IBM Research

Blue Gene/L: Delivering Large Scale Parallelism

Jose Gabriel Castaños

castanos@us.ibm.com

BlueGene/L Overview

- Collaboration with LLNL for 180/360TF system announced in November 2001
  - target scientific and emerging commercial apps.
  - ~64K low power nodes for density, low cost
  - scalable from small to large systems (2048 processors/rack)
  - 512 node prototype this year, full system 2004/5
- Leverages high speed interconnect and system-on-a-chip technologies
  - embedded PowerPC 440 cores
  - embedded DRAM
  - integrated switch

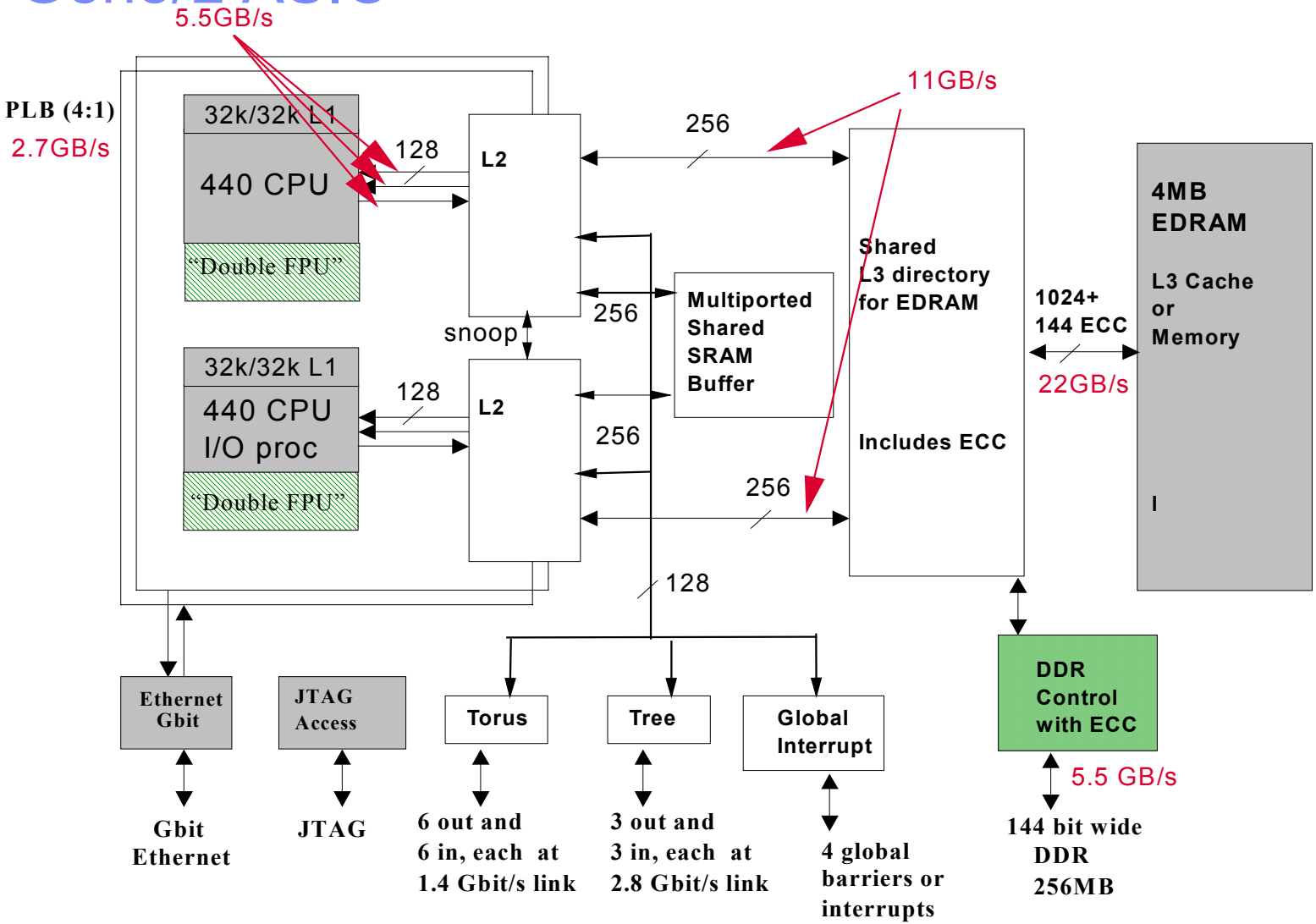
Outline

- Blue Gene/L architecture
  - "An Overview of the Blue Gene/L Supercomputer". SC'02
- Blue Gene/L system software and software development environment
  - "An Overview of the Blue Gene/L System Software Organization". Euro-Par 2003
- Simulation infrastructure
  - "Full Circle: Simulating Linux Clusters on Linux Clusters". LCIHPC'03
- Preliminary results
- Conclusions

