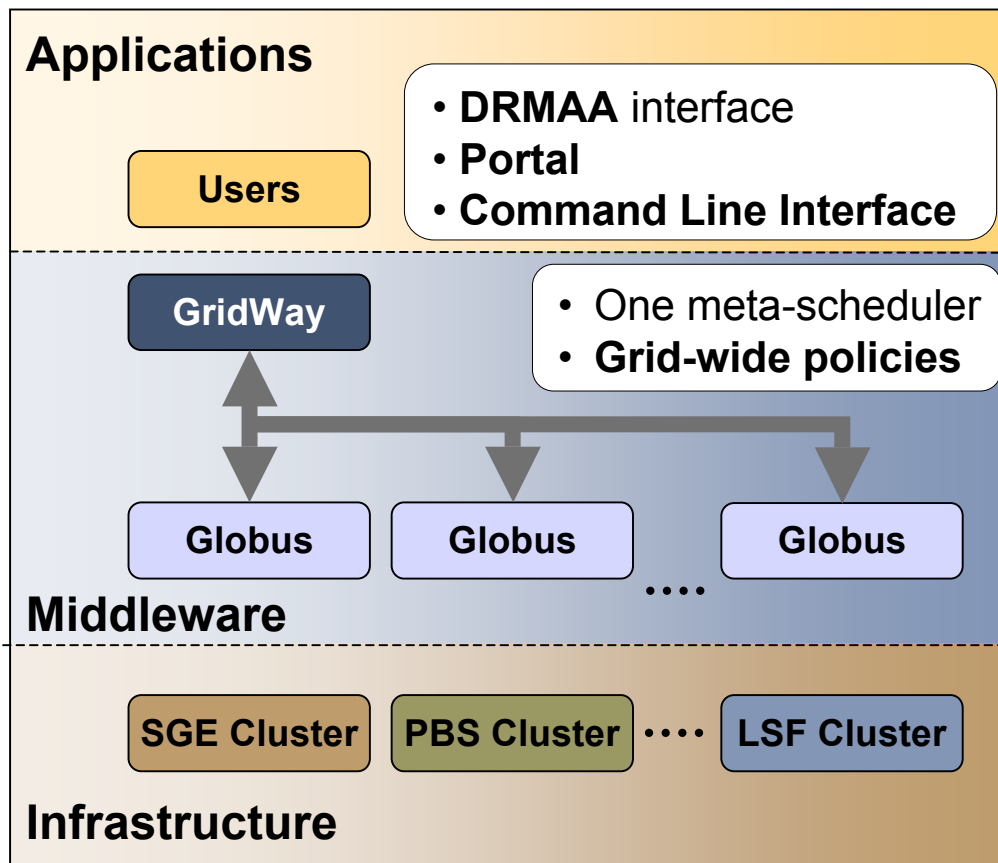
Goal & Benefits

- – Integrate heterogeneous systems
- – Improve return of IT investment
- – Performance/Usage maximization

