GWD-E

Jose Herrera Universidad Complutense de Madrid
Eduardo Huedo Universidad Complutense de Madrid
Ruben S. Montero Universidad Complutense de Madrid
Ignacio M. Llorente Universidad Complutense de Madrid

Distributed Resource Management
Application API (DRMAA) Working Group

2/22/2007

GridWay DRMAA 1.0 Implementation – Experience Report

Status of This Document

This document provides information to the Grid community about the adoption of the OGF specification GFD-R-P.022 in the GridWay meta-scheduler. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

This document reports about experiences made with running DRMAA working group compliance test for C bindings against GridWay 5.2 implementation of the Distributed Resource Management Application API (DRMAA). GridWay (www.gridway.org) is an open source meta-scheduling technology that, on top of Globus services, performs job execution management and resource brokering. GridWay performs all the job scheduling and submission steps transparently to the end user and adapts job execution to changing Grid conditions by providing fault recovery mechanisms, dynamic scheduling, migration on-request, and opportunistic migration.

This document also includes some implementation details of the DRMAA API, and remarks some special considerations made to implement the DRMAA standard on a Grid (Globus based). The main difficulties arise because the nature of the Grid itself, as opposed to a classical Distributed Resource Management System (DRMS), which provides a single system image where all resources are in the same administration domain.

Contents

1 Introduction

The Distributed Resource Management Application API specification (GFD-R.P.022) specifies a generalized API to Distributed Resource Management Systems (DRMS) in order to facilitate integration of application programs. Soon after the DRMAA API reached "proposed recommendation" status various DRM vendors and Grid community oriented open source projects started implementing DRMAA bindings for C language. This document reports about experiences made with running DRMAA working group maintained compliance test against GridWay 5.2 implementation of the Distributed Resource Management Application API (DRMAA).

The DRMAA GridWay implementation is based on the Distributed Resource Management Application API C Bindings v1.0.

2 The GridWay Meta-scheduler System

GridWay is a widely-used metascheduling technology that performs job execution management and resource brokering, allowing unattended, reliable, and efficient execution of jobs, job arrays, and workflows on heterogeneous and dynamic Globus grids. GridWay performs all the job scheduling and submission steps transparently to the end user and adapts job execution to changing Grid conditions by providing dynamic scheduling, fault recovery mechanisms, migration on-request and opportunistic migration.

GridWay provides the following benefits to the different stakeholders involved in a Grid environment: (i) for project and infrastructure directors, GridWay is an open-source community project, adhering to Globus philosophy and guidelines for collaborative development; (ii) for system integrators, GridWay is highly modular, allowing adaptation to different grid infrastructures, and supports several OGF standards; (iii) for system managers, GridWay gives a scheduling framework similar to that found on local DRM systems, supporting resource accounting and the definition of scheduling policies; (iv) for application developers, GridWay implements the DRMAA API (C and JAVA bindings) OGF standard, assuring compatibility of applications with LRM systems that implement the standard, such as SGE, Condor or Torque; and (v) for end users, GridWay provides a LRM-like CLI for submitting, monitoring, synchronizing and controlling jobs that could be described using the JSDL OGF standard.

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, offering the highest functionality, quality of service and reliability on this kind of infrastructures.

The GridWay project started in September 2002. The first releases of the metascheduler were developed for research purposes in adaptive and dynamic scheduling and were only distributed on request in binary format. The first open source version, GridWay 4.0, and the project website were released in January 2005. Since then, the GridWay Metascheduler has been downloaded from more than a thousand of companies and research centres in 60 different countries. GridWay is being used as metascheduler in several infrastructures and has been referenced in research papers, market overviews and OGF documents. The last release, GridWay 5.2, is the result of the knowledge and experience gained through years of research and development and the feedback from our user community.

The GridWay project is being developed by the Distributed Systems Architecture Group from Universidad Complutense de Madrid. GridWay joined the dev.globus incubation process in May 2006, being the first ever project to escalate to a full Globus project in January 2007. GridWay so adheres to Globus philosophy and guidelines for collaborative development.

