...And eat it too: High read performance in write-optimized HPC I/O middleware file formats

Milo Polte, Jay Lofstead, John Bent, Garth Gibson, Scott A. Klasky, Qing Liu, Manish Parashar, Norbert Podhorszki, Karsten Schwan, Meghan Wingate, Matthew Wolf

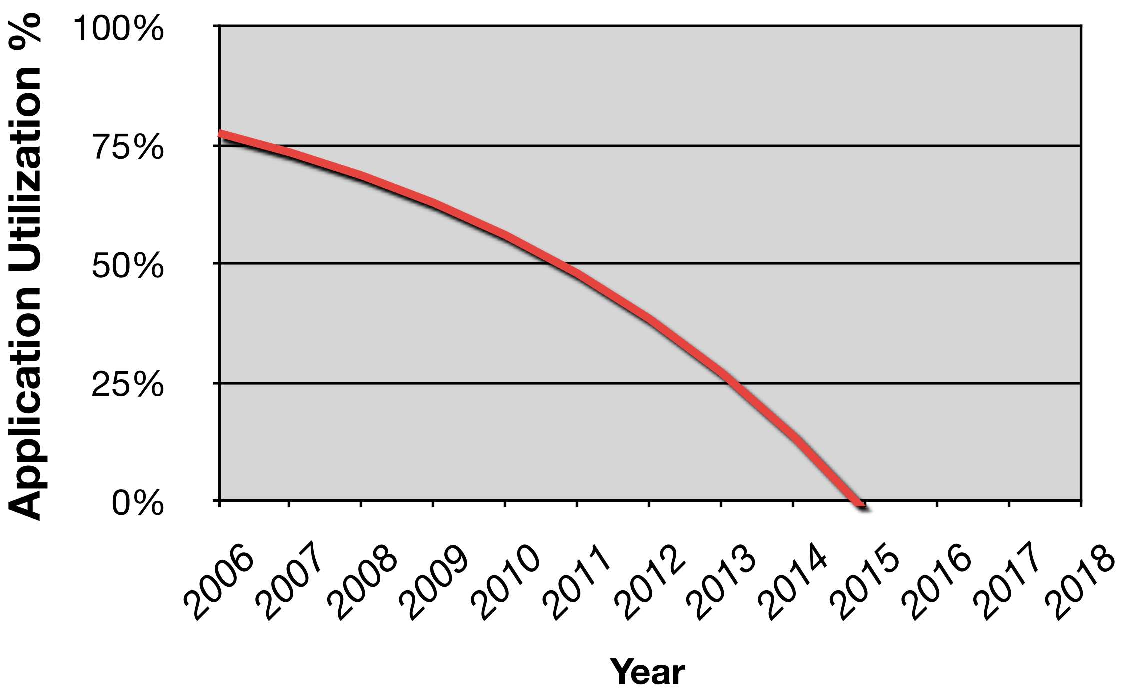
Carnegie Mellon University, Georgia Institute of Technology, Los Alamos National Lab, Oak Ridge National Lab, Rutgers University

HPC systems need faster I/O

- Building bigger computers for bigger apps
  - Jaguar - Over 200,000 cores
  - Roadrunner - Over 100,000 cores

