Announcements

- Project #6 is available
  - It's due Tuesday at 9:00 AM
- Office Hours Next Week
  - Tu 11-12
  - W 10-12
  - Th 11-12

My Research Interests

- Parallel Computing
  - There are limits to how fast one processor can run
  - solution: use more than one processor
- Issues in parallel computing design
  - do the processors share memory?
    - is the memory “uniform”?
    - how do processors cache memory?
  - if not how do they communicate?
    - message passing
    - what is the latency of message passing

Parallel Processing

- What happens in parallel?
- Several different processing steps
  - pipeline
  - simple example: grep foo | sort > out
  - called: *multiple instruction multiple data* (MIMD)
- The same operation
  - every processor runs the same instruction (or no-instruction)
  - called: *single instruction multiple data* (SIMD)
  - good for image processing
- The same program
  - every processor runs the same program, but not “lock step”
  - called: *single program multiple data* (SPMD)
  - most common model

Issues in effective Parallel Computation

- Load balancing
  - every processor should to have some work to do.
- Latency hiding/avoidance
  - getting data from other processors (or other disks) is slow
  - need to either:
    - hide the latency
      - processes can “pre-fetch” data before they need it
      - block and do something else while waiting
    - avoid the latency
      - use local memory (or cache)
      - use local disk (of file buffer cache)
- Limit communication bandwidth
  - use local data
  - use “near” data (i.e. neighbors)

My Research:

- Given a parallel program and a machine
- Try to answer performance related questions
  - Why is the programming running so slowly?
  - How do I fix it?
- Issues:
  - how to measure a program without changing it?
  - how do you find (and then present) the performance problem, not tons of statistics?
- Techniques:
  - dynamic data collection
  - automated search
  - analysis of process interactions

