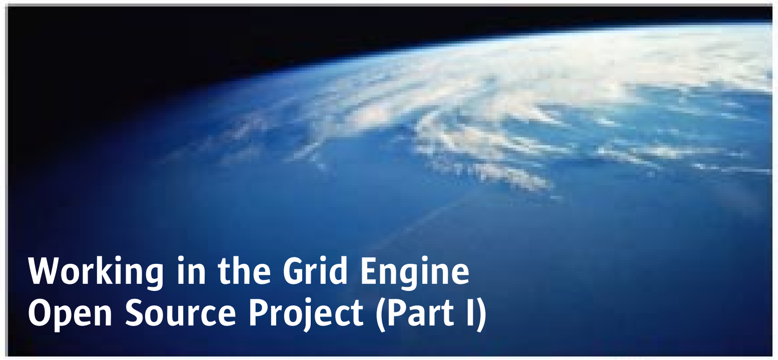


Working in the Grid Engine Open Source Project (Part I)

Andre Alefeld

Software Engineering

We are talking about...

- 373 source files in C
- 448 header files
- 443 source and header files of 3rd party code
- 5 daemons and 19 client cmds.
- A software that runs on most Unix systems

Agenda

- SourceCast a collaborative environment
- Code Organization and coding style
- Building Grid Engine from scratch

SourceCast

SourceCast: wide area collaborative software development environment

- allows distributed development
- ease of collaboration, webbased interface
- set of common tools
- central information and feedback exchange
- central administration
- extensibility

SourceCast
Web-based Project Environment

- set up and advance projects easily
- centralize project information, avoiding synchronization problems
- location independent access
- changes immediately visible
- focus on SW development instead of system maintenance

SourceCast
Revision Control

- CVS ensures accurate revision information
- exchange among any # of developers
- color-coded version tracking/diffing
- code/documentation/webcontent use same procedures
- revisions perfected locally and published in master repository

SourceCast
Issue Tracking

- monitor issue progress, better QA
- automatic assignment of tasks to the right people
- organize information around an issue
- metrics/report generation

SourceCast
Mailing Lists

- mail archives help new developers
- open forum for discussion and voting
- enhance collaboration
- mailing list search can give hints/contacts on a special topic

SourceCast

Unified Web-based Administration

- Access control via role based permissions
- easily maintainable and customizable development environment
- administration from anywhere
- Hiding of underlying details for project admins

SourceCast

Grid Engine Registration

Two step registration process

- Register as user of sunsource.net
  http://gridengine.sunsource.net/servlets/Join
- Register as a project member of Grid Engine
  http://gridengine.sunsource.net/servlets/ProjectMembershipRequest

SourceCast

Grid Engine Roles

- Observer
  Read-only access to project code; can also submit issues
- Content Developer
  Read/Write access to project web content
- Developer
  Read/Write access to all of project
- DeveloperPlus
  Developer with Upload permissions
- ProjectOwner
  DeveloperPlus with project specific admin priviledges

SourceCast

Grid Engine Mailing Lists

http://gridengine.sunsource.net/project/gridengine/maillist.html

- User Mailing List
  - announce@gridengine.sunsource.net
  - users@gridengine.sunsource.net
- Developer Mailing Lists
  - dev@gridengine.sunsource.net
  - issues@gridengine.sunsource.net
  - cvs@gridengine.sunsource.net

SourceCast
Grid Engine Issuezilla

Issuezilla: Bug Reports, Enhancements, Patch Submission, Task Mgmt

- currently 7 GE issue categories
  - cleanup: source code cleanup tasks
  - doc: man pages and documentation
  - gui: qmon related issues
  - install: installation related issues
  - kernel: CLI and daemon specific issues
  - testsuite: testsuite related issues
  - www: Grid Engine web site issues

SourceCast
Grid Engine Upload/Download

- Download Binaries
  Prebuilt courtesy binaries for the most important Unix architectures
  http://gridengine.sunsource.net/project/gridengine/download.html
- File Exchange Tool
  - source tarballs
  - contributions
  - source code patches either here or in Issuezilla

SourceCast
Grid Engine cvs / cvsweb

- cvsweb allows browsing, diffing and downloading of single files
- Getting the source with cvs
  - must be at least Observer
  - cvs version 1.11 client/server needed
  - ssh tunnel when behind firewall
  - ssh must support tunneling (-L option)

