Argonne NATIONAL LABORATORY

PVFS2 and Parallel I/O on BG/L

Rob Ross

Mathematics and Computer Science Division

Argonne National Laboratory

Special acknowledgements

- Rob Latham – did most of the work to get PVFS2 up and running
- Susan Coghlan – makes all our lives easier
- Kazutomo Yoshii – figured out how to get things built for the IO nodes at Argonne
- Kamil Iskra – testing of CN to ION performance, ciod hacking
- LLNL group (Robin, Ira, others) – provided us with a great start for building IO node kernels
- IBM – provided source to key components and insight into system components that made this possible

Outline

- PVFS2 introduction and background
  - What it is, who it is, and why it's interesting for BG/L
- Base functionality for PVFS2 on BG/L
  - Status
  - Performance on BGW
- Beyond the baseline
  - Pursuing higher I/O performance
- Wrap up

The PVFS2 Parallel File System

- Parallel file system
  - Distributed data and metadata
  - Tuned for performance and concurrency
- Production ready
  - In use at ANL, OSC, Univ. of Utah CHPC, and others, mainly on clusters
- Open source and open development
  - Free to download and use for anyone
  - LGPL license on all but kernel module, GPL on kernel module
  - Current CVS is anonymously accessible
  - Mailing lists where developers can track and initiate discussions
    - Support at no cost
- Community research vehicle
  - Heterogeneous system support
  - Predominantly user-space code
  - Rapid porting via network and storage abstractions
  - Many labs and universities extend or modify PVFS2 to explore new ideas

