Argonne NATIONAL LABORATORY

ZeptoOS

Pete Beckman, Susan Coghlan,

Kamil Iskra, Kazutomo Yoshii

Univ of Oregon: Al Malony, Sameer Shende

THE UNIVERSITY OF CHICAGO

Office of Science U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory managed by The University of Chicago

ANL Visitors

- Pete Beckman, Radix Lab
- Narayan Desai, Radix Lab
- Bill Gropp, Radix Lab
- Rob Ross, Radix Lab
- Rajeev Thakur, Radix Lab
- Kazutomo Yoshi, Radix Lab

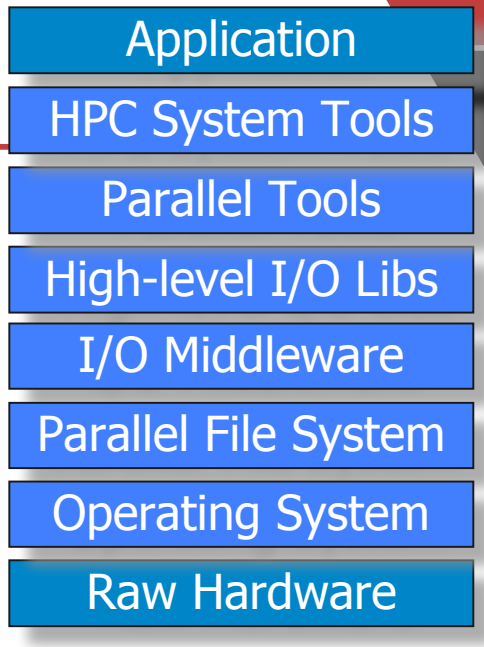
- (could not attend) Andrew Siegel, BG/L Applications

