Academic Computer Centre
CYFRONET AGH

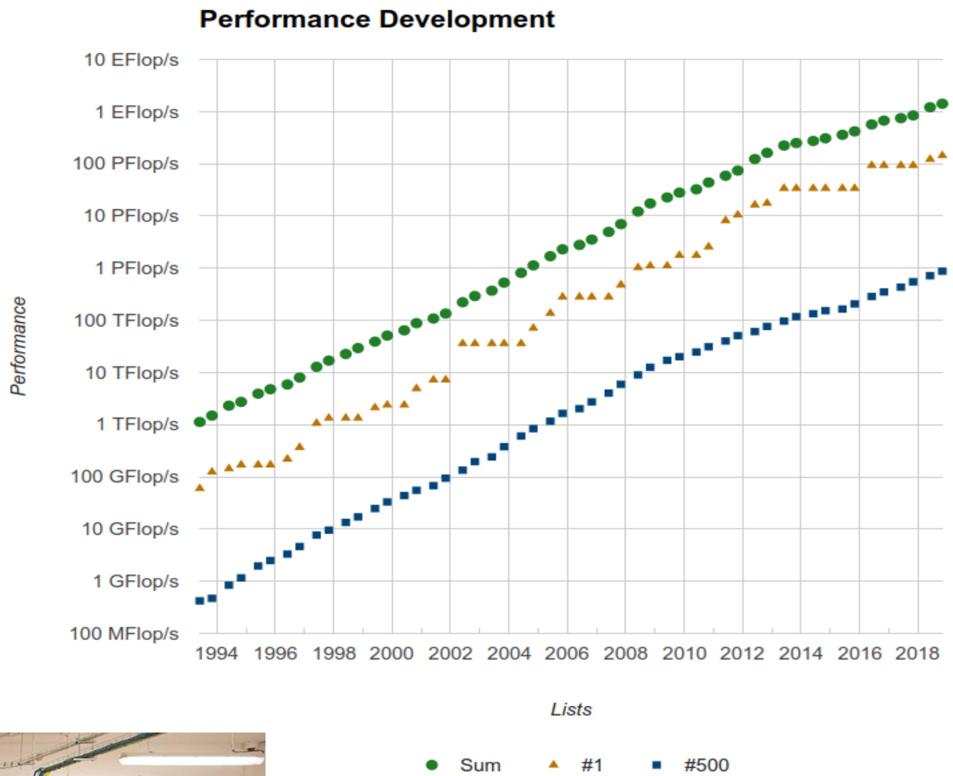
How to exploit parallelism of HPC storage based on Lustre File System and Hierarchical Data Format

A. Dorobisz, M. Pawlik, M. Czuchry, D. Kałafut, K. Noga

Presentation plan

- Introduction
- HPC Storage in practice
  - Lustre case study
- HDF
- Test results
- Conclusions

- HPC Systems are a key instrument in multiple fields of science
- Exponential growth of performance
- What is a HPC System?
  - compute (cpu + memory)
  - network
  - storage
  - software... etc.
- How is storage doing?



Test system

- Infiniband 4x FDR (56Gb/s)
- Lustre FS as main storage:
  - scratch: 5 PB @ 120 GB/s
  - archive: 5 PB @ 60 GB/s
- No. of disks:
  - scratch: 1600
  - archive: 1080
- We do use NFS!
  - $HOME dirs
  - software

CYFRONET

Lustre filesystem

- Parallel distributed file system used in large-scale computing
- In contrast to a standard filesystem:
  - components of the system communicate with each other using network
  - multiple clients can share files and homogeneous space
    - data access, file locks, permissions etc.
  - data is stored on OSTs (actually disks)

The key components of Lustre filesystem:

- Object Storage Servers (OSS)
- Object Storage Targets (OST)
- Metadata Servers (MDS)
- Metadata Targets (MDT)

