PLFS: Parallel LFS

John Bent, Garth Gibson, Gary Grider, Ben McClelland,
Paul Nowocynzki, Milo Polte, Meghan Wingate

LA-UR-08-07314, LA-UR 09-02117, LA-CC-08-104

LANL Computational Science

- Lots of tightly coupled parallel simulations
  - Weapons design and verification
  - Bioscience
  - Astrophysics
- Require large computers w/ low latency interconnects
  - Currently at a petaflop
  - Simulations always want MORE resolution
  - Already designing exaflop machines

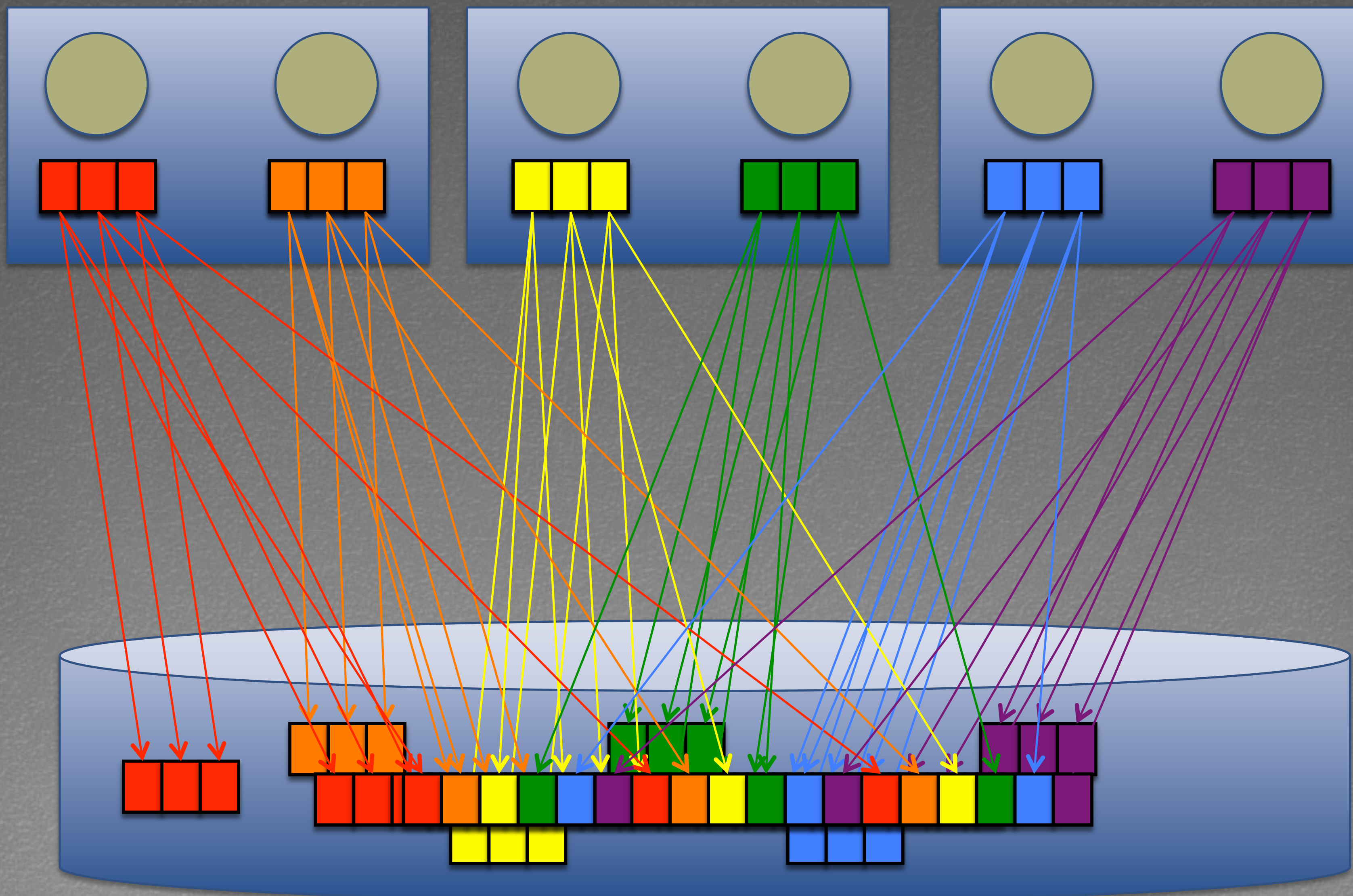
Roadrunner

- LANL's petaflop supercomputer
  - First to petaflop! (sort of)
- 3060 compute nodes
  - Quad-core opterons with cell accelerators
  - Low latency infiniband for IPC
  - High bandwidth ethernet for data storage
- 5 miles and multiple tons of networking cables

Parallel Apps do Parallel IO

- Large distributed systems are not free
  - Some component is always about to fail
- Periodic checkpoint writes
  - Also visualization writes
- Writes are synchronized
- Tens of thousands of synchronized writes can be difficult for the file system
- Two most common write patterns
  - N-1 where N procs write to 1 shared file
  - N-N where N procs write to N non-shared files

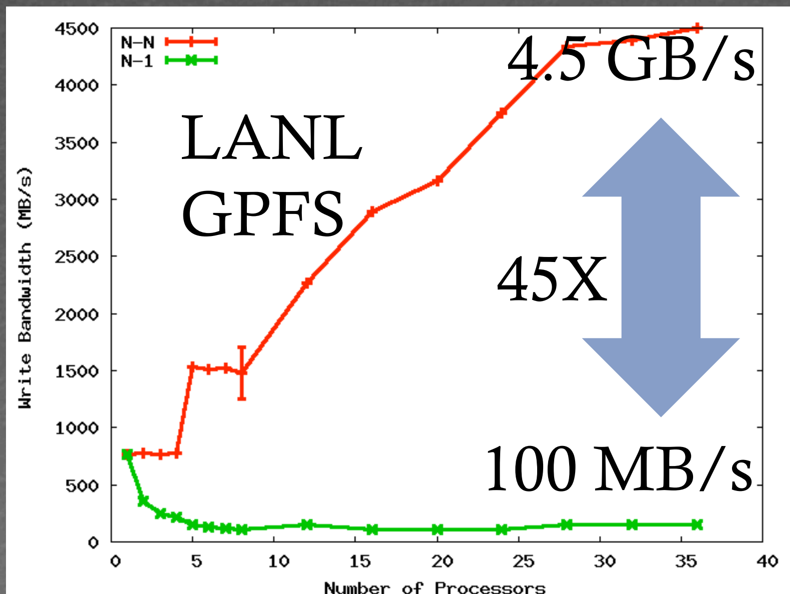
N-1 Strided File IO
N-1 Segmented File IO

