



Scalable I/O Middleware and File System Optimizations for High-Performance Computing (CCF-0621443)

PI: Alok Choudhary, Professor

Director: Center for Ultra-Scale Computing and Security

Electrical Engineering and Computer Science & Kellogg School of Management

Northwestern University

choudhar@ece.northwestern.edu

Co-PIs (M. Kandemir, PSU and R. Thakur, ANL)

Difficult Questions to wkliao@ece.northwestern.edu

Acknowledgements:

DOE SCIDAC program (SDM center), ST-HEC, NGS

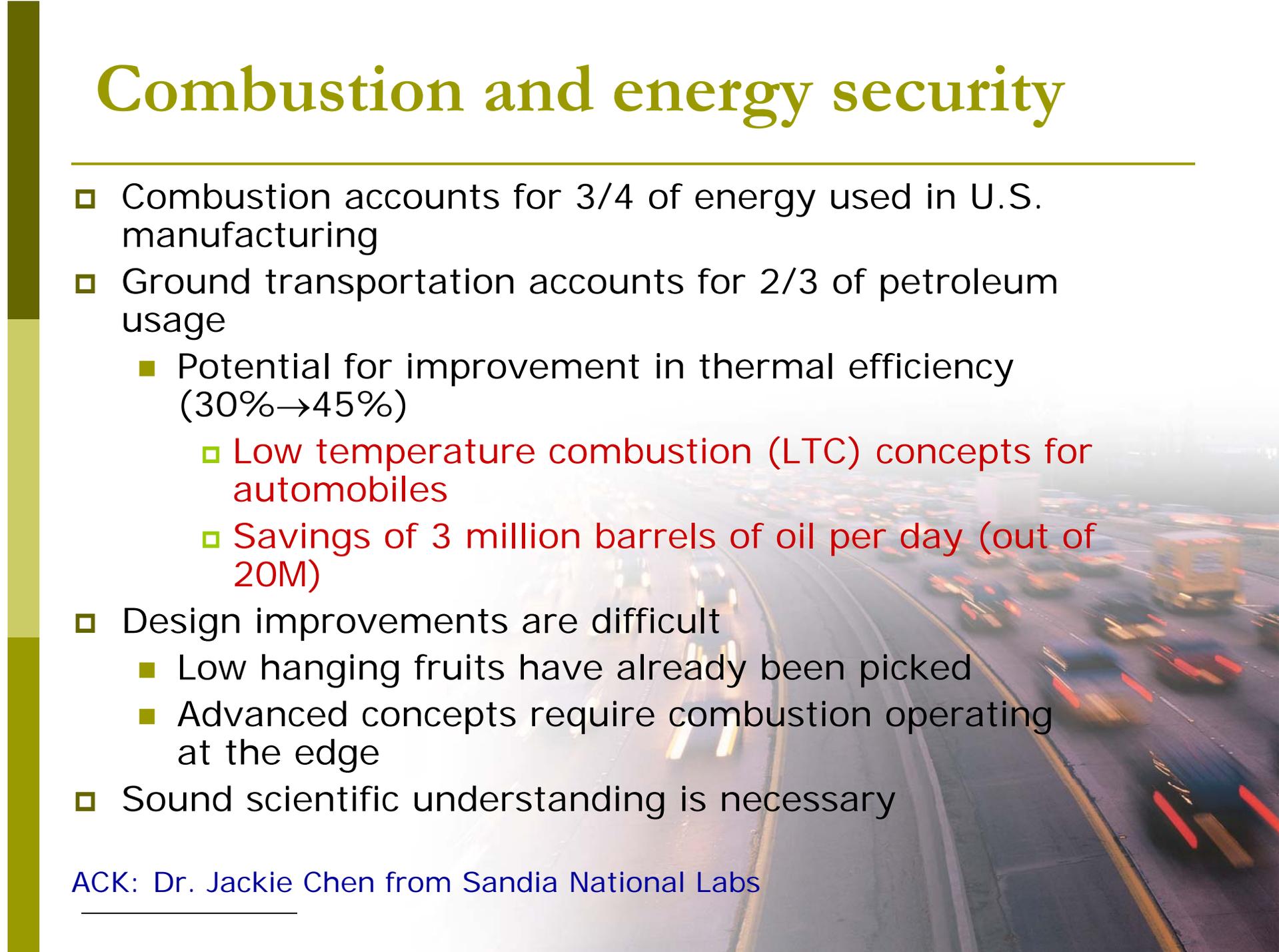
Students: Kenin Coloma, Avery Ching, Ramanathan, Berkin, Jianwei Li (Now at Wallstreet), Ying Liu (now faculty at Chinese Academy of Sciences), Joe Zambreno (now faculty at Iowa State), Wei-Keng Liao (Research prof at NWU),

Other Collaborators: Rob Ross (ANL) L Ward (SNL), G. Grider, J. Nunez, J. Bent (LANL)

Combustion and energy security

- Combustion accounts for 3/4 of energy used in U.S. manufacturing
- Ground transportation accounts for 2/3 of petroleum usage
 - Potential for improvement in thermal efficiency (30%→45%)
 - Low temperature combustion (LTC) concepts for automobiles
 - Savings of 3 million barrels of oil per day (out of 20M)
- Design improvements are difficult
 - Low hanging fruits have already been picked
 - Advanced concepts require combustion operating at the edge
- Sound scientific understanding is necessary

ACK: [Dr. Jackie Chen from Sandia National Labs](#)

