

# Scaling (Waaay) Up

*or*

*who needs a (linux) machine with 65536 dual CPUs,  
what it should look like,  
and how to tame the beast.*

Oleg Goldshmidt

olegg@il.ibm.com

IBM Haifa Research Laboratories



# What This Talk Is **Not** About

- scaling SMP kernel to 8, 16, 32, 64 processors
  - SGI, IBM, and others who have  $N$ -CPU machines for  $N > 4$
- scaling in terms of load for **enterprise**
  - HP, Sun, Dell, IBM and every other major player in the market
  - every John, Dick, and Harry who has successfully conned a VC into investing a few bucks
- linux, really...



# What This Talk **Is** About

## Massively parallel computations

- who needs them
- how they are performed
- how to design parallel computers
- state of the art — Top 500 and Top 5
- the next frontier — IBM's BlueGene/L
- how to manage a **really** large system
- how to schedule, run and manage parallel jobs



# Some Preliminary Questions...



# Some Preliminary Questions...

Who reads  ?



# Some Preliminary Questions...

Who reads  ?  
Regularly?



# Some Preliminary Questions...

Who reads  ?

Regularly?

For the last few years?



# Some Preliminary Questions...

Who reads  ?

Regularly?

For the last few years?

What is the obligatory /. comment  
whenever a cool piece of  
technology is discussed?



# And A Very Significant Answer...



**The Fine Print:** The following comments are owned by whoever posted them. We are not responsible for them in any way.

**Re:Imagine a Beowulf cluster of these babies! (Score:0)**  
by Anonymous Coward on Monday April 21, @03:44PM  
([#5775329](#))



# The Main Argument **For** Clusters

## Cheap off-the-shelf hardware

- provides low price/performance ratio compared to “big iron”
- easy to run Linux on, with familiar software readily available
  - compilers and standard libraries
  - numerical libraries
  - specialized parallel computation software: MPI, PVM, etc



# The Main Argument **Against** Clusters

Cheap off-the-shelf hardware

- means low MTBF



# The Main Argument **Against** Clusters

Cheap off-the-shelf hardware

- means low MTBF

Back of the envelope calculation:

- $U$  — uptime (a month is generous for a busy machine)
- $N$  — number of computers in a cluster ( $\approx 1000$  in a large cluster)

Assume independent failures with time-independent probability:

$$MTBF = \frac{U}{N} < 45 \text{ min} \frac{(U/30 \text{ days})}{(N/1000)}$$



# Scaling **Waaay** Up

## Part I

# Parallel Computations



# Who Needs Parallel Computations?

Just about everybody in science and engineering: if you want to

- forecast the weather for a continent (or just for Israel)
- study galaxy formation
- price financial derivatives
- launch rockets to intercept incoming Scuds
- bring a space shuttle back from orbit
- be able to blow up a specific chunk of the planet

or do any number of other interesting and important things, you will want to parallelize computations. Why?



# Typical Mathematical Problem: PDE

Many, if not most, scientific and engineering problems, once formulated mathematically, boil down to solving ordinary or partial differential equations, e. g. (with  $f = f(t, x, y, z)$ )

$$\frac{\partial f}{\partial t} = F\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}, \frac{\partial^2 f}{\partial x^2}, \dots, f, x, y, z, t\right).$$

Examples:

- Schrödinger's equation (quantum physics, chemistry)
- Maxwell's equations (electrodynamics)
- Navier-Stokes equation (fluid dynamics)
- diffusion equation (e. g. thermodynamics)
- Black-Scholes equation (option pricing)



# PDE: Numerical Solution (I)

The **most primitive** finite difference scheme:

- create discrete 3D grid in  $(x, y, z)$  with (equal) steps  $\Delta x$ ,  $\Delta y$ ,  $\Delta z$
- make (equal) discrete steps in time  $\Delta t$

Now recall that, by definition,

$$\frac{\partial f}{\partial t} = \lim_{\Delta t \rightarrow 0} \frac{f(t + \Delta t) - f(t)}{\Delta t},$$

etc., and substitute the finite differences for the partial derivatives. With small enough “deltas” the approximate solution will converge to the true one.



# PDE: Numerical Solution (II)

Express derivatives in our PDE as finite differences:

$$\frac{\partial f}{\partial t} = \frac{f(t + \Delta t) - f(t)}{\Delta t},$$

$$\frac{\partial f}{\partial x} = \frac{1}{2} \left[ \frac{f(x + \Delta x) - f(x)}{\Delta x} + \frac{f(x) - f(x - \Delta x)}{\Delta x} \right]$$

$$= \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x},$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x} \left[ \frac{\partial f}{\partial x} \right]$$

$$= \frac{1}{\Delta x} \left[ \frac{f(x + \Delta x) - f(x)}{\Delta x} - \frac{f(x) - f(x - \Delta x)}{\Delta x} \right]$$

$$= \frac{f(x + \Delta x) - 2f(x) + f(x - \Delta x)}{(\Delta x)^2}$$

# PDE: Numerical Solution (III)

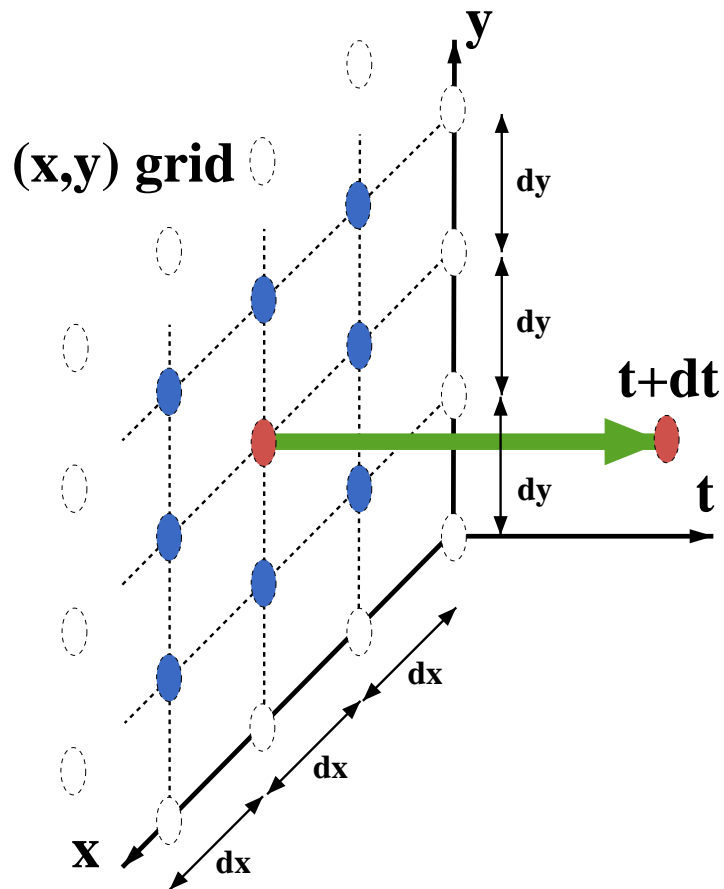
Denote  $f = f(t, x, y, z)$ ,  $f_+ = f(t, x + \Delta x, y, z)$ ,  
 $f_- = f(t, x - \Delta x, y, z)$ , etc., and rewrite the PDE as

$$\frac{f(t + \Delta t) - f(t)}{\Delta t} = \hat{F}[f, f_+, f_-, \dots, t, x, y, z, \Delta x, \Delta y, \Delta z], \text{ or}$$
$$f(t + \Delta t) = f(t) + \hat{F}[f, f_+, f_-, \dots, t, x, y, z, \Delta x, \Delta y, \Delta z] \cdot \Delta t$$