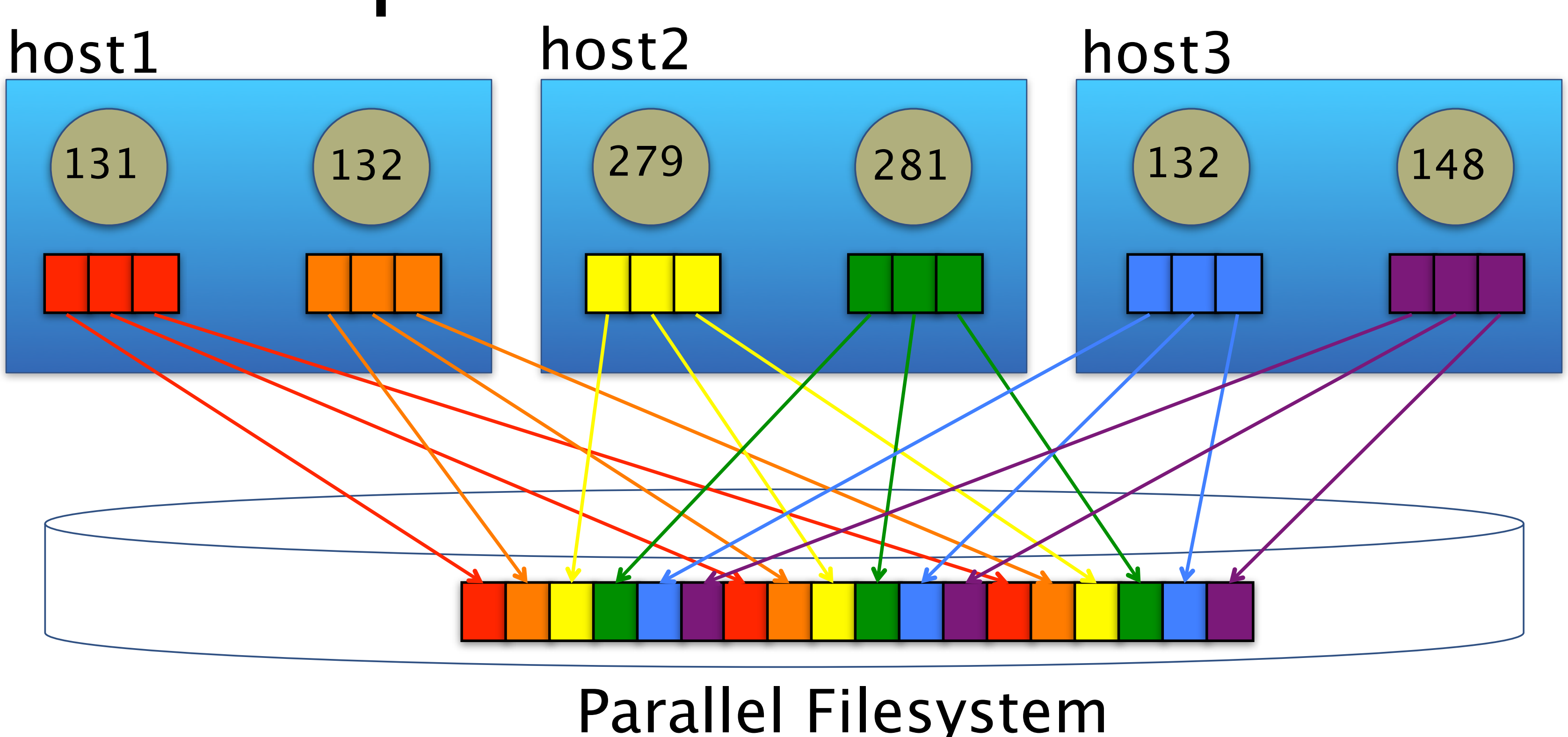
- More data being written by more writers
  - Checkpoints, simulation output
- Having time to compute requires fast output!

Projection of app util. in the face of checkpointing



What makes writing slow?

- Often small, strided writes to a single file (N-1)
- Filesystem lock contention for safety
- RAID parity updates
- Unaligned writes
- Result: Poor spindle utilization



Middleware Layers:
The Right Write Solution

- Write acceleration software between apps and storage
- Typically try to avoid extensive application, filesystem changes
- Usually writes log-structured or "spiritually" log-structured files
- Examples: PLFS, ADIOS, ZEST, LBIO, etc.
- Shown to dramatically improve write speeds

Middleware Layers: The Right Write Solution

- Write acceleration software between apps and storage
- Typically try to avoid extensive application, filesystem changes
- Usually
  log-stru

Writes aren't the subject of the talk today.

itually"

- Examples: PLFS, ADIOS, ZEST, LBIO, etc.
- Shown to dramatically improve write speeds

Outline

- Intro - Writing
- Intro - Reading
- Case Studies:
  - PLFS
  - ADIOS
- Future work
- Conclusions

The importance of reading

- Checkpoint
  - ‘Write Once Read Maybe’
  - Restart time still important
  - Typically read back in entirety
- Simulations
  - Read back for visualization, analysis
  - May read back many times, odd patterns

