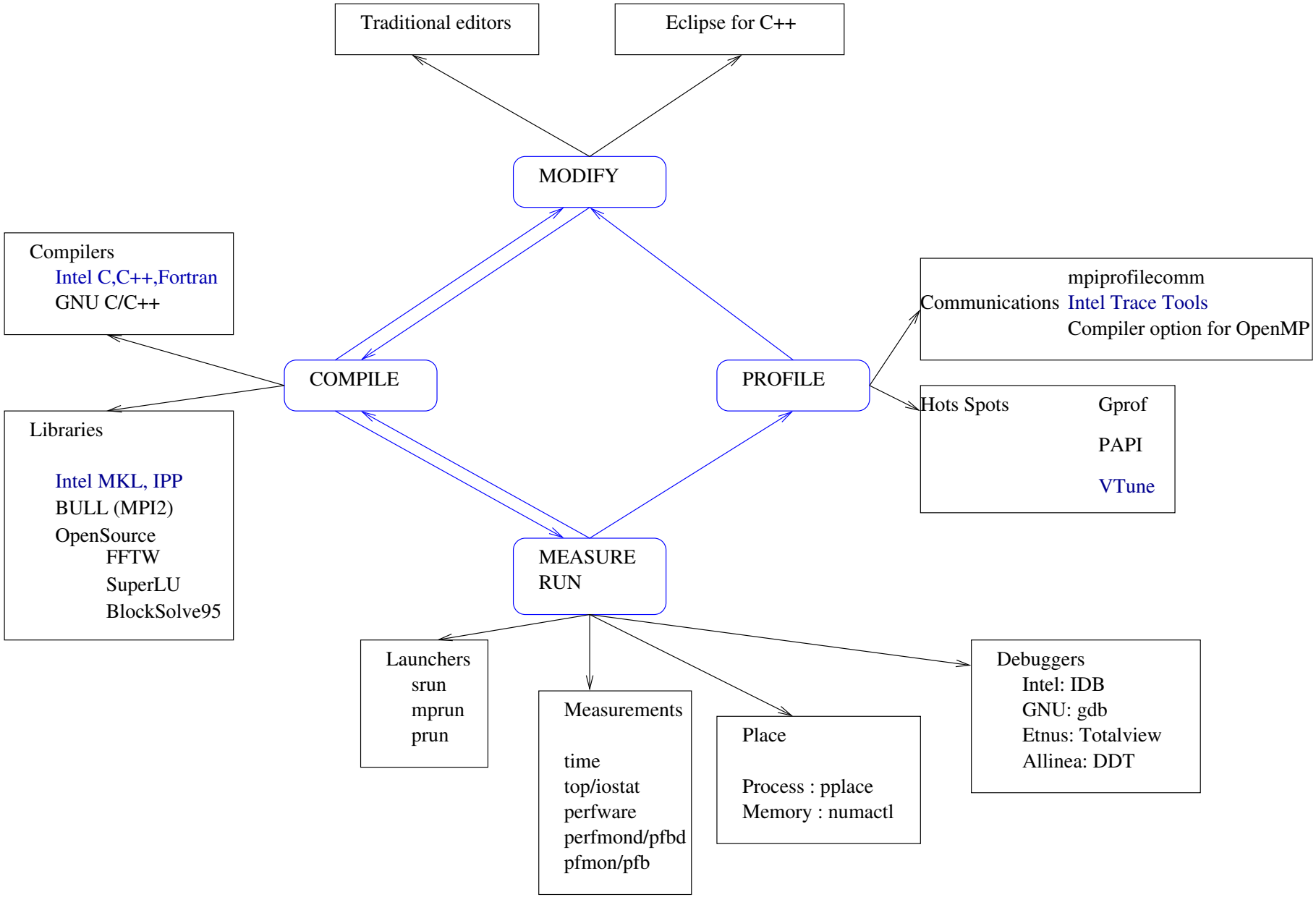
Performance HPC Linux
Bull Echirolles

Software Development Environment

(Part two)

C.Berthelot
Christophe.Berthelot@bull.net

2008


Traditional editors
Eclipse for C++
MODIFY
Compilers
Intel C,C++,Fortran
GNU C/C++
COMPILE
Libraries
Intel MKL, IPP
BULL (MPI2)
OpenSource
FFTW
SuperLU
BlockSolve95
PROFILE
Communications
mpiprofilecomm
Intel Trace Tools
Compiler option for OpenMP
Hots Spots
Gprof
PAPI
VTune
MEASURE
RUN
Launchers
srun
mprun
prun
Measurements
time
top/iostat
perfware
perfmond/pfbd
pfmon/pfb
Place
Process : pplace
Memory : numactl
Debuggers
Intel: IDB
GNU: gdb
Etnus: Totalview
Allinea: DDT