The single time step yields the solution  $f$  at point  $(x, y, z)$  at time  $t + \Delta t$  from the values at several **adjacent** points on the 3D grid at time  $t$ . Starting from the initial conditions at  $t = 0$  we can trace the evolution in time.

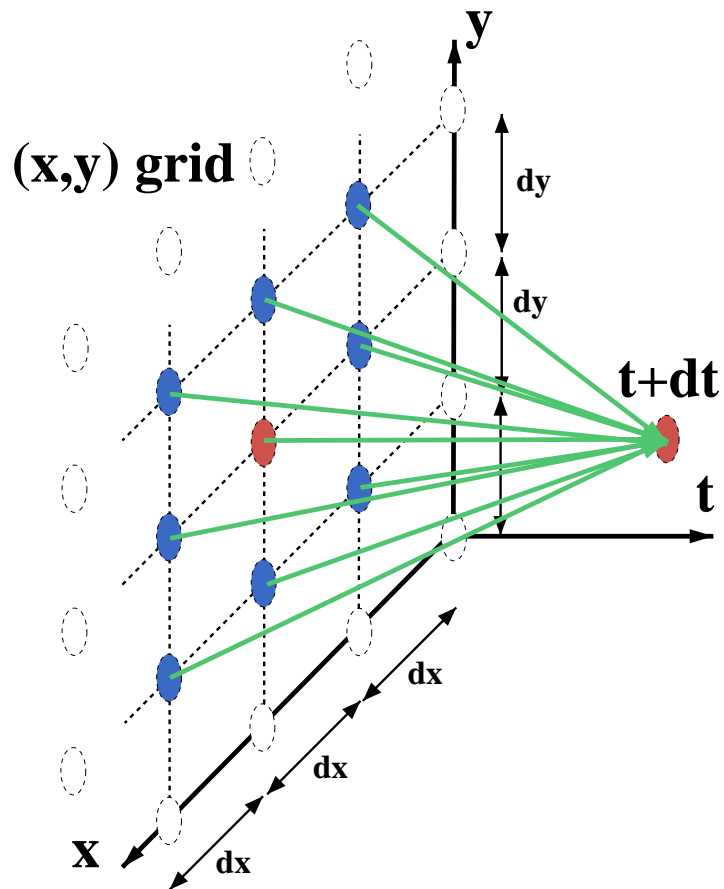


# PDE: Numerical Solution (IV)



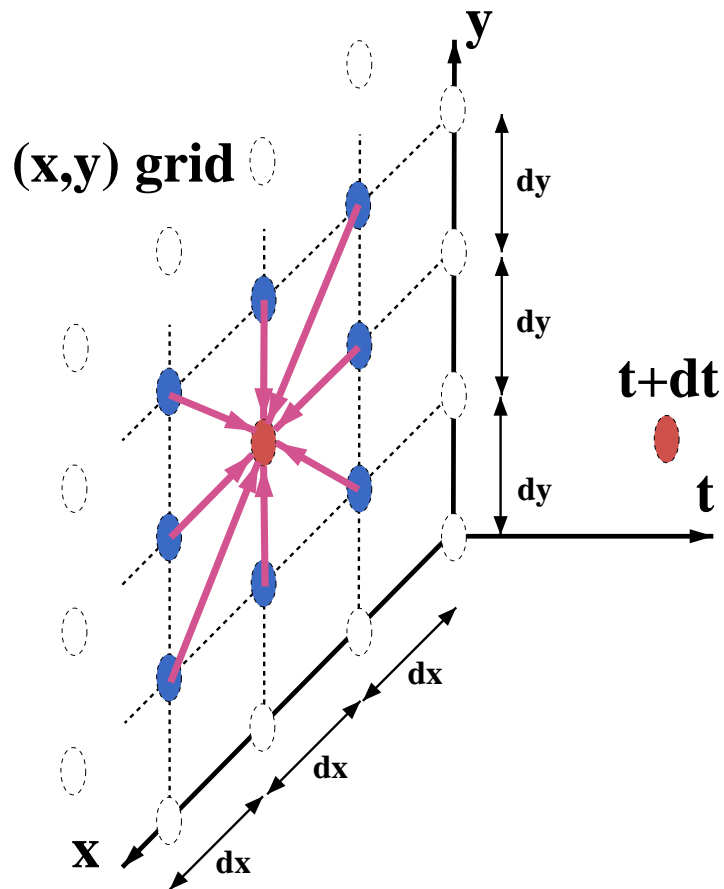
- make successive steps in time for all  $(x, y)$  on the spatial grid

# PDE: Numerical Solution (IV)



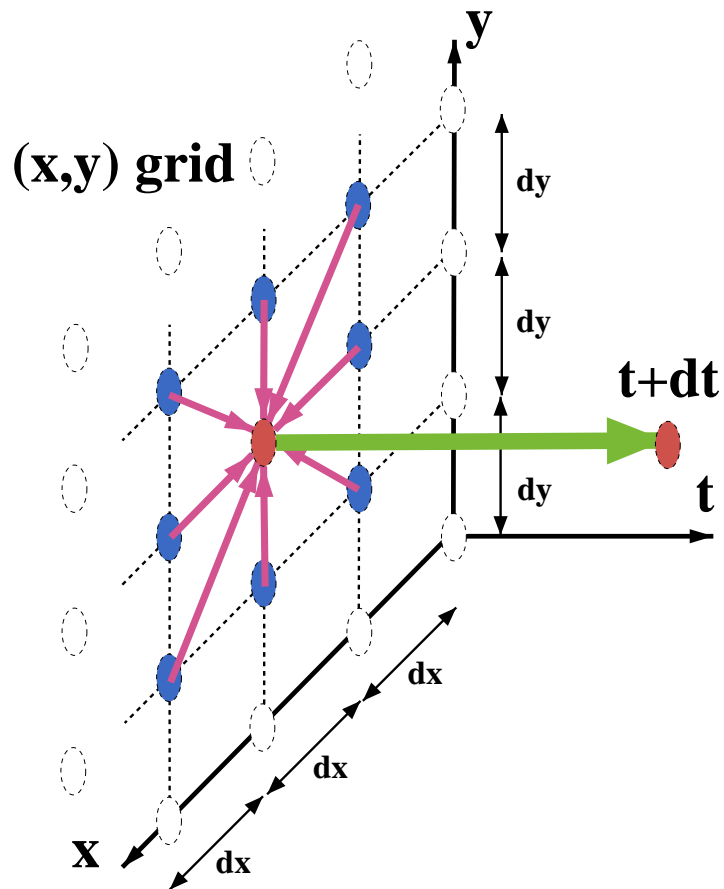
- make successive steps in time for all  $(x, y)$  on the spatial grid
- values at different  $(x, y)$  can be computed in parallel

