First Light with BlueGene/L At Argonne

William Gropp
Argonne National Laboratory
www.mcs.anl.gov/~gropp

BlueGene/L

- What it is
- Scalability Research on BlueGene/L
- Current Status
- BlueGene Consortium

What is BlueGene/L?

- It is *not* a cluster
- Uses a modified commodity processor
- Trades clock rate for power consumption
- A sort of second generation system
  - Many technologies developed as part of the QCDOC (Quantum ChromoDynamics On a Chip) project
- 5.6 TF in a single rack (!)
  - Twice the size of the Apollo workstation I had in 1982 – and 100,000,000 times as fast
- High-performance (but proprietary) networks
  - Torus for aggregate bandwidth (e.g., neighbor exchanges)
  - Tree for collective communication (e.g., broadcast, allreduce)
  - Gigabit Ethernet, JTAG, Barrier
- Heavily exploits existing proprietary and open source software
  - IBM XL family of compilers
  - Linux, K42, GPFS
  - MPICH2
- Installations on 3 continents
  - Get your orders in now!
  - Special 128 node starter system available through the BG Consortium (more later)

Worlds Fastest Computer (Nov 2004)

1. IBM/DOE
   United States/2004*BlueGene/L beta-System*
   BlueGene/L DD2 beta-System (0.7 GHz PowerPC 440) / 32768
   IBM 70720GF (91750 peak)
2. NASA/Ames Research Center/NAS
   United States/2004*Columbia*
   SGI Altix 1.5 GHz, Voltaire Infiniband / 10160
   SGI 51870GF (60960 peak)
3. The Earth Simulator Center
   Japan/2002 **Earth-Simulator** / 5120
   NEC 35860GF (40960 peak)
4. Barcelona Supercomputer Center
   Spain/2004 *MareNostrum*
   eServer BladeCenter JS20 (PowerPC970 2.2 GHz), Myrinet / 3564
   IBM 20530GF (31363 peak)
5. Lawrence Livermore National Laboratory
   United States/2004*Thunder*
   Intel Itanium2 Tiger4 1.4GHz - Quadrics / 4096
   California Digital Corporation 19940GF (22938 peak)

- **And this for a ¼ size system!**

World's Fastest Computer

BG/L Node

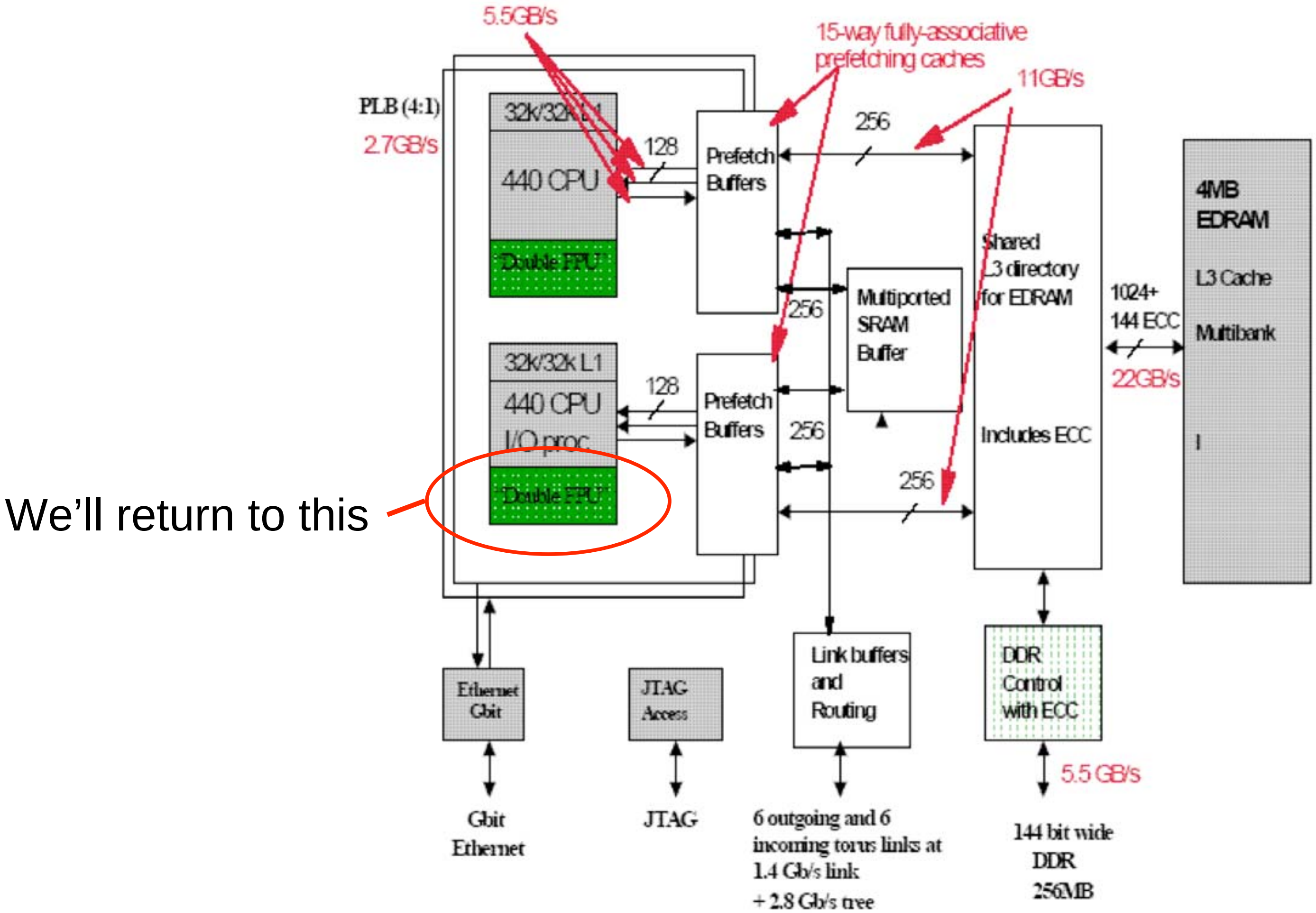
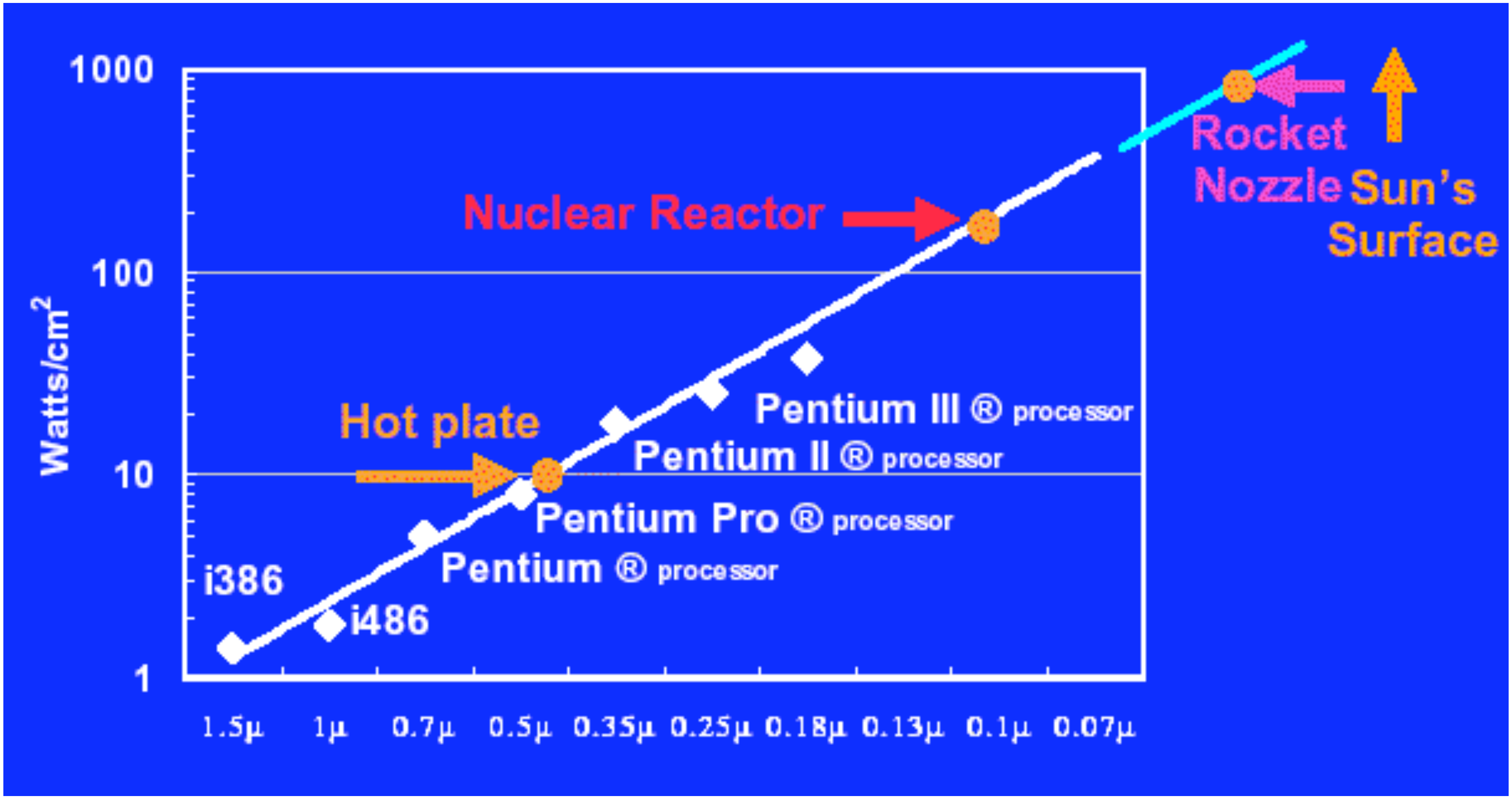