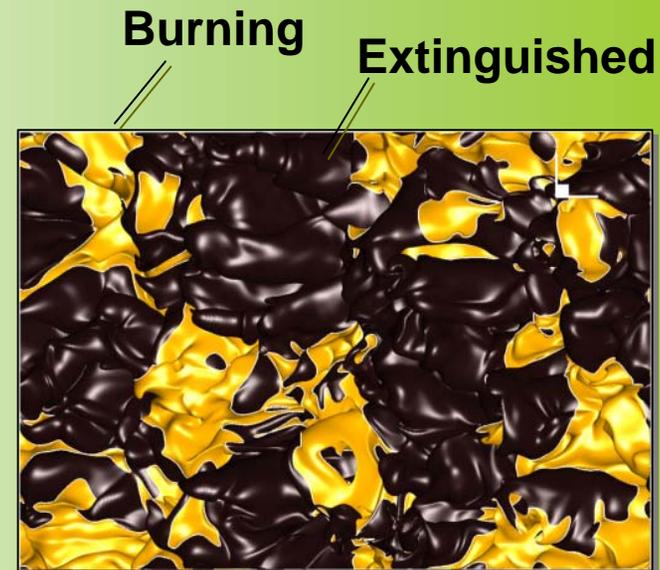
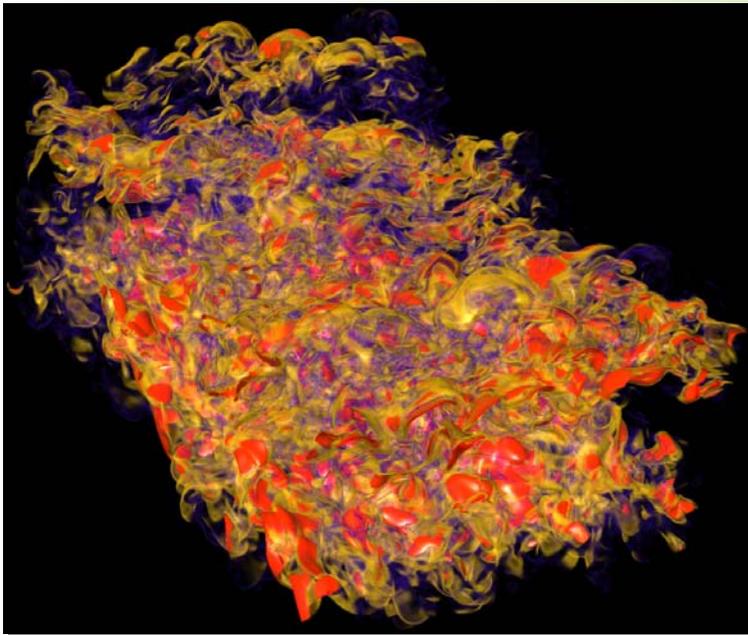


Combustion Application using DNS: Extinction and reignition in a CO/H₂ jet flame

Understanding extinction/reignition in non-premixed combustion is key to flame stability and emission control in aircraft and power producing gas-turbines

Discovered dominant reignition mode is due to engulfment of product gases, not flame propagation

Scalar dissipation rate

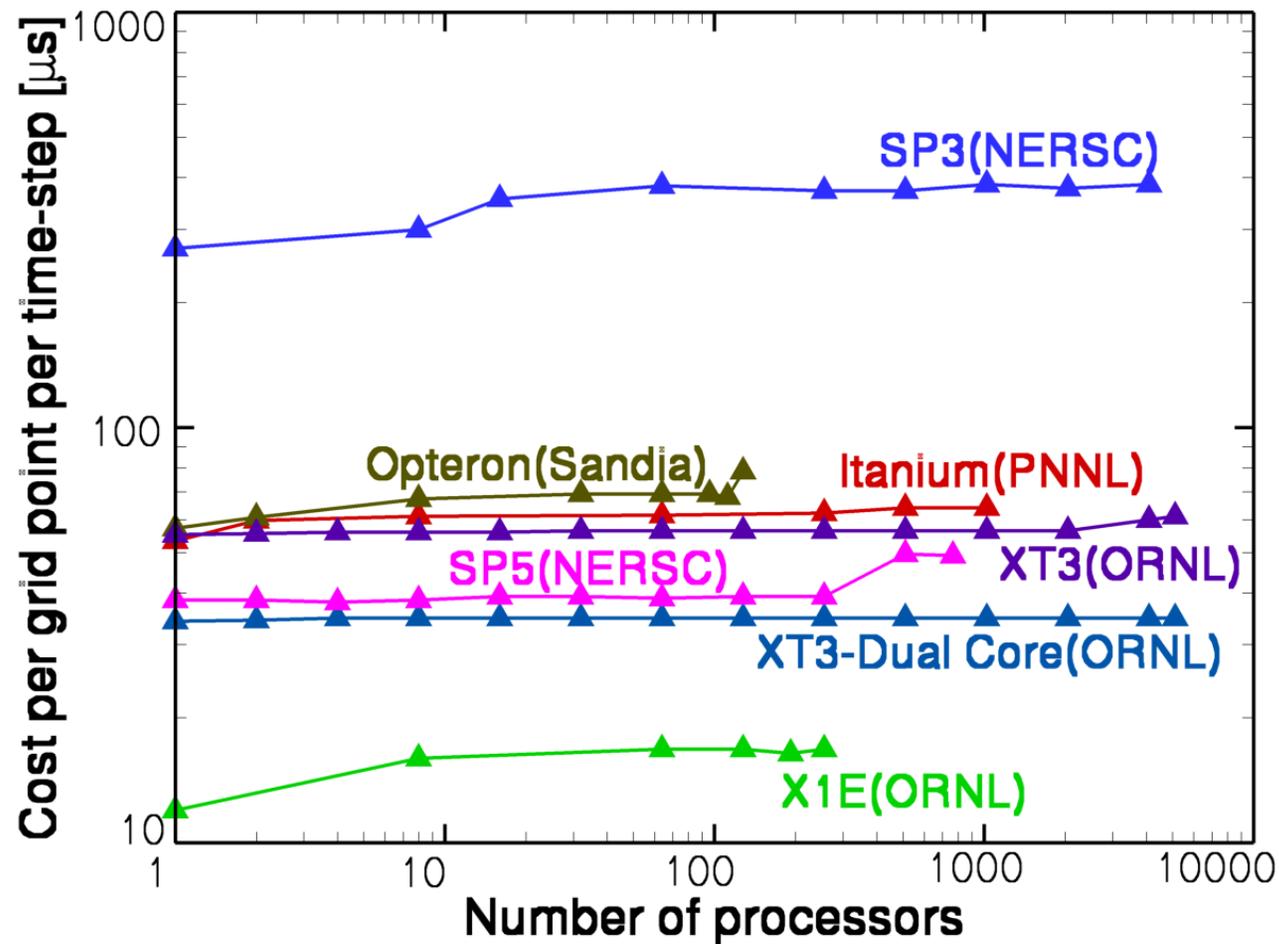


The ***largest ever simulations of combustion*** have been performed to advance this goal:

- 500 million grid points
- 11 species and 21 reactions
- 16 DOF per grid point
- 512 Cray X1E processors
- 30 TB raw data
- 2.5M hours on IBM SP NERSC (INCITE); 400K hours on Cray X1E (ORNL)