- Susan Coghlan, Head HPC Team

- Remy Evard, ANL CIO

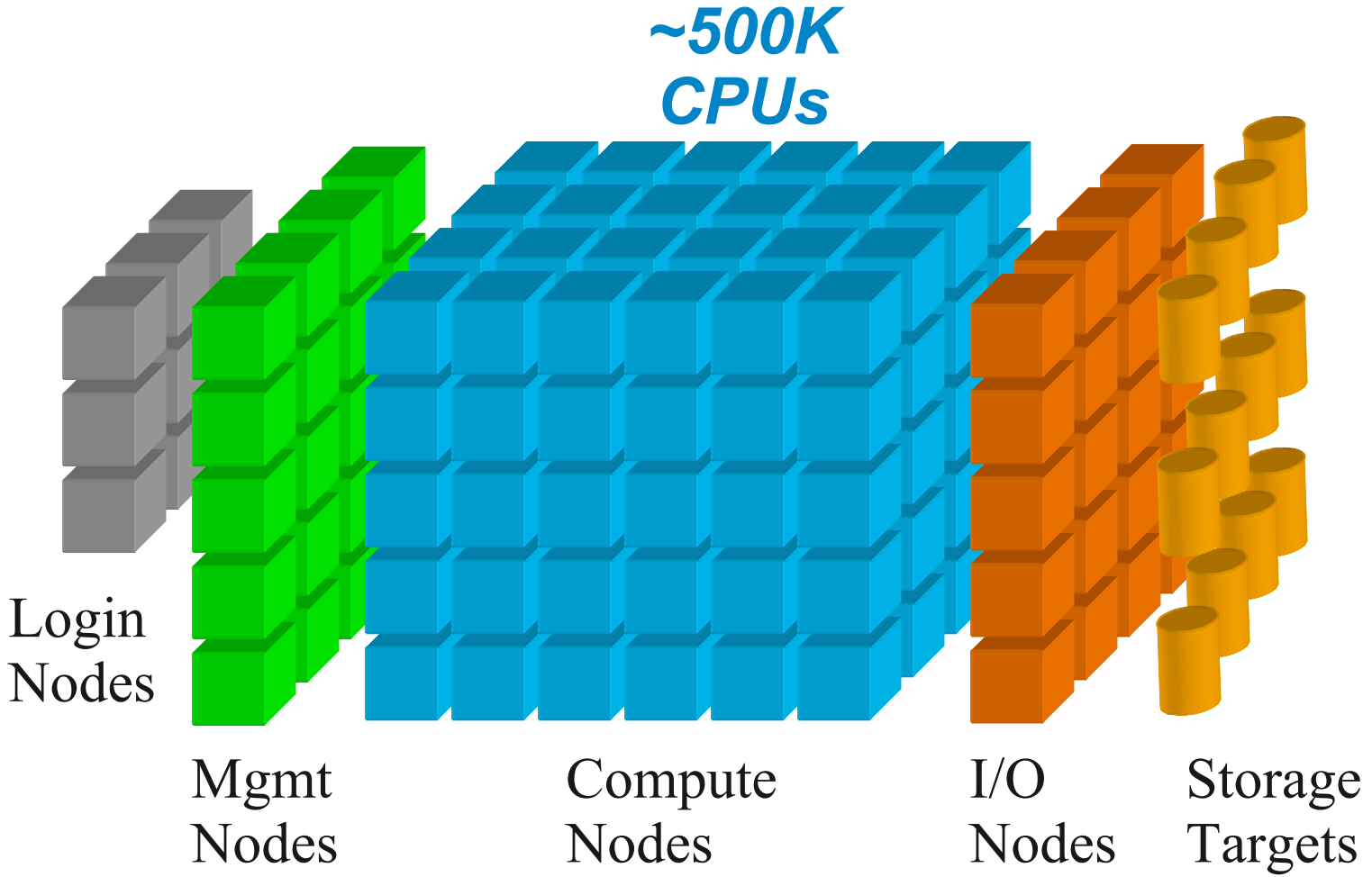
- Ed Jedlicka, ANL BG Consortium

Quick Introduction: Radix Lab



- Argonne's System Software research laboratory for scalable systems
- Senior Staff:
  - Pete Beckman, Bill Gropp, Rusty Lusk, Rob Ross, Rajeev Thakur
  - Other Staff: 5 Postdocs, ~9 Students, ~5 Programmers
- Projects:
  - MPICH2, FPMPI, ROMIO
  - ZeptoOS, Cobalt
  - PVFS2, pNetCDF
  - Jumpshot, Programming Models
- Motto:
  - "Research Software" does not mean broken, impossible to install prototype, it is software that is both practical and a vehicle for CS research

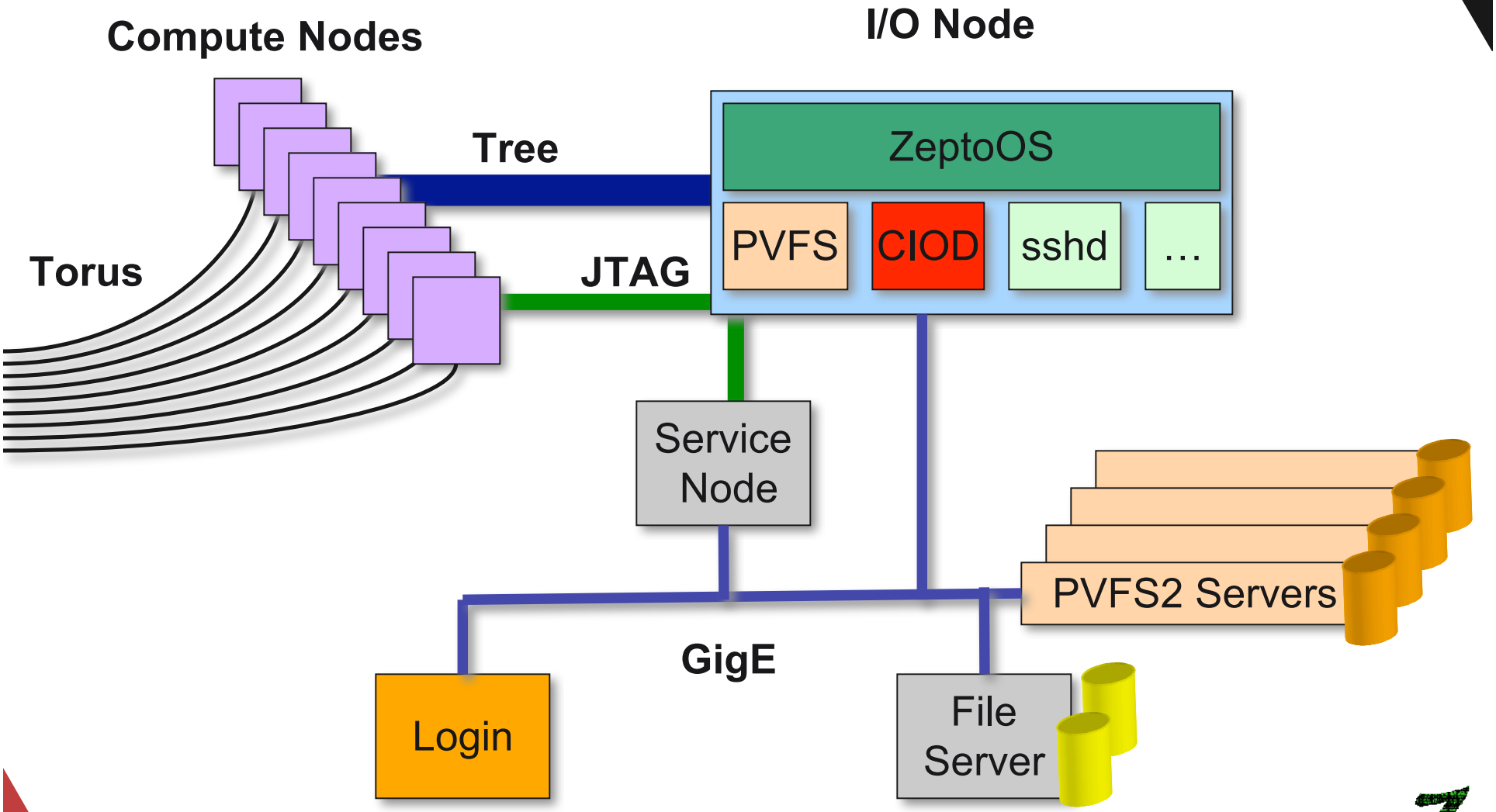
World View: The Road to Petaflops & Beyond
OS Suites, Heirarchy, and Functional Decomposition



