# Supercomputers

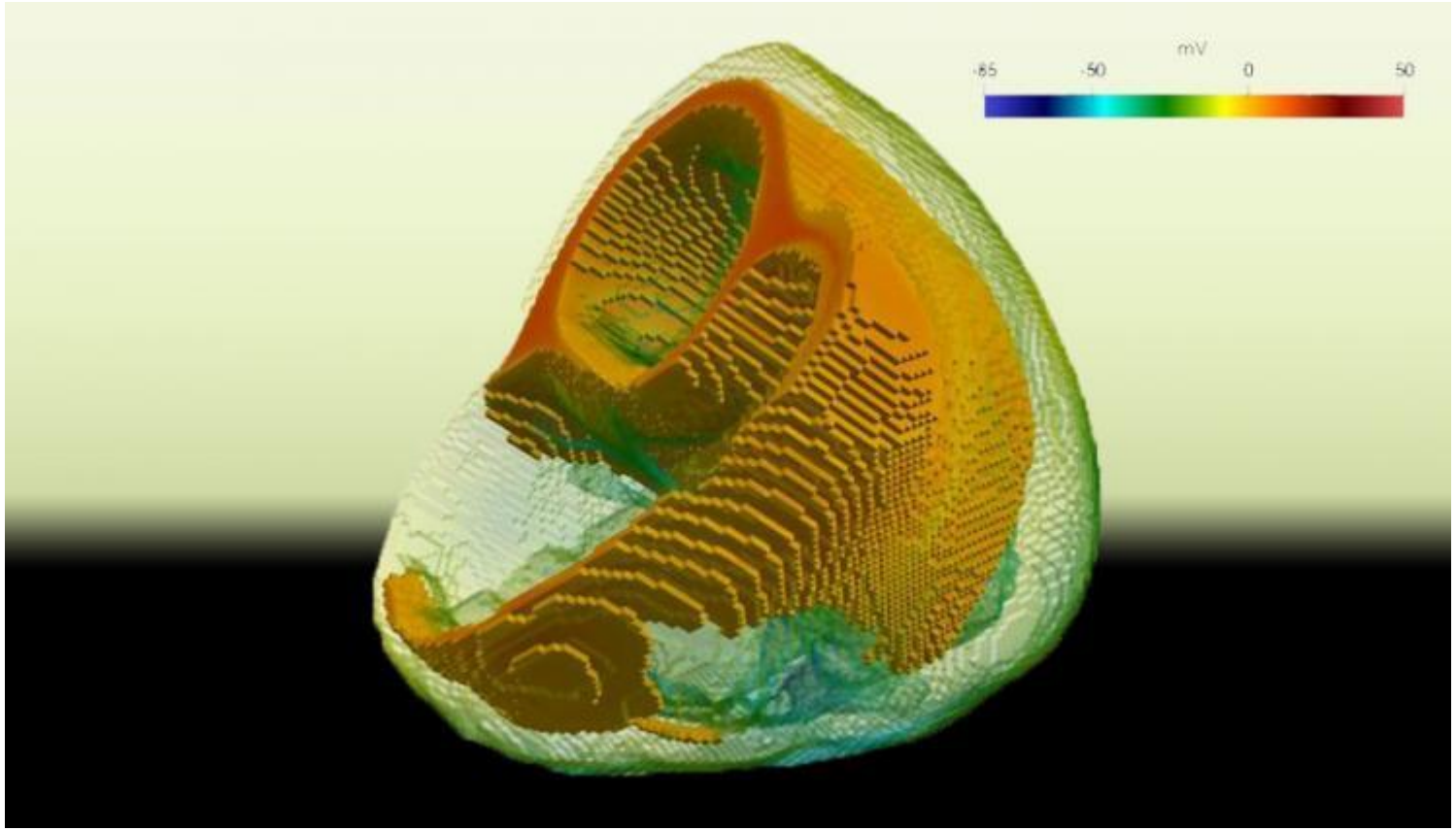
Apr 1, 2024

# IBM Blue Gene/Q

- November 2011
  - 4,096-node BG/Q (Sequoia)
  - #17 on top500 at 677.10 TF
  - #1 Graph 500 at 254 Gteps (Giga traversed edges/second)
  - #1 on Green 500 list at 2.0 Gflops/W
- June 2012
  - #1 Sequoia at Lawrence Livermore National Laboratory (#13 in 2019)
    - 96K nodes, 16.3 PF Max, 20 PF Peak, 7.8 MW
  - #3 Mira at Argonne National Laboratory (#24 in 2019)
    - 48K nodes, 8.1 PF Max, 10 PF Peak, 3.9 MW
    - Decommissioned in Dec 2019

