I/O Threads to Reduce Checkpoint Blocking for an EM Solver on Blue Gene/P and Cray XK6

Jing Fu*†, Misun Min¶, Robert Latham¶, Christopher Carothers†

†Department of Computer Science, Rensselaer Polytechnic Institute
¶ Mathematics and Computer Science Division, Argonne National Laboratory

June 29, 2012

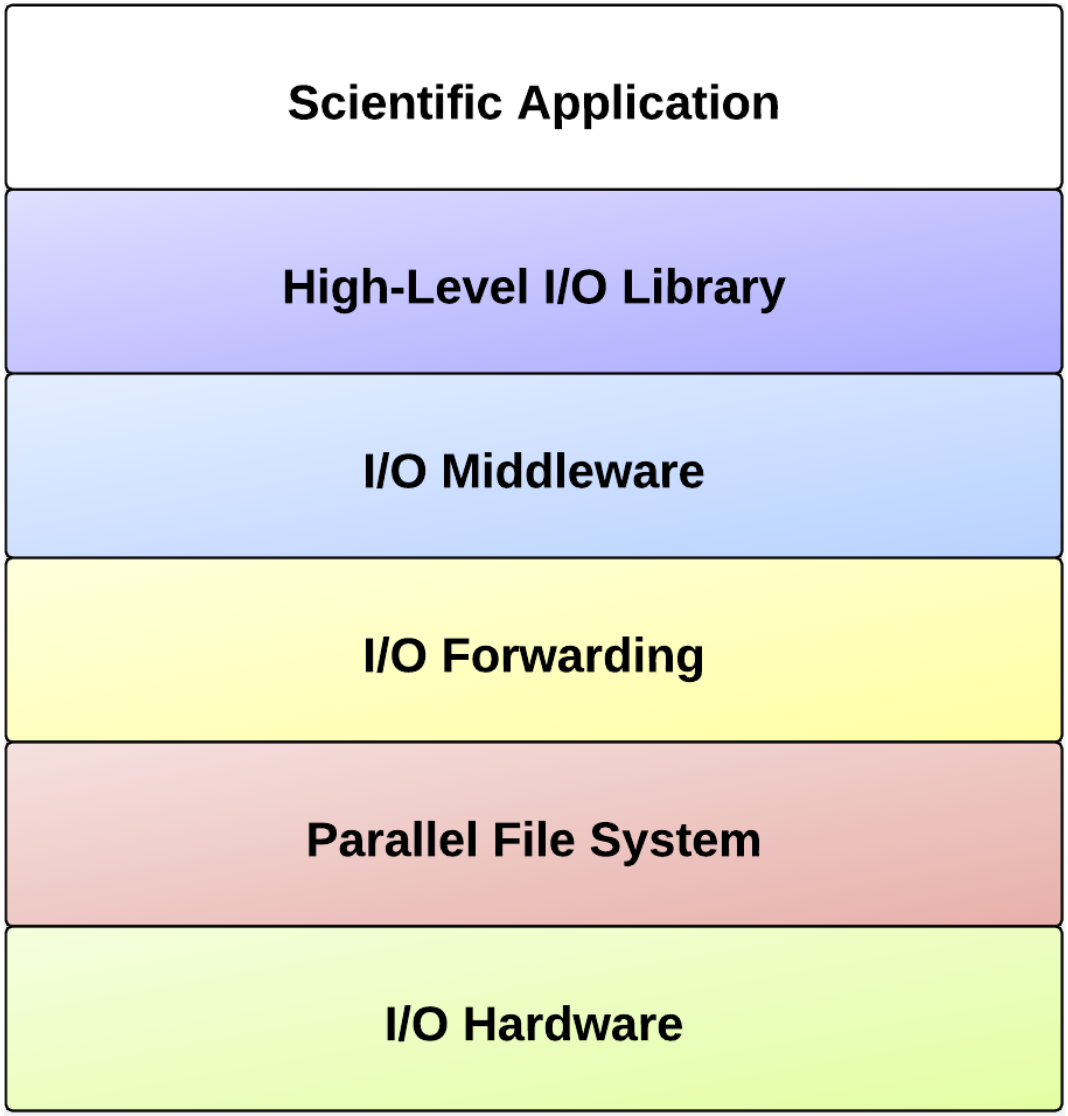
Presentation Outline

- **Introduction**
- **Approaches**
- **Performance and Analysis**
- **Summary**

Solver systems and checkpointing

- *Parallel Partitioned Solver Systems* are being applied to tackle hard problems in science & engineering, e.g. *PHASTA* (CFD), *Nek5000* (CFD), *NekCEM* (CEM)
- These applications scale well on massively parallel platforms (strong scaling on 100,000s of cores)
- Traditional I/O doesn't scale as well, may suffer at large scale
- In this talk, we focus on the use of I/O threads for an EM solver (NekCEM) checkpoint on BG/P and Cray XK6

