

Poznan Supercomputing
and
Networking Center

Summer Projects

Michał Matłoka

michal.matloka@student.put.poznan.pl

Date: September 23, 2010, Poznań

Contents

1	Introduction	1
2	Automated code review of SMOA Services	1
2.1	Flawfinder	1
2.2	Rough Auditing Tool for Security - RATS	2
2.3	Splint	2
3	Integration with PL-grid accounting infrastructure - bat_updater module	3
4	PBS DRMAA - implementing "triggered" mode	4
5	Improved DRMAA Scalability program	4
6	DRMAA implementation for SLURM	5
6.1	Introduction	5
6.2	Input Documents	5
6.3	Mapping of DRMAA interface on SLURM API	5
6.3.1	DRMAA job attributes mapping to SLURM	5
6.3.2	DRMAA states mapping	6
6.3.3	drmaa_control operations mapping	7
6.4	Native specification	7
6.5	DRMAA library implementation using DRMAA Utils	8
6.6	End User Manual	8
6.7	DRMAA testsuite	8

1 Introduction

During my summer work in Poznan Supercomputing Networking Center I have worked on a few projects where the most important are listed below

1. Automated code review of SMOA Services
2. Integration with PL-grid accounting infrastructure - bat_updater module
3. PBS DRMAA - implementing "triggered" mode
4. DRMAA implementation for SLURM

Total time spent on projects was one month.

2 Automated code review of SMOA Services

Automated code review software checks source code in order to find violations of program standards and displays them as the list of warnings. My task was to use these tools to check the SMOA Core library.

2.1 Flawfinder

Output fragment:

```
Hits = 6
Lines analyzed = 308 in 0.52 seconds (18364 lines/second)
Physical Source Lines of Code (SLOC) = 225
Hits@level = [0] 0 [1] 3 [2] 2 [3] 0 [4] 1 [5] 0
Hits@level+ = [0+] 6 [1+] 6 [2+] 3 [3+] 1 [4+] 1 [5+] 0
Hits/KSLOC@level+ = [0+] 26.6667 [1+] 26.6667 [2+] 13.3333 [3+] 4.44444 [4+] 4.44444 [5+] 0
Minimum risk level = 1
```

Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
Flawfinder version 1.27, (C) 2001-2004 David A. Wheeler.
Number of dangerous functions in C/C++ ruleset: 160
Examining ipc.c

```
ipc.c:52: [2] (buffer) char:
    Statically-sized arrays can be overflowed. Perform bounds checking,
    use functions that limit length, or ensure that the size is larger than
    the maximum possible length.
ipc.c:123: [2] (buffer) char:
    Statically-sized arrays can be overflowed. Perform bounds checking,
    use functions that limit length, or ensure that the size is larger than
    the maximum possible length.
ipc.c:128: [2] (buffer) char:
    Statically-sized arrays can be overflowed. Perform bounds checking,
    use functions that limit length, or ensure that the size is larger than
    the maximum possible length.
ipc.c:227: [1] (buffer) read:
    Check buffer boundaries if used in a loop.
```

Program output had 1858 lines. Flawfinder displays mainly tips for programmers what they should be aware of while using certain functions. Eventually all verified by me warnings were were not dangerous.

2.2 Rough Auditing Tool for Security - RATS

Output fragment:

```
Total lines analyzed: 309
Total time 0.000270 seconds
1144444 lines per second
Entries in perl database: 33
Entries in ruby database: 46
Entries in python database: 62
Entries in c database: 334
Entries in php database: 55
Analyzing ipc.c
ipc.c:52: High: fixed size local buffer
ipc.c:123: High: fixed size local buffer
ipc.c:128: High: fixed size local buffer
Extra care should be taken to ensure that character arrays that are allocated
on the stack are used safely. They are prime targets for buffer overflow attacks.
```

Program output had 593-791 lines depends on check level. Like in Flawfinder user gets list of tips and in all checked cases they were not dangerous in SMOA Core.

2.3 Splint

Output fragment:

```
ipc.c: (in function sm_ipc_send_fd)
ipc.c:85:5: Arrow access of non-pointer (struct iovec [1]): iov->iov_base
    Types are incompatible. (Use -type to inhibit warning)
ipc.c:86:5: Arrow access of non-pointer (struct iovec [1]): iov->iov_len
ipc.c: (in function sm_ipc_recv_fd)
ipc.c:133:126: Function assert expects arg 1 to be boolean gets int *:
    fd_received
ipc.c:157:5: Arrow access of non-pointer (struct iovec [1]): iov->iov_base
ipc.c:158:5: Arrow access of non-pointer (struct iovec [1]): iov->iov_len
ipc.c: (in function sm_simple_read)
```

```

ipc.c:220:110: Function assert expects arg 1 to be boolean gets void *: buf
ipc.c: (in function sm_simple_write)
ipc.c:256:110: Function assert expects arg 1 to be boolean gets void *: buf
ipc.c:39:19: File static variable rcsid declared but not used
    A variable is declared but never used.