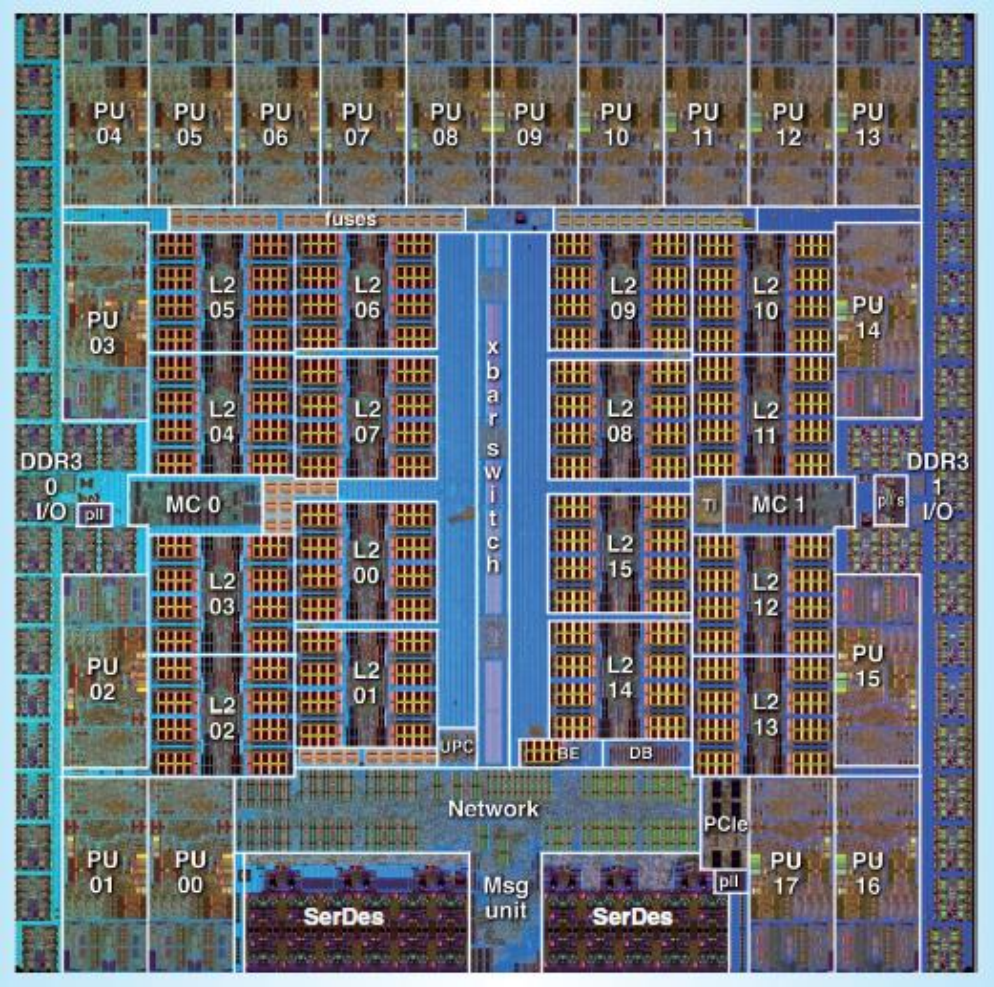
# Real Applications on Sequoia

Cosmology code HACC 14 PFLOPS




Heart simulation code Cardioid 12 PFLOPS

# BG/Q Compute Chip




The IBM Blue Gene/Q Compute Chip, IEEE MICRO, 2012

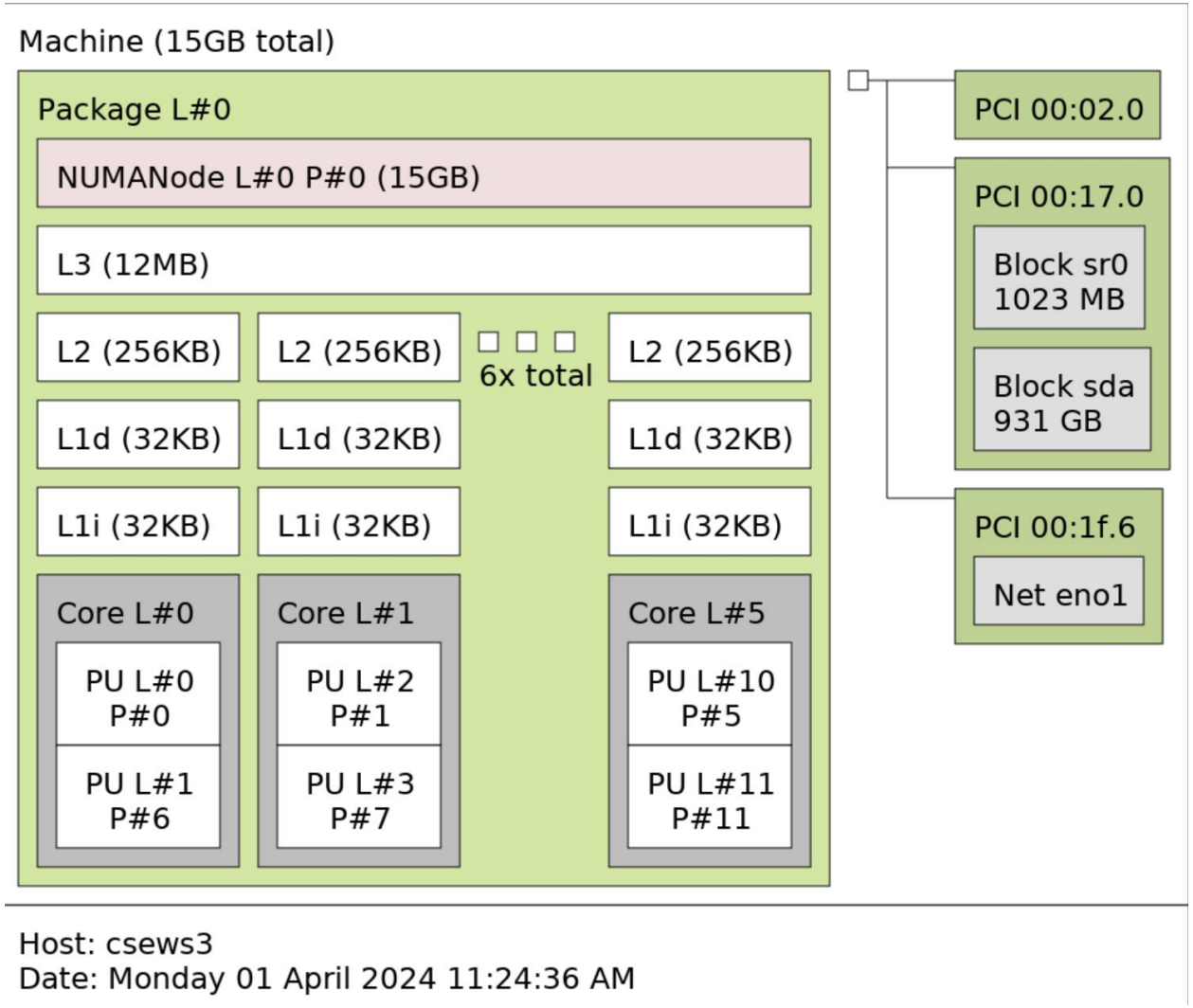
Wikipedia

**Supercomputer** [ edit ]

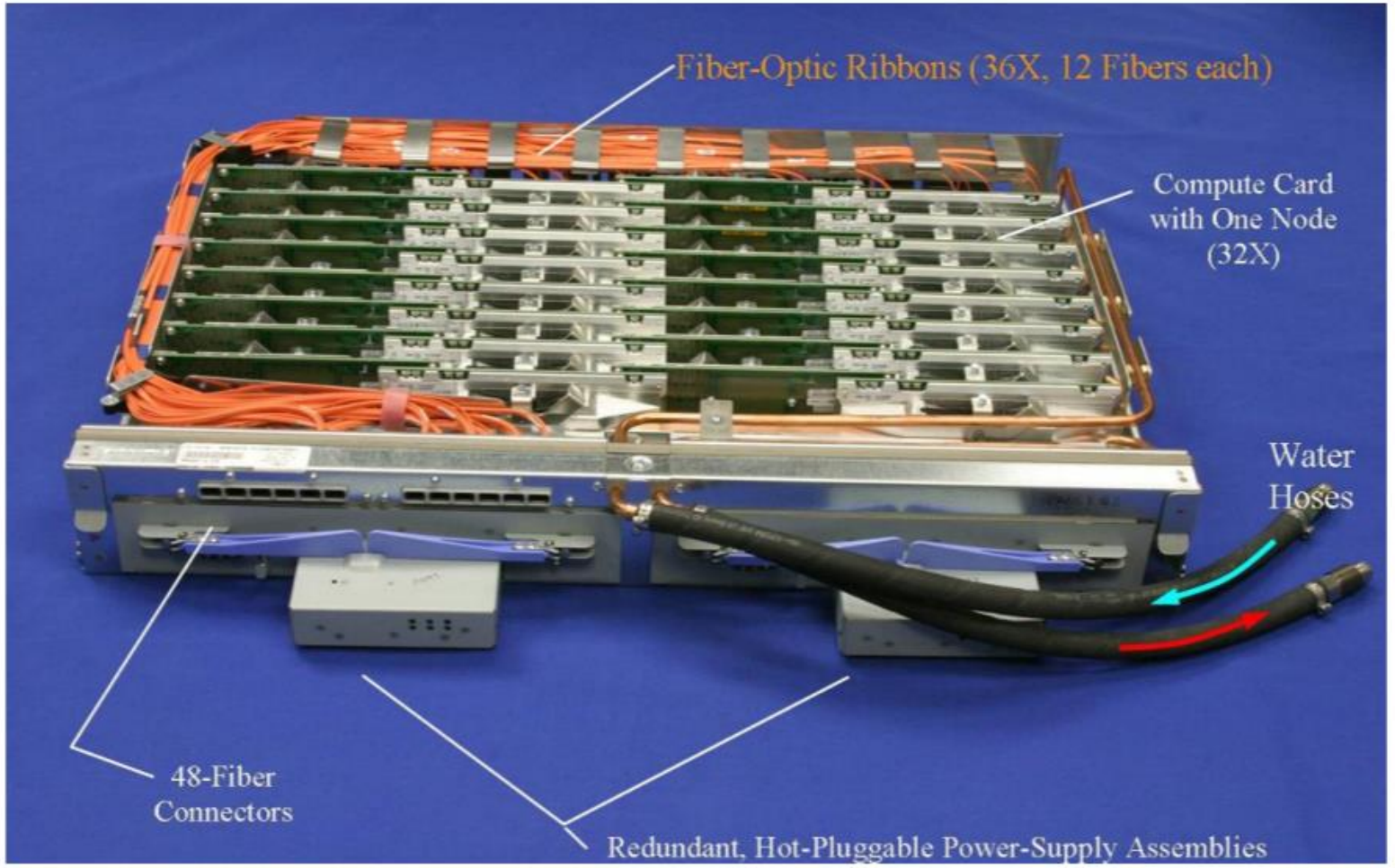
- Blue Gene/L, dual core PowerPC 440, 700 MHz, 2004
- Blue Gene/P, quad core PowerPC 450, 850 MHz, 2007
- Blue Gene/Q, 18 core PowerPC A2, 1.6 GHz, 2011

- 18.96 x 18.96 mm chip (45 nm, 1 billion transistors)
- 16 active cores, memory, cache, NoC
- PowerPC A2 Processor Core
  - 1.6 GHz
  - 64-bit Power ISA
  - In order execution
  - 4-way SMT
  - 2-way concurrent instruction issue
- Quad FPU

