Lustre Performance Investigations on Theta

Francois Tessier, George Brown, Preeti Malakar, Rick Zamora, Venkat Vishwanath, Paul Coffman
(ftessier, gbrown, pmalakar, rzamora, venkat, pcoffman)@anl.gov

ALCF

Argonne NATIONAL LABORATORY

Overview

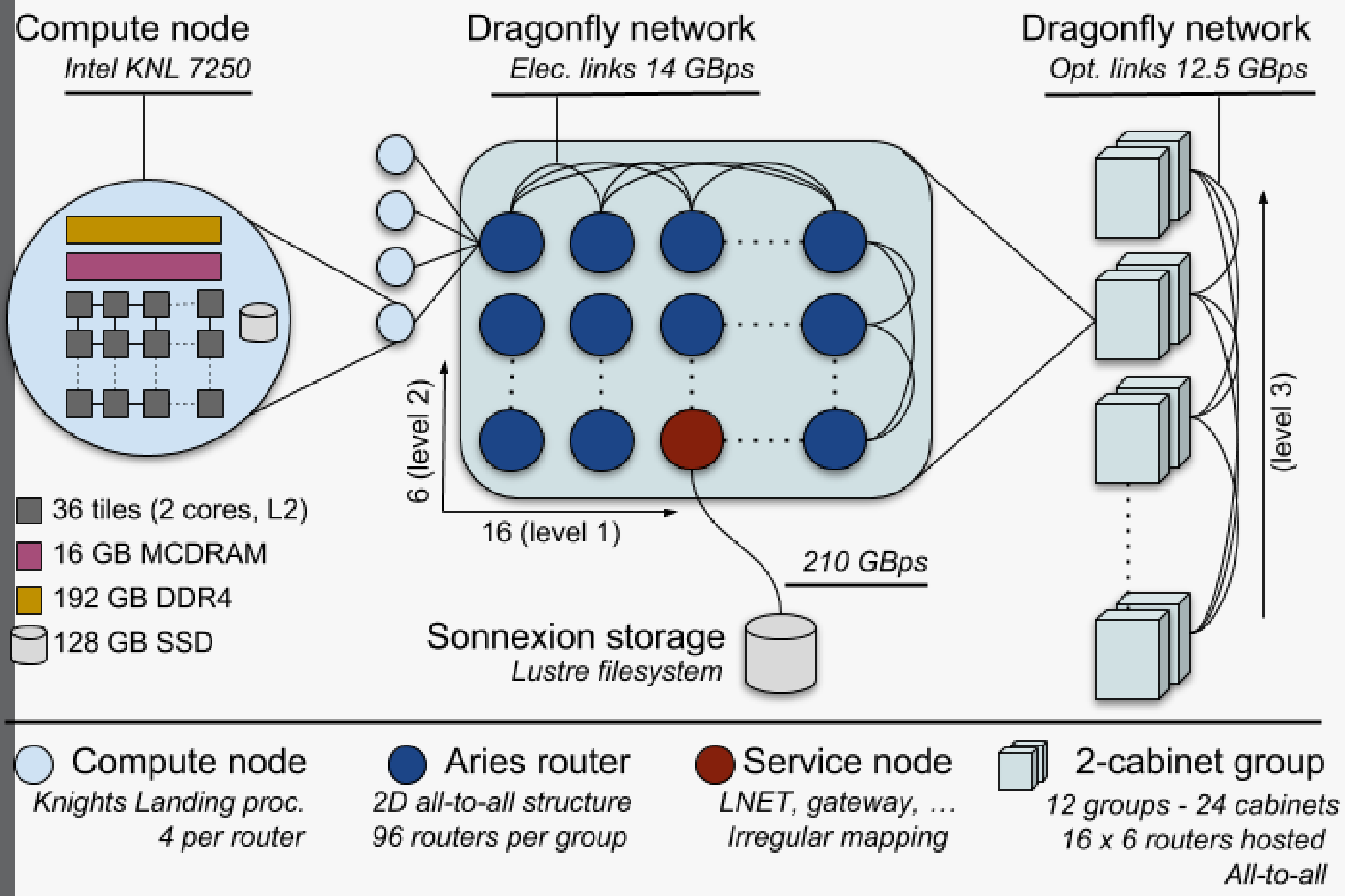
- Theta Lustre Overview
- Performance Characterizations using Cray MPI-IO within IOR
- HDF5 ECP Work - Custom Collective IO VFD
- Operations Metrics

Theta Overview

Argonne Leadership Computing Facility

2004	Blue Gene/L at LLNL: 90-600 TF system #1 on Top 500 for 3.5 years
2005	Argonne accepts 1 rack (1024 nodes) of Blue Gene/L (5.6 TF)
2006	Argonne Leadership Computing Facility (ALCF) created
2008	ALCF accepts 40 racks (160k cores) of Blue Gene/P (557 TF)
2009	ALCF approved for 10 petaflop system to be delivered in 2012
2012	48 racks of Mira Blue Gene/Q (10 PF) in production at ALCF
2014	Development partnership for Theta and Aurora begins
2016	ALCF accepts Theta (10 PF) Cray XC40 with Xeon Phi (KNL)
2021	Aurora (>1 EF) will be delivered

Theta System Overview



Architecture: Cray XC40
Processor: 1.3 GHz Intel Xeon Phi 7230 SKU
Cores/node: 64
Racks: 24
Nodes: 4,392
Memory/node: 192 GB DDR4 SDRAM
High bandwidth memory/node: 16 GB MCDRAM
SSD/node: 128 GB
Aries interconnect with Dragonfly configuration
Total cores: 281,088
Total MCDRAM: 70 TB
Total DDR4: 843 TB
Total SSD: 562 TB
10 PB Lustre file system
Peak performance of 11.69 petaflops

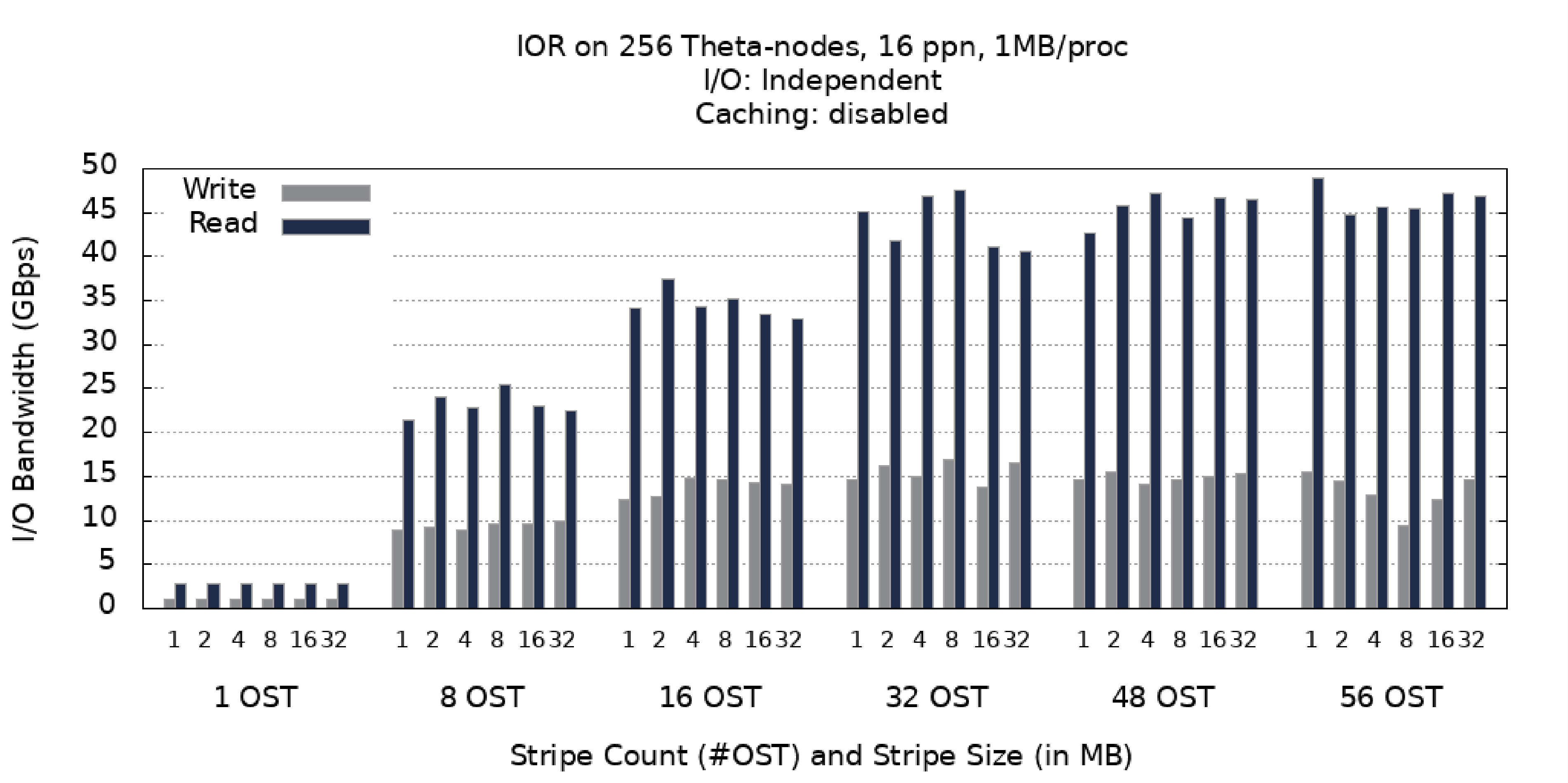
LUSTRE Specifications on Theta

- lfs 2.7.2.26
- Sonexion Storage
  - 4 cabinets
  - 10 PB usable RAID storage
  - Total Lustre Performance Write BW 172 GB/s Read BW 240 GB/s
    - 56 OSS (1 OST per OSS)
      - Peak Performance of 1 OST is 6 GB/s
    - Lustre client cache effects only for much higher BW
      - OSS cache disabled by Sonexion - Cray has seen issues with RAID array bitmap being pushed out of memory due to OSS cache consuming memory on the OSS nodes

Performance Characterizations using MPI-IO within IOR

- Lustre as a component of MPI-IO performance
  - Collective vs Independent, cache effects, shared files vs fpp
- HPC-IOR version
  - Enhanced for MPIIO –e fsync support (MPI_File_sync)
  - https://xgitlab.cels.anl.gov/ExaHDF5/HPC-IOR
- All results show MAX Bandwidth (best times) for each experiment