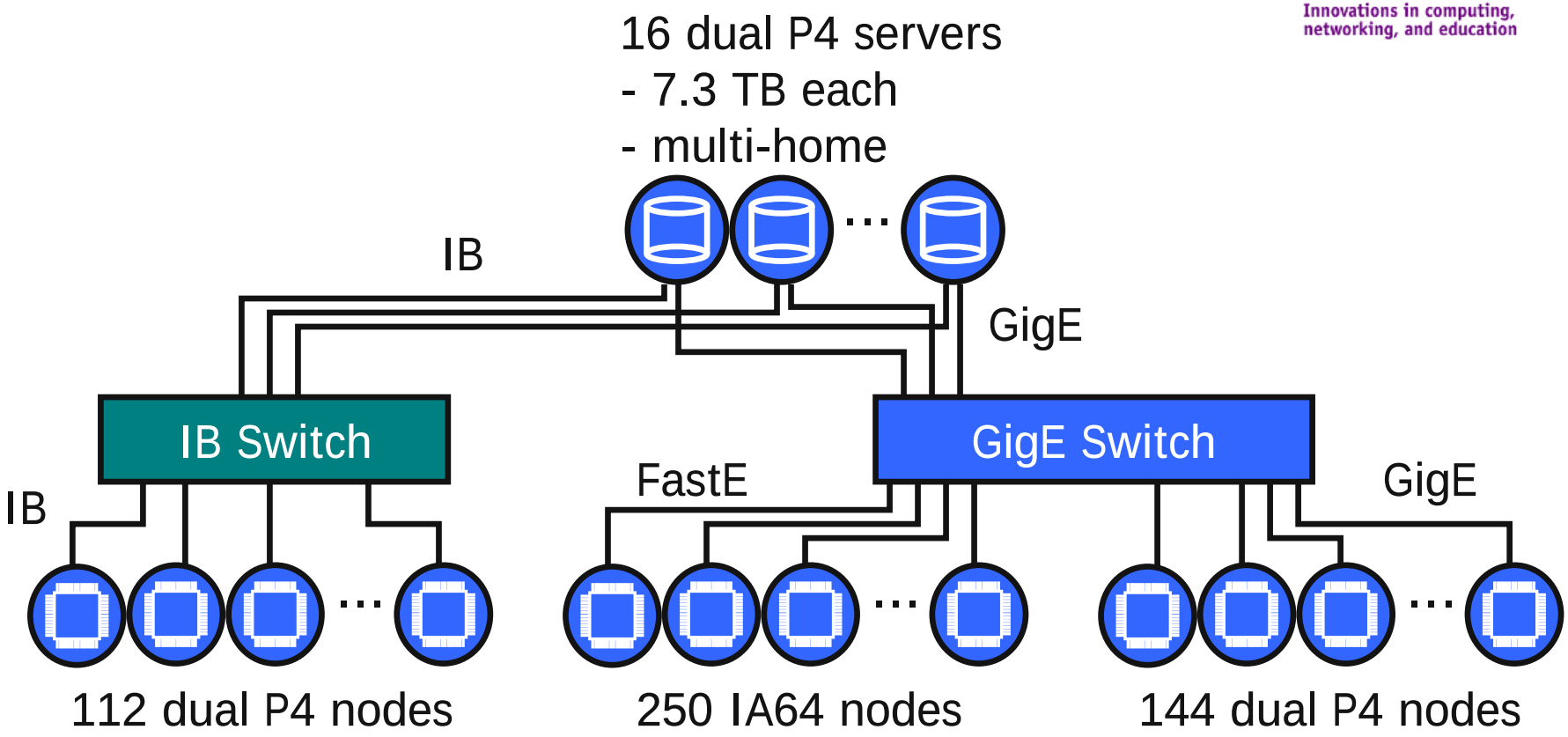
PVFS2 Development and Collaborators

- Colleagues at ANL
  - W. Gropp, R. Thakur, P. Beckman, S. Lang, R. Latham, M. Vilayannur, S. Coghlan, K. Yoshii, and K. Iskra
- Community partners
  - W. Ligon and B. Settlemyer
    Clemson University
  - P. Wyckoff and T. Baer
    Ohio Supercomputer Center
  - P. Carns and D. Metheny
    Acxiom Corporation
  - A. Choudhary
    Northwestern University
  - D.K. Panda
    Ohio State University
  - T. Ludwig and J. Kunkel
    University of Heidelberg
  - P. Honeyman and D. Hildebrand
    University of Michigan