Log-structured Reading

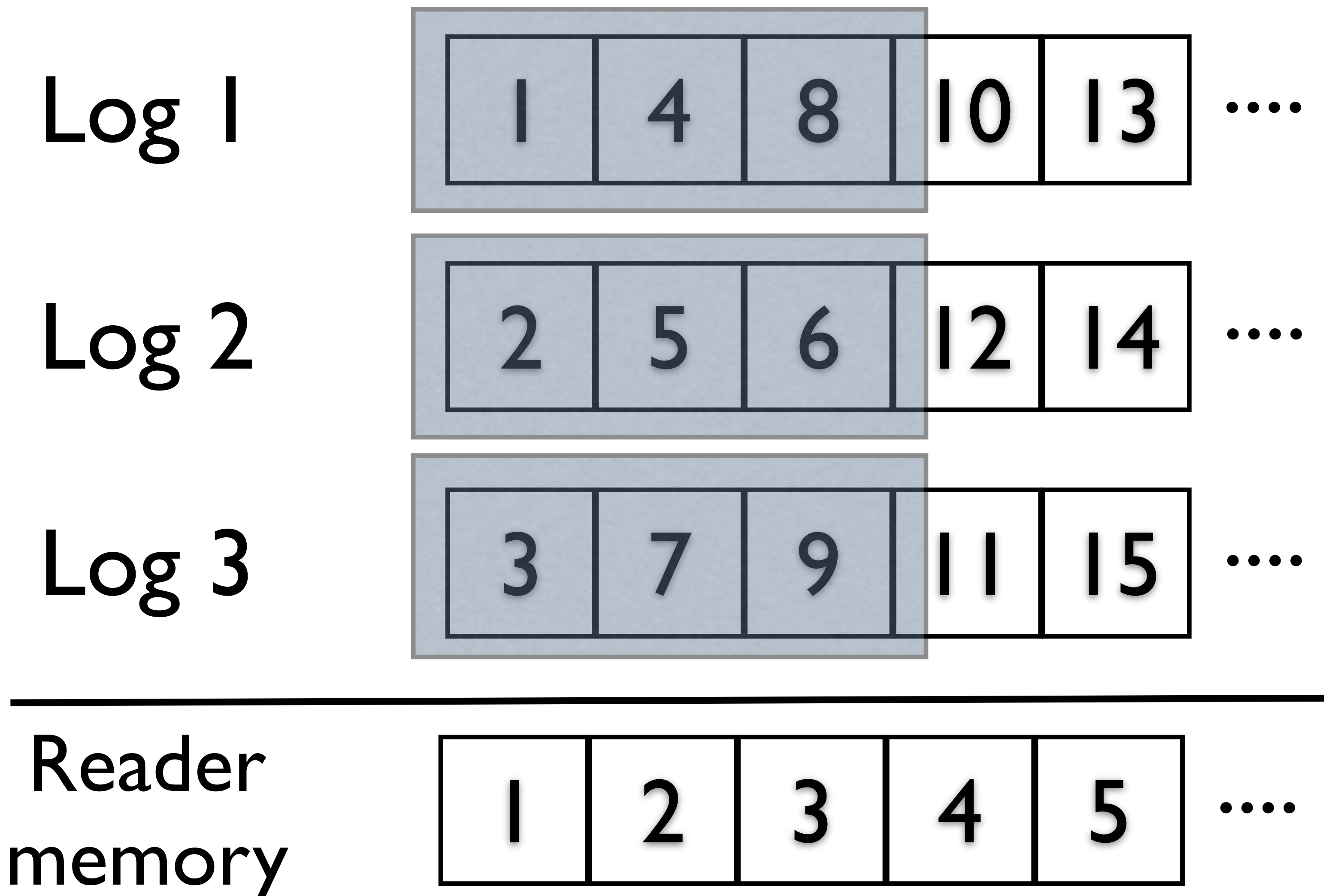
- Recall: middleware layers typically write logs
- Reading log-structured writes can be slow if the reads don't match the write pattern [Rosenblum, 91]
- e.g. reading a randomly written file sequentially:

On disk log:	C	A	D	E	B

Client memory:

When can it go right?

- When logs are read the same as the write pattern
- e.g. Processes writing monotonically increasing logical offsets, one reader reads sequentially



Numbers are logical offsets

Grey boxes are read ahead buffers

When can it go right?

- Example: Readers have same access pattern as equal number of writers
- Example: Writers write monotonically increasing offsets; readers read sequentially
- Typical of checkpoint restarts, archiving
- What about restarting from a different number of writers?
- What about analysis workloads?

