PLFS and Lustre Performance Comparison

Lustre User Group 2014
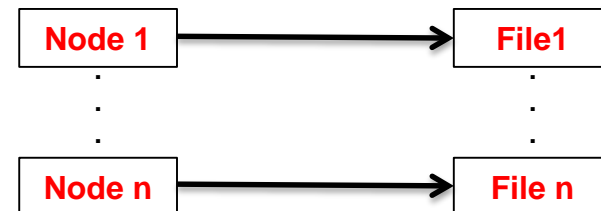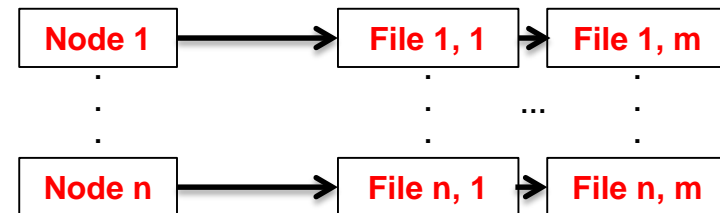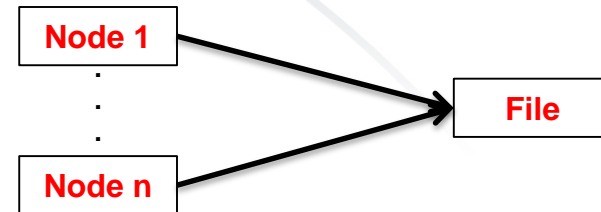
Miami, FL

April 8-10, 2014

Many currently contributing to PLFS

- LANL: David Bonnie, Aaron Caldwell, Gary Grider, Brett Kettering, David Shrader, Alfred Torrez
  - Past Contributors: Ben McClelland, Meghan McClelland, James Nuñez, Aaron Torres
- EMC: John Bent & The EMC Engineering Team
- Carnegie Mellon University: Chuck Cranor
- Other Academics: Jun He (UW-Madison), Kshitij Mehta (U. Houston)
- Cray: William Tucker and the MPI Team

Two of the three primary I/O models really benefit from PLFS

- Shared File, N-1
  - Strided: Data is organized by type
  - Segmented: Data is organized by pe
- Small File, 1-N
  - Each pe to its own set of many small files
- Flat File, N-N
  - Each pe to its own file
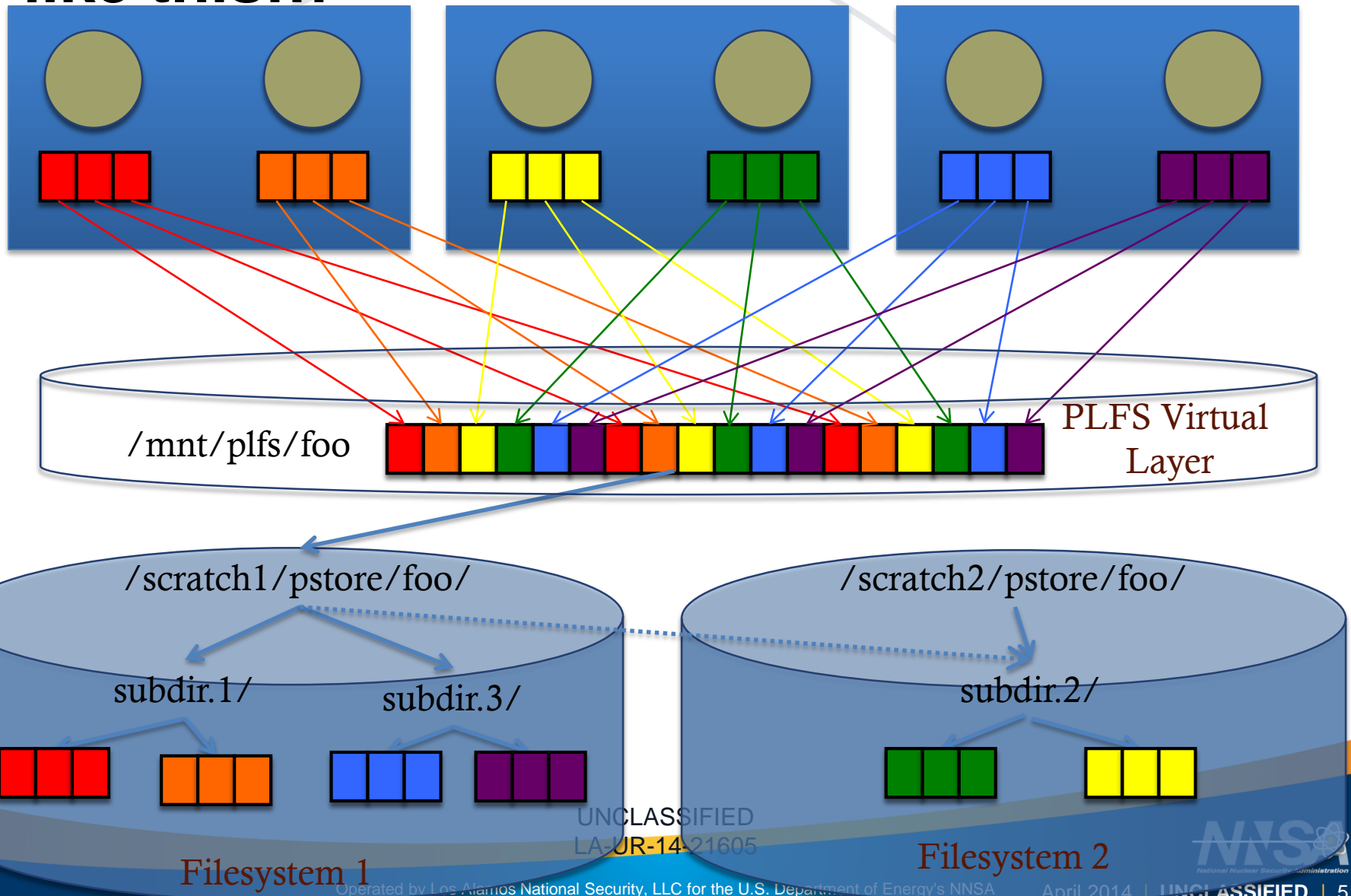 - Distributes files across directories

Node 1
.
.
.
Node n

File

Node 1
.
.
.
Node n

File 1, 1 → File 1, m
...
File n, 1 → File n, m

Node 1
.
.
.
Node n

File1
.
.
.
File n

Shared File (N-1) addresses normal N-1 performance issues

- **Issue: Excessive seeking**
  - Use log-structured file approach
    - Write data sequentially as it arrives
      - One data file/writer
      - Index files to map physical location to logical location
        - Association with writers/nodes varies – see overhead information
  - Convert N-1 to N-N in PLFS containers
    - One or more backend storage locations (can be on different file systems)

- **Issue: Region locking**
  - Convert N-1 to N-N in PLFS containers
    - 2 or more writers cannot access the same file in the same data region (e.g. OST on Lustre)

- **Consult the index to resolve reads**

Shared File (N-1 strided) looks like this…

/mnt/plfs/foo

PLFS Virtual Layer

/scratch1/pstore/foo/

/scratch2/pstore/foo/

subdir.1/

subdir.3/

subdir.2/

Filesystem 1

Filesystem 2

Small File (1-N) combines many, small files into a file per process

- Write multiple files to a single file per writer
  - Create a "PLFS container" on one or more backend file systems
  - Append each write to a log file per writer
  - Create an "index" for each writer to track what went where

- Cache locally for performance

  - Must explicitly flush to storage to provide current data to other nodes
    - Not POSIX compliant in this

- Consult the index to resolve reads

Small File (1-N) looks like this...