Figure 4: BlueGene/L Node Diagram. The bandwidths listed are targets.

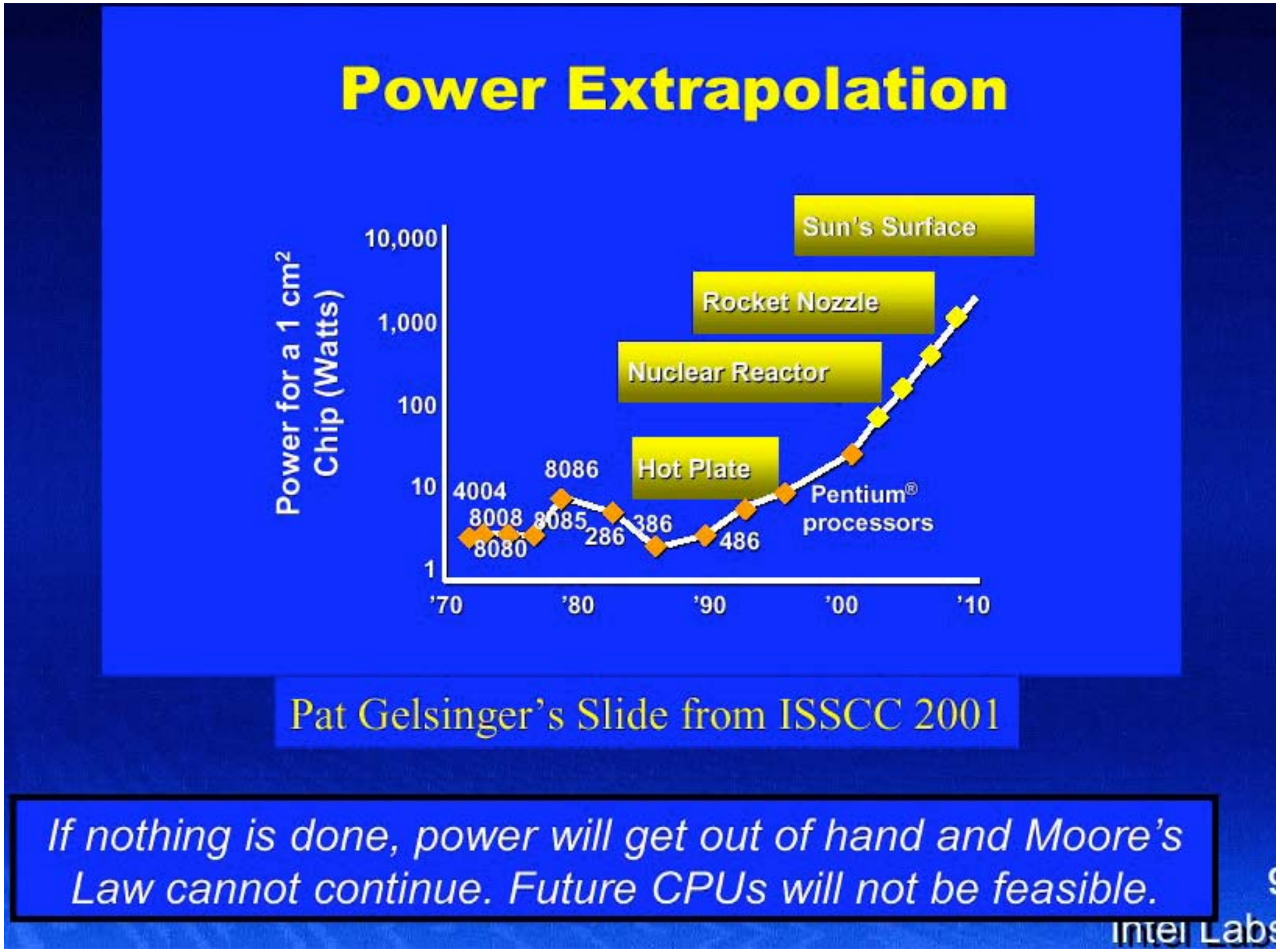
Why is BG/L Important?