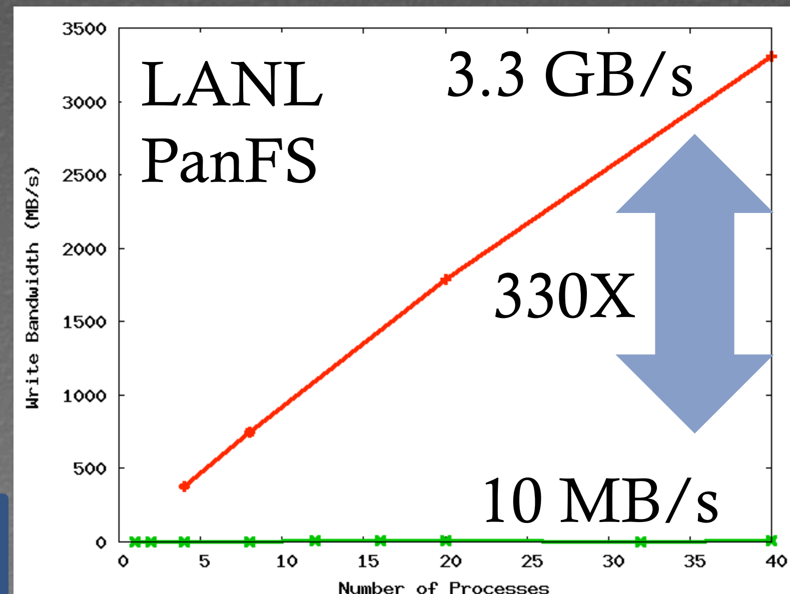


Checkpoint Patterns

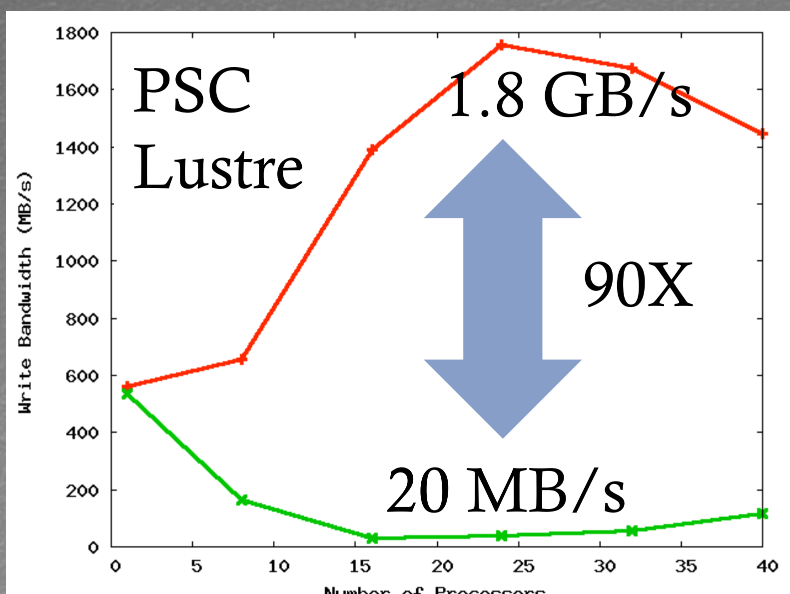
- N-N
  - Writes and reads easy for file system
  - Opens can be hard
  - Hard for application and user
    - Archiving, non uniform restart, viz, etc.
- N-1 Segmented
  - Writes and reads slightly harder for FS
  - Opens easier
  - A little easier for the application and user
  - Rare in practice
- N-1 Strided
  - Writes and reads very hard
  - Easy for application and user
  - Common pattern at LANL and elsewhere

A Shared File is a Shared Problem



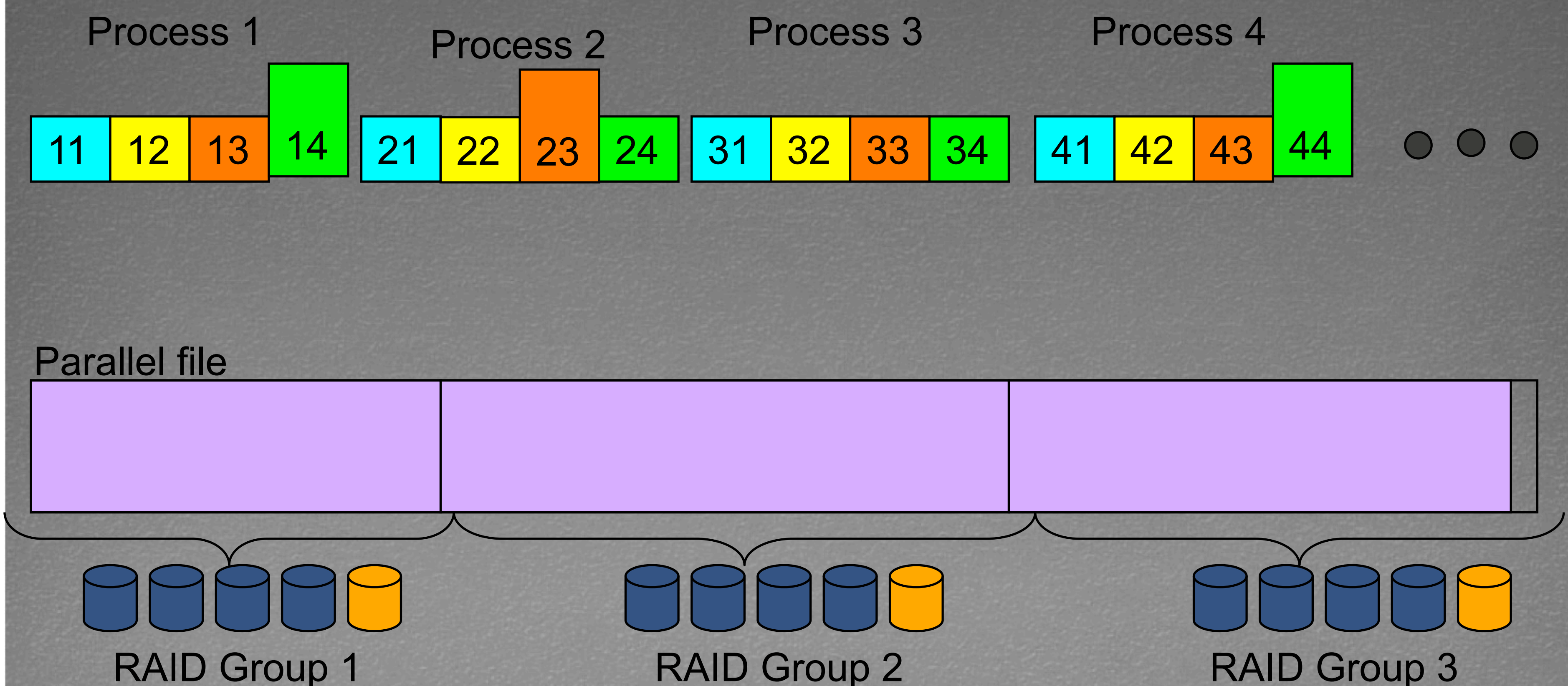


N-N
N-1



Cross graph comparisons not meaningful

Potential PanFS storage implications of N-1 strided



N-1 is prominent

- Several old LANL codes use N-1 (over 50% of cycles)
- Newly written codes still choosing N-1
  - 2 of 8 open science applications on Roadrunner
  - NetCDF and HDF5 formatting libraries
- N-1 also prominent elsewhere
  - At least 10 of 23 on the PIO benchmarks page are N-1
  - BTIO, FLASH IO, Chombo IO, QCD, etc. (GTC?)