PVFS2 at OSC

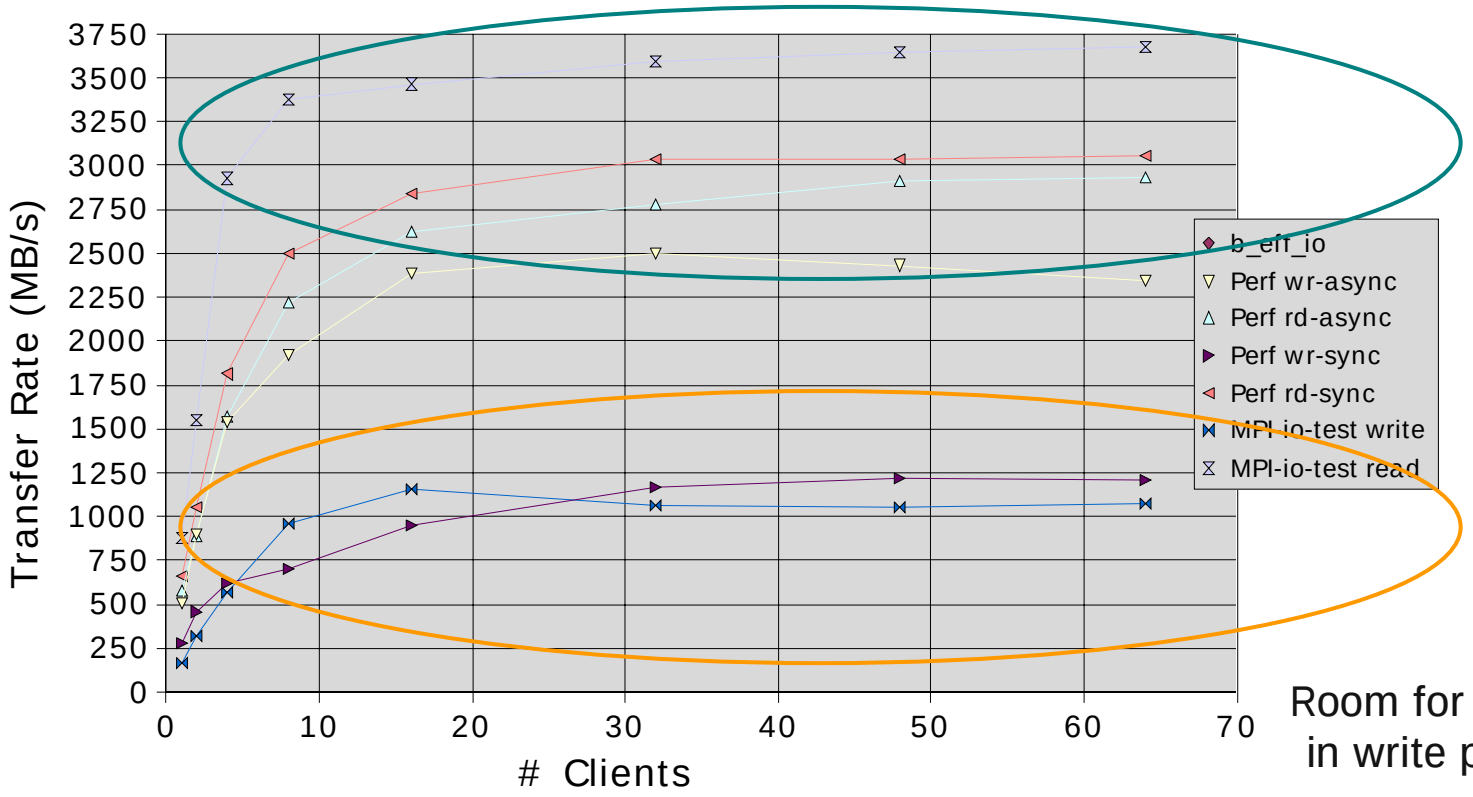
- 506 total clients
- 116.8 TByte file system

OSC
Innovations in computing, networking, and education



PVFS2 Performance at OSC

- 16 I/O servers with 7.3TBytes of storage each
- InfiniBand interconnect



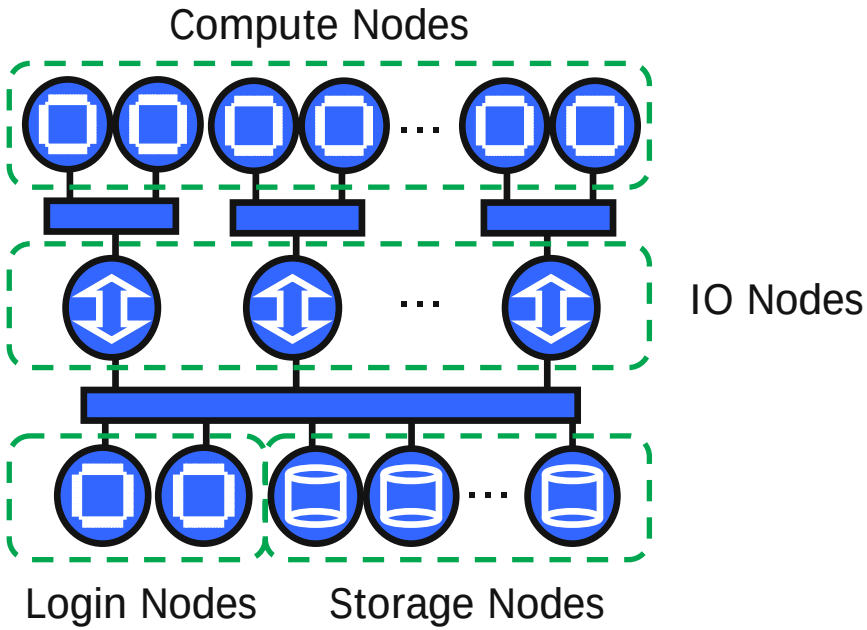
3GB/sec Reads

1GB/sec Writes

Room for improvement in write performance.