S3D parallel performance

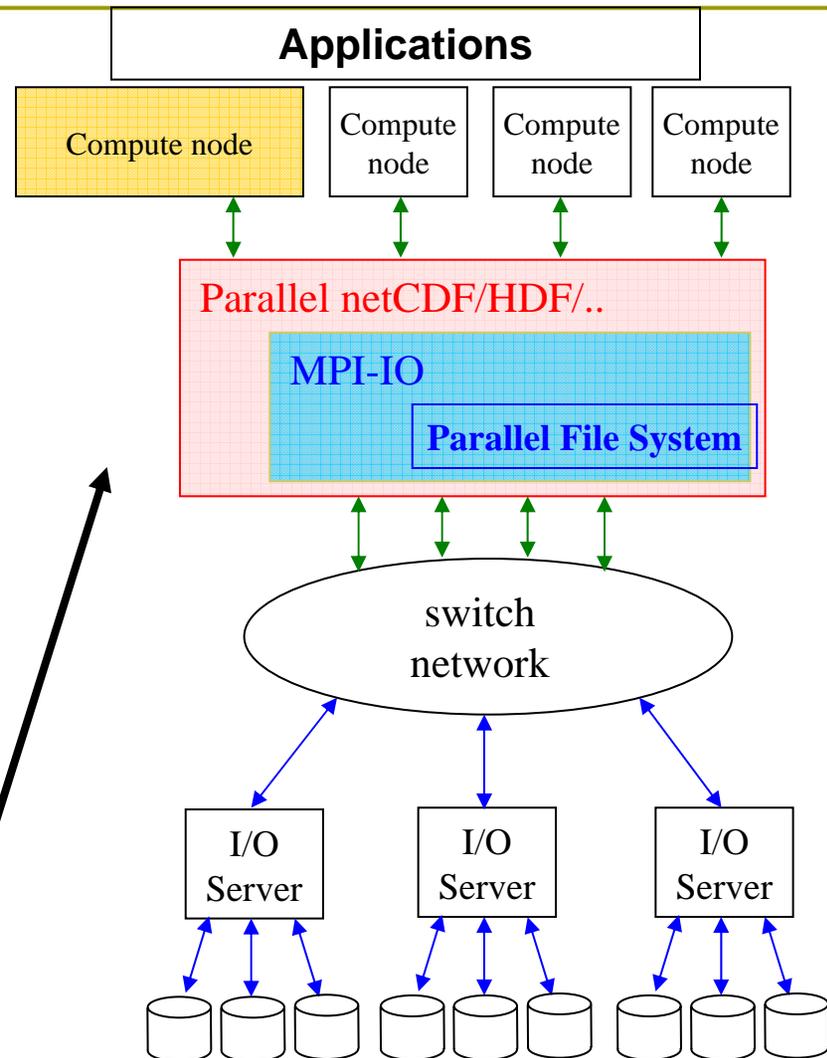
- S3D scales with 90% parallel efficiency on 10000 cores on CrayXT3 (ORNL)



Typical Software Layers for I/O in HEC

- Based on a lot of current apps
- High-Level
 - E.g., NetCDF, HDF, ABC
 - Applications use these
- Mid-level
 - E.g., MPI-IO
 - Performance experience
- Low Level
 - E.g., File Systems
 - Critical for performance in above

End-to-End Performance critical



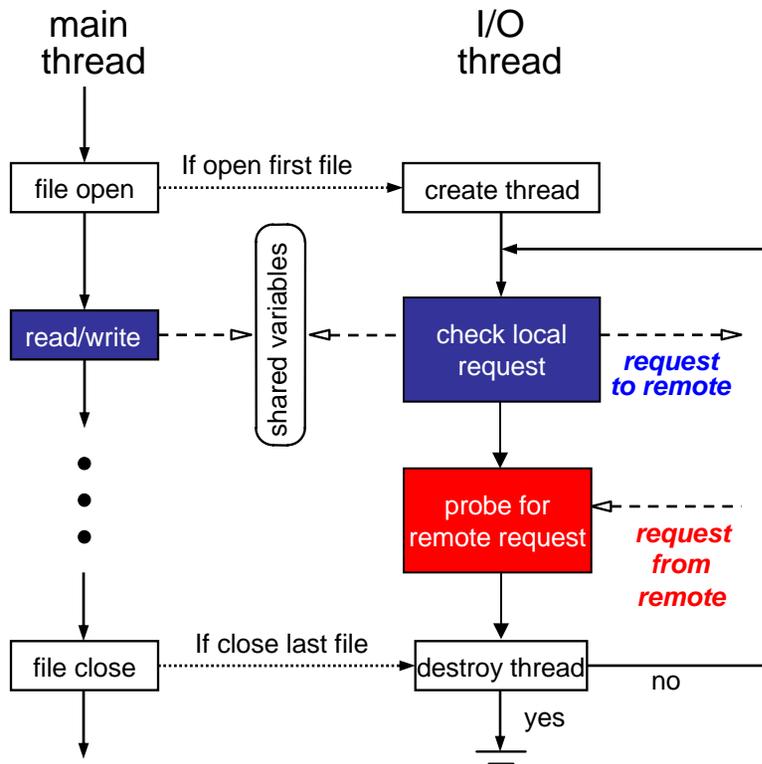
Client Process Collaboration

- ❑ Proved to be scalable for parallel I/O
 - Well-known example: 2-phase I/O adopted in ROMIO
 - ❑ Relying on I/O servers is not scalable
 - Number of servers is much less than clients
 - Servers use file locking to maintain file consistency
 - ❑ Locking is not scalable, should be avoided if possible
 - Servers do not tell if requests from one client are related to another
 - ❑ Clients collaboration on I/O
 - Reduce communication link contention on servers
 - Faster communication among clients
 - Resolve access conflicts within the group of clients
 - Rearrange I/O for best underlying file system performance
-

MPI-IO Client-side File Caching

- Goals
 - A fully functional, application-aware caching layer in MPI
 - Inter-process collaboration for coherence control, file system lock boundary alignment
 - Reduce I/O servers' workload
- Design
 - Global cache metadata management
 - Metadata of file blocks are statically distributed in round robin
 - A distributed lock management for keeping metadata integrity
 - Local cache page management
 - Page eviction, migration
 - I/O thread
 - Enable remote cache data access at the background
 - Enable overlapping of computation and I/O

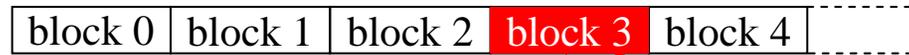
I/O Thread



- One thread per MPI process
 - Created at the first file open
 - Destroyed at the last file close
- Handle local requests
 - Keep watching a local mutex protected shared variable
 - Process all I/O related requests, leaving the main thread alone
- Handle remote requests
 - MPI_Iprobe() is used to probe remote requests
- I/O
 - Makes read/write calls to the file system