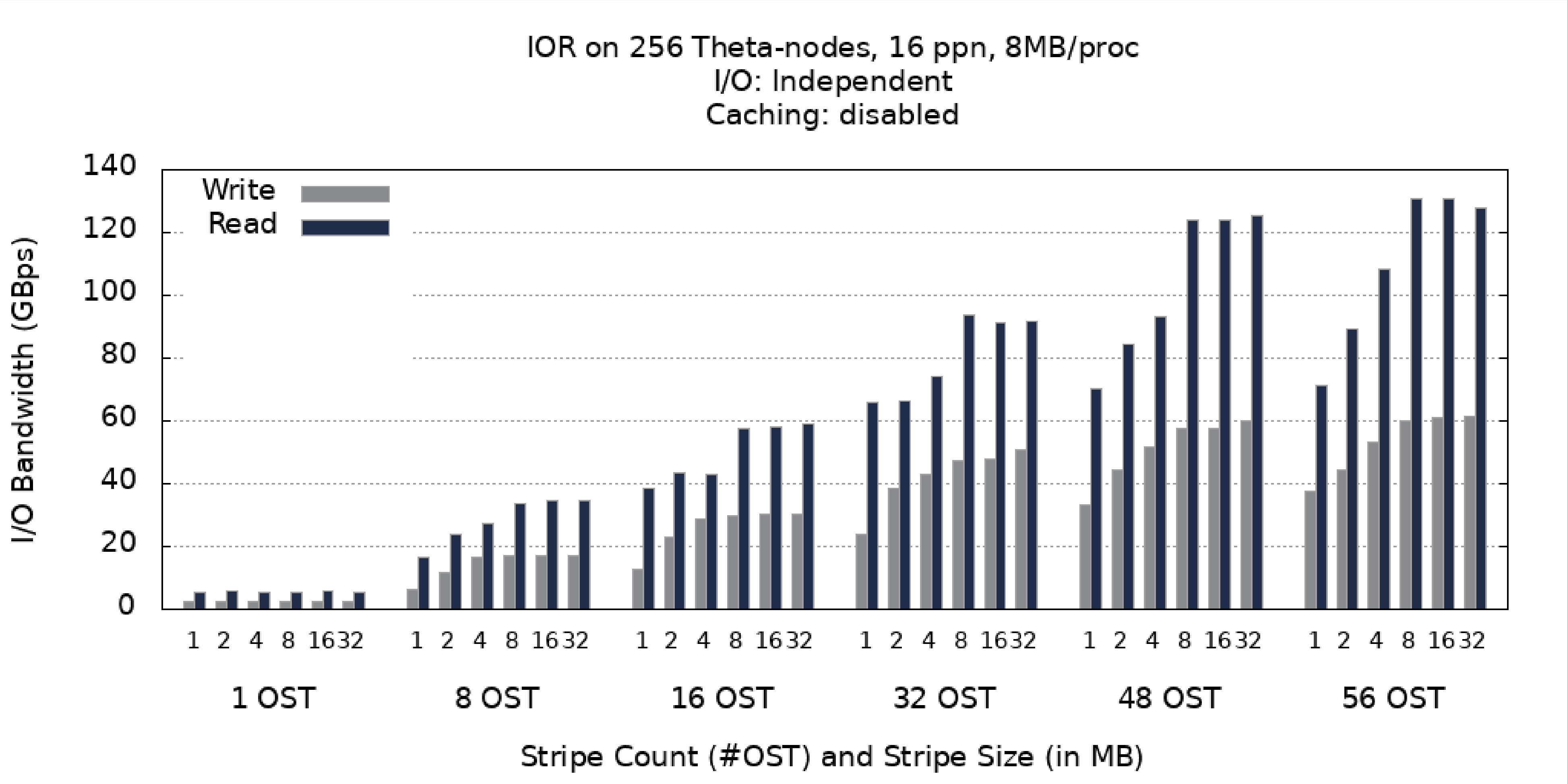
Shared File Stripe Size vs Count Affect on Performance (Independent I/O - No Lustre Client Cache Effects)



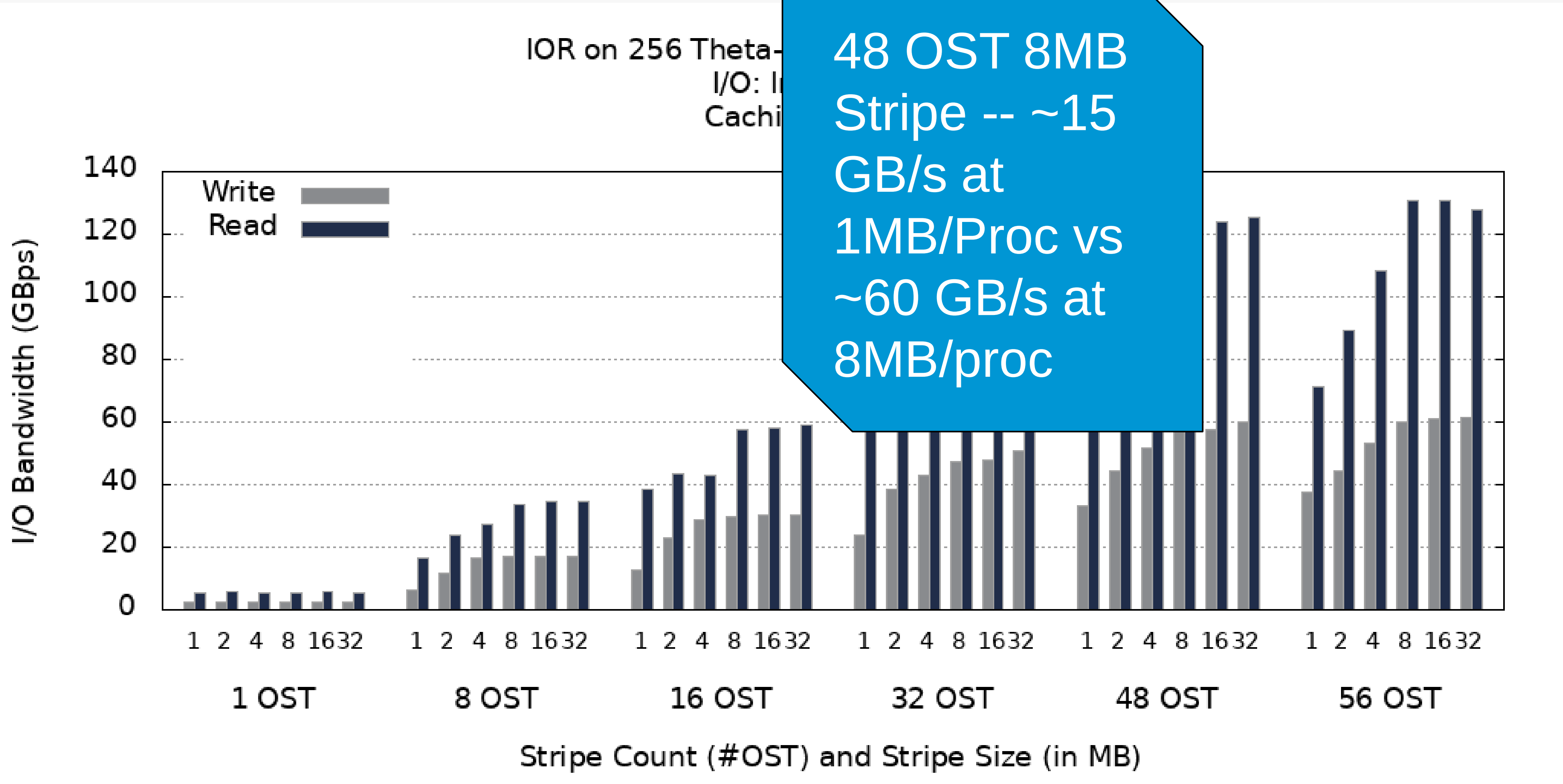
Argonne NATIONAL LABORATORY

Shared File Stripe Size vs Count Affect on Performance (Independent I/O - No Lustre Client Cache Effects)