# PDE: Numerical Solution (IV)



- make successive steps in time for all  $(x, y)$  on the spatial grid
- values at different  $(x, y)$  can be computed in parallel
- each point on the grid needs to sync with the neighbours

# PDE: Numerical Solution (IV)



- make successive steps in time for all  $(x, y)$  on the spatial grid
- values at different  $(x, y)$  can be computed in parallel
- each point on the grid needs to sync with the neighbours

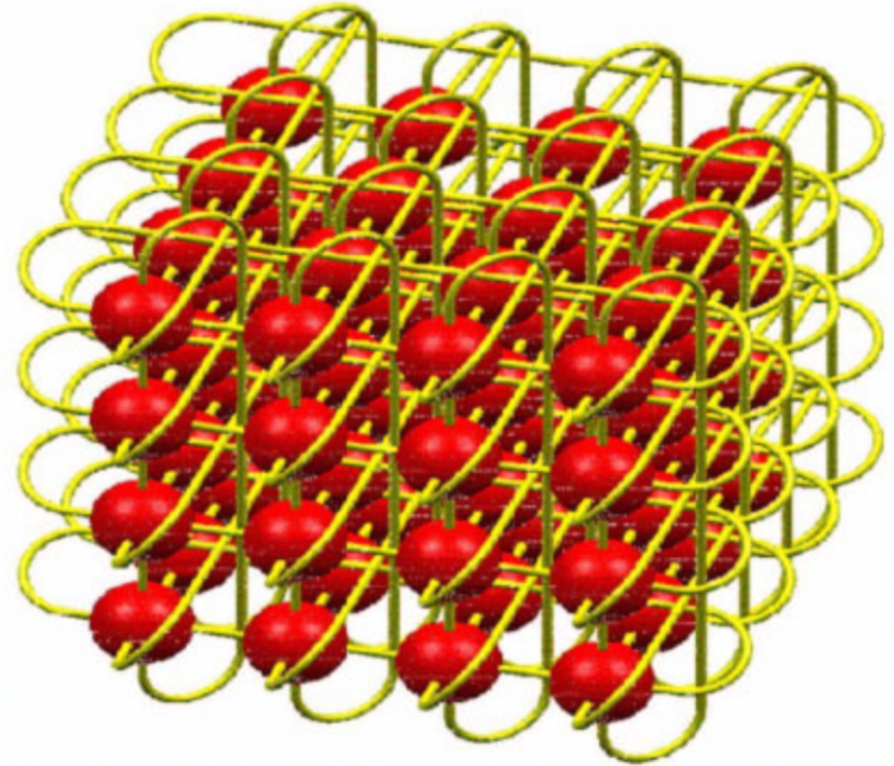
# PDE: Boundary Conditions

- normally satisfied by a mesh



# PDE: Boundary Conditions

- normally satisfied by a mesh
- for periodic boundary conditions a torus network is advantageous



# Parallel Computers: Requirements

- large number of processors: as many as possible, up to Amdahl's limit:

$$T = S + \frac{P}{N}$$

- uniform capabilities (fast CPUs don't wait for slow ones)
- limited paging (slows down some processes)
- very fast network so that CPUs don't waste time waiting for sync messages (+ preemption)
- especially fast message passing between nearest neighbours in a mesh
- torus topology for periodic boundary conditions etc.



# Parallel Computers: Networks

- high-speed (high bandwidth, low latency) interconnects
  - Gigabit Ethernet
  - SCI — Scalable Coherent Interface (e.g., Dolphin)
  - Myrinet
  - Quadrics — QsNet
- provide programmability, scalability, performance, integrability into large-scale systems
- global virtual memory (Quadrics)
  - data transfer directly between address spaces
  - maintaining hardware protection
  - extension to virtual memory
- fault detection and fault tolerance in hardware



# Network Topology

- mesh and torus
  - tradeoff: cannot choose an arbitrary set of nodes for a job
- full crossbar
  - very expensive
- fat tree
  - efficient broadcasts and collective operations
- often more than one network



# Parallel Computers: Top 500

- <http://www.top500.org> — trends in HPC
  - released twice a year since 1993
  - based on a set of benchmarks (LINPACK)
  - current (# 20): 11/2002, next (# 21): 06/2003
- some statistics
  - top countries: USA (45%), Germany, Japan, UK, France, Italy
  - computers: constellations (40%), MPP (40%), clusters (20%), SMP (a few)
  - manufacturers: HP (27.4%), IBM (25.8%), Sun (17.6%), SGI (9%), Cray (4.4%)
  - purpose: industry (45%), research (23%), academic (20%), classified, government



# Nominee # 5: Linux Cluster at LLNL

- MCR Linux Cluster at LLNL: Linux NetworX/Quadrics
- Each Node: QsNet ELAN3 by Quadrics, 4GB SDRAM, 120GB HD
- 2304 Intel 2.4GHz Xeon processors
- 1152 nodes plus separate hot spare and development clusters
- Lustre OS cluster-wide FS from Cluster File Systems, Inc
- 4.6 TB of aggregate memory
- 11.2 Tflops peak performance



# Nominee # 4: ASCI White at LLNL

- ASCI: Accelerated Strategic Computing Initiative (DOE)  
Purpose: safety of nuclear stockpile without underground testing
- 3 IBM RS/6000 SP systems: White, Frost, and Ice
  - White: 512 node × 16-way SMP, classified
  - Frost: 68 node × 16-way SMP, unclassified
  - Ice: 28 node × 16-way SMP, classified
- 12.3 Tflops peak performance
- 8192 375MHz Power3 CPUs
- 6TB total RAM
- 160TB total storage (6 times the Library of Congress)
- 200 cabinets in 2 basketball courts, weight: 106 ton



# Nominees # 3 and # 2: ASCI Q at LANL

- 3072 AlphaServer ES45s from HP
- 12288 EV-68 1.25GHz CPUs with 16MB cache
- 33 TB RAM
- 664TB global storage on Gbit fiber-channel disks
- Quadrics network with 6144 PCI adapters and 6 1024-way switches (fat tree) in 2 rails
  - bandwidth 250 MB/s/rail
  - latency  $\approx 5\mu s$
- 900 cabinets in 20,000 sq. feet
- 204 miles of cables under the floor
- 30 Tflops theoretical peak performance when fully operational



# And The Winner Is ...

The Earth Simulator, The Earth Simulator Center, Yokohama, Japan. Purpose: climate modelling.