3 The GridWay DRMAA Library

3.1 Extend of GridWay DRMAA

The GridWay DRMAA distribution includes:

- A DRMAA include file, `drmaa.h`, placed in `$GW_LOCATION/include`
- A dynamic implementation library, `libdrmaa.so`, placed in `$GW_LOCATION/lib`
- A JAR package with the JAVA implementation of the DRMAA API, `drmaa.jar`, placed in `$GW_LOCATION/lib`
- A collection of How-to program examples in C and JAVA language, showing the main functionalities of the DRMAA API, at http://www.gridway.org/documentation/guides.php
- API Documentation for the C and JAVA implementations available online in html and pdf formats at http://www.gridway.org/documentation/guides.php
- DRMAA Programmers Tutorial in html and pdf formats available online at http://www.gridway.org/documentation/guides.php
- A JAVA test suite at http://www.gridway.org/documentation/guides.php

3.2 Implementation of Gridway DRMAA

In this section, we will review some particular aspects of the GridWay DRMAA implementation. The following remarks arise by the Grid middleware (Globus) in which GriWay is based on, and by the nature and philosophy of the Grid itself. In addition, some practical modifications have been introduced for the shake of usability. Finally, there are also some limitations by the functionality offered by GridWay.

- *JOB STATES*

  The following job states are not defined within the GridWay system, and therefore are never returned by the `drmaa_job_ps()` function call:

  - `DRMAA_PS_SYSTEM_ON_HOLD`
  - `DRMAA_PS_USER_SYSTEM_ON_HOLD`
  - `DRMAA_PS_SYSTEM_SUSPENDED`
  - `DRMAA_PS_USER_SYSTEM_SUSPENDED.`

- *JOB TEMPLATE ROUTINES*

  `DRMAA_WD`: Refers to a local directory in the DRMAA GridWay implementation. In a Grid, the working directory on the remote host can not be defined as its existence can not be guaranteed. The local working directory is "recreated" by GridWay in the remote host, and it will be the working directory of the job on the execution host. File names can be therefore safely generated by using the DRMAA_WD predefined constant.

  `DRMAA_NATIVE_SPECIFICATION:` GridWay does not use the `NATIVE_SPECIFICATION` to generate specific template directives. Native strings with DRM specific attributes (eg. input files) could be too long and difficult to handle in a program. Moreover, the DRMAA API provides a powerful interface to handle job template attributes. The GridWay implementation takes advantage of this interface by defining its own preprocessor directives, which can be easily handle with the `drmaa_get/set_attribute()` and `drmaa_get/set_vector_attribute()` functions. Please note, that this does not reduce portability of program, since a program using the

`DRMAA_NATIVE_SPECIFICATION` attribute will be only compatible with that DRM. Conditional compilation of job template set up fragments can be used to generate DRMAA binaries for different DRM.

DRMAA mandatory job template attributes not relevant for GridWay can be safely used with the get/set DRMAA function calls. However, GridWay will not consider them.

File naming: In a Grid, several services or protocols can be used to transfer files, so GridWay uses the following file naming pattern:

```
[protocol://hostname/path/]filename
```

The current GridWay implementation supports the following protocols: `gsiftp://`, `https://`, `http://`, also `file://` can be used to refer to local filenames. This syntax is only enforced for GridWay specific variables, i.e. `DRMAA_V_GW_INPUT_FILES`, `DRMAA_V_GW_OUTPUT_FILES, and DRMAA_V_GW_RESTART_FILES.`

The DRMAA file naming pattern can also be used to specify the `DRMAA_INPUT_PATH`, `DRMAA_OUTPUT_PATH` and `DRMAA_ERROR_PATH` attributes:

```
[hostname]:file
```

In this case, the gsiftp protocol will be used to transfer files from/to hostname resource.

- *JOB EXECUTION ROUTINES*

  `drmaa_run_bulk_job():` GridWay internally rescales the start-end range to [0,(total_tasks-1)]. The coherence of `start`, `end` and `incr` values are not checked by `drmaa_run_bulk_job().` The calling program SHOULD guarantee their coherence.