- Much greater scale
  - 1k nodes/rack
  - LLNL has already run applications on 16 racks; 32 currently being installed, 64 racks by end of this year
- Partial commodity technology
  - Standard processor core
    - Leverages compiler, OS work
- New design tradeoff in HPC
  - Slower clock (0.7GHz) to reduce power consumption
  - Processor rich
    - You could say memory poor (per processor), but I won’t
  - DRAM Memory only 90 cycles away from the processor
    - But no global shared memory
- Emphasizes Scalability
- First (we hope!) in a family of production systems

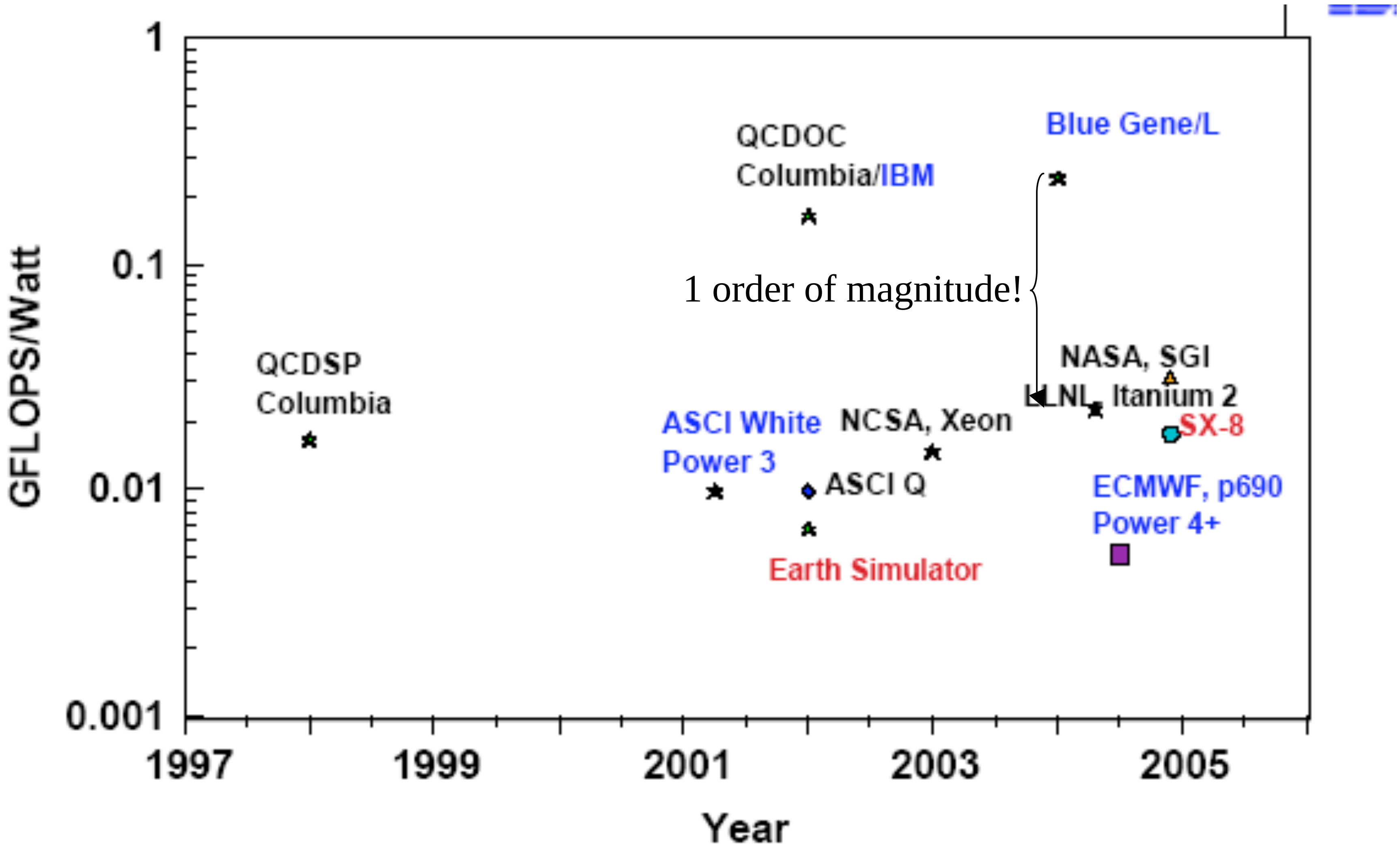
Power Dissipation



Power Dissipation



Power Efficiency



BG as Scalability Research Platform

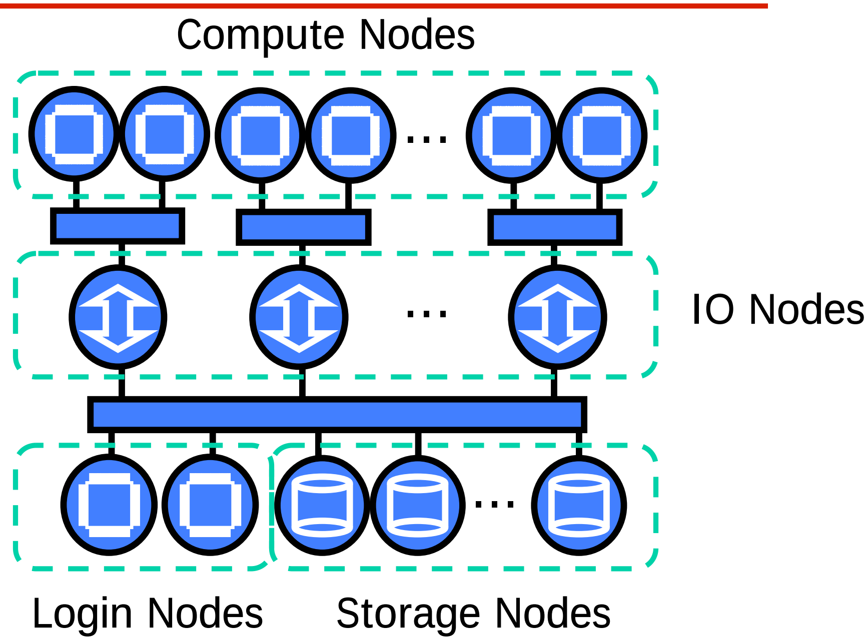
- Parallel I/O
  - PVFS2 (slides courtesy of Rob Ross)
- Scalable Operating Systems
  - Lightweight Linux Kernel
    - Can you leverage OS work for HPC? If not why not? Can you quantify that?
  - Collective system calls
    - What happens when 64k nodes make a system call within 25 ns?
    - Can you use a "system call cache"? For which calls?
    - Should a parallel OS have different system calls?
- Programming models and Algorithms
  - Making use of parallelism in the interconnect

What is PVFS2?