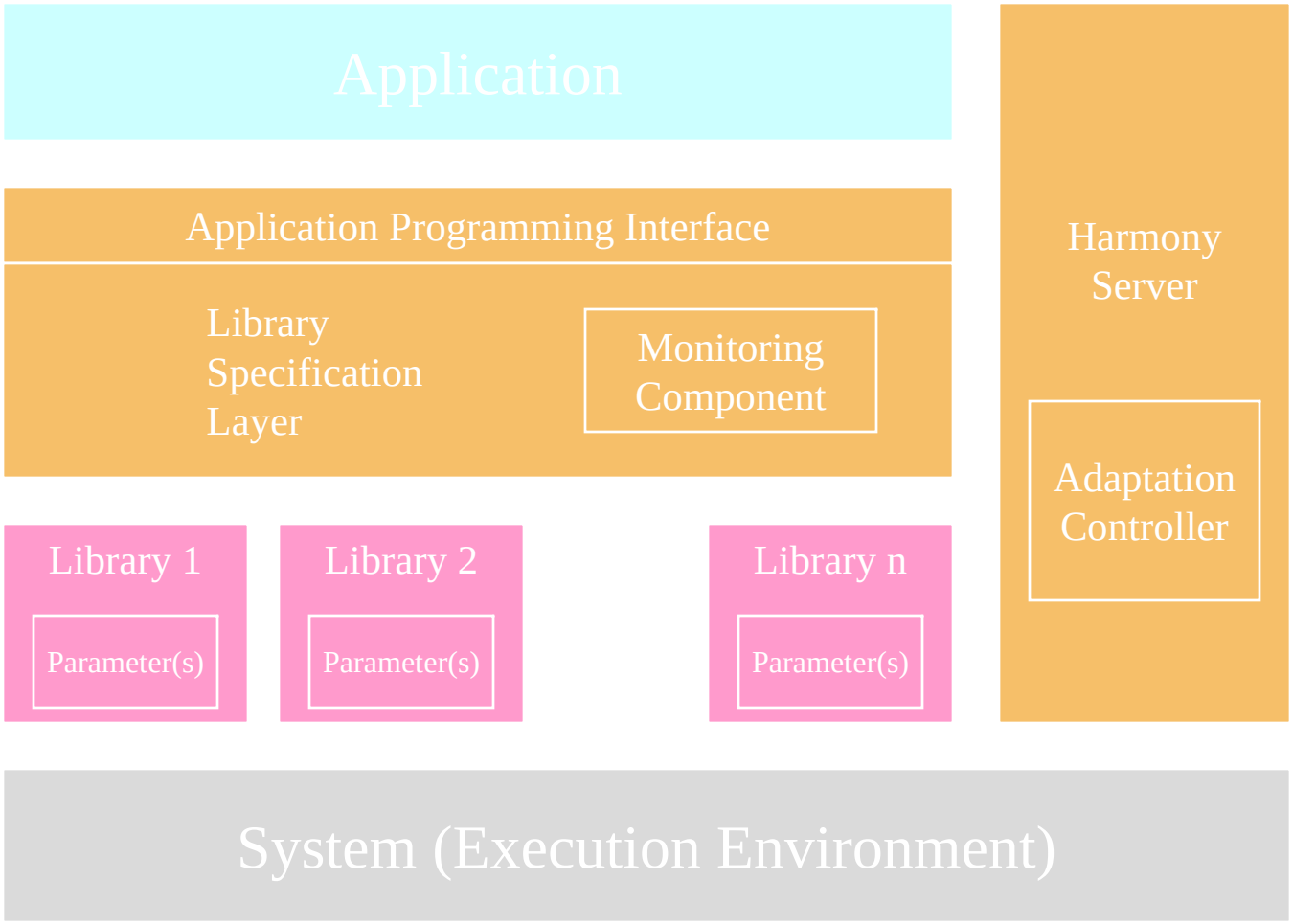
Introduction

- Software today
  - makes extensive use of libraries and re-usable components
  - Libraries used by an application may not be tuned to the application's need
- Fast software development/distribution with built-in (default) configurations
  - Applications may not run well in all environments
  - There may be no single configuration good for all environments