PLFS Virtual Layer
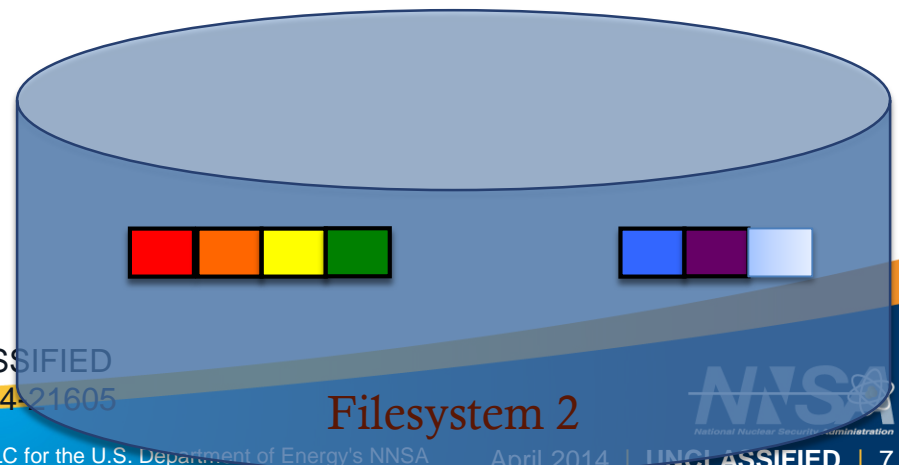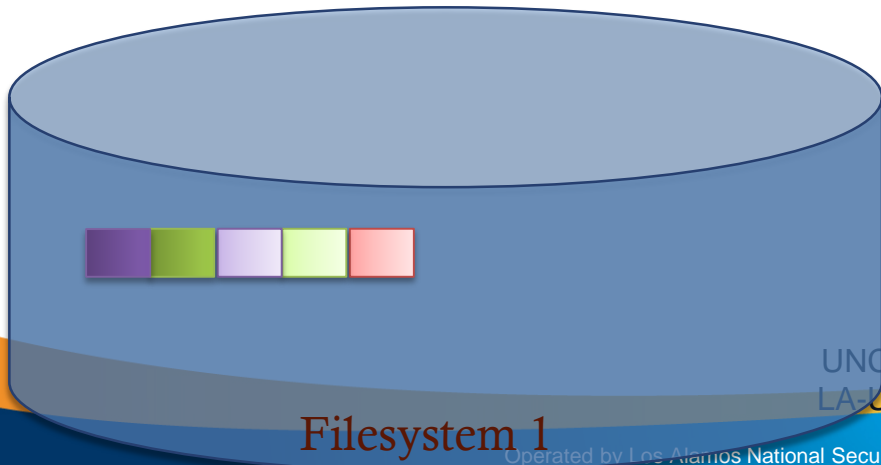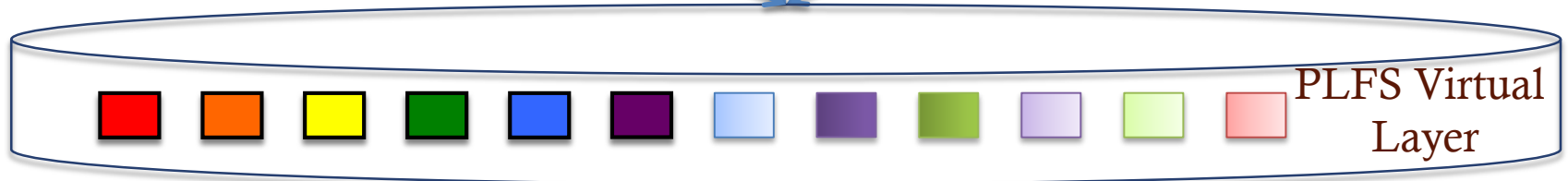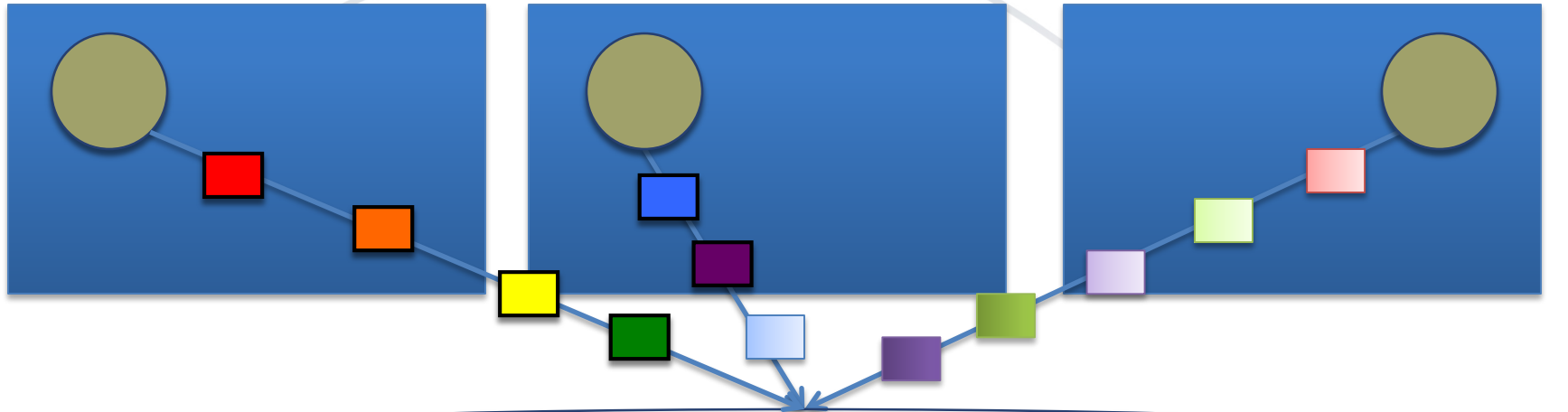
Filesystem 1

Filesystem 2

Flat File (N-N) distributes the files over multiple directories

- Creates/accesses files in an apparent shared directory
  - Distributes files over multiple directories that are potentially on multiple file systems
  - Constructs a user view that shows the files in the directory to which the application wrote
- Allows underlying file system to distribute the metadata load
  - Lustre can with clustered metadata servers
  - Panasas can with different volumes

Flat File (N-N) looks like this...