I/O software stack of a typical HPC system



Bursty I/O

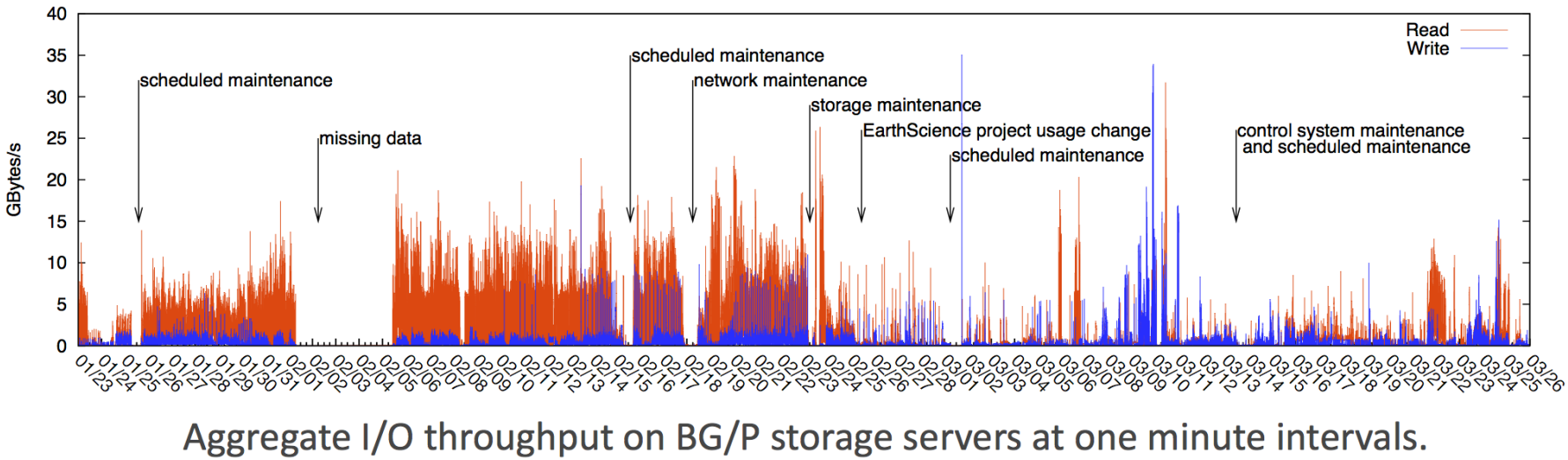


Figure: I/O workload in ANL, image courtesy of Rob Ross

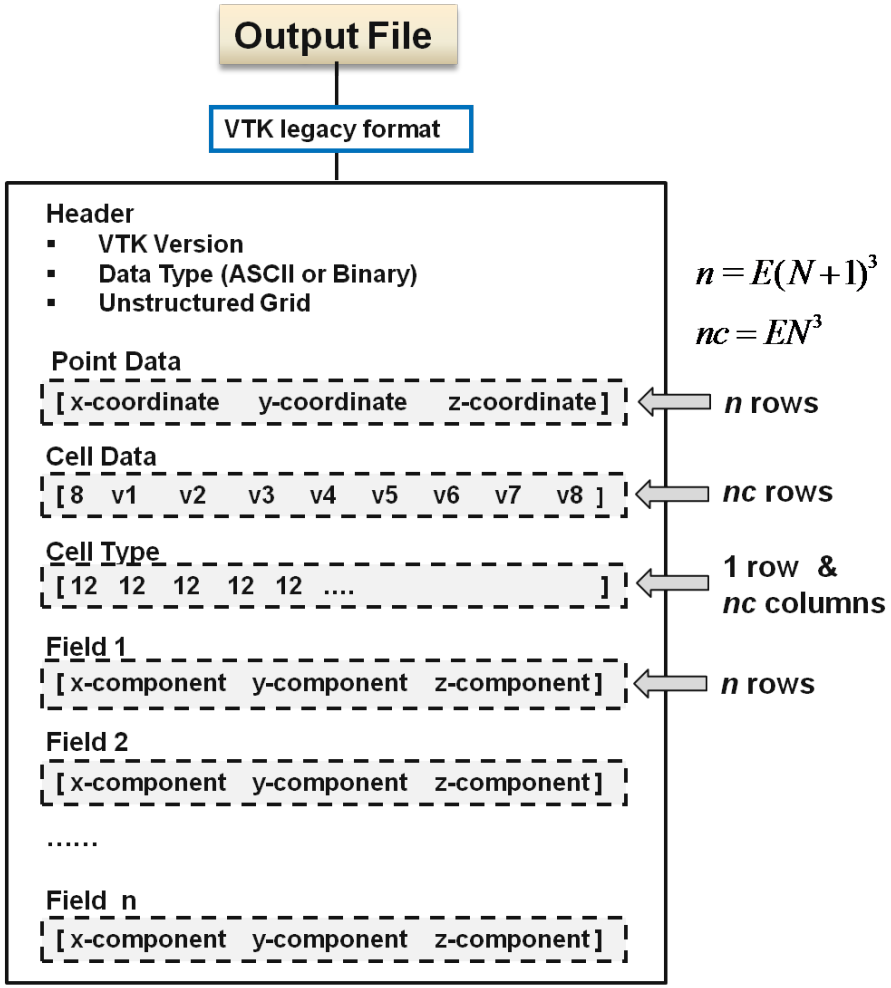
- Pattern: X steps comp. $\rightarrow$ checkpoint $\rightarrow$ X steps comp. ...
- Core assumption: **synchronized** writes among all processors (lack of well-supported asynchronous I/O on supercomputers)

Checkpoint File Structure

File Body

Header:	part 0, field 0
Data:	part 0, field 0
Header:	part 1, field 0
Data:	part 1, field 0
Header:	part 2, field 0
Data:	part 2, field 0
Header:	part 3, field 0
Data:	part 3, field 0
Header:	part 0, field 1
Data:	part 0, field 1
Header:	part 1, field 1
Data:	part 1, field 1
Header:	part 2, field 1
Data:	part 2, field 1
Header:	part 3, field 1
Data:	part 3, field 1

(a) Typical File Structure



(b) NekCEM File Structure

Related Work and Our Objective

Related Work

- Scalable Checkpoint/Restart, Lawrence Livermore National Lab
- ADaptable IO System, Oak Ridge National Lab
- I/O Delegate Cache System, Northwestern University

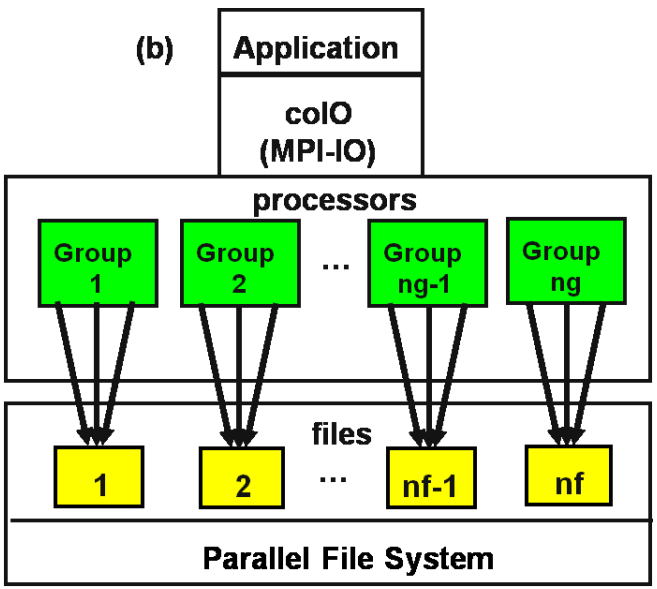
Design Factors

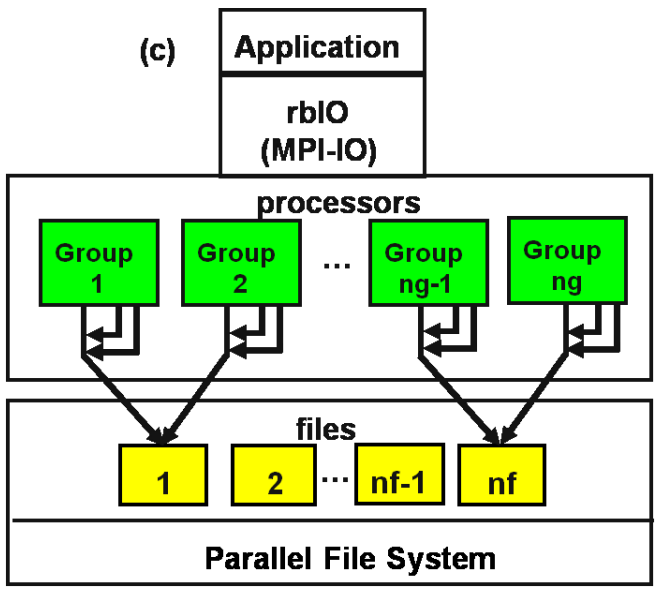
- design space; platform dependency; application transparency

Our Objective

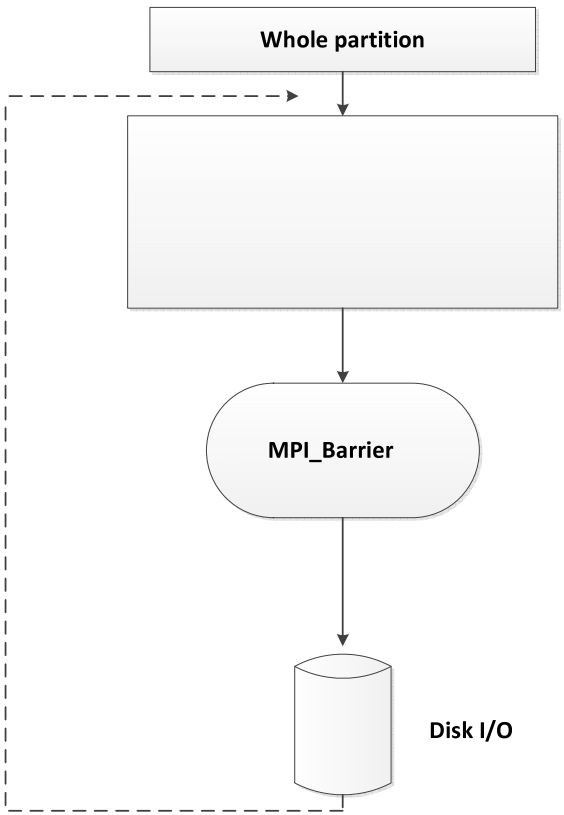
- Goal: performance at scale
- user space, portable, reasonably generic

