Grid Engine

Application Integration

Getting Stuff Done.

- Batch
- Interactive - Terminal
- Interactive - X11/GUI
- Licensed Applications
- Parallel Jobs
- DRMAA

Batch Jobs

- Most common
- What is run:
  - Shell Scripts
  - Binaries
  - Scripts that call binaries
- Method:
  - ‘qsub’

Batch Jobs

- Example usage:
  - `qsub -cwd $SGE_ROOT/examples/jobs/sleeper.sh`

Batch Jobs

- Gotchas
  - ‘qsub’ assumes your submission is a script
    - Use “-b y” when directly calling an executable
  - Qmaster param ‘shell_start_mode’
    - Can override your choice of shell/interpreter
    - Are you really getting the shell you requested?

Batch Jobs

- Shortcut
  - Remember "#$" for embedding args into scripts
  - Example:

```
#!/bin/sh

## SGE Arguments
#$ -q all.q
#$ -P MyProject
#$ -l -hard matlabLicense=2

/path/to/my/application.exe
```

Interactive Jobs - Terminal based

- Less common
- What is run:
  - Short executables
  - Remote shells
  - Scripts/apps requiring human input
- Methods:
  - ‘qrsh’
  - ‘`qlogin`’

Interactive Jobs - Terminal based

- Example usage:
  - `qlogin`

```
[hedeby@vcentos-a ~]$ qlogin
Your job 6 ("QLOGIN") has been submitted
waiting for interactive job to be scheduled ...
Your interactive job 6 has been successfully scheduled.
Establishing builtin session to host vcentos-a ...
[hedeby@vcentos-a ~]$ exit
logout
```

Interactive Jobs - Terminal based

- Example usage:
  - qrsh

```
$ qrsh "uname -a; /bin/hostname"

Linux vcentos-a 2.6.18-128.1.6.el5 #1 SMP Wed Apr 1 09:10:25 EDT \
2009 x86_64 x86_64 x86_64 GNU/Linux

vcentos-a
```

Interactive Jobs - Terminal based

- Note
  - ‘qrsh’ is a great way to run quick commands on the least loaded node in the system
- But
  - No guarantee of success
    - If cluster is full, qrsh will return error
    - In workflows need to trap for this
      - or use ‘qrsh -sync y … ’

Interactive Jobs - Terminal based

- Quickly ‘cluster enable’ a binary
  - Trivial NCBI blastall wrapper
  - Give this to your users
    - They use it in the same way they always have
    - Behind the scenes we are invoking it via qrsh

```
#!/bin/sh

# Cluster enable NCBI blastall

qrsh -cwd -now no /opt/bin/blastall $*
```