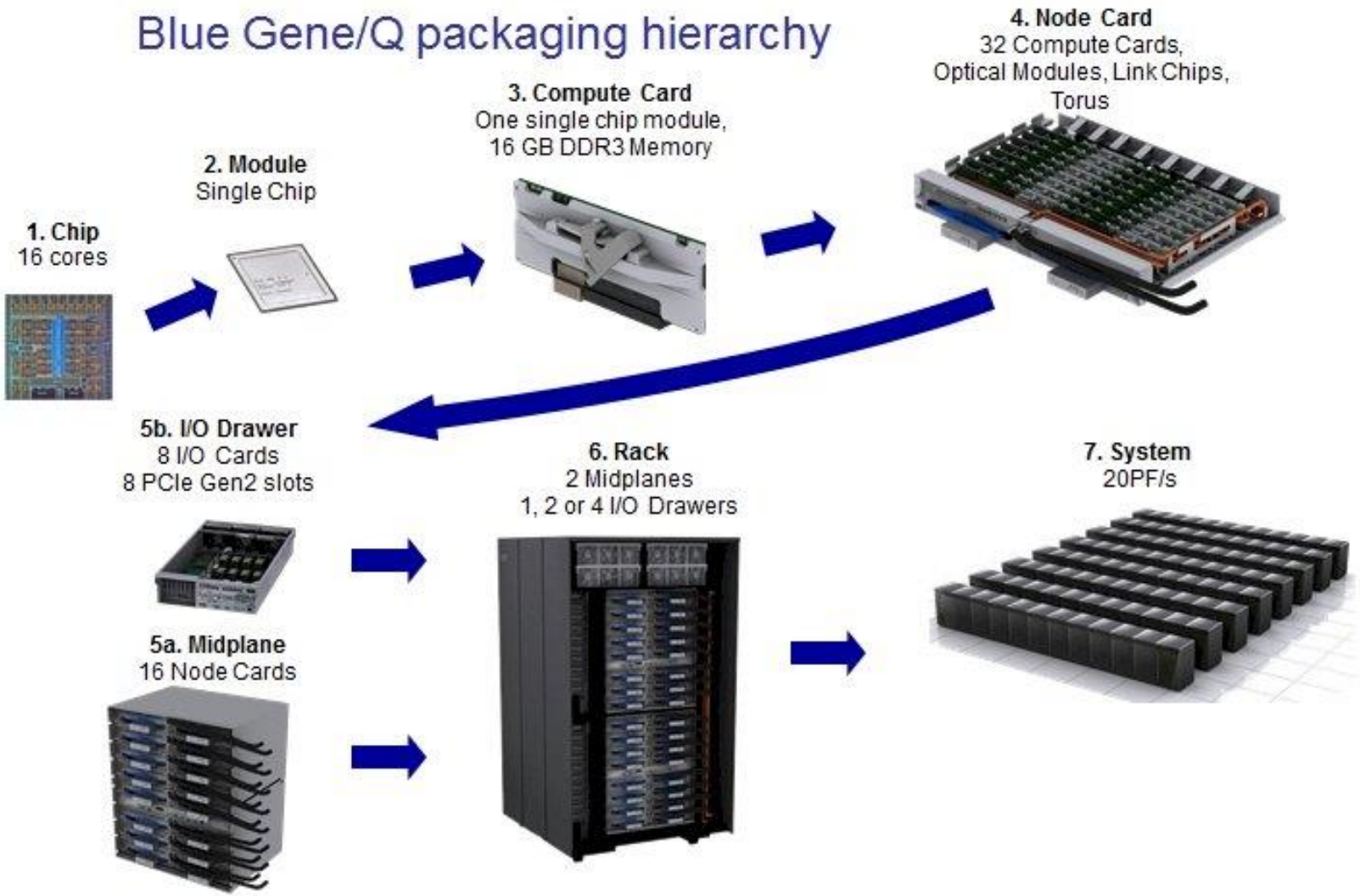
# Machine Architecture (lstopo)



# BG/Q Compute Node Board (32 nodes)



# BG/Q Hierarchy




1 Rack (1024 nodes)->
2 Midplanes (512 nodes)->
16 Node boards (32 nodes)

# Interconnects in BG

- BG/P has a 3D torus with 425 MB/s per link
- BG/Q has a 5D torus with 2 GB/s per link

Why 5D torus?

- Lower diameter, higher bisection width, lower latency than 3D torus
- High nearest neighbour bandwidth

# BG/Q Messaging Unit and Network Logic

- A, B, C, D, E dimensions (5D torus)
  - Last dimension E is of size 2 (reduces wiring)
  - Link chips on each node board connect via optics to node boards on other midplanes
  - Dimension-order routing
- On-chip per hop latency: 40 ns (20 network cycles)
  - 16x16x16x12x2 P2P latency is about 2.6 µs
  - 0.6 µs at 1 hop, 1.17 µs at 13 hops
- Injection and reception FIFOs (More than half latency incurred here)
  - Packets arriving on A- receiver are always placed on A- reception FIFO

# BG/Q Network Device

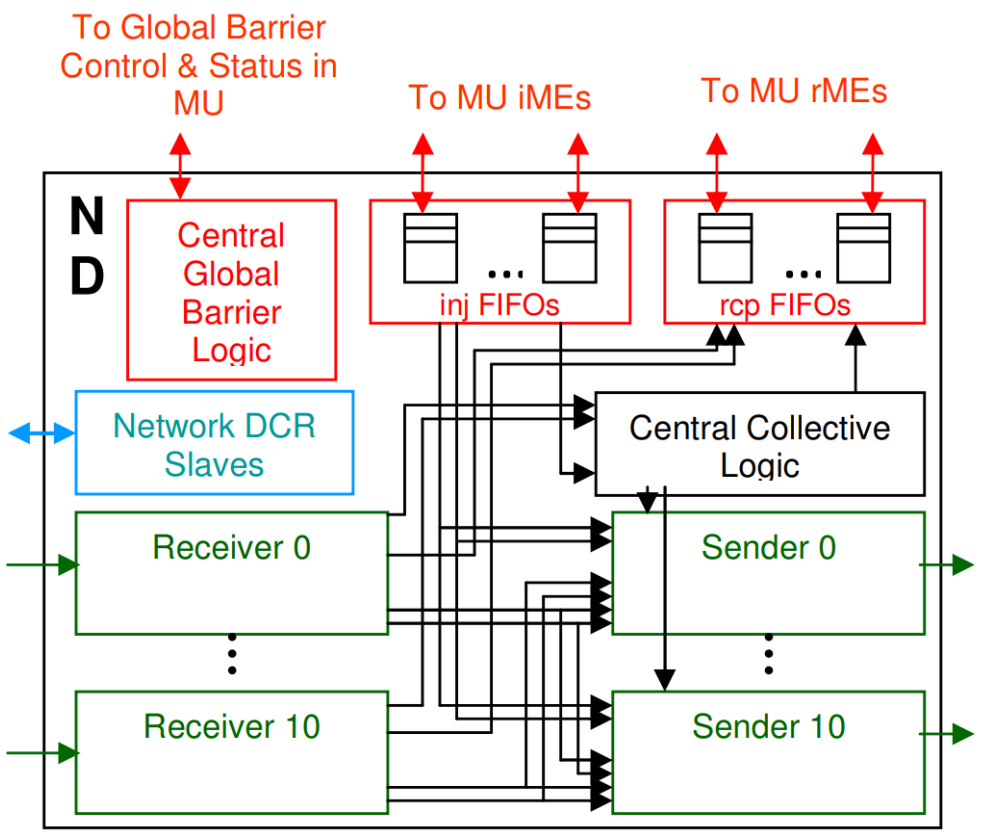
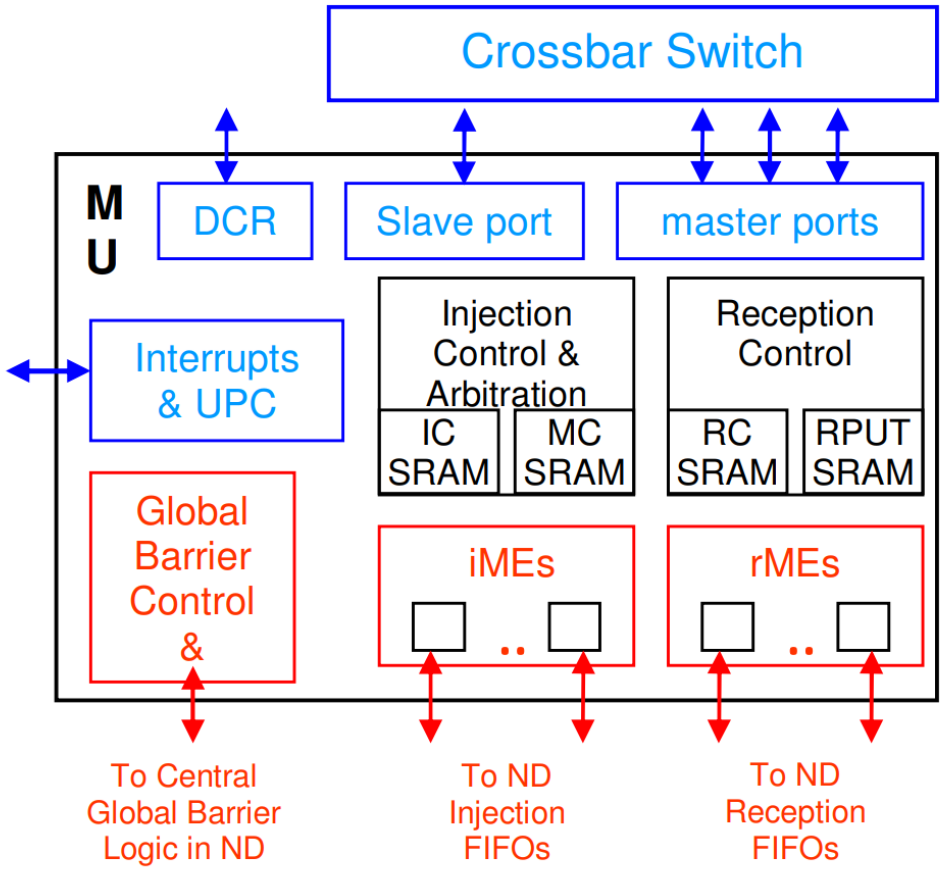



