The TORQUE Resource Manager

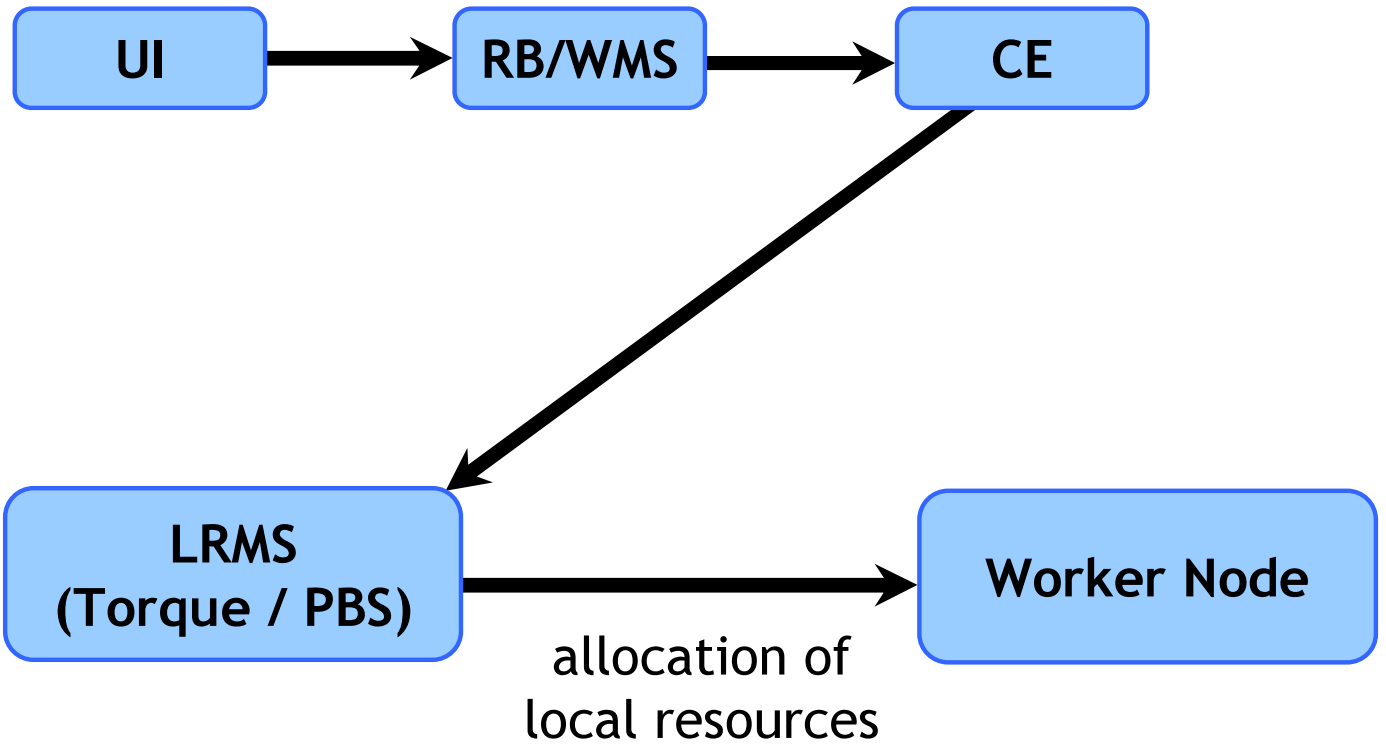
Vangelis Koukis
Computing Systems Laboratory – ICCS
vkoukis@cslab.ece.ntua.gr

Εκπαίδευση στο πλαίσιο του έργου «Εγκατάσταση κόμβου GRID στο Πολυτεχνείο Κρήτης»
2009/05/07