- PVFS2 is an open, collaborative effort
- Core development
  - Argonne National Laboratory
    - Ross, Latham, Gropp, Thakur
    - Supported by DOE Office of Science
  - Clemson University
    - Ligon, Settlemyer
  - Ohio Supercomputer Center
    - Wyckoff, Baer
- Collaborators
  - Northwestern University
    - Choudhary, Ching
  - Ohio State University
    - Panda, Yu
  - Penn State University
    - Sivasubramaniam, Kandemir, Vilayannur

View of I/O on BG/L

- Storage nodes
  - Local access to disks
  - GigE connections to login and IO nodes
- Login nodes
  - Interactive machines
  - Place where data staging will occur
- IO nodes
  - Aggregators for compute node I/O
    - 1:8 to 1:64 ratio of IO nodes to compute nodes
  - Tree connection to compute nodes
- Compute nodes
  - Source/sink of runtime I/O

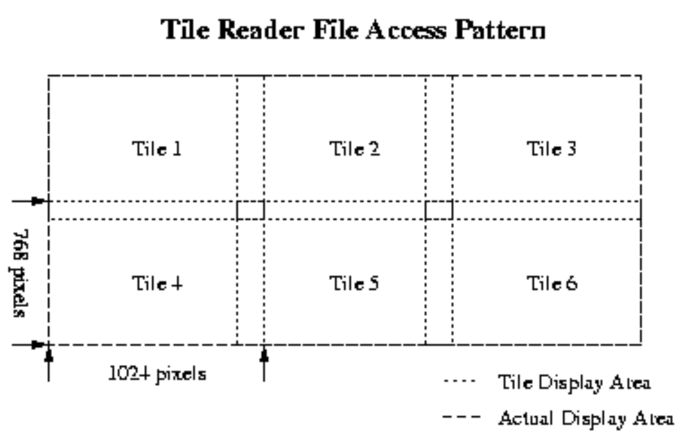


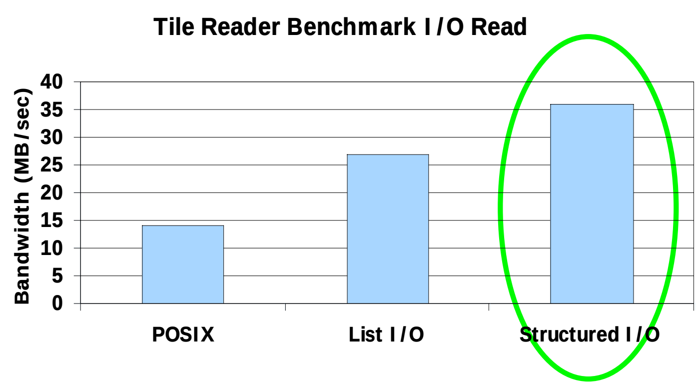
Why put PVFS2 on BG/L?

- PVFS2 addresses three key scalability problems for parallel file systems:
  - I/O performance (especially for noncontiguous data)
  - Metadata performance (in particular open/close)
  - Failure tolerance
- Because of these advantages, we believe that PVFS2 has the best chance of extracting the highest possible I/O performance from BG/L

Scaling effective I/O rates

- POSIX I/O APIs aren't descriptive enough
  - Don't allow us to generally describe noncontiguous regions in both memory and file
- POSIX consistency semantics are too great a burden
  - Stronger than sequential consistency (!)
  - Require too much additional communication and synchronization, not really required by many HPC applications
  - Will never reach peak I/O with POSIX at scale, only penalize the stubborn apps
  - Use more relaxed semantics at the FS layer as the default, build on top of that

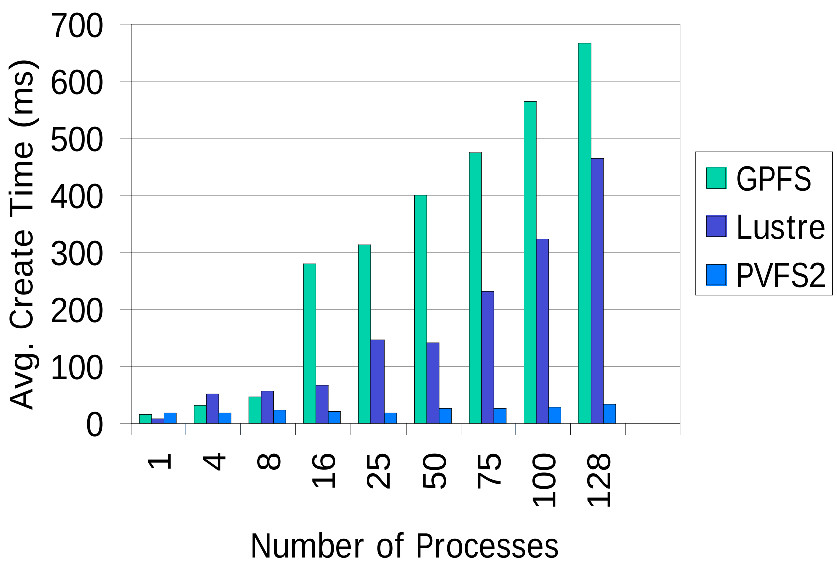


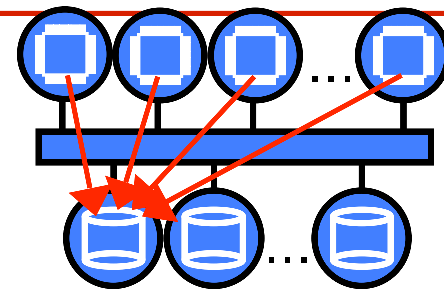


Scaling metadata operations

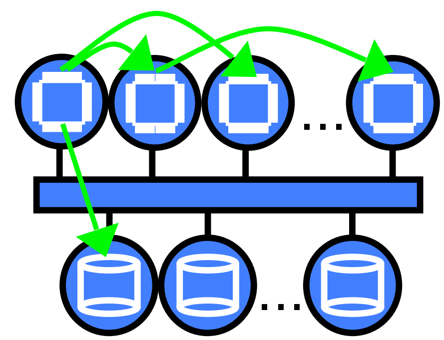
- POSIX API hinders scalability here too
  - POSIX open/close access model imposes constraints on how we implement MPI-IO operations like MPI_File_open
  - Similar issues with fsync and other operations

MPI File Create Performance (small is good)





POSIX file model forces all processes to open a file, causing *system call storm*.