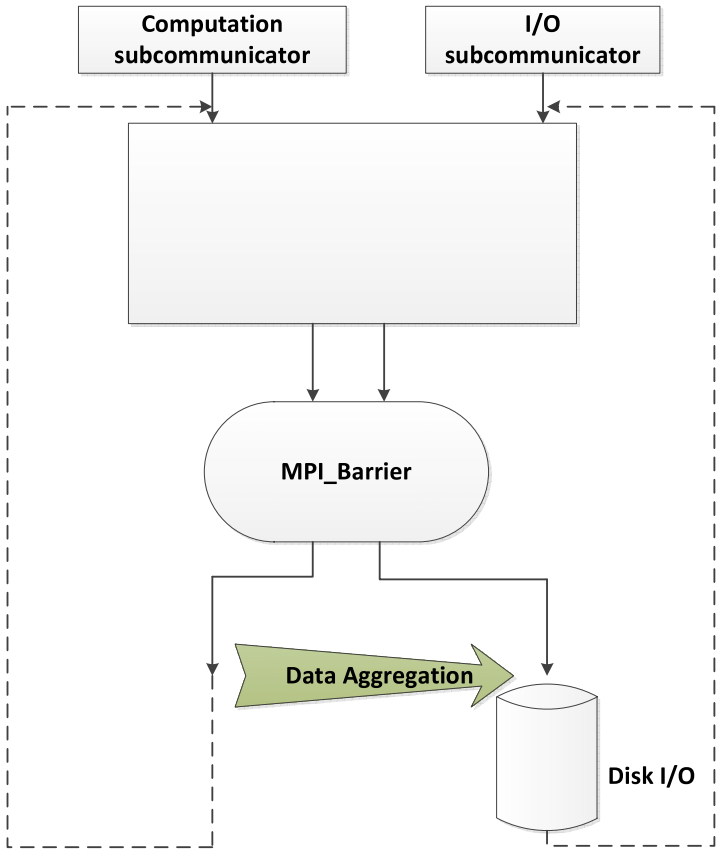
Previous work: from *coIO* to naive *rbIO*





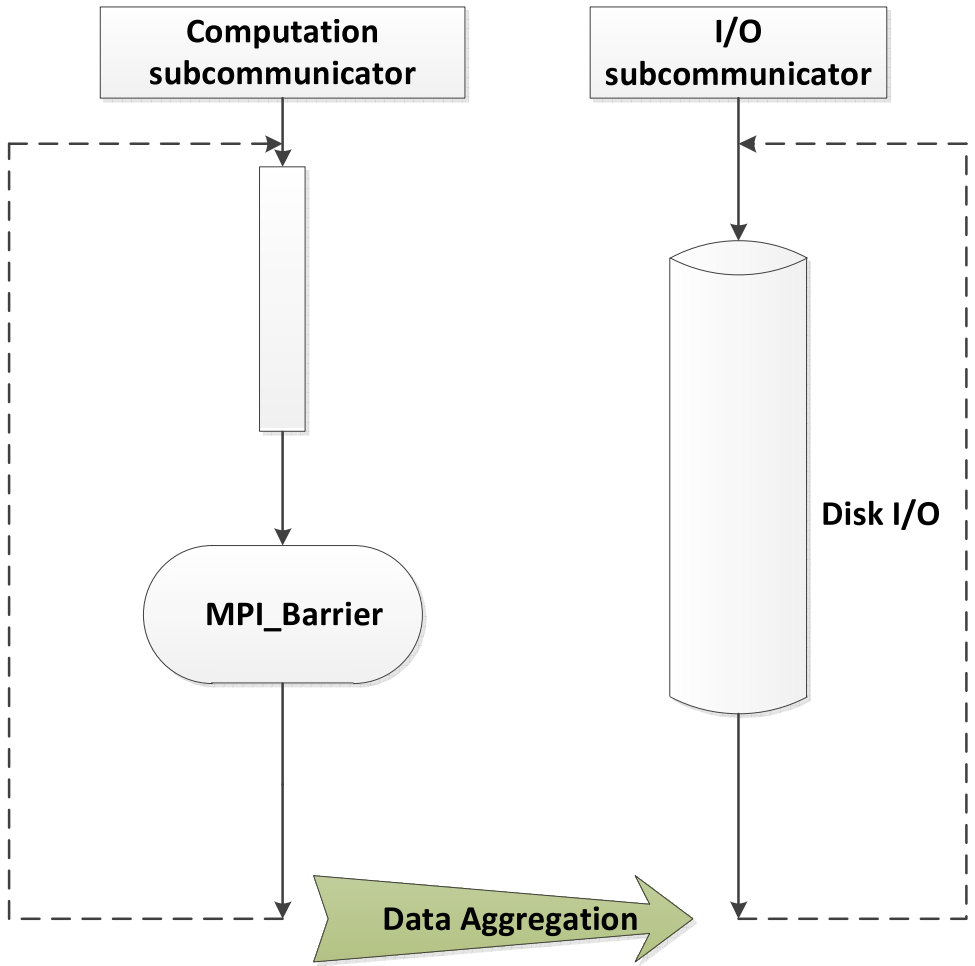
from *coIO* to naive *rbIO*





Method 1: Completely split rbIO

- dedicated I/O writers
- overlap computation and I/O
- lose a small portion of computing resources
- other problems?



Potential limitations with completely split rbIO

- break collective operation optimizations on Blue Gene systems
- collective operations on subcomm go through torus not tree
- 10× slower on torus