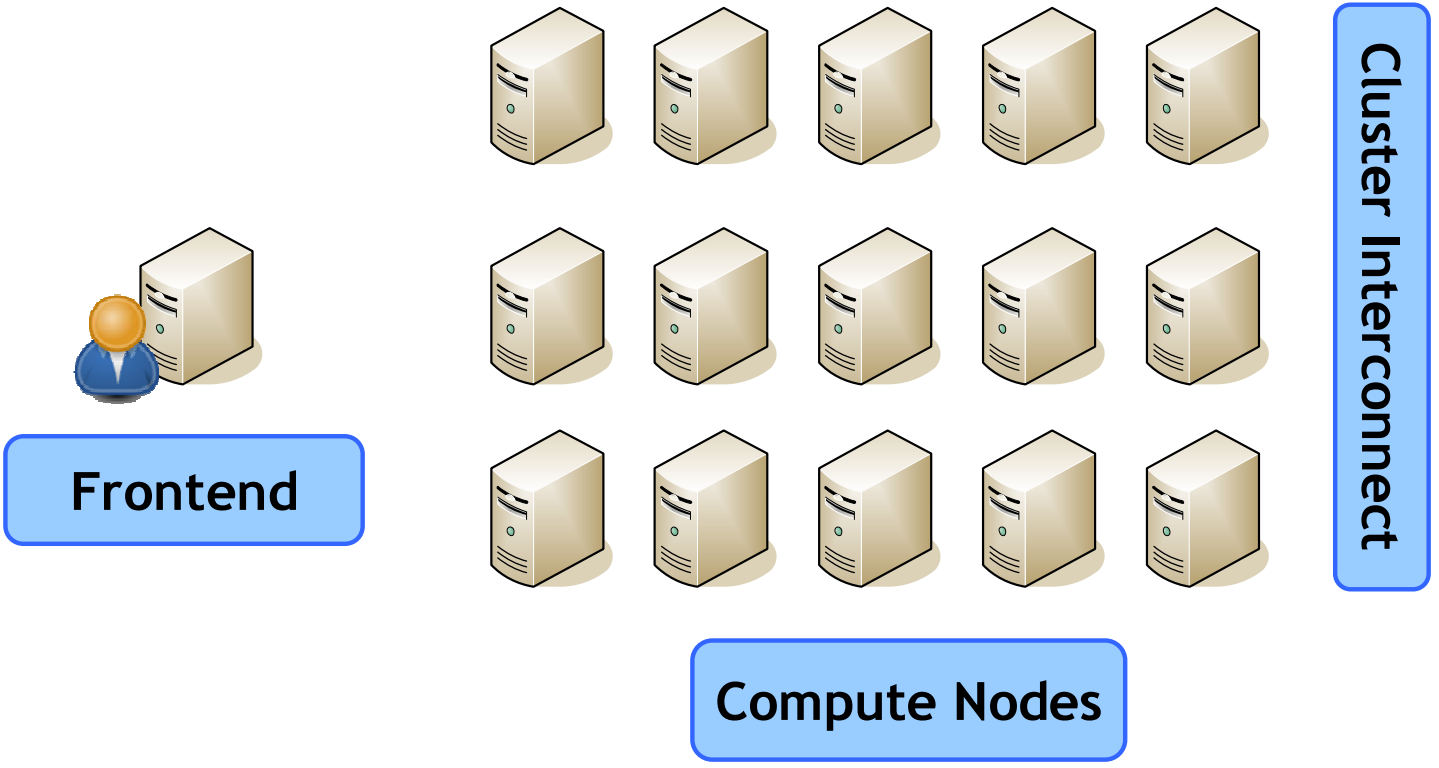
What is TORQUE?

- TORQUE: Terascale Open-source Resource and QUEue manager
  - A batch scheduling system for clusters
    - Based on *PBS (PBS, OpenPBS, PBS Pro), NASA, early ’90s
  - Managing local resources
    - Compute nodes, perhaps multicore systems
    - Allocates cores and memory to jobs
  - And job queues
    - Accepts job submission requests
    - Manages jobs in queues
    - Creates needed processes, responsible for starting/suspending/killing/jobs