Figure 1. The BG/Q Network Device (ND) Router Logic

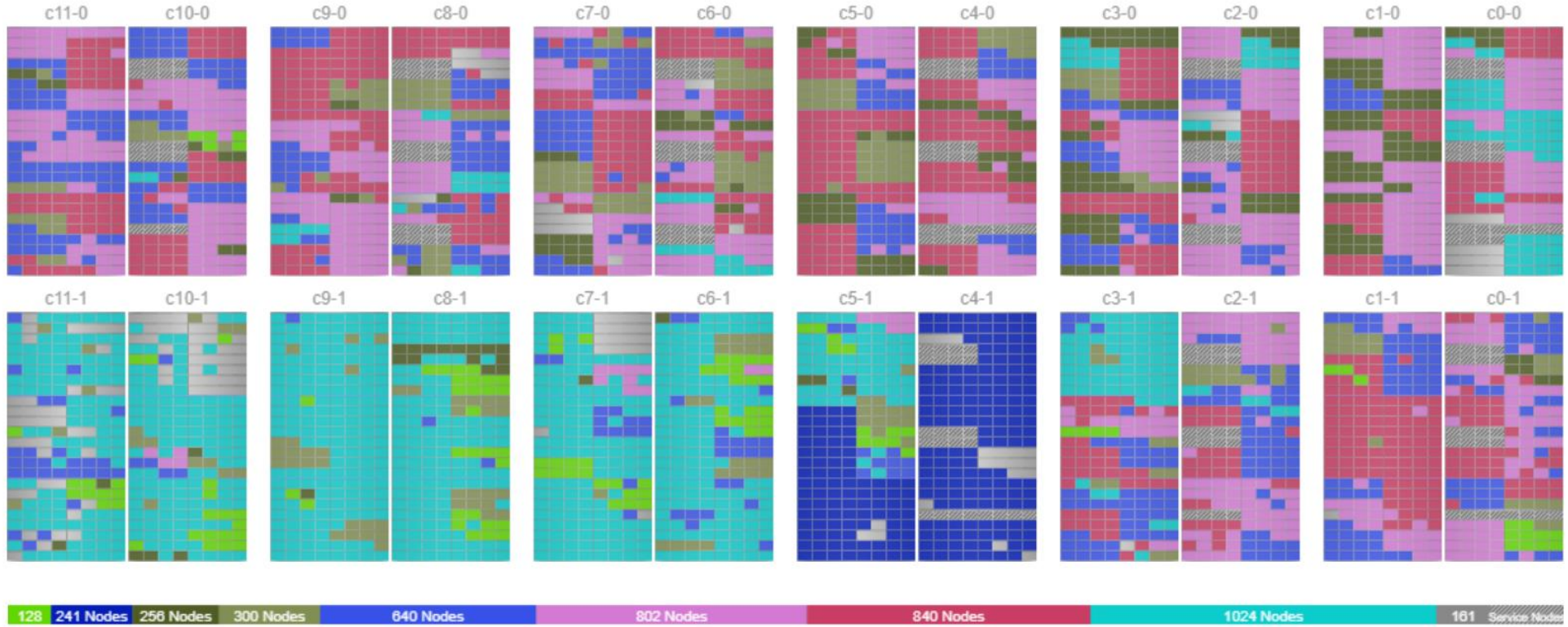



Messaging Unit (MU)

# References for BG/Q


- The IBM Blue Gene/Q Compute Chip, IEEE MICRO, 2012.
- The IBM Blue Gene/Q Interconnection Fabric, IEEE MICRO, 2012.
- The IBM Blue Gene/Q Interconnection Network and Message Unit, SC 2011.
- Looking Under the Hood of the IBM Blue Gene/Q Network, SC 2012.
- IBM System Blue Gene Solution: Blue Gene/Q Application Development, IBM Redbooks, 2013.

# Supercomputer Job Allocation




Running | Starting | Queued | Reservations

Total Running Jobs: 8

| Job Id | | Project | Nodes | Start Time | Run Time | Walltime | Queue | Mode |
|---|---|---|---|---|---|---|---|---|
| 512304 | | EstopSim_2 | 1024 | 9:13:30 AM | 00:12:13 | 16:00:00 | default | script |
| 512623 | | TurbShockWalls | 840 | 7:57:42 AM | 01:28:01 | 1d 00:00:00 | default | script |
| 498557 | | TurbShockWalls | 802 | 9:43:57 PM | 11:41:46 | 1d 00:00:00 | default | script |
| 513358 | | PSFMat_2 | 640 | 8:29:27 AM | 00:56:16 | 12:00:00 | default | script |
| 511830 | | ReconDepth | 300 | 7:50:53 AM | 01:34:50 | 06:00:00 | default | script |
| 514000 | | HighLumin | 256 | 8:50:22 AM | 00:35:21 | 06:00:00 | default | script |
| 514114 | | CVD_CityCOVID | 241 | 1:28:47 AM | 07:56:56 | 1d 12:00:00 | CVD_Research | script |
| 514178 | | FDTD_Cancer_2a | 128 | 8:00:51 AM | 01:24:52 | 03:00:00 | default | script |

status.alcf.anl.gov -> Theta (retired)

# Resources Required

- Number of nodes
- Wall-clock time
- Users are charged for node-hours

Should there be any constraints on the above requirements?