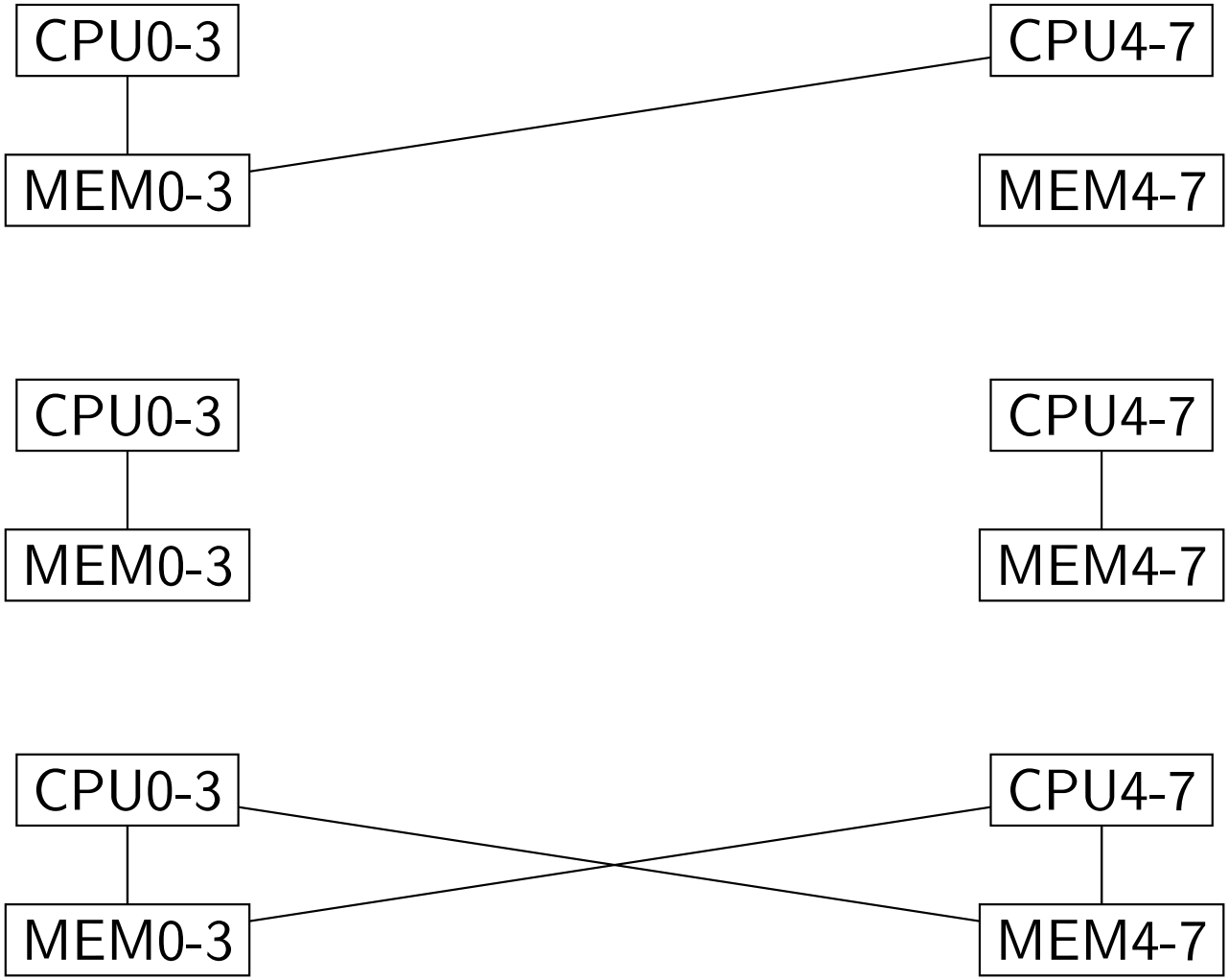
- Introduction
- Tools
- How to run with Torque
- How to measure

Data placement policy

First touch

- First processor to touch a page of memory causes it to be allocated from its local memory
- Works well for fully parallelized programs, but serial initialization cause non-local accesses and bottlenecks on memory bandwidth.

Memory placement



- Introduction
- Tools
  CPU placement
  Memory placement
- How to run with Torque
- How to measure

Process placement: Taskset

- `taskset` is used to set or retrieve the CPU affinity of a running process given its PID or to launch a new COMMAND with a given CPU affinity

Example

- To run inside on the first 4 cores

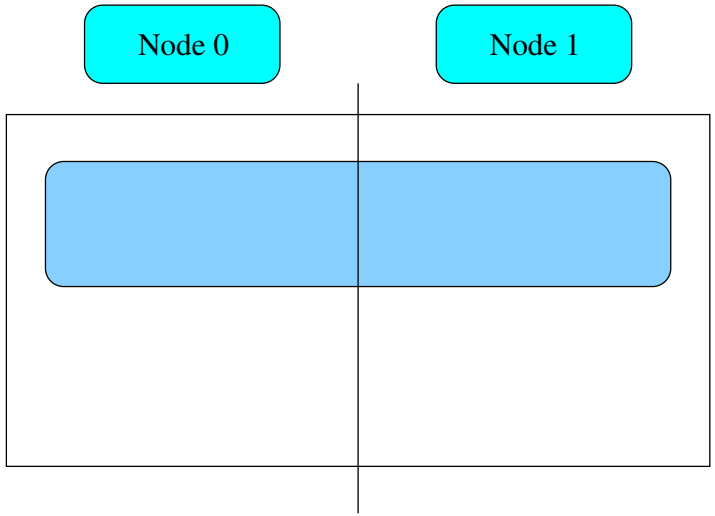
```
taskset -c 0-3 ./a.out
```

- Introduction
- Tools
  CPU placement
  Memory placement
- How to run with Torque
- How to measure

Memory placement : Numactl

Numactl runs processes with a specific NUMA scheduling or memory placement policy. The policy is set for command and inherited by all of its children. Policy settings are:

- Memory will be allocated using round robin on nodes.
- Allocates only memory from nodes.
- Does always local allocation on the current node.