Enterprise Grids

Architecture

Applications

Users

- **DRMAA** interface
- **Portal**
- **Command Line Interface**

GridWay

- One meta-scheduler
- **Grid-wide policies**

Globus   Globus   ....   Globus

Middleware

SGE Cluster   PBS Cluster   ....   LSF Cluster

Infrastructure

Examples

European Space Astronomy Center
- Data Analysis from space missions
- DRMAA

·eesa

UABGrid, University of Alabama
- Bioinformatics applications

UAB THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

Partner Grids

Characteristics

– "Large" scale infrastructures with one or several meta-schedulers

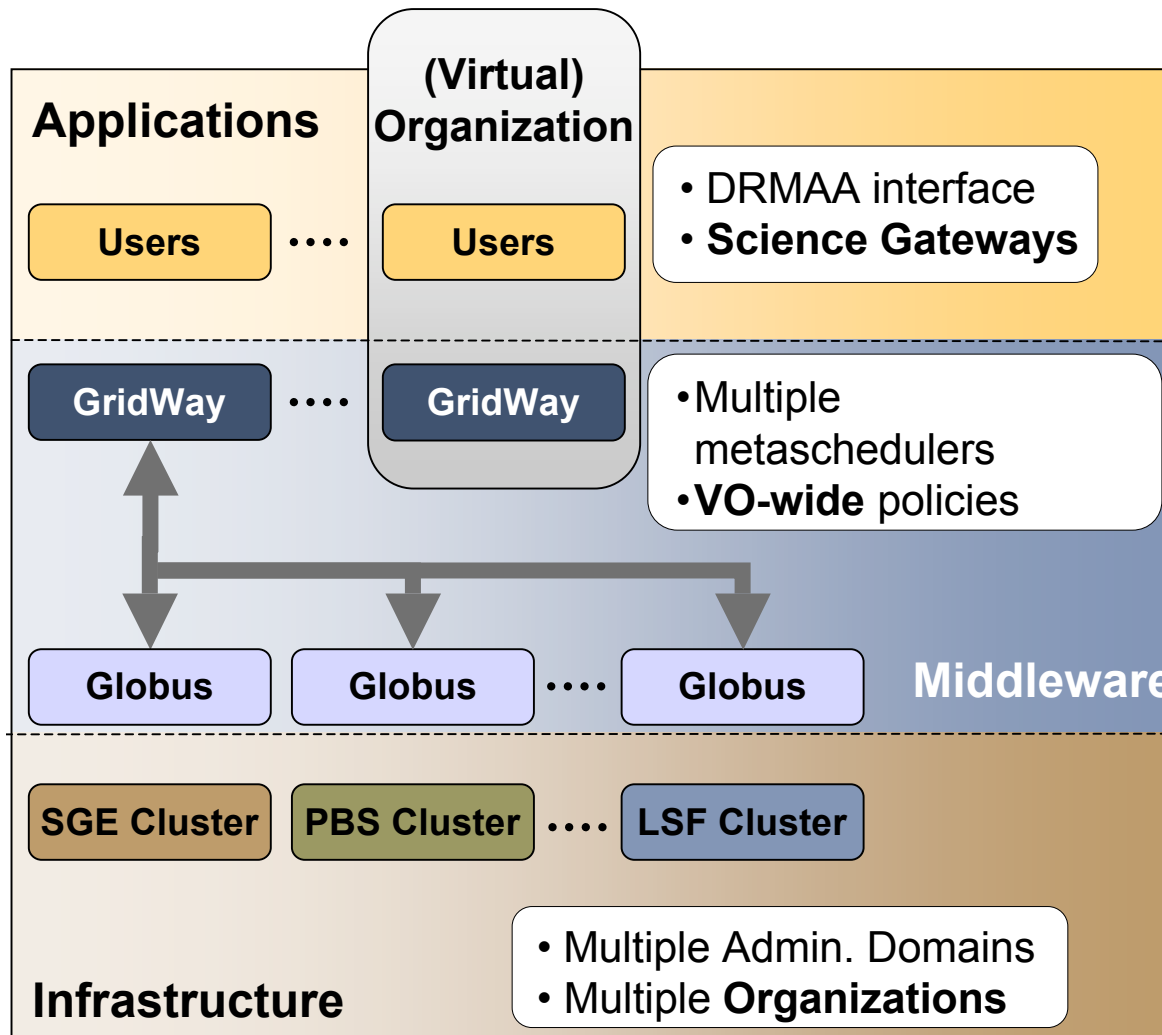– Resources belong to different administrative domains

Goal & Benefits

– Large-scale, secure and reliable sharing of resources

– Support collaborative projects

– Access to higher computing power to satisfy peak demands

Partner Grids

Architecture

Applications

(Virtual) Organization

Users · · · · **Users**

- DRMAA interface
- **Science Gateways**

GridWay · · · · **GridWay**

- Multiple metaschedulers
- **VO-wide** policies

Globus **Globus** · · · · **Globus** **Middleware**

SGE Cluster **PBS Cluster** · · · · **LSF Cluster**

- Multiple Admin. Domains
- Multiple **Organizations**

Infrastructure

Examples

EGEE-II
- gLite-LHC interoperability
- Virtual Organizations
  Fusion: Massive Ray Tracing
  Biomed: CD-HIT (Worflow)
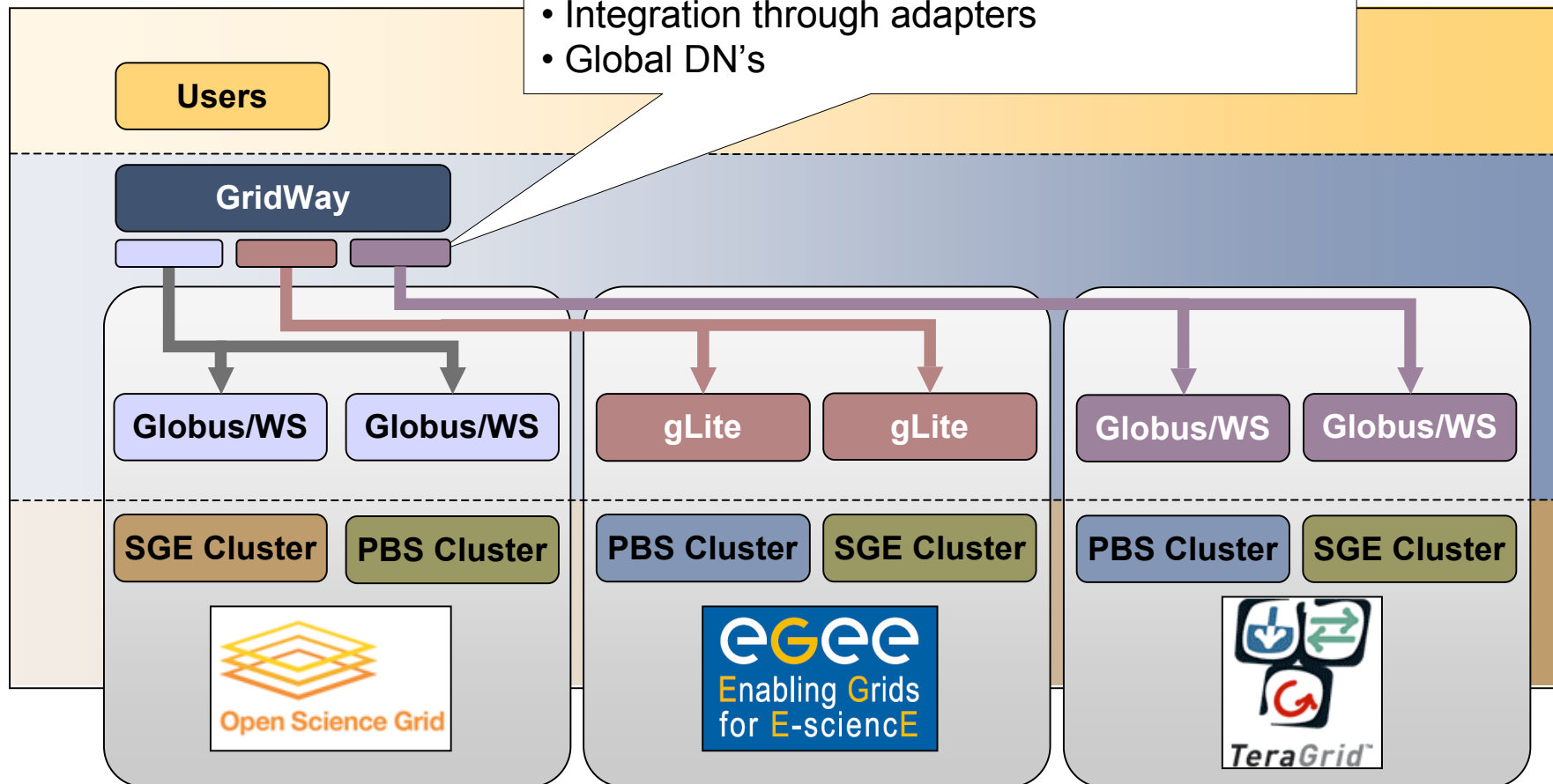
AstroGrid-D, German Astronomy Community Grid
- Supercomputing resources
- Astronomy-specific resources
- GRAM interface

