GWD-R, GWD-I or GWD-C

Łukasz Cieśnik, Piotr Domagalski,
Krzysztof Kurowski, Paweł Lichocki,
Poznan Supercomputing and Networking Center, Poland
FedStage Systems Inc.

Distributed Resource Management Application API
(DRMAA) Working Group

April 20th, 2007

PBS/Torque 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 PBS/Torque workload management system and Open DRMAA Service Provider (OpenDSP v1.0).
It does not define any standards or technical recommendations. Distribution is unlimited.

Abstract

This document describes experiences in the implementation of the Distributed Resource Management Application API (DRMAA) specification for the PBS/Torque workload management system and Open DRMAA Service Provider (OpenDSP v1.0). The document reports about issues that where identified during implementation and test of a DRMAA C library for PBS/Torque, which was evaluated successfully with the DRMAA working group compliance test for C bindings. We will also give suggestions for improvement of the specification, mainly concerning readability of the GFD-R-P.022 document.

Contents

Łukasz Cieśnik, Piotr Domagalski
Krzysztof Kurowski, Paweł Lichocki,
Poznan Supercomputing and Networking Center
FedStage Systems Inc.

Distributed Resource Management Application API
(DRMAA) Working Group

April 20th, 2007

1. Introduction

The Distributed Resource Management Application API specification (GFD-R.P.022) [1] specifies a generalized API to distributed resource management systems (DRMS) in order to facilitate integration of application programs. Soon after DRMAA reached "proposed recommendation" status, various DRM vendors and Grid community-oriented projects started implementing DRMAA language bindings. Today, there are implementations for DRM systems, such as Sun Grid Engine or Condor as well as different languages (C, Java, Perl, Python).

PBS/Torque DRMAA library is an implementation of mentioned Open Grid Forum DRMAA specification GFD-R.P.022 for PBS/Torque. The scope of the API covered all the high level functionality required for Grid applications to consign jobs to Torque, and it included common operations on jobs like termination or suspension. This library enables also the integration of OpenDSP with the underlying Torque system.

Open DRMAA Service Provider (OpenDSP) [3] is an open source implementation of SOAP Web Service which provides on-demand compute capacity for remote applications and services. It enables to build a secure virtual computing environment and take advantage of a large number of computing resources managed by a DRM. Local DRMs are integrated efficiently with OpenDSP via DRMAA 1.0 routines defined in GFD-R.P.022. OpenDSP v1.0 exposes to the Internet multi-user access and remote job execution and management capabilities supported by the following DRMs:

* SGE 6u7+,
* Condor 6.9+ ,
* PBS/Torque 2.1.3+.

OpenDSP is also using OGF JSDL-based XML schema with some extensions to describe the requirements of computational jobs for submission and control to underlying DRMs.

2. PBS/Torque experiences report

The DRMAA C compliance test is an application which tests for binary compatibility of a DRMAA C binding implementation with the DRMAA C binding specification. The test application includes over 40 tests which test various aspects of compliance of the binding implementation. The test application also includes an automated test, which runs all of the other tests in series and reports comprehensively on compliance. This automated test is the test used to verify the compliance of the N1 Grid Engine 6.0 update 9 DRM system DRMAA C binding implementation.

Below, we present a summary of PBS/Torque DRMAA v0.11 tests for Torque 2.1.8

DRMAA 1.0 C compliance test name	**Status**
ST_MULT_INIT	OK
ST_MULT_EXIT	OK
ST_SUPPORTED_ATTR	OK
ST_SUPPORTED_VATTR	OK
ST_VERSION	OK
ST_DRM_SYSTEM	OK

ST_DRMAA_IMPL	OK
ST_CONTACT	OK
ST_EMPTY_SESSION_WAIT	OK
ST_EMPTY_SESSION_SYNCHRONIZE_DISPOSE	OK
ST_EMPTY_SESSION_SYNCHRONIZE_NODISPOSE	OK
ST_SUBMIT_WAIT	OK
ST_BULK_SUBMIT_WAIT	OK
ST_BULK_SINGLESUBMIT_WAIT_INDIVIDUAL	OK
ST_SUBMITMIXTURE_SYNC_ALL_DISPOSE	OK
ST_SUBMITMIXTURE_SYNC_ALL_NODISPOSE	OK
ST_SUBMITMIXTURE_SYNC_ALLIDS_DISPOSE	OK
ST_SUBMITMIXTURE_SYNC_ALLIDS_NODISPOSE	OK
ST_EXIT_STATUS	FAILED
ST_SUBMIT_KILL_SIG	OK
ST_INPUT_FILE_FAILURE	OK
ST_OUTPUT_FILE_FAILURE	OK
ST_ERROR_FILE_FAILURE	OK
ST_SUBMIT_IN_HOLD_RELEASE	OK
ST_SUBMIT_IN_HOLD_DELETE	OK
ST_BULK_SUBMIT_IN_HOLD_SESSION_RELEASE	OK
ST_BULK_SUBMIT_IN_HOLD_SINGLE_RELEASE	OK
ST_BULK_SUBMIT_IN_HOLD_SESSION_DELETE	OK
ST_BULK_SUBMIT_IN_HOLD_SINGLE_DELETE	OK
ST_SUBMIT_POLLING_WAIT_TIMEOUT	OK
ST_SUBMIT_POLLING_WAIT_ZEROTIMEOUT	OK
ST_SUBMIT_POLLING_SYNCHRONIZE_TIMEOUT	OK
ST_SUBMIT_POLLING_SYNCHRONIZE_ZEROTIMEOUT	OK
ST_ATTRIBUTE_CHANGE	OK
ST_SUBMIT_SUSPEND_RESUME_WAIT	OK
ST_USAGE_CHECK	OK
MT_SUBMIT_WAIT	OK
MT_SUBMIT_BEFORE_INIT_WAIT	OK
MT_EXIT_DURING_SUBMIT	OK
MT_SUBMIT_MT_WAIT	OK
MT_EXIT_DURING_SUBMIT_OR_WAIT	FAILED