IBM Research

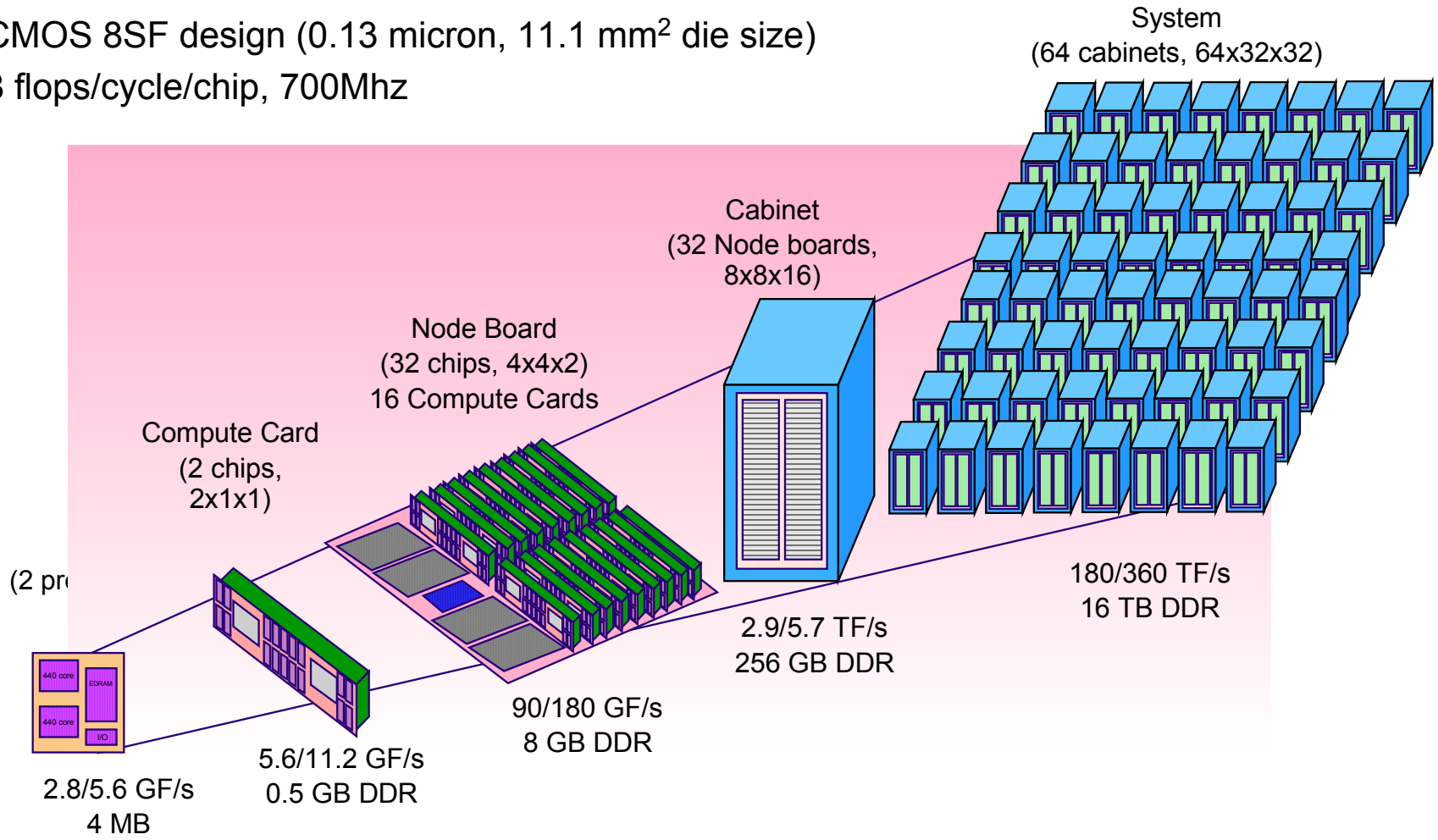
Blue Gene/L Architecture

Blue Gene/L ASIC

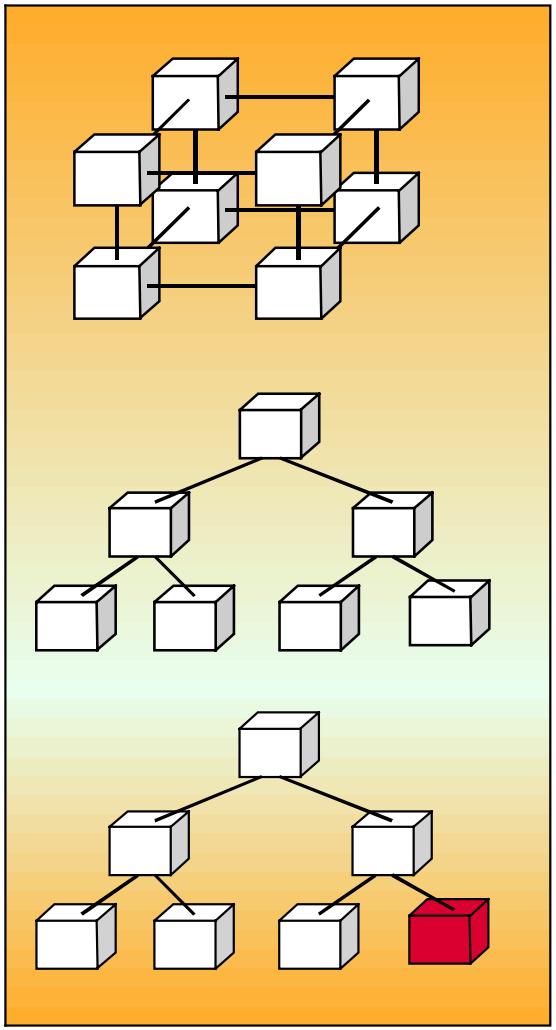


Building Blue Gene/L

- CMOS 8SF design (0.13 micron, 11.1 mm^2 die size)
- 8 flops/cycle/chip, 700Mhz



BlueGene/L Five Interconnection Networks



3-Dimensional Torus

- Interconnects all compute nodes (65,536)

Global Tree