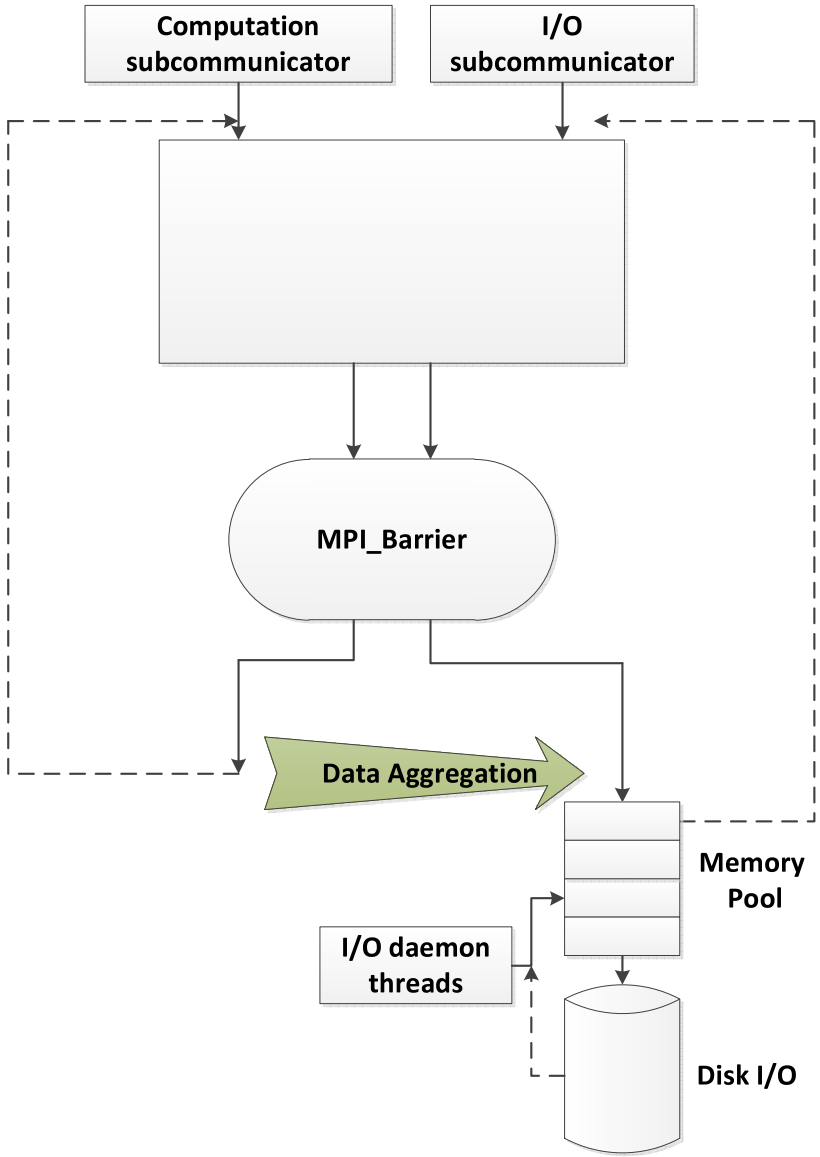
Table: The time (in μs) *MPI_Allreduce* spends on BG/P

#nodes	Time on Tree	Time on Torus	Ratio
4096	7.68	55.65	7.24
8192	7.72	61.88	8.01
16384	8.19	67.66	8.26

- performance impact on applications: 1 - 2% time spent on collective now means 10 -20%
- can be verified by running application with tree network off on BG/P

Method 2: rbIO with I/O daemon threads

- global communicator
- simple control flow
- threading supercomputers?



Potential limitations of threading rbIO

- BG/P has limited threading capability
- default to one, up to three threads per core
- does not support automatic thread switching
- have to use hardware thread in SMP mode
- experiment for demo purpose

- load balancing issue for those that fully support threads, e.g. Cray XK6?

NekCEM I/O on Blue Gene/P

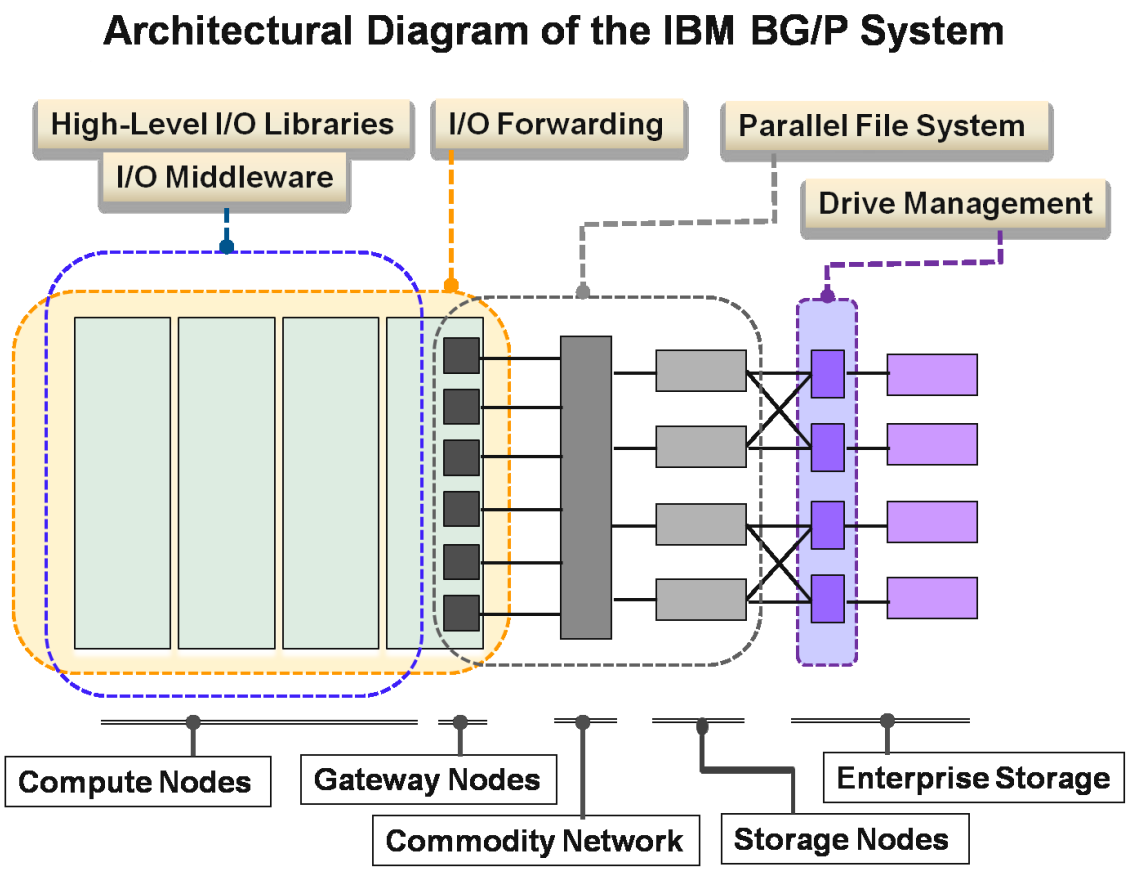
Blue Gene/P Spec

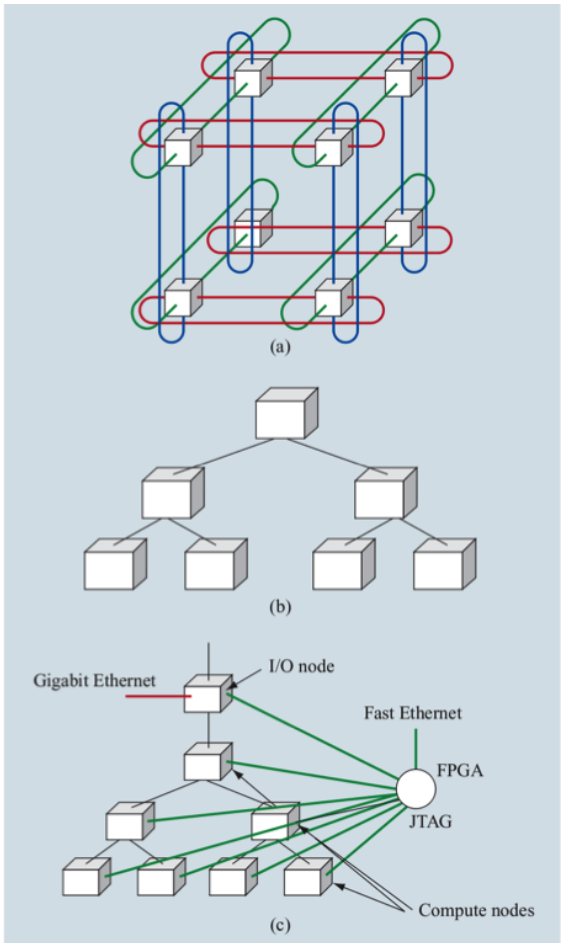
- 163,840 cores, 80 TB RAM, 557 teraflops ("*Intrepid*"@ANL)
- GPFS/PVFS, 128 file servers connected to 16 DDN 9900, 10 PB
- pset (1 ION to 64 4-core CN), 640 ION to 128 file servers by 10GB/s Myricom switch
- 4MB block size, read peak 60 GB/s, write peak 47 GB/s