SourceCast
Grid Engine cvs (cont'd)

- Two choices with/without ssh tunnel

without ssh tunnel:
```
% setenv CVSROOT :pserver:<username>@cvs.gridengine.sunsource.net:/cvs
```
with ssh tunnel on the machine where the ssh tunnel is set up:
```
% setenv CVSROOT :pserver:<username>@localhost:/cvs
```

- ssh tunnel setup
```
% ssh -f -x -L 2401:localhost:2401 \
tunnel@cvs.gridengine.sunsource.net echo hallo
```
(enter password "tunnel" when prompted)

SourceCast
Grid Engine cvs (cont'd)

- Login at CVS repository
  % cvs login
  (enter your Grid Engine website password when prompted)
- Source checkout
  % cvs -z 9 co gridengine
  (maintrunk checkout under ./gridengine)
  % cvs -z 9 co -r V53_BRANCH gridengine -d V53_BRANCH
  (checkout of V53_BRANCH under ./V53_BRANCH instead of ./gridengine)
  % cvs -z 9 co -r V53_TAG gridengine
  (checkout of a tagged snapshot)

SourceCast
Grid Engine cvs (cont'd)

- Branching
  - release versions as branch of maintrunc
  - fixes applied to both branches
  - for special implementations subbranches
  - for new release branches are obsoleted and die
  - main development in maintrunc
  - avoid branching if possible, development overhead

SourceCast
Grid Engine Patches

- Patch integration procedure
  - cvs diff for patch creation
  - discuss patch on dev mailing list
  - submit patch either via Issuezilla or FileExchange Tool
  - get patch review
  - Developer integrates patch
  - Become Developer
- Code Stability - Nightly Build

SourceCast - Important Links

- Open Source Site
  http://gridengine.sunsource.net
- Sun Grid Engine
  http://www.sun.com/gridware
- Registration/Membership
  http://gridengine.sunsource.net/servlets/Join
  http://gridengine.sunsource.net/servlets/ProjectMembershipRequest
  (approval by project manager required)
- Mailing Lists
  http://gridengine.sunsource.net/project/gridengine/maillist.html
- Issuezilla
  http://gridengine.sunsource.net/issues/query.cgi
- Courtesy Binaries
  http://gridengine.sunsource.net/project/gridengine/download.html
- Documentation, Howtos
  http://gridengine.sunsource.net/project/gridengine/documentation.html
  http://gridengine.sunsource.net/project/gridengine/howto/howto.html
- CVS
  http://www.cvshome.org and http://www.wincvs.org

SGE code overview

- Overview of CVS repository
  - the top level directories
  - the project site 'infrastructure'
  - the 3rd party modules
  - THE source

SGE code overview (cont'd)

- Overview of CVS repository

Changelog

- logging of all (significant) changes
- clear definition of source tags (and releated versions)
- most accurate source of information for a developer, supporter and user about (fixed) issues and bugs

doc/

- files which go in distribution
- sometimes linked from project site

doc/man

- manual pages in nroff format

SGE code overview (cont'd)

doc/devel

- Developer documentation
- still quite incomplete

source/

- everything which is needed during a build

testsuite/

- all files and tests for Tcl/Tk/Expect based testsuite
- used by engineering for QA, nightly builds
- also necessary for compatibility tests (if partner creates his own (commercial) distribution

www/

- files of project site

SGE code overview (cont'd)

A closer look at the source/ directory

3rdparty/	- 3rd party source code
aimk*	- architecture independent make
aimk.site	- global settings for additional libraries und programs
clients/	- all non-SGE daemons
common/	- code shared by daemons and clients
daemons/	- all SGE daemons
dist/	- files which go in a SGE distribution
experimental/	- unsupported stuff and prototypes
libs/	- libraries used but SGE daemons and clients
scripts/	- utility scripts
security/	- security framework, including OpenSSL
utilbin/	- helper programs which go in $SGE_ROOT/utilbin

SGE code overview (cont'd)

A glance at the source/3rdparty directory

adoc	- Adoc a tool for generating documentation
fnmatch	- needed for the NEC port
openssl	- only LICENCE, not part of the GE project
qmake	- preconfigured qmake 3.78.1
qmon	- 3rd party widgets used by qmon
qtcsh	- a modified tcsh
remote	- modified rlogin/rsh to run with SGE
sge_depend	- tool for creating make dependencies
strptime	- needed for NEC port, MacOS X port
zlib	- zlib - currently not part of standard build

SGE code overview (cont'd)

source/common directory

- mostly files which did not make their way in a library
- `basis_types.h`
  - defines architecure specific format strings
  - contains other system wide defines
- `read_object.c, read_write_*.c`
  - `read_object()` generic function to read spool files from disc. Not used for all spool files
  - uses function pointers which point to actual functions which do the work (`read_write_*.c` in common/ directory)
  - writing of objects is done directly: `write_ckpt()` ....