ZeptoOS: The Small Linux for Big Computers

- Research Exploration:
  - What are the fundamental limits and advanced designs required for petascale Operating System Suites?
    - *Ultra scale behavior*
    - *Managing & optimizing OS suites*
    - *Collective OS behavior*
    - *Fault tolerance*
    - *How can we gather and study performance data from OS Suites on very large machines?*
- Strategy:
  - Build modified Linux for BG/L I/O nodes
    - *Study*
  - Build modified Linux for BG/L compute nodes
    - *Study*
  - Build specialized I/O daemon (ZOID)
    - *Study*

ZeptoOS + PVFS2 on BG/L



Some ZeptoOS Features

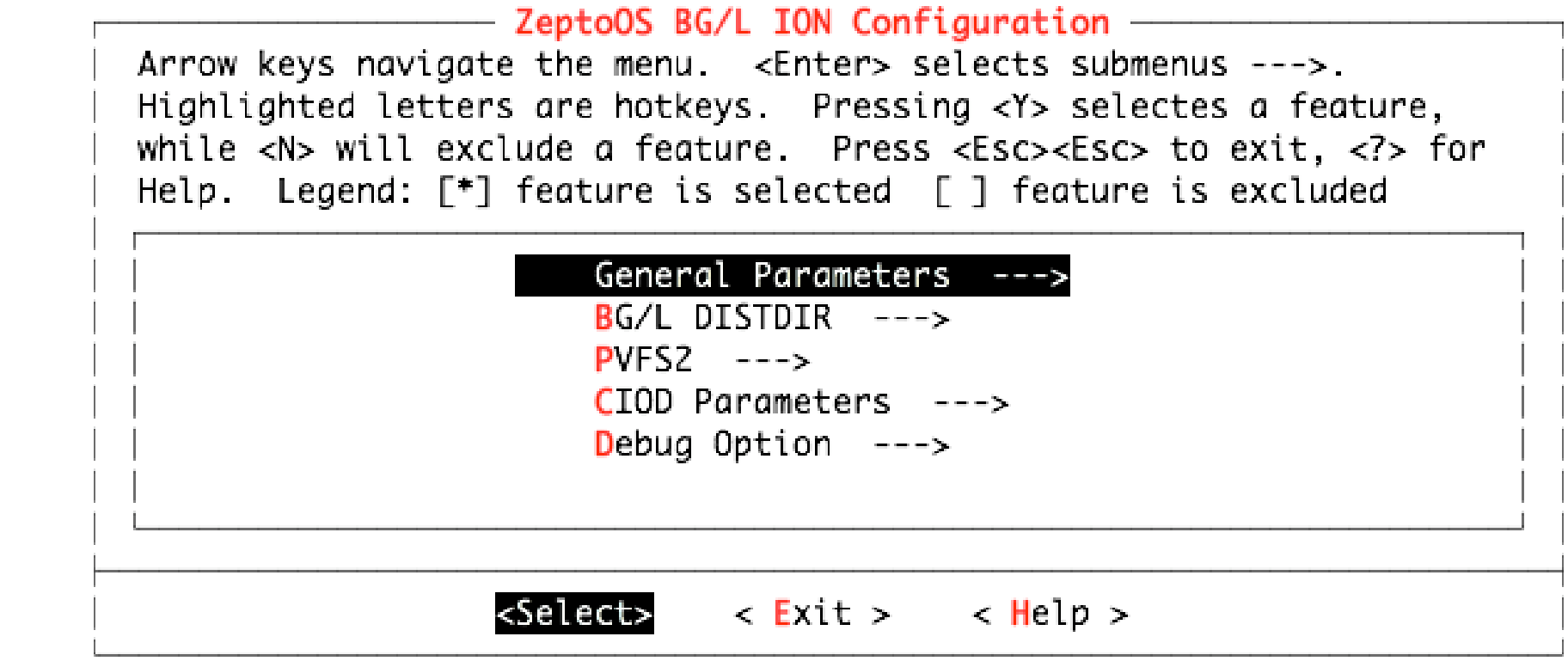
- SSH into your ZeptoOS I/O Nodes
  - for developers and non-root access
- Built-in PVFS2 Support
- Unified Build Environment and Compiler Tool Chain
- Smaller Ramdisk & Faster Partition Booting
- Cross-compilation Support
- Simple Toolchain & Ramdisk Creation
- Additional Utilities (strace, lsof, benchmarks, etc)
- Kernel Parameter Passing Supported
- Easier Kernel Logging and Debuging
- Dynamic Status Information with zinfo
- Customizable Boot Scripts

