Inside the torque-roll

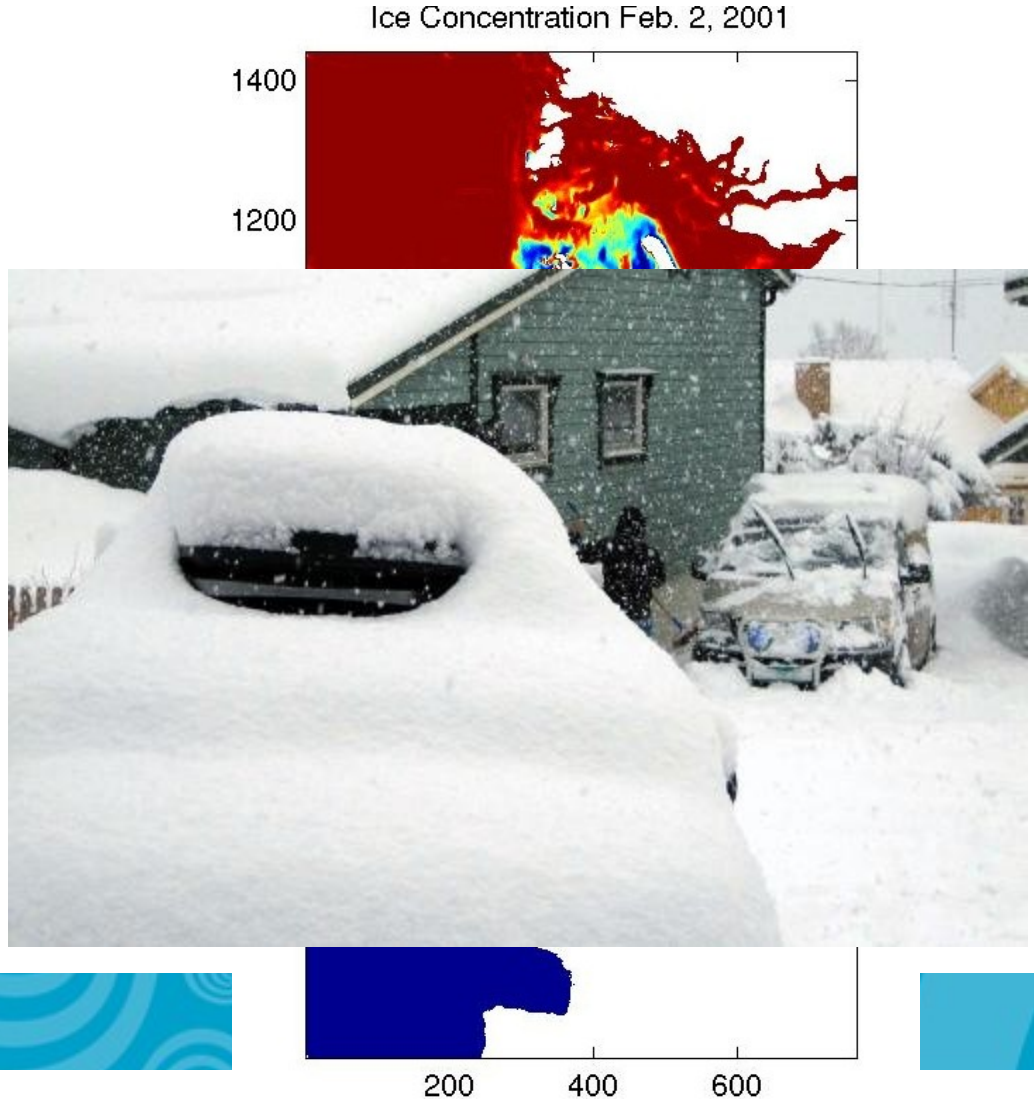
Roy Dragseth
Team Leader, HPC
The Computer Center
University of Tromsø
roy.dragseth@uit.no

University of Tromsø

Northernmost university in the world.