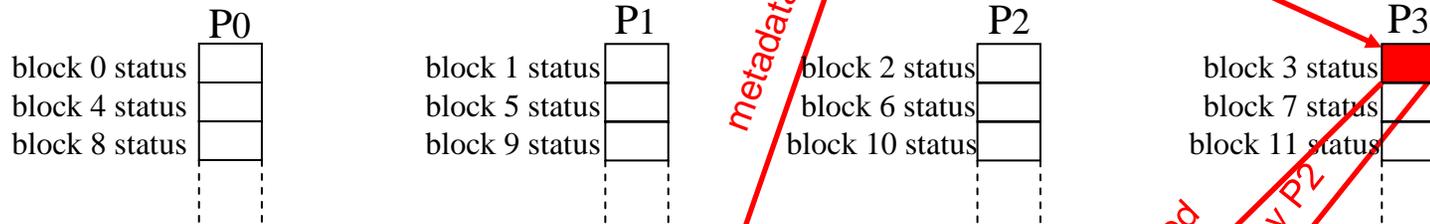
Design: An Example

Logical partitioning view of a file

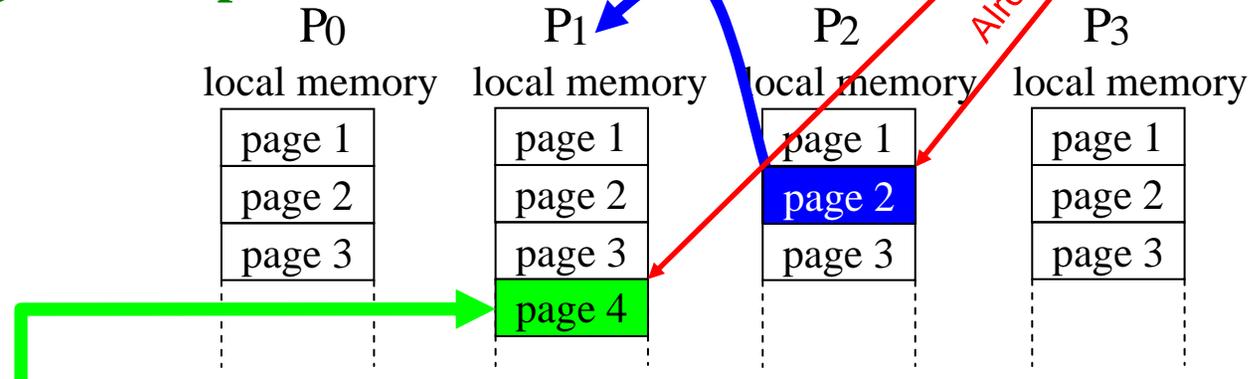


- Metadata communication
- Cache data communication
- System call

Distributed metadata



Cache pages at compute nodes



File system

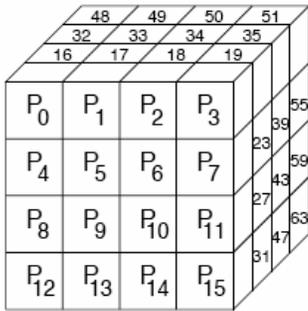
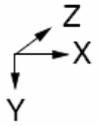
Experiment Setup

- Machines
 - Tungsten, a Linux cluster @ NCSA running Lustre
 - Use 16 I/O nodes, 512 KB stripe size
 - Mercury, an IBM cluster @ NCSA running GPFS
 - Use default 54 I/O nodes, 512 KB stripe size
- Our implementation is placed in the ADIO layer of MPICH2-1.0.5

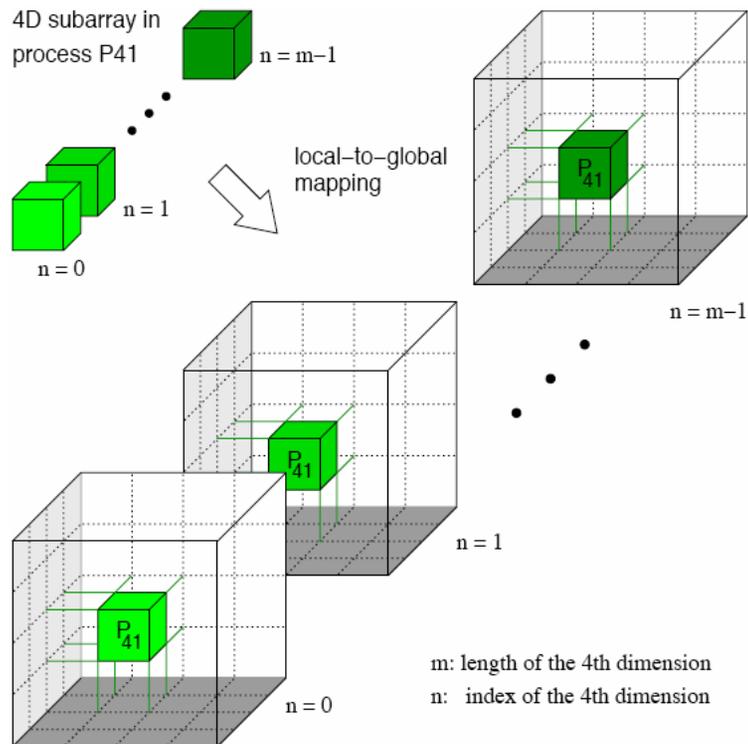
Contact Prof. Wei-Keng Liao for questions

wkliao@ece.northwestern.edu

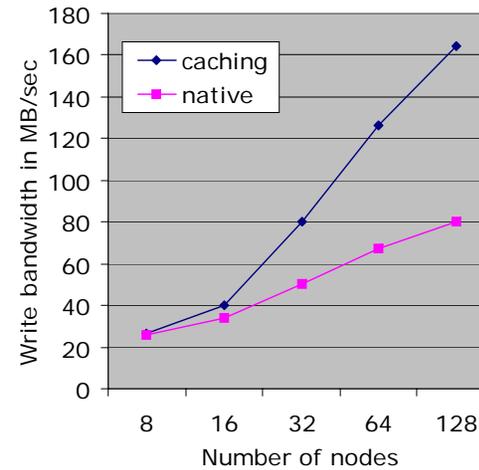
S3D I/O



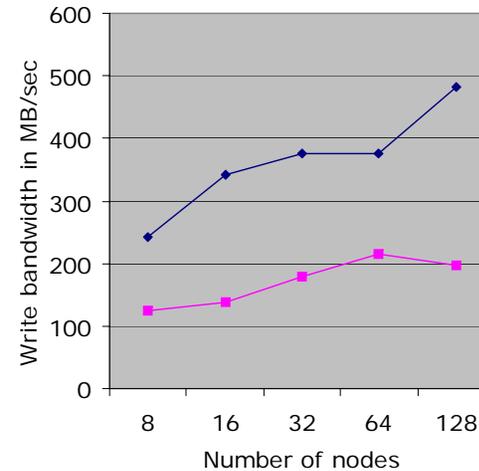
- S3D -- A parallel turbulent combustion application developed at Sandia National Laboratories