- 5120 (640 8-way SMP) 500MHz NEC CPUs
- 2GB RAM per CPU (10 TB total RAM)
- 640×640 crossbar switch
- 16GB/s inter-node bandwidth
- 320 cabinets for CPUs, 65 cabinets for interconnect
- 65m × 50m area
- hundreds of km of cabling
- theoretical peak performance around 40 Tflops



# Top 10 Highlights

- ES is firmly # 1
- ASCI Q got into slots 2 and 3 with 7.7 Tflops each
- 2 new PC clusters made # 5 and # 8
- 2 other new systems are at # 9 and # 10 (IBM Power4)
- ASCI Red out (in Top 10 since 1997, 7 times # 1)
- 3.2 Tflops are needed to get into Top 10
- Top vendors: HP (4), IBM (3), NEC, Linux NetworX, HPTi (1 each)



# Top 500 Highlights

- 47 Tflops systems (up from 23 in 6 months)
- # 500 was # 318 6 months before
- clusters present at all levels of performance
  - 93 clusters (up from 80 in 6 months), 14 “self-made”
  - 55 Intel, 6 AMD clusters, 4 Sun, 3 Alpha, 23 HP AlphaServer
- US is the leading user (46% of systems, 51% of installed performance) and producer (91%)
- HP leads slightly by number of installed systems with 137, IBM second with 131, Sun third with 88
- IBM leads firmly by installed performance (31.8%), followed by HP (22.1%) and NEC (14.6%)



# Scaling **Waaay** Up

## Part II

### BlueGene/L



# BlueGene/L: Overview

- joint reserach partnership between IBM and LLNL
- part of ASCI Advanced Architecture Research Program
- new SOC architecture
- target peak performance 360 Tflops
- operational in 2004/2005
- price/performance and power/performance unobtainable with conventional architectures
- first step in IBM's multiyear initiative to build a petaflop scale machine for life sciences
- standard compilers and message passing environment
- autonomous computing



# BlueGene/L: Motivation

Traditional approach: clustering large, fast SMPs

- Physical constraints
  - power consumption
  - footprint

Both LLNL and LANL construct buildings with 2-4 times floor space and 10 times the cooling capacity of existing facilities

- Technological constraints
  - CPU cycle times decrease faster than memory access times

The fastest processors will deliver a continuously decreasing fraction of their peak performance, despite ever more sophisticated memory hierarchies



# BlueGene/L: Design Goals

- scalable parallel supercomputer of up to 65536 compute nodes
- cost effectiveness
- low power (1 MW)
- low cooling (300 tons)
- low floor space (less than 2500 sq. ft)
- target peak performance of 360 Tflops
  - if both CPUs on a node do computations
  - often one CPU does computations and one is used for messaging, for peak performance of 180 Tflops
- MTBF of the order of 10 days



# BlueGene/L: Node Design

- very large number of nodes
  - SOC: 2 PowerPC CPU, embedded DRAM, 3 cache levels, and multiple high-speed interconnection networks with sophisticated routing on a single ASIC
  - modest clock rate
  - low power consumption
  - low cost
- memory access times are close to CPU cycle times
  - advantageous for power consumption
  - dense packaging (1024 nodes in a single rack)
- integration of communication on the same ASIC
  - no separate high-speed switch