Obvious solution: Convert N-N into N-1

- But many applications won't do it
  - Archiving, mgmt, visualization, non-uniform restart
  - Developers are aware of the N-1 problems
    - But are loathe to change to N-N
    - One app wrote 10K lines of code, bulkio, to try to improve N-1
- If the apps won't do it, interposition can
  - Desirable characteristics
    - Low overhead (performance and resource)
    - User transparency (i.e. NO CODE REWRITING)
    - Portable and maintainable
  - Our contribution: **PLFS**

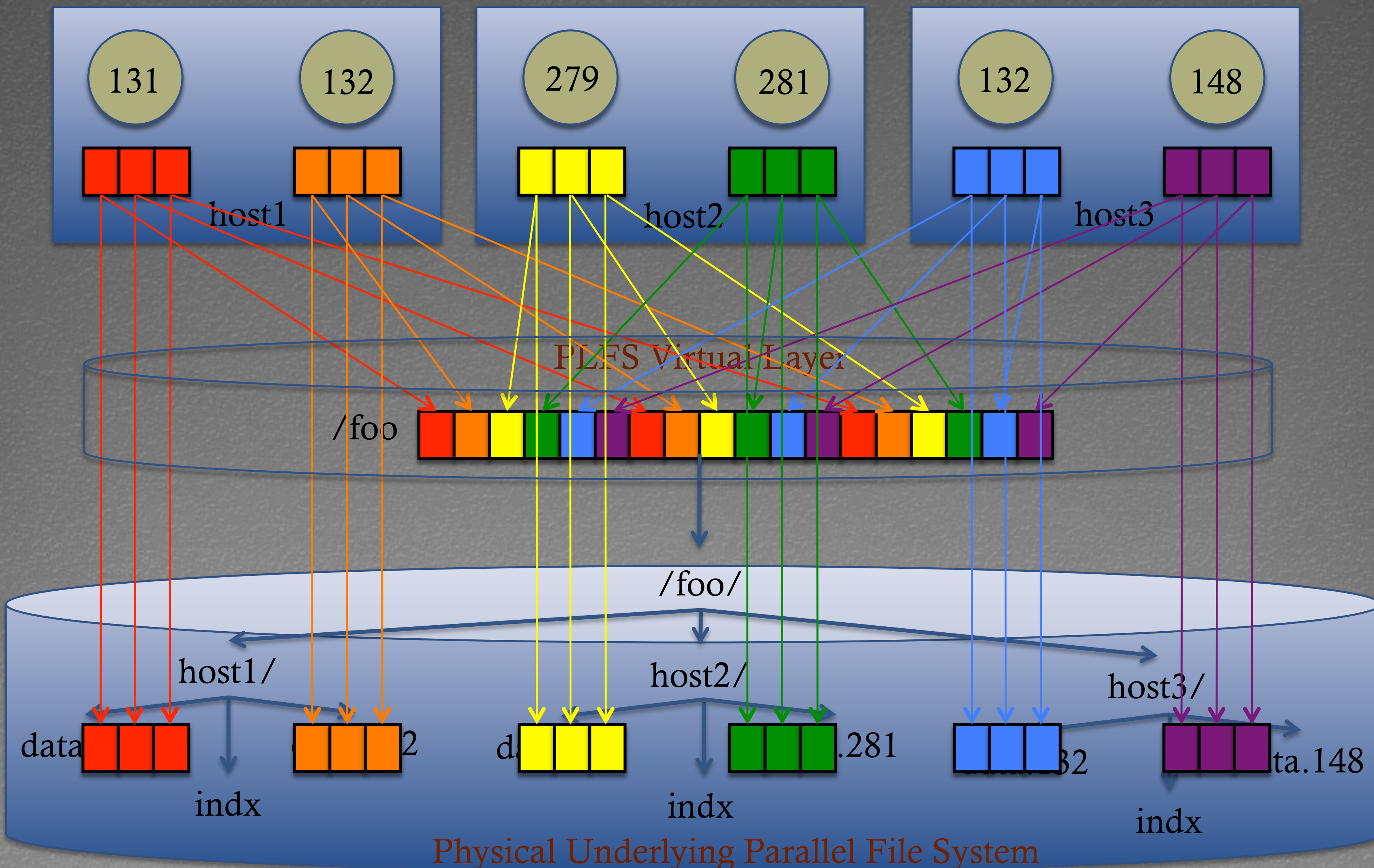
Outline

- Introduction
- PLFS Design and Implementation
- Evaluation
- Trade-offs
- Related Work
- Future Work and Conclusions
- Other outstanding problems in HPC

PLFS:
Parallel Log-structured FS

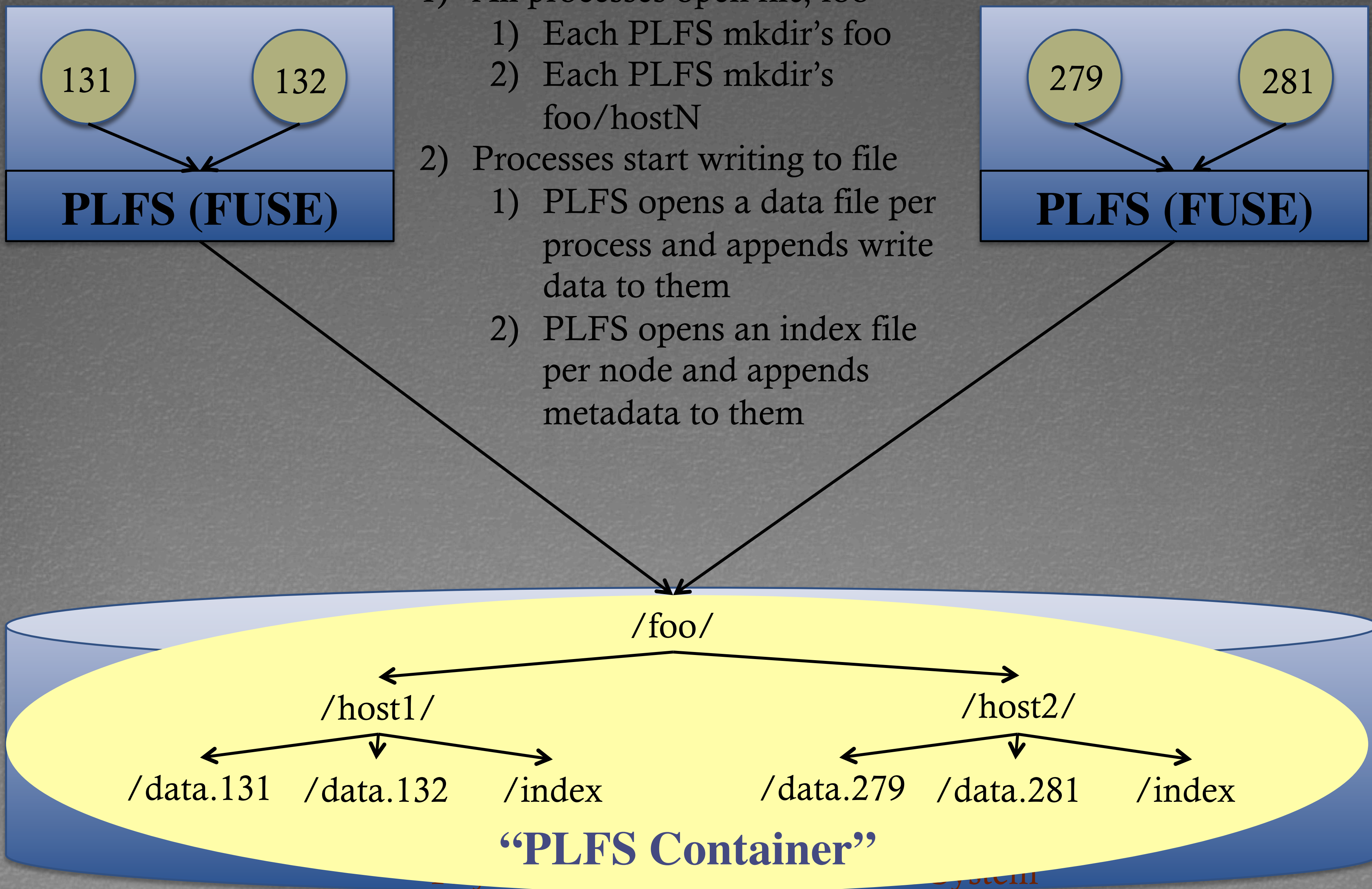
- Virtual interposition file system using FUSE
- Transparently rearranges N-1 checkpoints into N files
  - Very similar to Lustre Split Writing
- Two main optimizations
  - Decouples concurrent access
  - Append-only writing