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

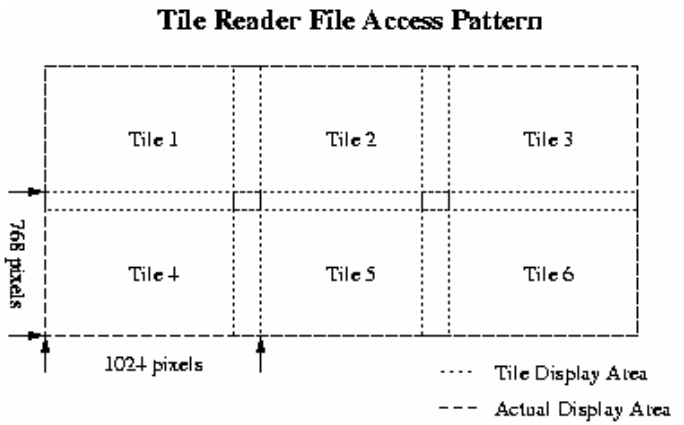


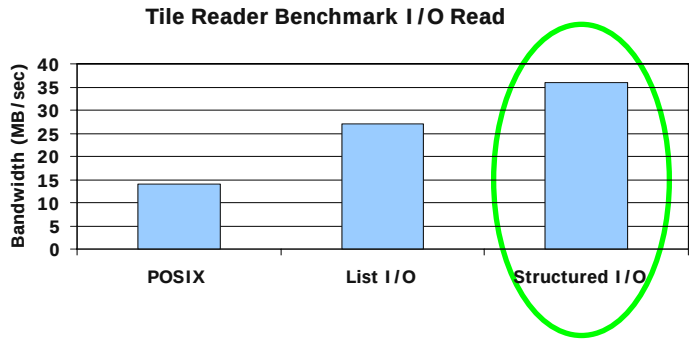
Why put PVFS2 on BG/L?

- Free parallel file system alternative for BG/L
  - Supported on wide variety of server hardware
  - BG/L clients can co-exist with clusters
- Provides another data point for I/O performance
- Most importantly, 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
  - Talk about these three issues next…

Scaling Effective I/O Rates

- POSIX I/O APIs aren't descriptive enough and consistency semantics are too great a burden
  - Don't allow us to generally describe noncontiguous regions in both memory and file
  - Require too much additional communication and synchronization, not really required by many HPC applications
- PVFS2 relaxes I/O semantics and provides a more rich API for describing I/O accesses
  - Can achieve better performance for real application patterns