Experiment Setup

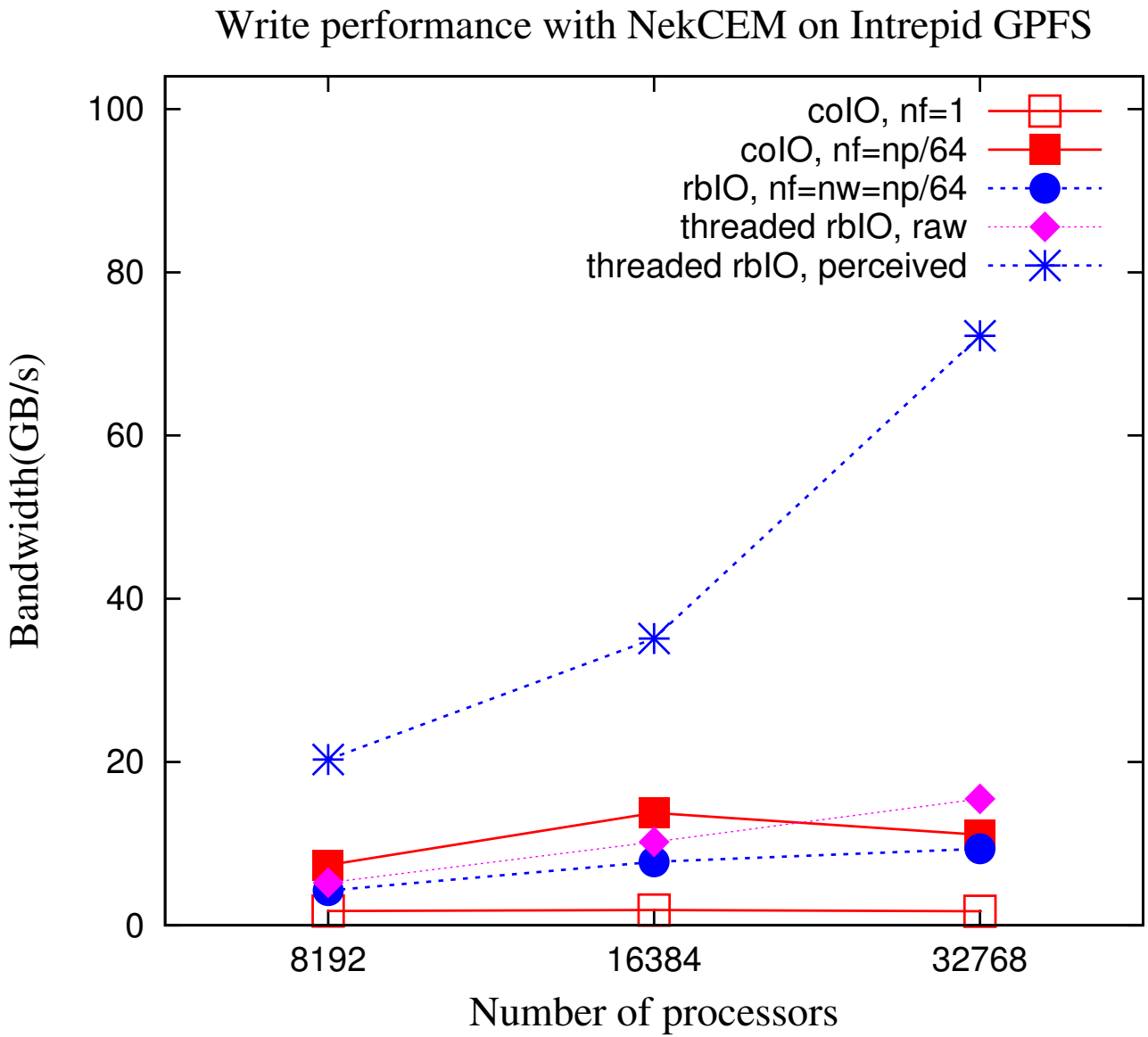
- 3D cylindrical waveguide simulation for different meshes
- (grid points, total size) = {(143M, 13GB), (275M, 26 GB), (550M, 52 GB)}
- Weak scaling tests