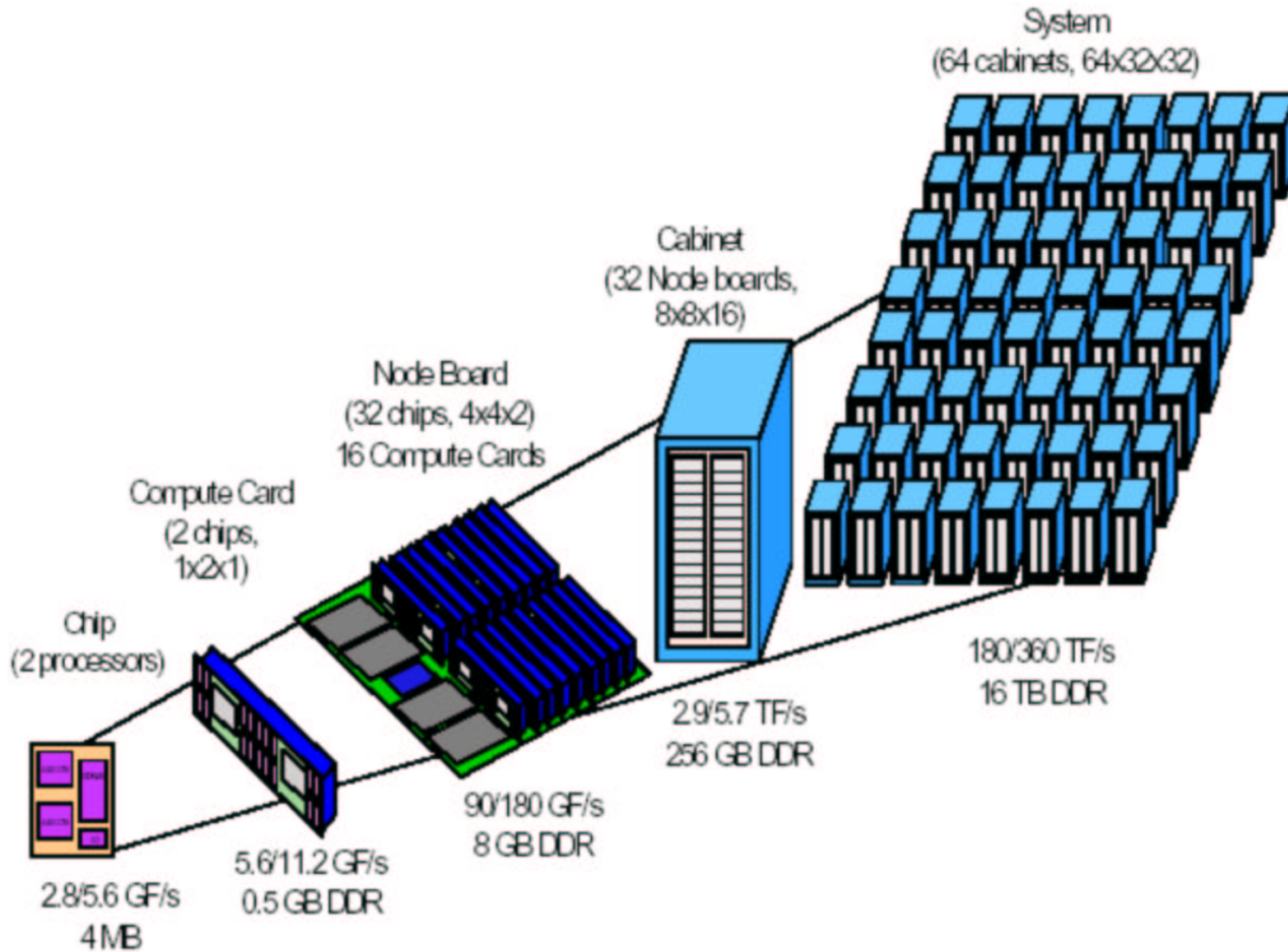


# BlueGene/L: Compute Node Details

- up to 2GB of local memory
  - current plan: 9 SDRAM chips with 256MB/node
- ASIC
  - 0.13 micron technology
  - target clock rate 700 MHz
- performance
  - 4 flop/cycle (2 64-bit floating point multiply-adds per cycle)
- lightweight kernel (Linux-based)
  - all the functionality needed for scientific code
  - basic communication tasks



# BlueGene/L: Hierarchy



# BlueGene/L: I/O Nodes

- handle communications between compute nodes and everything else
- same ASIC as in compute nodes
- expanded memory
- Gigabit Ethernet connections
- Linux OS
- number is flexible
  - up to 1 I/O node for every 8 compute nodes
  - for a 65536-node machine we expect 1 I/O node for every 64 compute nodes, 1024 I/O nodes in the system



# BlueGene/L: Networks

At least 5 different networks:

- 3D torus network interconnecting all the c-nodes
  - computational backbone
  - 2.8 Gb/s/link
- Global combining/broadcast tree
  - connects c-nodes with I/O nodes
  - collective operations over the entire application
- Gigabit Ethernet to file systems, external hosts, etc
  - I/O nodes only
- Gigabit Ethernet for machine control
- Global barrier and interrupt network



# BlueGene/L: Operating System

- design goals:
  - familiar Unix-like look and feel
  - high performance levels
- split the OS between c-nodes and I/O nodes
  - each c-node executes a single application process
  - I/O nodes run Linux and provide
    - FS interface
    - system control
    - job management: launch, termination, and signalling
    - debug capabilities
    - etc



# BlueGene/L: C-Node OS

- single-user OS
- supports execution of a single dual-threaded application
- each thread is bound to one of the 2 CPUs
- static virtual address space
- no context switches
- no demand paging
- user level library for direct access to torus and tree networks



# BlueGene/L: I/O Node OS

- Linux in full multiprocess mode
- only system software, no application code
- I/O operation (on a file or a socket) path:
  - the operation is shipped from the c-node through the tree network to a service process on the I/O node
  - the service process issues the operation against the I/O node OS
  - the result (return code in case of write, data in case of read) is shipped back to the c-node
- debugging operations are shipped from an I/O nodes to a c-node for execution against the compute process
- authentication, accounting, authorization



# BlueGene/L: Issues Of Scale

- a jump of almost 2 orders of magnitude from hundreds (ASCI White, Earth Simulator) or thousands (ASCI Q) of nodes
- scalability challenges for system management
  - boot
  - software installation
  - user account management
  - system monitoring
- scalability challenges for job management
  - job scheduling
  - resource allocation
  - job execution and control



# BlueGene/L: System Hierarchy

- conventional approach: central control workstation (CWS)
  - centralized management is easy
  - successfully used in numerous large systems
  - inherently non-scalable
- BG/L approach: organize 65536 c-nodes into 1024 “processing sets” (p-sets)
  - p-set: 64 c-nodes + I/O node, c-nodes look like devices attached to the I/O node
  - a p-set is a monolithic entity for system management
  - the 1024 p-sets are managed from a service node (CWS)



# BlueGene/L: Benefits Of P-Sets

- cluster management is non-intrusive to the applications that run on c-nodes
- only I/O nodes, forming a much smaller effective system, are visible
- I/O nodes can be self-managed
  - a self-managed set of I/O nodes does not need a CWS
  - can rely on automatic failover to recover from failures



# BlueGene/L: System Boot

- controlled by the service node through the control network
  - they must be booted first
  - discovery mechanism to find I/O and compute nodes
  - boot the I/O node first, then the c-nodes it controls
- stateless nodes: no ROM, disks, BIOS, or boot loader
- a small boot loader is written directly into the node memory
- a larger boot image is loaded using a communication protocol executed by the boot loader
- in the future we may use a boot server for I/O nodes, and c-nodes will load their boot images from I/O nodes



# BlueGene/L: Node Boot

- single images for c-nodes (64KB) and I/O nodes (2MB + 16MB ramdisk)
- additional filesystems are NFS-mounted
- system, configuration, utilities ( `/etc` , `/bin` , `/usr` , `/proc` ) are in the ramdisk
- user directories ( `/home` ) are NFS-mounted
- per-node configuration (“personality”)
  - for I/O node: MAC and IP addresses, unique node ID, the set of c-nodes in its p-set, routing tables, etc
  - for c-node: unique ID, the ID of the corresponding I/O node, routing tables



# BlueGene/L: Software Installation

- reconfiguration of the I/O node ramdisk
- requires direct manipulation of the ramdisks
- requires I/O node reboot, cannot be done when jobs are running in the p-set
- unless all I/O nodes are rebooted at the same time, may lead to inconsistencies between I/O nodes



# BlueGene/L: User Management

- only I/O nodes have a notion of a “user”
- compute nodes know nothing about users
- I/O nodes run Linux, so we can use normal UNIX/Linux utilities
- can use services such as NIS, and keep user configuration at the service node (NIS server)
- for new users it is necessary to create a new home directory on the file server



# BlueGene/L: System Logs

- I/O nodes generate a normal Linux system log
- the system log contains information about the c-nodes in the p-set
- I/O nodes collect and filter information about events generated by hardware and software in the c-nodes
- information is passed by severity level and other criteria to a centralized repository
- the repository can be mined for statistics, prediction of failures, etc
- existing Linux cluster tools can be adapted for BG/L



# BlueGene/L: Device Monitoring

- some device information can be obtained only by the service node through the control network
  - fan speeds
  - power supply voltages
- special communication protocol with I/O and c-nodes
- nodes report events that can be logged or acted upon by the service node
- completely separate, independent (of I/O nodes, other networks) monitoring service
- can be used in cases of multiple or system-wide failures to retrieve important information