SGE code overview (cont'd)

libs/ - libraries used by most commands

- comm/
  - The low level communication library
  - communicates works with sge_commd only
  - single threaded
  - most functionality isolated in `comm/commlib.c`

SGE code overview (cont'd)

libs/ (cont'd)

cull/

- Common Usuable List Library
- provides reusable list functions
- provides "database" like abstraction for creating, accessing etc. list objects:
  - `lCreateList(), LFreeList()`
  - `lCreateElem(), LFreeElem()`
  - `lWhat(), lWhere()`
  - `lCopyList()`

SGE code overview (cont'd)

libs/cull (cont'd)

- packing routines needed to
  - spool CULL lists to/from disk (job spooling)
  - send lists to communication partners (by using communication library)
  - zlib compression can be enabled through compilation and at runtime
  - packing code in
    `pack.c`

SGE code overview (cont'd)

libs/ (cont'd)

- gdi/ - Grid Engine Database Interface
  - standardizes the mechanism for creating, retrieving, changing and deleting objects stored within the qmaster process.
  - GDI is the main interface for communication between a client and qmaster
  - Implementation is two-fold:
    - qmaster: implements the server functionality
    - client side: implement calls for getting/adding/deleting/modifying objects in qmaster

SGE code overview (cont'd)

libs/gdi/ - Grid Engine Database Interface (cont'd)

- Uses CULL for internal representation of stored objects
- Also implements functions for sending data between and to daemons:
  - requests - the most general form for data excahnge
  - orders - what the scheduler sends to qmaster
  - events - what qmaster sends to event client
  - (load reports, job reports) - what the execd sends to q.
  - helper functions for tight PE integration

SGE code overview (cont'd)

libs/gdi/ - Grid Engine Database Interface (cont'd)

- all CULL lists (all persistent and temporary objects) are defined in the gdi/ directory:
  `sge_<objname>L.h`
- All SGE CULL lists are "glued" together by
  `sge_boundaries.h`

SGE code overview (cont'd)

Libs/ (cont'd)

- rmon/
  - Designed as a remote monitoring library
  - Used for implementing debugging macros which print to stdout/err if commands which use rmon functions are started with environment variable `$SGE_DEBUG_LEVEL` set.
  - Most popular macros/functions are
    - `DENTER/DEXIT` - when a function is entered/left
    - `DTRACE` - print name and line of current file
    - `DPRINTF - printf()`
  - See `rmon/sgermon.h` for macro definition

SGE code overview (cont'd)

libs/ (cont'd)

- sched/
  - Implements most of the scheduling code
  - needed also by qmaster, qstat, qhost...
    - e.g. qstat is a fat client. To display the information provided by qstat the "raw" data it receives, it needs to process it with the help of the scheduling routines
  - SGEEE code in `sgeee.c`

SGE code overview (cont'd)

libs/ (cont'd)

- uti/
  - low level library, independent from rest of source code
  - "self contained", no global variables....
  - Implements utility and helper functions used by naerly all SGE commands
  - Does not use libcull, libgdi, libsched
  - only uses librmon for debugging macros

SGE code overview (cont'd)

The daemons/ directory

– commd/	- multiplexer for communication
– common/	- code shared by several daemons
– execd/	- report load, start/reap jobs
– qmaster/	- the "database" server
– schedd/	- event client which gives orders
– shadowd/	- a kind of "watchdog"
– shepherd/	- a self contained job starter

daemons/common

- shared code needed by more than one daemon
  - reading/writing jobs
  - setup environment for child process
  - sending mail

daemons/commd

- commd is a message multiplexer
- uses a store-and-forward principle
- communication can be routed over two commd's
- commd does not open sockets actively to its clients (except: routing messages to another commd)
- commd clients are called "commprocs"
- client needs to use "commlib" for communication with commd

daemons/qmaster

- Important files:
  - `sge_c_gdi.c, sge_c_ack.c`
    - server part of GDI
  - `sge_job.c`
    - accept new job, delete job, qalter job, handle job dependencies
  - `sge_give_jobs.c`
    - send job to execd
    - process job reports and job end

daemons/qmaster

- `sge_follow.c`
  - follow orders from scheduler
- `sge_m_event.c`
  - send events to event client (scheduler)