Decouples Logical from Physical



Data Reorganization in PLFS

1) All processes open file, foo
   1) Each PLFS mkdir's foo
   2) Each PLFS mkdir's foo/hostN
2) Processes start writing to file
   1) PLFS opens a data file per process and appends write data to them
   2) PLFS opens an index file per node and appends metadata to them



PLFS Index Record

Data ID	**Phys Off**	**Len**	**TS Begin**	**TS End**	**???**

- Sort records by physical offsets
  - Lookup map
- Sort records by timestamps
  - IO Trace

Other operations in PLFS

- Writes are much better but
  - Overall only improved if other ops not much much worse
- Reads
  - Construct a global index by aggregating all the index files
  - Map logical offsets to a physical offset within a data file
  - Overlapping writes are undefined
- Chmods, Chowns, Chgrps, Utimes, etc.
  - Use a container/access file
- Stats
  - Pull permissions, ownership from access file
  - Construct a global index to get file capacity and file size

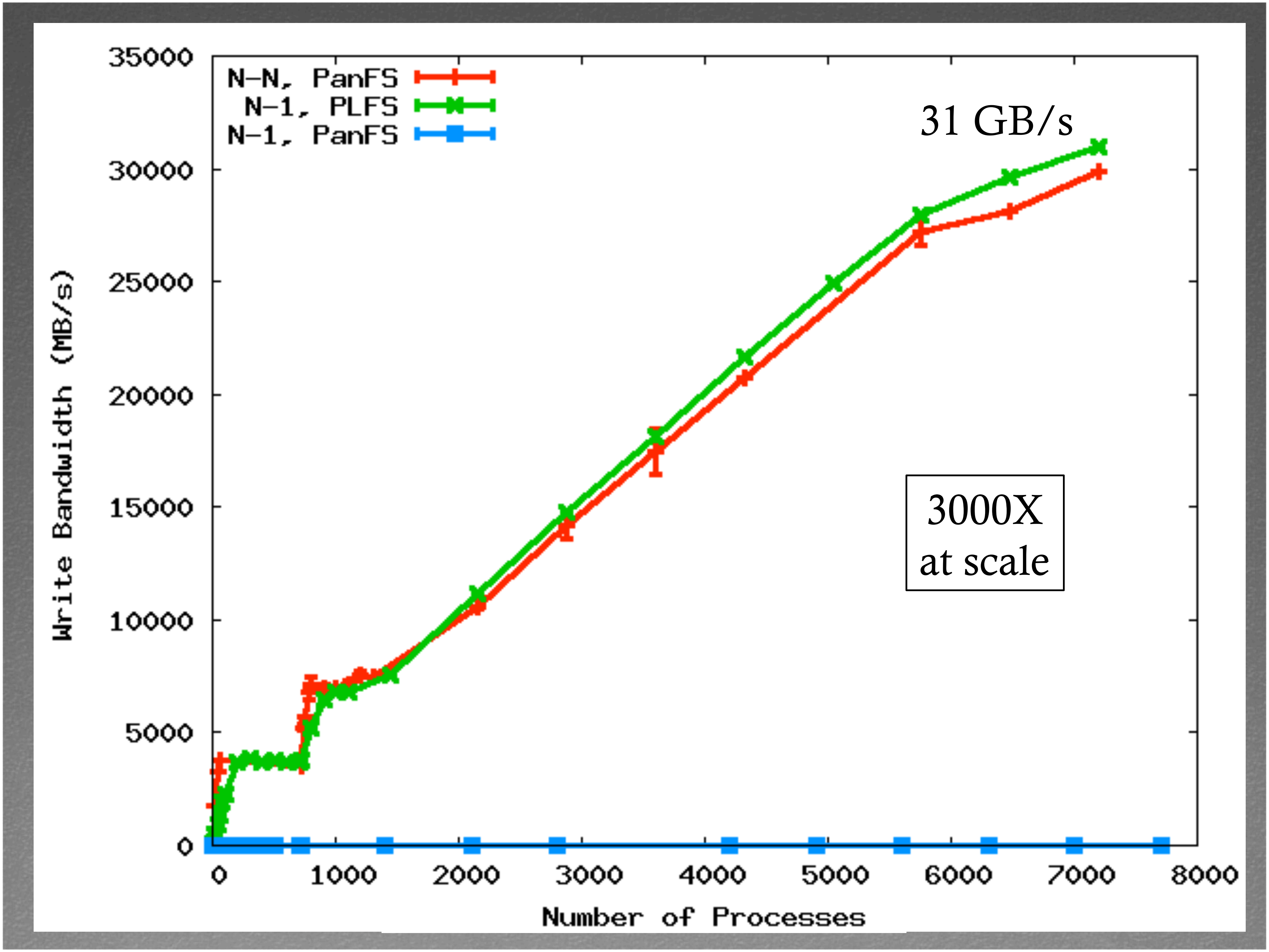
!!!! Constructing a global index can be SLOW !!!!

PLFS Optimizations

- Reads
  - When possible (i.e. O_RDONLY), construct global index on the open, reuse for each read call
- Stats
  - On close, create a container/metadata/host.B.L.T
    - B = blocks of capacity
    - L = last offset (i.e. file size)
    - T = timestamp of last write
  - Stat can be implemented with a readdir
  - Invalidate cache on subsequent re-opens