Active Harmony

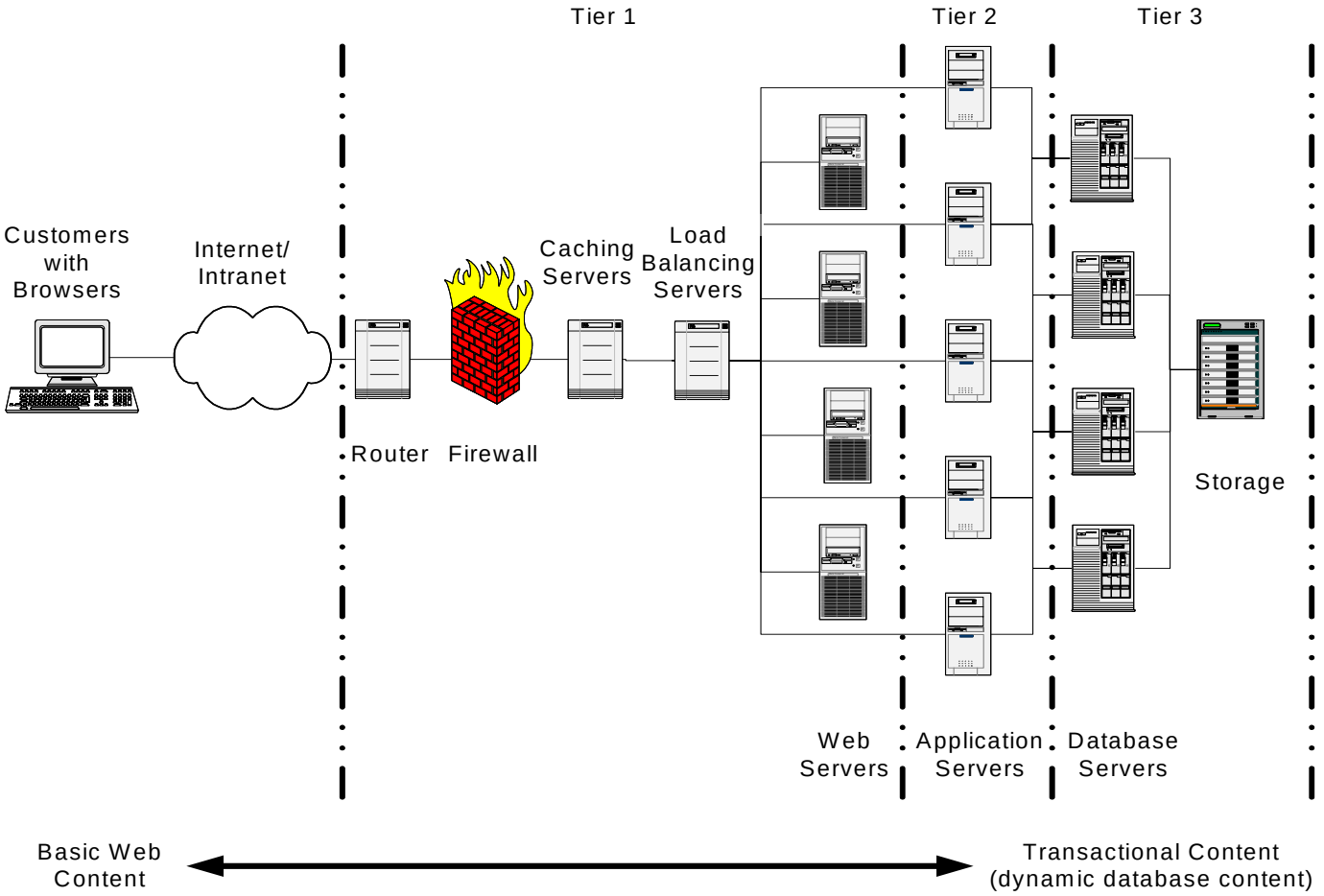
- Real-time performance optimization
- Automatic library selection (code)
  - Monitor library performance
  - Switch library if necessary
- Automatic performance tuning (parameter)
  - Monitor system performance
  - Adjust runtime parameters

