FedStage DRMAA for LSF

Author:	ukasz Cienik ¡lukasz.ciesnik@fedstage.com¿, Mariusz Mamoski ¡mamonski@man.poznan.pl¿
Organization:	FedStage System, Poznan Supercomputing and Networking Center
Contact:	Mariusz Mamonski ¡mamonski@man.poznan.pl¿
Date:	2009-05-05
Version:	1.0.5
Revision:	2340

Abstract

This document describes installation, configuration and usage of FedStage DRMAA for LSF version 1.0.4.

Contents

Introduction

FedStage DRMAA for LSF is an implementation of Open Grid Forum DRMAA 1.0 (Distributed Resource Management Application API) specification for submission and control of jobs to Platform LSF. Using DRMAA, grid applications builders, portal developers and ISVs can use the same high-level API to link their software with different cluster/resource management systems.

This software also enables the integration of FedStage Computing with the underlying LSF system for remote multi-user job submission and control over Web Services.

Installation

To compile and install the library just go to main source directory and type:

```
$ ./configure [options] && make
$ sudo make install
```

The library requires LSF version 7.0 or further. To work with older versions it may require some patching.

Notable ./configure script options:

--with-lsf-inc LSF_INCLUDE_PATH Path to LSF header files (include dir). This and --with-lsf-lib options are unnecessary if LSF_ENVDIR environment variable is set correctly (e.g. by $LSF_TOP/conf/profile.lsf).

--with-lsf-lib LSF_LIBRARY_PATH Path to LSF libraries (lib dir).

--with-lsf-static Link DRMAA against static LSF libraries instead of shared ones.

> Note
>
> In LSF 7.0.3 the shared libraries are broken in the way they have some undefined symbols which should be defined. Using this option fixes this problem as static libraries are built correct.

--prefix INSTALLATION_DIRECTORY Root directory where FedStage DRMAA for LSF shall be installed. When not given library is installed alongside with LSF.

--enable-debug Compiles library with debugging enabled (with debugging symbols not stripped, without optimizations, and with many log messages enabled). Useful when you are to debug DRMAA enabled application or investigate problems with DRMAA library itself.

There are no unusual requirements for basic usage of library: ANSI C compiler and standard make program should suffice. If you have taken sources directly from SVN repository or wish to run test-suite you would need additional developer tools. For further information regarding GNU build system see the INSTALL file.

Configuration

During DRMAA session initialization (drmaa_init) library tries to read its configuration parameters from locations: /etc/lsf_drmaa.conf, ~/.lsf_drmaa.conf and from file given in LSF_DRMAA_CONF environment variable (if set to non-empty string). If multiple configuration sources are present then all configurations are merged with values from user-defined files taking precedence (in following order: $LSF_DRMAA_CONF, ~/.lsf_drmaa.conf, /etc/lsf_drmaa.conf).

Currently recognized configuration parameters are:

pool_delay Amount of time (in seconds) between successive checks of queue(s).

Type: integer, default: 5

cache_job_state According to DRMAA specification every drmaa_job_ps() call should query DRM system for job state. With this option one may optimize communication with DRM. If set to positive integer drmaa_job_ps() returns remembered job state without communicating with DRM for cache_job_state seconds since last update. By default library conforms to specification (no caching will be performed).

Type: integer, default: 0

wait_thread If set to 0 every call to drmaa_wait() or drmaa_synchronize() pools DRM for selected/all jobs. By default library creates additional thread which checks state of all job for duration of DRMAA session. drmaa_wait()/drmaa_synchronize() calls block until finished job is found.

Type: integer, default: 1

job_categories Dictionary of job categories. It's keys are job categories names mapped to native specification strings. Attributes set by job category can be overridden by corresponding DRMAA attributes or native specification. Special category name default is used when drmaa_job_category job attribute was not set.

Type: dictionary with string values, default: empty dictionary

lsb_events_file The location of the lsb.events file. If set the library polls the LSF events logfile instead of the LSF deamons.

Type: path, default: none

Configuration file syntax

Configuration file is in form a dictionary. Dictionary is set of zero or more key-value pairs. Key is a string while value could be a string, an integer or another dictionary.

configuration: dictionary — dictionary_body
dictionary: '{' dictionary_body '}'
dictionary_body: (string ':' value ',')*
value: integer — string — dictionary
string: unquoted-string — single-quoted-string — double-quoted-string
unquoted-string: [^ \t\n\r:,0-9][^ \t\n\r:,]*
single-quoted-string: '[^']*'
double-quoted-string: "[^"]*"
integer: [0-9]+

Native specification

DRMAA interface allows to pass DRM dependent job submission options. Those options may be specified by settings drmaa_native_specification or drmaa_job_category job attribute. drmaa_native_specification accepts space delimited bsub options while drmaa_job_category is name of job category defined in configuration file. -a and bsub options which are meant for interactive submission of jobs (-I, -Ip, -Is, -K) are not supported.

Attributes set in native specification overrides corresponding DRMAA job attributes which overrides those set by job category.

Table 1: Native specification strings with corresponding DRMAA attributes.