S3D I/O on Lustre

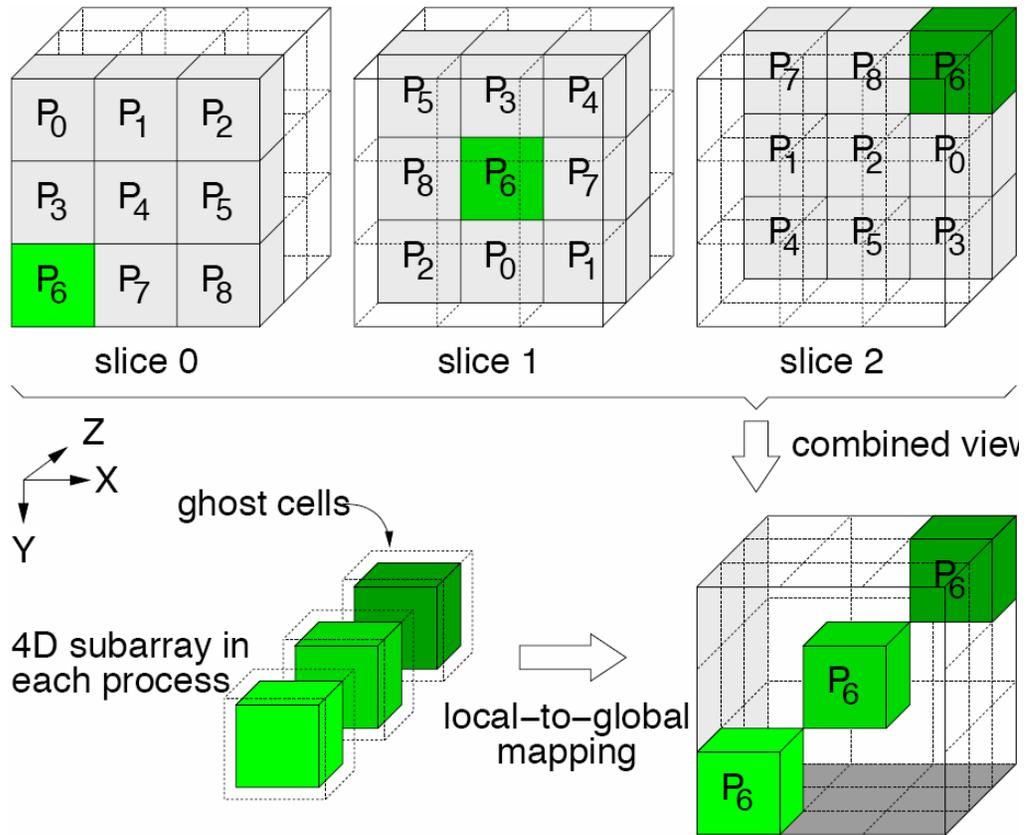
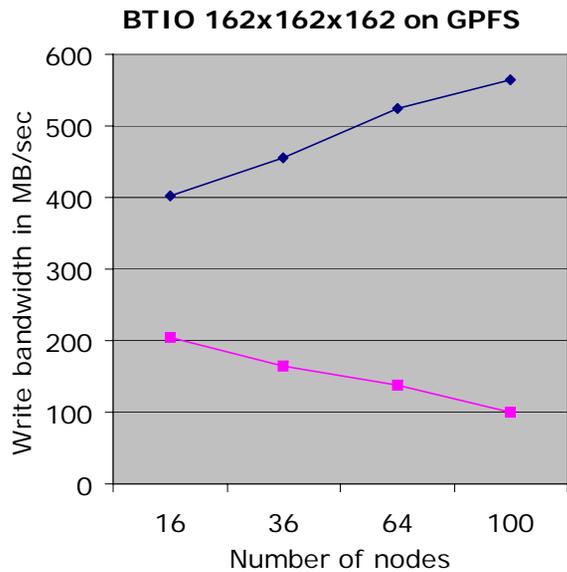
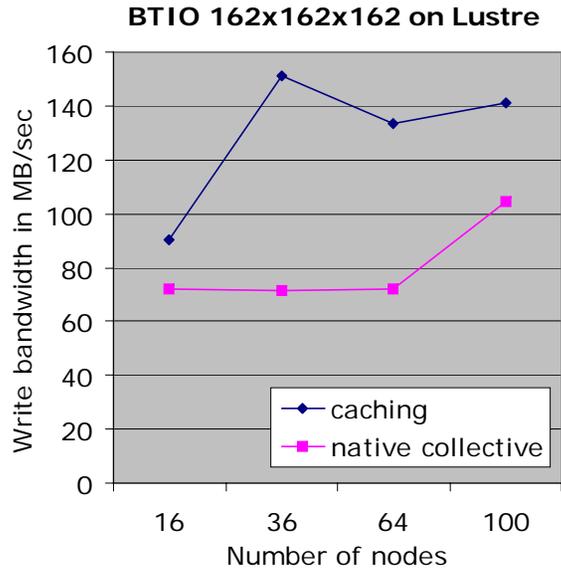


S3D I/O on GPFS

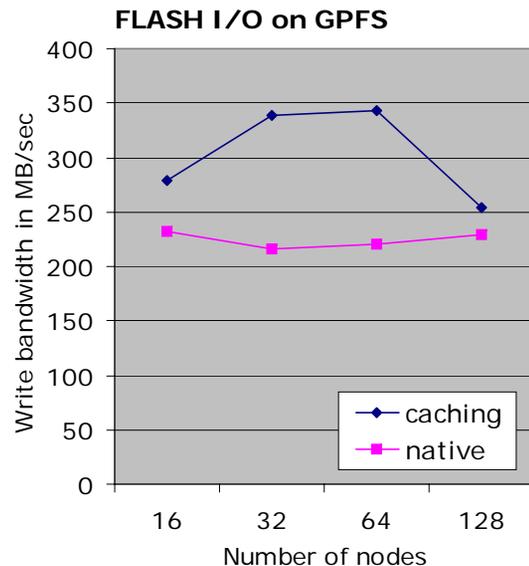
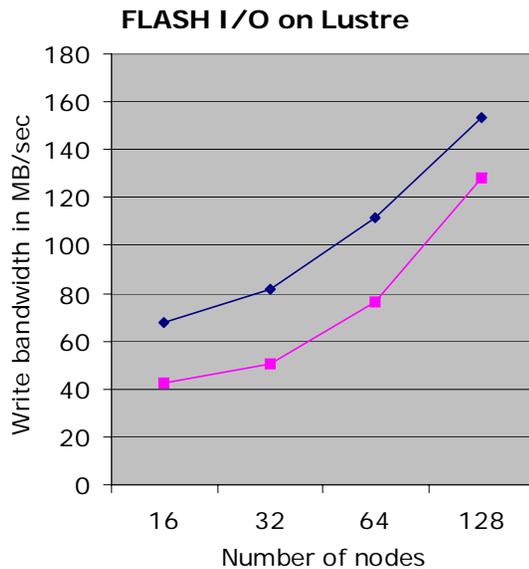