System design

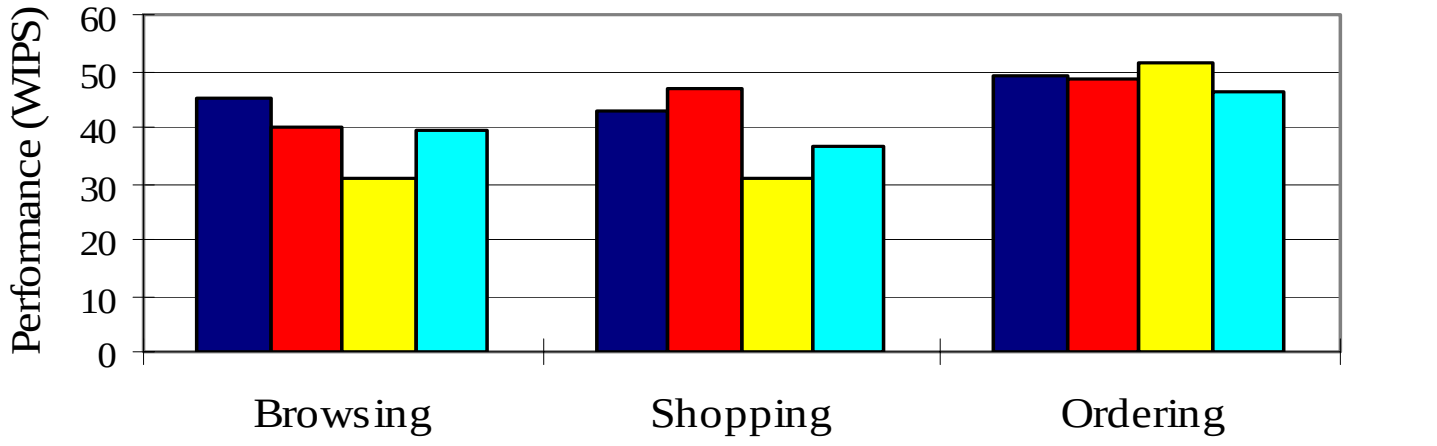


Cluster-based web service tuning

- Scalable, available, cost-effective architecture



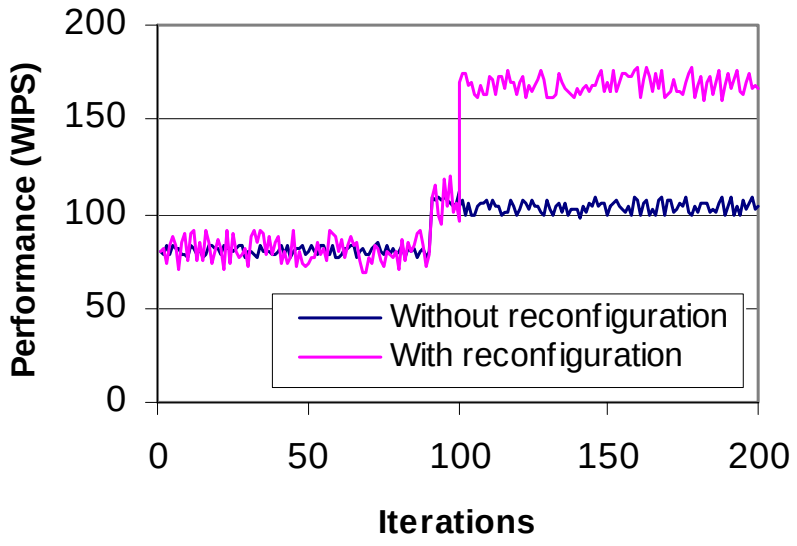
Cluster-based web service tuning

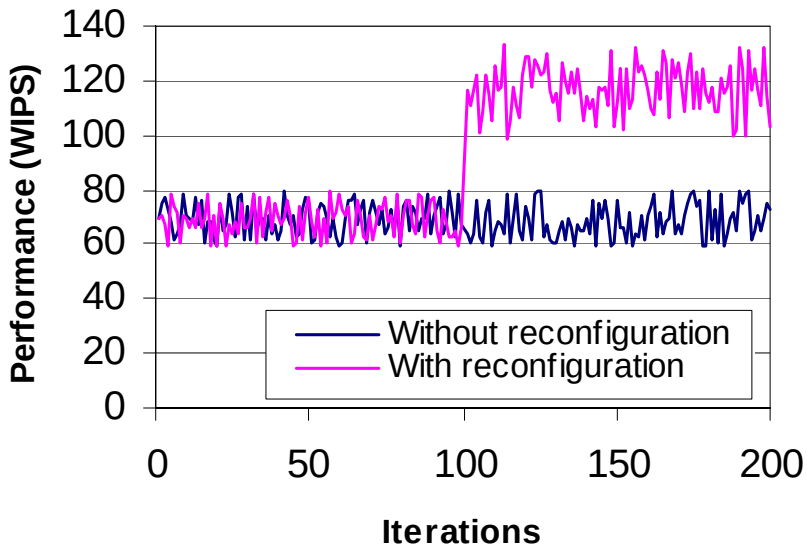


	Best configuration after 200 iterations		
	Browsing	Shopping	Ordering
Improvements compared to the default configuration	15%	16%	5%

Cluster-based web service tuning

- External tuning – reconfiguration
  - Initial: 3 proxy servers, 3 application servers