Overview of the Blue Gene system

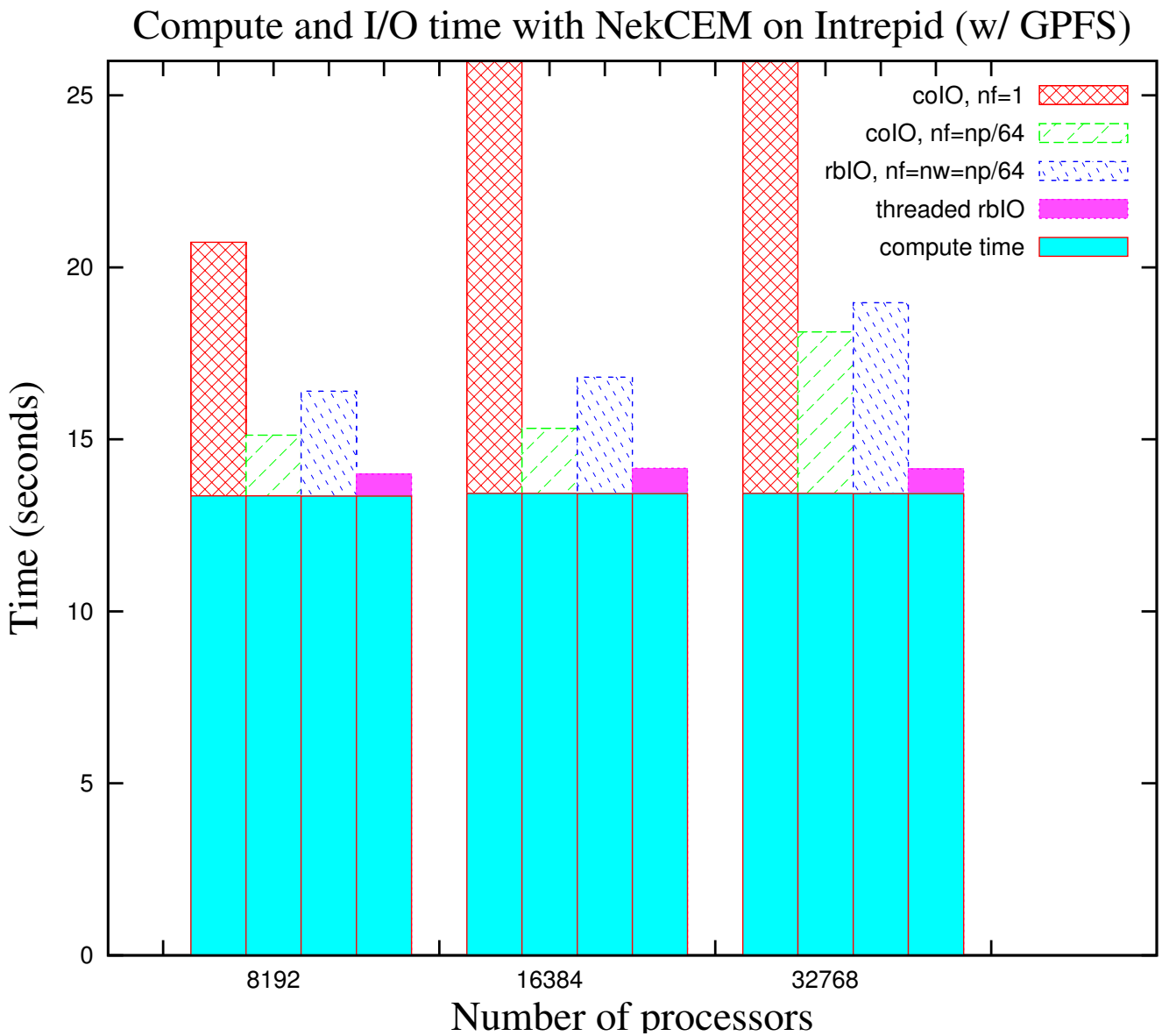




NekCEM I/O on BG/P: *bandwidth*



NekCEM I/O on BG/P: *overall time*



NekCEM I/O on Cray XK6

Cray XK6 Spec

- 299,008 cores (AMD Opteron Interlagos, on Cray Linux microkernel), 598 TB RAM, 2.63 petaflops ("*Jaguar*"@ORNL)
- Lustre, 192 OSS servers to 96 DDN 9900s (7 RAID-6 (8+2)/OSS), 10 PB
- 4MB block size, peak 120 GB/s