8MB/proc

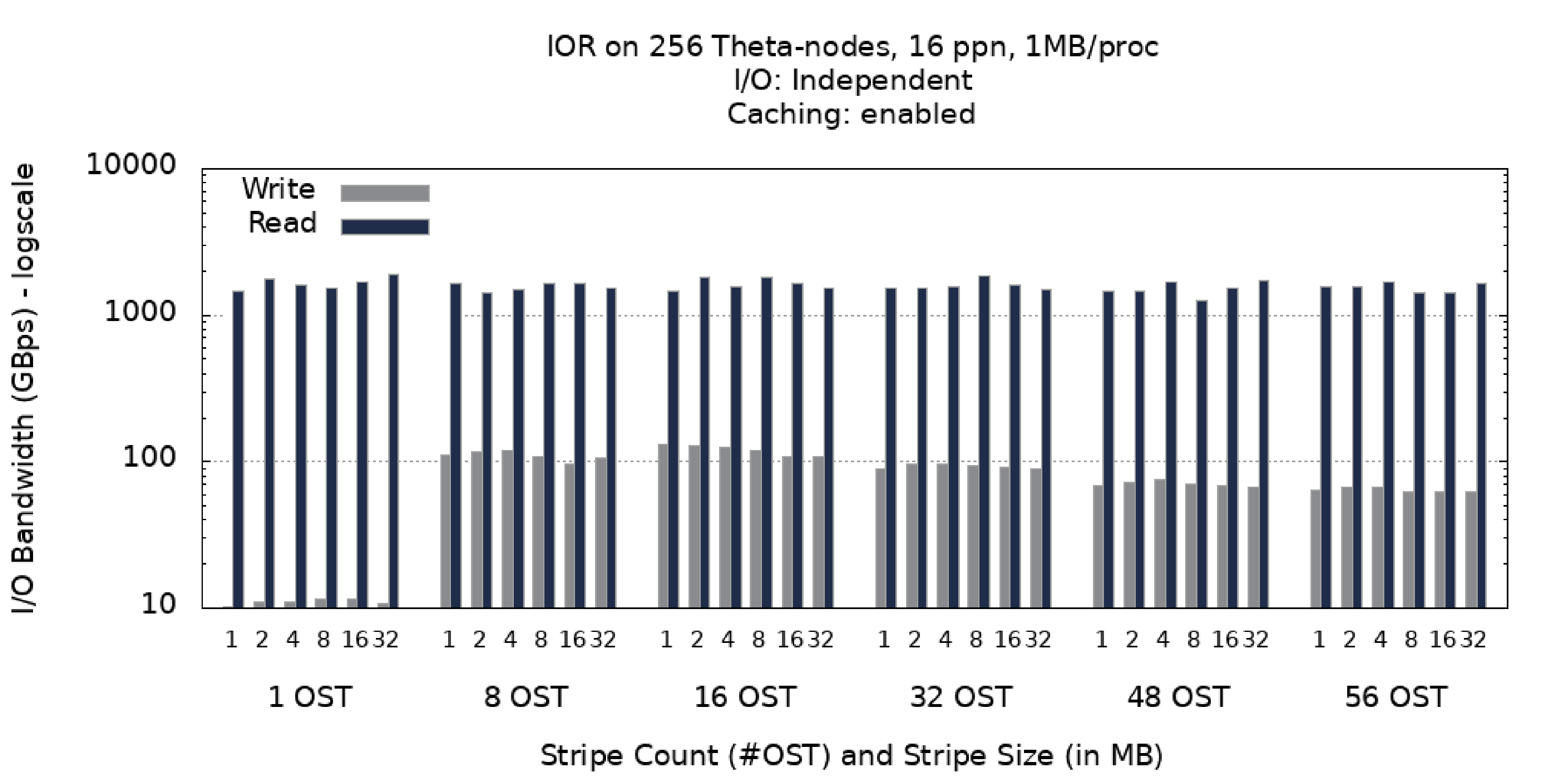


Shared File Stripe Size vs Count Affect on Performance (Independent I/O - No Lustre Client Cache Effects) 8MB/proc



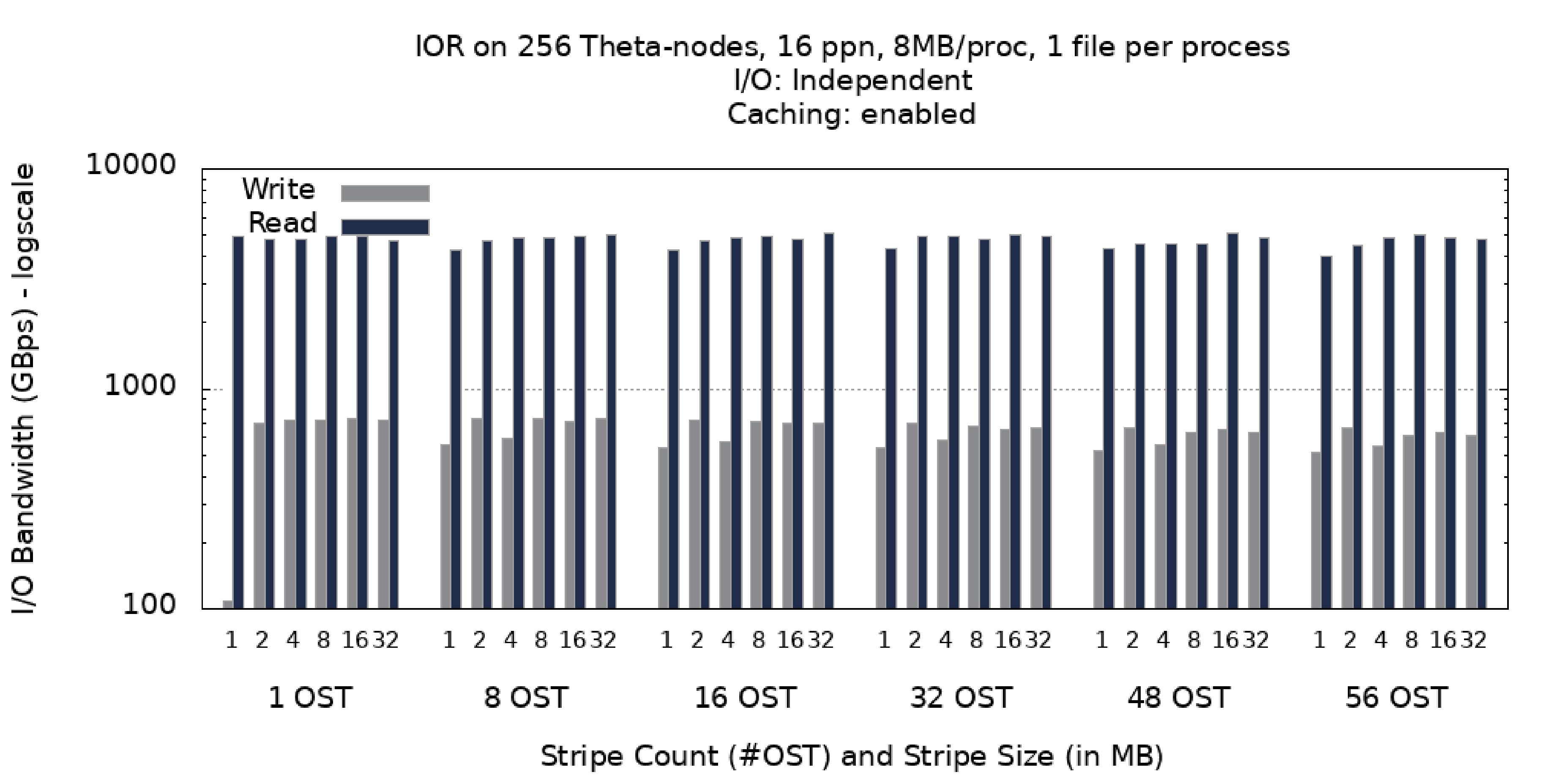
48 OST 8MB Stripe -- ~15 GB/s at 1MB/Proc vs ~60 GB/s at 8MB/proc

Shared File Stripe Size vs Count Affect on Performance (Independent I/O – with Lustre Client Cache Effects)

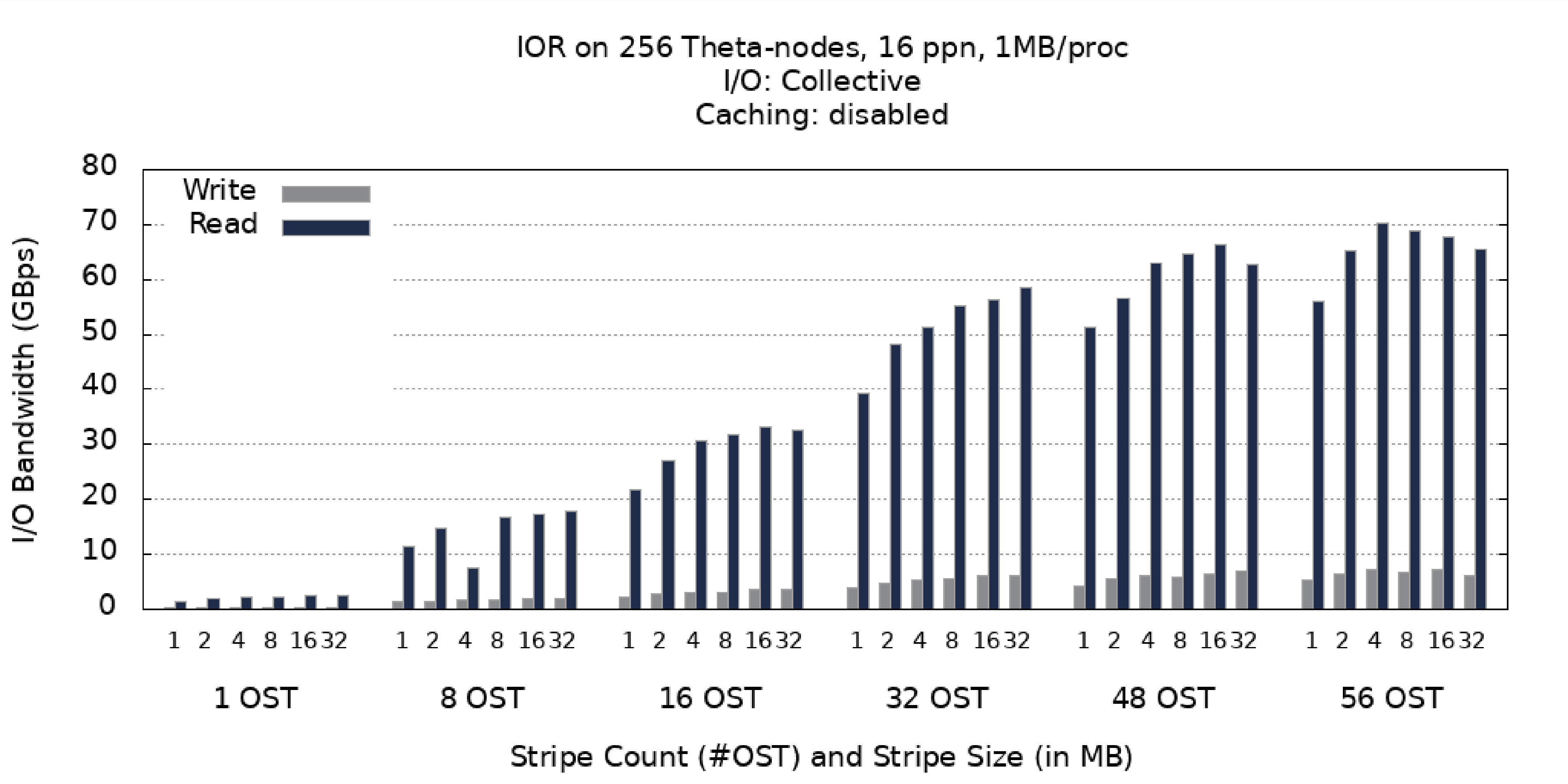


File Per Process - Stripe Size vs Count Affect on Performance (Independent I/O – with Lustre Caching)