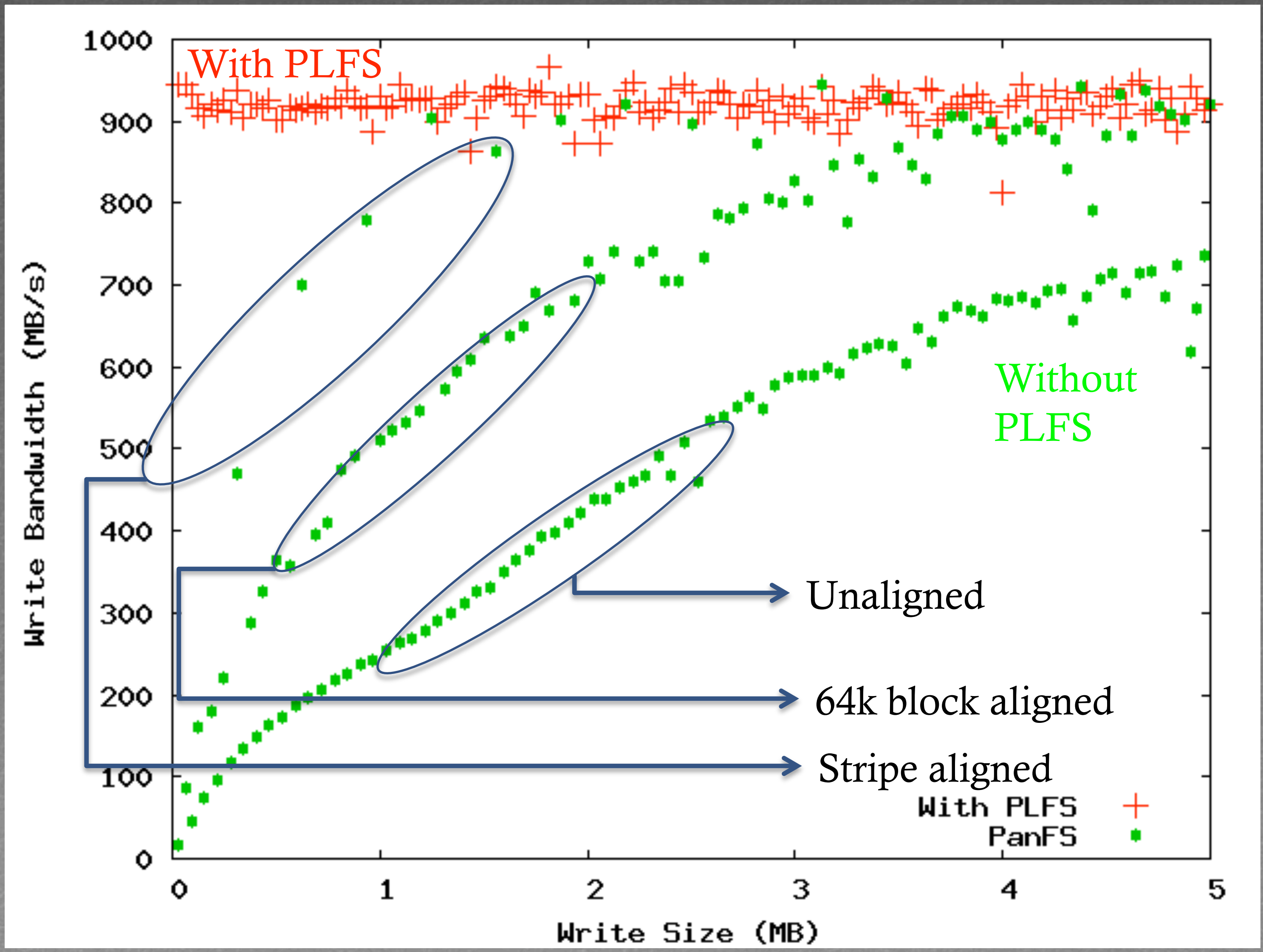
Thoroughly Evaluated

- File Systems
  - GPFS
  - Lustre
  - Panfs
- Synthetic Checkpoint Benchmarks
  - LANL MPI-IO test
  - NERSC Pattern-IO
- Applications and IO Kernels
  - LANL1, LANL2, LANL3
  - Office of Science
    - FLASH-IO benchmark with HDF5
    - Chombo-IO benchmark with HDF5
    - QCD QIO
  - NASA BT-IO benchmark



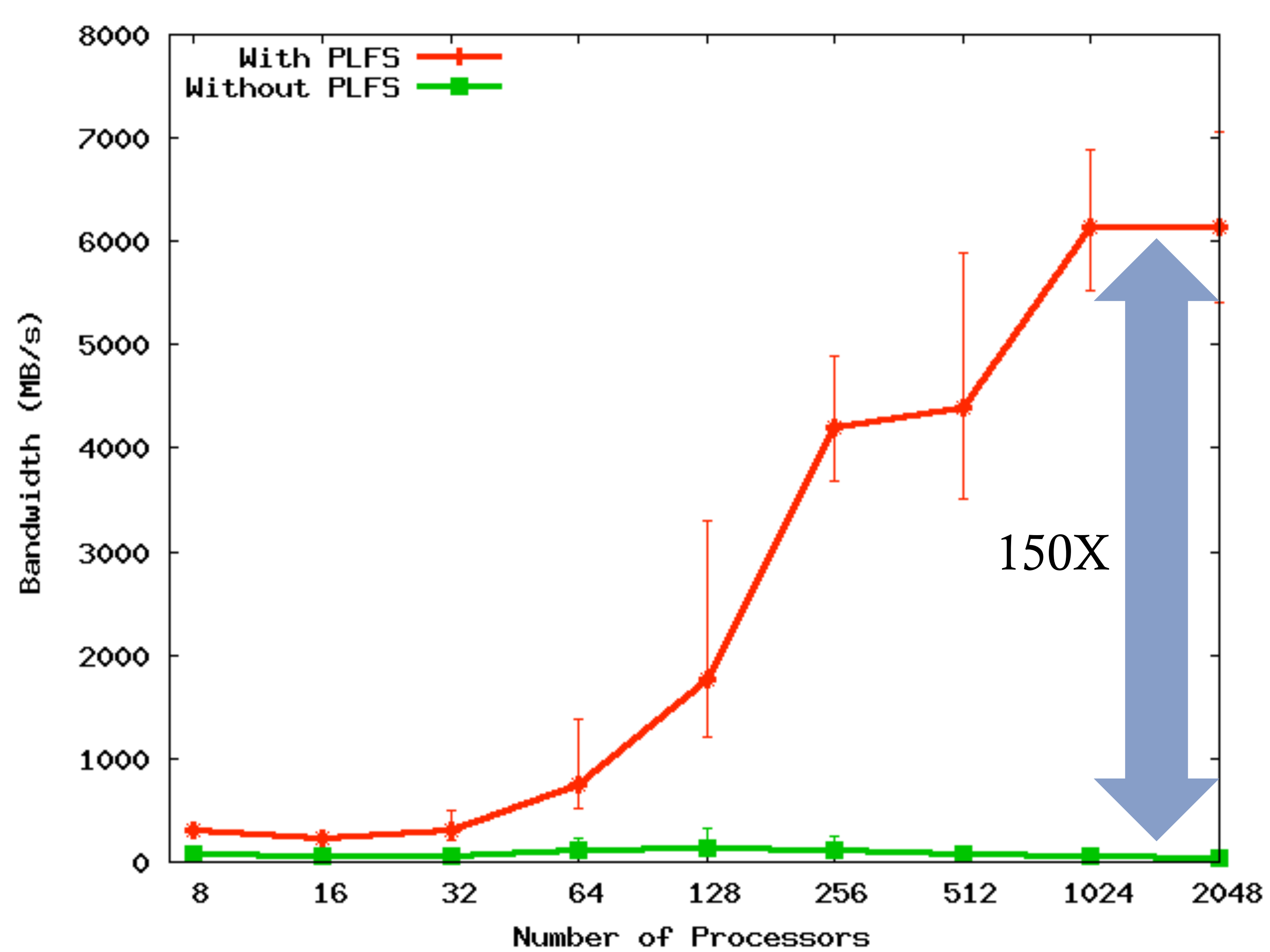
N-N, PanFS
N-1, PLFS
N-1, PanFS
31 GB/s
3000X
at scale
Write Bandwidth (MB/s)
Number of Processes
35000
30000
25000
20000
15000
10000
5000
0
0
1000
2000
3000
4000
5000
6000
7000
8000