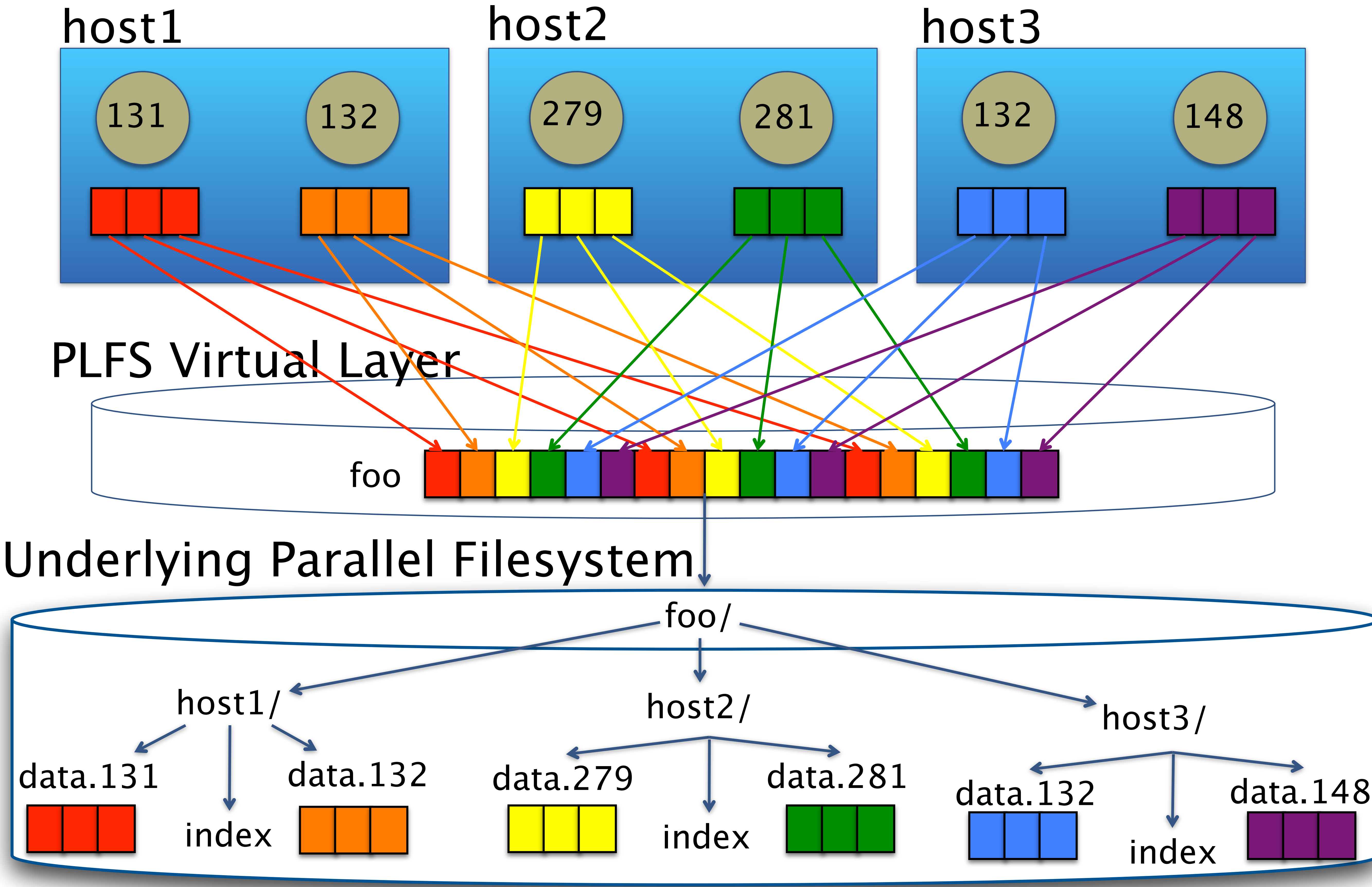
Our Investigations In Reading

- For log-structured middleware layers:
- Validate good performance on 'uniform' restart
- Examine the more challenging case of restart on a different number of processes
- Do so with two case studies: PLFS, ADIOS

PLFS Design

Requirement	Solution
Extreme parallelism	Decouples writers to individual files
Fast, efficient writes	Writes in a log structured manner
No application changes	Exposes POSIX filesystem interface
Portable across filesystems	Implemented as a 'stackable' FUSE filesystem
Low comp. node footprint	Uses existing parallel FS storage