Let's take a quick tour…

Configuring ZeptoOS: ./configure ; make ; make install

```
$ ./configure
Configuring ZeptoOS Version 1.4.1
*****************************************

 Create root password for I/O Node
   Leave the password field empty if you want to disable
    root login
   New password:
   Retype new password:
*****************************************

$ make install
```

A Complete Build and Configuration System



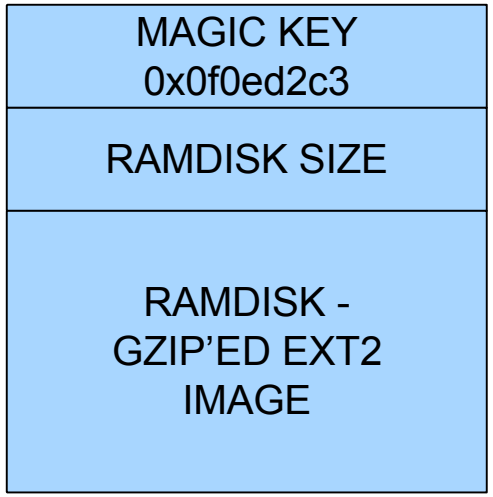
Kernel Parameter Passing

- The original ION kernel:
  - Has a compile-time option for kernel parameter, which requires kernel re-compilation.
- Kernel parameter passing with ZeptoOS
  - Does not require kernel recompilation
  - Extends ramdisk image format and kernel boot loader
- Zkparam-elfrd: a tool for changing/extracting kernel parameters

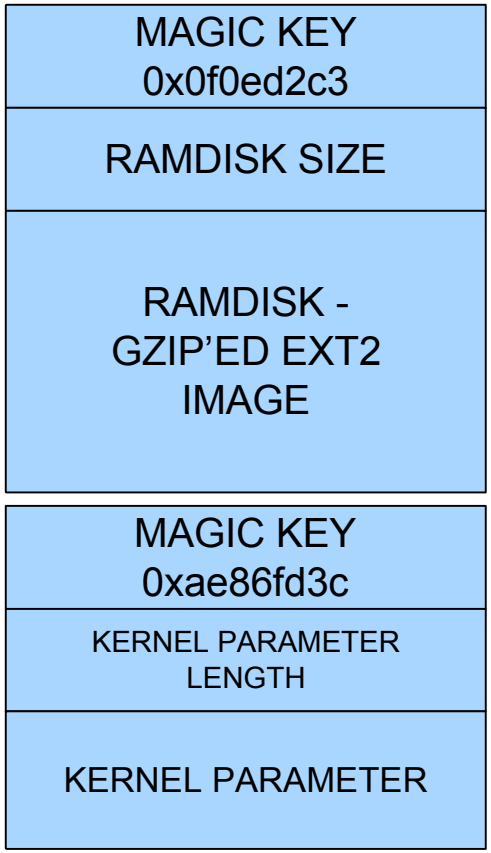
```
$ zkparam-elfrd replace ramdisk.elf zkernel_log=on new_opt=123

$ zkparam-elfrd print ramdisk.elf
 zkernel_log=on new_opt=123

$ zkparam-elfrd add ramdisk.elf new_opt2=on
```