Overview of the Cray system

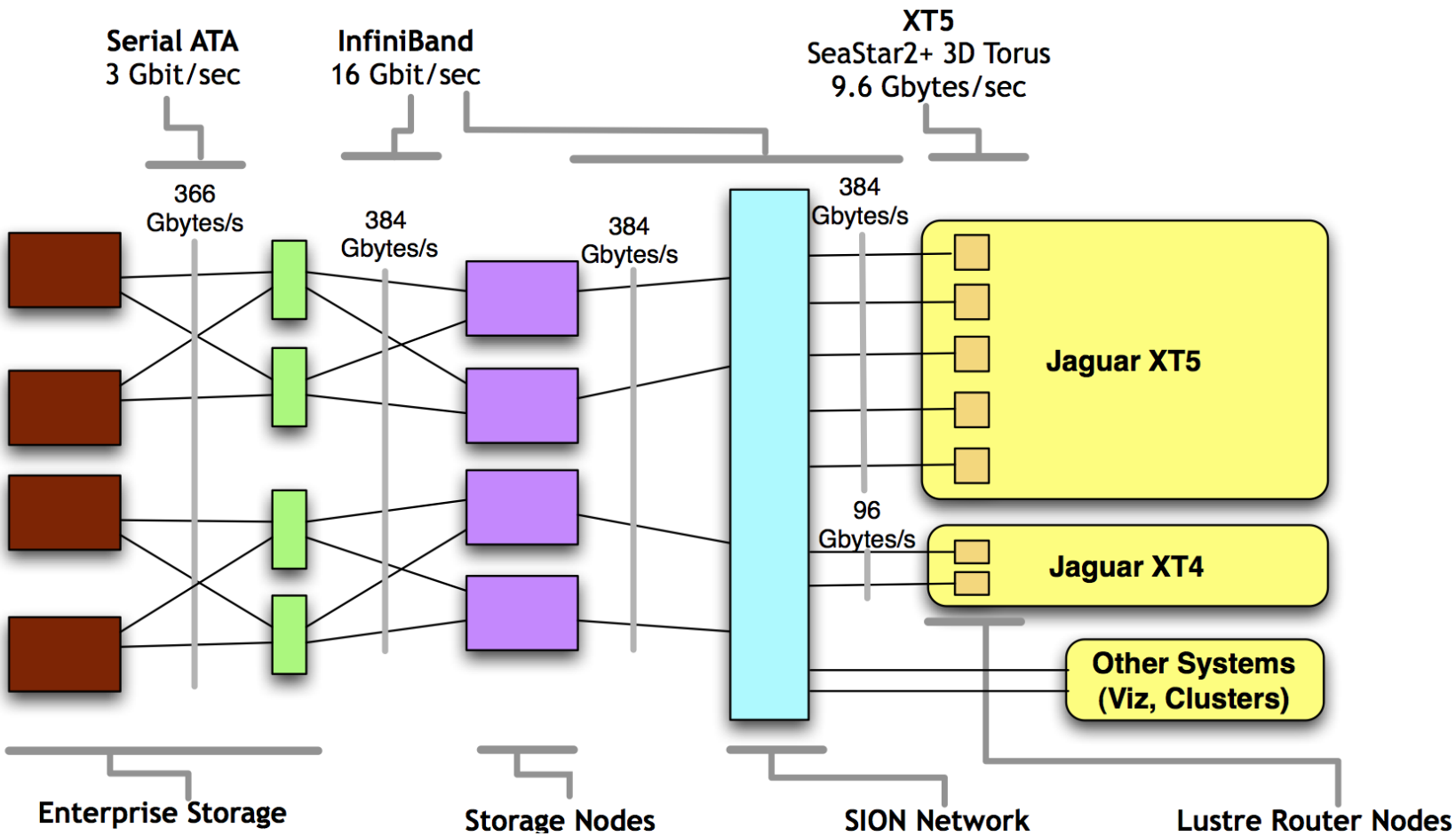
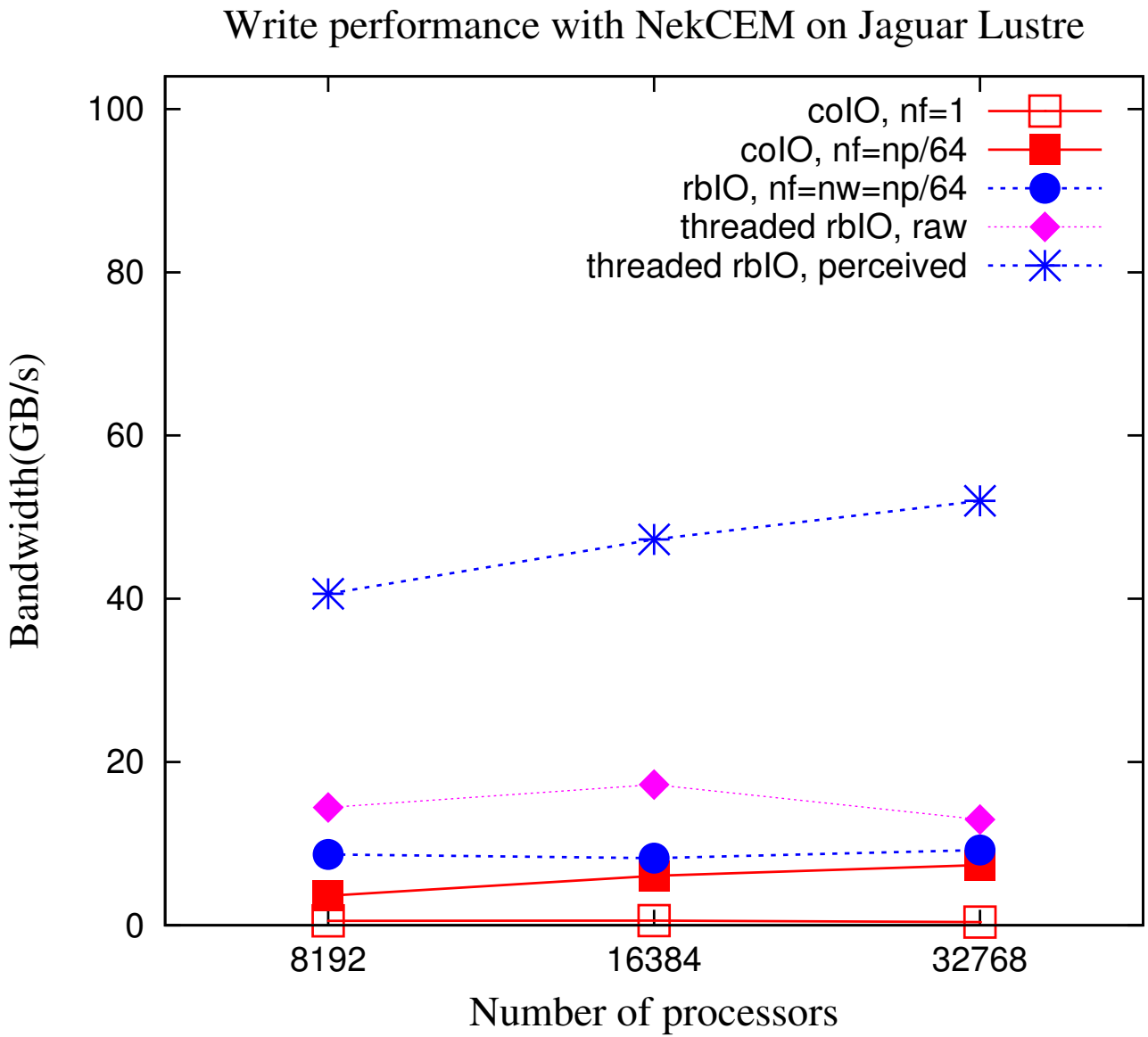
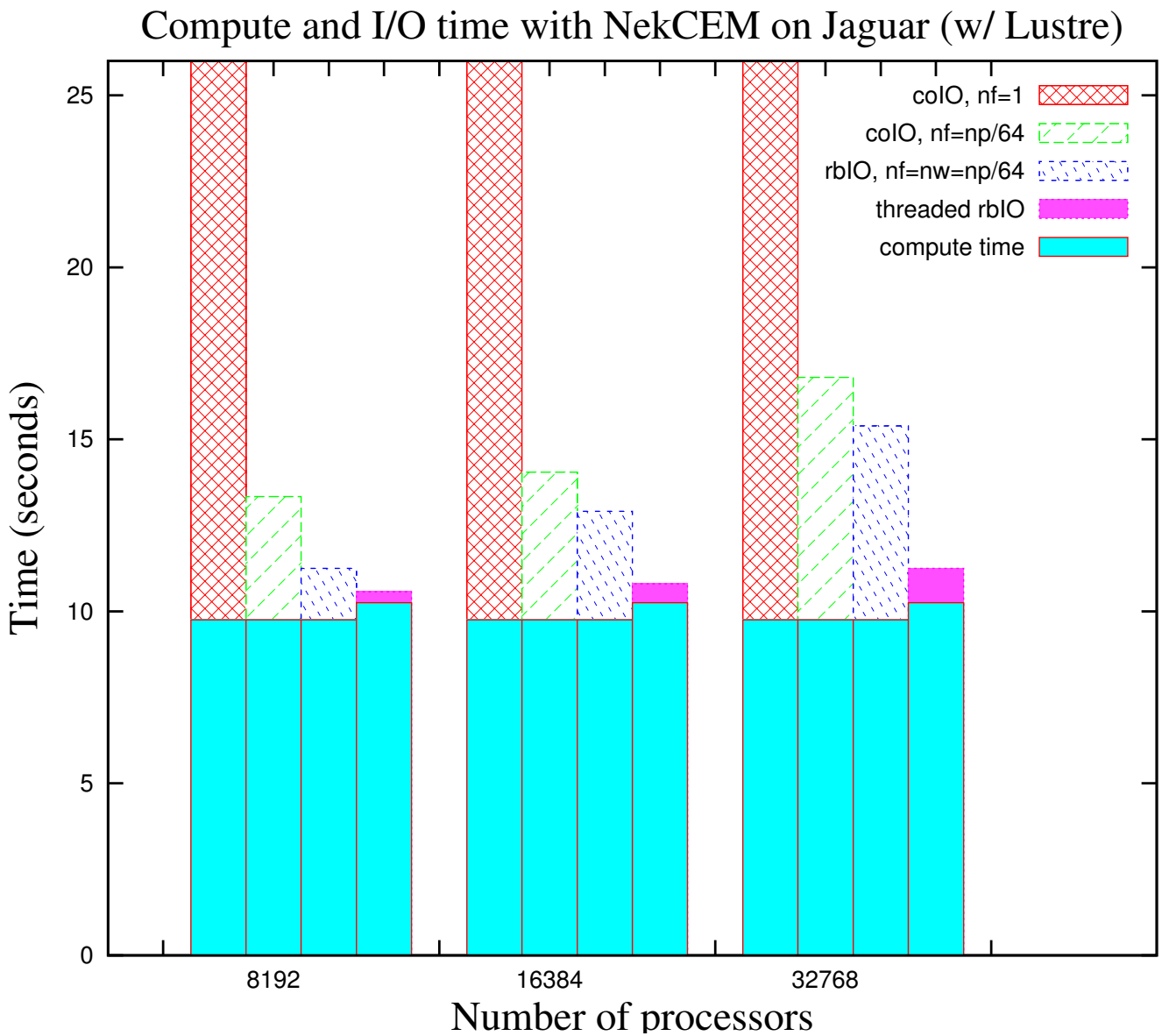