# User Jobs

- Different types of applications
- Interactive vs. batch jobs
  - Debug in interactive mode
- Exclusive vs. shared access
- Charged based on total resource usage
  - Job is killed when requested wall-clock time is over
  - Need to plan resource usage apriori

# David Lifka, The ANL/IBM SP Scheduling System, JSSPP 1995

# ANL IBM SP System Observations (Typical User Requirement)

| Required Nodes | Required Time |
|---|---|
| 1 - 8 nodes | 8 - 48 hours |
| 16 - 32 nodes | 1 - 8 hours |
| 64 - 128 nodes | 30 minutes - 3 hours |

Users were asked to use their scheduler and provide feedback

# Desirable Features of Scheduler

- Fair
- Simple
- Low average queue wait times
- High system utilization
- Provide optimum performance for all kinds of jobs
- Support different job classes (interactive vs. batch)
- Provide priority for special jobs

# FCFS with Backfilling

- FCFS scheduling
  - Poor system utilization
- Backfilling – to overcome inefficiency of FCFS
- Scan the queue of jobs for a job that does not cause the first queued job to wait for any longer than they otherwise would
- Improve system utilization
- Lower queue waiting times

# Backfilling – 128-node Example

128

| User Name | Number of Nodes | Number of Minutes | Job Status |
|---|---|---|---|
| User A | 32 | 120 | Startable |
| User B | 64 | 60 | Waiting |
| User C | 24 | 180 | Waiting |
| User D | 32 | 120 | Waiting |
| User E | 16 | 120 | Waiting |
| User F | 10 | 480 | Waiting |
| User G | 4 | 30 | Waiting |
| User H | 32 | 120 | Waiting |

96

| User Name | Number of Nodes | Number of Minutes | Job Status |
|---|---|---|---|
| User A | 32 | 120 | Running |
| User B | 64 | 60 | Startable |
| User C | 24 | 180 | Waiting |
| User D | 32 | 120 | Waiting |
| User E | 16 | 120 | Waiting |
| User F | 10 | 480 | Waiting |
| User G | 4 | 30 | Waiting |
| User H | 32 | 120 | Waiting |

| User Name | Number of Nodes | Number of Minutes | Job Status |
|---|---|---|---|
| User A | 32 | 120 | Running |
| User B | 64 | 60 | Running |
| User C | 24 | 180 | Startable |
| User D | 32 | 120 | Waiting |
| User E | 16 | 120 | Waiting |
| User F | 10 | 480 | Waiting |
| User G | 4 | 30 | Waiting |
| User H | 32 | 120 | Waiting |

32

| User Name | Number of Nodes | Number of Minutes | Job Status |
|---|---|---|---|
| User A | 32 | 120 | Running |
| User B | 64 | 60 | Running |
| User C | 24 | 180 | Running |
| User D | 32 | 120 | Blocked |
| User E | 16 | 120 | Ineligible |
| User F | 10 | 480 | Ineligible |
| User G | 4 | 30 | Startable |
| User H | 32 | 120 | Waiting |

8

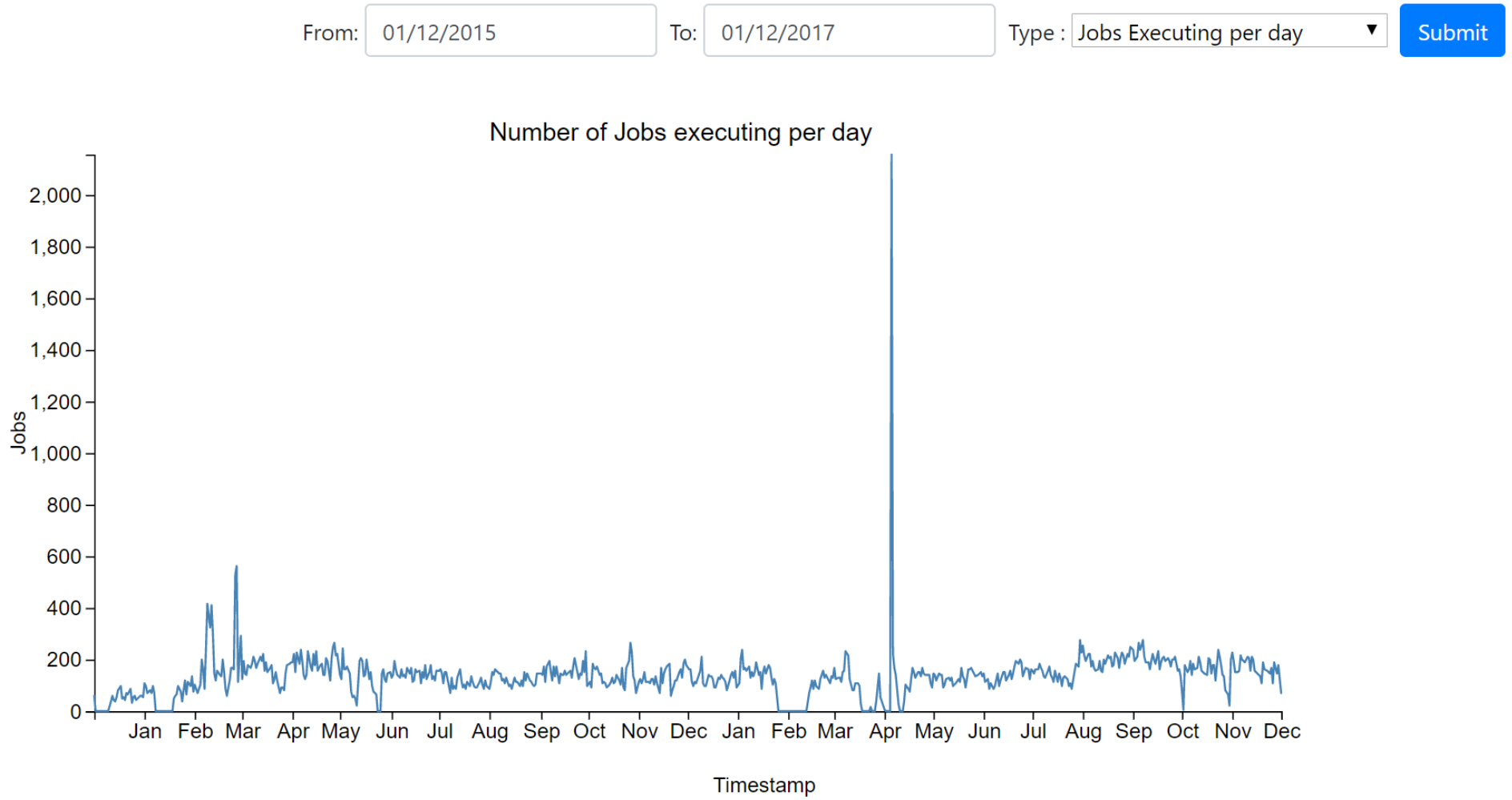
| User Name | Number of Nodes | Number of Minutes | Job Status |
|---|---|---|---|
| User A | 32 | 120 | Running |
| User B | 64 | 60 | Running |
| User C | 24 | 180 | Running |
| User D | 32 | 120 | Blocked |
| User E | 16 | 120 | Ineligible |
| User F | 8 | 480 | Startable |
| User G | 4 | 30 | Waiting |
| User H | 32 | 120 | Waiting |

0

# Scheduler Queues

- Jobs are submitted to a queue
- Different queuing policies (decided by the administrator)
- Multiple queues in some systems
  - Based on the usage
  - Queue waiting time different
  - Static vs. dynamic partitioning

# Anomaly




# Jobs executing per day on HPC2010

# An Example Scheduling Policy (144 nodes)

1. Prime time, 6 AM to 6 PM
   a. When less than 113 nodes in use:
      1-32 node jobs limited to < 4 hours.
      >32 node jobs limited to < 10 minutes.
   b. When more than 112 nodes are already in use:
      jobs limited to < 10 minutes. This maintains high availability on the last 32 nodes.
2. Interactive Extension Period, 4 AM to 6 AM and 6 PM to 10 PM
   a. When less than 113 nodes in use:
      1 - 128 node jobs limited to < 6 hr.
      > 128 node limited to < 10 min.
   b. Jobs using last 16 nodes are limited to less than 10 minutes
   c. Jobs are not started if they might not complete before the end of the shift.
3. Night time, 10 PM to 4 AM Monday through Friday and all day Saturday and Sunday.
   a. 1-144 node jobs limited to < 6 hours.
   b. Jobs are not started if they might not complete before the end of the shift.

Henderson, "Job Scheduling Under the Portable Batch System", JSSPP 1995.