As it is presented above, only two DRMAA 1.0 C compliance tests have failed, namely ST_EXIT_STATUS, and MT_EXIT_DURING_SUBMIT_OR_WAIT. The ST_EXIT_STATUS test initializes the session and submits 255 single “exit” jobs, each with a different exit status. (An “exit job” is a job which can be configured to exit with a specific status code.) The test then waits individually for each job to end and checks that each job exited with the correct exit status before exiting the session. A typical UNIX process exit code ranges from 0..255, however in the current Torque implementation the set is limited to 0..127 which naturally causes the error during this test. In the second case, the specification does not state clearly what is the expected behavior of drmaa_submit(), drmaa_wait() and other routines when they are running in one thread and

drmaa_exit() is called in another parallelly. The test suite in MT_EXIT_DURING_SUBMIT, MT_SUBMIT_MT_WAIT, MT_EXIT_DURING_SUBMIT_OR_WAIT test cases only checks whether drmaa_exit() succeeds - the return codes of drmaa_wait() are not examined at all.

3. DRMAA 1.0 specification comments

During the extensive work on various DRMAA implementations, we have noticed several specification drawbacks. Bellow, we point them out with a few words of comment:

- In some cases it might be really hard or even impossible to implement some of the mandatory DRMAA attributes. For different DRMS this could vary. We therefore suggest adding a new error code to the specification: DRMAA_ERRNO_ATTRIBUTE_NOT_IMPLEMENTED.

  This value could be returned by drmaa_set_attribute() and drmaa_set_vector_attribute(). Furthermore, in case of future development of DRMAA, it might be desired to introduce optional routines and therefore DRMAA_ERRNO_NOT_IMPLEMENTED error code could come in handy. One use of such an error code could when using drmaa_control() one tries to request a job state change not feasible in specific DRMS.

- To ease handling of bulk jobs, we recommend adding the ability to use $drmaa_incr_ph$ not only in file paths but also in program arguments and environment variables.
- The DRMAA 1.0 specification does not clarify whether drmaa_wait() running in one thread, should react to job submission event in the other thread. This leads to ambiguity and makes this behavior implementation dependant.
- Using drmaa_wifaborted(), drmaa_wifexited(), drmaa_wifsignaled(), etc. is tedious and error-prone. For example, it's not clear what the right order of all the calls should be and what if none of them gives a meaningful answer, i.e. the job exit status could not be determined somehow. Some one-call semantics to get job exit status should be considered. Of course, this would kill all these formal POSIX semantics of wait and wif* functions.
- Removing drmaa_wif* routines and introducing one-call semantics could lead to more clear, consistent and informative job life cycle graph. For example, additionally to the existing job states returned by drmaa_job_ps(), we could imagine also states such as DRMAA_JOB_PS_COREDUMPED or DRMAA_JOB_PS_SIGNALLED.
- As for the current DRMAA 1.0 specification, we also find it inconvenient to have only state DRMAA_JOB_PS_FAILED that does not differentiate between job failure and user requested job termination.
- There is a confusion regarding *_len input parameters (e.g. error_diag_len for error messages): whether they should include 1 additional byte for '\0' character or not.
- C binding specification makes it even more ambiguous. It uses the term "buffer length" which might be not clear for a C programmer. Usually "buffer size" means sizeof(buf) and "string length" means that one needs to supply sizeof(buf) – 1 as the argument for function call. It would be good to see some explicit explanation.

4. Suggestions for future development

Working with DRMAA we have also came up with ideas for future additional functionalities:

- File transfer routines are desired by many users and currently they are only limited to standard input, output and error streams.
- We recommend offering a new routine for sending signals to submitted jobs. A lot of DRMs support various signaling mechanisms that can be used for applications steering. For instance, it could enable forcing application-level checkpointing, dynamic application image saving or other specific application behavior. Mentioned routines would help to create sort of batch-like application streaming relevant in some application use cases.
- Currently, users can specify resource requirements only using native specification or using job categories. This approach is DRMs dependant and not portable. It would be good to add some new optional attributes (e.g. number_cpu, min_memory) to deal with this problem.
- Due to accounting requirements it is desirable to define common names for resource usage attributes so they can be easily portable and interpreted by various applications.
- Basic, both static (e.g. a number of CPUs in DRM) and dynamic (e.g. a number of waiting and running jobs), monitoring information should be supported in order to provide a feedback for upper level middleware, brokering mechanisms, etc.
- We believe that a new routine for job status change monitoring, e.g. transition into the RUNNING state, would provide a usefully feedback for the end user.

5. 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.

6. Contributors

Łukasz Cieśnik,
Piotr Domagalski,
Krzysztof Kurowski
Paweł Lichocki,

Poznan Supercomputing and Networking Center
Noskowskiego 10 Street,
61-704 Poznań,
Poland

7. Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

8. Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

9. Full Copyright Notice

10. References

[1] Hrabri Rajic, Roger Brobst, Waiman Chan, Fritz Ferstl, Jeff Gardiner, Andreas Haas, Bill Nitzberg, and John Tollefsrud. Distributed Resource Management Application API Specification 1.0. http://forge.ggf.org/projects/drmaa-wg/, 2004

[2] Open DRMAA Service Provider v.1.0, http://sourceforge.net/projects/opendsp/, 2006

[3] PBS/Torque DRMAA library v 0.11, http://sourceforge.net/projects/pbs-drmaa/, 2006