- Point-to-point, One-to-all broadcast, reduction functionality

Global Interrupts

- AND/OR operations for global barriers
- 1.5 microseconds latency (64K system)

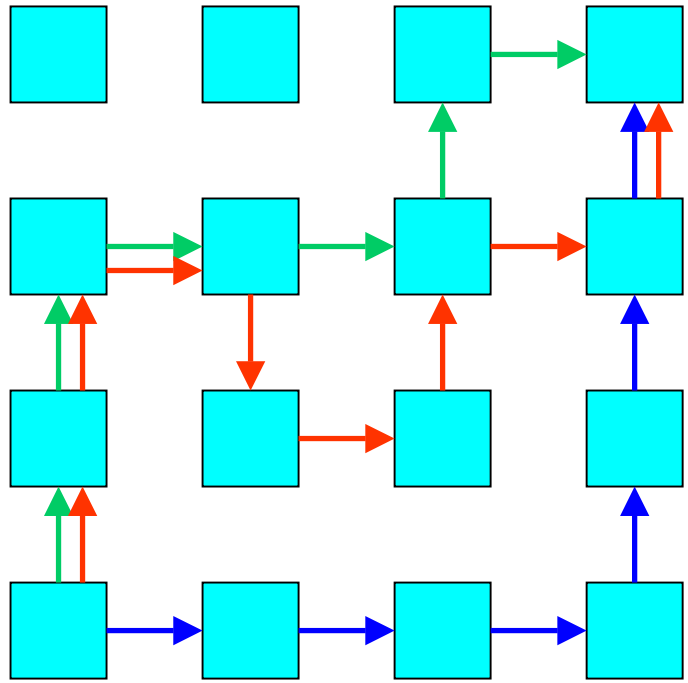
Ethernet

- Incorporated into every node ASIC
- Active in the I/O nodes (1:64 in LLNL configuration)
  - 1K 1Gbit links
- All external comm. (file I/O, control, user interaction, etc.)