SGE code overview (cont'd)

daemons/schedd

- uses schedlib
- `sge_category.c`
  - Categorize jobs for efficiency reasons
- `sge_access_tree.c`
  - user_sort=true for pending jobs
- `sge_c_event.c`
  - generic event client interface
- `sge_process_events.c`
  - trigger scheduling, get events, send orders

SGE code overview (cont'd)

daemons/execd

- `dispatcher.c`
  - all work is done from here
- `exec_job.c`
  - setup files for shepherd, start shepherd
- `execd_ck_to_do.c`
  - monitor (->limits), reprioritize jobs
  - signal jobs (via shepherd)
- `execd_job_exec.c`
  - handle jobs received from qmaster

SGE code overview (cont'd)

daemons/execd (cont'd)

- `execd_signal_queue.c`
  - send signals to job (via shepherd)
- `execd_ticket.c`
  - process new tickets for job and call PTF
- `job_report_execd.c`
  - create job reports and put in send queue
- `load_avg.c`
  - retrieve load values and create load reports
- `ptf.c`
  - SGEEE: Priority Translation Facility

SGE code overview (cont'd)

daemons/execd (cont'd)

- `reaper_execd.c`
  - handle exited job (shepherd)
- `report.c`
  - send job/load reports
- `sge_load_sensor.c`
  - interface for starting external load sensor

SGE code overview (cont'd)

daemons/shepherd

- `shepherd.c`
  - read config
  - start prolog/pe/job/epilog
  - do signal handling
  - write usage
  - exit
- `builtin_starter.c`
  - `fork()/exec()` job

SGE code overview (cont'd)

daemons/shepherd (cont'd)

- `setrlimits.c`
  - set Unix resource limits
- `builtin_starter.c`
  - `fork()/exec()` job
- uses code from common/
  - `pdc.c` - kill job
  - `setosjobid.c` - set `'JOBID'`

SGE code overview (cont'd)

- The clients/ directory

– common/	- shared code
– qacct/	- (nearly) a standalone client
– qalter/	- same code as "qresub"
– qconf/	- administrative interface
– qdel/	- delete jobs
– qevent/	- a sample qevent client
– qhost/	- view cluster by hosts
– qsh/	- same code "qrsh", "qlogin"
– qmod/	- change queue and job status

SGE code overview (cont'd)

The clients/ directory (cont'd)

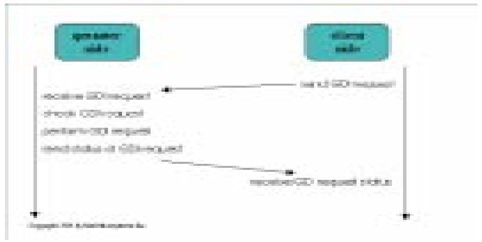
– qmon/	the graphical interface
– qstat/	view cluster by queues
– qsub/	submit jobs
– qhold/qrls	just wrappers for "qalter"

SGE code overview (cont'd)

- Most of the clients commands (exception is qmon of course) have only one or few source files
- The clients mostly do
  - command line parsing
  - setup the request
  - send a GDI request
  - interpret answer from qmaster

SGE code overview (cont'd)

Flow of GDI requests



SGE code overview - I18N

I18N support

- One of the BIG rules at Sun
- I18n/l10n requires to extract messages from source code t ocreate message catalogues
- We needed a mechanism to i13ze source code in a short time which was not designed for it
- Not every (Unix) system supports I18N in an acceptable way (gettext() vs. gettxt()):

  gettext("How now brown cow");

  gettxt("COW:4711", "How now brown cow");

SGE code overview - I18N (cont'd)

- Solution
  - Solaris and Linux support gettext()
  - gettext library available under GPL, therefore not part of the open source project
- SGE messages are defined through a macro in `msg_<dirname>.h`
- Messages are defined as follows:

```
#define MSG_CONF_GETCONF_S _("getting config: %s")
```