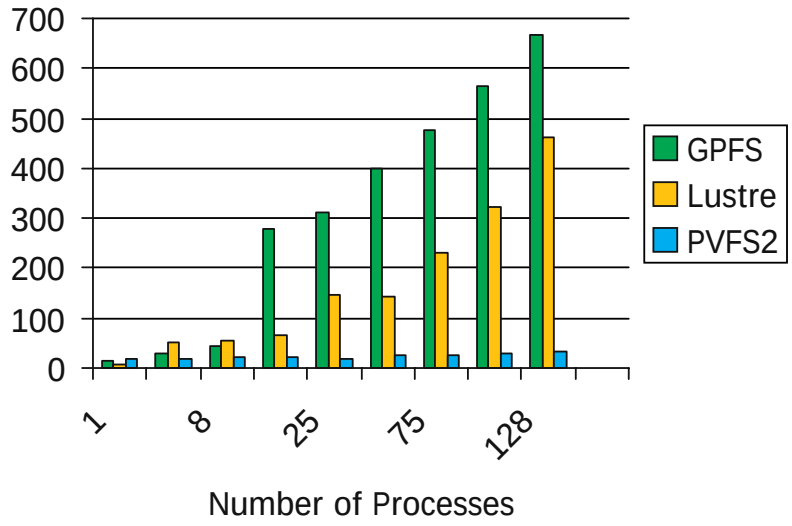


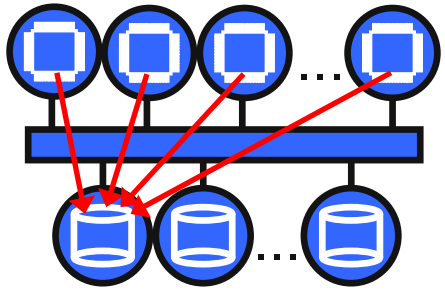


Scaling Metadata Operations

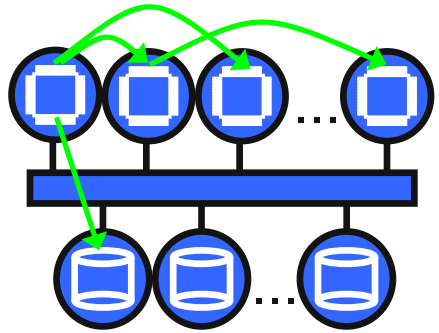
- POSIX API hinders metadata scalability too
  - Access model limits how we implement MPI-IO operations like MPI_File_open
  - Similar issues with fsync and other operations

MPI File Create Performance (small is good)





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



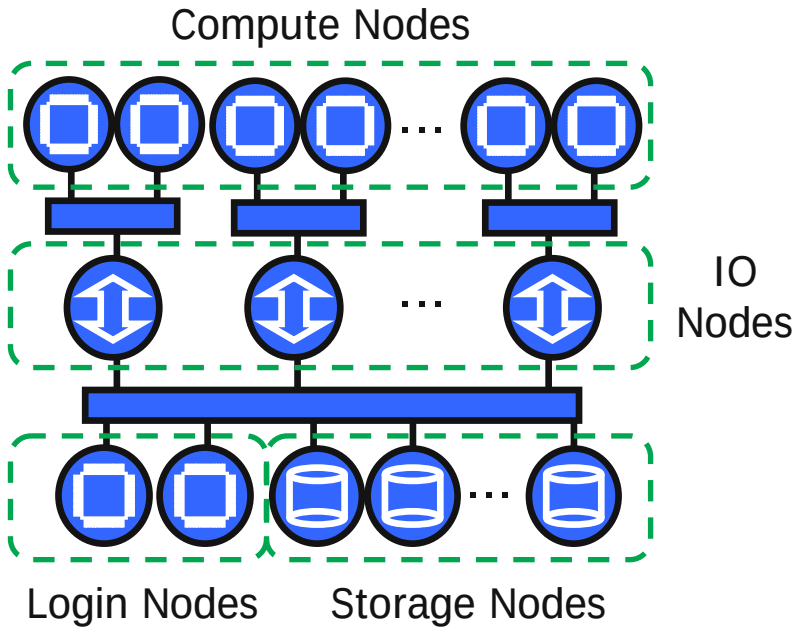
Handle-based model uses a single FS lookup followed by broadcast of handle (implemented in ROMIO/PVFS2).

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
  - BG/L hardware is reliable, but failures will occur at scale
- 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