A Tool for Interoperability

the globus alliance
www.globus.org

- Different Middlewares (e.g. WS and pre-WS)
- Different Data/Execution architectures
- Different Information models
- Integration through adapters
- Global DN's

Users

GridWay

| Globus/WS | Globus/WS | gLite | gLite | Globus/WS | Globus/WS |

| SGE Cluster | PBS Cluster | PBS Cluster | SGE Cluster | PBS Cluster | SGE Cluster |

Open Science Grid

eGee
Enabling Grids
for E-sciencE

TeraGrid

History

- Started in **2002**, as a research only effort (releases were distributed on request)

- First open source release (v4.0) in **January 2005** (Apache license v2.0)

- In **2006** started the incubation - ended in January 2007 (the first incubator to become a Globus project)

- In June **2007** GridWay became part of the Globus Toolkit

- Since January 2005, more than 1000 downloads from 80 different countries, 25% are private companies and 75% are universities and research centers.

- Best-effort support provided

History

- Community – **Open Source Project**.

- Adheres to Globus Development Philosophy

- Development Infrastructure (thanks to Globus Project!)

  - Mailing Lists

  - Bugzilla

  - CVS

- You are very welcome to contribute:

  - Reporting Bugs (gridway-user@globus.org)

  - Making feature requests (gridway-user@globus.org)

  - Contributing with your own developments (gridway-dev@globus.org)

Questions?

Using GridWay: CLI

Constantino Vázquez and Eduardo Huedo
dsa-research.org

Open Grid Forum 22
Boston, February 27, 2008

Contents

- **User set up**

- **Job Definition**

- **Job Life-cycle**

- **Job Submission**

- **Job & Host Monitoring**

- **Job Control**

- **Sample Session**

User Set Up

- Usually in a multi-user setting (single-user also possible)

- User environment (sh)
  - `export GW_LOCATION=/usr/local/gw`
  - `export PATH=$PATH:$GW_LOCATION/bin`
  - `CLASSPATH=$GW_LOCATION/lib/drmaa.jar:$CLASSPATH`
  - `LD_LIBRARY_PATH=$GW_LOCATION/lib:$LD_LIBRARY_PATH`

- Check $GW_LOCATION
  - `etc` configuration files
  - `share/doc` documents (www.gridway.org)
  - `share/examples` templates and howtos
  - `var` log information (debugging)

Accounts

- `cepheus.dacya.ucm.es - accessible through ssh`

- `gwtutorialXX`

- `gwtutorialXX`

    `where XX=00,01,02,03…`

- `Certificate passphrase: gridcv07`

- `$HOME/examples/drmaa`

  - `drmaa_c`
  - `drmaa_java`
  - `drmaa_perl`
  - `drmaa_ruby`
  - `drmaa_python`

Job Definition

- Each job is defined within an **experiment directory** (default paths)

- **Execution variables**

    - `EXECUTABLE = bin.${ARCH }`

    - `ARGUMENTS = ${TASK_ID }`

    - `ENVIRONMENT = SCRATCH_DIR=/tmp` (Also GW_* vars. are set)

- **I/O files relative to exp dir (also abs path, file://, gsiftp://, http://)**

    - `INPUT_FILES = param.${TASK_ID } param, inputfile`

    - `OUTPUT_FILES = outputfile, bin bin. ${ARCH }`

- **Standard streams**

    - `STDIN_FILE = /dev/null`

    - `STDOUT_FILE = stdout_file.${JOB_ID }`

    - `STDERR_FILE = stderr_file.${JOB_ID }`

Job Definition

- **Resource selection parameters**
  - `REQUIREMENTS = ARCH = "i686" & CPU_MHZ > 1000`
  - `RANK = (CPU_MHZ * 2) + FREE_MEM_MB`