# BlueGene/L: Job Scheduling

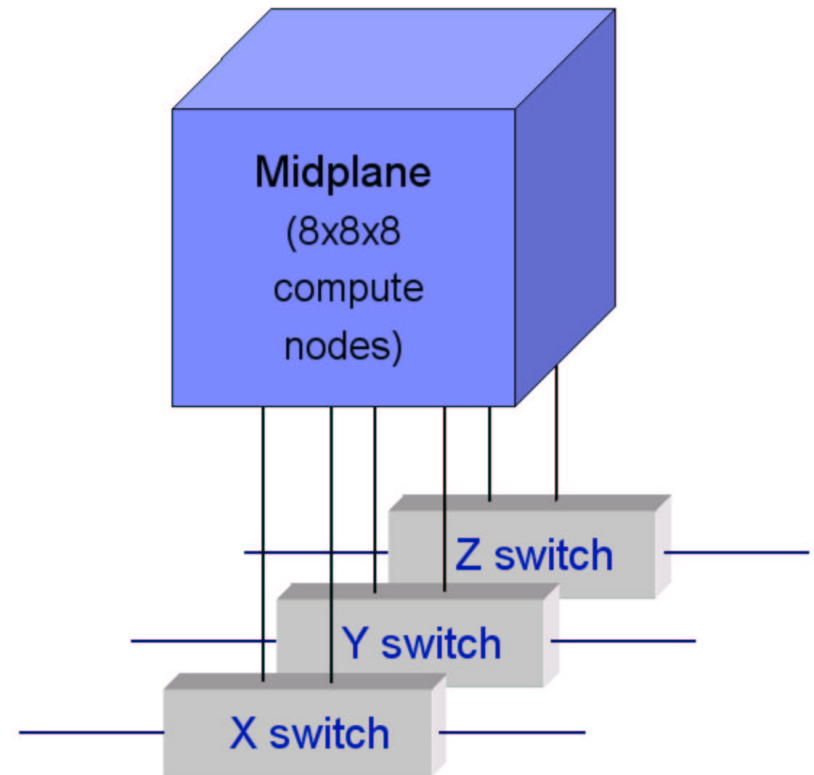
- time slicing
  - FCFS
  - backfilling
  - etc.
- space slicing
- gang scheduling

We cannot use gang scheduling or migrate jobs on BG/L.  
Job scheduling and allocation are essentially separate tasks.



# BlueGene/L: Wiring

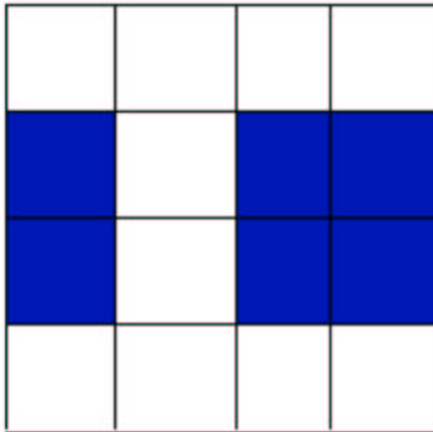
- midplane — a wiring unit ( $8 \times 8 \times 8 = 512$  c-nodes)
- midplanes are connected via switches
- 3 switches per midplane (X,Y,Z)
- each switch has 6 ports (2 to midplane)
- midplanes can have more than 6 neighbours
- it is possible to create sub-tori



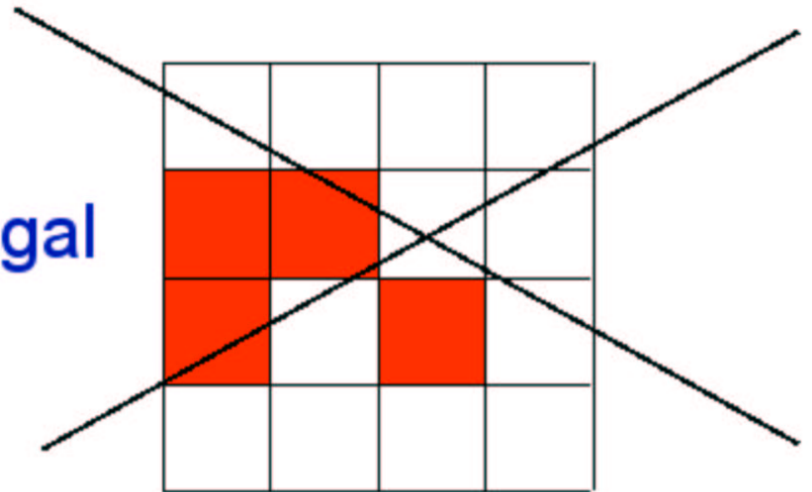
# BlueGene/L: Partitions

- base partition (BP) — a 3D rectangle of c-nodes
  - the minimal allocation unit
  - currently  $8 \times 4 \times 4 = 128$  c-nodes (2 p-sets)
- partition — a set of BPs allocated to a job
  - can be wired as a sub-mesh or a sub-torus