/mnt/plfs/foo/

PLFS Virtual View

file00 file01 file02 file03 file04 file05 file06 file07
file08 file09 file10 file11 file12 file13 file14 file15

/scratch1/pstore/foo/

/scratch2/pstore/foo/

file00 file02 file04 file06
file08 file10 file12 file14

file01 file03 file05 file07
file09 file11 file13 file15

Filesystem 1

Filesystem 2
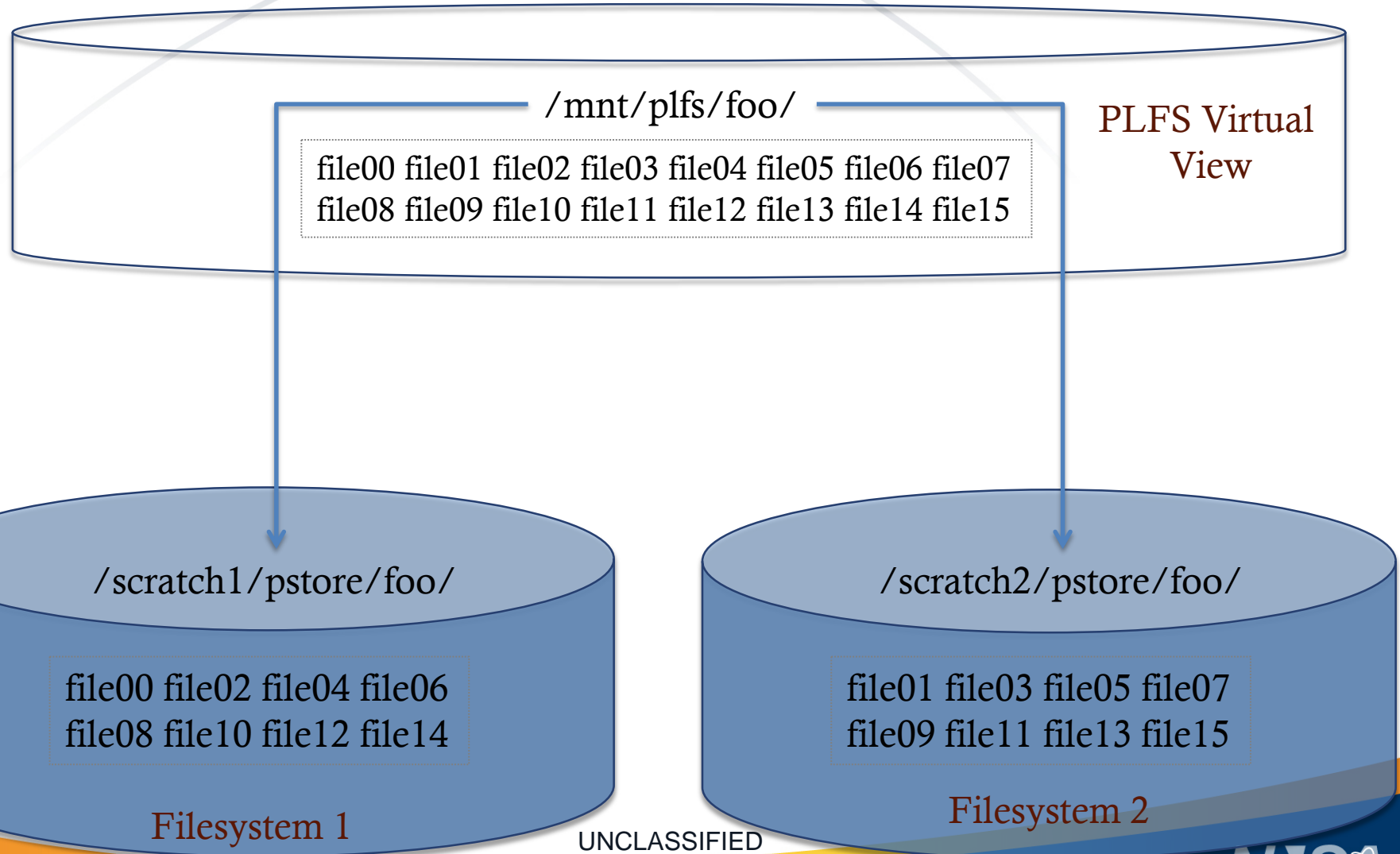
UNCLASSIFIED
LA-UR-14-21605

Use one of 3 interfaces depending on your needs

- MPI/IO is the highest performance interface
  - Load a MPI module that has been patched for PLFS
  - Prepend "plfs:" to the file path used in MPI_File_open
  - Mount point must be defined in PLFSRC

- FUSE is primarily intended for interactive command access (e.g. ls, rm, mv, non-PLFS-aware archive tools, etc.)
  - Can use it for POSIX I/O, but performance suffers due to FUSE overhead
  - Mount point must be defined in PLFSRC and mounted using the PLFS daemon
  - Ensure FUSE buffers are larger than 128 KiB default

- PLFS API is high performance, but requires major programming changes
  - Must change source code to use plfs_* calls to handle the files

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 10

N-1 PLFS file overhead depends on interface – MPI/IO

- Recommend 1 backend/parallel file system for N-1
- 1 top-level directory (container)/backend to hold the rest of the files
- 1 hostdir/node/container to hold data and index files
  - PLFSRC num_hostdirs parameter sets limit on maximum hostdirs/top-level directory
- 1 meta directory/container to hold metadata caching files
- 1 ".plfsaccess" file to enable distinguishing PLFS container from logical directory
- 1 metadata file/hostdir to cache "stat" information
- 1 version file to help with backward compatibility checks
- 1 data file/writer process
- 1 index file/writer process

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 11

N-1 PLFS file overhead depends on interface – PLFS FUSE

- Recommend 1 backend/parallel file system for N-1

- 1 top-level directory (container)/backend to hold the rest of the files

- 1 hostdir/node/container to hold data and index files
  - PLFSRC num_hostdirs parameter sets limit on maximum hostdirs/top-level directory

- 1 meta directory/container to hold metadata caching files

- 1 ".plfsaccess" file to enable distinguishing PLFS container from logical directory

- 1 version file to help with backward compatibility checks

- 1 index file/node/container

- 1 metadata file/hostdir to cache "stat" information

- 1 data file/writer process

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 12

N-1 PLFS file overhead depends on interface – PLFS Lib

- Recommend 1 backend/parallel file system for N-1
- 1 top-level directory (container)/backend to hold the rest of the files
- 1 hostdir/node/container to hold data and index files
  - PLFSRC num_hostdirs parameter sets limit on maximum hostdirs/top-level directory
- 1 meta directory/container to hold metadata caching files
- 1 ".plfsaccess" file to enable distinguishing PLFS container from logical directory
- 1 version file to help with backward compatibility checks
- 1 data file/writer process
- 1 index file/writer process
- 1 metadata file/writer process

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 13

N-N and 1-N PLFS file overhead is simpler than N-1

- N-N:
  - Recommend ~10 backends/parallel file system
  - 1 data file/writer process distributed to one of the backends
  - Create 1 directory/backend when director is created
    - Not necessarily at file creation time
- 1-N:
  - Recommend 1 backend/parallel file system
  - 1 data file/writer process in which the N files are stored
  - 1 index file/writer process with reference to which of N files in the 1 file it refers
  - 1 map file/process that maps the index number to a string name for the file to which it refers