  - macros are (more or less) readable
  - macros define the type of their parameters in the end

SGE code overview - I18N (cont'd)

In common/basis_types.h

```
#ifdef __SGE_COMPILE_WITH_GETTEXT__
#  include <libintl.h>
#  include <locale.h>
#  include "sge_language.h"
#  define _(x)   sge_gettext(x)
#else
#  define _(x)     (x)
#endif
```

In the code

```
ERROR((SGE_EVENT, MSG_CONF_GETCONF_S, lGetString(lFirst(alp),
  AN_text)));
```

Messages can be found with tags:

```
% setenv C '*/*.[ch] */*/*.[ch]'
% ctags $C
```

SGE code overview - I18N (cont'd)

Language setup

often done in

`libs/gdi/gdi_setup.c:sge_gdi_setup()`

```
#ifdef __SGE_COMPILE_WITH_GETTEXT__
   /* init language output for gettext() , it will use the right
   language */
   install_language_func((gettext_func_type)        gettext,
                         (setlocale_func_type)      setlocale,
                         (bindtextdomain_func_type) bindtextdomain,
                         (textdomain_func_type)     textdomain);
   sge_init_language(NULL,NULL);
#endif /* __SGE_COMPILE_WITH_GETTEXT__  */
```

SGE code overview - I18N (cont'd)

Functions for I18N

`libs/uti/sge_language.c`

`sge_init_languagefunc`(): Initialize I18N

`sge_gettext`(): "wrapper" for gettext()

SGE code overview - Coding Conventions

- see "coding standards" under Docs
  - gridengine/source/scripts/format.sh
  - expand tabs to blanks, intend 3, wrap at 80
  - function, concept description with adoc for automatic documentation generation
  - ansi C prototypes
  - reuse, extend existing code
  - document new functionalities -> man pages

SGE code overview - TCP

- TCP (Technical Compute Portal)
  - Java servlet glue to GE cli
  - working prototype setups
  - planned for open source May/June 02
  - installation and setup streamlining needed
  - Iplanet Portal Server required
  - Portal Server authentication facility
  - Portal Server channels
  - Proof of concept -> Refinement Projects

SGE code overview - TCP (cont'd)

- TCP - Portal capabilities
  - login to Unix system via web
  - project creation
  - adding of new apps
  - adding of new content channels
  - user specific content layout
  - administer channel availability
  - netlet technology
  - upload/download functionality

SGE code overview - TCP (cont'd)

- TCP - Desirable Features
  - default submit definition
  - undo/redo for portal setup
  - upload/download of a bunch of files
  - project file deletion/renaming
  - Inputform improvements
  - data sharing between projects
  - authentication delegation (e.g. kerberos)
  - easy adding of new apps to portal (XML ?)

Building Grid Engine from scratch

- It's easy and straightforward.
- Open source is the development platform
- get sources via cvs or a source tarball

Compiling SGE (in 4-150 minutes)

– "gmake" required

– Use Sun WorkShop compiler 5.0 or 6.2

```
% cd gridengine/source
(adapt aimk.site and aimk.private if needed)
% aimk -only-depend   - build dependency tool
% scripts/zerodepend  - null out dependencies
% aimk depend         - create dependencies
% aimk                - build SGE
% aimk -secure        - build SGE with OpenSSL
% aimk -debug -j 4    - pass -j 4 option to "make"
% aimk -help
```

Installing SGE in 5 minutes

```
% ln -s scripts/distinst myinst
% setenv SGE_ROOT <my_sge_root>
( % setenv COMMD_PORT 7777 )
% su                  - (optional)
% ./myinst -allall solaris solaris64
% cd $SGE_ROOT
% util/setfileperm.sh - (only if root installs)
% ./install_qmaster
% source default/common/settings.csh
% ./install_execd
% qstat -f
```

More documentation on building SGE

– In gridengine/source:
  - README.BUILD
    – main BUILD document with pointers to further docs
  - README.aimk
    – how aimk works
  - dist/README.arch
    – details of the "arch" script
  - scripts/README.distinst
    – howto install a compiled version of SGE in $SGE_ROOT
  - scripts/README.mk_dist
    – only needed for creators of a distribution

Andre Alefeld
andre.alefeld@sun.com