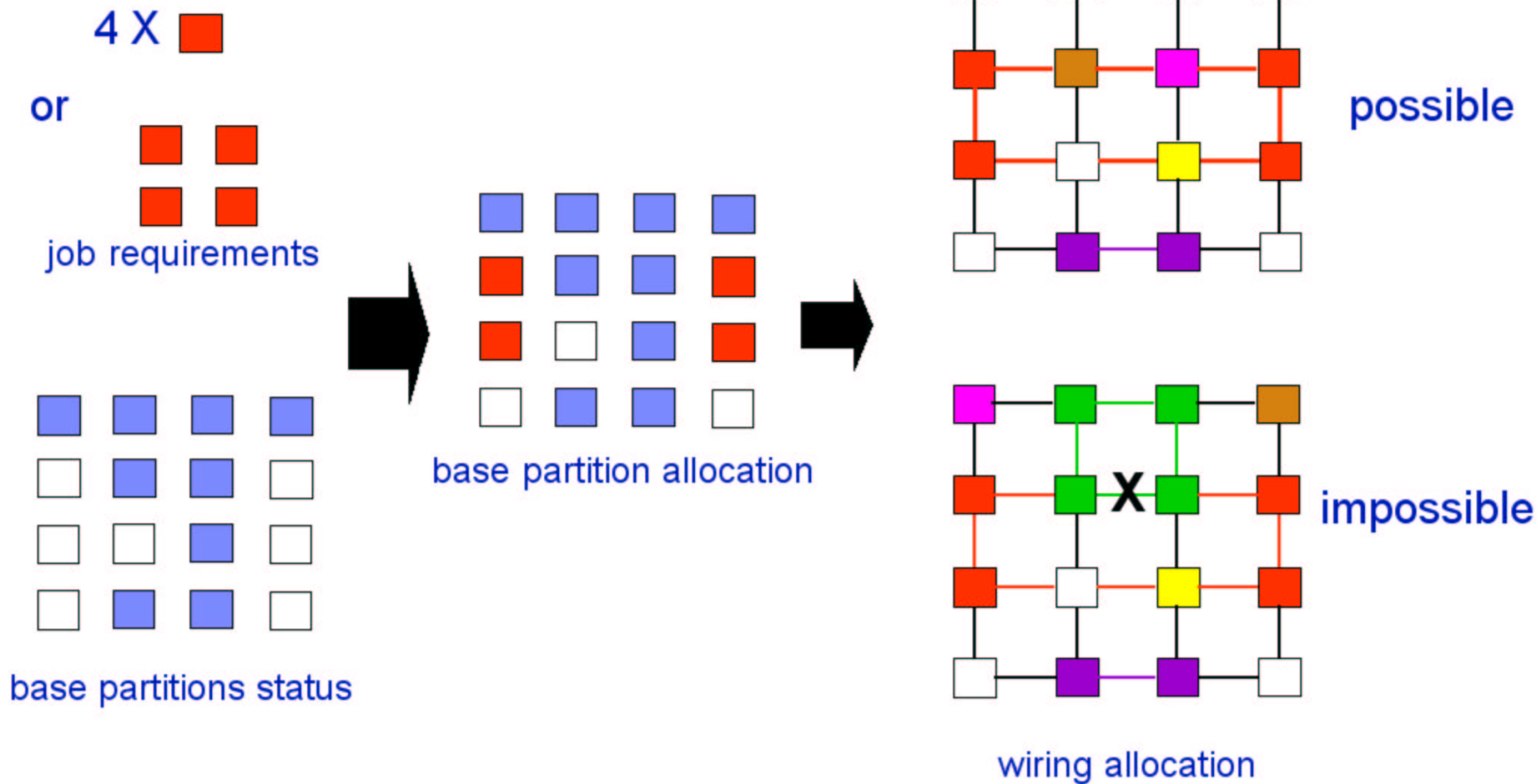
legal



illegal

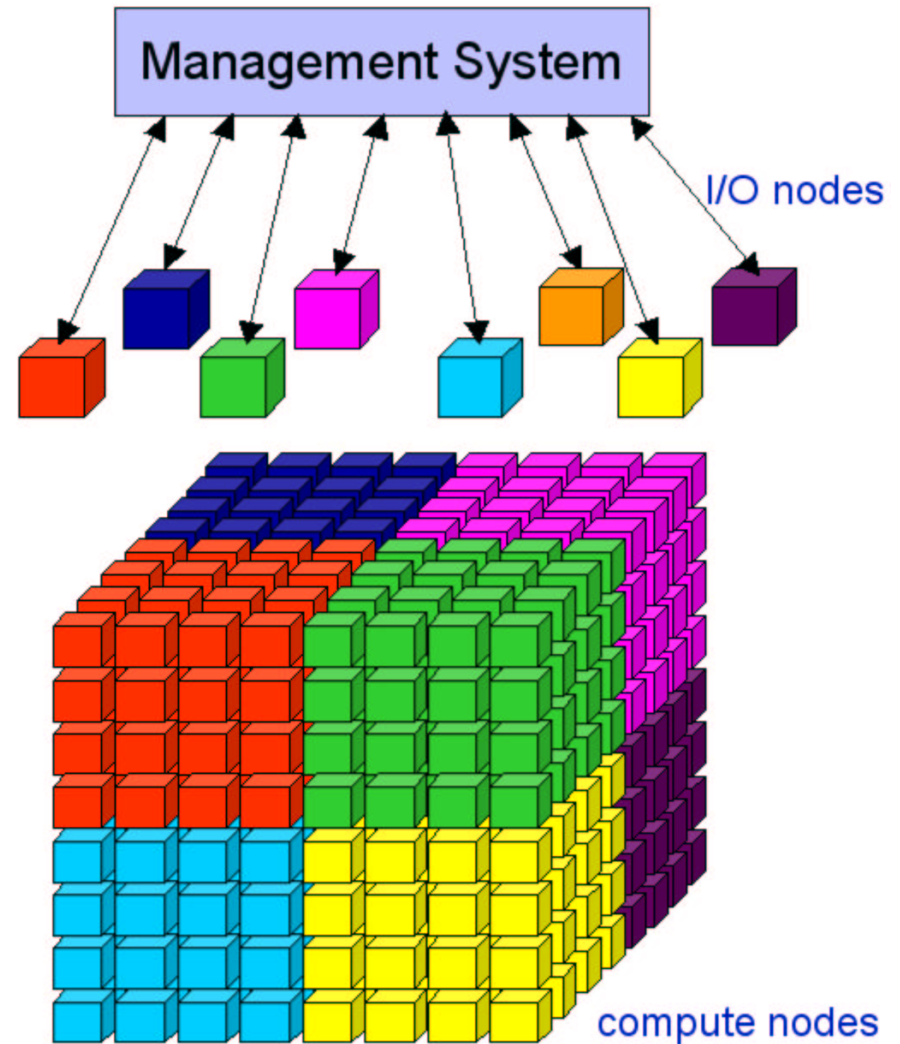


# BlueGene/L: Job Allocation

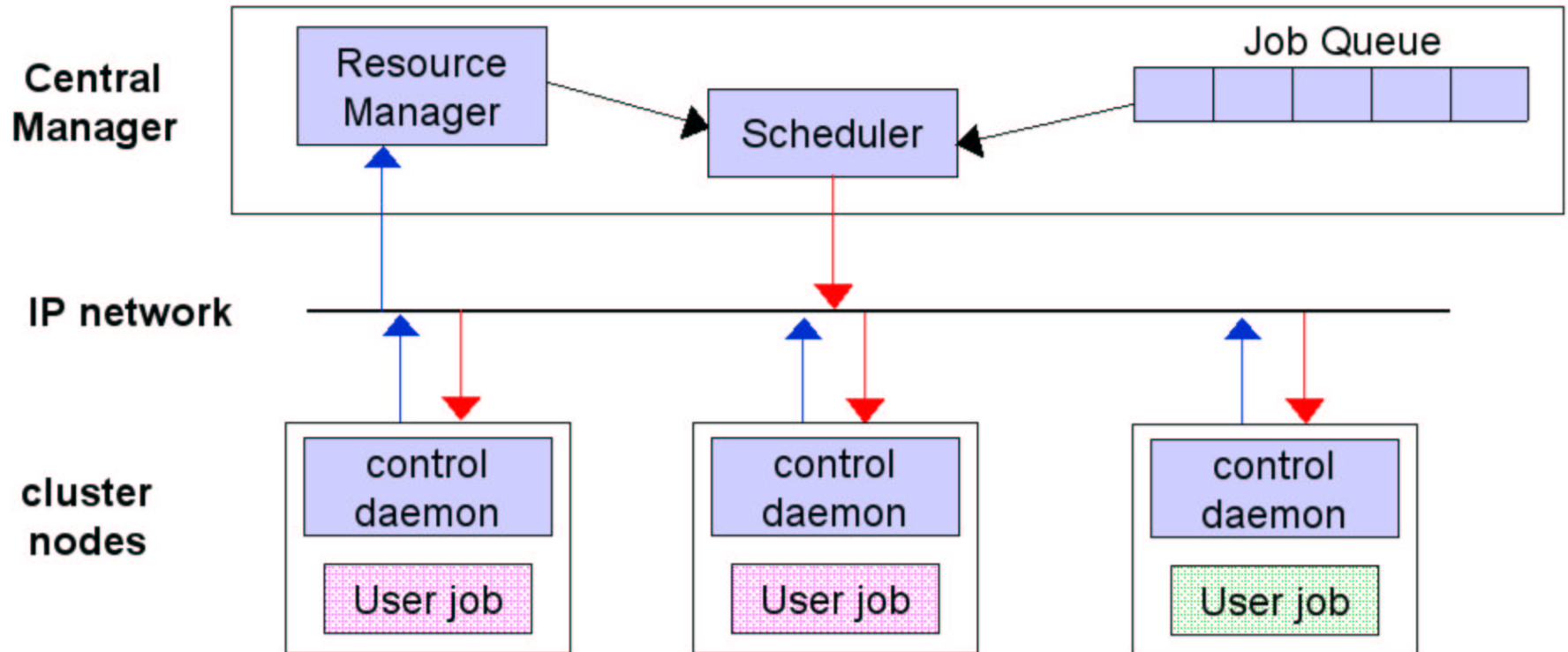


# BlueGene/L: Job Management

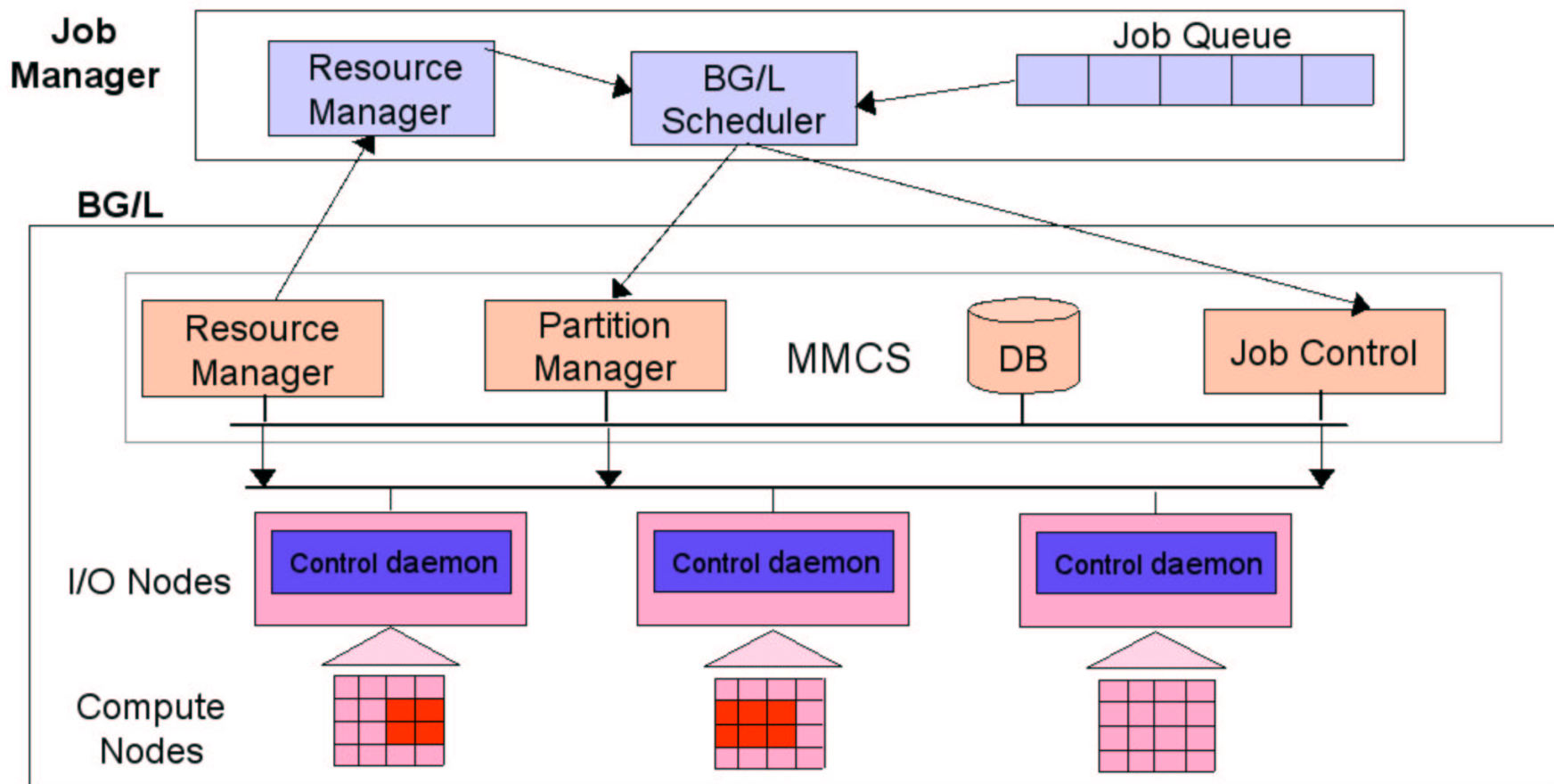
- need to manage jobs running on 64K c-nodes
- from system prospective, jobs run in 1K I/O nodes
- job management
  - start, monitor, signal, terminate
  - authentication, authorization
  - debugging



# Commodity Cluster Job Management



# BlueGene/L Job Management



# Summary

- new scale of parallel computing
  - 64K nodes
  - 180/360 Tflops
  - unmatched price/performance
  - cellular approach allows almost limitless scalability
- hierarchical organization facilitates management
- mixed conventional/special purpose Linux OS
- default programming model — MPI
  - other are being investigated
- many challenges and open questions ahead
  - performance and reliability



# Closing The Torus



# Closing The Torus



**The Fine Print:** The following comments are owned by whoever posted them. We are not responsible for them in any way.

**Re:Imagine a Beowulf cluster of these babies! (Score:0)**  
by Anonymous Coward on Monday April 21, @03:44PM  
(#5775329)



# Final Credits

BG/L team at IBM Haifa Research Laboratory

- Yariv Aridor
- Tamar Domany
- Edi Shmueli
- Yoav Gal
- Oleg Goldshmidt



# For The Inquisitive

- BlueGene/L (IBM Research)
- Job Management on BlueGene/L (IBM Haifa Labs)
- <http://www.top500.org>
- Top 5
  - MCR cluster at LLNL
  - ASCI White at LLNL
  - ASCI Q at LANL
  - The Earth Simulator
- The ASCI Program