On NovaScale

- A node is a NDC
- Option –hardware shows inventory of available nodes on the system

```
available: 2 nodes (0-1)
node 0 size: 16166 MB
node 0 free: 11246 MB
node 1 size: 16384 MB
node 1 free: 13846 MB
```

- Option –show shows NUMA policy settings of the current process

```
policy: default
preferred node: current
cpubind: 0 1
membind: 0 1
```

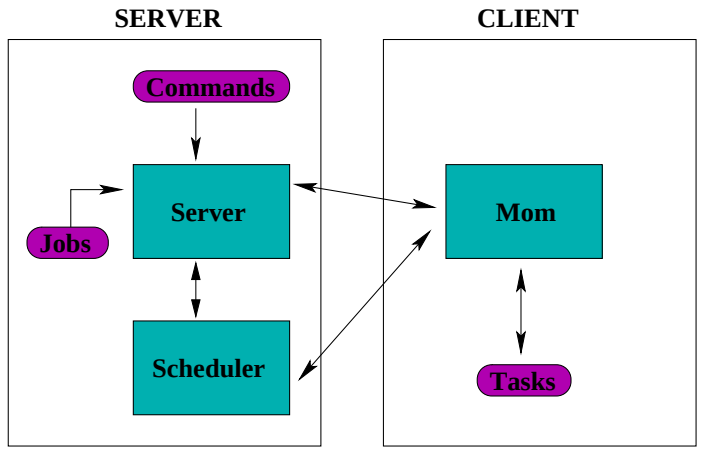
Commands

- `--interleave=nodes, -i nodes`: Set an memory interleave policy.
- `--membind=nodes, -m nodes` : Only allocate memory from nodes.
- `--localalloc, -l` : Do always local allocation on the current node
- `--cpubind=nodes, -c nodes` : Only execute process on the CPUs of nodes

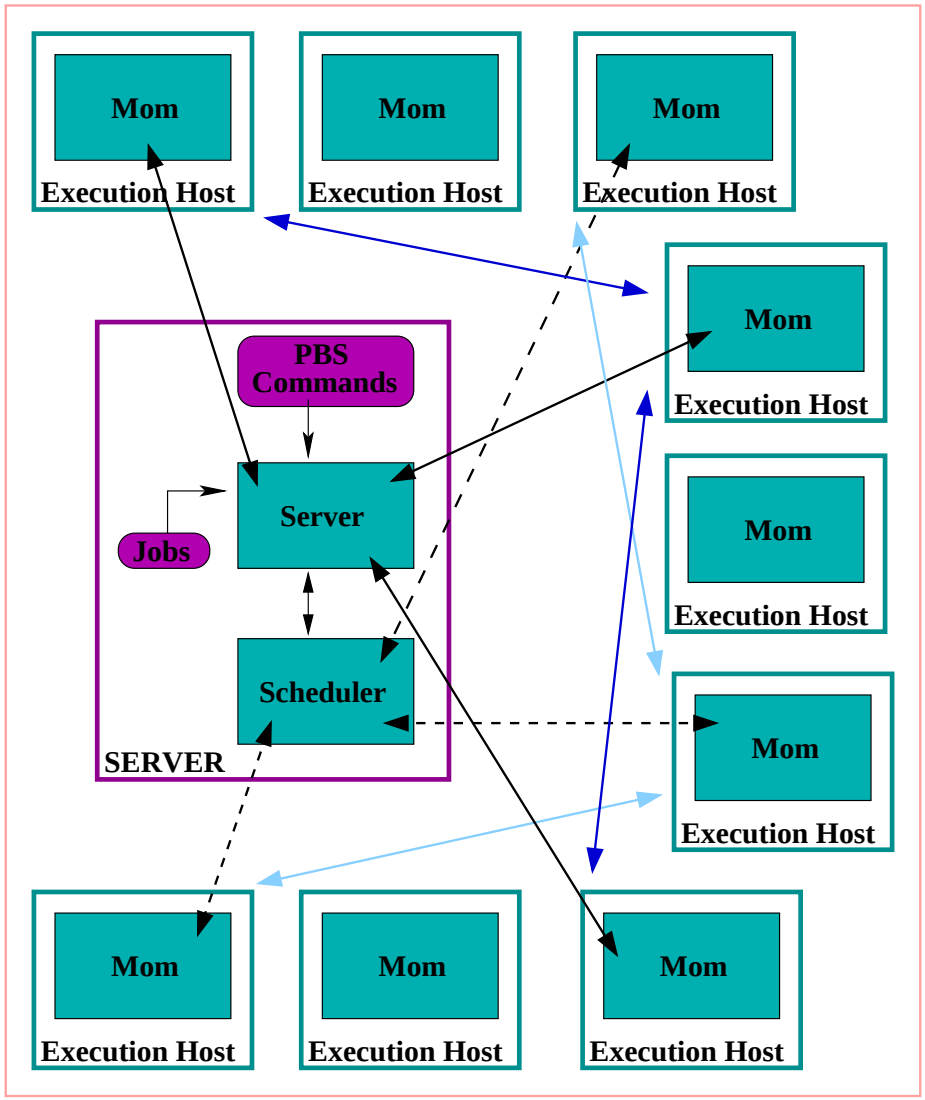
- Introduction
- Tools
- How to run with Torque
- How to measure