- **Job Type**
  - `TYPE = mpi`
  - `NP = 16`

- **Advanced definition parameters**
  - Checkpointing parameters
  - Failure handling
  - Performance
  - Re-scheduling
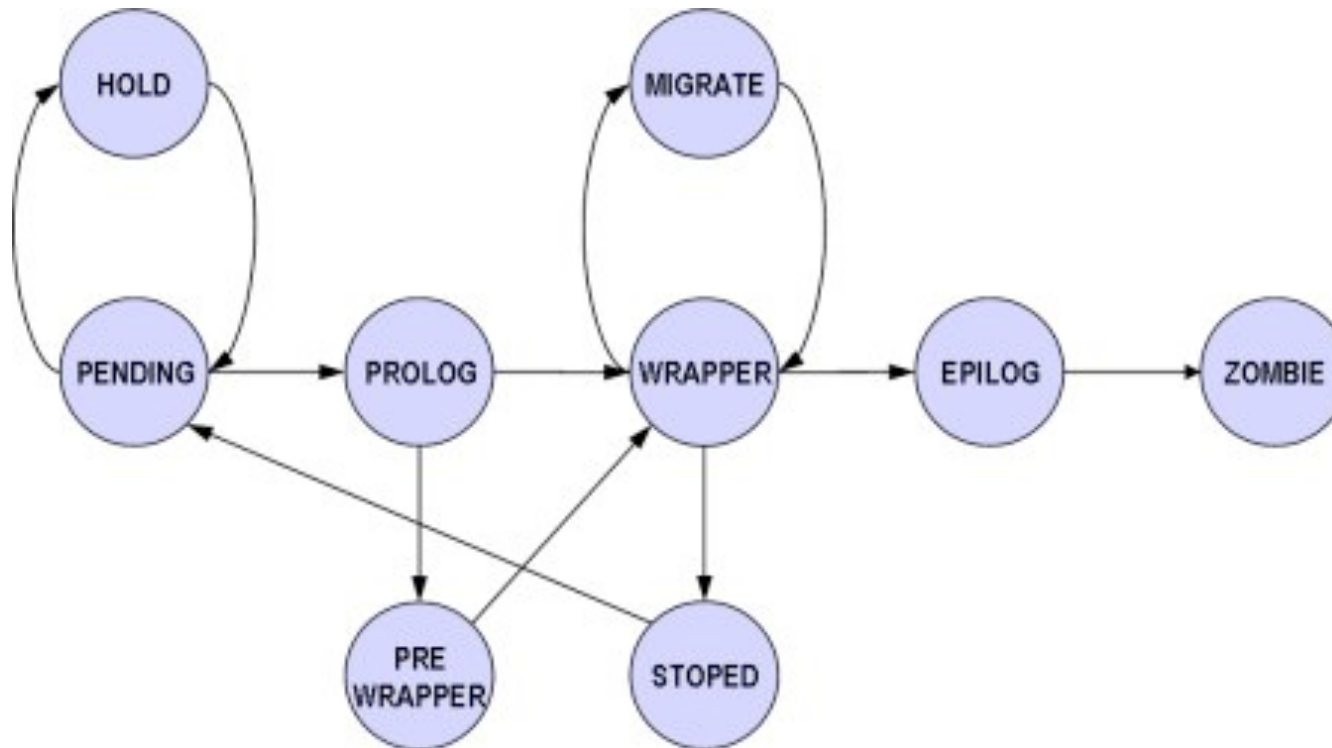 - Execution Configuration

Job Definition

Job Submission Description Language

- describing the job requirements for submission to resources (equivalent to job templates)

- OGF standard (*https://forge.gridforum.org/sf/projects/jsdl-wg*)

- `jsdl2gw` command to do the translation

```
…
<jsdl:Application>
 <jsdl:ApplicationName>ls</jsdl:ApplicationName>
  <jsdl-posix:POSIXApplication>
  <jsdl-posix:Executable>/bin/ls</jsdl-posix:Executable>
  <jsdl-posix:Argument>-la file.txt</jsdl-posix:Argument>
  <jsdl-posix:Environment name="PATH">/sbin</jsdl-posix:Environment>
…
```

Job Life-cycle

Job Submission

- Simple Jobs

```
$ gwsubmit example/jt
```
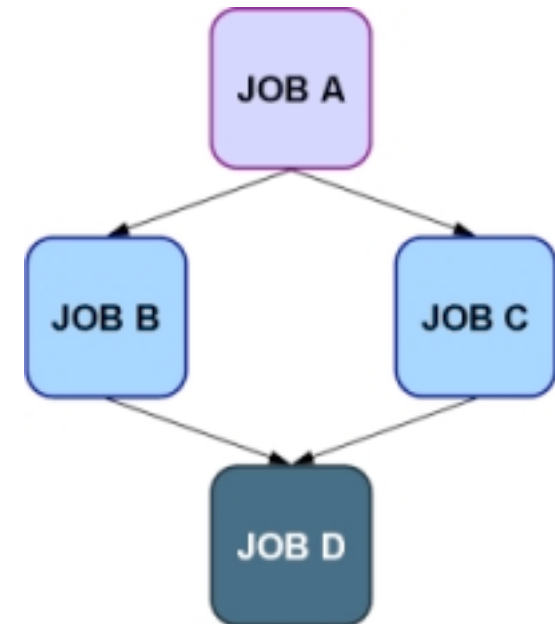
- Array Jobs ($TASK_ID and custom parametric var.)

```
$ gwsubmit -v  -n 4  pi.jt
ARRAY ID: 0
TASK JOB
0     3
1     4
2     5
3     6
```