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 14

Use PLFS for very large parallel file I/O, primarily N-1 or 1-N

- Need to work with lots of data to amortize additional file open/sync/close overhead
  - Restart and graphics dumps
- Small files should go to NFS (/usr/projects, /users, /netscratch)
  - Executables, problem parameters, log files, etc.
- Small File (1-N), in particular, is not completely POSIX-compliant for performance reasons
  - Explicit sync calls required to ensure latest goes to or comes from storage
- Avoid O_RDWR to enhance performance
  - In RDWR mode any read must completely rebuild the index from storage – slow!
- Avoid stat'ing files to monitor progress

fs_test for maximum bandwidth scenario

- Get it at https://github.com/orgs/fs-test
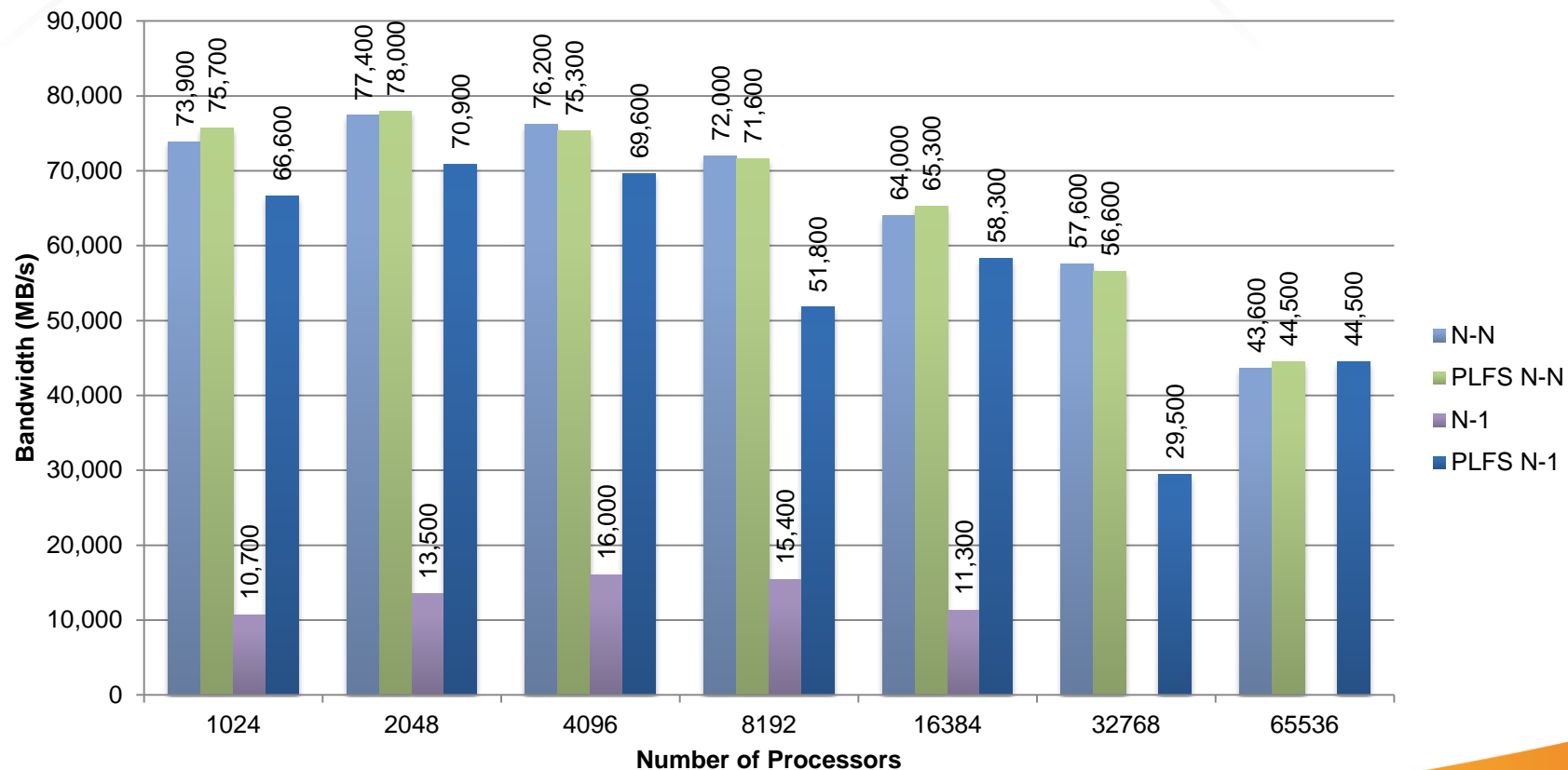
- Used PLFS's MPI/IO ADIO interface

- Unable to complete N-1 runs for 32K & 64K pes

- Used ACES Cray XE6 lscratch3 PFS
  - ½ the total Lustre hardware on the system

- Competed with other jobs running on system
  - See PLFS N-1 results at 32K pes

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 16

Significantly better N-1 performance & N-N comparable to non-PLFS

Write Effective Bandwidth

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 17

Read results similar to write results

Read Effective Bandwidth

Measured small & large Silverton problem performance

- 3-D Eulerian finite difference code to study high-speed compressible flow & high-rate material deformation
  - Multiple MPI/IO modes: N-1 strided, N-1 segmented, and N-N
- Used a small file size problem first in just N-1 strided
- Measured a large file size problem using all modes
- Used ACES Cray XE6 Iscratch3 PFS
- Competed with other jobs running on system