TORQUE

- A Server
  - Receives batch jobs
  - Invokes the scheduler
  - Instructs moms to execute jobs
- A scheduler
  - Contains the job scheduling policy
  - Communicates with "moms" to learn about state of system
  - Communicates with "server" to learn the jobs availability
- Mom(s) (Machine Oriented Miniserver)
  - Places jobs into execution
  - Takes instruction from "server"



Software Architecture on the Cluster



Torque Server

Torque Mom

Communications:

Torque

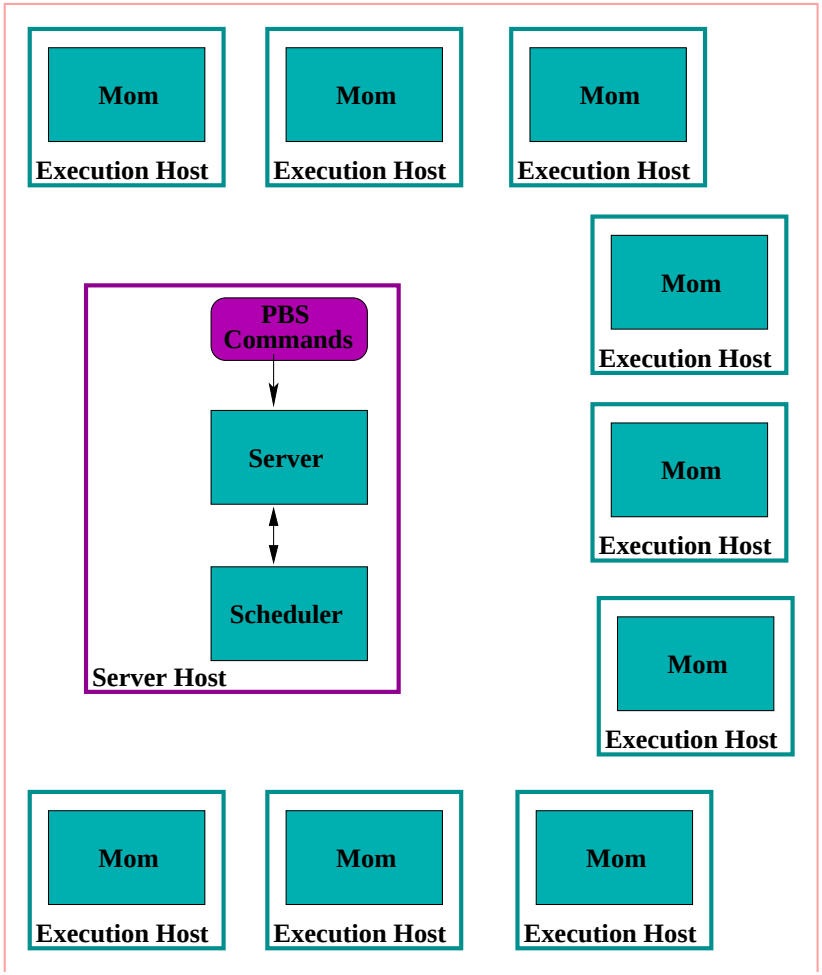
Appli 1

Appli 2

Job session scheme

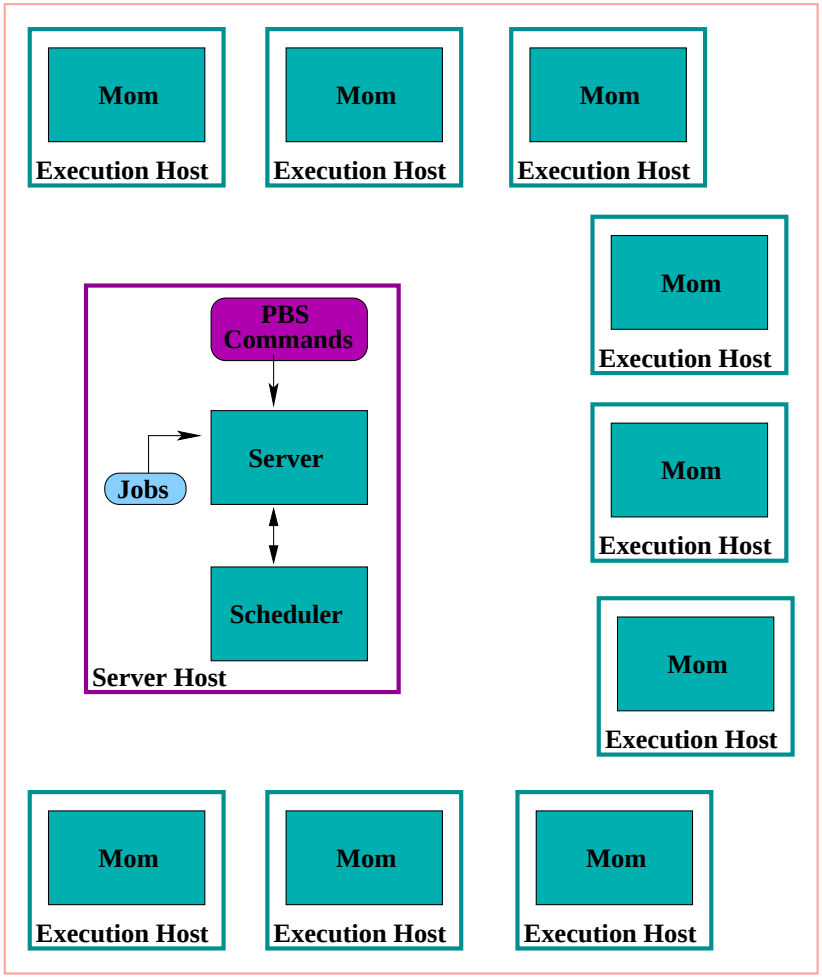
Structure:

- There is one Server
- There is one Scheduler
- One Mom for each compute node



Job session scheme

1- User specifies resource requirements and submits his job (qsub command)



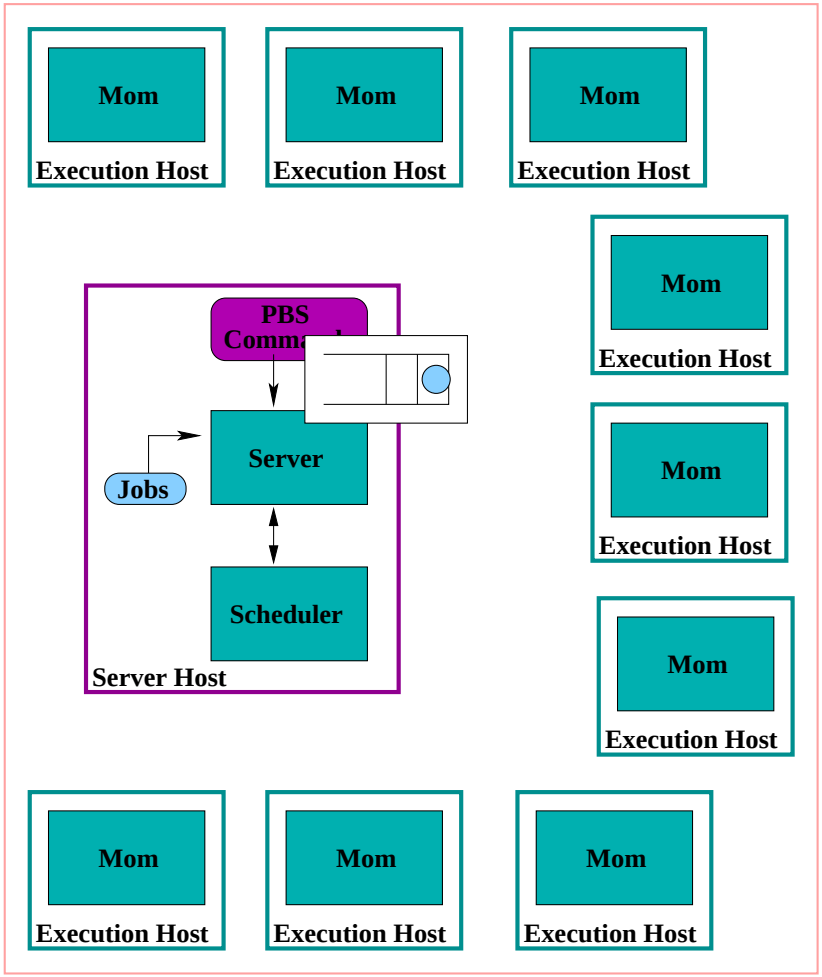
Job session scheme

1- User specifies resource requirements and submits his job (qsub command)

2- Server places the job into a queue

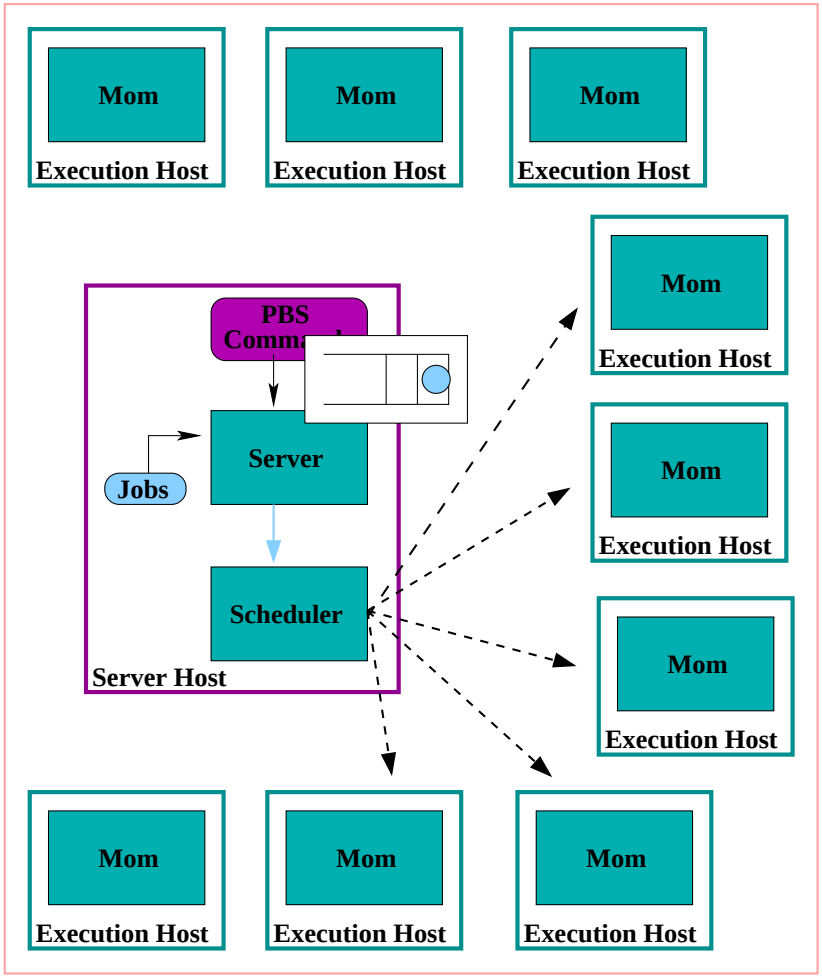
(the queue choice depends on the resource requests)