Job Submission

- ### DAG Workflows

```
• $ gwsubmit -v -t A.jt
  JOB ID: 5

• $ gwsubmit -v -t B.jt -d "5"
  JOB ID: 6

• $ gwsubmit -v -t C.jt -d "5"
  JOB ID: 7

• $ gwsubmit -t C.jt -d "6 7"
```


- **from GW5.2 onwards, you can use dagman workflows with gwdagman…**

Job & Host Monitoring

- ● Use gwhost command to see available resources:

```
HID PRIO  OS          ARCH    MHZ %CPU   MEM(F/T)  DISK(F/T)    N(U/F/T) LRMS    HOSTNAME
0   1     Linux2.6 x86      3216  0     44/2027   26742/71812  0/0/2    Fork    cygnus
1   1                         0   0        0/0             0/0 0/0/0            orion
2   1     Linux2.6 x86_6 2211  100 819/1003  27083/77844  0/2/4    PBS     hydrus
3   1     Linux2.6 x86      3216  163 393/2027  11257/11812  0/2/2    Fork    draco
4   1     Linux2.6 x86_6 2211   66 943/1003  72485/77844  0/5/5    SGE     aquila
```

- ● and get more detailed information specifying a Host ID:

```
$ gwhost 0
HID PRIO  OS          ARCH    MHZ %CPU   MEM(F/T)   DISK(F/T)   N(U/F/T) LRMS     HOSTNAME
0   1     Linux2.6 x86      3216   0     50/2027   6393/18812  0/0/2 Fork      cygnus

QUEUENAME            SL(F/T) WALLT CPUT   COUNT MAXR   MAXQ   STATUS    DISPATCH
PRIORITY
default              0/2     0     -1     0     -1     0      enabled   NULL          0
```

Job & Host Monitoring

- ● Resources that match job requirements with gwhost -m 0:

```
$ gwhost -m 0
HID QNAME        RANK    PRIO    SLOTS   HOSTNAME
0    default     0       1       0       cygnus.dacya.ucm.es
2    default     0       1       3       hydrus.dacya.ucm.es
2    qlong       0       1       3       hydrus.dacya.ucm.es
4    all.q       0       1       3       aquila.dacya.ucm.es
```

- ● Follow the evolution of the job with gwps & gwhistory:

```
$ gwps
USER      JID DM    EM     START     END       EXEC     XFER     EXIT NAME   HOST
gwtut00   0   done  ----   20:16:28 20:18:16 0:00:55 0:00:08 0       stdin aquila/SGE
tinova    1   done  ----   12:26:46 12:31:15 0:03:55 0:00:08 0       stdin hydrus/PBS
tinova    2   pend  ----   12:38:38 --:--:-- 0:00:00 0:00:00 --      t.jt  --
```

```
$ gwhistory 4
HID START       END        PROLOG   WRAPPER  EPILOG   MIGR     REASON QUEUE    HOST
2    12:58:04 12:58:16 0:00:06 0:00:04 0:00:02 0:00:00 ----     default
hydrus/PBS
```

Job control

- **Jobs Signals**

  – Kill (default, if no signal specified).

  – Stop job.

  – Resume job.

  – Hold job.

  – Release job.

  – Re-schedule job.

  – Hard kill

```
gwkill [-h] [-a] [-k | -t | -o | -s | -r | -l | -9] <job_id \
[job_id2 ...] | -A array_id>
```

- **Synchronization**

```
gwwait [-h] [-a] [-v] [-k] <job_id...| -A array_id>
```

Sample Session

```
$ grid-proxy-init
   Creating proxy .+++++++++++
   ..................................++++++++++
    Done
$ vi job.template
   EXECUTABLE=/bin/ls
   STDOUT=stdout.$(JOB_ID)
   STDERR=stderr.$(JOB_ID)
$ gwsubmit job.template
$ gwhost -m 0
   HID QNAME         RANK  PRIO  SLOTS HOSTNAME
   0   default       0     1     0     cygnus.dacya.ucm.es
   1   default       0     1     3     hydrus.dacya.ucm.es
$ gwps -c 1
   USER     JID DM   EM    EXEC     XFER     EXIT NAME            HOST
   gwtut00 0   done ---- 0:00:05 0:00:04 0    job.template hydrus/PBS
$ls -lt stderr.4 stdout.4
   -rw-r--r-- 1 gwtut00 gwtut00 0  2007-09-07 12:58 stderr.4
   -rw-r--r-- 1 gwtut00 gwtut00 72 2007-09-07 12:58 stdout.4
$ cat stdout.4
   job.env
   stderr.execution
   stderr.wrapper
```
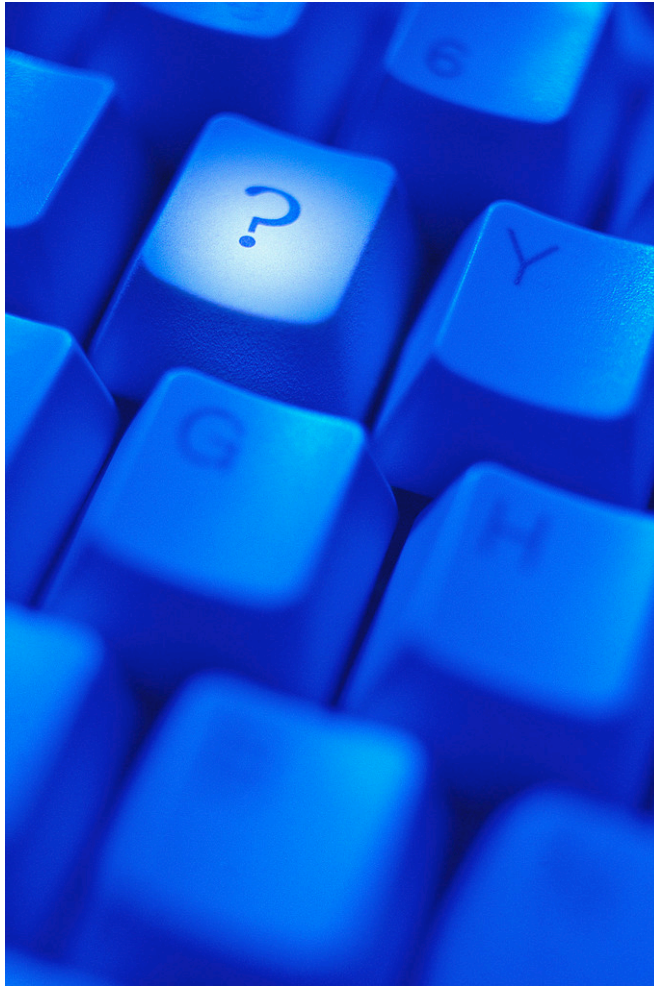
For More Information