PLFS Design

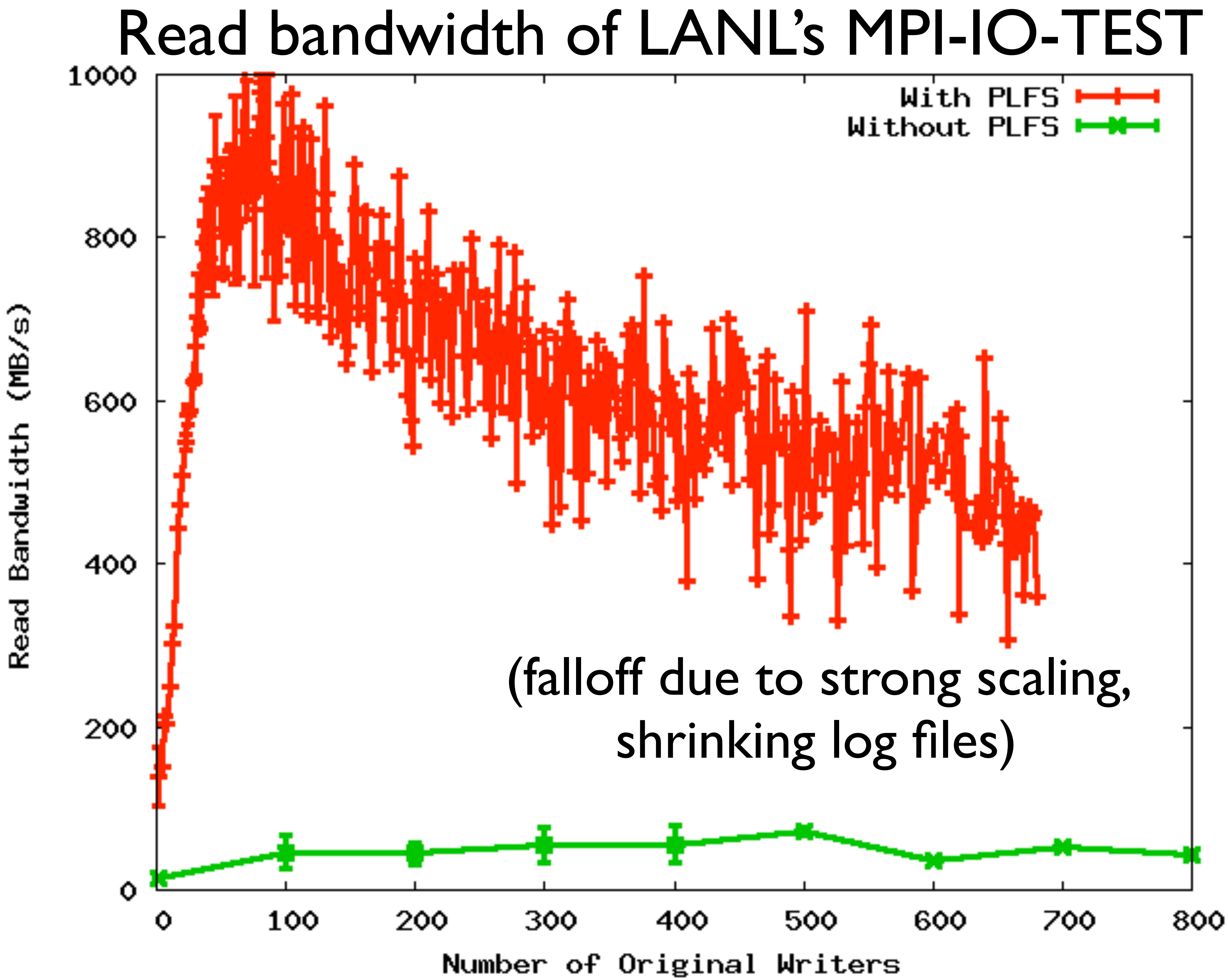


host1
host2
host3
131
132
279
281
132
148
PLFS Virtual Layer
foo
Underlying Parallel Filesystem
foo/
host1/
host2/
host3/
data.131
index
data.132
data.279
index
data.281
data.132
index
data.148

PLFS Testing Setup

- Run on Roadrunner at LANL
  - Storage was PanFS filesystem
- Checkpoint benchmark: MPI_IO_TEST
  - Can run both N-1 and N-N checkpoints
- 20 GB checkpoint file written in 47KB strides
- Compare read back performance with and without PLFS