Status of PVFS2 on BG/L

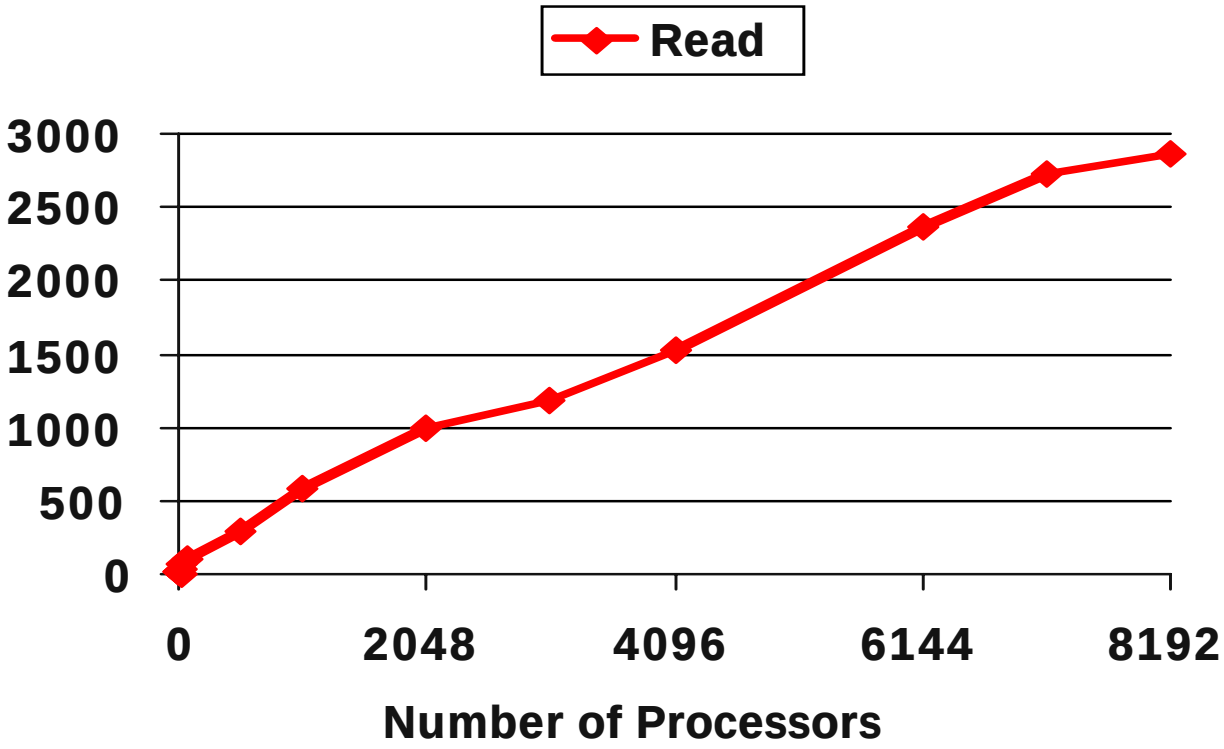
- **Data staging and runtime I/O to a PVFS2 file system all operational**
  - Run PVFS2 servers on storage nodes
    - *dual Xeon nodes running SLES Linux*
  - Mount PVFS2 file system on login nodes
    - *PowerPC 970 nodes running SLES Linux*
  - Mount PVFS2 file system on IO nodes
    - *BG/L PowerPC nodes running Linux*
- Available in ZeptoOS releases
  - Pete Beckman will discuss this in the next session