and initiates the scheduling cycle



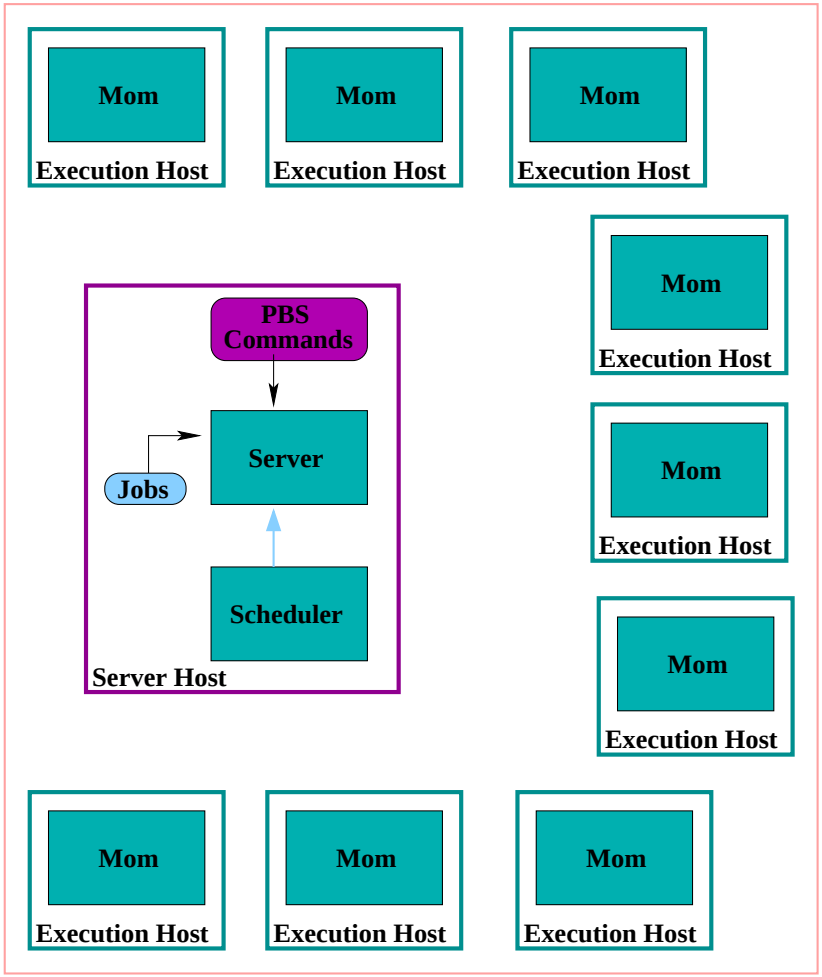
Job session scheme

1- User specifies resource requirements and submits his job (qsub command)
2- Server places the job into a queue
(the queue choice depends on the resource requests)
and initiates the scheduling cycle
3- Scheduler asks moms to determine available resources