BTIO

- Block tri-diagonal array partitioning pattern
- Run on Lustre at NCSA's Tungsten
- Run on GPFS at NCSA's TeraGrid machine



FLASH - I/O



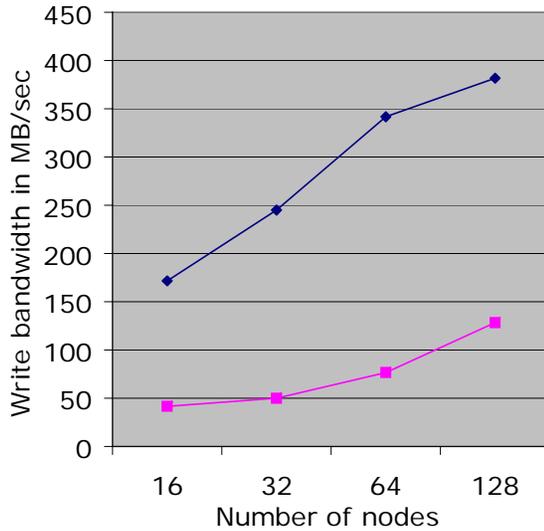
- I/O kernel of the FLASH application, a block-structured code developed for the study of nuclear flashes on neutron stars and white dwarfs
- I/O method: HDF5
- Each process writes 80 arrays
 - Aggregate I/O amount increases as the number of MPI processes
- I/O pattern
 - Non-interleaved writes among processes

Two-stage Write Behind

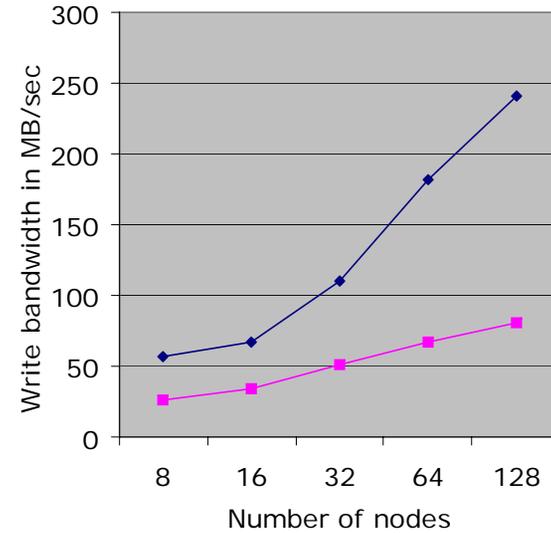
- A Large Number of application I/O patterns are:
 - Write-only
 - Non-overlapping (in byte range)
- Two-stage method
 - Locally in each process
 - Enable write-behind (1st-stage buffering)
 - Globally among all processes
 - Avoid file system lock conflict (2nd-stage buffering)
 - Requirements
 - File is opened in write-only mode
 - MPI atomic mode must be disabled (default mode in MPI)