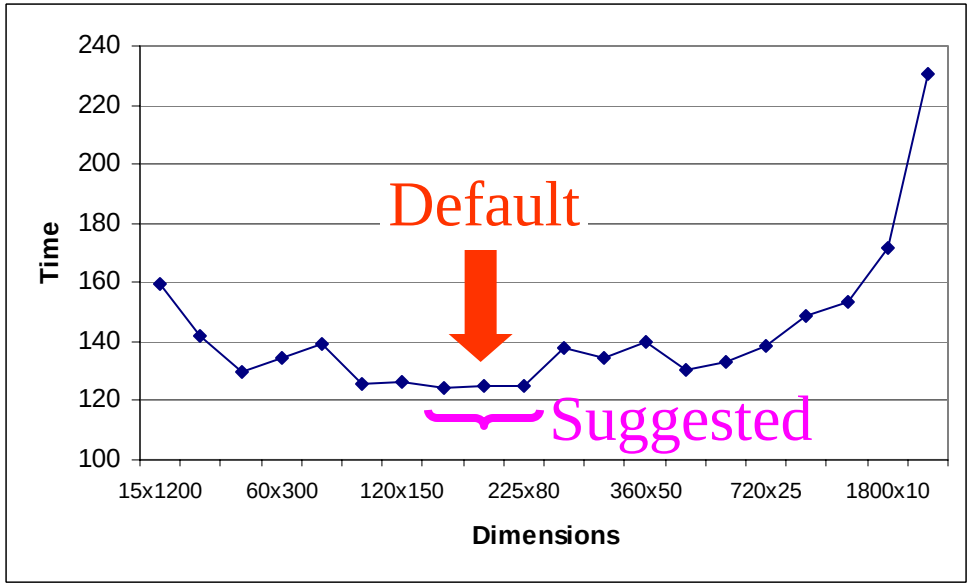


(a) One node:
proxy -> application server
(browsing -> ordering workload)

(b) One node:
application -> proxy server
(Browsing workload)

Application tuning – Parallel Ocean Program (POP)