JTAG (Control)

Torus Network

- The 3D torus network provides point-to-point delivery of packets between all (64K) compute nodes
  - packets can be from 32 to 256 bytes long, in 32-byte increments
- 1.4Gb/s on all 12 node links (2.1 GB/s per node)
  - 350/700 GB/s bisection bandwidth
- Each packet includes a header with (X,Y,Z) of destination and a “deposit bit” flag
- Reliable, deadlock free
- Minimum dynamic adaptive routing in HW
  - multiple routes are possible
  - always moves closer to destination
  - can arrive out of order



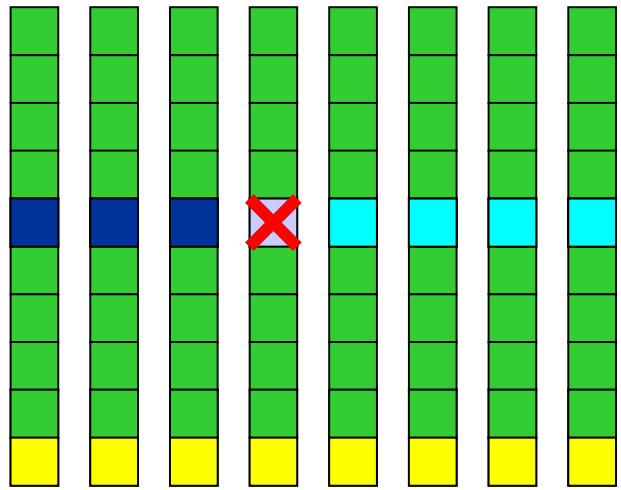
Tree Network

- Fast, configurable, point-to-point messages of fixed length packets (256 bytes)
- Interconnects all compute and IO nodes
  - function shipping (IO and control)
- Broadcasts and integer reductions (integrated ALU) in ~2.5 microseconds for a 64K system
  - floating point reductions in two passes
- 2.8 Gb/s bandwidth per link

Partitioning and Reliability

- Every partition of the machine is dedicated to run a single user/single job
  - electrically isolated
  - no crossing packets from other partitions
- All wires (torus, tree, global interrupts) between midplanes are routed through custom chips (Link chip)
  - allows partition of a large torus into two torii (with restrictions) by rerouting signals
- Therefore, the default unit of allocation is a midplane
  - half a rack or 512 compute nodes forming an 8x8x8 torus
  - smaller **meshes** are possible (2x2x2)
- Swap-off bad sectors for reliability
  - packaging allows easy access to replace broken cards
  - restart from previous checkpoint

Top view of BG/L



IBM Research

Blue Gene/L System Software and Software Development Environment

Principles of Blue Gene/L System Software Design

- Simplicity
- Efficiency
- Familiarity

Simplicity

- Strictly space sharing
  - one job (one user) per electrical partition of the machine
  - one process per compute node
  - one thread of execution per processor
- Offload engines
  - system services (I/O, process control, debugging) are offloaded to I/O nodes – identical hardware to compute nodes
  - system control and monitoring offloaded to service nodes
- Hierarchical organization for management and operation
  - single-point of control at service node(s)
  - manage system as a cluster of I/O nodes
- Flat view for application programs – collection of compute processes