The lifetime of a job on the Grid



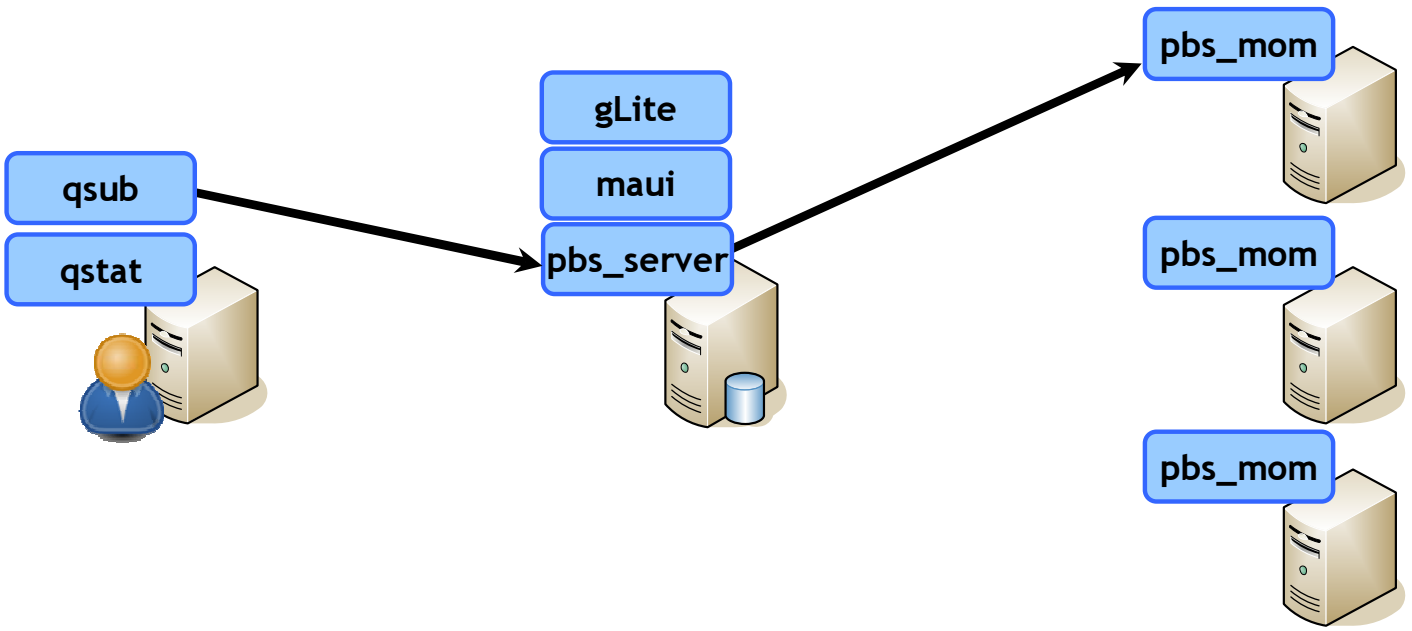
What exactly is a compute cluster?



What exactly is a compute cluster?

- No single definition but:
- Many, similar compute nodes that share:
  - Administrative authority
  - User accounts and privileges
  - Home directories for users (/home/user1)
    - Although local scratch space is usually available
- Combined with a scheduling system
  - e.g., TORQUE and MAUI
- Used for running parallel applications
  - e.g., combined with an MPI implementation

Using TORQUE locally



Submit host (frontend)

PBS Server (CE)

Compute Nodes (WNs)

Basic Job Management

- qsub
  - submit a job to a queue
- qdel
  - cancel/delete a job
- qstat
  - view queue and job status
- qsig
  - signal a job (e.g. SIGTERM, SIGUSR1)
- qhold / qrls
  - hold / release a job

qsub: Job Submission (1)

- Submit a job script to a queue
  - qsub -q tuc -I -k oe -l nodes=4 -N testjob1
- -q *queue*: the queue to submit to ("tuc")
- -I: request an interactive job
  - gets scheduled normally, but with I/O redirection to user's terminal
- -l nodes=*node_specification*
  - More on that later on

qsub: Job Submission (2)

- Submit a job script to a queue
  - qsub -q tuc -I -k oe -l nodes=4 -N testjob1
- -N *jobname*
  - Define a nice human readable name for the job
- -k oe
  - *Important*: Keep both standard output and standard error on the execution host
  - Useful, because UI shares homes with WNs for local accounts

qdel: Job Cancellation / Deletion

- Remove a job from the queue
  - kill it if it's already running
- Sends a SIGTERM first, then a SIGKILL
- qdel –W 3 1122
  - Delete job with job id 1122
  - Wait 3 seconds between SIGTERM and SIGKILL

qstat: Queue / Job Statistics

- Show processes currently in the queue
- Along with their state and attributes
  - E: Job is exiting after having run
  - H: Job is held
  - Q: Job is queued, eligible to run or be routed
  - R: Job is Running
  - T: Job is in transition (being moved to a new location)
  - W: Job is waiting for its requested execution time to be reached
  - S: Job is suspended
- More arguments: -a [all jobs], -f [full status]