```

Program output had 2909 lines. Splint is a more "strict" software. It does not display tips but warnings concerning given source code. Program output had 2909 lines. I have verified 10 most frequently occurring warnings and all of them was not dangerous in SMOA Core. Example:

```

uuid.c:111:3: Format argument 4 to snprintf (%02x) expects unsigned int gets
    int: (int)uuid->time_low[3]

```

```

sprintf set to get integer but got size_t.

```

```

odbc.c:537:2: Assignment of unsigned short int to int:  posix_time->tm_mon = timestamp->month - 1

```

3 Integration with PL-grid accounting infrastructure - bat_updater module

bat_updater is a SMOA Computing module which allows publication of resource usage data via DRMAA. Transport layer is based on JMS (Java Messaging Service) and uses opensource implementation of ActiveMQ called Apache ActiveMQ CPP (<http://activemq.apache.org/cms/index.html>). Obtained rusage data are converted to XML format. Example generated XML via DRMAA interface:

```

<?xml version="1.0" encoding="UTF-8"?> <site name="psnc-smoa-plgrid">
  <job>
    <batch_server>mich-laptop</batch_server>
    <job_id>10732.mich-laptop</job_id>
    <user>mich</user>
    <group>mich</group>
    <queue>batch</queue>
    <ctime>1285112777</ctime>
    <qtime>1285112777</qtime>
    <etime>1285112777</etime>
    <start>16518</start>
    <end>1285112868</end>
    <exec_host>
    <node>
      <nodename>mich-laptop</nodename>
      <cpu>0</cpu>
    </node>
    </exec_host>
    <cputime>0</cputime>
    <walltime>91</walltime>
    <mem>2547712</mem>
    <vmem>29581312</vmem>
    <estatus>0</estatus>
    <infrastructure>smoa</infrastructure>
    <grid_job_id>"368e7fb7-666f-467c-ab84-820804a0372d"</grid_job_id>
    <userDN>"(anonymous)"</userDN>
  </job>
</site>

```

In this module a C++ ActiveMQ implementation is used because C version is not fully functional. A C wrapper interface composed of three functions (activemqcpp_connect, activemqcpp_send_message, activemqcpp_disconnect) was written in order to allow communication with ActiveMQ from bat_updater module.

4 PBS DRMAA - implementing "triggered" mode

Triggered mode allows DRMAA to acquire job state-change events. This mode may be the only solution for many production clusters (e.g. reef.man.poznan.pl).

wait_thread	pbs_home	mode	keep completed needed	comments
0	not set	polling	yes	default configuration
1	not set	polling	yes	more effective than above
1	set	triggered	no	read access to server logs needed

Configuration - polling and triggered modes

At the start of the DRMAA program the current log file's size is stored. Later, in the "wait thread" log file parsing starts from the previously remembered position. Log file name is determined by pbs_home configuration variable (Path to Torque/PBS Pro spool directory that contains server logs e.g.: /var/spool/pbs). This way only new lines related to current execution are being parsed. On day change old file is being parsed once more and then new path to log file is generated.

Every Log file line is composed of 6 fields: FLD_DATE;FLD_EVENT;FLD_OBJ;FLD_TYPE;FLD_ID;FLD_MSG

FLD_EVENT	FLD_MSG	comments
0008	Job Run at request of	On Job run we launch job_ps in order to get execution host etc
0008	Job Modified at request of Scheduler@mich-laptop	From FLD_DATE we get modify time
0008	Job Queued at request of , owner = , job name = , queue =	
0008	Job deleted at request of	
0010	Exit_status= resources_used.cput= resources_used.mem= resources_used.vmem= resources_used.walltime=	Job completed, status and resource usage information parse

Every line gives information about job state.

Collected data in triggered mode is passed to pbsdrmaa_job_update_status routine.

PBS DRMAA in "triggered mode" passed official OGF DRMAA testsuite on Torque and PBS Pro systems.

5 Improved DRMAA Scalability program

A dedicated test program was written for scalability testing purposes. It simulates SMOA Computing service under continuous workload.