FLASH I/O and S3D I/O

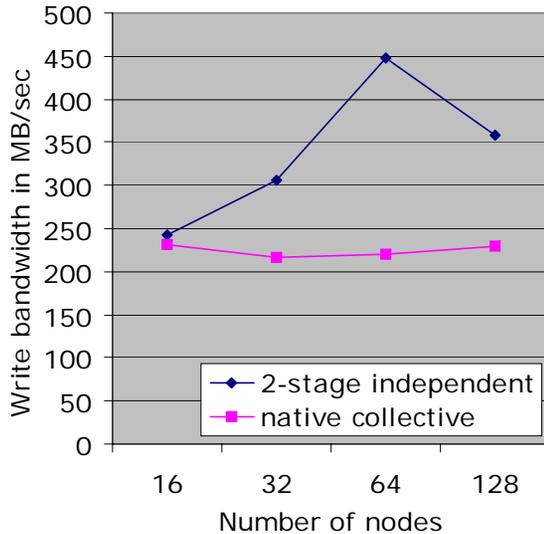
FLASH I/O on Lustre



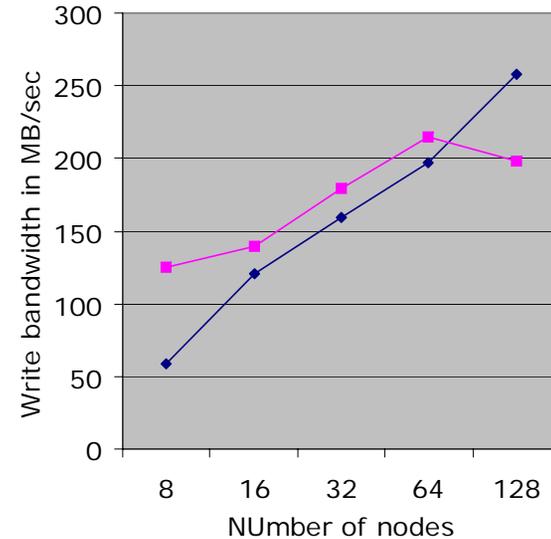
S3D I/O on Lustre



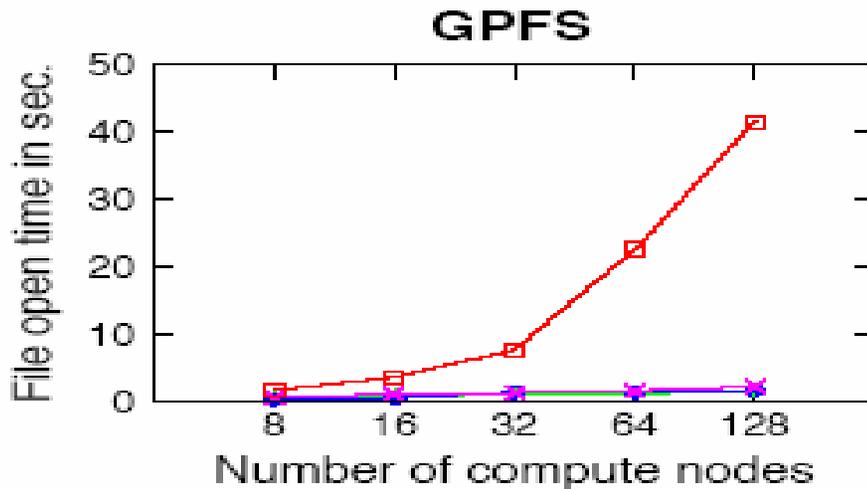
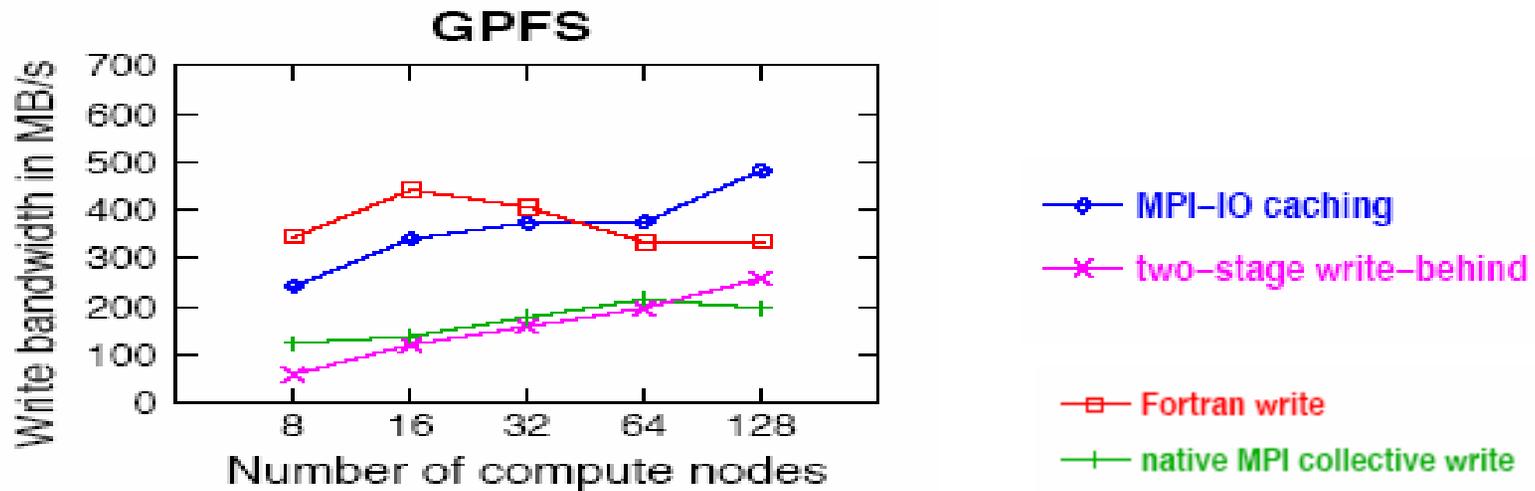
FLASH I/O on GPFS



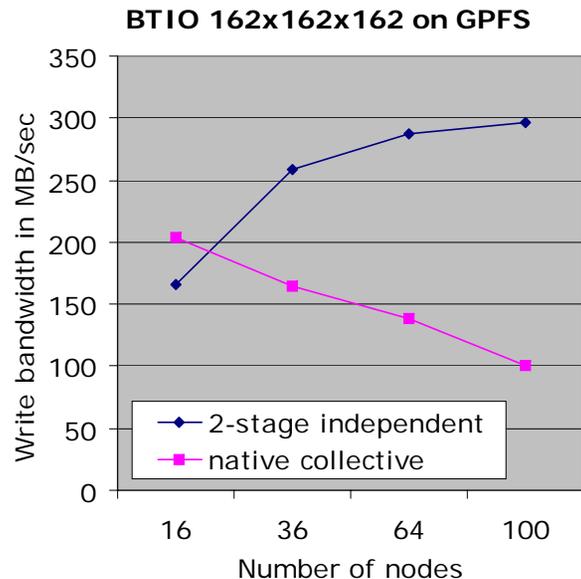
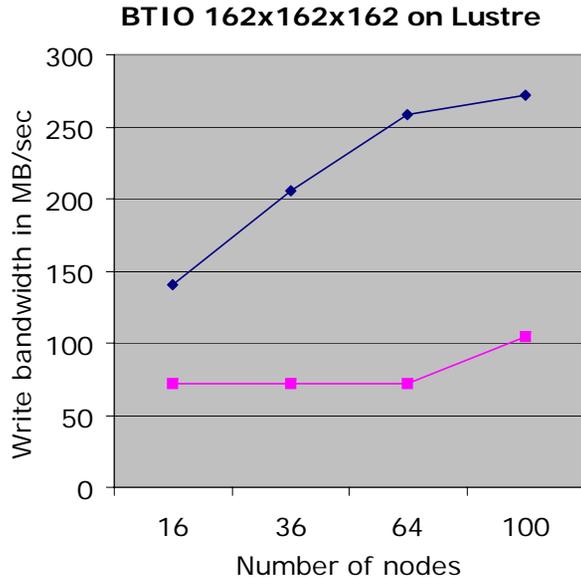
S3D I/O on GPFS



S3D-IO: Performance + Productivity



BTIO



- Native MPI independent write
 - **Not shown**: bandwidth < 5 MB/sec, due to huge number of requests
- Independent writes with two-stage write-behind
 - Dare to compare with MPI collective write
 - Collective I/O is known to outperform independent I/O significantly