Example PVFS2 Installation: BGW at IBM Watson

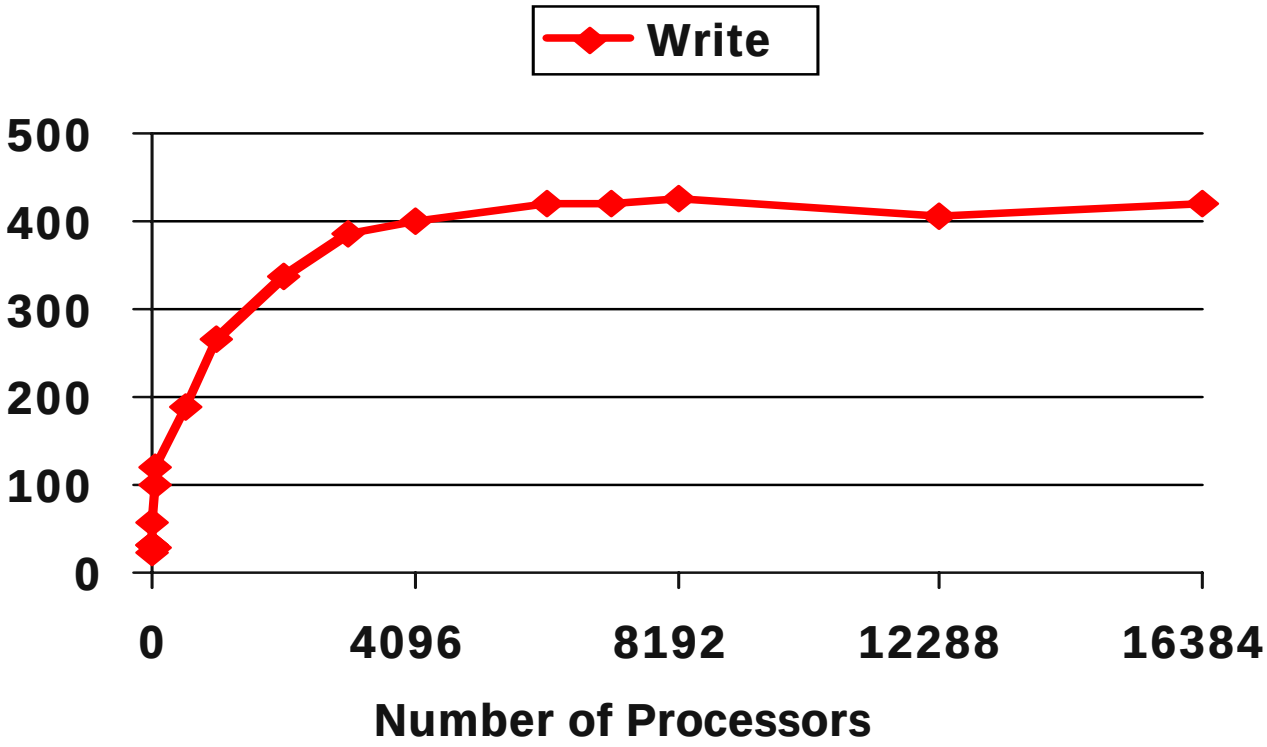
- IBM nicely allowed us to run on their large BG/L machine for a couple of days
- BGW: 16 rack BG/L system
- 64 compute nodes per IO node
- 33 PVFS2 servers
  - 86 MB/sec per PVFS2 server
  - 2.8GB/sec peak possible

BGW Read Performance, Up to 8K nodes



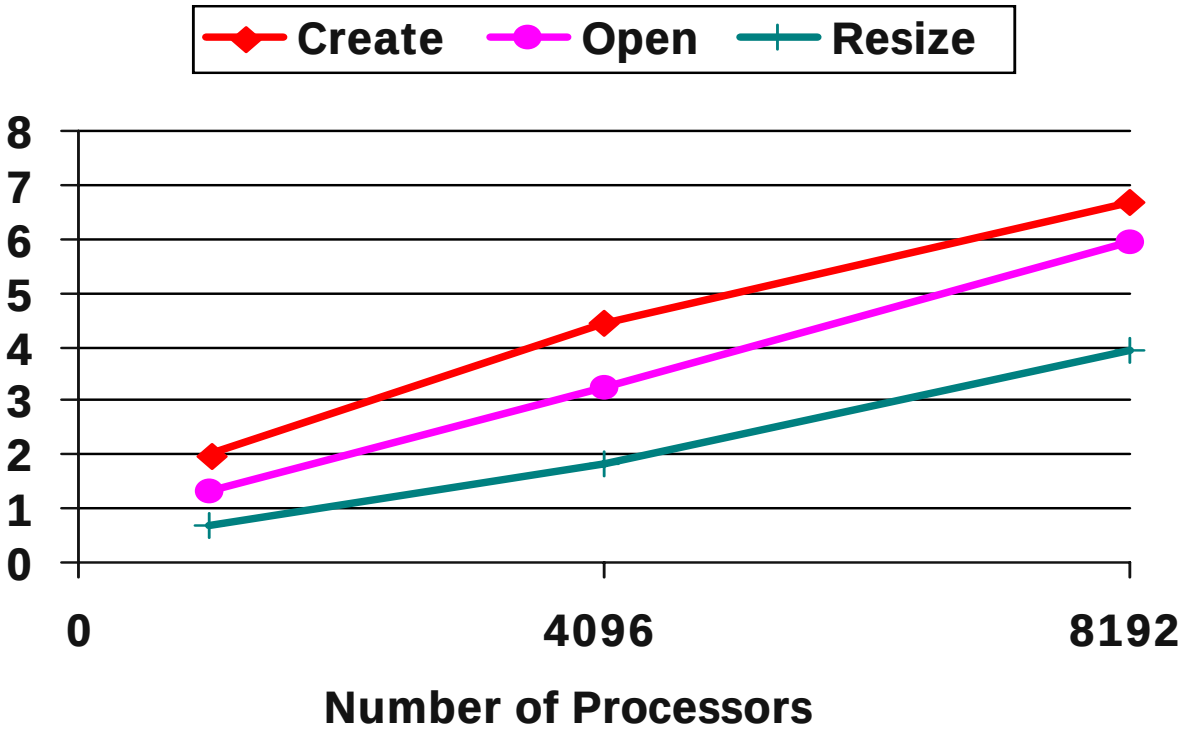
- Performance was strange for 16K configuration (8K run dropped to ~500MB/sec!)