Handle-based model uses a single FS lookup followed by broadcast of handle (implemented in ROMIO/PVFS2).

Tolerating client failures

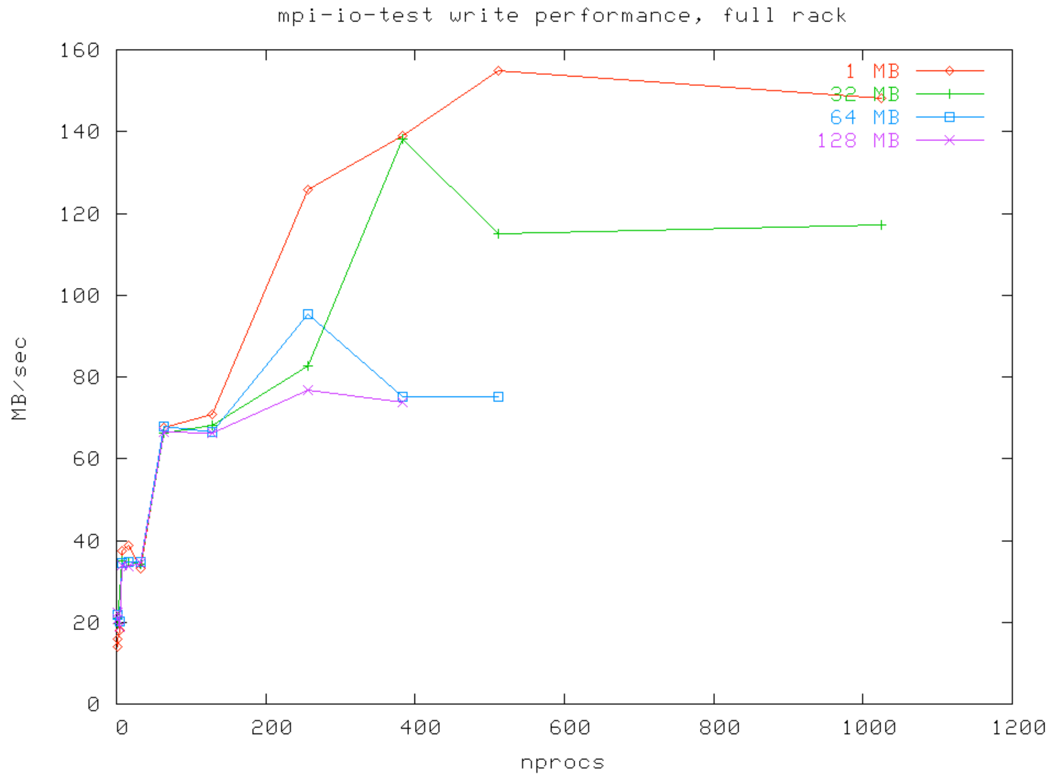
- Client failures are likely to be common with high node counts
  - 99.99% up indicates ~6 nodes down at any time on a 64K node system
  - 99.9% up indicates ~65 down at any time on same
- Unlike other options, PVFS2 uses a **stateless I/O model**
  - No locking system to add complications
  - No other shared data stored necessary for correct operation (no tracking of open files, etc.)
- Client failures can be ignored completely by servers and other clients
  - As opposed to locking systems, where locks and dirty blocks must be recovered!
- Server restarts are easily handled as well

First steps in running PVFS2 on BG/L

- **Goal: Enable data staging and runtime I/O to a PVFS2 file system**
  - Run PVFS2 servers on storage nodes
    - dual Xeon nodes running SLES Linux and 2.6.5 kernel
  - Mount PVFS2 file system on login nodes
    - PowerPC 970 nodes running SLES Linux and 2.6.5 kernel
  - Mount PVFS2 file system on IO nodes
    - BG/L PowerPC nodes running MontaVista 2.4.19 kernel
- **This only took two weeks to accomplish!**
  - Mostly learning/creating build environment
  - Minimal patching to PVFS2 (all in CVS)
  - 12 PVFS2 servers providing a single coherent file system

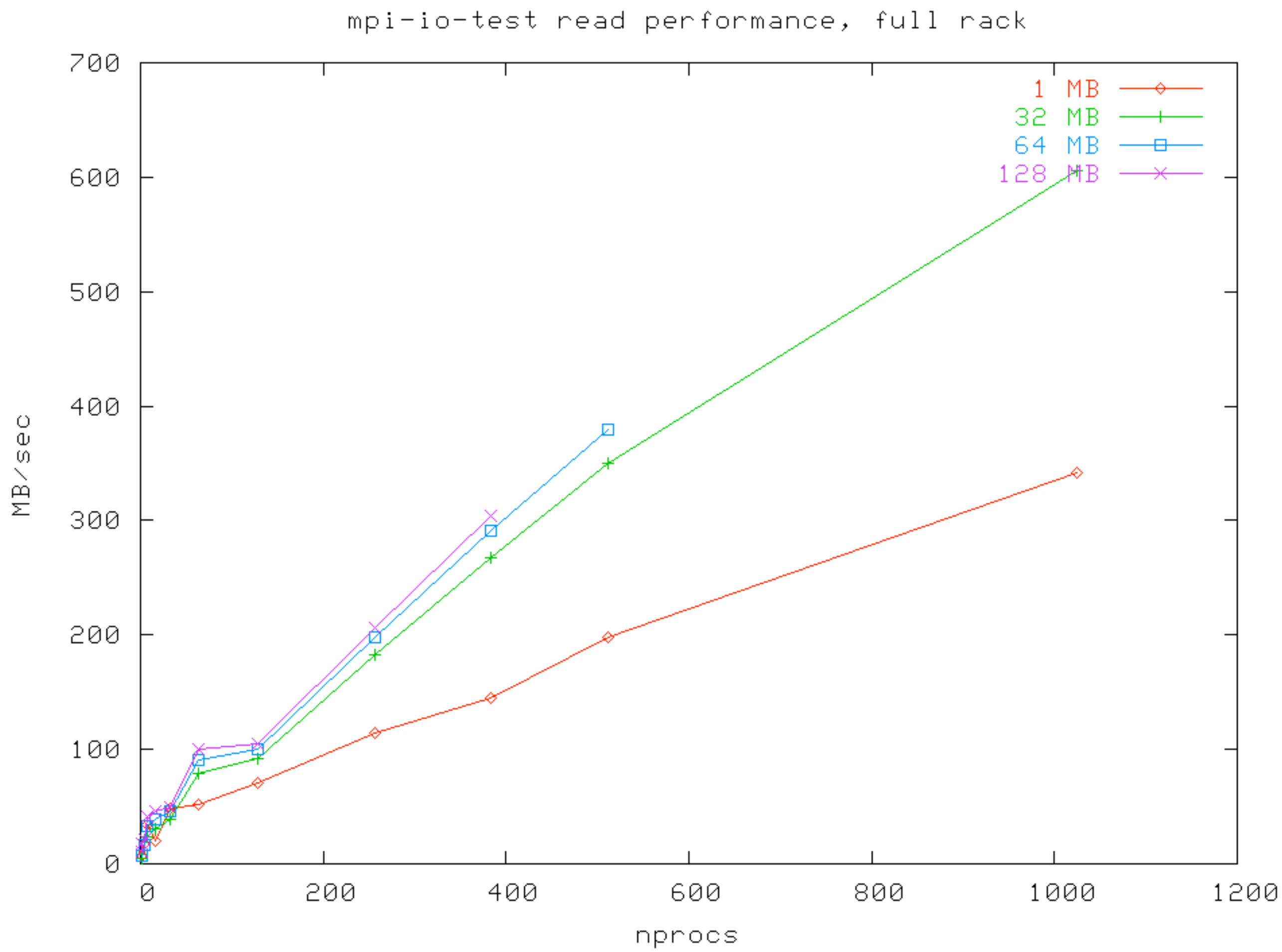
    (Assuming 900mbit/sec network to each, peak of ~1.3GB/sec raw BW)