Small problem N-1 strided write showed 1.68x improvement

Silverton Small Problem Restart Write Bandwidth
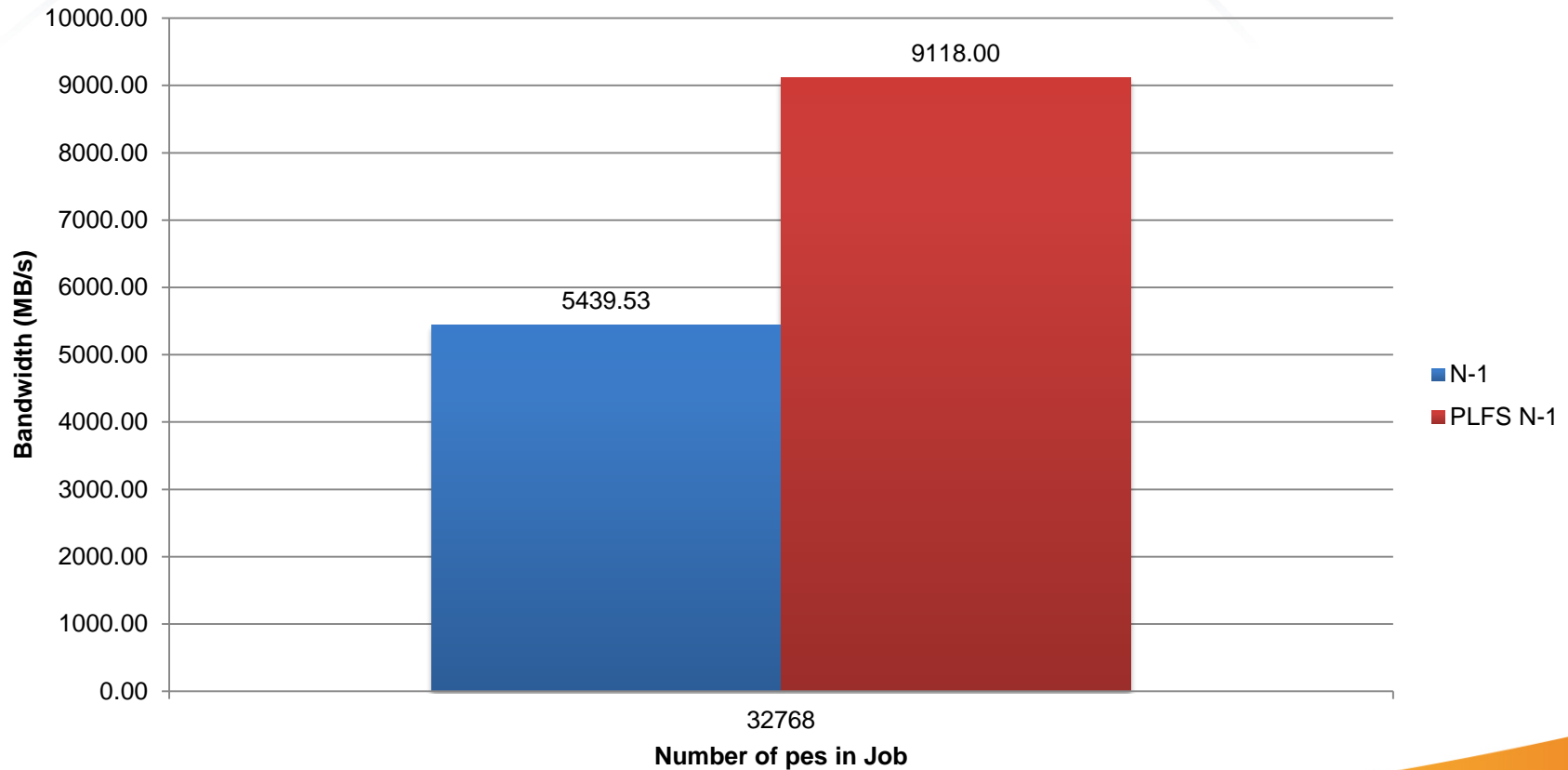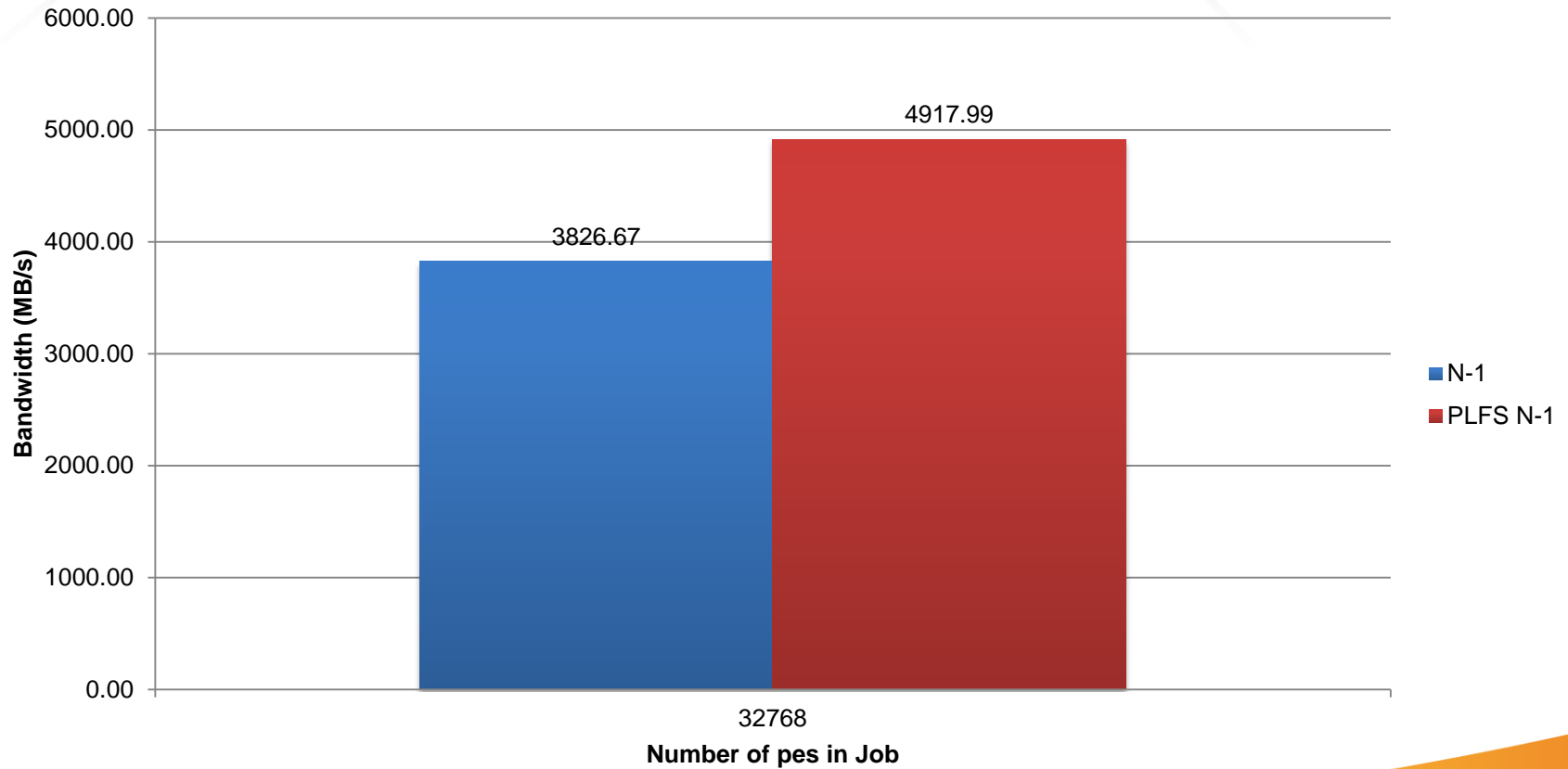
Small problem N-1 strided read showed 1.28x improvement