- The ocean component of CCSM (Community Climate System Model)
- Problem size – 3600x2400
- 480 processes (32 nodes, 15 processes/node using seaborg.nersc.gov)



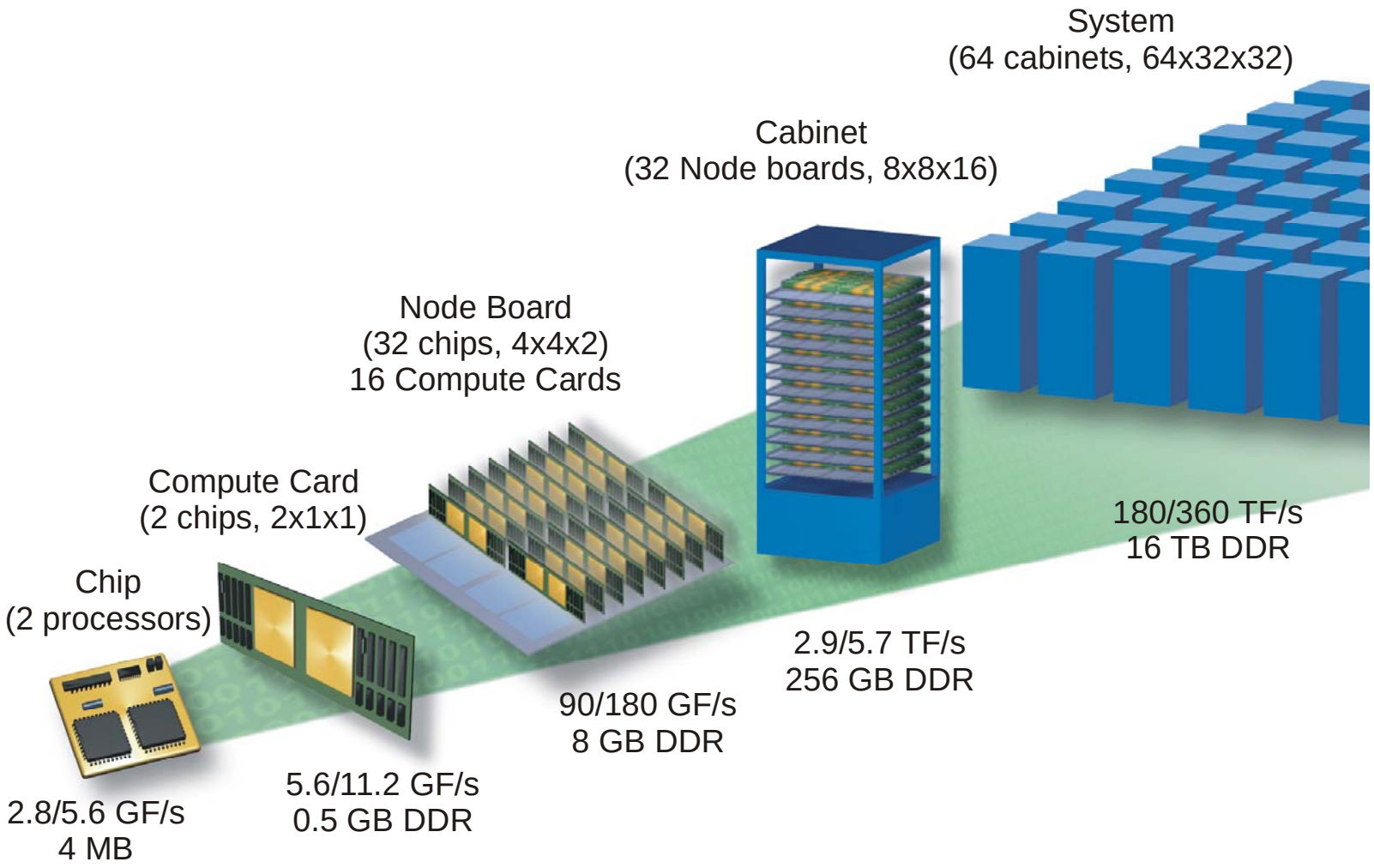
Default block dimension	Suggested block dimension
180x100	150x120 180x100 225x80