Write performance (the bad news)



- Clearly we have more work to do here!
- ciod is breaking accesses into 95520 byte blocks (?)

Read performance (the good news)



- Peak of 600MB/sec (44% of raw BW, no tuning)

Beyond base functionality

- Our research indicates that the POSIX interface limits the scalability of I/O systems
  - Noncontiguous read and write performance
  - Open/close problems
- We cannot leverage PVFS2 improvements in these areas if we simply mount PVFS2 file systems
  - We're still going through the VFS
  - The ciod is using POSIX calls
- **To obtain the highest possible performance we must circumvent (or change!) the VFS**
- Two options:
  - Direct compute node to storage server communication
  - **Retool communication between compute and IO nodes and mechanism IO node uses to access file system**

Changing the I/O language

- **Really what we'd like to do is change how compute processes talk to the file system**
  - Ideas prototyped in PVFS2 already
  - Allow for efficient noncontiguous I/O
  - Eliminate open() and close()
- This means changing how compute processes communicate with the IO node
  - Replace or augment existing ciod functionality
  - Map new language to PVFS2, GPFS, Lustre operations
    - **These changes can benefit any underlying file system**
- More efficiently leverage the tree network

Making Use of Parallelism in the Interconnect

We want to use these links simultaneously

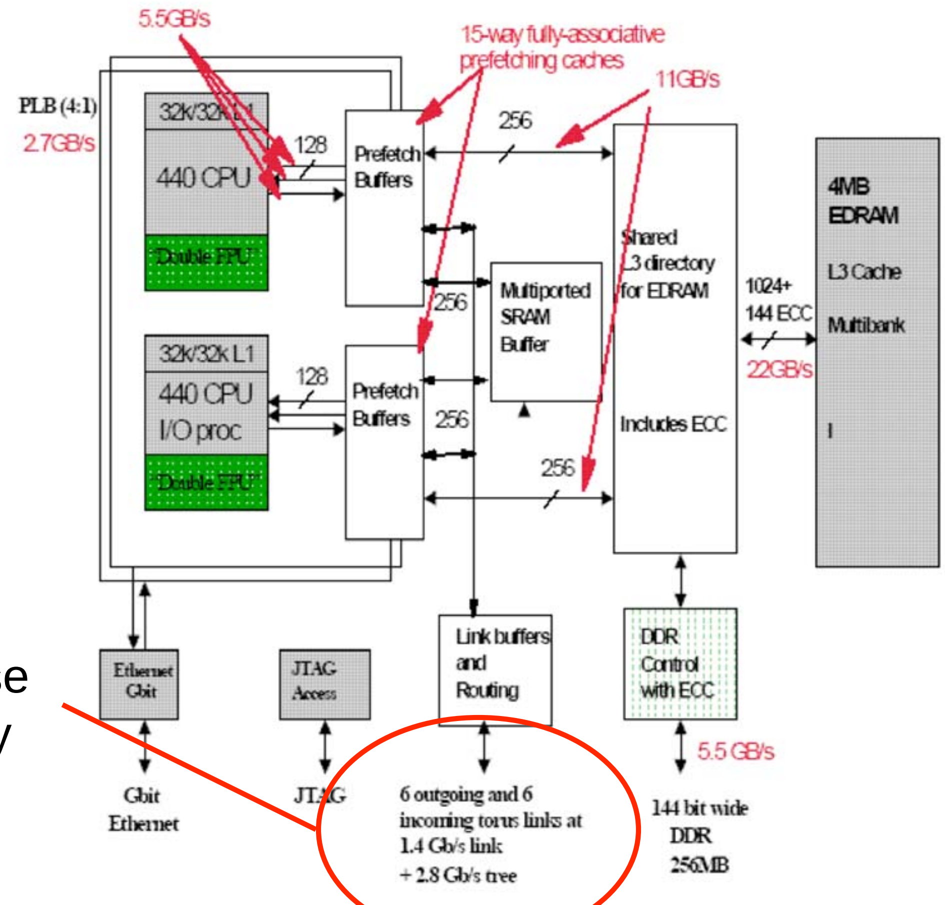
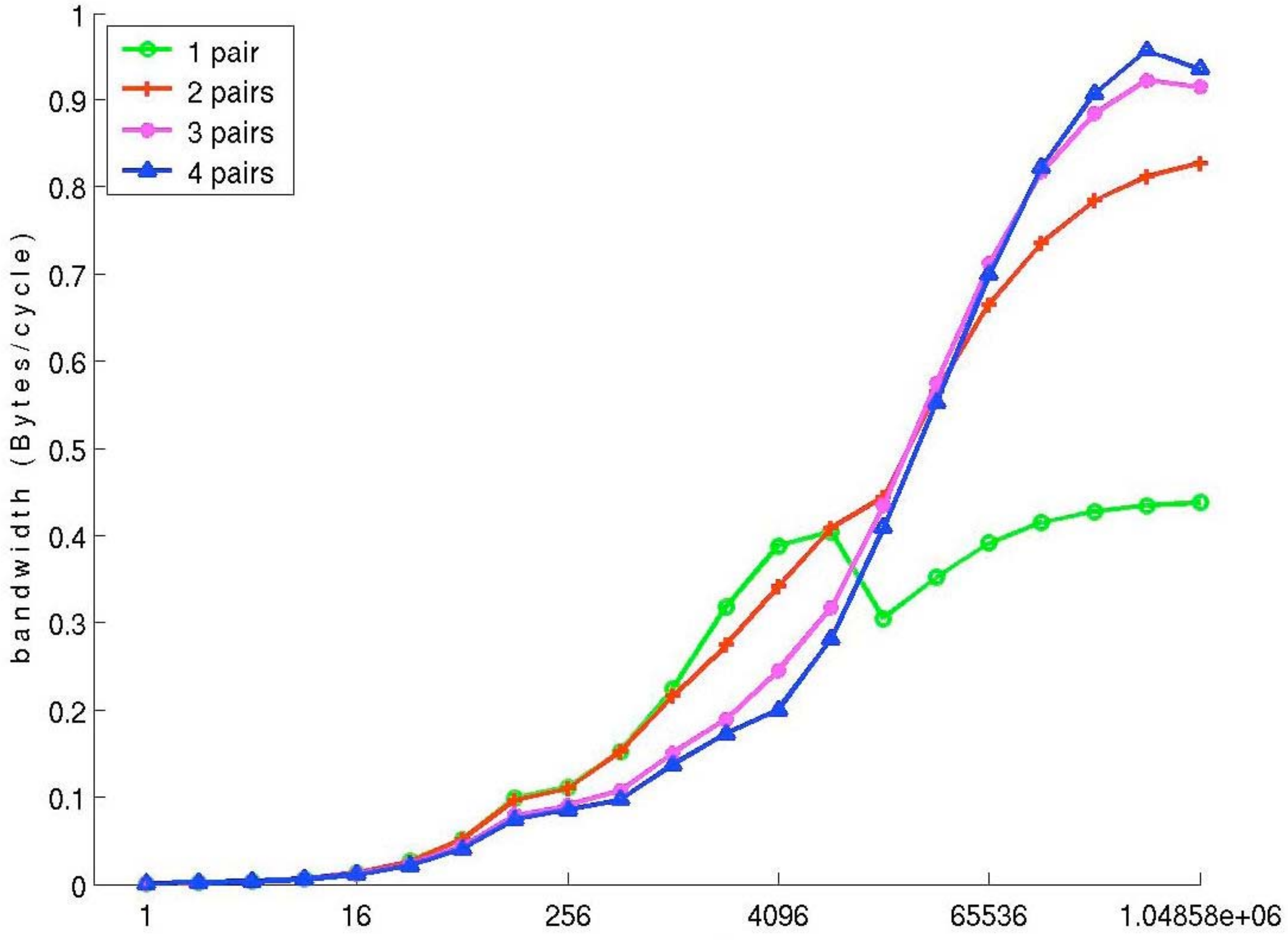