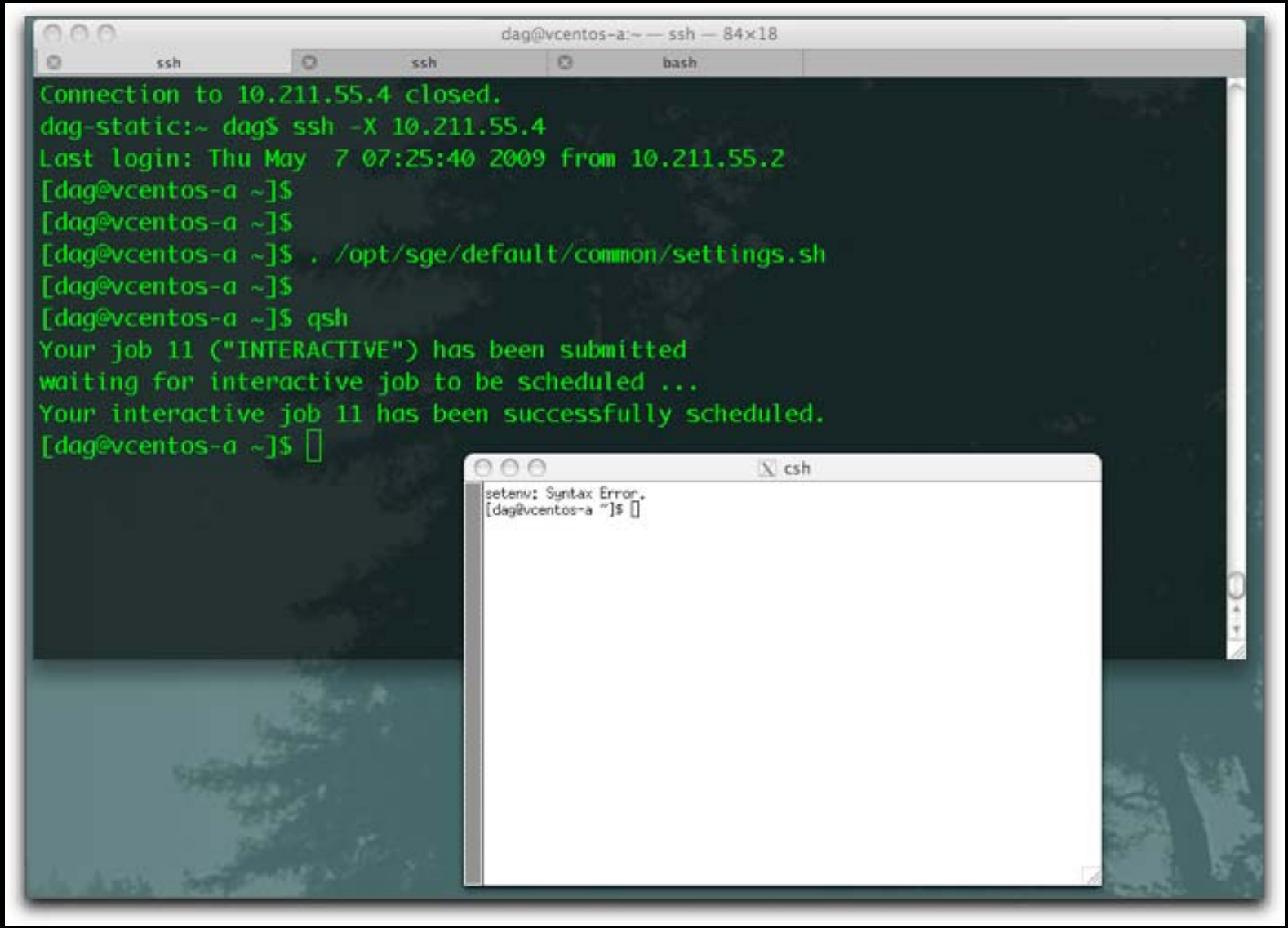
Interactive Jobs - Terminal based

- Gotchas
  - Remember no guarantee of success
    - By default only works if job slots are free
  - Subject to sysadmin modification
    - SGE prior to 6.2 uses custom rsh* methods
      - Many admins replace this with calls to SSH
    - SGE 6.2 and later has better ‘builtin’ method
      - Direct communication with SGE execd
  - Other problems seen at scale
    - Running out of TCP ports or filehandles

Interactive Jobs - Graphical

- Less common
- What is run:
  - Xterm
  - X11 applications
    - MatLab etc.
- Methods:
  - ‘qrsh’
    - Requires SSH & X11 forwarding to be working
  - ‘qsh’
    - SGE spawns xterm for you

Example: Grid xterm via ‘qsh’



Interactive Jobs - Graphical

- Gotchas
  - ‘qrsh’ and ‘qlogin’ methods won’t work in the default config
    - Requires
      - Replace rlogin_* methods with calls to SSH
      - Configure/text X11 forwarding over SSH
  - In SGE 6.2 and later
    - The brand new “builtin” methods don’t support X11 forwarding yet
    - You must replace the builtin methods with SSH based techniques

Licensed Applications

Licensed Applications

- May not may not be a big deal
  - Depends on your license type

Licensed Applications

- You don’t have to worry when:
  - You have a site license
  - There is no scarcity of entitlements
    - You have more FLEXlm tokens than job slots
  - Method
    - Don’t do anything, should just work as usual

Licensed Applications

- Moderate effort required when:
  - More nodes/slots than license entitlements
  - No license server, or
    - Dedicated license server on cluster
- Method
  - SGE Admins will
    - create "*user-requestable consumable resource*" to match license count
  - All you need to do is request this resource when submitting

Licensed Applications

- Somewhat significant effort required when:
  - More nodes/slots than license entitlements
  - An external shared/central license server must be queried
- Method
  - Integrate external license data with Grid Engine job scheduler
  - Old way: “load sensor” scripts
  - Best practice way: Olesen FLEXlm method
    - External perl daemon queries license server
    - Rapid modification of SGE consumable resource values to reflect real world license availability data
    - Much faster than the load sensor method
      - Less chance of race condition

Parallel Jobs

Parallel Jobs

- A parallel job runs simultaneously across multiple servers
  - Biggest SGE job I've heard of:
    - Single application running across 63,000 CPU cores
      - TACC "Ranger" Cluster in Texas

Parallel Jobs

- Setting up the PE is often the hardest part
  - Application specific PE's are normal:
    - To account for:
      - Specific MPI implementation required
      - Desired communication fabric
      - Special network topologies
      - Application-specific MPI starter or stop methods
      - Preference for specific CPU allocation_rule

Parallel Jobs

- No magic involved
  - Requires work
  - Your application must support parallel methods