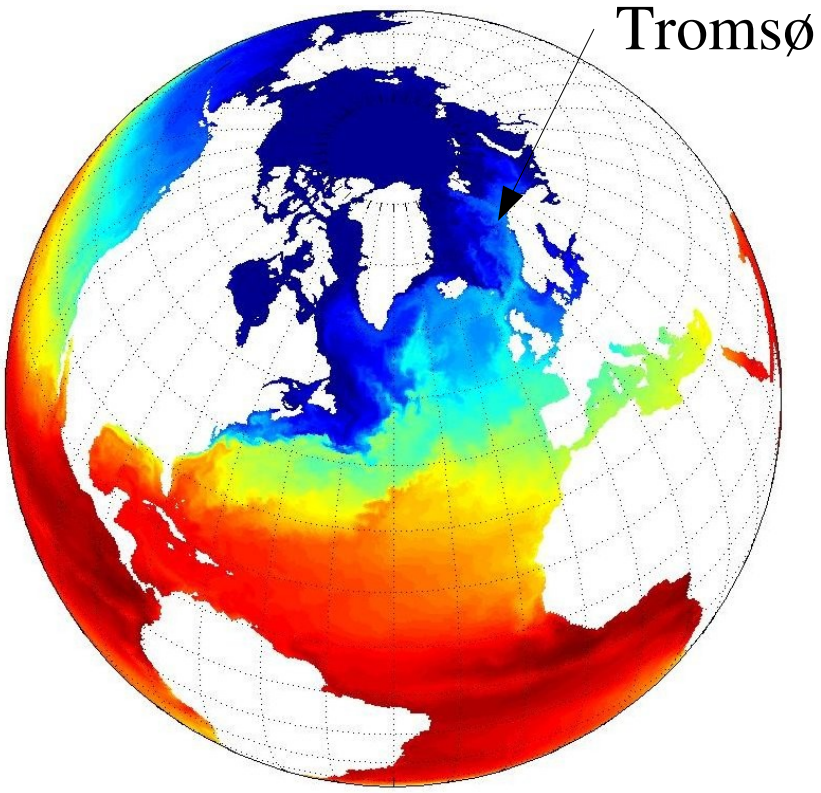
Staff: 2000

Students: 6000



Rocks installation

- stallo.uit.no
- 704 nodes
- 5632 cores
- 12TB RAM
- 60 Tflop/s
- Rocks 4.3
- PBS roll 4.3.0



Topics

- The torque roll
- Scheduling tips and tricks
- Future directions
- Q/A

Topics

- The torque roll
- Scheduling tips and tricks
- Future directions
- Q/A

Software

- Torque, http://www.clusterresources.com/products/torque
- Maui, http://www.clusterresources.com/products/maui
- mpiexec, http://www.osc.edu/~pw/mpiexec/
- pbstools, http://www.osc.edu/~troy/pbs/
- pbspython, ftp://ftp.sara.nl/pub/outgoing/

Where to get it

- Homepage (sadly outdated)

  http://uit.no/itavd/HPC-Rocks-PBS-Roll/
- Download

  ftp://ftp.uit.no/pub/linux/rocks/torque-roll
- Source code

  http://devsrc.cc.uit.no/hg/torque/

DIY

- Clone the repository

  hg clone http://devsrc.cc.uit.no/hg/torque/

- Building is a three step process

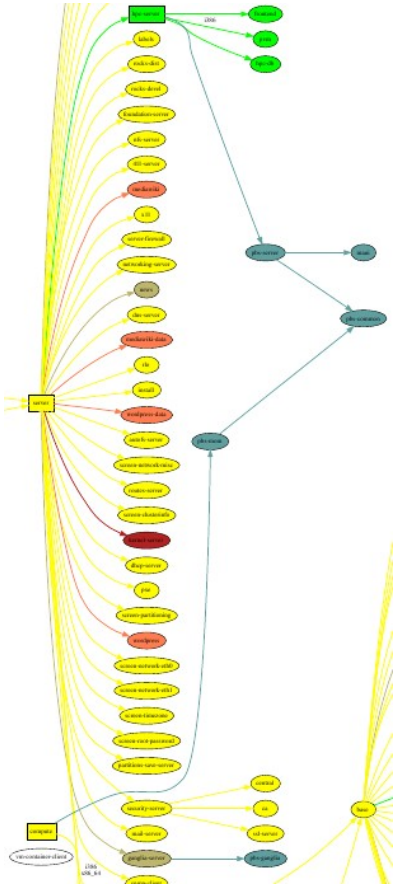
  cd torque/src/torque
  make rpm
  cd ../..
  rpm -i RPMS/x86_64/torque*.rpm

  make roll

The torque rpm build depends on readline-devel and tclx-devel rpms being installed.

Roll graph structure

- Really simple
- No choices at install time
- Default setup should give you a simple FIFO queuing system that just works.



Installed daemons etc

- Frontend
  - maui
  - pbs_server
  - pbs_mom (not running)
  - mpiexec (mostly for the man-page)
- Compute
  - pbs_mom
  - mpiexec

Modifying the setup

- Highly advanced batch scheduling features support by maui.
- If you want to hack the torque node list you need to turn off automatic updates.

  See

  `/etc/torque-roll.conf`

Topics

- The torque roll
- Scheduling tips and tricks
- Future directions
- Q/A

Scheduling features

- Maui provides a rich set of scheduling features
- Maui can schedule on
  - cpus, walltime, memory, disk size, network topology and more...
- We will focus on node distribution and how to make your users behave.