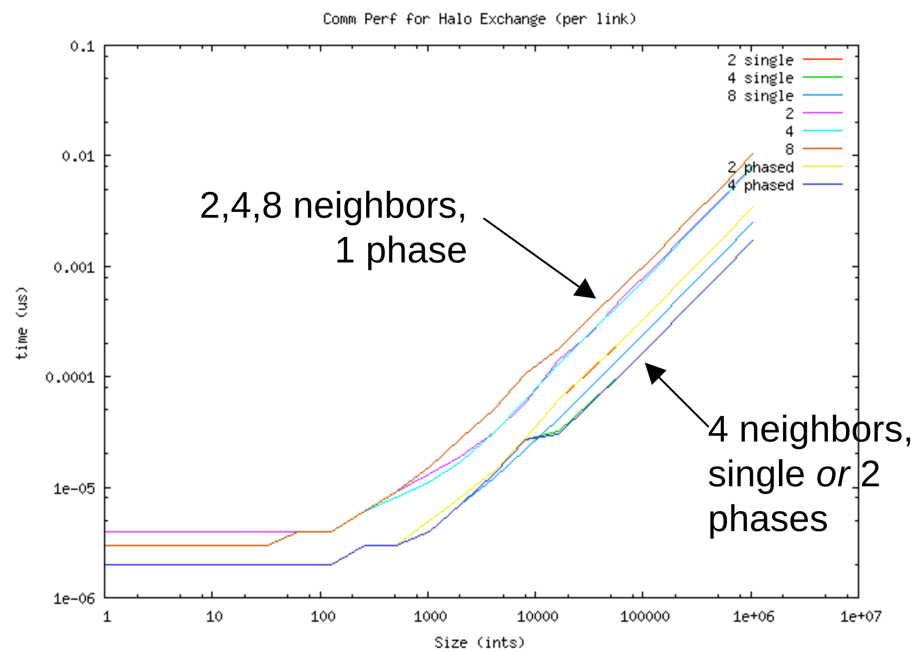
Figure 4: BlueGene/L Node Diagram. The bandwidths listed are targets.

Multiple Communication Paths From a Single Node Can Operate Simultaneously



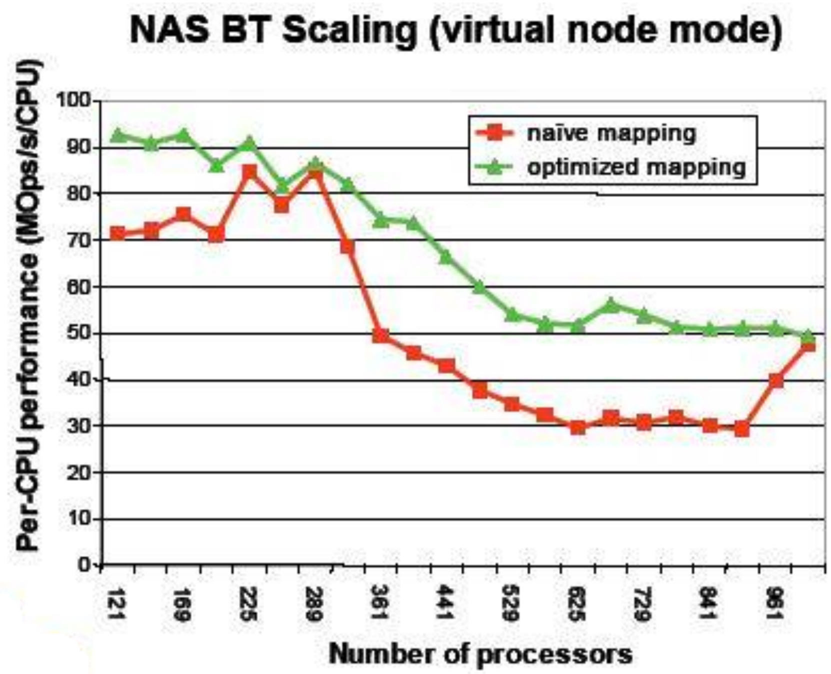
Can Applications Make Use of Multiple Links?