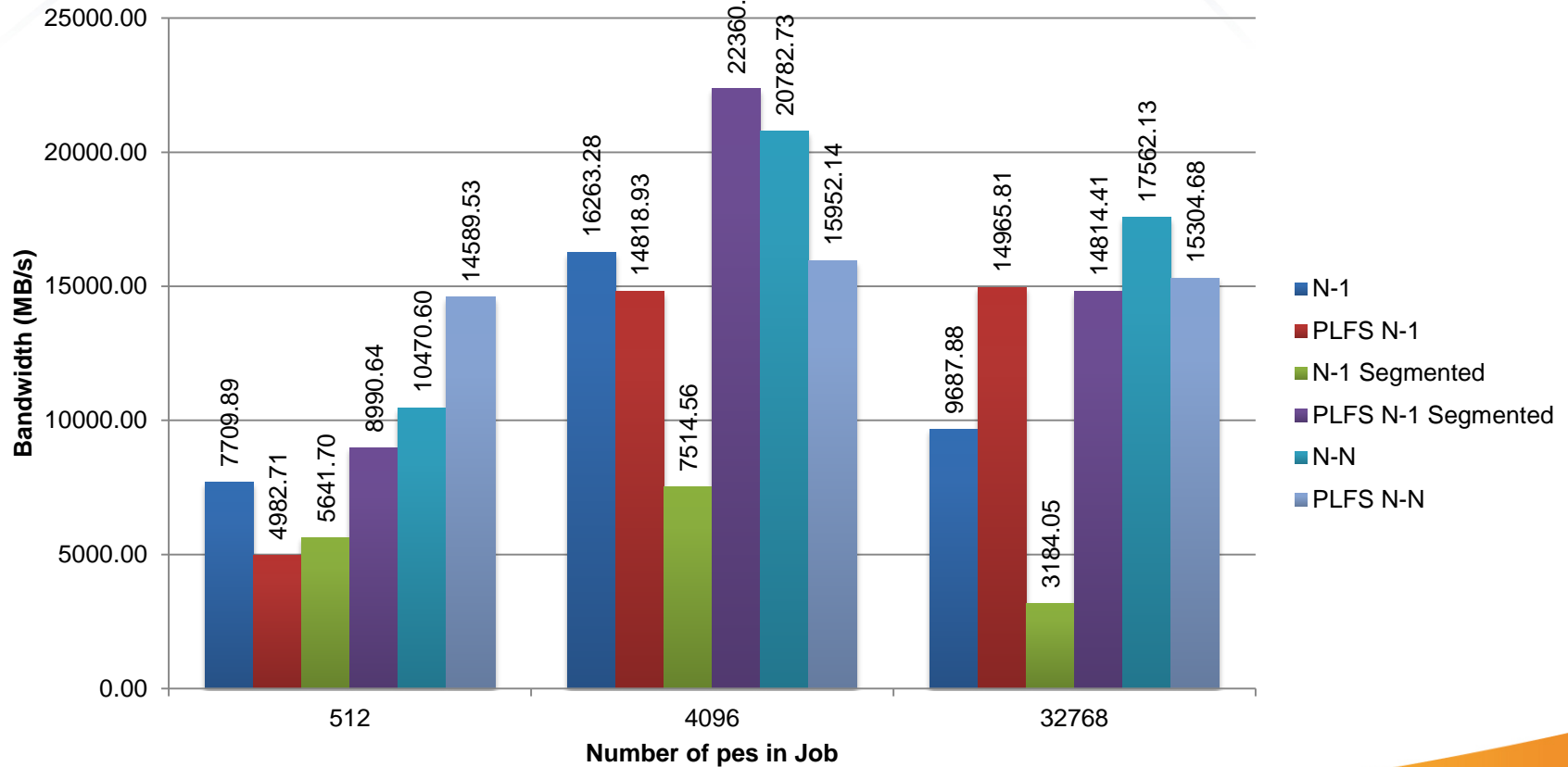**Silverton Small Problem Restart Read Bandwidth**

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 21

As writer count increased so did PLFS advantage

Silverton Large Problem Restart Avg Write Bandwidth

Bandwidth (MB/s)

Number of pes in Job

Legend:
- N-1
- PLFS N-1
- N-1 Segmented
- PLFS N-1 Segmented
- N-N
- PLFS N-N

512: 7709.89, 4982.71, 5641.70, 8990.64, 10470.60, 14589.53

4096: 16263.28, 14818.93, 7514.56, 22360.67, 20782.73, 15952.14

32768: 9687.88, 14965.81, 3184.05, 14814.41, 17562.13, 15304.68
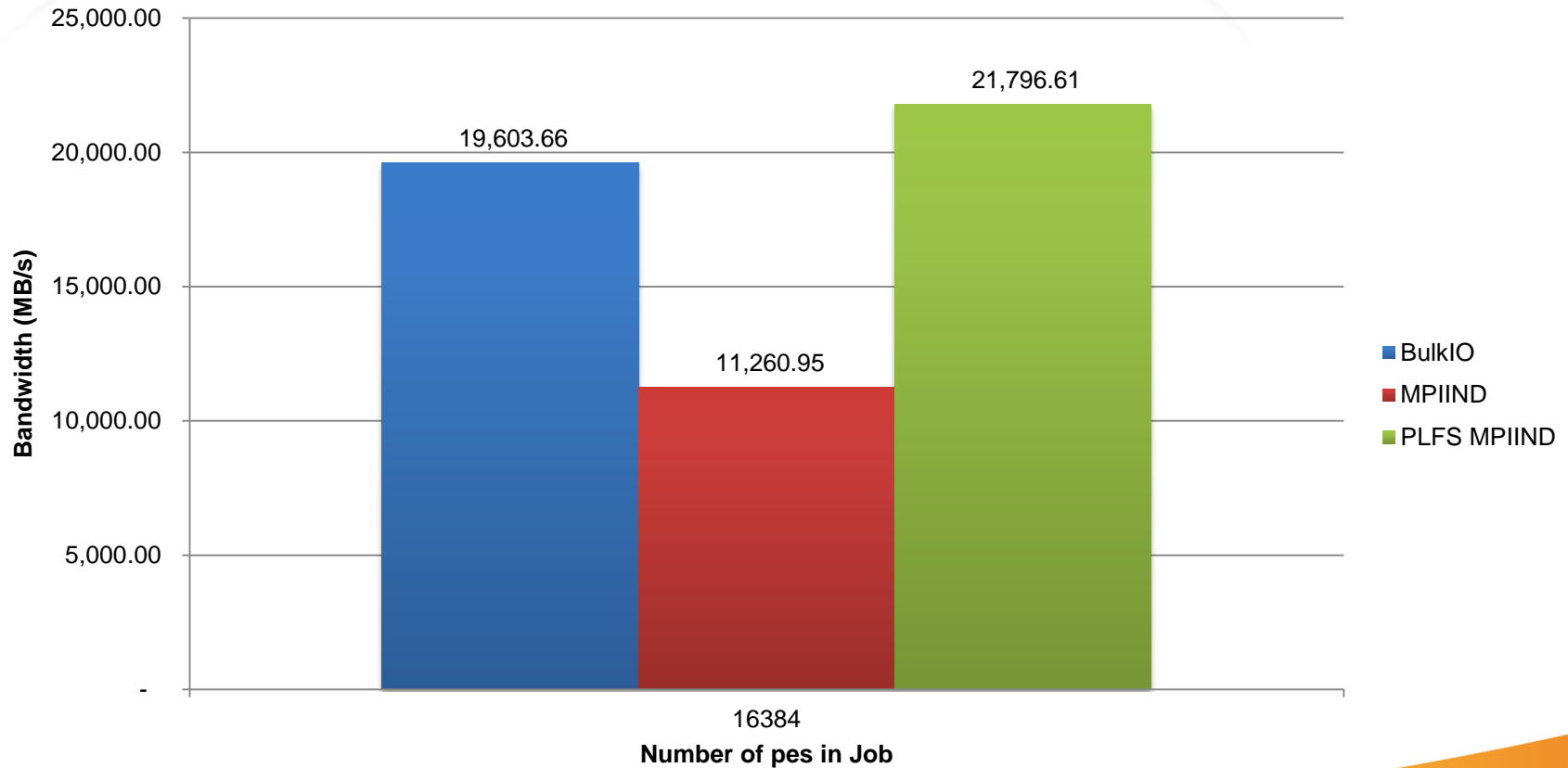
UNCLASSIFIED
LA-UR-14-21605