Needed job info

- For scheduling to be useful one needs info about the jobs
  - At least number of cpus and walltime
  - Memory requirements also useful

    #PBS -lwalltime=HH:MM:SS

    #PBS -lnodes=10:ppn=8

    #PBS -lpmem=1gb

Memory handling on linux

- torque/maui supports two memory specification types, (p)mem and (p)vmem on linux.
- pmem is not enforced, used only as information to the scheduler.
- pvmem is enforced, terminating procs that cross the limit.
  - limiting vmem size by setting ulimit -v on the processes

Torque hacking

- Torque is installed in /opt/torque
- qmgr is the torque mgt. command
- Friendly advice: backup your working config

```
# qmgr -c “print server” >
/tmp/pbsconfig.txt
```

Torque hacking

- Roll back to escape from a messed up system:

```
# qterm; pbs_server -t create
# qmgr < /tmp/pbsconfig.txt
```

- This will bring you back to where you started.

  Remark: this will wipe the whole queue setup and all currently queued and running jobs will be lost!

Maui hacking

- Most things can be achieved by modifying

  /opt/maui/maui.cfg

- Maui needs restart after changing the config file

  service maui restart

Advice

- If you can achieve the same thing by changing either torque or maui, use maui.
- Restarting maui is rather lightweight operation, and seldom causes problems for live systems.
- Restarting pbs_server can make the system oscillatory for a few minutes.
  - pbs_server needs to contact all pbs_moms to get back in state.

Prioritizing short jobs

- Often it is useful to give shorter jobs higher priority.
- Use the XFACTOR feature in maui rather than torque queues with different priorites.

  XFACTORWEIGHT 1000

Prioritizing short jobs

- XFACTOR is defined as

  XFACTOR=(walltime+queuetime)/walltime
- XFACTOR will increase faster for shorter walltimes thus giving higher priorities for short jobs.
- Depends on users giving reasonable walltime limits.

Prioritizing large jobs (maui)

- In a cluster with a diverse mix of jobs it is useful to prioritize the large jobs and make the smaller ones fill in the gaps.

  CPUWEIGHT 1000

  MEMWEIGHT 100

- This should be combined with fairshare to avoid starving users falling outside this prioritization.

Fairshare (maui)

- Also known as

  "*Keeping all users equally unhappy*"
- Can be done on several levels
  - users, groups.....
- Set a threshold

  USERCFG[DEFAULT] FSTARGET=10

  FSWEIGHT 100
- Users having used more than 10% will get reduced priority and vice versa.

Adjusting your policy

- You can play with the weights to fine-tune your scheduling policies

  XFACTORWEIGHT 100

  FSWEIGHT 1000

  RESWEIGHT 10

  CPUWEIGHT 1000

  MEMWEIGHT 100

- Analyze the prioritization with

  diagnose -p

Job node distribution

- Default is MINRESOURCE
  - Run on the nodes which gives the least unused resources.
- Spread or pack?

  NODEALLOCATIONPOLICY PRIORITY
  - Select the most busy nodes

    NODECFG[DEFAULT] PRIORITYF=JOBCOUNT
  - Select the least busy nodes

    NODECFG[DEFAULT] PRIORITYF=-1.0*JOBCOUNT

Node access policy

- Default access policy is SHARED
- Can choose to limit this to SINGLEJOB or SINGLEUSER, for instance

  NODEACCESSPOLICY SINGLEUSER
- Single user access prevents users from stepping on each others toes while allowing good utilization for serial jobs.

Throttling policies

- Sometimes one needs to limit the user from taking over the system...
  - MAXPROC, MAXPE, MAXPS, MAXJOB, MAXIJOB
- All can be set for all or individual users and groups
  - USERCFG[DEFAULT], USERCFG[UserA] etc.

Debugging and analyzing

- Lot of tools:
  - pbsnodes -- node status
  - qstat -f -- all details of a job
  - diagnose -n -- node status from maui
  - diagnose -p -- job priority calculation
  - showres -n -- job reservation per node
  - showstart -- obvious
  - checkjob/checknode – also pretty obvious..

Example: express queue

- Goal: Supporting development and job script testing, but prevent misuse
- Basic philosophy:
  - Create a separate queue
  - Give it the highest priority
  - Throttle it so it is barely usable

Example: express queue

Create the queue with qmgr

```
create queue express
set queue express queue_type = Execution
set queue express resources_max.walltime = 08:00:00
set queue express resources_default.nodes = 1:ppn=8
set queue express resources_default.walltime = 08:00:00
set queue express enabled = True
set queue express started = True
```

Example: express queue

Increase the priority and limit the usage

```
CLASSWEIGHT             1000
CLASSCFG[express] PRIORITY=1000 MAXIJOB=1
MAXJOBPERUSER=1 QLIST=express QDEF=express
QOSCFG[express] FLAGS=IGNUSER
```

This will allow users to test job scripts and run interactive jobs with good turnaround

Summary

- Limit the number of queues
- You need good info about walltime

Topics

- The torque roll
- Scheduling tips and tricks
- Future directions
- Q/A

Future

- Fix integration with openmpi
  - Many versions to support, will probably focus on the one supplied with Rocks.
- BLCR support
  - Would be really nice to have job migrating between nodes, job preemption etc.
  - Still a long way to go...
- New ganglia frontend.