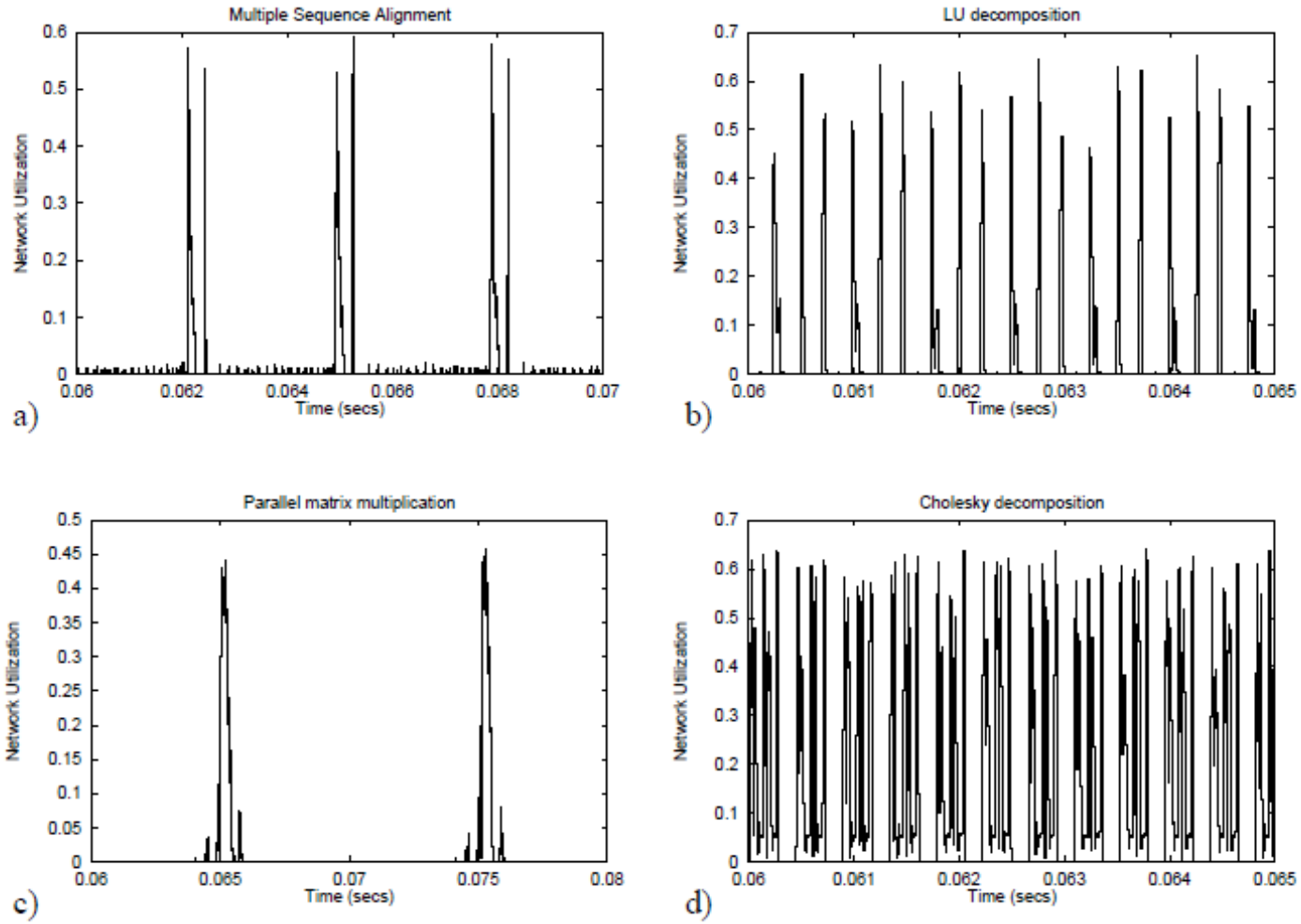
# An Example Scheduler Script

```
 foreach job {
  if (job_state == "Q") {
   if ((totpool - usepool) > 32) {
    if ((nodect<33)&&(walltime>4h))
     continue;
    if ((nodect>32)&&(walltime>10m))
     continue;
   } else {
    if ((nodect>=32)||(walltime>10m))
     continue;
   }
   if (anodes =="yes") {
    run;
    break;
   } else if (anodes == "never")
    delete JID REASON;
  }
 }
} else if ((DAY>=Mon)&&(DAY<=Fri) &&
   ((NOW>=4:00:00)&&(NOW<16:00:00)) ||
   ((NOW>=18:00:00)&&(NOW<22:00:00))){
# Interactive night
 foreach job {
  if ((job_state == "Q") &&
      (queue_type == "E")) {
```

Henderson, "Job Scheduling Under the Portable Batch System", JSSPP 1995.

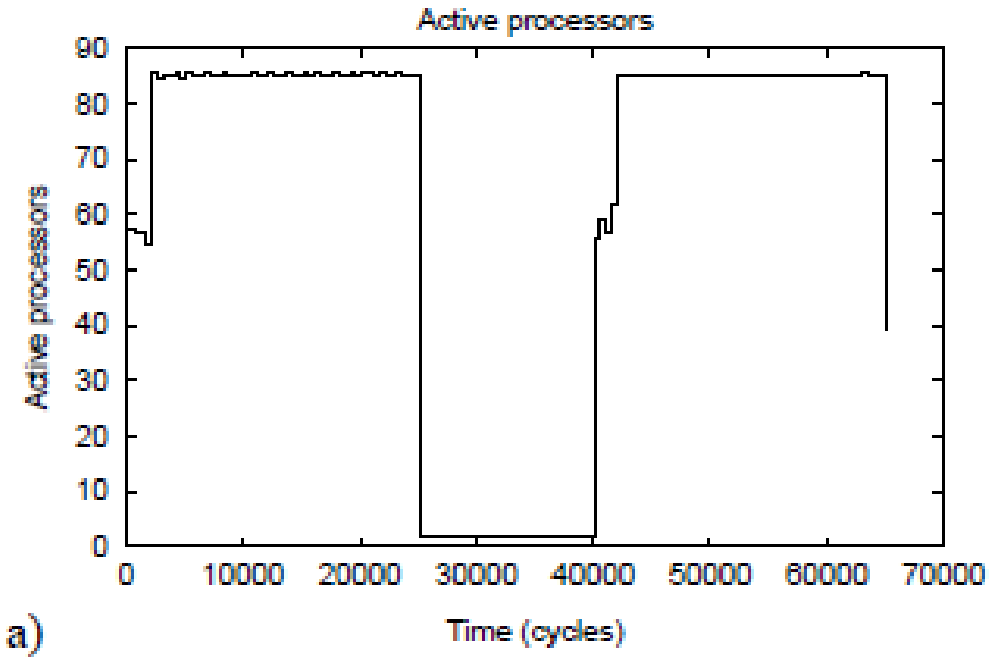
# What is missing?

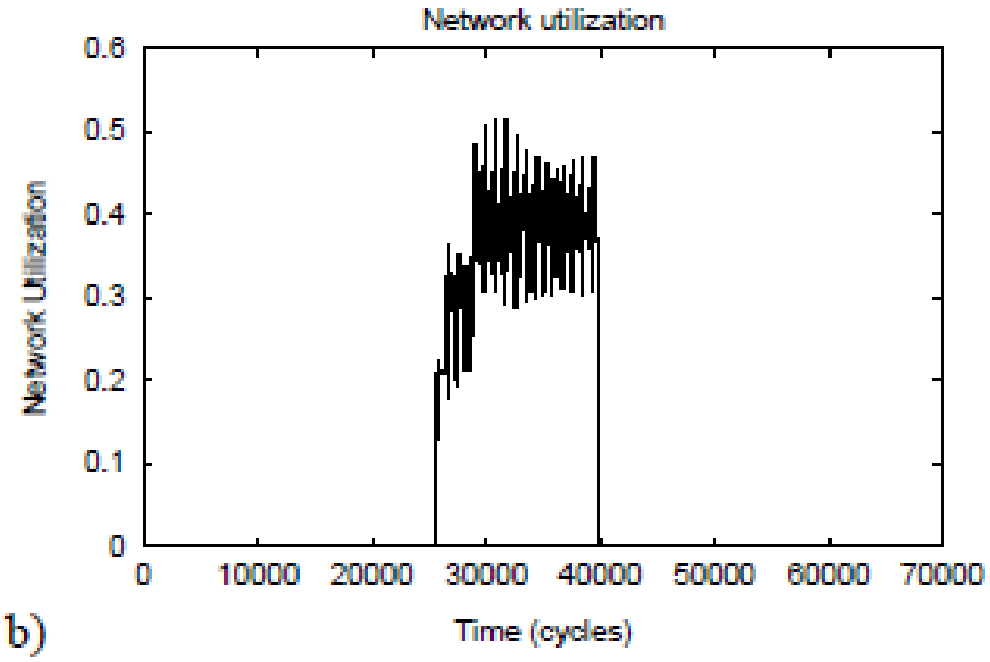
# Network Utilization in Different Applications