8MB/proc

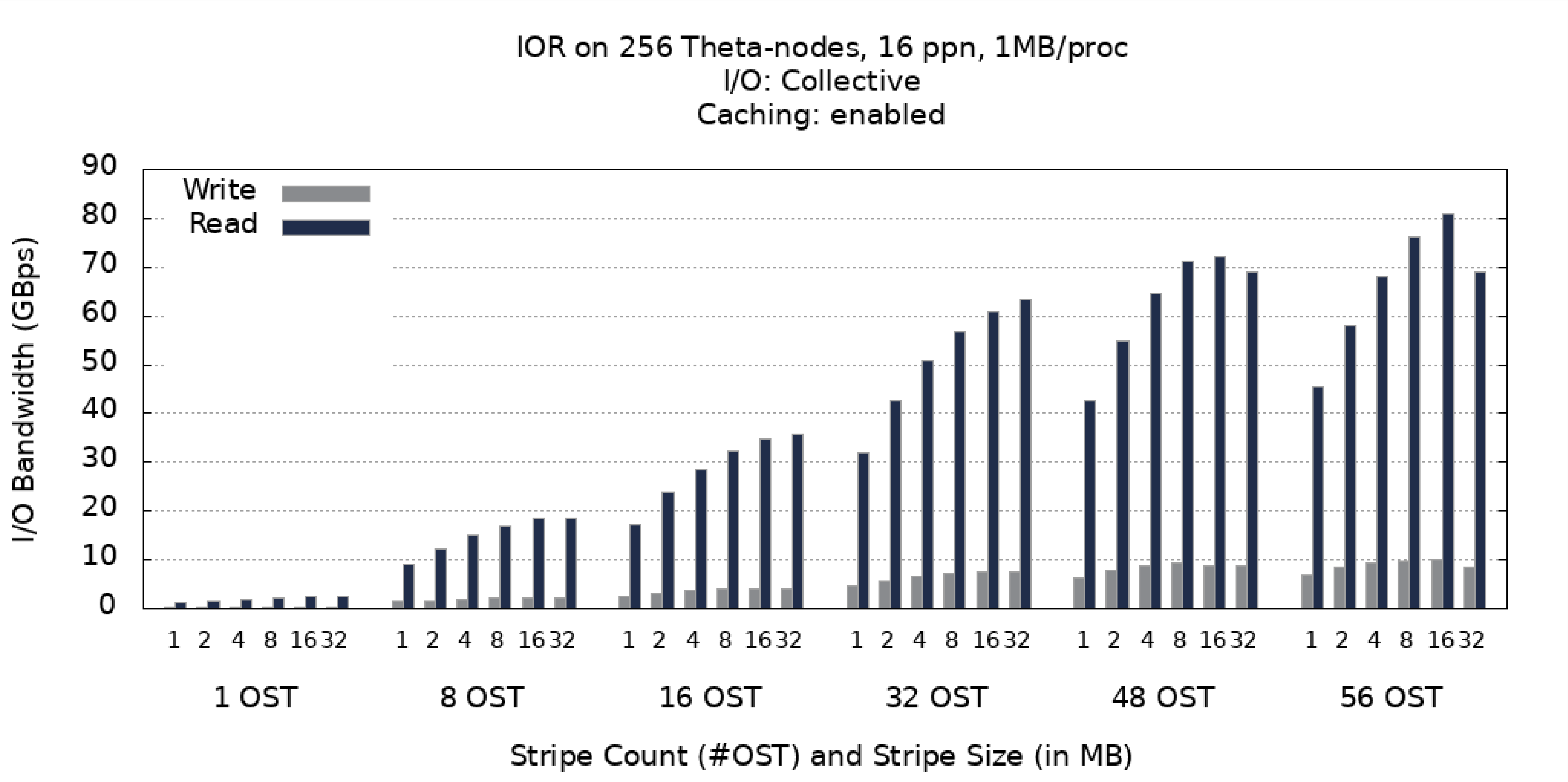


Shared File Stripe Size vs Count Affect on Performance (Collective I/O - No Lustre Client Cache Effects)

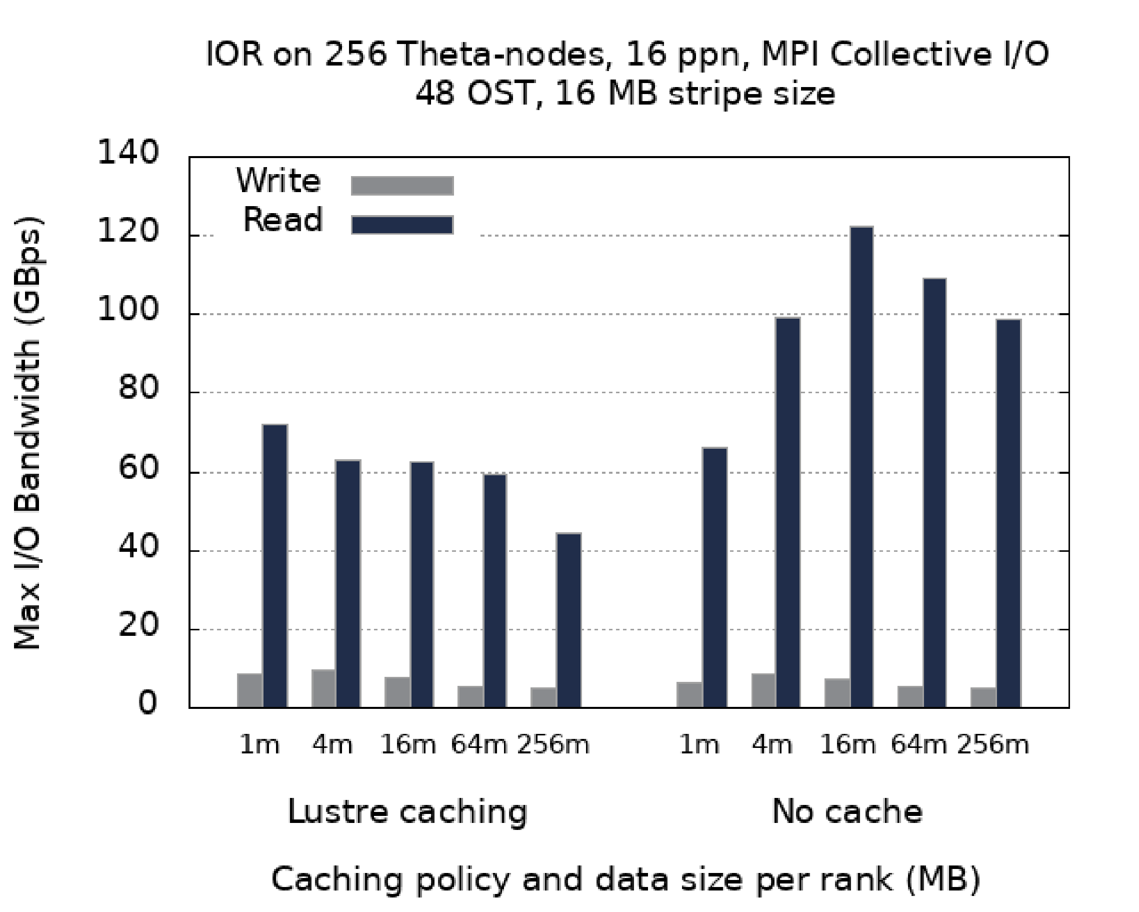


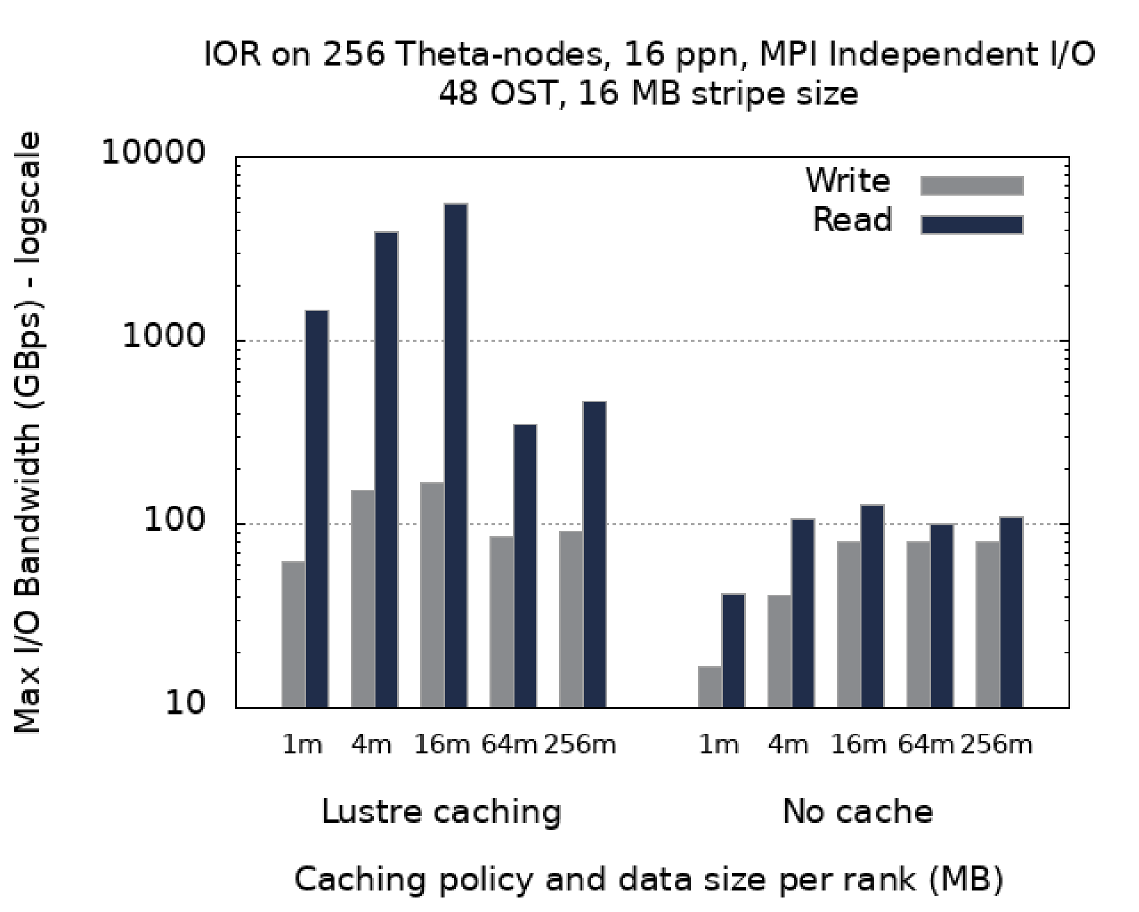
Argonne NATIONAL LABORATORY

Shared File Stripe Size vs Count Affect on Performance (Collective I/O – with Lustre Client Cache Effects)



Impact of data size on Lustre Cache Performance for 48 OST / 16 MB Stripe – Collective vs Independent IO