Number of write requests per MPI process

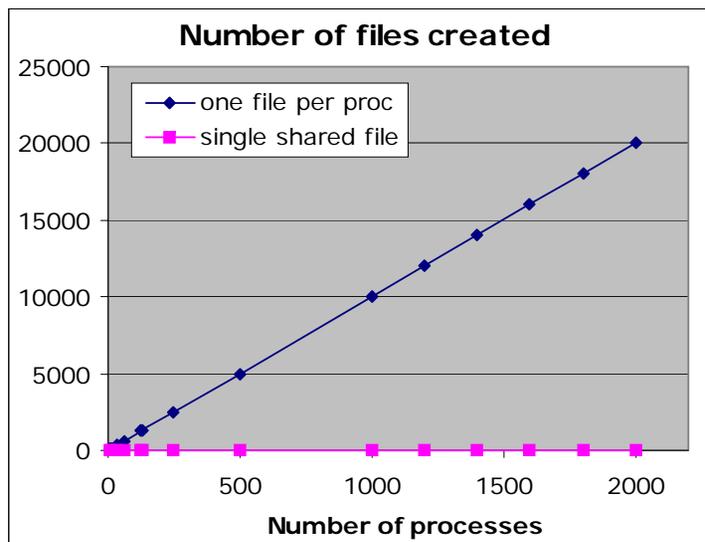
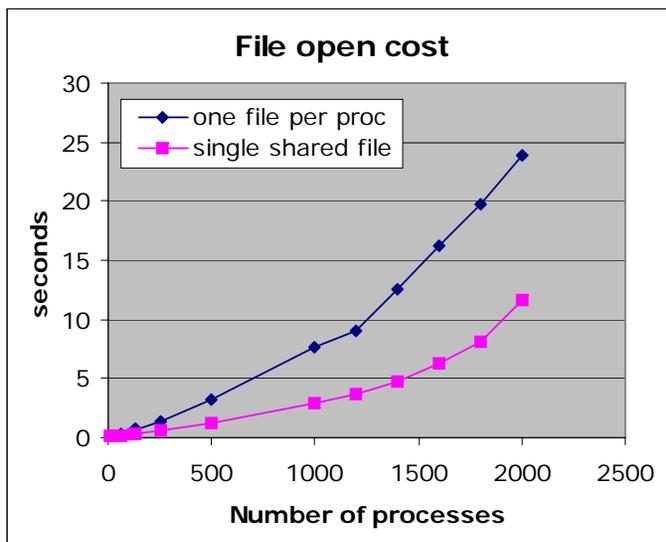
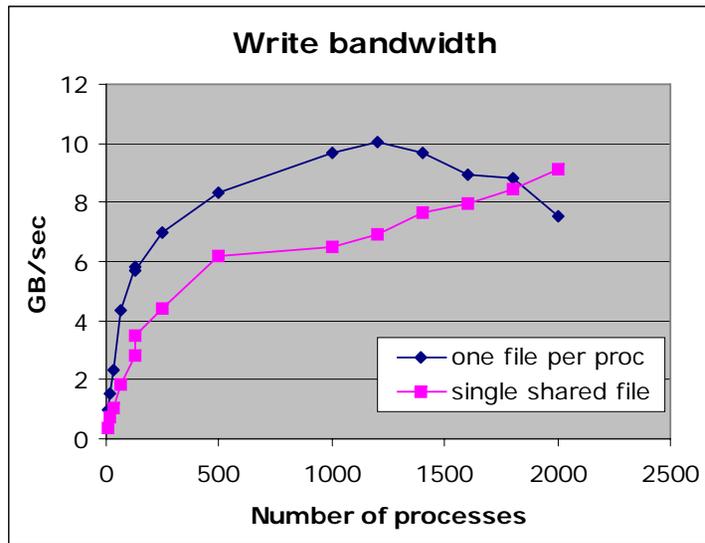
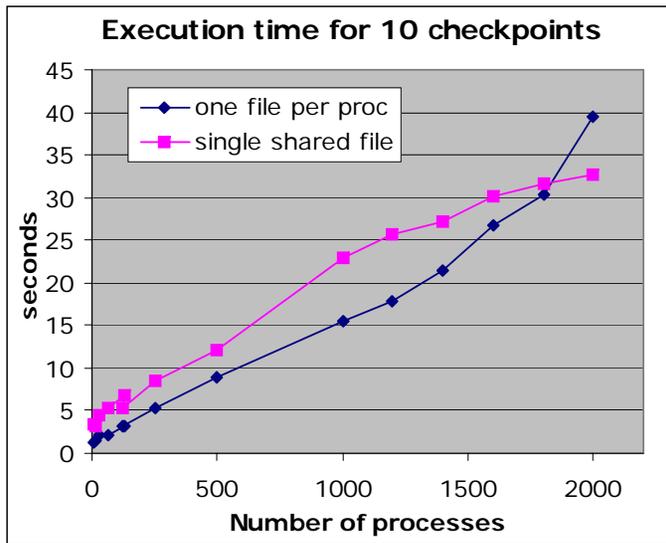
Number of processes	Collective Writes		Independent Writes	
	NWRPP	WAPRPP	NWRPP	WAPRPP
16	40	10.14 MB	262440	1620 B
36	40	4.51 MB	174960	1080 B
64	40	2.53 MB	131240	810 B
100	40	1.27 MB	105000	405 B

NWRPP: number of write requests per process
 WAPRPP: write amount per request per process

S3D-IO on Cray XT3/4 (Performance/Productivity)

- No of files increases linearly with No of processors
- Managing 10s of thousands of files is a SDM and productivity nightmare
- Our initial results are encouraging for scalability, performance and productivity
- Some system software needs to be fixed for us to experiment on larger systems

S3D-IO: Performance + Productivity



Accomplishments

- Thread-based collaborative caching
 - RMA based alternative is under development/testing
- Faculty/students funded
 - 2/2 (for 2 out of three years, 1 student funded for 1 year)
- Collaborators
 - Rob Ross, Rob Latham, Rajeev Thakur @ANL
 - Ramanan Sankaran, Scott Klasky @ORNL
 - Lee Ward, Jackie Chen@SNL
 - IIT (HECURA project) provided with caching software for server-push I/O
- Issues/needs: Most important: Access to larger machines and getting the vendors to fix software when requested, incorporation of our S/W in production S/W etc.

Publications

Published:

- Wei-keng Liao, Kenin Coloma, Alok Choudhary, and Lee Ward. Cooperative Client-side File Caching for MPI Applications. In the International **Journal of High Performance Computing Applications** , Volume 21, Number 2, pp. 144-154, May 2007.
- Wei-keng Liao, Avery Ching, Kenin Coloma, Alok Choudhary, and Lee Ward. An Implementation and Evaluation of Client-side File Caching for MPI-IO. In the Proceedings of the 21st International **Parallel and Distributed Processing Symposium**, Long Beach, California, March 2007.

Accepted:

- Avery Ching, Robert Ross, Wei-keng Liao, Lee Ward, and Alok Choudhary. Noncontiguous locking techniques for parallel file systems. In Proceedings of **Supercomputing**, November 2007 (to appear).
- Wei-keng Liao, Avery Ching, Kenin Coloma, Arifa Nisar, Alok Choudhary, Jacqueline Chen, Ramanan Sankaran, and Scott Klanksy. Using MPI file caching to improve parallel write performance for large-scale scientific applications. In Proceedings of **Supercomputing**, November 2007 (to appear).