Extended the RAMDISK format for kernel parameter passing

Original format



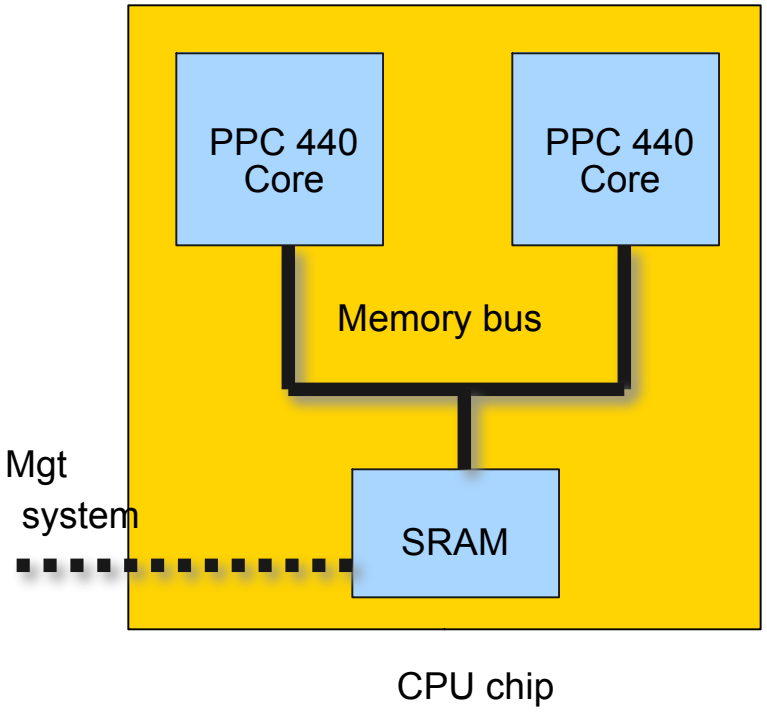
ZeptoOS extended format



Debugging tools

- Remote log-in very useful for debugging / monitoring ION
  - SSH (recommended method)
  - TELNET (debugging before /bgl is mounted)
- Handy tools : strace, lsof, etc.
- Debug message via UDP broadcast
  - convenient for early stage debugging
- RAS Messages to Service Node
- Compute node extension to send RAS messages from user-space WITHOUT passing through CIOD

MBOX Interface



- BG/L uses MBOX for
  - RAS
  - console,kernel log
- MBOX is a memory mapped device
- Zepto provides /proc/zepto/mbox as a useful debugging interface

Logging

- /proc/zepto/mbox forwards message to BG/L log file

/bgl/BlueLight/logs/BGL/R00-M0-N1-I:J18-U01.log
Thu May 19 14:36:40 2005 [1964038] kazutomo:R000_J104-32 {0}.0: loaded at: 00800000 00897288
Thu May 19 14:36:40 2005 [1964038] kazutomo:R000_J104-32 {0}.0: zimage at: 008057D4 00893808
Thu May 19 14:36:40 2005 [1964038] kazutomo:R000_J104-32 {0}.0: avail ram: 00400000 00800000
Thu May 19 14:36:40 2005 [1964038] kazutomo:R000_J104-32 {0}.0:
Thu May 19 14:36:40 2005 [1964038] kazutomo:R000_J104-32 {0}.0: Linux/PPC load: root=/dev/ram console=bgl bglverbose
Thu May 19 14:36:41 2005 [1964038] kazutomo:R000_J104-32 {0}.0: Uncompressing Linux...done.
Thu May 19 14:36:41 2005 [1964038] kazutomo:R000_J104-32 {0}.0: Copying personality
Thu May 19 14:36:41 2005 [1964038] kazutomo:R000_J104-32 {0}.0: Now booting the kernel
Thu May 19 14:36:41 2005 [1964038] kazutomo:R000_J104-32 {0}.0: On node 0 totalpages: 131072
Thu May 19 14:36:42 2005 [1964038] kazutomo:R000_J104-32 {0}.0: zone(0): 4096 pages.
Thu May 19 14:36:42 2005 [1964038] kazutomo:R000_J104-32 {0}.0: zone(1): 126976 pages.
Thu May 19 14:36:42 2005 [1964038] kazutomo:R000_J104-32 {0}.0: zone(2): 0 pages.

$ echo "Hey! I'm Zepto INK" > /proc/zepto/mbox

/bgl/BlueLight/logs/BGL/R00-M0-N4-I:J18-U01.log:

Oct 04 03:34:31[16336903] kazutomo:R000_J111-32 {0}.1: Hey! I'm Zepto INK

Comprehensive Development Environment

- Unified Build System
  - Original IBM tool chain
    - *Compile kernel and modules with modified GCC 3.2*
    - *Ramdisk tools compiles with GCC 3.2*
  - ZeptoOS tool chain unifies compiler to GCC 3.3.4
    - *Single build env. for kernel, modules, ramdisk & NFS-mounted utilities*
- 'Diff' tool for ramdisks (version comparison)
  - compare file attribute such as timestamp,size,etc
  - find newly added files or deleted files
- Extract a ramdisk image
  - BG/L ramdisk format is ELF
  - Extract gzip'ed ext2 image from ELF data