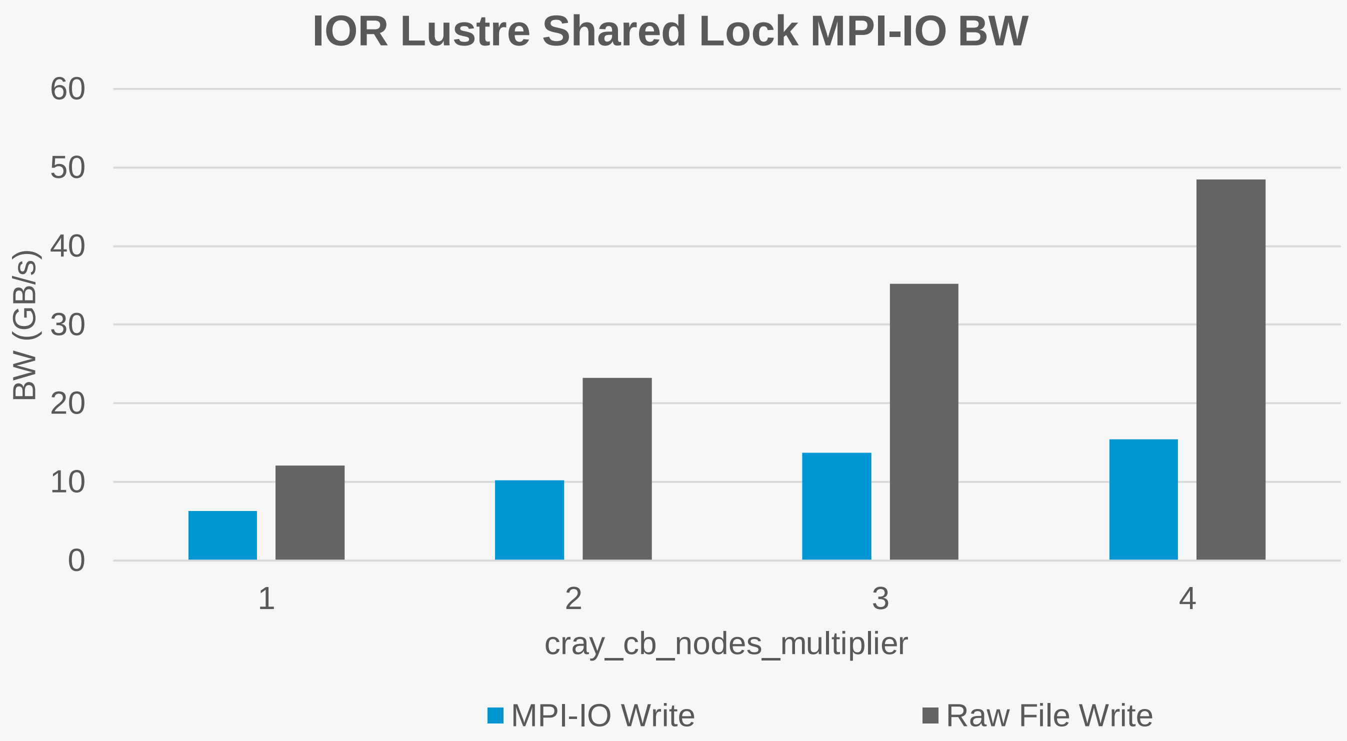


Mitigation of Extent lock contention within Cray MPI-IO

- Each rank (client) needs its own lock when accessing striped data for a given file on an OST
  - If more than one rank concurrently accesses same file on OST, causes extent lock contention, cancels out performance improvement
  - Concurrent access improves storage bandwidth
- Cray MPI-IO has a current limited mitigation for this (cray_cb_write_lock_mode=1 – shared lock locking mode)
  - A single lock is shared by all MPI ranks that are writing the file.
  - Lock ahead locking mode (cray_cb_write_lock_mode=2) not yet supported by Sonexion
  - Following slide run with: MPICH_MPIIO_HINTS=*:cray_cb_write_lock_mode=1:cray_cb_nodes_multiplier=<N>:romio_no_indep_rw=true

IOR MPI-IO Collective Shared Lock Performance Tests

IOR on 256 nodes 16 ppn 48 OSTs 1MB Stripe 1 MB Transfer size



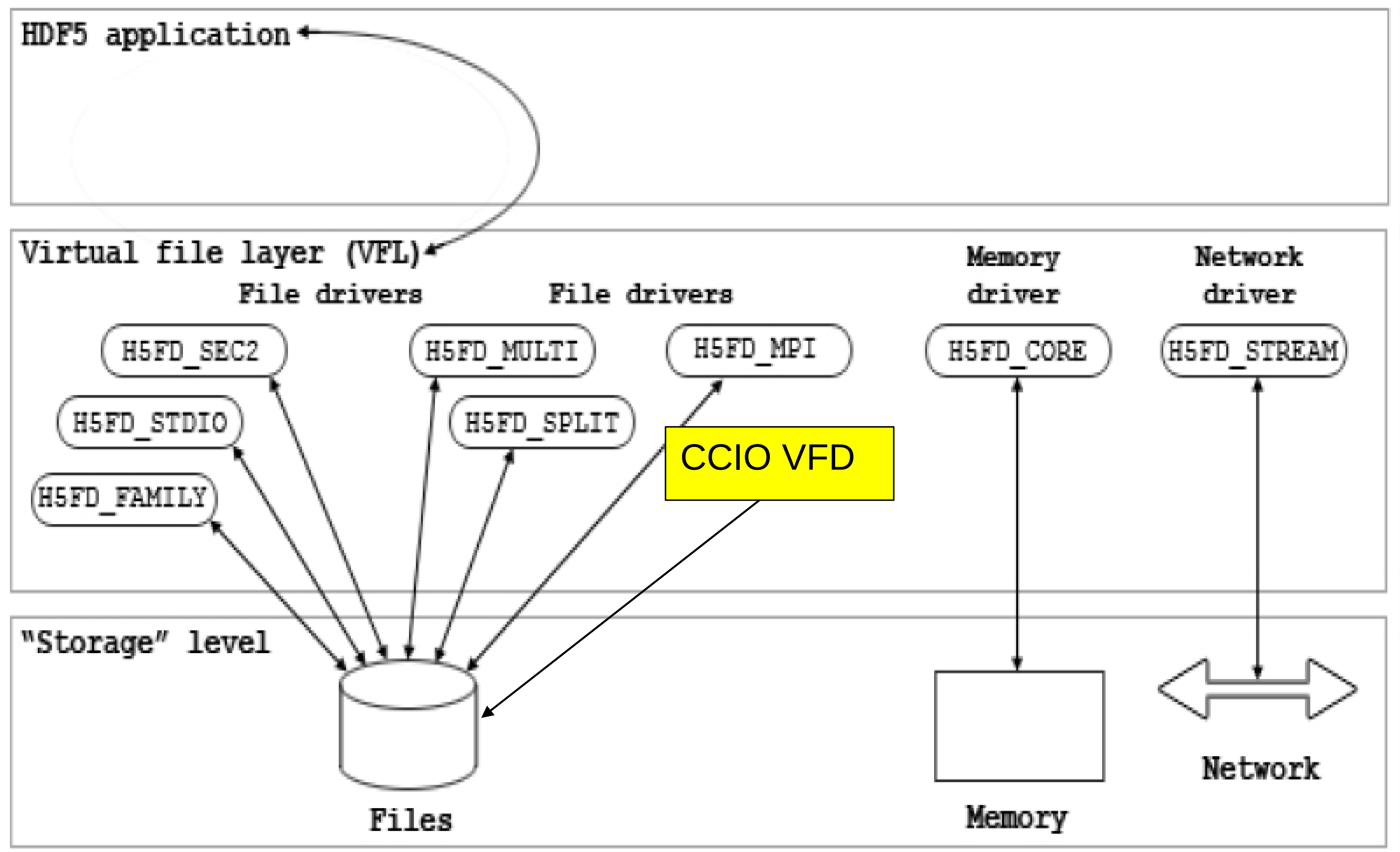
'Raw File Write' times taken from MPICH_MPIIO_TIMERS=1 trace

Raw File write linearly better - MPI-IO 1.5x faster at 4

HDF5 ECP Work – Custom Collective IO Virtual File Driver

- Lustre as a component of HDF5 performance
- CCIO VFD vs MPI VFD

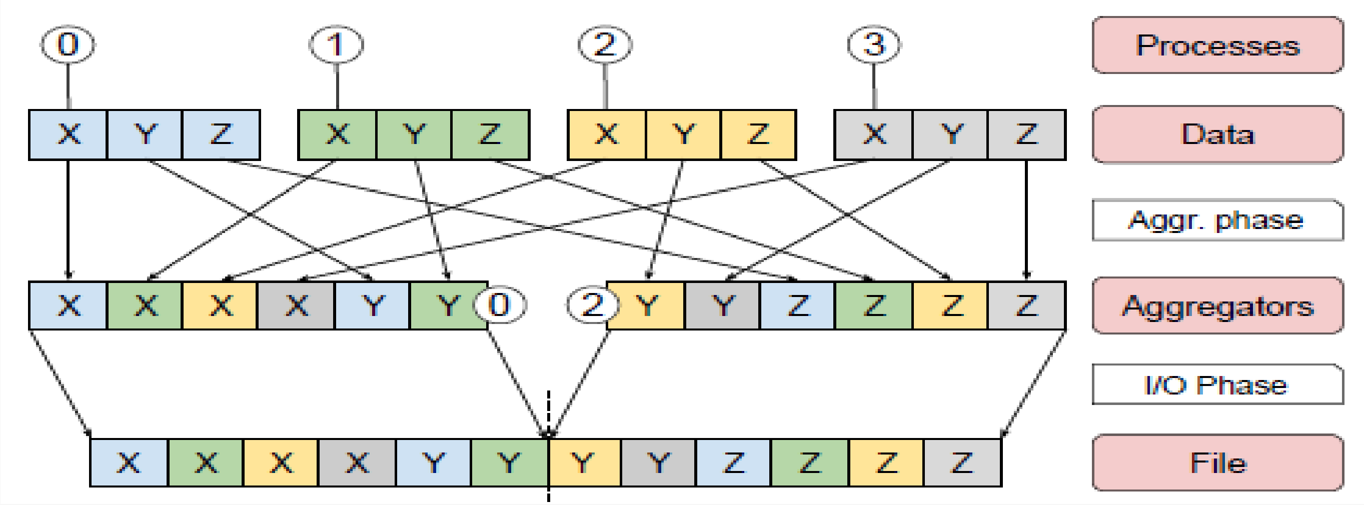
HDF5 Virtual File Layer



Parallel HDF5 CCIO VFD

- Custom Collective IO Virtual File Driver
  - Many HDF5 data access patterns get better performance with collective vs independent (large scale, discontiguous data)
  - Clone of most commonly used H5FD_MPI VFD to support customized collective IO file algorithms outside of MPI
- Highly instrumented for detailed performance profililng
- Current performance enhancements over MPI VFD
  - Avoids performance overhead of construction/deconstruction of MPI_Datatype
    - MPI constructs MPI_Datatype from dataset selection, MPI-IO implementation then needs to deconstruct to get offset/len pairs
    - For highly discontiguous data this can be expensive
  - Implementation of one-sided collective aggregation algorithm – detailed in following slides
- Many targets for future optimization