BGW Write Performance



- Synchronization forced after write
- ciod serialization impacts performance
- ciod effects much larger than job placement (as opposed to reads)
- PVFS2 needs tuning also

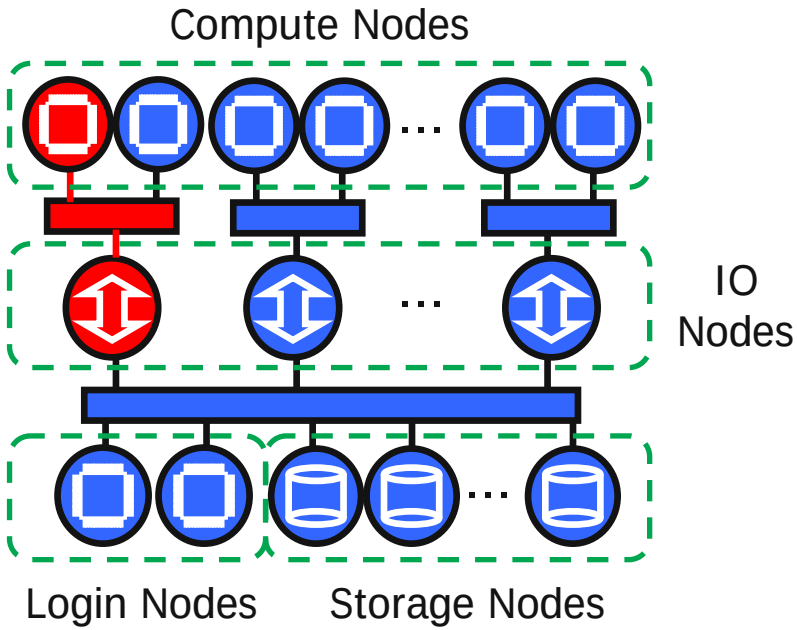
BGW MPI-IO metadata



- Slower than we would like
- BG/L MPI-IO does not implement collective metadata optimizations…

Improving the I/O Infrastructure

- We'd like to change how compute processes talk to the file system
  - Eliminate open() and close() scalability issues
  - Improve noncontiguous I/O
- **Must change 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
- **ZoidFS effort will replace ciod**



Wrap up

- PVFS2 is up, running, and in use on BG/L systems
  - Open source operating systems played a key role in rapid deployment
  - Very positive experience!
- IBM developers have been very helpful
  - Will aid greatly in MPI-IO research and tuning for BG/L
- This is turning into an ideal platform for testing and deployment of next-generation I/O systems!
- High level libraries will follow as well
- We could use just a little more source… ☺

Additional Information on PVFS2

- PVFS2 web site: http://www.pvfs.org/pvfs2
  - Documentation, mailing list archives, and downloads
- PVFS2 mailing lists (see web site)
  - Separate users and developers lists
  - Please use these for general questions and discussion!
- Email
  - Rob Ross <rross@mcs.anl.gov>
  - Rob Latham <robl@mcs.anl.gov>
- See BG/L Consortium Newsletter for more information of PVFS2 and other parallel file systems on BG/L