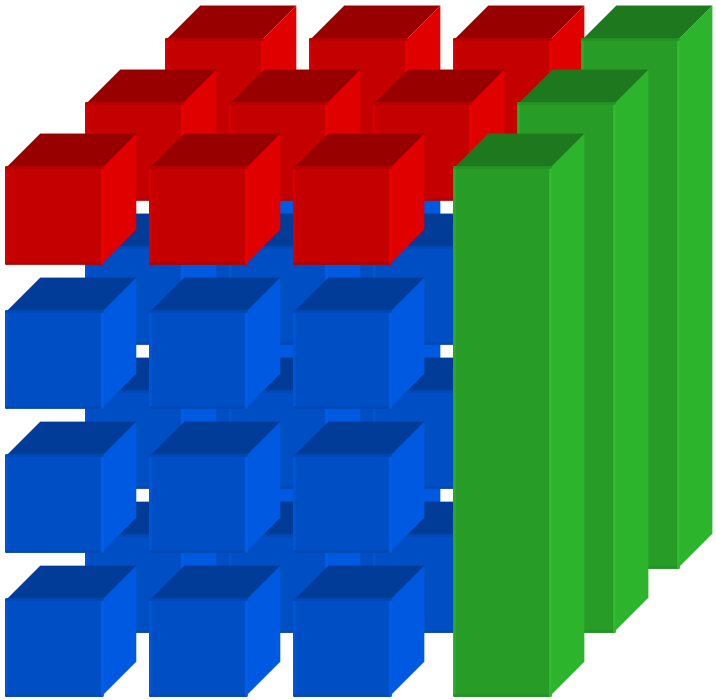
Efficiency

- Dedicated processor for each application-level thread
  - deterministic, guaranteed execution
  - maximum performance for each thread
  - physical memory directly mapped to application address space – not TLB misses (also, no paging)
- System services executed on dedicated I/O-node
  - no daemons interfering with application execution
  - no asynchronous events on computational volume
  - IO over tree
    - do not pollute the torus
- User-mode access to communication network
  - electrically isolated partition dedicated to one job + compute node dedicated to one process = no protection necessary!
  - user-mode communication = no context switching to supervisor mode during application execution (except for I/O)

Familiarity

- Fortran, C, C++
- MPI
- Linux development environment
  - cross-compilers and other cross-tools execute on Linux front-end nodes
  - users interact with system from front-end nodes
- Tools – support for debuggers, hardware performance monitors

Blue Gene/L System Software Architecture



- User applications execute exclusively in the **compute nodes**
  - avoid asynchronous events (e.g., daemons, interrupts)
  - avoid complex operations on compute nodes
  - custom solution: Compute Node Kernel (CNK)
- The outside world interacts only with the **I/O nodes**, an offload engine
  - standard solution: Linux
- Machine monitoring and control also offloaded to **service nodes**: large SP system or Linux cluster
  - IBM middleware (LoadLeveler, CSM), JTAG
  - custom components for booting, configuration, partitioning, monitoring

Compute Node Kernel (CNK)

- User applications execute exclusively in the compute nodes
  - custom solution: Compute Node Kernel (CNK)
- Minimalist kernel
  - user space/kernel space
  - single user, single program
  - at most two threads (actually 1 ½)
  - flat, fixed size 256 MB address space
    - TLBs statically allocated at startup
      - no dynamic TLB management
      - no TLB misses
      - same set of TLBs for both cores: each core can read and write to the other's memory
  - torus mapped to users space
  - standard POSIX API
    - GLIBC 2.2.5 runtime library
    - function shipping to I/O nodes over tree
  - simple
    - ~5000 lines of C++

Using the 2nd CPU in a Blue Gene/L Compute Node

- **Communication coprocessor mode:** CPU 0 executes user application while CPU 1 handles communications
  - preferred mode of operation for communication-intensive and memory bandwidth intensive codes
  - requires coordination between CPUs, which is handled in communications libraries
- **Computation offload mode:** CPU 1 executes some parts of user application offloaded by CPU 0, in addition to communication
  - can be selectively used for compute-bound parallel regions
  - need careful sequence of cache line flush, invalidate, and copy operations to deal with lack of L1 cache coherence in hardware
  - asynchronous coroutine model (co_start / co_join)
- **Virtual node mode:** CPU0 and CPU1 handle both computation and communication
  - two MPI processes on each node, one bound to each processor
  - distributed memory semantics – lack of L1 coherence not a problem

Linux

- The outside world interacts only with the I/O nodes, an offload engine
- 2.4.19 kernel
- BG/L extensions to standard kernel
  - booting sequence
    - no BIOS in BG/L
  - new Interrupt Controller (BIC)
  - save and restore FPU Double Hummer registers on context switch
  - new memory layout
  - new set of Device Control Registers (DCRs)
  - driver for new Ethernet macro (EMAC4) based on EMAC3
- ~40 files changed/added to standard distribution
- Ramdisk with utilities, libraries, kernel modules, initialization scripts, …