Standard Two-Phase Collective MPI-IO



- Standard two-phase algorithm as exists in MPICH MPI-IO (ROMIO)
  - Actually has a 3rd (0th) initial collective meta-data planning phase where aggregators determine what data goes where and when
    - Involves send/recv and/or collectives (eg MPI_Alltoall)
  - Data movement for aggregation phase done with send-recv or collectives (MPI_Alltoallv)
    - Done in 'rounds' defined by collective buffer size * number of aggregators

Argonne NATIONAL LABORATORY

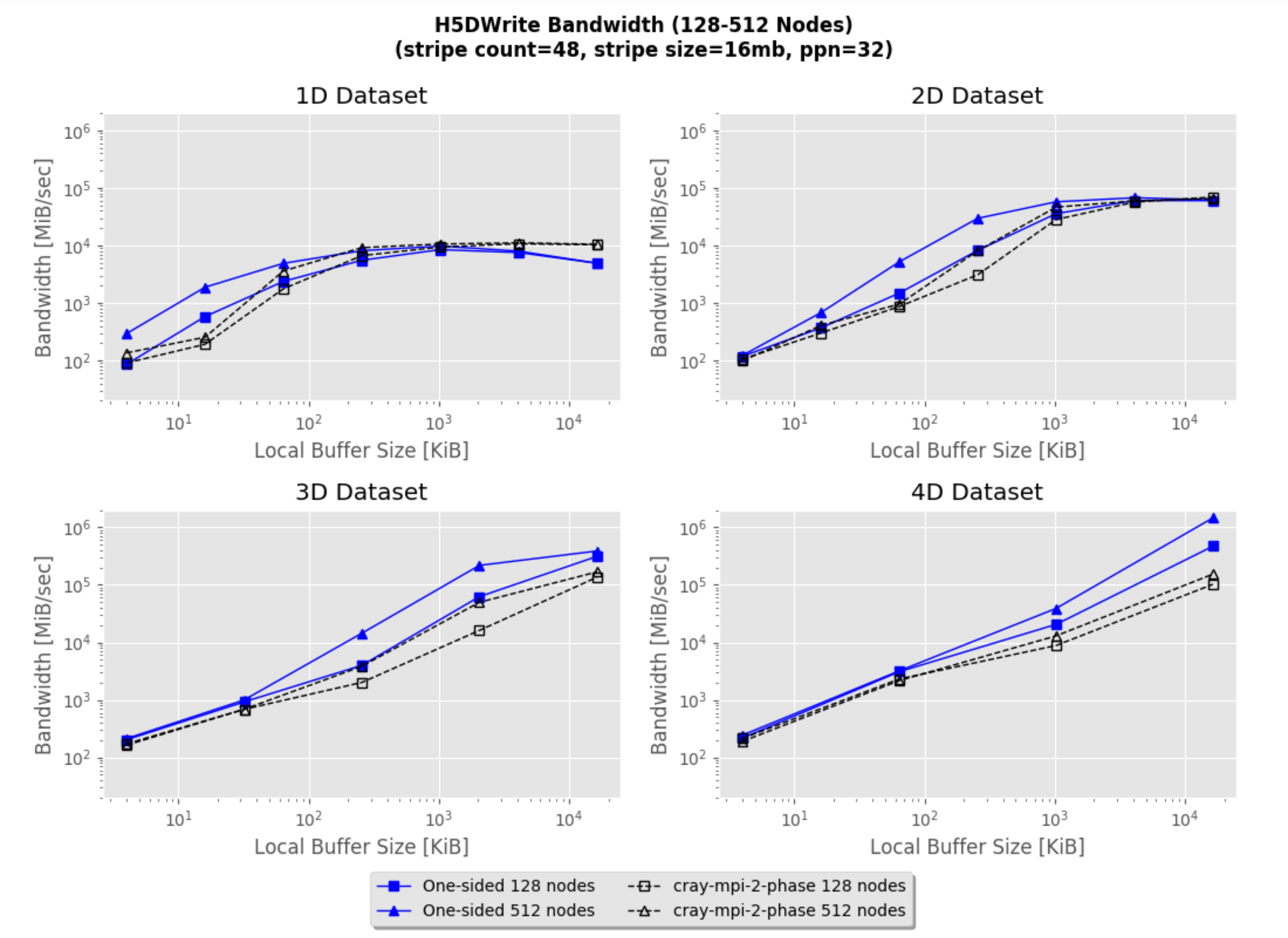
One-Sided Two-Phase Collective MPI-IO

- Currently implemented in MPICH-MPI-IO (ROMIO) with support for lustre (write only)
  - No collective meta-data planning phase
  - Data movement phase does RMA (MPI_Put) from computes into aggregator collective buffers
    - Dependent on architecture RMA implementation for performance
  - Aggregator memory footprint for standard algorithm can be significant at scale
    - Applications can run out of memory
    - Can be detrimental to lustre client cache-effects
  - Various performance improvement - depending on IO pattern and architecture can see 10x speedup or no speedup
  - Vendors evaluating it

Parallel HDF5 Exerciser

- Performance profiling c code exercising most intensive HDF5 functions in common user scenarios for both meta-data and raw data
- Created by ExaHDF5 ECP team
- Includes concepts from other HDF5 Performance benchmarks (IOR, VPIC-IO, FLASH-IO) and expands on them
- Highly customizable via many run-time options
  - Independent/Colletive IO for raw and meta-data, contiguous/chunked storage, multi-dimensions , discontiguous buffers/strides
  - Can craft many complex data access patterns
  - Can run at small and large scale
- Working on getting various HDF5 data access patterns from applications into the HDF5 Exerciser to reproduce and solve performance issues

HDF5 Exerciser One-Sided CCIO VFD vs Cray MPI-IO VFD



EXASCALE COMPUTING PROJECT

ExaHDF5

At 1-d smaller messages benefit because of relative increase in overhead for the collective meta-data planning phase, for larger message sizes degradation from 4mb to 16mb may be some sort of synchronization issue – each agg does 86 rounds of 16mb writes, syncing (MPI_Barrier) with each round

3+ dimensional datasets benefit the most from one-sided aggregation

Argonne NATIONAL LABORATORY

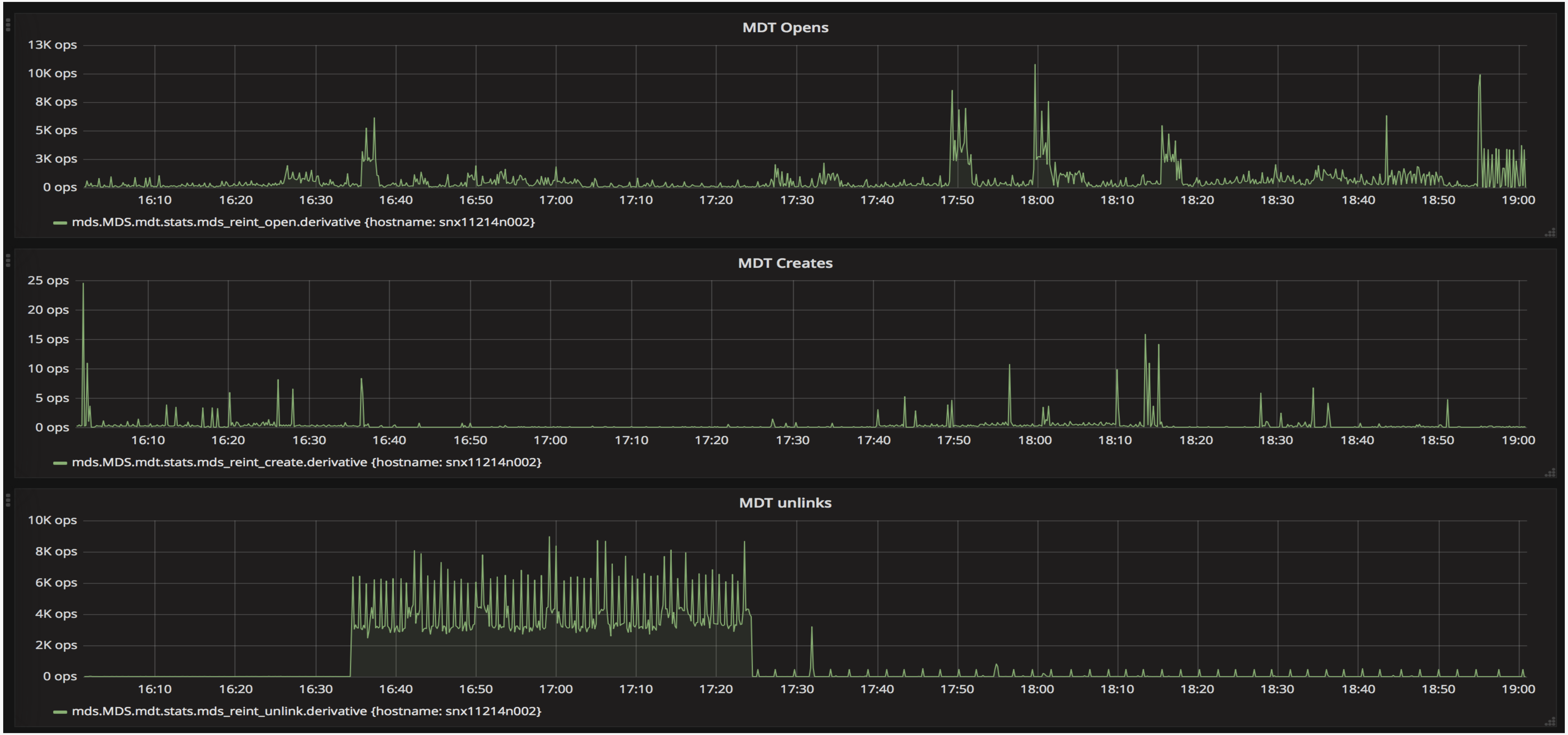
Operations Metrics

- A few examples of the types of Lustre metrics being collected on Theta
- Working on direct correlation to IO performance on job basis
  - Currently indirect correlation we will show in later slides