DRMAA attribute	native specification
drmaa_job_name	-J job name
drmaa_input_path	-i input_path
	-is input_path
drmaa_output_path	-o output path
	-oo output_path
drmaa_error_path	-e error path
	-eo error_path
drmaa_start_time	-b start time
drmaa_deadline_time	-t end_deadline
drmaa_js_state	-H
drmaa_transfer_files	-f file_stage_op
drmaa_v_email	-u mail_user
	-B, -N
	-m asked_hosts
	-x
	-n min_proc[,max_proc]
	-R res_req
drmaa_duration_hlimit	-c cpu_limit
drmaa_wct_hlimit	-W runtime_limit
drmaa_wct_slimit	-We estimated_runtime
	-M memory_limit
	-D data_limit
	-S stack_limit
	-v swap_limit
	-F file_limit
	-C core_limit
	-p process_limit
	-T thread_limit
	-ul
	-U reservation_id
	-ar reservation_id
	-wt warning_time
	-wa warning_action
	-s signal
	-q queue_name
	-w dependency
	-sp priority

... continued on next page

Table 1: Native specification strings with corresponding DRMAA attributes. (... continued)

DRMAA attribute	native specification
	-r, -rn
	-G user_group
	-g job_group_name
	-P project_name
	-Lp ls_project_name
	-E pre_exec_cmd
	-Ep post_exec_cmd
	-app app_profile
	-ext sched_options
	-jsdl jsdl_doc
	-jsdl_strict jsdl_doc
	-k checkpoint_dir
	-L login_shell
	-sla service_class_name
	-Z

Release notes

Changes in 1.0.4 release

- Fixed the core limit (-C) parsing in the native specification attribute.
- Fixed infinite loop on calling drmaa_wait/drmaa_synchronize routines after the CLEAN_PERIOD

Changes in 1.0.3 release

- Fixed segfault when drmaa_v_env was set. Now uses setenv and unsetenv calls to modify environ instead of substituting environ pointer.
- drmaa_transfer_files works (in progress).
- By default when --prefix is not given at configure time library is installed alongside with LSF.
- When waiting for any job or with waiting thread enabled status of all jobs is pooled from DRM in one LSF API call.
- New configuration option: cache_job_state.
- More detailed error messages.
- It now compiles against LSF version 6.0 or futher although it was not tested at runtime.

Changes in 1.0.2 release

- drmaa_remote_command and drmaa_v_argv are quoted and not interpreted by shell (e.g. spaces are allowed in command and arguments). Jobs are created with exec command i.e. unnecessary shell process dangling for duration of job was eliminated.
- drmaa_wifexited follow refinement on DRMAA Working Group mailing list - returns 1 only for exit statuses not greater than 128. Previously it returned 1 for all jobs which were run (not aborted).
- It has been reported that in some situations job which was recently submitted is not always immediately visible through LSF API. There is now workaround for such behaviour.
- drmaa_transfer_files is ignored because of segfaults produced by it.
- Bugfixes: Segfault when drmaa_v_argv is not set. Native specification parsing bugs. Various other segfaults and memory leaks.

Changes in 1.0.1 release

> Note
>
> Version 1.0.1 of library was previously released with 2.0 version number. Afterwards we decided this is misleading and does not follow versioning scheme established by DRMAA Working Group (i.e. it does not reflect the version of DRMAA specification implemented by the library).

- Many attributes implemented:
  - drmaa_start_time,
  - drmaa_native_specification,
  - drmaa_transfer_files,
  - job limits.
- Integrates with FedStage Advance Reservation Library for LSF.
- Job category now points to native specification string in configuration file instead of job group.
- Thread safe design.
- Configuration file(s).
- Lots of bug fixes.
- More robust code.
- Meaningful logging, error messages and codes.

Known bugs and limitations

Library covers all DRMAA 1.0 specification with exceptions listed below. It was successfully tested with Platform LSF 7.0.3 on Linux OS and passes the official DRMAA test-suite. All mandatory and nearly all optional job attributes (except job run duration soft limit) are implemented.

Known limitations:

- $drmaa_incr_ph$ is replaced only within input, output and error file paths while according to specification it should be also substituted in job working directory.
- Host name is ignored in input, output and error path. They are always copied from and to submission host.
- Input file is copied from submission host when it is not present on execution host even when i was not in transfer files attribute.
- drmaa_wcoredump() always returns false.

Developers

Core functionality of DRMAA is put into drmaa_utils library. This library was created in order to keep consistent common functionality of FedStage DRMAA for PBS Pro and FedStage DRMAA for LSF library. As it is independent from any particular DRM you may found this library useful for developing other DRMAAs. For detailed information please take a look at source code documentation.

Developer tools

Although not needed for library user the following tools may be required if you intend to develop FedStage DRMAA for LSF:

- GNU autotools (autoconf, automake, libtool),
- Bison parser generator,
- gperf perfect hash function generator,
- Ragel finite state machine compiler,
- Docutils for processing this README,
- LaTeX for creating documentation in PDF format,
- Doxygen for generating source code documentation.

Contact

Please send your comments or questions to the following mailing list:

https://www.fedstage.com/lists/listinfo/drmaa-lsf-users (drmaa-lsf-users@lists.fedstage.com)

Please also visit the project webpage to find news and new releases of our software:

http://www.fedstage.com/wiki/FedStage˙DRMAA˙for˙LSF

License