- User guide
- Command reference guide

http://www.gridway.org/documentation/guides.php

Questions?

Developing Applications with GridWay: DRMAA

Constantino Vázquez and Eduardo Huedo

dsa-research.org

Open Grid Forum 22

Boston, February 27, 2008

Contents

- Introduction

- Program Structure and Compilation

- DRMAA Directives and Functions

- More Information

What is DRMAA?

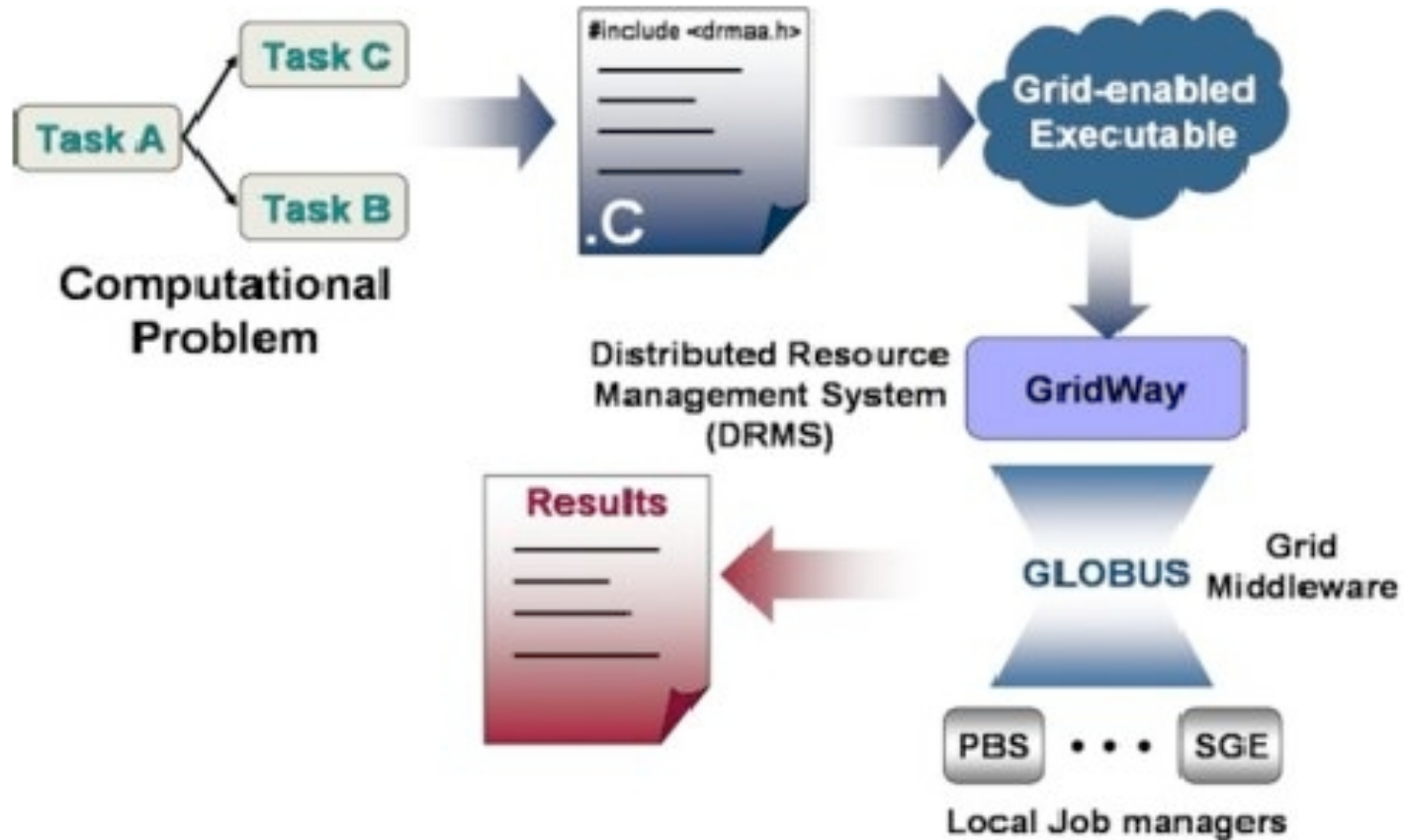- Distributed Resource Management Application API
  - http://www.drmaa.org
- Open Grid Forum Standard
- Homogeneous interface to different Distributed Resource Managers (DRM):
  - SGE, Condor, PBS/Torque…
- GridWay implementation
  - C & JAVA
  - Perl, Ruby & Python
    - > check the development release GW 5.3
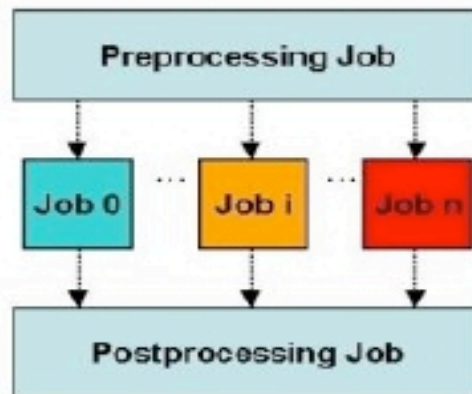
Programming Model
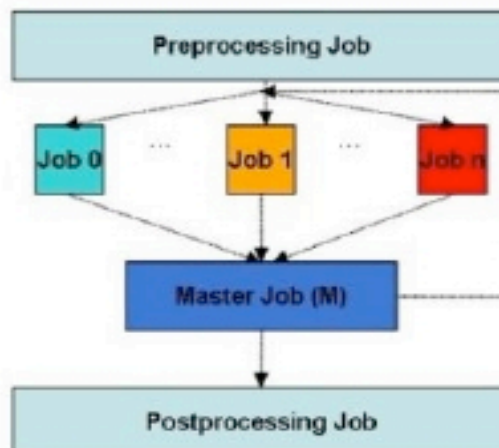
Application Profiles