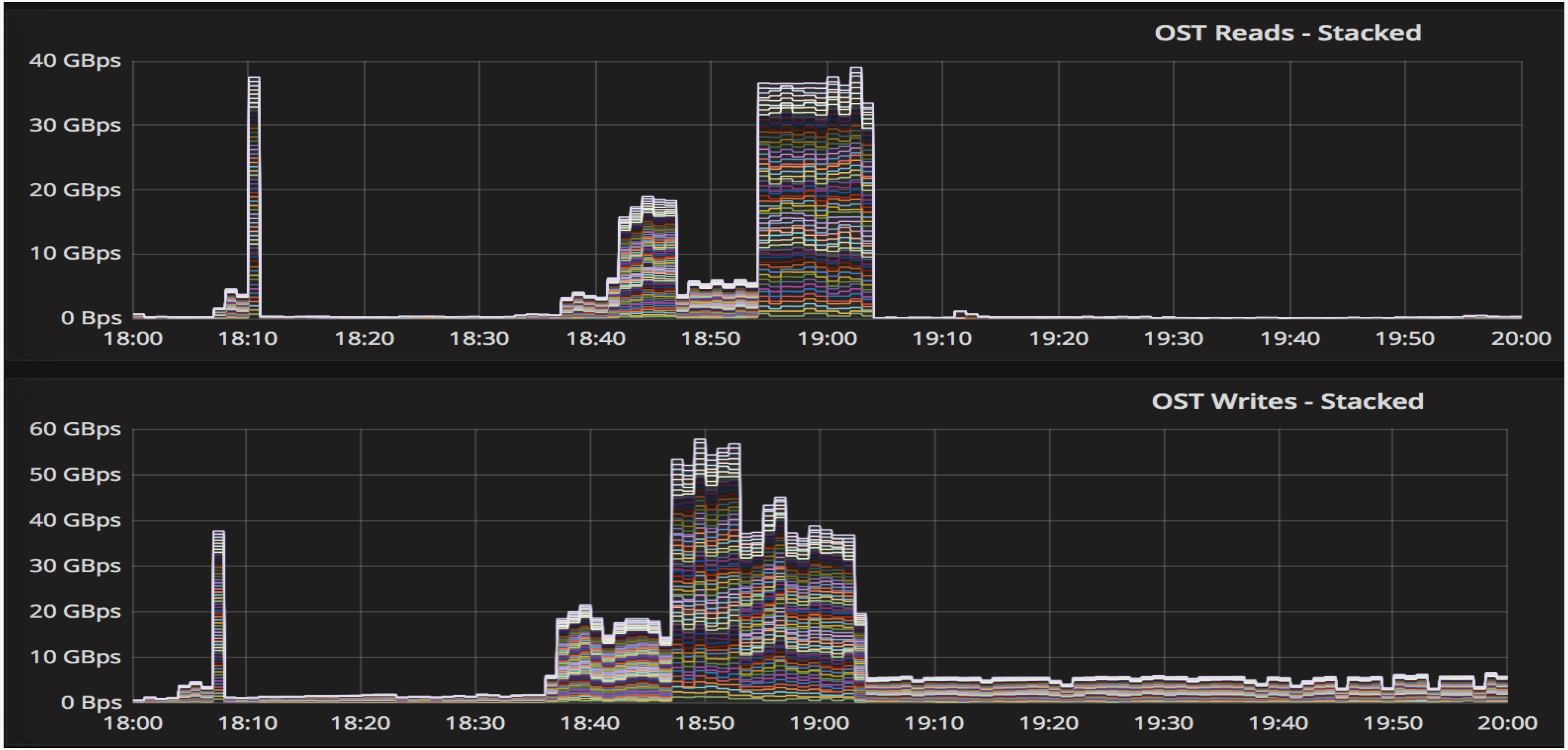
Lustre Metrics

- Operations team records the majority of Lustre stats but focus on monitoring a subset of them
- MDS
  - Monitor all typical metadata operations, e.g. opens, creates, unlinks, renames, (get|set)(x)attr
- OSS
  - Monitor reads/writes grouped by OST and OSS
  - Monitor number of files and space
- Eventually be able to directly tie back to job id and direct correlation with user achieved performance