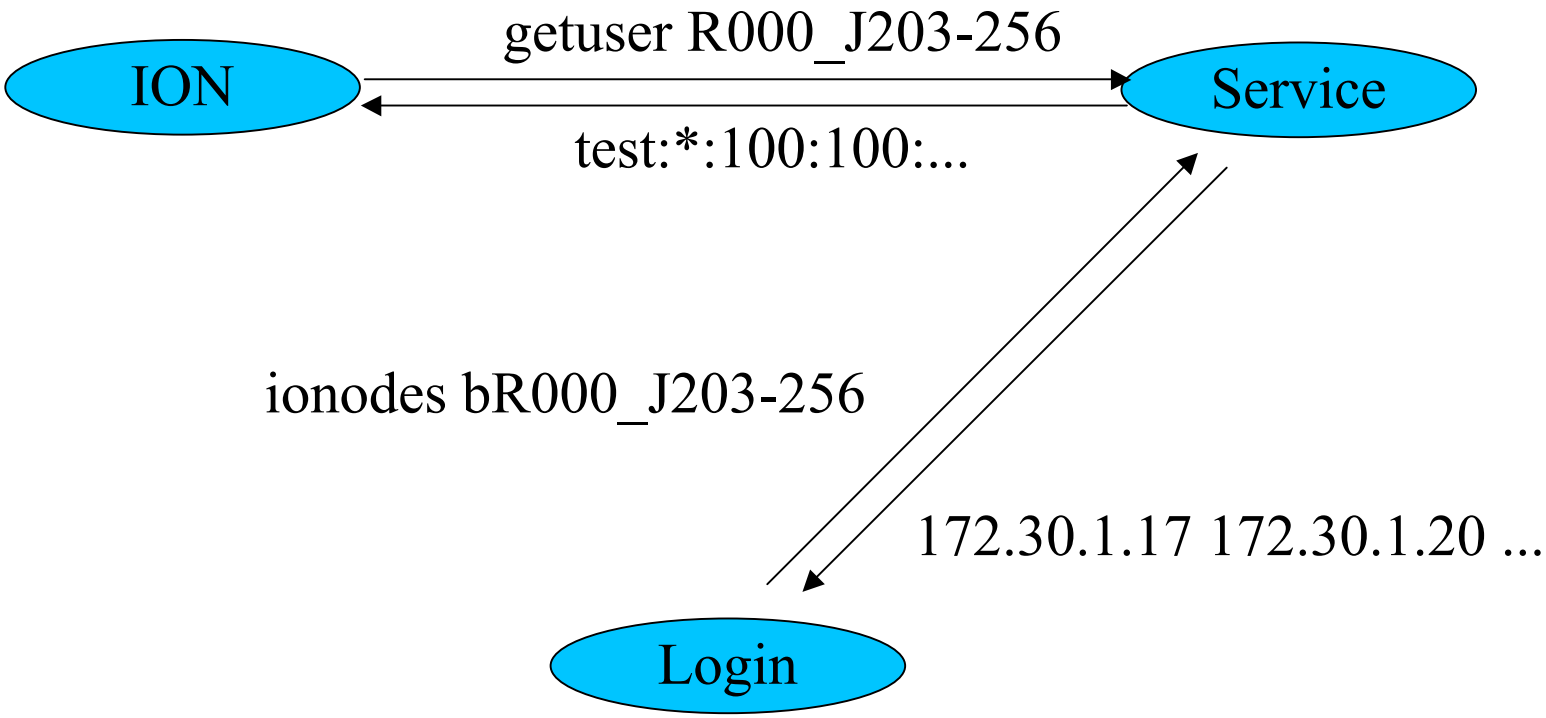
I/O Node NFS mounting

- I/O Nodes mount /bgl via NFS
  - NFS normally exported/mounted read/write with full root privileges!
  - (exporting /bgl read-only to IONs works for us)
- Startup scripts on ION can then launch customized programs
- Sshd:
  - Password authentication for root only by default.
  - How to provide access to ordinary users… the partition owner?
    - *See Zinfo*
  - Authentication for other users added at run time
    - *No problems with consistency between machines.*
    - *Only partition owner can log on.*
  - Authentication for other users based on:
    - *Personal public/private key pair.*
    - *Host-based authentication.*
- ntpdate

zinfo

- Provide login access for partition owner
- Bonus:
  - Job initialization and termination events
- Components:
  - Zinfod – running system-wide on the service node
  - Zinfo-ion – part of the ramdisk (running on IONs)
  - Zinfo – user-level query command (login nodes)

zinfo


getuser R000_J203-256
ION
Service
test:*:100:100:...
ionodes bR000_J203-256
172.30.1.17 172.30.1.20 ...
Login