qalter: Alter Job attributes

- Alters job attributes
  - Either while in the queue or running
  - E.g. maximum wallclock or CPU time
- Job will be cancelled if its new attributes do not fit queue requirements

qsig: Signal a Running Job

- Sends a UNIX signal to a running job
- E.g., SIGINT, or a user-defined signal
- User-defined signals (SIGUSR1/2) used to invoke user-specified reactions
  - Output progress statistics to a predefined file
  - Perform internal checkpointing, to resume work from this point later on
- Special signals: “suspend”, “resume”
  - suspend: send SIGTSTP (^Z), then SIGSTOP
  - resume: send SIGCONT

qsub: Specifying resources (1)

- -l resource1=value1,resource2=value2
- CPU resources:
  - -l nodes=X
  - -l nodes=X1:ppn=Y1+X2:ppn=Y2
  - -l nodes=X1:ppn=Y1:myrinet
  - -l nodes=wn030.grid.tuc.gr +wn002.grid.tuc.gr

qsub: Specifying resources (2)

- -l resource1=value1,resource2=value2
- Either while in the queue or running
  - E.g. maximum wallclock or CPU time
- Job will be cancelled if it's no longer runnable based on its new attributes

qsub: Specifiying dependencies

- -W depend=type[:argument…]
- E.g., start the job:
  - Any time after *j* has started:
    -W depend=after:j
  - Only if *j* completes successfully:
    -W depend=afterok:j
  - Only if j fails:
    -W depend=afternotok:j
- More options in the manual page for qsub
  - man qsub

PBS Scripts (1)

- Used as input for qsub
  - Commonly a simple bash script
- Deals with job initialization and finalization
- Can contain PBS-specific comments
  - Begin with “#PBS”
  - No need to specify long argument lists to **qsub**
- Runnable directly at the command line

PBS Scripts (2)

- A simple example:

```
#!/bin/bash
#PBS -l nodes=4:ppn=2
#PBS -l walltime=01:00
#PBS -q tuc
#PBS -k oe

echo Running on `hostname`

echo The PBS node file contains:
if [ ! -z $PBS_NODEFILE ]; then
        cat $PBS_NODEFILE
else
        echo No $PBS_NODEFILE found.
fi
```

Job Arrays

- Single script, submitted in multiple jobs
- -t argument to qsub:
  - -t *index_range*[,*index_range*]
- For example
  - qsub –t 1-100 array.pbs
  - qsub –t 1,3,5,7,10-15 array.pbs
- Each job can find its place in the array by examining $PBS_ARRAYID

Multiprocessor Jobs

- Job has been allocated a number of nodes, now what?
  - Use the TORQUE Task Management (TM) interface to spawn peer tasks on remote nodes
- **pbsdsh:** TORQUE/PBS-specific rsh/ssh replacement
- TORQUE-aware MPI implementation
  - OpenMPI has excellent support

Multiprocessor Jobs: pbsdsh

- TORQUE-aware rsh/ssh replacement
- pbsdsh *command*
  - Discovers set of nodes in job
  - Execute command on *all* tasks
- Useful arguments
  - -h *host*: execute on a single host
  - -u: execute command once on each host
    - Ignores number of allocated processors per node
- Careful! command must be an absolute path: e.g., `pwd`/task.sh

Hands-on Time!

- http://www.cslab.ece.ntua.gr/tuc/torque.html

Multiprocessor Jobs: MPI

- OpenMPI has full integration with TORQUE
  - Based on TM interface
  - Determines number and placement of processors automatically
    - examines environment variable
    - Queries TORQUE through TM interface
- Proper signaling to MPI job
- Proper job initialization, suspension, and termination
- Correct accounting, real CPU time

Multiprocessor Jobs: Control

- No unrestricted rsh/ssh between nodes
- TORQUE stays in control of processor allocation and CPU time usage
- Linux cpusets used to minimize process interference between unrelated jobs
  - Even if process fork()s, all related processes time-share
- Enables user to oversubscribe allocated processors, for testing purposes
  - number of MPI peers > number of processors

Torque accounting / logs

- TORQUE keeps detailed logs under /var/spool/pbs
- PBS server: server_logs/, server_priv/
- PBS mom: mom_logs/, mom_priv/
- Various debug levels settable via **qmgr**
- Every state transition gets logged and can be analyzed during troubleshooting