- Embarrasingly Distributed


```
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, &stat, 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,&stat, timeout, rusage, err);
rc = drmaa_exit(err_diag);
```

- Master-Worker


```
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, &stat, 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, &stat, timeout, rusage,
err_diag);
 rc = drmaa_wexitstatus(&exitstatus, stat, err_diag);
}
rc = drmaa_exit(err_diag);
```

Program Structure and Compilation

- Include the DRMAA library:

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

- Verify the following environment variable:

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

- Include the compiling and linking options for DRMAA:

```
-L $GW_LOCATION/lib
-I $GW_LOCATION/include
-ldrmaa
```

- Example:

```
$ gcc example.c -L $GW_LOCATION/lib \
    -I $GW_LOCATION/include -ldrmaa -o example
```

Session Management

- ## Initialization and finalization

  int **drmaa_init** (const char *contact, char *error_diagnosis, size_t error_diag_len);

  int **drmaa_exit** (char *error_diagnosis, size_t error_diag_len);

Auxiliary Functions

- Getting Information

  const char * **drmaa_strerror** (int drmaa_errno);

  int **drmaa_get_contact** (char *contact, size_t contact_len, char *error_diagnosis, size_t error_diag_len);

  int **drmaa_version** (unsigned int *major, unsigned int *minor, char *error_diagnosis, size_t error_diag_len);

  int **drmaa_get_DRM_system** (char *drm_system, size_t drm_system_len, char *error_diagnosis, size_t error_diag_len);

  int **drmaa_get_DRMAA_implementation** (char *drmaa_impl, size_t drmaa_impl_len, char *error_diagnosis, size_t error_diag_len);

**Hands on!
Howto1**

Job Template

● Allocation and Deletion

int **drmaa_allocate_job_template** (drmaa_job_template_t **jt, char
*error_diagnosis, size_t error_diag_len);

int **drmaa_delete_job_template** (drmaa_job_template_t *jt, char *error_diagnosis,
size_t error_diag_len);

● Parameter Setting/Getting

int **drmaa_set_attribute** (drmaa_job_template_t *jt, const char *name, const char
*value, char *error_diagnosis, size_t error_diag_len);

int **drmaa_get_attribute** (drmaa_job_template_t *jt, const char *name, char *value,
size_t value_len, char *error_diagnosis, size_t error_diag_len);

int **drmaa_set_vector_attribute** (drmaa_job_template_t *jt, const char *name, const
char *value[ ], char *error_diagnosis, size_t error_diag_len);

int **drmaa_get_vector_attribute** (drmaa_job_template_t *jt, const char *name,
drmaa_attr_values_t **values, char *error_diagnosis, size_t error_diag_len);

int **drmaa_get_attribute_names** (drmaa_attr_names_t **values, char
*error_diagnosis, size_t error_diag_len);

int **drmaa_get_vector_attribute_names** (drmaa_attr_names_t **values, char
*error_diagnosis, size_t error_diag_len);