Addition thread killing jobs was added. This program takes 5 arguments: SUB_INT, SLEEP_TIME, POLL_INTERVAL, WAIT_INTERVAL, KILL_RANGE and starts 4 threads:

- first thread submits sleep job with SLEEP_TIME parameter
- second thread every POLL_INTERVAL checks status of every job
- third thread with WAIT_INTERVAL (timeout) and SESSION_ANY (job) parameters runs drmaa_wait and updates finished jobs list.
- fourth thread kills a random job every k seconds where ks is chosen randomly from interval (0,KILL_RANGE)

Jobs are submitted until user press ctrl+c. Then first, second and fourth thread stops and program waits for every submitted job end.

6 DRMAA implementation for SLURM

6.1 Introduction

PSNC DRMAA for Simple Linux Utility for Resource Management (SLURM) is an implementation of Open Grid Forum DRMAA 1.0 (Distributed Resource Management Application API) specification for submission and control of jobs to Simple Linux Utility for Resource Management (SLURM). 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 SMOA Computing with the underlying LoadLeveler system for remote multi-user job submission and control over Web Services.

6.2 Input Documents

- DRMAA 1.0 Grid Recommendation '
<http://www.ogf.org/documents/GFD.133.pdf>
- DRMAA C Binding v1.0
https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.drmaa-wg/docman.root.ggf_13/doc5545
- Simple Linux Utility for Resource Management (SLURM) Documentation
<https://computing.llnl.gov/linux/slurm/documentation.html>

6.3 Mapping of DRMAA interface on SLURM API

6.3.1 DRMAA job attributes mapping to SLURM

SLURM job is described by `job_desc_msg_t` structure.

DRMAA	SLURM	Comment
drmaa_block_email	do not set mail_user	
drmaa_deadline_time	OPTIONAL ATTRIBUTE	NOT IMPLEMENTED
drmaa_duration_hlimit	OPTIONAL ATTRIBUTE	NOT IMPLEMENTED
drmaa_duration_slimit	OPTIONAL ATTRIBUTE	NOT IMPLEMENTED
drmaa_error_path	char * std_err	
drmaa_input_path	char * std_in	
drmaa_job_category	library configuration .slurm_drmaa.conf, use native specification function	Syntax like in native specification attribute "--account My_job -N 1=2"
drmaa_job_name	char * name	
drmaa_join_files	same values of std_err and std_out	
drmaa_js_state	uint32_t priority	0 = held
drmaa_native_specification	arguments like in sbatch	Syntax example "--account My_job -N 1=2"
drmaa_output_path	char * std_out	
drmaa_remote_command	char * script	contains generated bash script
drmaa_start_time	time_t begin_time	
drmaa_transfer_files	OPTIONAL ATTRIBUTE	NOT IMPLEMENTED, practically not used because every cluster has shared file system
drmaa_v_argv	Add to bash script	
drmaa_v_email	char * mail_user	
drmaa_v_env	char ** environment, uint32_t env_size	name=value pairs, one per line
drmaa_wct_hlimit	uint32_t time_limit	
drmaa_wct_slimit	NOT IMPLEMENTED	
drmaa_wd	char * work_dir	

6.3.2 DRMAA states mapping

The DRMAA states list was compared with SLURM states that can be retrieved using API.

SLURM	DRMAA
JOB_PENDING	in next table
JOB_RUNNING	DRMAA_PS_RUNNING
JOB_SUSPENDED	DRMAA_PS_USER_SUSPENDED
JOB_COMPLETE	DRMAA_PS_DONE
JOB_CANCELLED	DRMAA_PS_FAILED
JOB_FAILED	DRMAA_PS_FAILED
JOB_TIMEOUT	DRMAA_PS_FAILED
JOB_NODE_FAIL	DRMAA_PS_FAILED

JOB_PENDING is additionally described by state_reason variable:

SLURM	DRMAA
WAIT_NO_REASON	DRMAA_PS_QUEUED_ACTIVE
WAIT_PRIORITY	DRMAA_PS_QUEUED_ACTIVE
WAIT_DEPENDENCY	DRMAA_PS_QUEUED_ACTIVE
WAIT_RESOURCES	DRMAA_PS_QUEUED_ACTIVE
WAIT_PART_NODE_LIMIT	DRMAA_PS_QUEUED_ACTIVE
WAIT_PART_TIME_LIMIT	DRMAA_PS_QUEUED_ACTIVE
WAIT_PART_STATE	DRMAA_PS_QUEUED_ACTIVE
WAIT_HELD	DRMAA_PS_USER_ON_HOLD
WAIT_TIME	DRMAA_PS_QUEUED_ACTIVE
WAIT_LICENSES	DRMAA_PS_QUEUED_ACTIVE
WAIT_ASSOC_JOB_LIMIT	DRMAA_PS_QUEUED_ACTIVE
WAIT_ASSOC_RESOURCE_LIMIT	DRMAA_PS_QUEUED_ACTIVE
WAIT_ASSOC_TIME_LIMIT	DRMAA_PS_QUEUED_ACTIVE
WAIT_RESERVATION	DRMAA_PS_QUEUED_ACTIVE
WAIT_NODE_NOT_AVAIL	DRMAA_PS_QUEUED_ACTIVE
WAIT_TBD1	DRMAA_PS_QUEUED_ACTIVE
WAIT_TBD2	DRMAA_PS_QUEUED_ACTIVE
FAIL_DOWN_PARTITION	DRMAA_PS_FAILED
FAIL_DOWN_NODE	DRMAA_PS_FAILED
FAIL_BAD_CONSTRAINTS	DRMAA_PS_FAILED
FAIL_SYSTEM	DRMAA_PS_FAILED
FAIL_LAUNCH	DRMAA_PS_FAILED
FAIL_EXIT_CODE	DRMAA_PS_FAILED
FAIL_TIMEOUT	DRMAA_PS_FAILED
FAIL_INACTIVE_LIMIT	DRMAA_PS_FAILED
FAIL_BANK_ACCOUNT	DRMAA_PS_FAILED

For value greater than 0x4000 we do not update DRMAA state because describes a transient state.

6.3.3 drmaa_control operations mapping

DRMAA	LoadLeveler	Comments
DRMAA_CONTROL_SUSPEND	slurm_suspend	Administrators only
DRMAA_CONTROL_RESUME	slurm_resume	Administrators only
DRMAA_CONTROL_HOLD	priority = 0	
DRMAA_CONTROL_RELEASE	priority != 0	Administrators only, on hold save job priority and on release bring old value. When job is submitted in held state after resume it gets MAX UINT32_T value
DRMAA_CONTROL_TERMINATE	slurm_kill_job(job_id,SIGKILL,0)	

6.4 Native specification

DRMAA interface allows to pass DRM dependent job submission options. Those options may be specified directly by setting drmaa_native_specification job template attribute or indirectly by the drmaa_job_category job template attribute. In SLURM DRMAA the following set of sbatch arguments were implemented:

Native specification	Description
-A, -account=name	Charge job to specified accounts
-acctg-freq	Define the job accounting sampling interval
-comment	An arbitrary comment
-C, -constraint=list	Specify a list of constraints
-contiguous	If set, then the allocated nodes must form a contiguous set
-exclusive	Allocate nodenumber of tasks to invoke on each nodes in exclusive mode when cpu consumable resource is enabled
-mem=MB	Minimum amount of real memory
-mem-per-cpu=MB	Maximum amount of real memory per allocated cpu required by a job
-mincpus=n	Minimum number of logical processors (threads) per node
-N, -nodes=N	Number of nodes on which to run (N = min[-max])
-ntasks-per-node=n	Number of tasks to invoke on each node
-p, -partition=partition	Partition requested
-qos=qos	Quality of Service
-requeue	If set, permit the job to be requeued
-reservation=name	Allocate resources from named reservation
-s, -share	Job allocation can share nodes with other running jobs
-w, -odelist=hosts	Request a specific list of hosts

Additional description of each parameter can be found in 'man sbatch'.

6.5 DRMAA library implementation using DRMAA Utils

Library was implemented in C based on the implementation of DRMAA for LoadLeveler (<http://gforge.man.poznan.pl/svn/smoaincubator/drmaa/11/>), drmaa_utils (http://gforge.man.poznan.pl/gf/project/smoaincubator/scmsvn/?action=browse&path=/drmaa/drmaa_utils/) and tools like GNU Compiler Collection and GNU Project Debugger accordingly to created mappings.

6.6 End User Manual

Documentation can be found in doc folder in the library package.

6.7 DRMAA testsuite

Library covers all DRMAA 1.0 specification with exceptions listed below. It was successfully tested with Simple Linux Utility for Resource Management (SLURM) 2.1.13 on Linux and passes 44/44 tests of the official DRMAA test-suite and is therefore DRMAA1.0-compliant for test suite version 1.7.2.

Using non-administrator account SLURM DRMAA passes 40/44 tests of the official DRMAA test-suite.

Known limitations:

- fuction `drmaa_control` with `DRMAA_CONTROL_HOLD`, `DRMAA_CONTROL_RELEASE`, `DRMAA_CONTROL_SUSPEND`, `DRMAA_CONTROL_RESUME` is administrator only
- `drmaa_wct_slimit` not implemented
- optional attributes `drmaa_deadline_time`, `drmaa_duration_hlimit`, `drmaa_duration_slimit`, `drmaa_transfer_files` not implemented