- Typical communication structure does *not* match BG/L needs
- Exploiting global knowledge about communication structure
  - General case — graph coloring (sparse matrix problems)
  - Regular meshes — compile-time information



Performance Sensitive to Layout

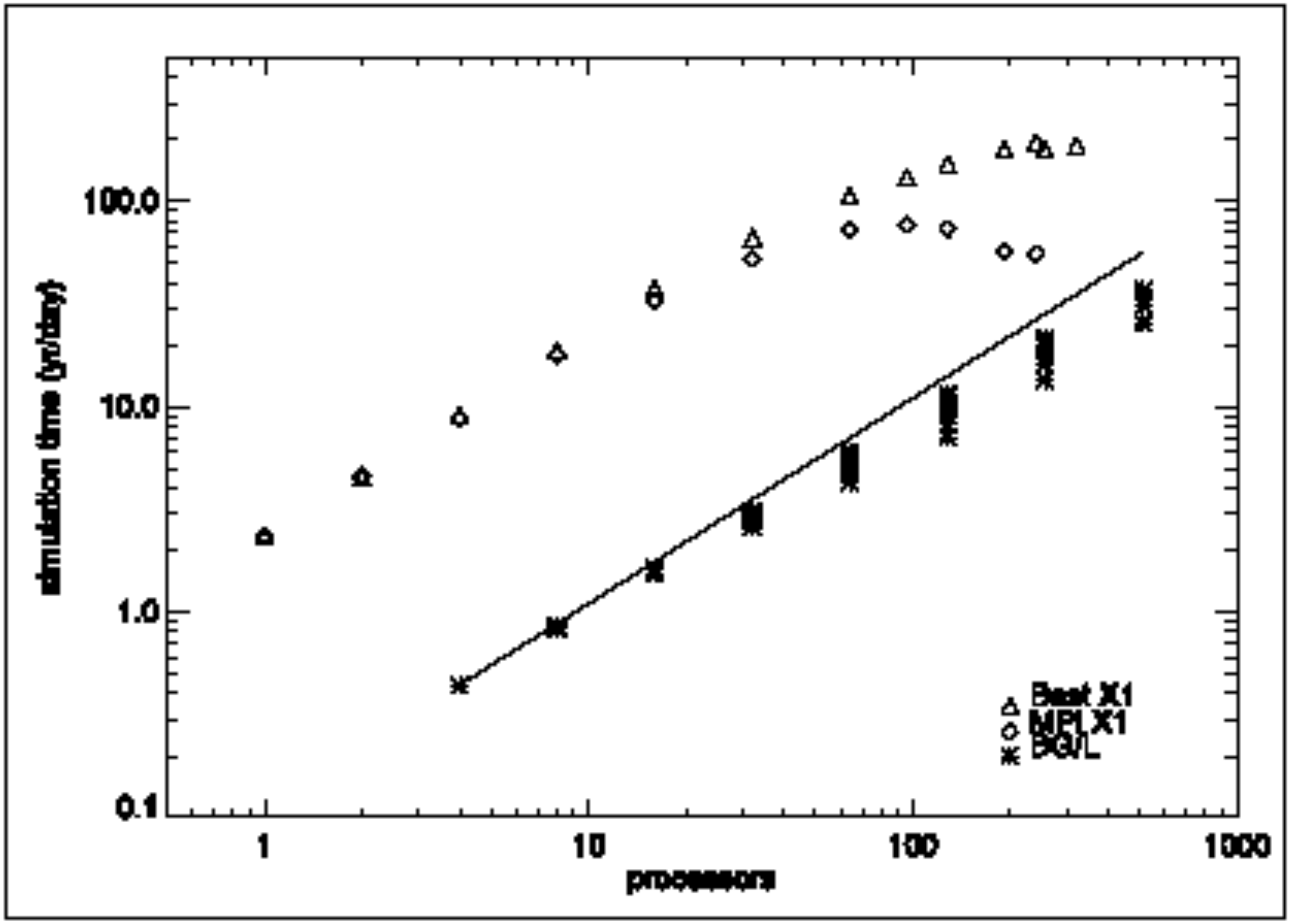
- Once again, process layout and network topology are important
- MPI provides hooks for applications to use
  - MPI_Cart_create
  - MPI_Dims_create
  - MPI_Graph_create
- MPICH2 gives the device access to these using a topology component, similar to the collective component introduced in MPICH in 1995
- Applications should be prepared
  - As should benchmarks!



Current Status

- 1 Rack (1024 nodes)
  - 32 I/O nodes (1/32 IO/Compute ratio)
  - 4 Frontends (JS20 blades – PPC970 2.1GHz dual-cpu 4GB RAM)
  - 1 Service Node (4-way 1.7 Ghz PPC (2 CPU cores), 16GB RAM)
  - 20 Storage servers (4 homedir, 16 PVFS) ~14TB
- Accepted on 1/31/05
  - Very fast (Install started 1/20/05)
  - Machine is solid
  - Software has warts but works
- Friendly user mode
  - Benchmarks
  - Application tests

POP Scalability



BlueGene Consortium

- http://www.mcs.anl.gov/bgconsortium
- Mission: Build a community of BG/L expertise that will foster the rapid scientific adoption of BG/L and develop experience in order to provide critical feedback to the architects and designers of the BG/P follow-on system.
- The specific goals of the consortium are to:
  - Build a critical mass of interest in this new family of systems.
  - Establish mutually supportive reference accounts.
  - Develop scientific and technical applications targeted to the capabilities of BG/L.
  - Port existing tools and open source community codes to BG/L.
  - Develop enhanced systems software and administration tools.
  - Train students and develop next generation user community.
  - Provide feedback to IBM and the BG designers.
  - Provide functional requirements for next generation systems.

Consortium Activities

- Recent Events
  - Semi-regular AG meetings
  - System Software Workshop (Feb 23—24)
- Groups
  - Applications
  - Architecture
  - System Software
  - Outreach
  - Operations
- Membership
  - 5 National Laboratories
  - 15 Universities
  - 4 International

Conclusion

- BlueGene/L represents a major direction change in HPC architecture
  - Not revolutionary, but not just more of the same
- ANL pursuing CS research with BlueGene
  - Goal is to solve CS problems for Petaflops and transpetaflops computing
  - Leading community effort
  - Providing access to our BG/L to consortium members
    - Typically, small amounts of time, but at scale
    - Applications scalability studies encouraged
  - Many opportunities for joint research efforts