Job Template Compilation

DRMAA_REMOTE_COMMAND

DRMAA_V_ARGV

DRMAA_V_ENV

DRMAA_INPUT_PATH

DRMAA_OUTPUT_PATH

DRMAA_ERROR_PATH

DRMAA_WD

DRMAA_JOB_NAME

DRMAA_JS_STATE

DRMAA_SUBMISSION_STATE_ACTIVE

DRMAA_SUBMISSION_STATE_HOLD

DRMAA_PLACEHOLDER_HD

DRMAA_PLACEHOLDER_INCR

DRMAA_PLACEHOLDER_WD

DRMAA_DEADLINE_TIME

DRMAA_DEADLINE_TIME

GridWay Specific Job Template Compilation

DRMAA_GW_TOTAL_TASKS

DRMAA_GW_JOB_ID

DRMAA_GW_TASK_ID

DRMAA_GW_PARAM

DRMAA_GW_MAX_PARAM

DRMAA_GW_ARCH

DRMAA_V_GW_INPUT_FILES

DRMAA_V_GW_OUTPUT_FILES

DRMAA_V_GW_RESTART_FILES

DRMAA_GW_RESCHEDULE_ON_FAILURE

DRMAA_GW_NUMBER_OF_RETRIES

DRMAA_GW_RANK

DRMAA_GW_REQUIREMENTS

DRMAA_GW_TYPE

DRMAA_GW_TYPE_SINGLE

DRMAA_GW_TYPE_MPI

DRMAA_GW_NP

Job Submission

- ● Simple Job Submission

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);

Job Synchronize and Wait

- ## Wait for Job Completion

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 **rusage, char *error_diagnosis, size_t error_diag_len);

> **job_id** value could be DRMAA_JOB_IDS_SESSION_ANY or DRMAA_JOB_IDS_SESSION_ALL

Auxiliary Functions

- Interpreting Job Status Code

  int **drmaa_wexitstatus** (int *exit_status, int stat, char *error_diagnosis, size_t
       error_diag_len);
  int **drmaa_wifexited** (int *exited, int stat, char *error_diagnosis, size_t
       error_diag_len);
  int **drmaa_wifsignaled** (int *signaled, int stat, char *error_diagnosis, size_t
       error_diag_len);
  int **drmaa_wtermsig** (char *signal, size_t signal_len, int stat, char *error_diagnosis,
       size_t error_diag_len);

Helper Functions

- ## String Lists

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

int **drmaa_get_next_attr_value** (drmaa_attr_values_t *values, char *value, size_t value_len);

int **drmaa_get_num_attr_names** (drmaa_attr_names_t *values, size_t *size);

int **drmaa_get_num_attr_values** (drmaa_attr_values_t *values, size_t *size);

void **drmaa_release_attr_names** (drmaa_attr_names_t *values);

void **drmaa_release_attr_values** (drmaa_attr_values_t *values);

**Hands on!
Howto2**

Job Status and Control

- ## Get Job Status

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

> **remote_ps** returns DRMAA_PS_QUEUED_ACTIVE, DRMAA_PS_RUNNING, DRMAA_PS_USER_ON_HOLD, DRMAA_PS_DONE, DRMAA_PS_FAILED or DRMAA_PS_UNDETERMINED

- ## Job Control

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

> **action** value can be DRMAA_CONTROL_SUSPEND, DRMAA_CONTROL_RESUME, DRMAA_CONTROL_TERMINATE, DRMAA_CONTROL_HOLD or DRMAA_CONTROL_RELEASE

Job Synchronize and Wait

- ## Synchronize Jobs

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

  > **timeout** value could be DRMAA_TIMEOUT_WAIT_FOREVER or DRMAA_TIMEOUT_NO_WAIT

Hands on! Howto3

Job Submission

- ## Bulk Job Submission

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);

Helper Functions

- ## String Lists

int **drmaa_get_next_job_id** (drmaa_job_ids_t *values, char *value, size_t
    value_len);

int **drmaa_get_num_job_ids** (drmaa_job_ids_t *values, size_t *size);

void **drmaa_release_job_ids** (drmaa_job_ids_t *values);

**Hands on!
Howto4**

DRMAA Error Codes

DRMAA_ERRNO_SUCCESS

DRMAA_ERRNO_INTERNAL_ERROR

DRMAA_ERRNO_DRM_COMMUNICATION_FAILURE

DRMAA_ERRNO_AUTH_FAILURE

DRMAA_ERRNO_INVALID_ARGUMENT

DRMAA_ERRNO_NO_ACTIVE_SESSION

DRMAA_ERRNO_NO_MEMORY

DRMAA_ERRNO_INVALID_CONTACT_STRING

DRMAA_ERRNO_DEFAULT_CONTACT_STRING_ERROR

DRMAA_ERRNO_DRMS_INIT_FAILED

DRMAA_ERRNO_ALREADY_ACTIVE_SESSION

DRMAA_ERRNO_DRMS_EXIT_ERROR

DRMAA_ERRNO_INVALID_ATTRIBUTE_FORMAT

DRMAA_ERRNO_INVALID_ATTRIBUTE_VALUE

DRMAA_ERRNO_CONFLICTING_ATTRIBUTE_VALUES

DRMAA_ERRNO_TRY_LATER

DRMAA_ERRNO_DENIED_BY_DRM

…

For More Information

- Application Developer Guide (DRMAA C/JAVA bindings)
- DRMAA C Howtos
- DRMAA JAVA Howtos
- DRMAA C Reference
- DRMAA JAVA Reference
- DRMAA JAVA TestSuite
- And info about the scripting languages bindings

http://www.gridway.org/documentation/guides.php

Questions?