LBNL PatternIO benchmark

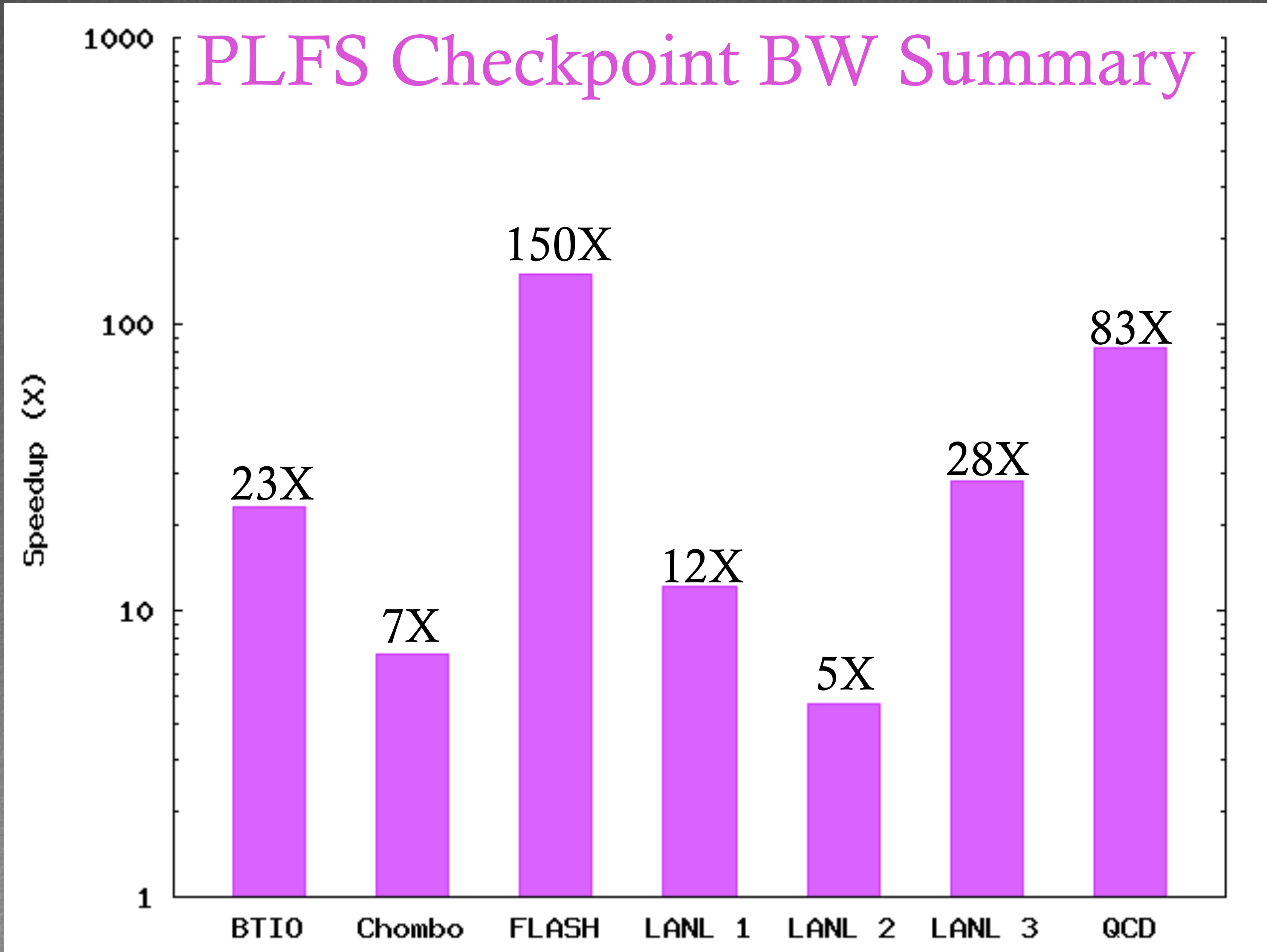


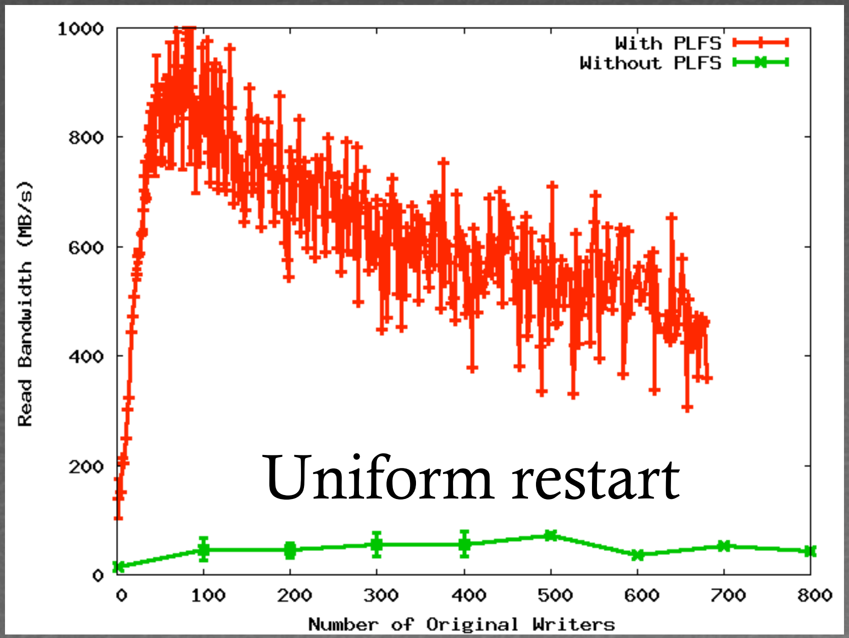
PLFS makes alignment and blocksize irrelevant!