Sample usage

- cqsub options executable
  - Outputs Cobalt job ID.
- zinfo -w -c cobalt_job_id
  - Waits until the job starts running, outputs IONs IPs.
  - (can also specify BG/L job ID or block/partition ID)
- zinfo -e -c cobalt_job_id
  - Waits until the job finishes.

```
#!/bin/sh
cjob_id=`cqsub parameters`
ion_ips=`zinfo -w -c $cjob_id`
echo "IO nodes are $ion_ips" | \
mail -s "Job $cjob_id started" $USER
zinfo -e -c $cjob_id >/dev/null
mail -s "Job $cjob_id finished" $USER </dev/null
```

Can I Try ZeptoOS and PVFS2 Easily?

Yes….

Argonne NATIONAL LABORATORY

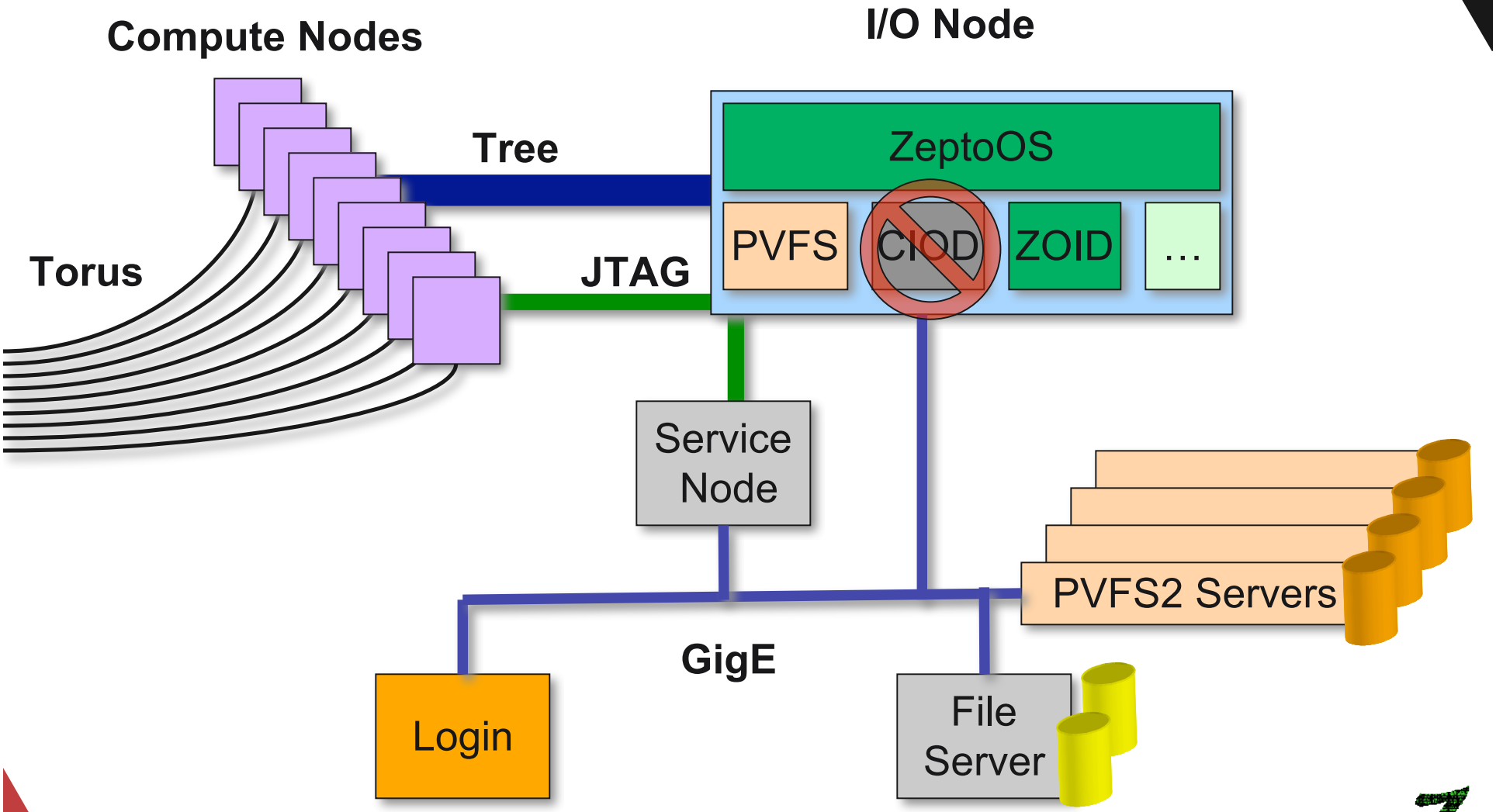
ZOID

THE UNIVERSITY OF CHICAGO

Office of Science
U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory managed by The University of Chicago