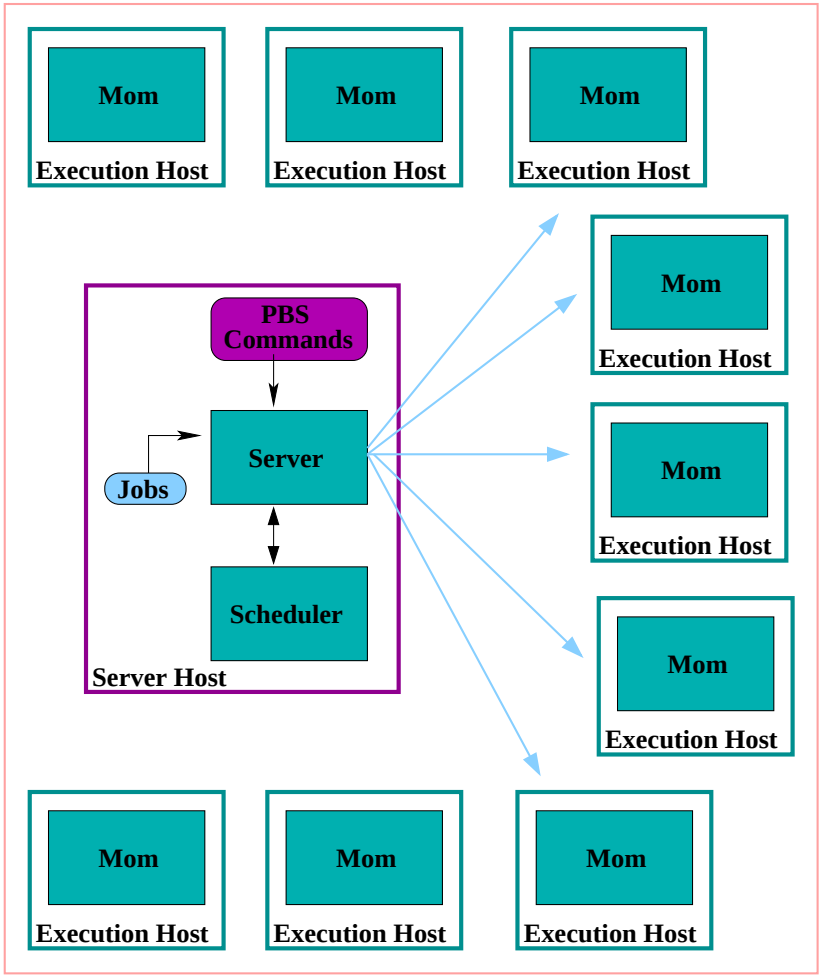
Job session scheme

1- User specifies resource requirements and submits his job (qsub command)
2- Server places the job into a queue
(the queue choice depends on the resource requests)
and initiates the scheduling cycle
3- Scheduler asks moms to determine available resources
4- Scheduler eventually allocates resources for the job
(when required resources become available)



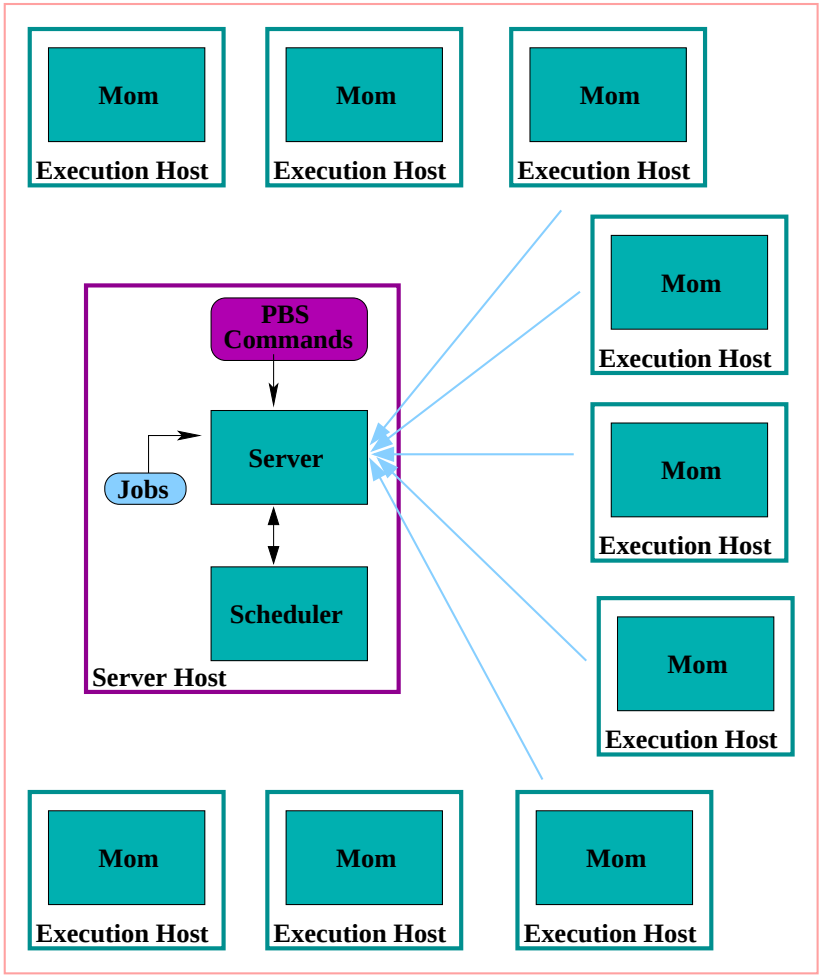
Job session scheme

1- User specifies resource requirements and submits his job (qsub command)
2- Server places the job into a queue
(the queue choice depends on the resource requests)
and initiates the scheduling cycle
3- Scheduler asks moms to determine available resources
4- Scheduler eventually allocates resources for the job
(when required resources become available)
5- Server dispatches tasks to moms deamons



Job session scheme

1- User specifies resource requirements and submits his job (qsub command)
2- Server places the job into a queue
(the queue choice depends on the resource requests)
and initiates the scheduling cycle
3- Scheduler asks moms to determine available resources
4- Scheduler eventually allocates resources for the job
(when required resources become available)
5- Server dispatches tasks to moms deamons
6- When task is finished, Mom sends result to the Server



Essential commands

- Submitting a job:
  - > **qsub** *myjob*
  - options:
- Monitoring
  - > **qstat -an**
  - > **pbsnodes -a**
- killing a job
  - > **qdel** *myjobID*

Torque: on the user side

Submission command: **qsub**

Interactive submission

- Example 1:
  - → qsub -I
  - → cat $PBS_NODEFILE (this variable gives the allocated nodes description file)
  - → exit
- Example 2:
  - → qsub -I -lnodes=2:bigmem
- Example 3:
  - → qsub -I -lnodes=1:bigmem+2:lowmem