Parallel Jobs

- Many different software implementations are used to support parallel tasks:
  - MPICH
  - LAM-MPI
  - OpenMPI
  - PVM
  - LINDA

Parallel Jobs

- Loose Integration
  - Grid Engine used for:
    - Picking ***when*** the job runs
    - Picking ***where*** the job runs
    - Generating the custom machine file
  - Grid Engine does not:
    - Launch or control the parallel job itself
    - Track resource consumption or child processes

Parallel Jobs

- Advantages of loose integration
  - Easy to set up
  - Can trivially support almost any parallel application technology

Parallel Jobs

- Disadvantages of loose integration
  - Grid Engine can’t track resource consumption
  - Grid Engine must “trust” the parallel app to honor the custom hostfile
  - Grid Engine can not kill runaway jobs

Parallel Jobs

- Tight Integration
  - Grid Engine handles all aspects of parallel job operation from start to finish
  - Includes spawning and controlling all parallel processes

Parallel Jobs

- Tight integration advantages:
  - Grid Engine remains in control
  - Resource usage accurately tracked
  - Standard commands like "`qdel`" will work
    - Child tasks will not be forgotten about or left untouched

Parallel Jobs

- Tight Integration disadvantages:
  - Can be really hard to implement
  - Makes job debugging and troubleshooting harder
  - May be application specific

Parallel Environment Config

```
pe_name           lam-loose
slots             4
user_lists        NONE
xuser_lists       NONE
start_proc_args   /opt/class/loose-lammpi/startmpi.sh $pe_hostfile
stop_proc_args    /opt/class/loose-lammpi/bin/lamhalt
allocation_rule   $fill_up
control_slaves    FALSE
job_is_first_task TRUE
urgency_slots     min
```

Parallel Environment Usage

- “`qsub -pe lam-loose 4 ./my-mpich-job.sh`”
- “`qsub -pe lam-loose 8-10 ./my-mpich-job.sh`”
- “`qsub -pe lam-loose 3 ./my-lam-job.sh`”

Behind the scenes: MPICH

- Very simple
- The "`startmpi.sh`" script is run before job launches and creates custom machine file
- The user job script gets date required by '`mpirun`' from environment variables:
  - `$NODES, $TEMPDIR/machines`, etc.
- The "`stopmpi.sh`" script is just a placeholder
  - Does not really do anything (no need yet)

Behind the scenes: MPICH

- A trivial MPICH wrapper for Grid Engine:

```
#!/bin/sh

MPIRUN="/usr/local/mpich-1.2.6/ch_p4/bin/mpirun"

## ---- EMBEDDED SGE ARGUMENTS ----
#$ -N MPI_Job
#$ -pe mpich 3-5
#$ -cwd
## ------------------------------------

echo "I got $NSLOTS slots to run on!"
$MPIRUN -np $NSLOTS -machinefile $TMPDIR/machines ./mpi-program
```

Behind the scenes: LAM-MPI

- Very simple
- Just like MPICH
- But 2 additions:
  - The “lamstart.sh” script launches LAMBOOT
  - The “lamstop.sh” script executes LAMHALT at job termination
- In an example configuration, lamboot is started this way:
  - `lamboot -v -ssi boot rsh -ssi rsh_agent "ssh -x -q" $TMPDIR/machines`

Behind the scenes: LAM-MPI

- A trivial LAM-MPI wrapper for Grid Engine:

```
#!/bin/sh

MPIRUN=“/usr/local/bin/mpirun”

## ---- EMBEDDED SGE ARGUMENTS ----
#$ -N MPI_Job
#$ -pe lammpi 3-5
#$ -cwd
## ------------------------------------
echo "I have $NSLOTS slots to run on!"
$MPIRUN C ./mpi-program
```

OpenMPI

- In absence of specific requirements, a great choice
- Works well over Gigabit Ethernet
- Trivial to achieve tight SGE integration
- Recent personal experience:
  - Out of the box: ‘cpi.c’ on 1024 CPUs
  - Out of the box: heavyweight genome analysis pipeline on 650 Nehalem cores

Behind the scenes: OpenMPI

- Incredibly easy/simple
  OpenMPI 1.2.x natively supports automatic tight SGE integration
  - Build from source with “--enable-sge”
  - Usage:
    - mpirun -np $NSLOTS /path-to-my-parallel-app
  - OpenMPI PE config:

```
pe_name           openmpi
slots             4
user_lists        NONE
xuser_lists       NONE
start_proc_args   /bin/true
stop_proc_args    /bin/true
allocation_rule   $round_robin
control_slaves    TRUE
job_is_first_task FALSE
urgency_slots     min
```