Petrini and Feng, Time-Sharing Parallel Jobs in the Presence of Multiple Resource Requirements, JSSPP 2000

# Network Utilization in FFT



# Batch Queueing Systems

- Schedules jobs based on queues
- Has full knowledge of queued, running jobs
- Has full knowledge of the resource usage
- Often combination of best fit, fair share, priority-based
- Designed to be generic, can be customized
- Suited to meet demands of the scheduling goals of the centre
- Typically FIFO/FCFS with backfilling

# Workload managers/Schedulers

- Portable Batch System (PBS)
- LoadLeveler
- Application Level Placement Scheduler (ALPS)
- Moab/Torque
- Simple Linux Utility for Resource Management (SLURM)

# Example Batch Scheduler

- Network Queueing System developed at NASA
- Supported multiple queues of several types
- Disable/enable each queue
- Tune the #jobs running in each queue

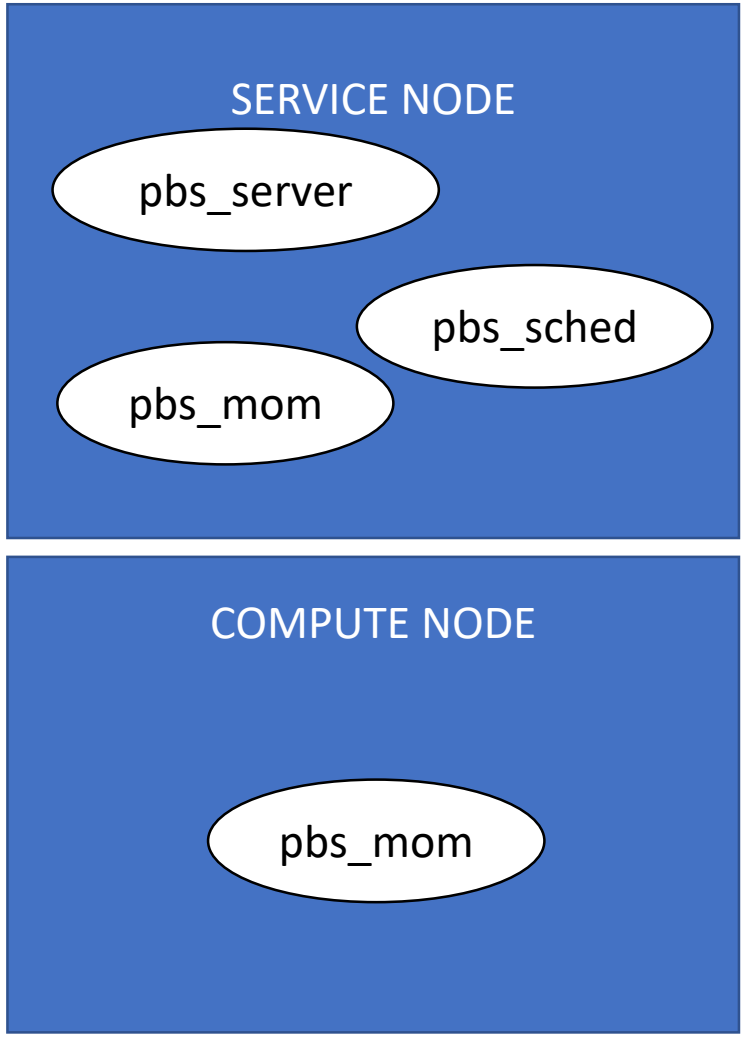
Henderson, “Job Scheduling Under the Portable Batch System”, JSSPP 1995.

# Portable Batch Scheduler

- Genesis of PBS in NASA (from NQS)
- Client commands for submission, modification, and monitoring jobs
- Daemons running on service nodes, compute nodes, and servers

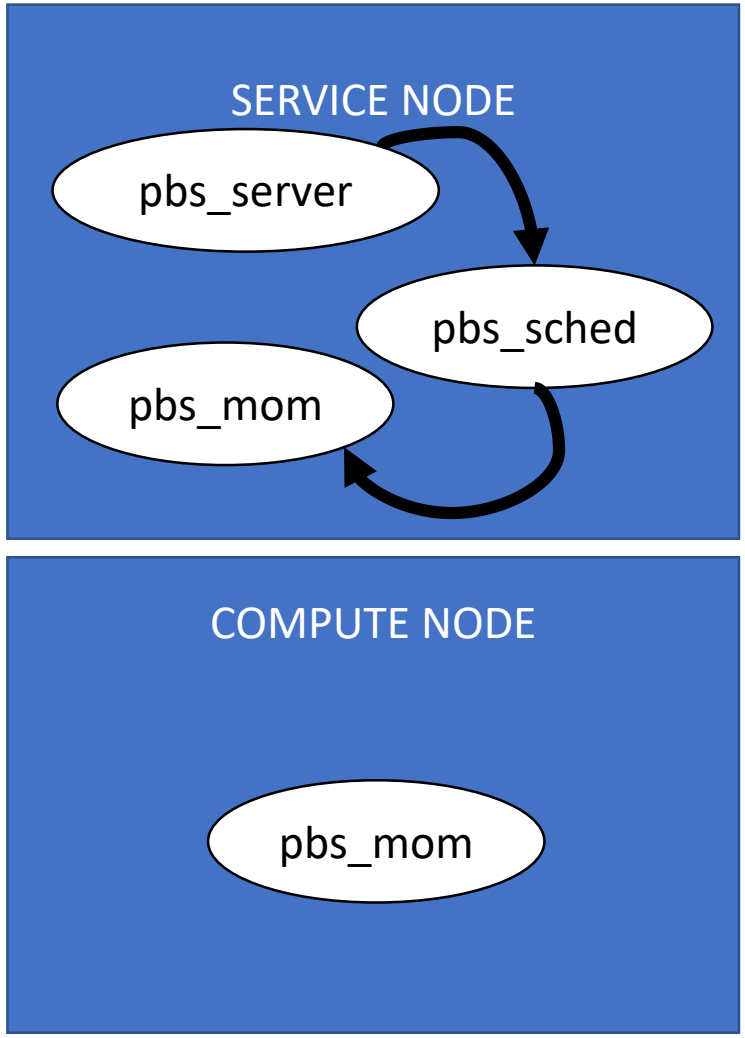
# PBS daemons

- Server (pbs_server)
  - Handles PBS commands
  - Creates batch jobs
  - Sends jobs for execution
- Scheduler (pbs_sched)
  - Schedules jobs according to system policy
- MOM (pbs_mom)
  - Manage job execution on hosts
  - Resource usage monitor
  - Record diagnostic messages
  - Notify server about job completion
  - Clean up after job completion