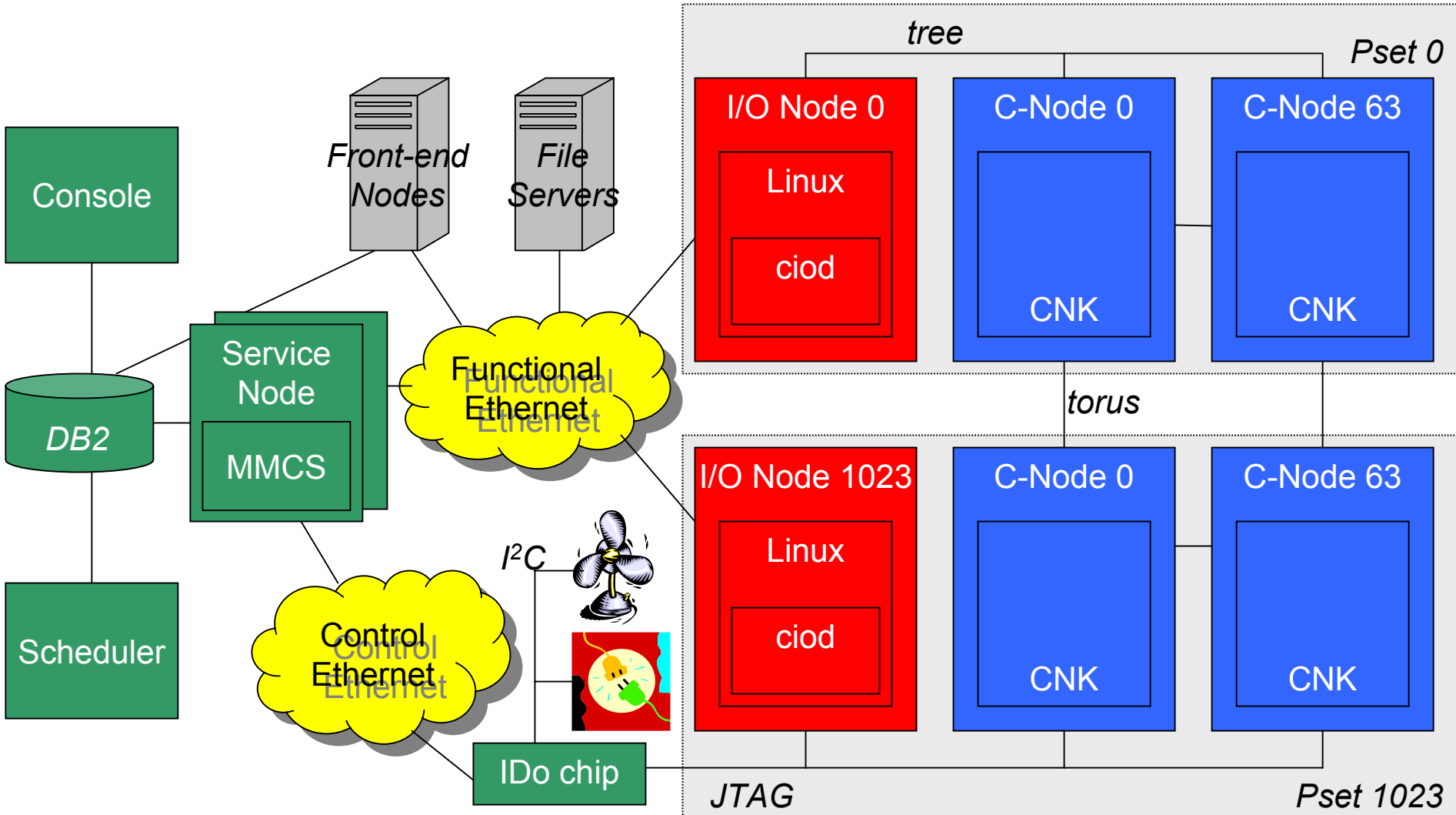
Linux (II)

- Same ramdisk and kernel image used for all I/O nodes
  - node specific configuration information exposed through proc file system
- No user code directly executes in I/O nodes
  - ciod (Control and I/O) daemon provides an interface between
    - Compute nodes in Pset (through tree)
    - Control services
    - File servers