Application tuning – Parallel Ocean Program (POP)

- Performance tuning by parameters adjustment
- Numerous parameters
  - About 20 of them are performance related
  - Each with 2-4 possible values
- Without prior knowledge
- 32 processes (8 nodes, 4 processes/node running on hockney.nersc.gov)
- 12.1% improvement in execution time after trying 12 configurations
- 16.7% improvement after tuning (27 iterations)

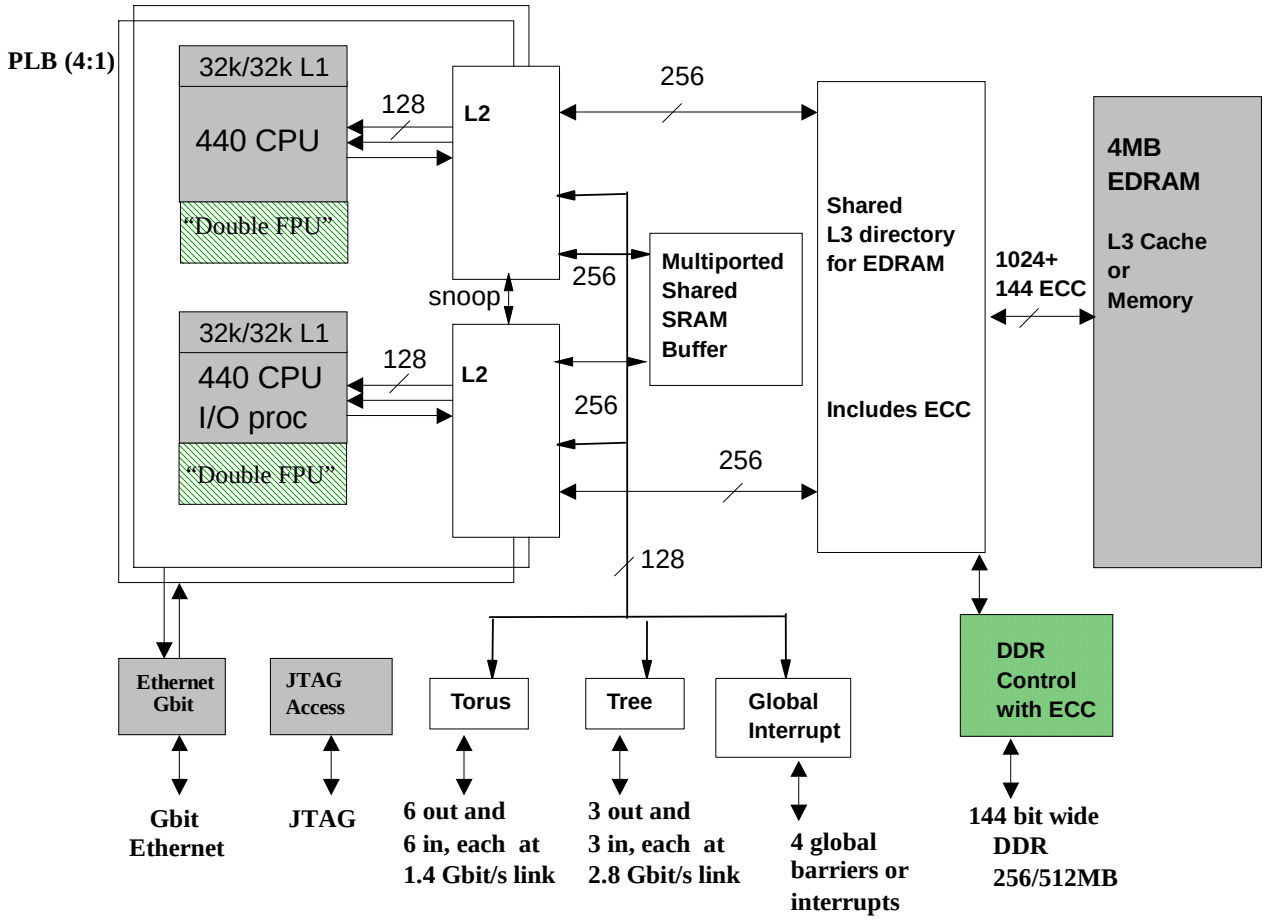
Parameter	Default	After tuning
num_iotasks	1	4
hmix_momentum_choice	anis	del2
hmix_tracer_choice	gent	del2
kappa_choice	constant	variable
slope_control_choice	notanh	clip
hmix_alignment_choice	east	grid
state_choice	jmcd	linear
state_range_opt	ignore	enforce
ws_interp_type	nearest	4point
shf_interp_type	nearest	4point
sfwf_interp_type	nearest	4point
ap_interp_type	nearest	4point