Figure: Architecture diagram of Jaguar@ORNL, image courtesy of Rob Ross

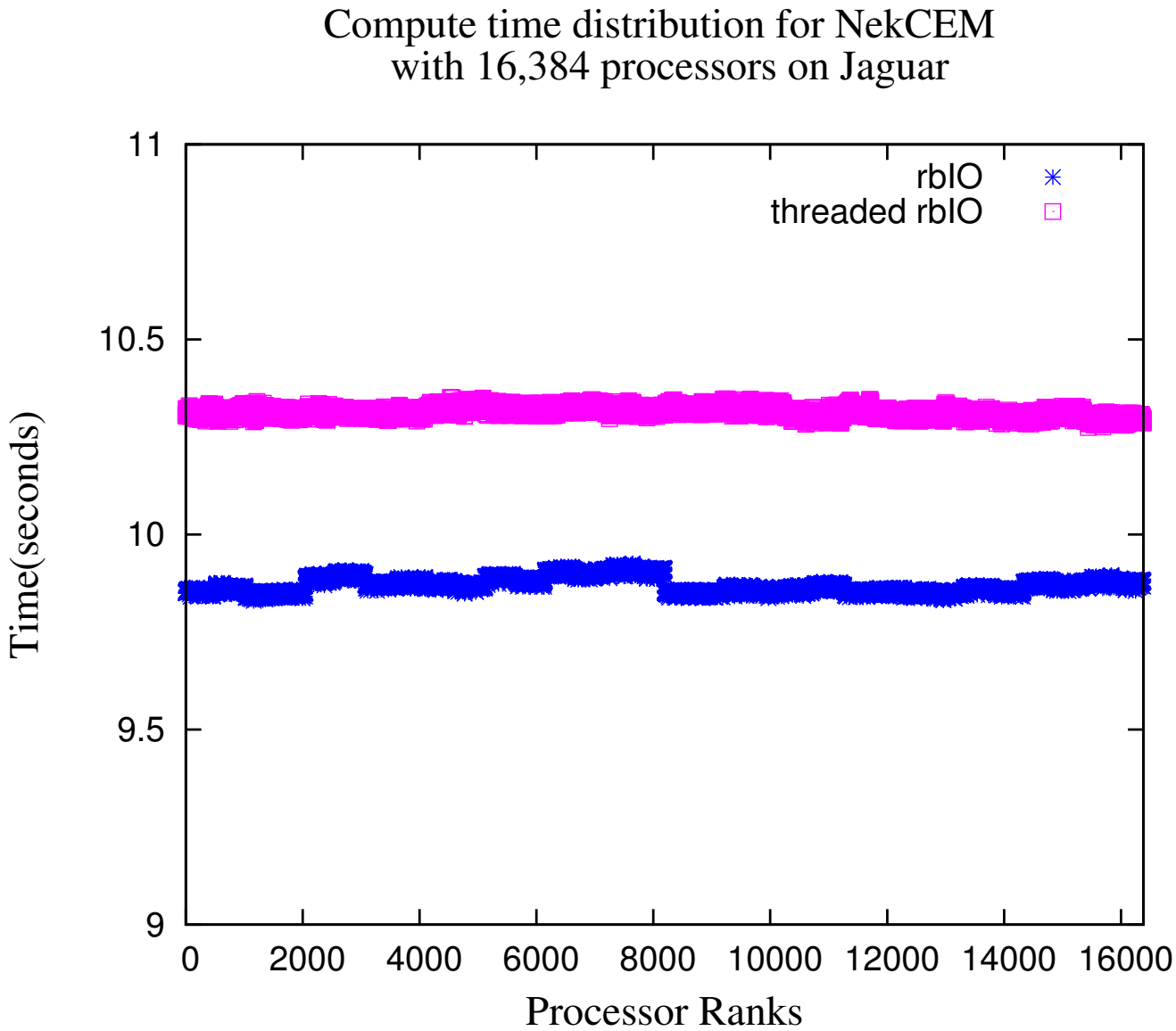
NekCEM I/O on Cray: *bandwidth*



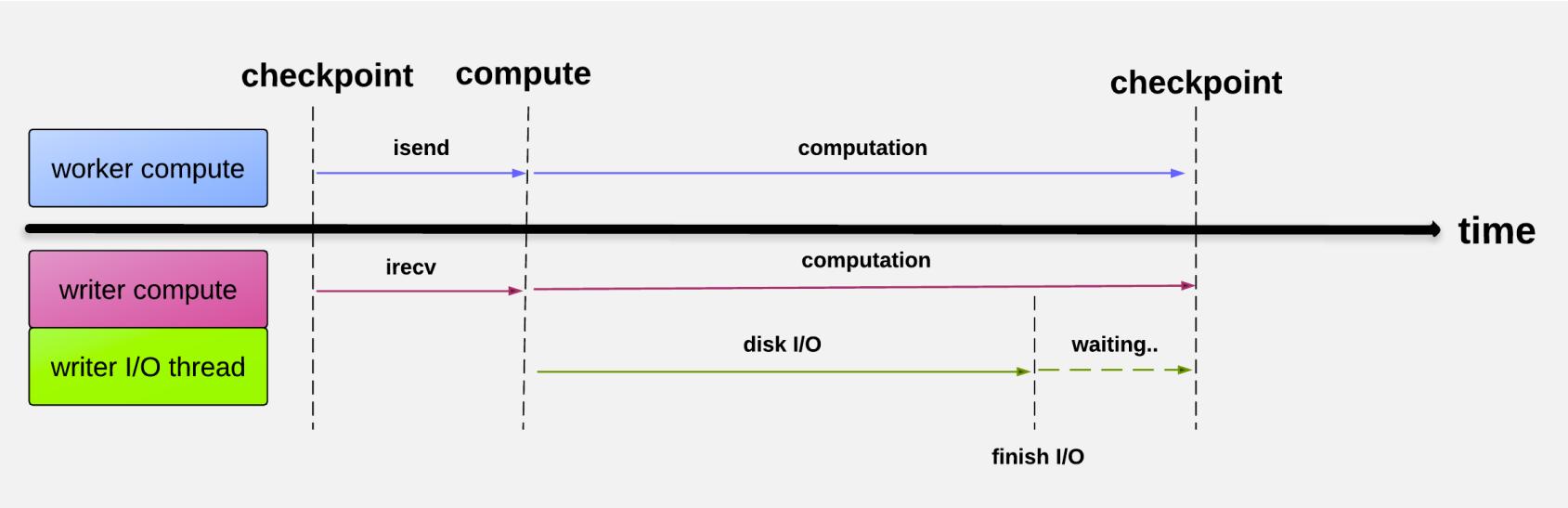
NekCEM I/O on Cray: *overall time*



NekCEM I/O on Cray: *profiling compute time*



NekCEM I/O on Cray: *Threaded rbIO Timing Analysis*



NekCEM I/O on Cray: *Speedup Analysis*

$$
\begin{aligned}
Speedup_{prod} &= \frac{T_{coIO} + T_{comp}^{\text{coIO}}}{T_{trbIO} + T_{comp}^{\text{trbIO}}} \\
&= \frac{X_{coIO} * t_{comp}^{\text{coIO}} + f_{cp} * t_{comp}^{\text{coIO}}}{X_{trbIO} * t_{comp}^{\text{trbIO}} + f_{cp} * t_{comp}^{\text{trbIO}}} \\
&= \frac{X_{coIO} + f_{cp}}{X_{trbIO} + f_{cp}} * \frac{1}{1+\delta},
\end{aligned}
$$

where X is the number of computation steps that a checkpoint time equals to, f_{cp} denotes number of computation steps between two checkpoints, and δ is the overhead of a single step computation with threaded rbIO compared with nonthreaded I/O (i.e., $t_{comp}^{\text{trbIO}} = (1+\delta) * t_{comp}^{\text{coIO}}$).

Roughly 50% speedup on 32K procs Jaguar.

Summary

- Application-transparent optimizations (MPI-IO collective) with good tuning practice can provide decent performance on some platforms
- Application-level optimizations exploit application-specific information and provide tuning options (*nf, ng, I/O thread*) and good performance on most platforms
- Data staging (on RAM, RAM disk, SSD) helps ease out pressure of bursty I/O for file system, trending technique in design of storage system for Exascale era

- What happens on Mira and Blue Waters?

Collaborators

Ning Liu, Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute

Onkar Sahni, Min Zhou, Mark Shephard
Scientific Computation Research Center (SCOREC)
Rensselaer Polytechnic Institute

Michel Rasquin, Kenneth Jansen
Aerospace Engineering Sciences
University of Colorado Boulder

Misun Min, Paul Fischer, Rob Latham, Rob Ross
Mathematics and Computer Science Division
Argonne National Laboratory

Questions?