Measured small & large EAP problem performance

- 3-D Godunov solver using Eulerian mesh with AMR (Adaptive Mesh Refinement)
  - Multiple MPI/IO modes: BulkIO (aggregating N-1 strided) and MPIIND (N-1 strided)
- Used the Asteroid, a small file size, problem first on Iscratch3
- Measured the "MD", a very large file size, problem on Iscratch2, Iscratch3, and, for PLFS, a virtual combination of Iscratch 2, 3, & 4
- Competed with other jobs running on system
  - Averages caught by competition, so will show maximum values as better comparison

UNCLASSIFIED
LA-UR-14-21605

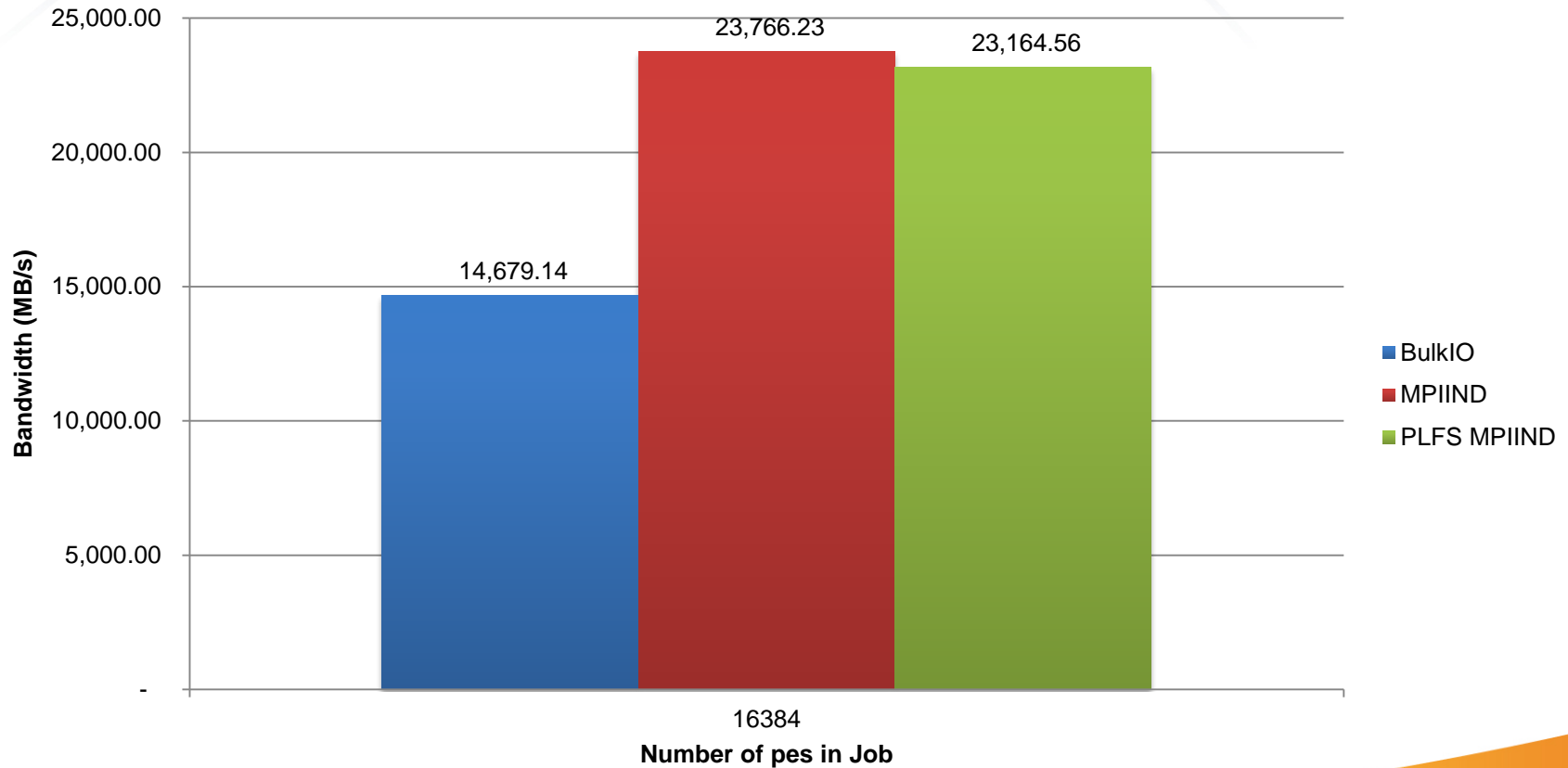# Asteroid write 1.1x over BulkIO & 1.94x over MPIIND

Los Alamos
NATIONAL LABORATORY
— EST.1943 —

EAP Asteroid Maximum Write Bandwidth

Bar chart showing Bandwidth (MB/s) vs Number of pes in Job (16384):
- BulkIO: 19,603.66
- MPIIND: 11,260.95
- PLFS MPIIND: 21,796.61

Y-axis: Bandwidth (MB/s), ranging from - to 25,000.00

Legend: BulkIO, MPIIND, PLFS MPIIND

Asteroid read 1.58x over BulkIO, effectively equal to MPIIND

EAP Asteroid Read Bandwidth

Chart: EAP Asteroid Read Bandwidth, Bandwidth (MB/s) vs Number of pes in Job (16384)

- BulkIO: 14,679.14
- MPIIND: 23,766.23
- PLFS MPIIND: 23,164.56

"MD" write 1.5x over BulkIO, big gain by combining multiple PFSes

EAP MD Write Bandwidth

Bar chart showing Bandwidth (MB/s) vs Number of pes in Job (65536):

- BulkIO to lscratch2: 13,045.45
- PLFS MPIIND to lscratch2: 19,516.13
- BulkIO to lscratch3: 18,100.00
- PLFS MPIIND to lscratch234: 56,279.07