Blue Gene/L

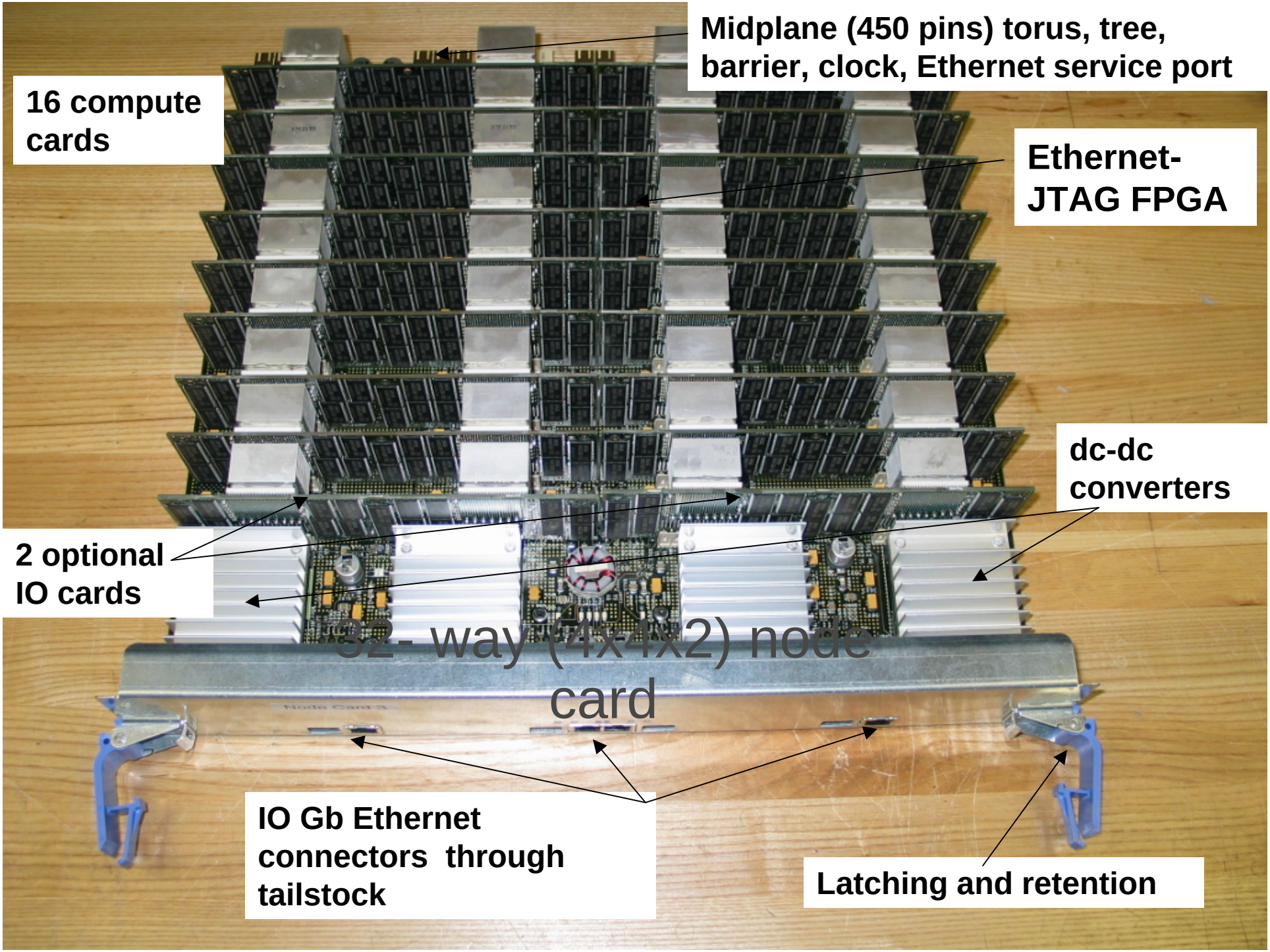


Courtesy IBM

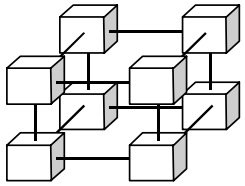
BlueGene/L Nodes



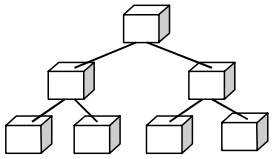
Courtesy IBM


Midplane (450 pins) torus, tree, barrier, clock, Ethernet service port
16 compute cards
Ethernet-JTAG FPGA
dc-dc converters
2 optional IO cards
32- way (4x4x2) node card
IO Gb Ethernet connectors through tailstock
Latching and retention

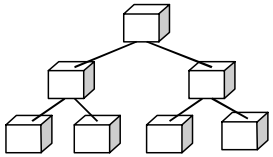
BlueGene/L Networks



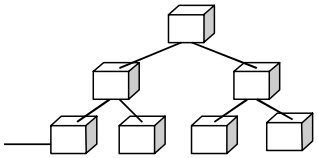
3 Dimensional Torus
- Point-to-point



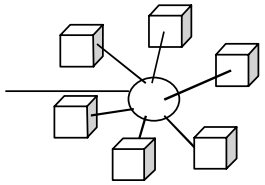
Global Tree
- Global Operations



Global Barriers and Interrupts
- Low Latency Barriers and Interrupts



Gbit Ethernet
- File I/O and Host Interface



Control Network
- Boot, Monitoring and Diagnostics

Courtesy IBM