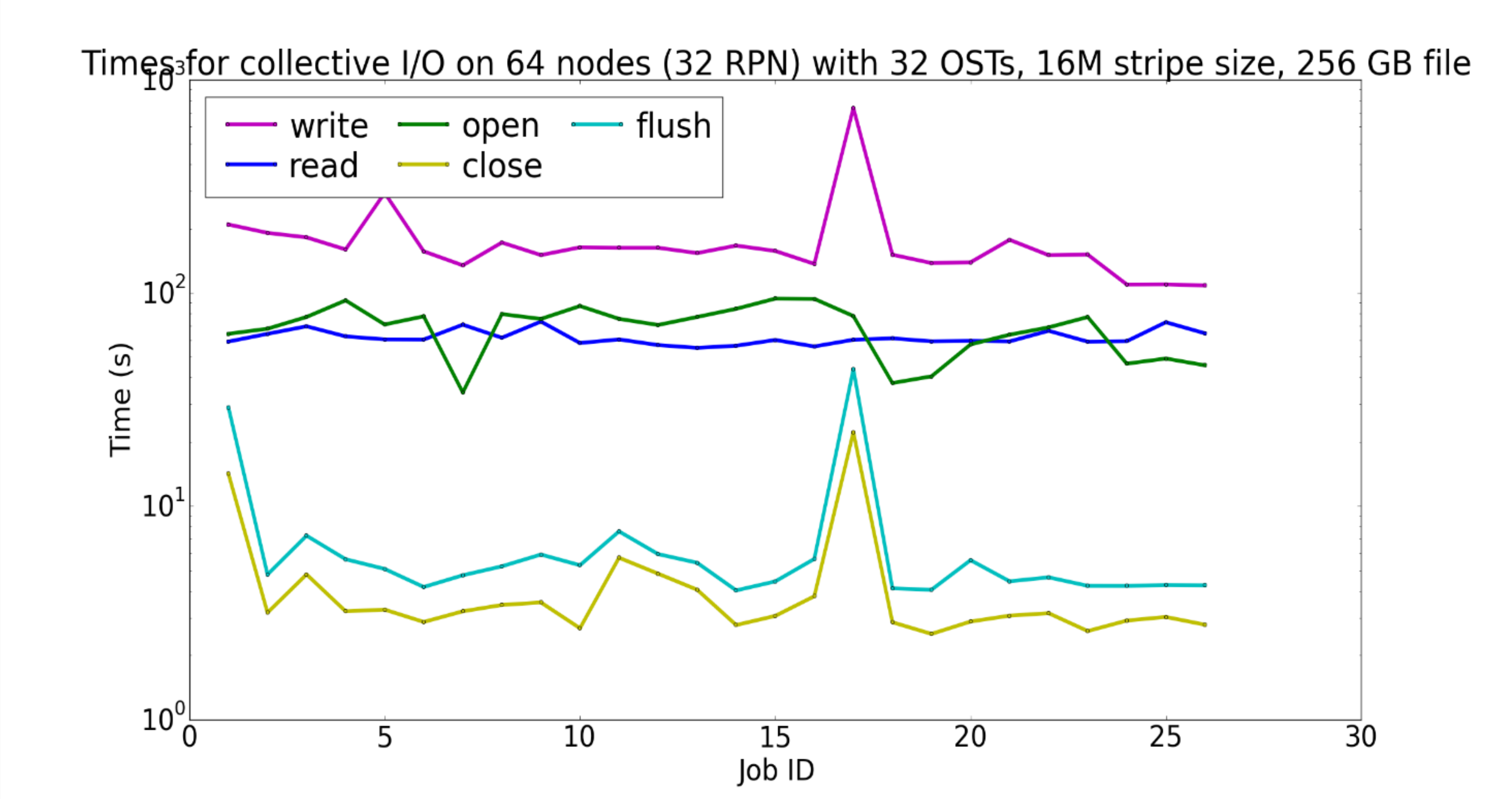
MDT Metrics Dashboard



OST metrics during IOR large data test 18:36 to 19:05

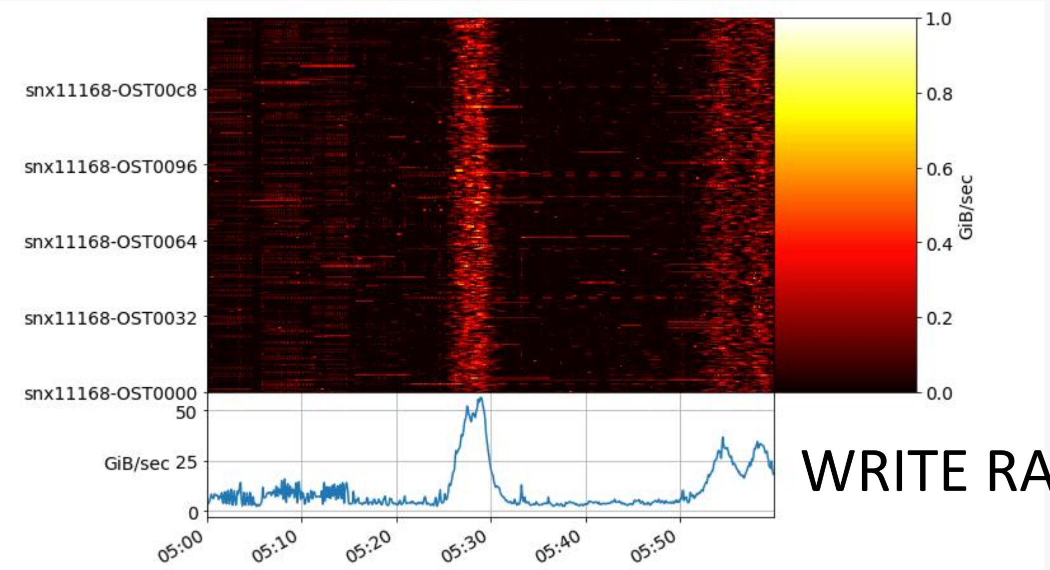


HDF5 Exerciser metrics for 26 jobs on Cori KNL



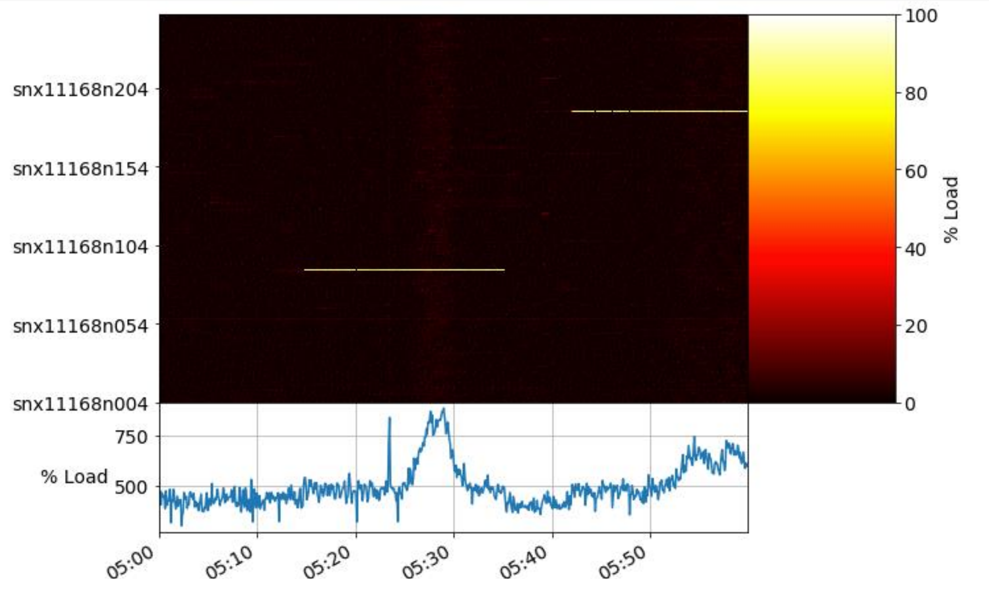
Argonne NATIONAL LABORATORY

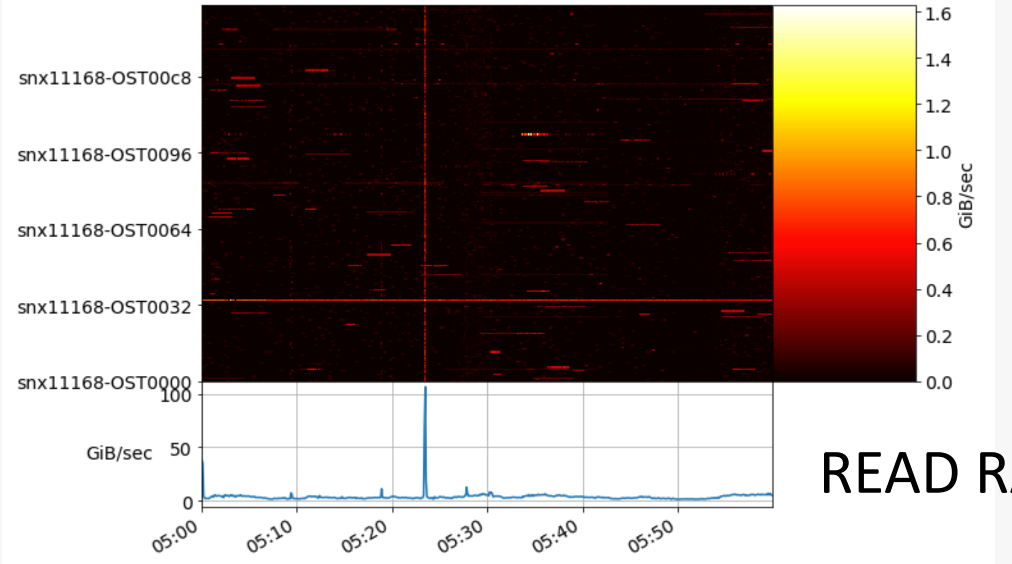
OST, OSS, MDS Statistics via NERSC pytokio API



WRITE RATE

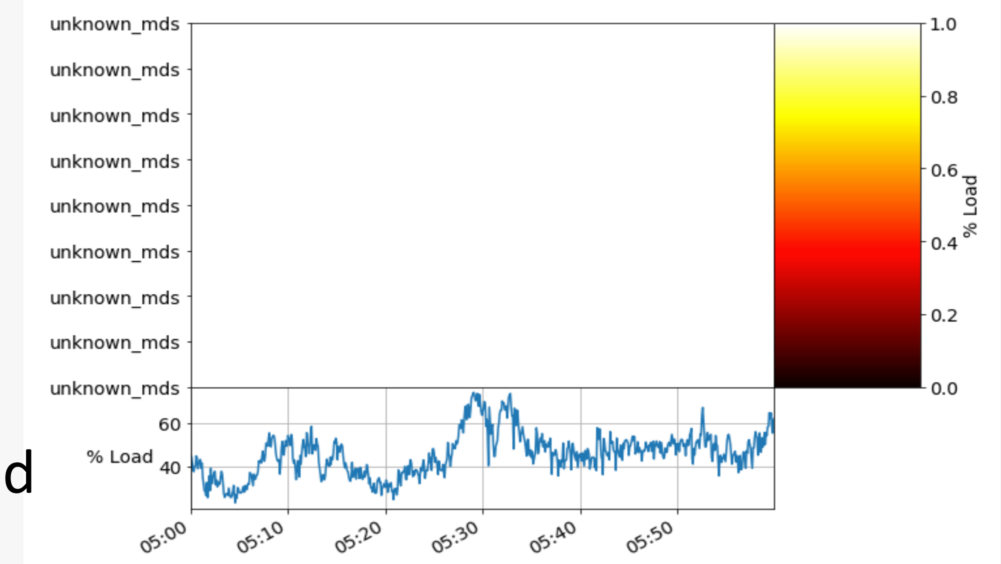
OSS Load





READ RATE

MDS Load



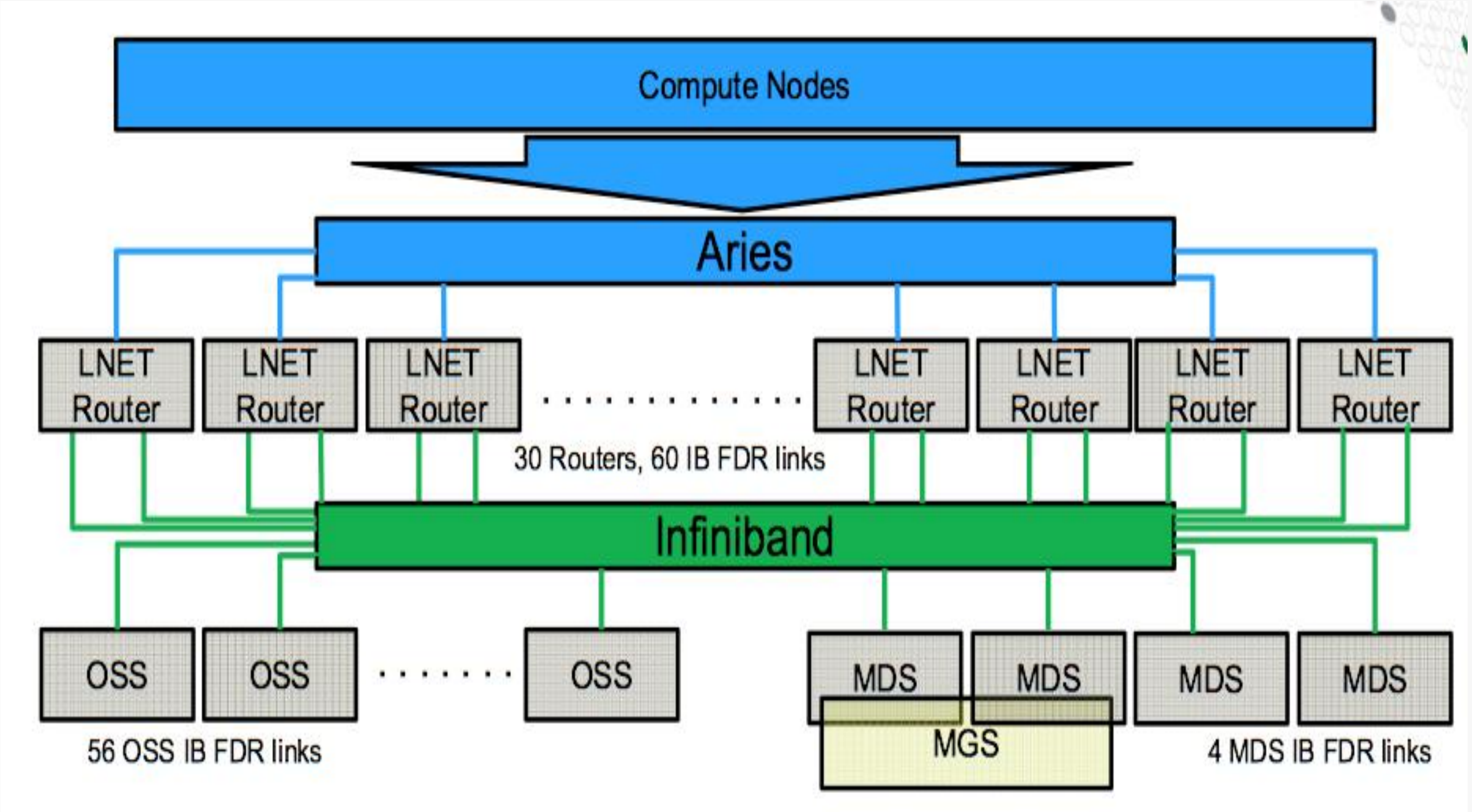
Acknowledgements

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Questions?

Appendix

Lustre Architecture On Theta



- **IO Forwarding from compute node to LNET Service Node / Router**
  - LNet Aries NIC on compute side, 2 IB links on Object Storage Server (OSS) side
  - OSS handles communication from LNet Router to Object Storage Target (OST) which is the physical storage device
  - Although there are 4 MDTs only 1 currently has directories placed on it

Argonne NATIONAL LABORATORY