PLFS: Uniform Restart

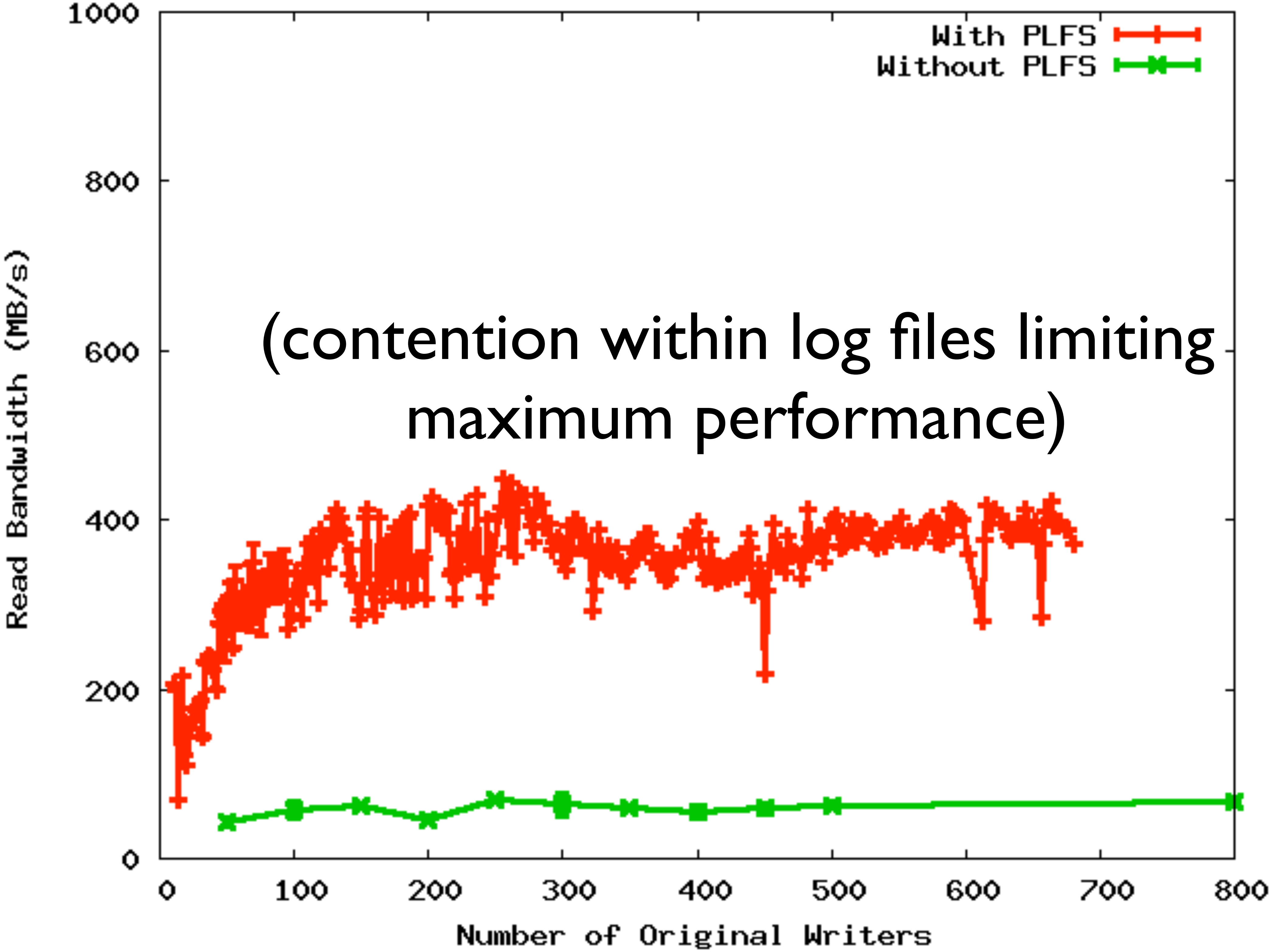


Why is reading directly so much worse?

- Due to strided reading pattern, most accesses in same region of single file
- Uses smaller number of disks at once
- Reading from multiple PLFS log files uses many spindles, read ahead buffers at once

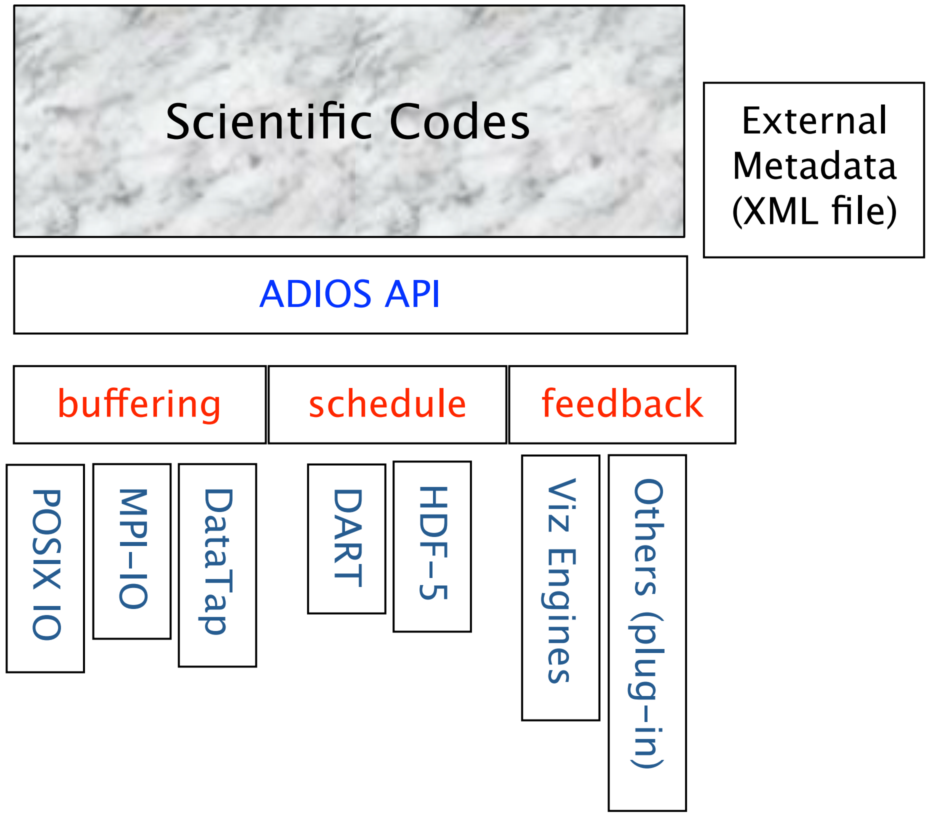
PLFS: Non-uniform Restart

Read bandwidth of LANL's MPI-IO-TEST



(contention within log files limiting maximum performance)

ADIOS Design



- Library used in place of other common middleware layers
- Simple API for Fortran and C
- Pluggable layers for writing through different file formats without code changes
  - ADIOS-BP, netCDF, HDF-5
  - Hooks into asynchronous and synchronous I/O
- Free hooks into visualization and workflow systems through the data flows
- Optimized IO implementations provided for each transport method (e.g., MPI-IO, HDF-5, etc.)
- Binary, tagged format provided by default: “Binary Packed”