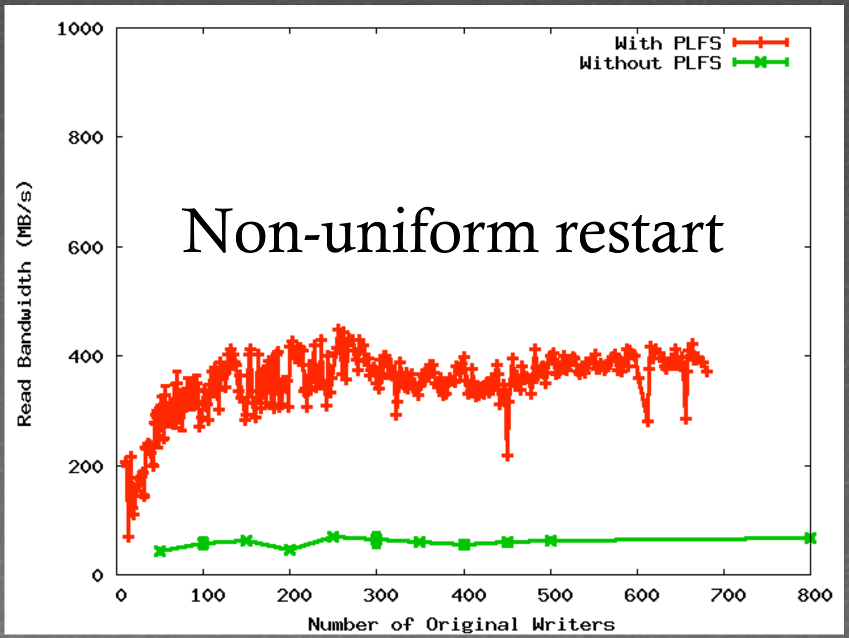
FLASH IO

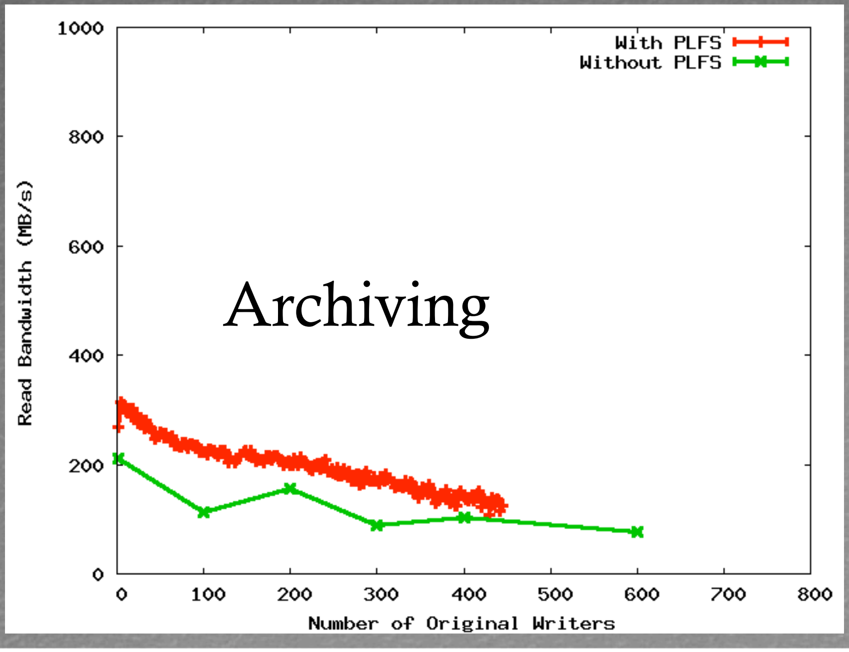


With PLFS
Without PLFS
8000
7000
6000
5000
4000
3000
2000
1000
0
Bandwidth (MB/s)
8
16
32
64
128
256
512
1024
2048
Number of Processors
150X


PLFS Checkpoint BW Summary
Speedup (X)
1000
100
10
1
23X
7X
150X
12X
5X
28X
83X
BTIO
Chombo
FLASH
LANL 1
LANL 2
LANL 3
QCD