ZeptoOS + PVFS2 on BG/L



ZOID Goals

- Provide advanced, high-performance I/O on massively parallel, petascale machines
  - Not just an I/O pass-through
- Augment/replace the basic IBM BG/L CIOD

- Functionality:
  - Provide system call forwarding from stripped-down compute nodes to the I/O nodes
  - Direct support for high-performance parallel I/O
  - Support for collective system calls and caching
  - Asynchronous I/O

Design

- On the compute nodes -- a client (library/separate process/kernel):
  - remote procedure calls
  - (possibly) local caching
- On the I/O node side -- a server infrastructure:
  - scheduler (also doing caching)
  - $n workers performing the I/O calls
- (CIOD on BGL currently performs no caching, is synchronous and single-threaded)
- Buffer copys are out of control for parallel disk I/O

Initial implementations

- Experimental ZOID infrastructure:
  - x86
  - client code in a linker library
  - call forwarding over TCP/IP
  - server a single, but multi-threaded process
- Experimental BGL support:
  - closed network protocols make call forwarding difficult
  - workaround: forward arbitrary calls over the existing forwarding
  - infrastructure (use readlink() and write()/read()).
  - drawback: inherits many of CIOD shortcomings (synchronicity, scalability)

Current implementation (work in progress):

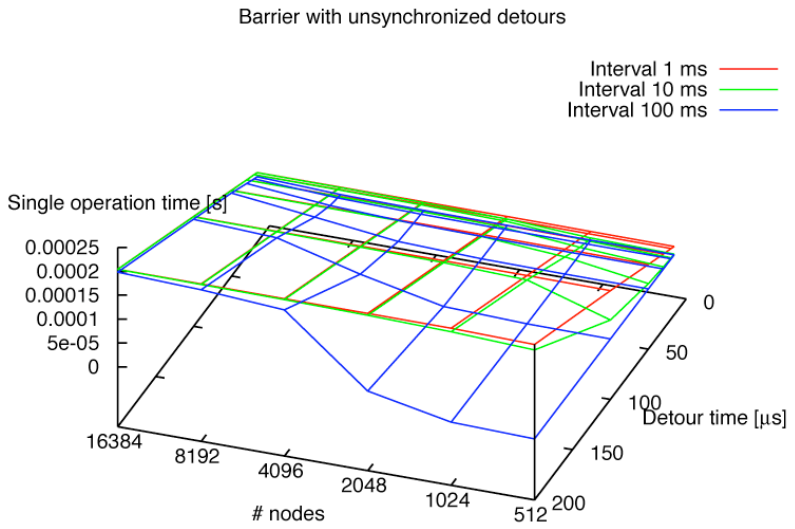
- Thanks to new information from IBM on network protocols, can now program the low-level network interfaces on our own
- Perform custom system call forwarding:
  - custom ZOID daemon on I/O nodes
  - disable CIOD right after job startup
  - link applications with replacement libc
  - IBM-provided compiler/MPI should not be affected
- Bonus: strictly user-space implementation on CN side can improve latency

ZOID Future

- Zero-copy read/write support
- Asynchronous I/O (hardware supports it, CNK not quite)
- Merge with multithreaded ZOID server
- Implement collective calls and caching
- Once Linux runs on compute nodes, it will be even more crucial than now

Ongoing and Future ZeptoOS Work:

- Developing ZOID, and advanced I/O node function call forwarding daemon
  - Supporting collective OS calls
  - Flexible consistency semantics
- ZeptoOS/Linux as a compute node kernel
  - Simplify EVERYTHING!
  - Build optimized MPICH2 & ROMIO for BG/L
  - Build debuggers
  - Simple PAPI support
  - Dynamically loaded libraries
  - Support Fork!
  - Support Python
  - Simple tool chain
  - Fantastic options for optimizations
  - Community, community, community!



Ongoing and Future ZeptoOS Work:

- Explore dynamic kernel config/build per job
- Collective OS operations
  - Dynamic library
  - I/O
- Integration of KTAU (from our U Oregon partners) into ZeptoOS
- Dual Core Support for I/O node
  - Dependency: lack of information regarding to hardware synchronization mechanism (cache issues)
- MPI coupling (using I/O Nodes)
- Memory management enhancements

- Ongoing work with LLNL and IBM for BG/P Design

www.mcs.anl.gov/zeptoos