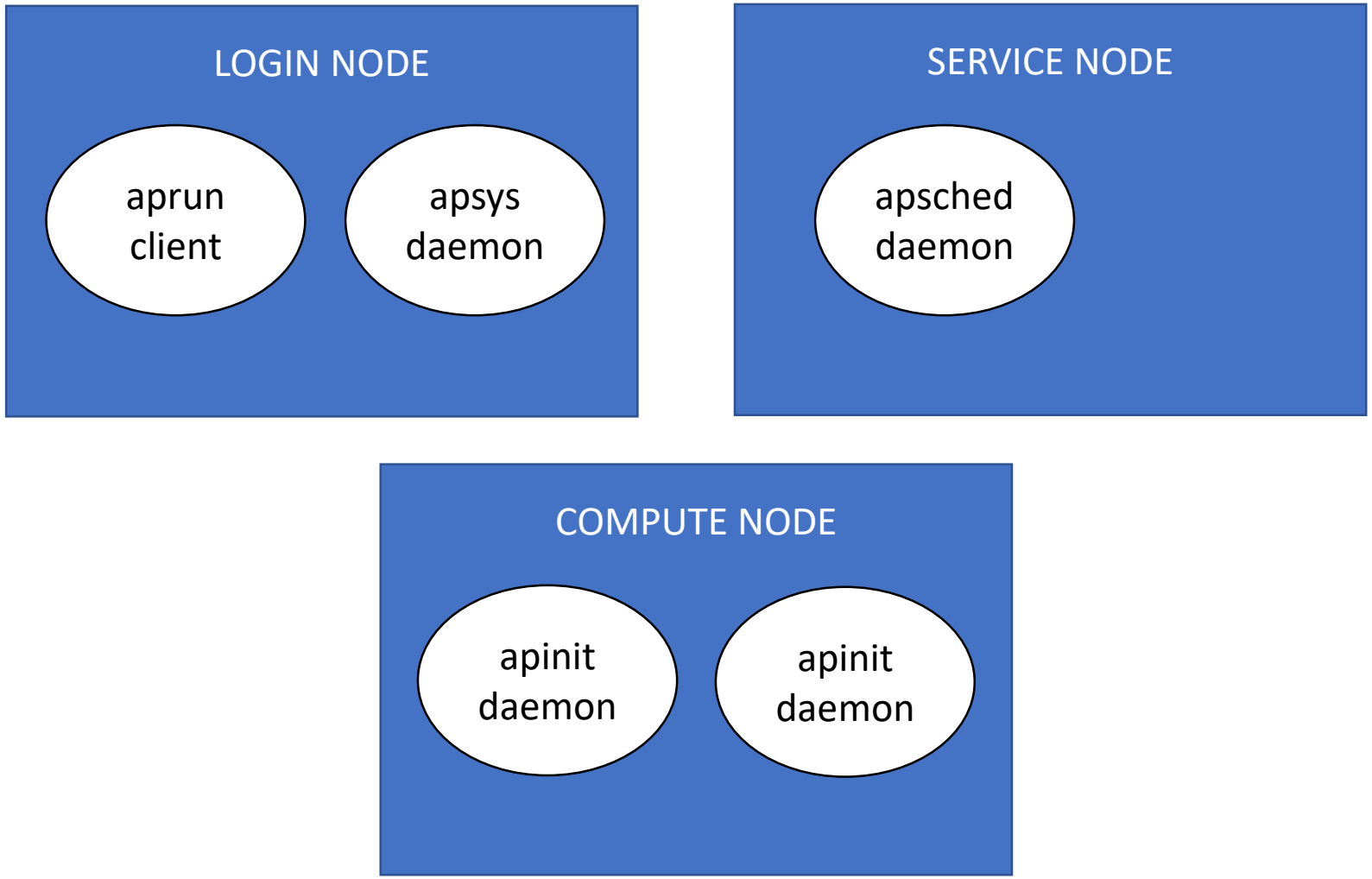


# PBS daemons

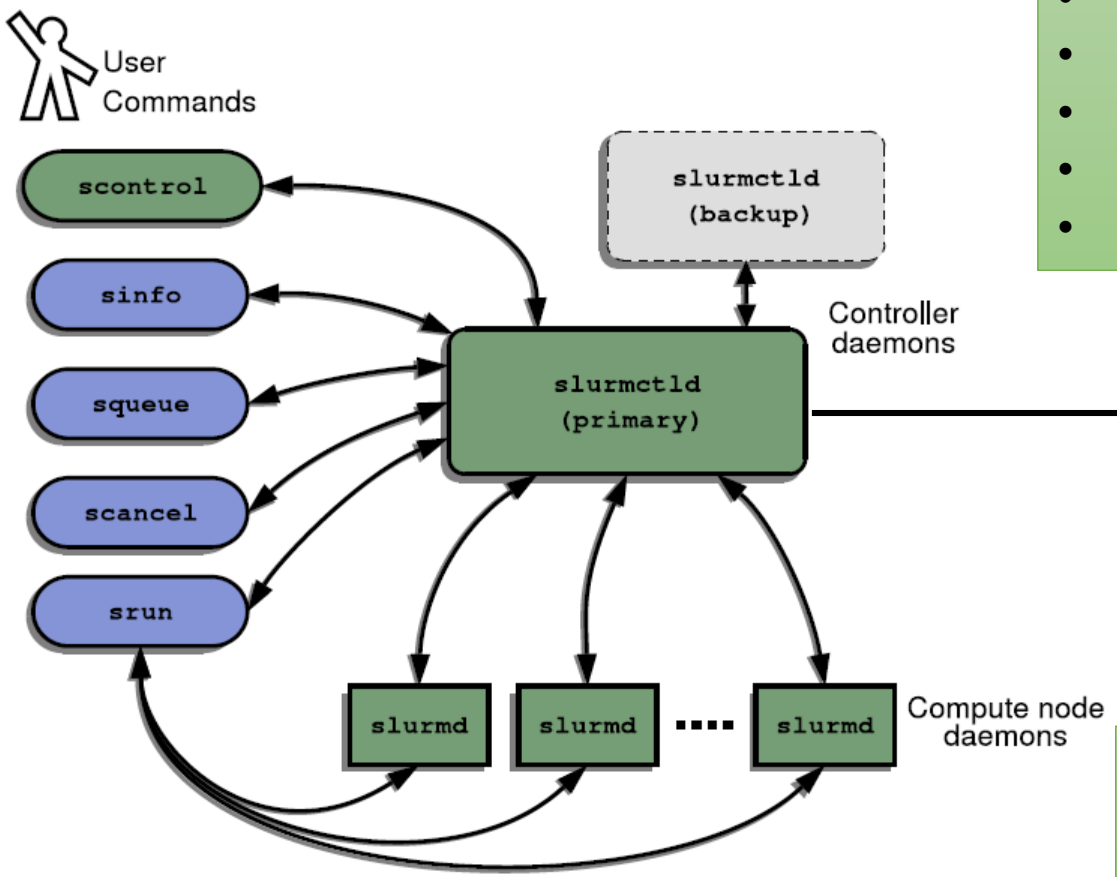
- Server contacts scheduler
  - Job is queued
  - Job terminates
- Scheduler contacts the resource monitor (MOM)
  - Queries resource usages
  - Records diagnostic messages



# Application Level Placement Scheduler (Cray)



# SLURM




qstat (PBS)

qdel (PBS)

qsub (PBS)

- Monitors states of nodes
- Accepts job requests
- Maintains queue of requests
- Schedules jobs
- Initiates job execution and cleanup
- Polls slurmd periodically
- Maintains complete state information

Slurm architecture [Jette et al.]

- Responds to controller requests
- Maintains job state
- Initiate, manage, cleanup processes
- I/O handling

# Scheduler Commands (Example)

| | |
|---|---|
| sphelp | list user commands and their functions |
| spfree | return the number of free nodes |
| sppause | pause a job waiting in the queue so that it will not be started |
| spunpause | unpause a job waiting in the queue |
| spq | show the jobs currently on the system and waiting in the queue |
| sprelease | release a node back to the free pool |
| spsubmit | submit a job to queue |
| spusage | return a current snap-shot of the resource file |
| spwait | block until a specific job has completed |
| spwhat | return what type of job could be run if submitted now |
| spwhen | tell when a specific job will start given the current queue |
| getjid | return the user job ID on a scheduled node. |

# HPC2010

- qsub –I –X
- mpiicc –o sample sample.c
- qsub sub.sh
- qstat
- http://172.31.30.3/new/code/index.html

```
#!/bin/bash
#PBS -N test
#PBS -q small
#PBS -l nodes=2:ppn=8
#PBS -l walltime=00:05:00

cd $PBS_O_WORKDIR

source /opt/software/intel/initpaths intel64
export I_MPI_FABRICS=shm:dapl

mpirun -np 8 ./sample
```