Control System: Vision and Principles

- Local O/S (Linux for I/O node and CNK for compute node)
  - implementation of services: I/O, creation of processes, signals, debug
  - responsible for local decisions that do not affect overall operation of the machine (e.g., process scheduling in the I/O node)
- “Global O/S” (MMCS processes in the service nodes)
  - makes all global and collective decisions
  - interfaces with external policy modules (e.g., scheduler)
  - provides system management services (booting, monitoring, job launching)
- Use commercial database technology (DB2) to store all static and dynamic state of the global O/S
  - scalability, robustness, security, recovery, logging
  - database is also used as a communication mechanism for control system
- Nonarchitected network (Ethernet+JTAG) supports nonintrusive monitoring and control
  - a separate network, does not interfere with application or I/O traffic
  - secure network, since operations are critical

Blue Gene/L System Software Architecture



Development Environment

- GNU C, Fortran, C++ compilers can be used with BG/L, but they do not generate Hummer^2 code
- IBM xlf/xlc compilers have been ported to BG/L, with code generation and optimization features for Hummer^2
  - automatic SIMD FPU exploitation
  - Fortran 90
- On external service nodes
  - cross-compilers
- Subset of ESSL

Communications in Blue Gene/L

- Three logical layers
  - Packet layer
    - send and receive packets
    - close to the HW
    - abstracts hardware features like multiple FIFOs, status registers etc.
  - Message layer
    - an active message layer similar to LAPI and GAMMA, but specifically built on top of packet layer
    - handles alignment restrictions, ordering
    - no packet retransmission required
    - provides mechanisms for cache coherence and processor use policy
  - MPI

MPI

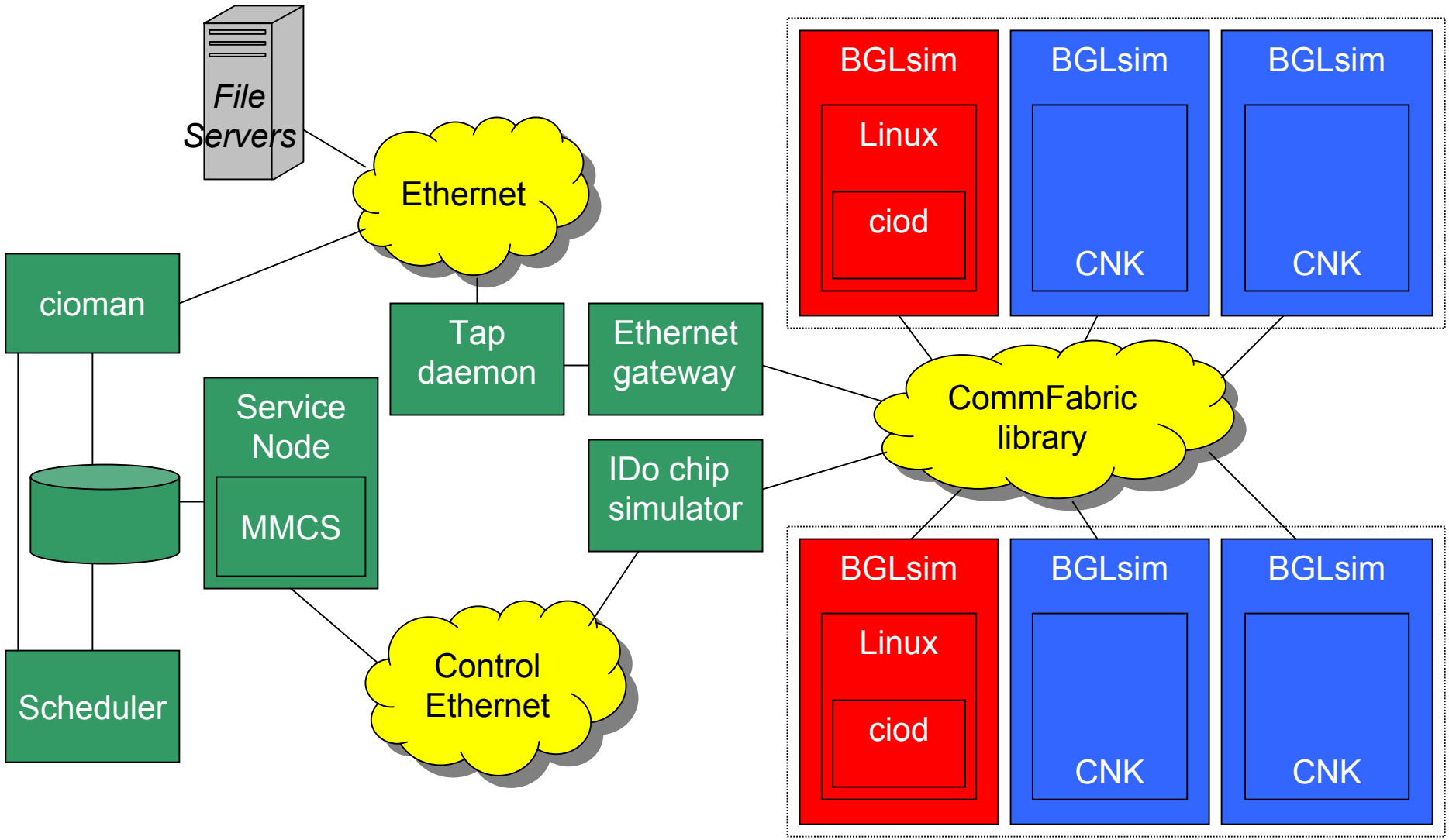
- Requirements: an MPI 1.1 compatible implementation for BG/L for message passing between compute nodes
  - only the most widely used features of MPI
- Based on MPICH 2 from ANL
  - freely available, supports large number of systems
- Point-to-point
  - implement a BG/L version ADI3 on top of message layer
- Global operations
  - torus provides efficient broadcasts on meshes
  - only MPI_COMM_WORLD operations over tree promised to LLNL
- From a user (application) point of view, system looks like a flat 64K machine
  - additional effort to embed applications in torus
- Process management
  - use BG/L's control system rather than MPICH's process managers