BP File Format

- Each process writes independently (a ‘process group’ (PG))
- Limited coordination
- File organization more natural for striping
- Rich index contents (PGs, vars, and attributes)
  - Easy, indirect access to any element
- Local data written to PG’s, annotations used to reconstruct global objects

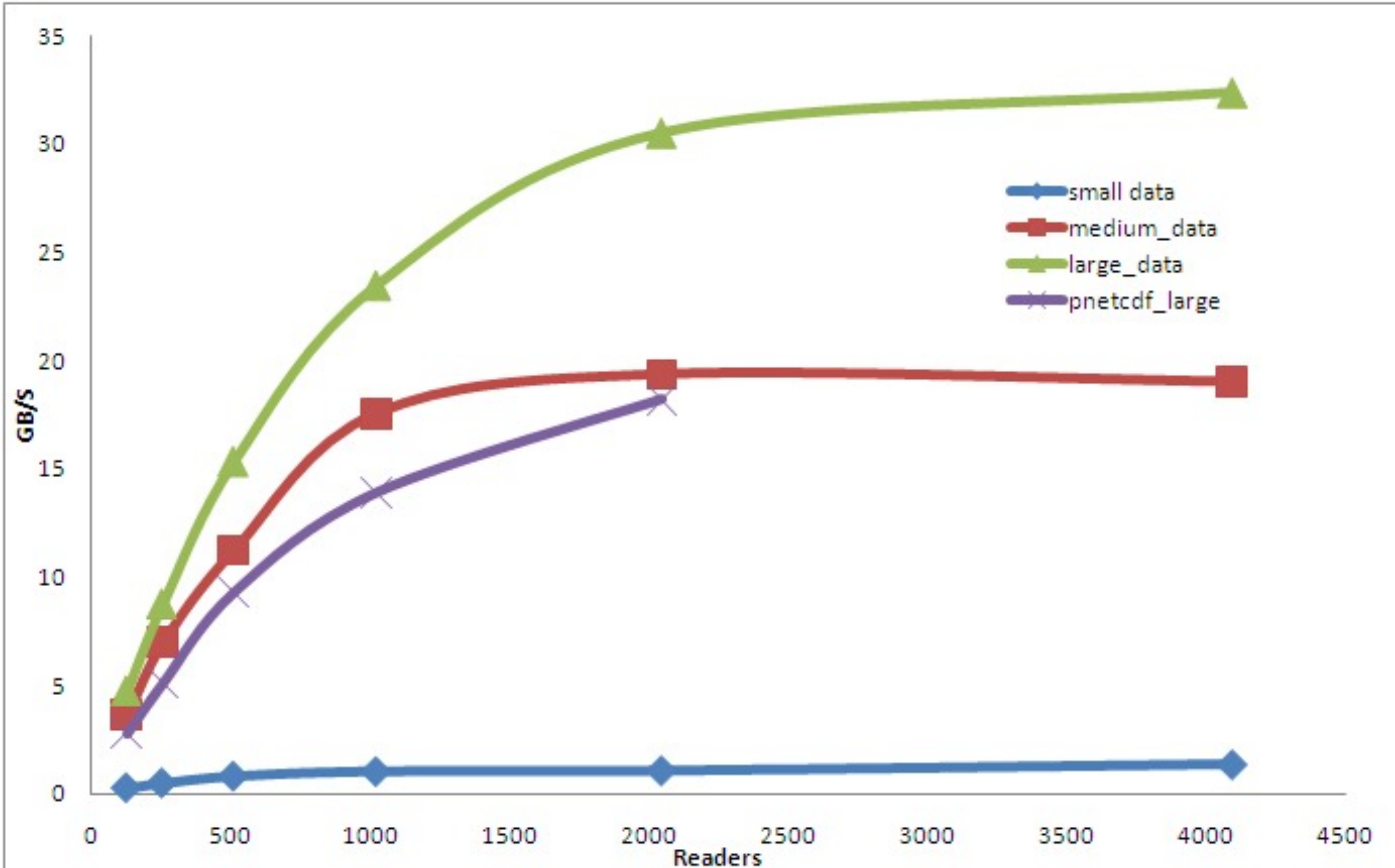
Process Group 1	Process Group 2	...	Process Group n	Process Group Index	Vars Index	Attributes Index	Index Offsets and Version #

ADIOS Testing Setup

- Run on Jaguar XT4 at ORNL with Lustre storage
- ADIOS using MPI writing to a BP format
- Parallel netCDF 1.1.0 as control
- Pixie3D IO Kernel, a 3D MHD fusion code
- 3-D domain decomposition
  - LOTS of memory reorganization for non-log-based formats
  - Looked at small, medium, and large data sizes