Future

- Make it SGE friendly?
- Integrate with the rocks command line
- Actually write some documentation...

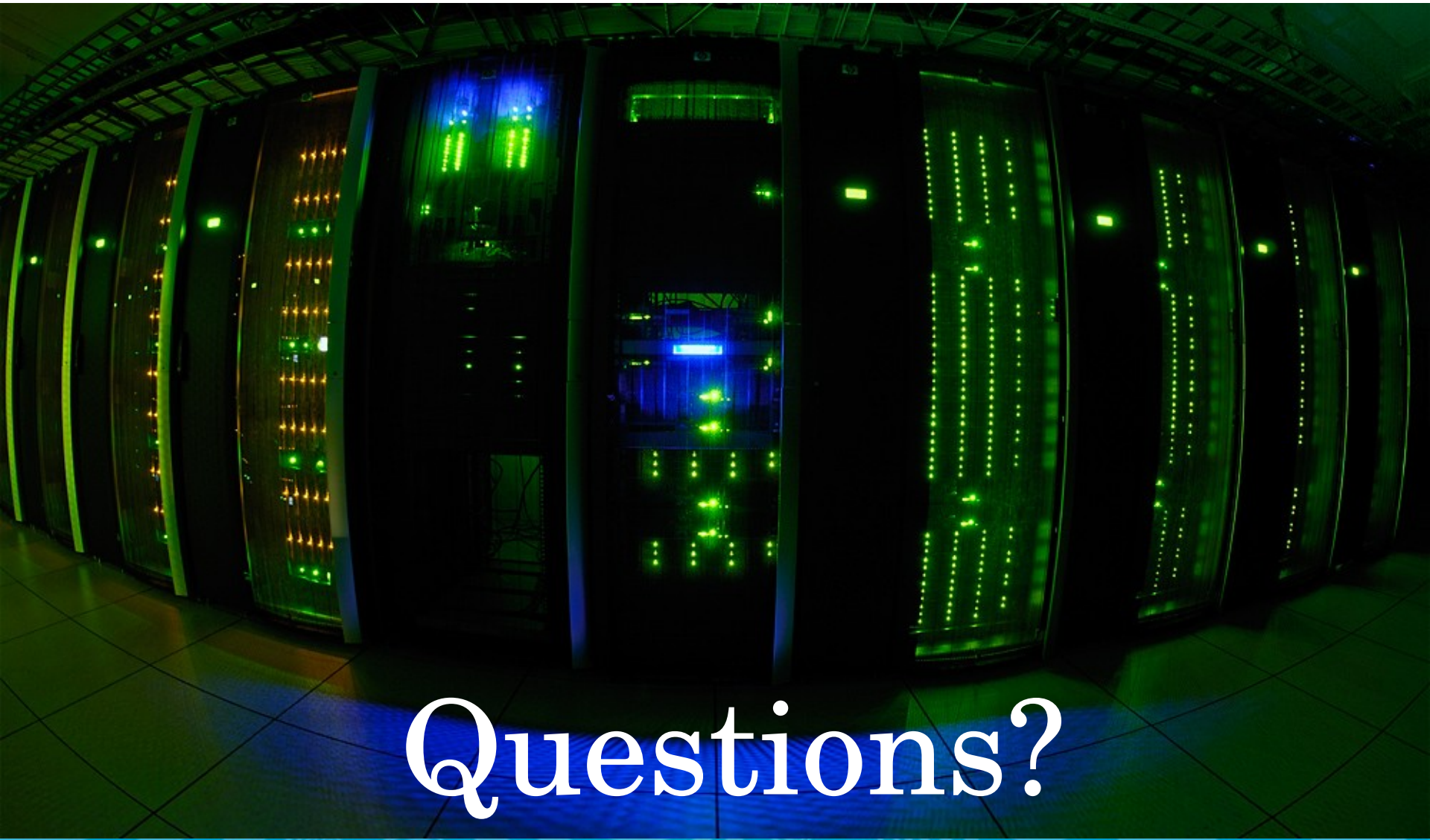
Topics

- The torque roll
- Scheduling tips and tricks
- Future directions
- Q/A

Getting help

- Rocks mailing list.

  (mention torque, maui or pbs in the subject)
- The torque and maui sites are good starting points for info.
- Torque and maui also have friendly mailing lists.
- The torque-roll is known to CR.
- You can buy moab from ClusterCorp


Questions?