Simulation Environment

BGLSim Overview

- Based on early version of Mambo (IBM Austin)
- Architectural simulator of a single BG/L node
  - consumes PPC440 binaries
  - one cycle per instruction
  - statistics as instruction histograms, traces
  - runs on Linux/x86 workstations
- BG/L specific features:
  - supports 2 PPC 440 cores per chip
  - 440GP instruction set
  - Hummer^2 (Oedipus ISA) floating point
  - architecture accurate caches
    - L1, L2, L3
  - BG/L interrupt controller (BIC)
  - torus, tree devices and other networks
- ~2,000,000 simulated instructions a second
  - 1000 cycles a second in VHDL accelerators, or 2 cycles a second in VHDL simulators
- Every BG/L node is simulated by a BGLsim (mambo) process
  - potentially running in different workstations
  - simulated a whole BG/L midplane (512 nodes) before HW arrived

Blue Gene/L Simulation Environment



Preliminary Results

Initial Results

- Initial BG/L chips (first RIT) arrived by mid June and were made available to system software one week after
  - CNK booted at first attempt
  - Linux console (over JTAG network) available after just 10 hours of work
  - kernels, benchmarks and ASCI applications (exercising the full system software stack) were running by the end of July on up to 32 nodes
  - no showstoppers
- HW team built ~200 nodes
  - periodically running tests on 128 node mesh

Summary: The Three Keys for Performance on BlueGene/L

- Effective use of the Hummer^2 floating-point unit
  - SIMD-like instructions that operate on 2-element vectors: loads, stores, arithmetic, comparison
  - more general than other SIMD units – cross and replicated operands
  - can double the performance of floating-point intensive code
  - strategy: compiler optimizations, hand-tuned libraries
- Judicious use of the 2nd CPU on a node
  - can double the performance of some applications
  - strategy: co-routine programming model (be careful with caches!)
- Proper mapping of MPI applications to machine topology
  - better utilization of torus bandwidth
  - leverage hardware support for multicasts
  - strategy: MPI has the expressiveness

Conclusions

- BG/L system software stack has Linux-like personality for user applications
  - custom solution (CNK) on compute nodes for highest performance
  - Linux solution on I/O nodes for flexibility and functionality
  - MPI is the default programming model
- BG/L is testing software approaches to management/operation of very large scale machines
  - hierarchical organization for management
  - “flat” organization for programming
- Many challenges ahead, particularly in performance and reliability
- > 100 researchers and engineers in IBM and LLNL

IBM T.J. Watson Research Center

www.research.ibm.com/bluegene