- *JOB CONTROL ROUTINES*

  The Job State Transition Diagram states that a job in a SUSPENDED state should return to the RUNNING_ACTIVE state. Current Grid middleware does not allow to suspend a job (as this is not allowed in general by the underlying DRM system). GridWay suspends a job by canceling it and retrieving any restart file to the client. When this job is resumed, it is re-scheduled (note that the previous host could not be available). Therefore, in the GridWay DRMAA implementation SUSPENDED jobs returns to the QUEUED_ACTIVE state.

- *drmaa_w FUNTIONS*

  `coredump()`: Current Grid middleware does not provide a method to determine if a program has generated a core file. This function always returns 0. Note also that its usefulness will be limited as the architecture of the remote host could be different to the architecture of the client host.

  `signaled(), termsig()`: Current Grid middleware does not provide a method to determine if a program has been signaled. However, GridWay executes a job within a wrapper (bash shell), as long as the user program follows the UNIX exit code convention, the DRMAA library is able to successfully detect signaled jobs and the signal name.

  `aborted():` A job is not able to enter the RUNNING_ACTIVE state if some input files cannot be transferred to the remote host, and therefore the jobs finish in the FAILED state. However if some output files can not be transferred back to the client the job also enters the FAILED state (this behavior is specified in the DRMAA job state transition diagram). GridWay is not able to distinguish these two situations, and so this function always returns 0.

- *THREAD SAFETY*

All the functions in the GridWay DRMAA library are thread safe. This thread-safety is provided at the following levels:

- Job Template: A Job template structure can be concurrently modified (set/get attribute functions) or used to submit jobs by several threads.
- Function Calls: Session management, job submission, job status and control, and job synchronization routines are thread-safe.

- GRIDWAY EXTENSIONS

  GridWay includes the following function not in DRMAA 1.0:

```
@param drmaa_state The state of a job as obtained with the
drmaa_ps() for which a string description is to be returned
const char *drmaa_gw_strstatus(int drmaa_state);
```

3.3 GridWay DRMAA JAVA bindings

The GridWay JAVA binding has been implemented with the JAVA Native Interface (JNI) so the same remarks made in this report also apply to the DRMAA JAVA bindings.

4 DRMAA Compliance Test

4.1 Requirements for compliance test

In order to execute the compilation test the following is needed:

- A functional Globus based testbed (only one execution host is needed to run the tests)
- A valid user certificate
- GridWay meta-scheduler

Compilation of the test suite:

```
$./configure CPPFLAGS="-I$GW_LOCATION/include/ -I./" \
 LDFLAGS=-L$GW_LOCATION/lib/
$ make all
```

4.2 Structure of compliance test

The DRMAA Test Suite version used is 1.4.0, that can be obtained from the following URL:

https://sourceforge.net/project/showfiles.php?group_id=139840

4.3 Running DRMAA C compliance test application with GridWay DRMAA

The following results have been obtained with GridWay version 5.2 on a testbed based on the Globus Toolkit version 4.0.3. The testbed includes three workstations (fork jobmanger), a PBS cluster, and a SGE cluster; both pre-WS and WS services have been tested.

In the table below, detailed execution results for all tests are presented; test modifications are detailed when appropriated. The following convention has been used:

- **S** (Success): The test can be successfully executed with GridWay DRMAA without modification.
- **F** (Failed): The test can not be successfully executed with GridWay.

TEST NAME	RES.	TEST NAME	RES.