OpenMPI Job Script

```
#!/bin/sh

## ---- EMBEDDED SGE ARGUMENTS ----
#$ -N MPI_Job
#$ -pe openmpi 3-5
#$ -cwd
## ------------------------------------

echo "I got $NSLOTS slots to run on!"
mpirun -np $NSLOTS ./my-mpi-program
```

Workflows

Workflows

- Getting a bit into SGE User territory here
- SGE admins often asked to assist with pipelines & workflows
  - You should say ‘yes’
  - Otherwise:
    - Don’t be surprised when a user writes a shell loop that qsub’s a few million jobs

Workflows

- A few simple SGE features can provide significant efficiency gains
  - Benefits both users & cluster operators
- Teach your users effective use of:
  - Job dependency syntax
  - Array Jobs
  - Proper use of “sync -y”

Job Dependencies

- Easy to understand, trivial to implement
  - Job Names
  - -hold_jid
- Very effective building block for workflows and pipelines

Array Jobs

- Perfect solution for use cases involving running the same application many times with only minor changes in input or output arguments
- Both User & Cluster will benefit
- Users will thank you

Array Jobs continued …

- Note:
  - Task interdependency within an array job is a new SGE feature (and open source success story)
  - Have been some issues with array job tasks and specific resource allocation implementation

Workflow cont.

- Consider prolog & epilog scripts for special cases
- Best pipeline integration I've seen
  - DNA Productions (film rendering)
  - Prolog does JDBC insert into SQL database to capture all job details
  - Epilog updates DB after job exit
  - End result:
    - 100% capture of workflow in DB
    - Any job can be repeated exactly at any time

Workflow cont.

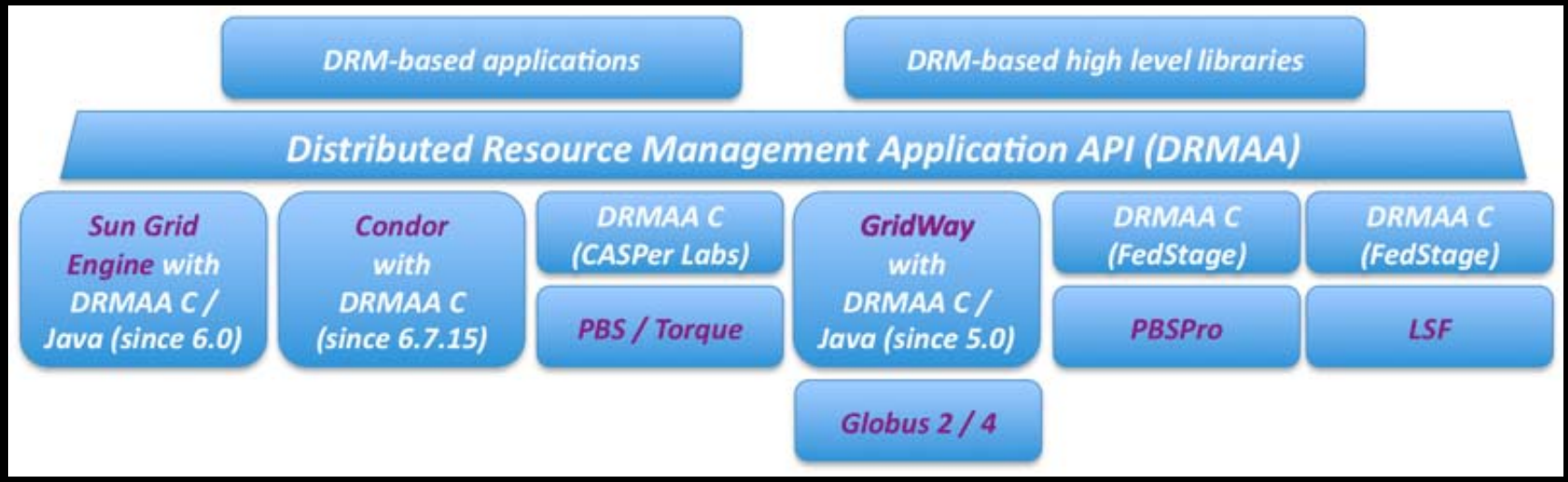
- JSV's may be of help as well
  - I've got a project now that needs one
- And finally
  - For serious cluster-ware code …

DRMAA

DRMAA

- Grid Engine's only public API
- Actually a standard
  - www.drmaa.org
  - "Distributed Resource Management Application API"

DRMAA



Source: www.drmaa.org

DRMAA

- Best practice way for writing “cluster aware” applications
- API concentrates solely on job submission and control
  - This is not a Grid Engine management API
- Multiple language bindings
  - Pick what you like
    - C, C++, Java, Perl, Python, Ruby
    - C and Java seem to be most active

Questions?