Batch submission

- User script:
  - → #PBS -lnodes=2
  - → uname -a
  - → cat $PBS_NODEFILE
- MPI script:
  - → #PBS -lnodes=2
  - → cd ${PBS_O_WORKDIR}
  - → mpirun -np N -machinefile ${PBS_NODEFILE} test_prog

Torque: on the user side

Batch script example (sequential prog)

```
#!/bin/sh
#PBS -N test                                  → Job name
#PBS -r n                                     → Declare job non-rerunable
#PBS -e test.err                              → Output files
#PBS -o test.log
#PBS -m ae                                    → Mail to user
#PBS -q long                                  → Queue name (small, medium,...)
#PBS -l nodes=1:bigmem                        → Number of nodes (and property)
echo Working directory is $PBS_O_WORKDIR      → This job's working directory
cd $PBS_O_WORKDIR
echo Running on host `hostname`
echo Time is `date`
echo Directory is `pwd`
a.out                                         → Run your executable
```

Torque: on the user side

Batch script example (parallel prog)

```
#!/bin/sh
#PBS -N test
#PBS -r n
#PBS -e test.err
#PBS -o test.log
#PBS -m ae
#PBS -q long
#PBS -l nodes=8:bigmem
uniq $PBS_NODEFILE > mpd.hosts
echo This jobs runs on the following processors:
echo `cat $PBS_NODEFILE`
NPROCS=`wc -l < $PBS_NODEFILE`
np=$(wc -l < mpd.hosts)
echo This job has allocated $NPROCS nodes
mpdboot -n $np -f mpd.hosts -r ssh -v
mpiexec -n $NPROCS a.out
mpiallexit
```

- → Job name
- → Declare job non-rerunable
- → Output files
- → Mail to user
- → Queue name (small, medium,...)
- → Number of nodes (and property)
- → Define file with nodes names
- → Define number of processors
- → Start mpd
- → Run the parallel MPI executable
- → Stop mpd

- Introduction
- Tools
- How to run with Torque
- How to measure
  - First Level
  - Advanced Level

First Level

Measurement the time: *time*

Only User, Real and System time

Measurement IO *iostat*

System input/output device loading by observing the time the devices are active

Measurement Virtual Memory Statistics *vmstat*

vmstat reports information about processes, memory, paging, block IO, traps, and cpu activity.

Measurement network *netstat*

netstat displays the contents of various network-related data structures.

Advanced Level: *perfmon*

The pfmon tool is a simple monitoring tool which can be used to collect simple counts or samples from unmodified binaries or an entire system. It is using the capabilities of the IA-64 PMU

- The pfmon command can be executed on a program, taking into account, or not, its descendants. It returns from 1 to 4 known events or counters on Madison and 12 on Montecito.
- Advantages
  - Powerful with all Itanium hardware counters
  - Light nothing to do inside your code

Metrics

- MFLOPS

$$\frac{FP\_OPS\_RETIRED \times frequence}{CPU\_CYCLES}$$

- IPC (maximu 6)

$$\frac{IA64\_INST\_RETIRED}{CPU\_CYCLES}$$

- FPI (maximu 1)

$$\frac{FP\_OPS\_RETIRED}{IA64\_INST\_RETIRED}$$

How to use pfmon

- Counters
  - `-e`*comp1,comp2,comp3,comp4,...*
- Two modes (kernel and user)
  - `-u` user
  - `-k` kernel
- follow fork, vfork, exec, pthreads
  - `--follow-all`
- Save results
  - `--outfile=xxx`
  - `--append`

Example

With Benchmark CG from NAS

- On 4 first CPUs (KMP_AFFINITY=fine,compact)

```
pfmon -e FP_OPS_RETIRED,IA64_INST_RETIRED,CPU_CYCLES ./cg.B
```

- Results pfmon

```
 6992942034 FP_OPS_RETIRED
89796803322 IA64_INST_RETIRED
72943569333 CPU_CYCLES
```

- Metrics

	FPI	IPC	GigaFlops
Maximum	1	6	6.40
DGEMM	0.79	4.83	6.14
cg	0.08	1.23	0.15

Example

With Benchmark CG from NAS

- On 4 CPUs (2/NDC) (KMP_AFFINITY=fine,scatter)

```
pfmon -eFP_OPS_RETIRED,IA64_INST_RETIRED,CPU_CYCLES ./cg.B
```

- Results pfmon

```
 6992941814 FP_OPS_RETIRED
59184694803 IA64_INST_RETIRED
43702826105 CPU_CYCLES
```

- Metrics

	FPI	IPC	GigaFlops
Maximum	1	6	6.40
DGEMM	0.79	4.83	6.14
cg	0.12	1.35	0.25

(To be continued)

Architect of an Open World™

- (c) Copyright Bull. All rights reserved
  - ✔ Users Restricted Rights - Use, duplication or disclosure restricted.
  - ✔ Any copy of these documents should keep all copyright, logos and other proprietary notices contained herein.
  - ✔ This publication may include technical inaccuracies or typographical errors.
  - ✔ This publication is provided "AS IS" without any warranty either expressed or implied including but not limited to the implied warranties of merchantabilities or fitness of the described product.
  - ✔ Course Material Licensing Terms : No sublicensing rights.
  - ✔ For other licensing needs, please contact Bull