ST_MULT_INIT	S	ST_INPUT_FILE_FAILURE	S
ST_MULT_EXIT	S	ST_OUTPUT_FILE_FAILURE	S
ST_SUPPORTED_ATTR	S	ST_ERROR_FILE_FAILURE	S
ST_SUPPORTED_VATTR	S	ST_SUBMIT_IN_HOLD_RELEASE	S
ST_VERSION	S	ST_SUBMIT_IN_HOLD_DELETE	S
ST_DRMAA_IMPL	S	ST_BULK_SUBMIT_IN_HOLD_SESSION_RELEASE	S
ST_CONTACT	S	ST_BULK_SUBMIT_IN_HOLD_SINGLE_RELEASE	S
ST_EMPTY_SESSION_WAIT	S	ST_BULK_SUBMIT_IN_HOLD_SESSION_DELETE	S
ST_EMPTY_SESSION_SYNCHRONIZE_DISPOSE	S	ST_SUBMIT_POLLING_SYNCHRONIZE_TIMEOUT	S
ST_EMPTY_SESSION_SYNCHRONIZE_NODISPOSE	S	ST_SUBMIT_POLLING_SYNCHRONIZE_ZEROTIMEOUT	S
ST_EMPTY_SESSION_CONTROL	S	ST_SUBMIT_POLLING_WAIT_ZEROTIMEOUT	S
ST_SUBMIT_WAIT	S	ST_SUBMIT_POLLING_WAIT_TIMEOUT	S
ST_SUBMIT_NO_RUN_WAIT	S	ST_ATTRIBUTE_CHANGE	S
ST_BULK_SUBMIT_WAIT	S	ST_SUBMIT_SUSPEND_RESUME_WAIT	F
ST_BULK_SINGLESUBMIT_WAIT_INDIVIDUAL	S	ST_USAGE_CHECK	S
ST_SUBMITMIXTURE_SYNC_ALL_DISPOSE	S	MT_SUBMIT_WAIT	S
ST_SUBMITMIXTURE_SYNC_ALL_NODISPOSE	S	MT_SUBMIT_BEFORE_INIT_WAIT	S
ST_SUBMITMIXTURE_SYNC_ALLIDS_DISPOSE	S	MT_EXIT_DURING_SUBMIT	S
ST_SUBMITMIXTURE_SYNC_ALLIDS_NODISPOSE	S	MT_SUBMIT_MT_WAIT	S
ST_EXIT_STATUS	S	MT_EXIT_DURING_SUBMIT_OR_WAIT	S
ST_SUBMIT_KILL_SIG	S		

The test ST_SUBMIT_SUSPEND_RESUME_WAIT fails as SUSPENDED jobs return to QUEUED_ACTIVE.

5 GridWay DRMAA end user reports and public records

Several GridWay users are using DRMAA to port their applications to large-scale Grid infrastructures (EGEE, UABgrid,...). Examples of application domains that are benefiting form GridWay DRMAA can be found at http://www.gridway.org/successstories/applicationporting.php. Some of theses works have been presented in international conferences and journals. A list of scientific papers on application porting using DRMAA can be found at http://www.gridway.org/research/publications.php.

6 Conclusion

The DRMAA GridWay implementation is fully usable, as its suitability to express distributed computations on a Grid, which represent typical scientific workloads, has been demonstrated on several research papers. Minor functionality issues required by the DRMAA can not be implemented on Globus Grids (core, and running from suspended). In addition, the DRMAA Test Suite has proven to be a valuable tool to test and develop DRMAA implementations, and a valuable source to interpret the DRMAA specifications.

7 Security Considerations

Security issues are not discussed in this document. For security considerations of the DRMAA specification, please refer to the GFD-R-P.022 document.

8 Contributors

Jose Herrera
Dpto. Arquitectura de Computadores y Automática
Facultad de Informatica
Universidad Complutense de Madrid
28040 Madrid, Spain

Eduardo Huedo
Dpto. Arquitectura de Computadores y Automática
Facultad de Informatica
Universidad Complutense de Madrid
28040 Madrid, Spain

Ruben S. Montero
Dpto. Arquitectura de Computadores y Automática
Facultad de Informatica
Universidad Complutense de Madrid
28040 Madrid, Spain

Ignacio M. Llorente
Dpto. Arquitectura de Computadores y Automática
Facultad de Informatica
Universidad Complutense de Madrid
28040 Madrid, Spain

9 Intellectual Property Statement

10 Full Copyright Notice