Y-axis: Bandwidth (MB/s), 0 to 60,000.00
X-axis: Number of pes in Job
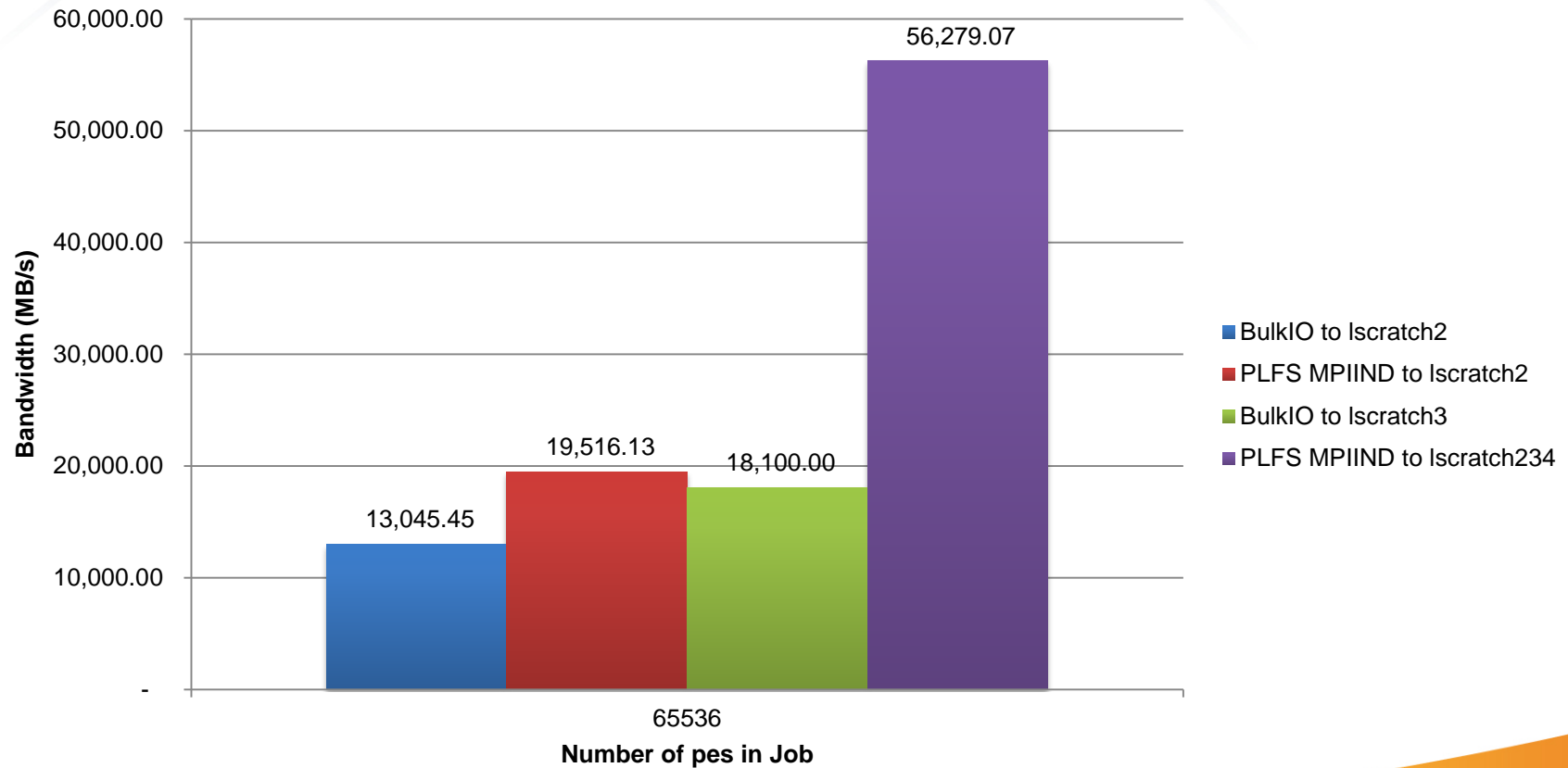
"MD" read 2.1x over BulkIO, not too big gain by combining multiple PFSes

EAP MD Read Bandwidth

UNCLASSIFIED
LA-UR-14-21605

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA   April 2014 | **UNCLASSIFIED** | 27
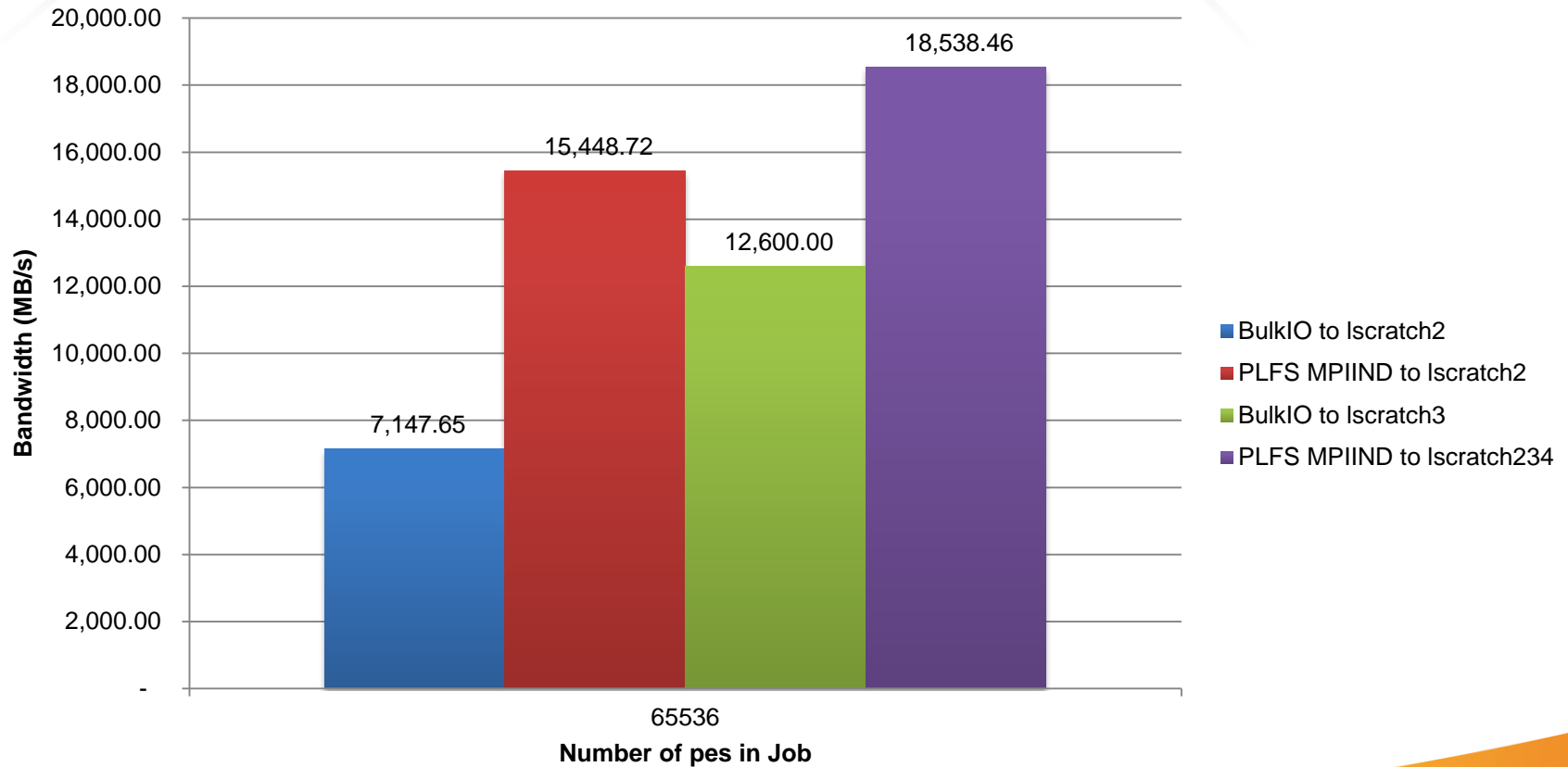
More work is needed for 7/24/365 production use

- Patch FUSE buffers from 128 KiB to at least 4 MiB
  - Allows tools reading files to amortize read overhead over more data
  - Early testing shows HSI archive tool sees 1.36x – 2.39x performance improvement, depending on file size
- Manage PLFS indexes in scalable manner
  - Every pe reads full index now, which can fill memory
  - Experimenting with MDHIM, a distributed key/value middleware
    - See https://github.com/mdhim