ADIOS: Uniform Reads

Read bandwidth of Pixie3D

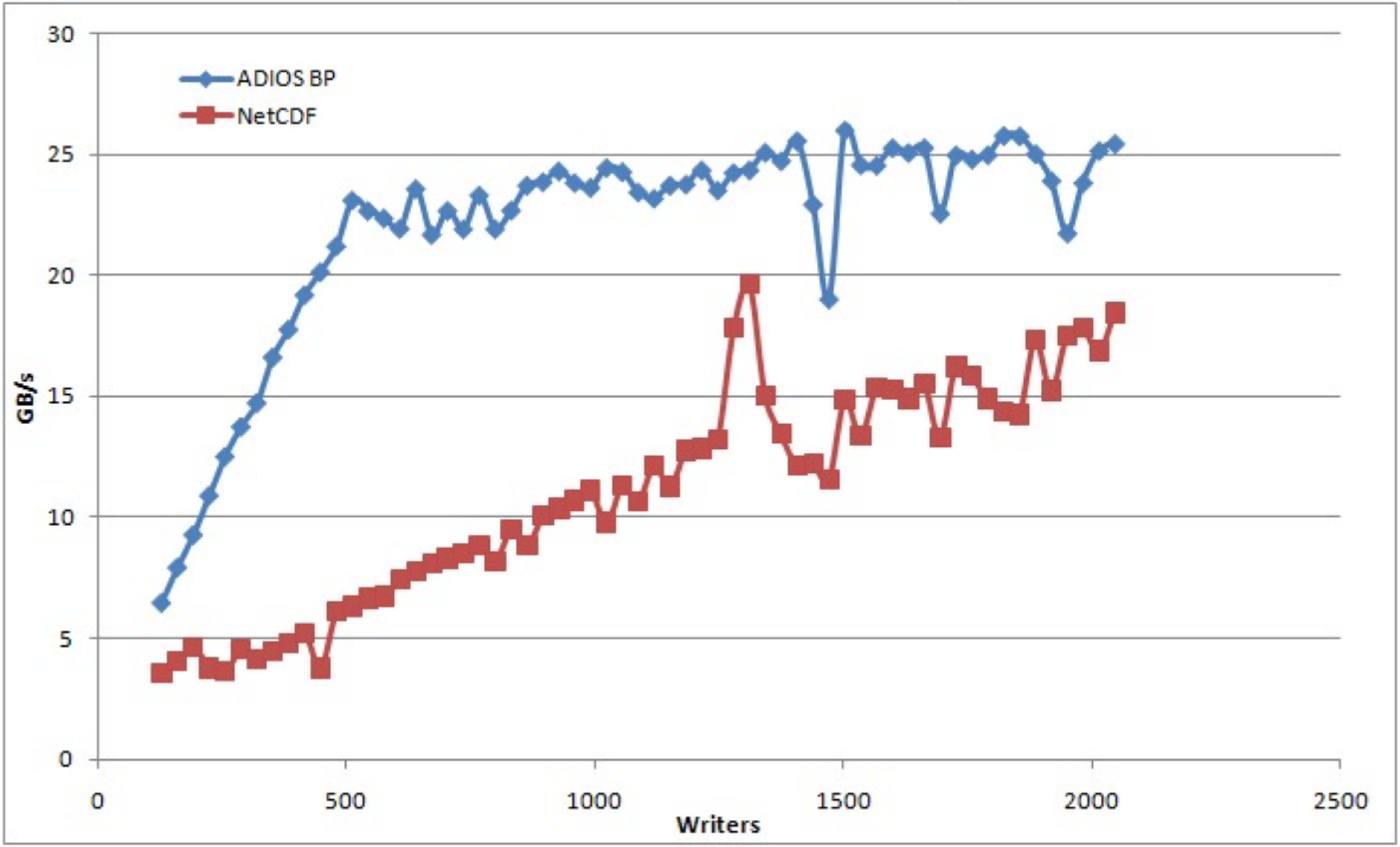


Why Does ADIOS Read Better?

- Reading data back in without having to shuffle as much gives fewer, larger reads
- No reorganization necessary for uniform restarts means no cost to write or read back in

ADIOS: Non-uniform Reads

Read bandwidth of Pixie3D, Large Data Size



Future Work

- More apps, analytic workloads
- How important is domain knowledge?
  - ADIOS knows about variables, PLFS only knows bytes
- If read back is slower, does write benefit still represent a net gain?
- How many times are files read after being written?

Conclusions

- I/O Middleware layers provide a large benefit to application write speeds
- Despite a log structured format, they also more efficiently utilize filesystem resources for reading
- This seems to be true for both uniform and non-uniform restarts and full reads
- Further research is planned into how these formats affect data analysis workloads