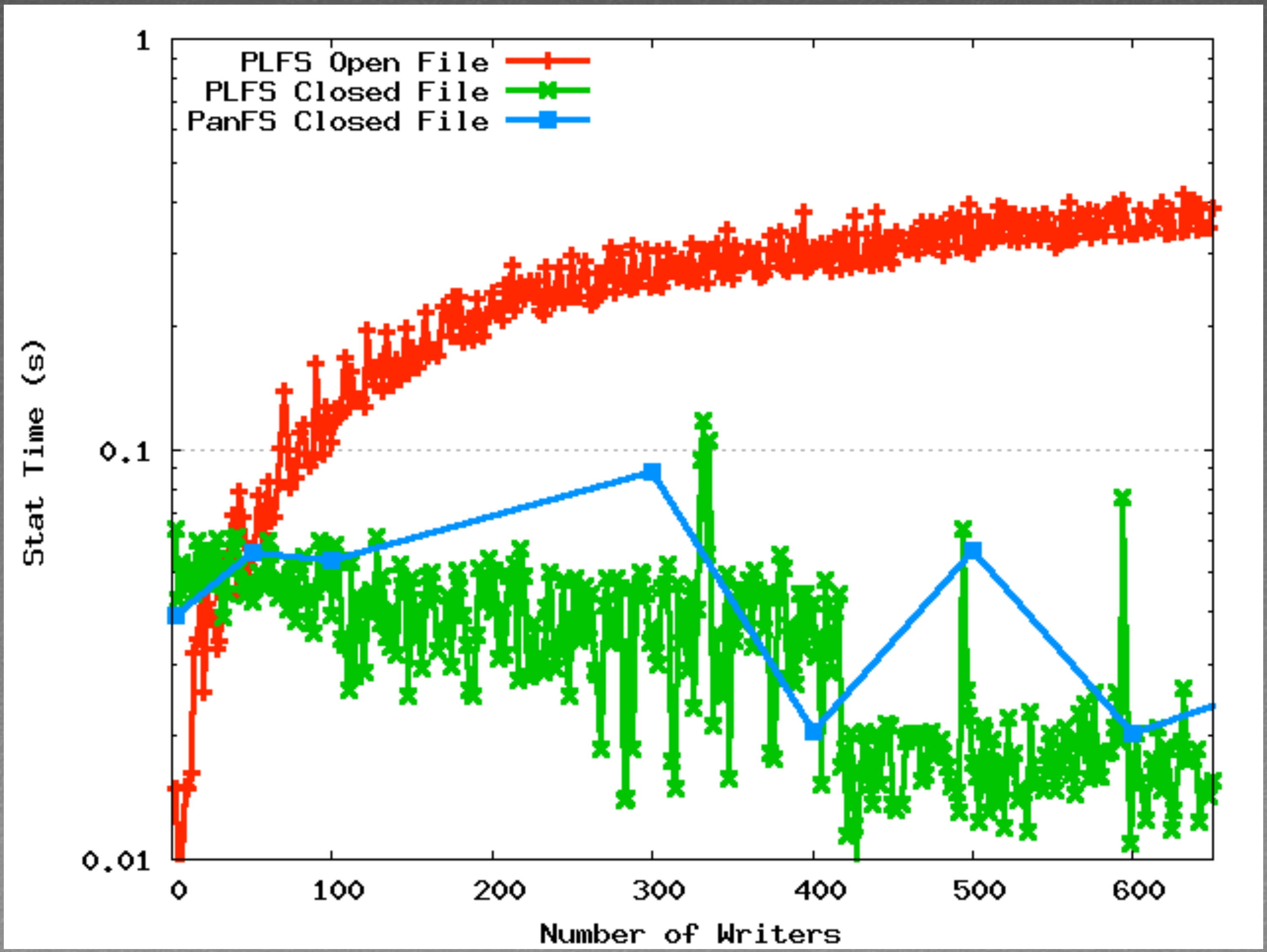




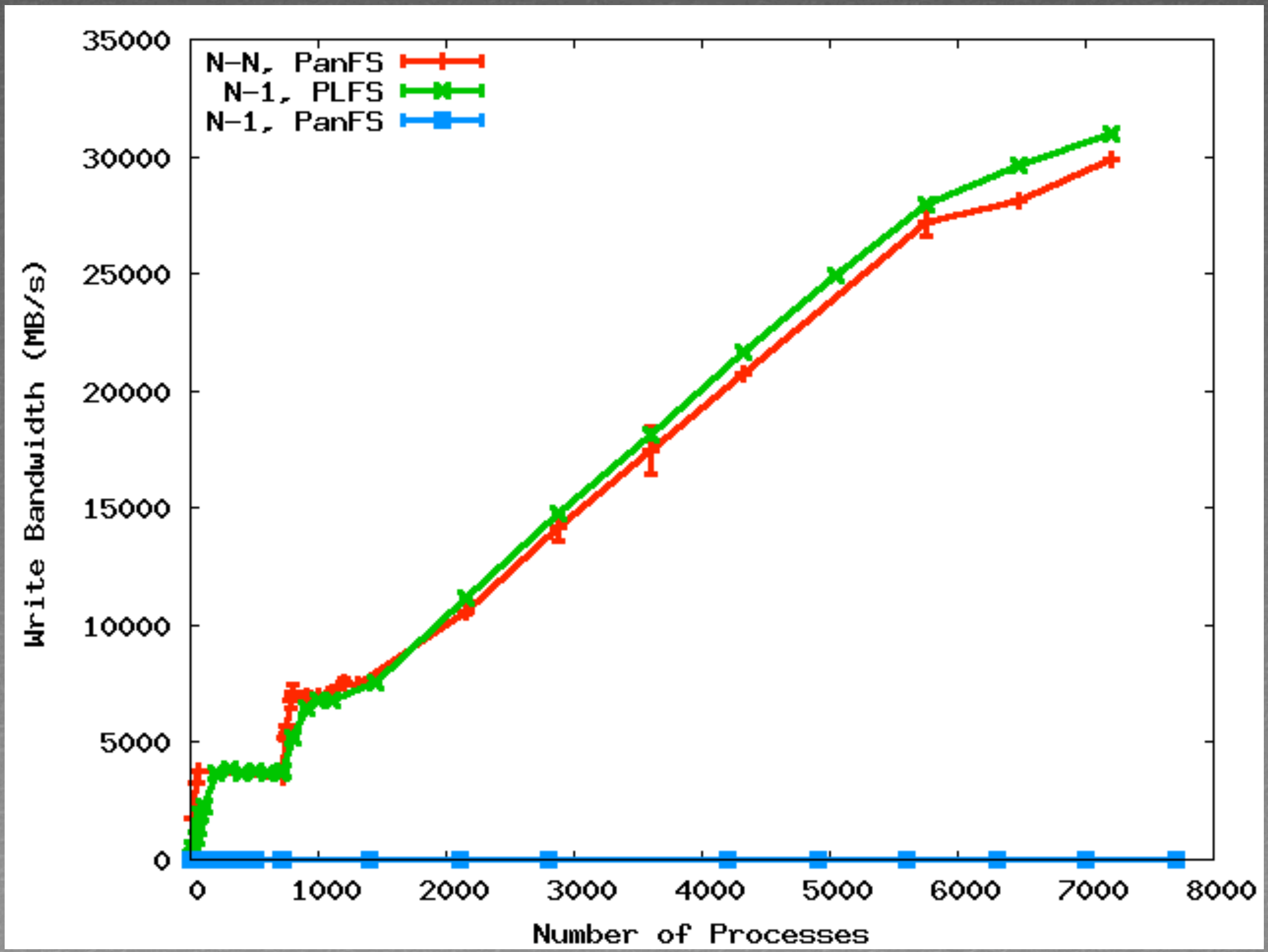


Read Bandwidths

Metadata rates



PLFS/FUSE Overhead



Trade-offs

- Small file bandwidth due to open overhead
- Single node bandwidth due to FUSE/PLFS overhead
  - Small job performance due to single node bandwidth
- Reads in read-write mode
- Possible reduction in read BW for strange read patterns
- Overlapping writes are not ordered
- Shift complexity to N-N challenge

Current and Future Work

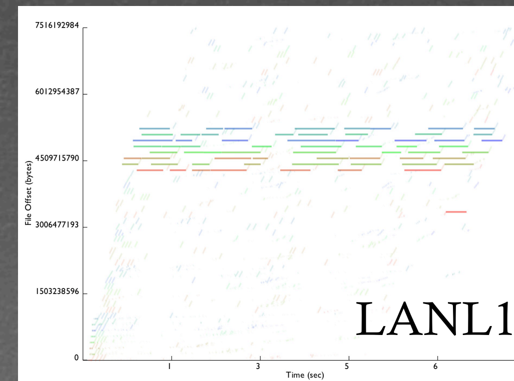
- Directory striping to ameliorate N-N parallel open
- Overhead graph shows
  - Problem for small jobs
  - Lots of idle CPU for large jobs . . .
    - Add compression to index record
    - Add checksums to index record
    - Add extensible metadata to index record

Data ID	**Phys Off**	**Len**	**TS Begin**	**TS End**	**???**

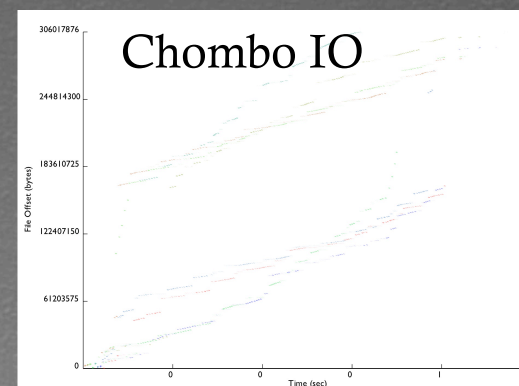
	Interpostion Technique Used	No Extra Resources Used During	No Extra Resources Used After	Maintains Logical Format	Works with Unmodified Applications	Data Immediately Available	Parallel Filesystem Agnostic
ADIOS	Library	Yes	Yes	Yes	No	Yes	Yes
stdck	FUSE	No (LD)	No (LD,N)	Yes	Yes	Yes	Yes
Neighbor	FUSE	No (M)	No (M,N)	Yes	Yes	No	Yes
Diskless	Library	No (M)	No (M)	No	No	Yes	Yes
ZEST	FUSE	No (RD)	No (RD)	No	No	No	No
Lustre Split Writ	Library	Yes	Yes	No/Yes	Yes	Yes	No
PLFS	FUSE	Yes	Yes	Yes	Yes	Yes	Yes

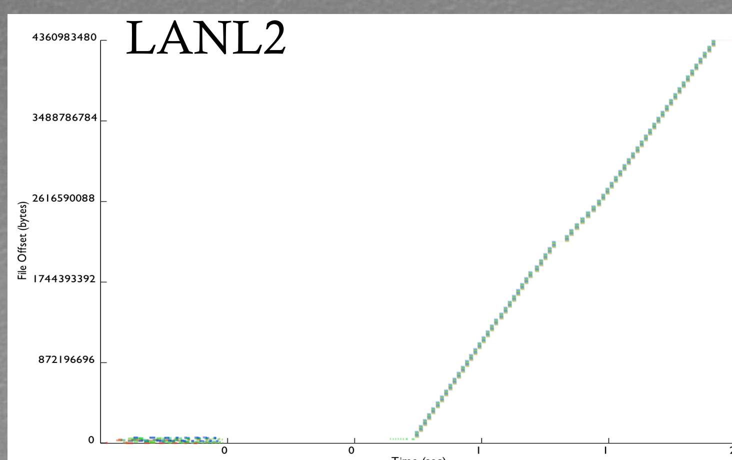
KEY: LD = local disk, M = memory, N = network, RD = remote disk

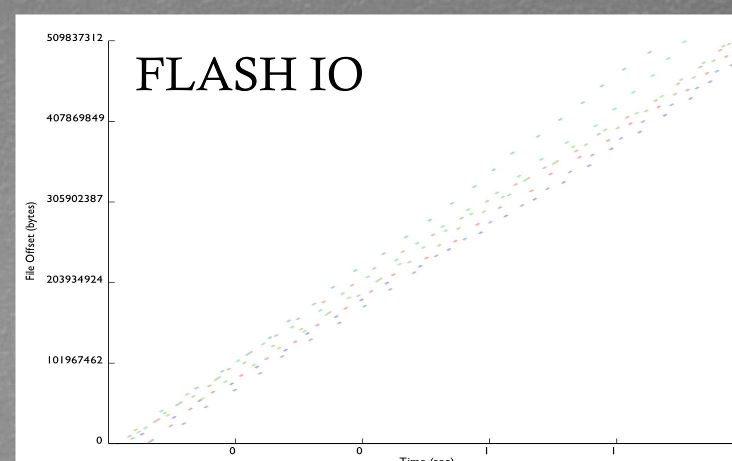
PLFS Conclusion



- 3000 lines of (soon to be open-source) C++
  - Installed on Roadrunner for Open Science
  - Moving onto other production machines next DST
- Improves reads, does not slow down lookups
- Enables easy tracing
  - Traces from all studied benchmarks now published
- Every real app tested significantly improved up to 300X
- Full paper available at http://institutes.lanl.gov/plfs







Outstanding HPC Problems

- Parallel open
- Resiliency
- Schedulers
- Scalable IO and MPI initialization
- Silent data corruption
- Programming models

johnbent@lanl.gov