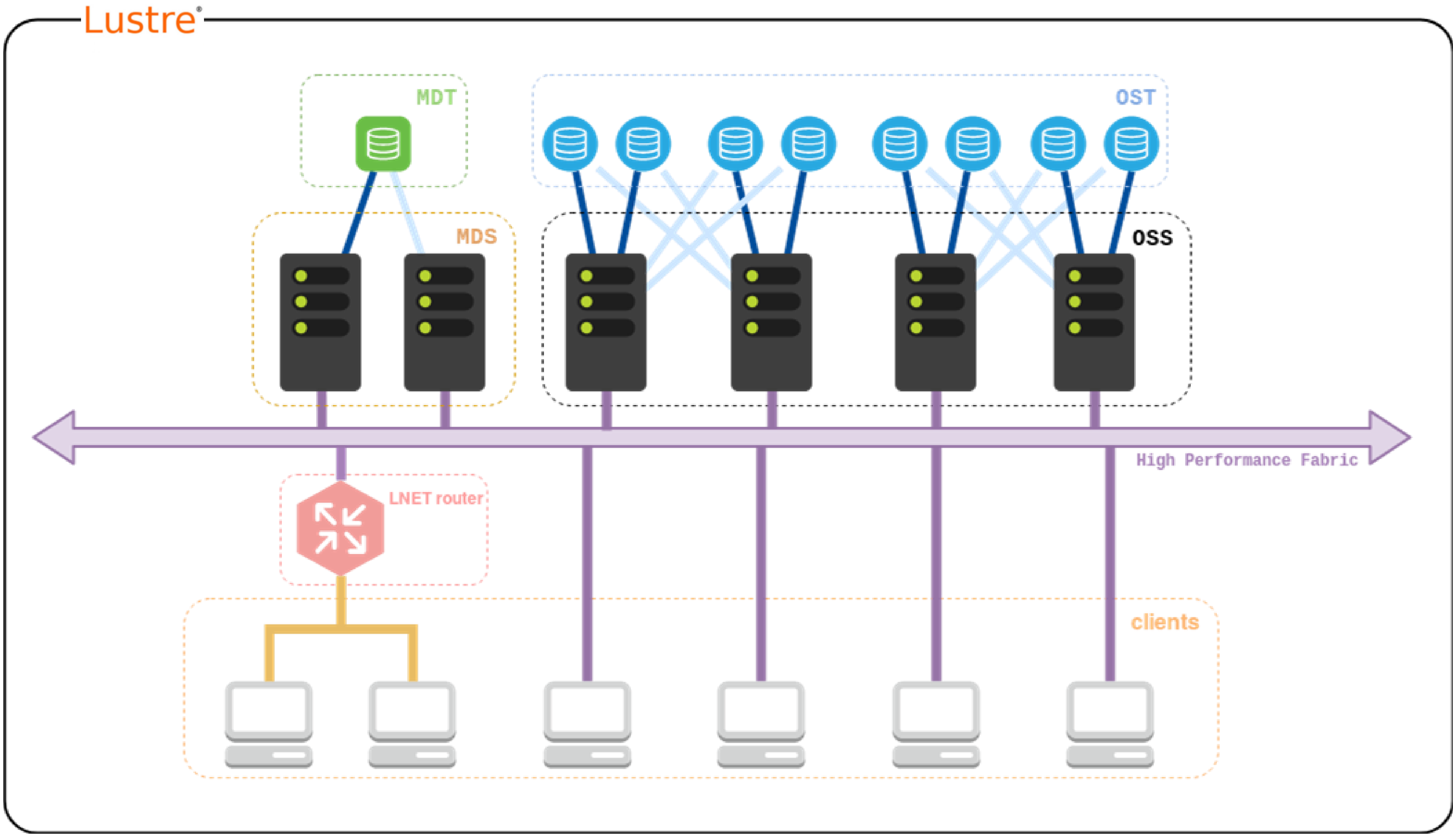
scratch

- 16 OSS
- 160 OST
- 2 MDS
- 1 MDT

archive

- 8 OSS
- 108 OST
- 2 MDS
- 1 MDT

Lustre fs architecture



Lustre installation

- Works mainly in kernel space
- Can be used in small and large-scale enterprise environments
- Available backend filesystems:
  - ldiskfs
    - based on ext4 filesystem
  - ZFS
    - additional recovery, compression, improved performance and security of stored data

Lustre benefits

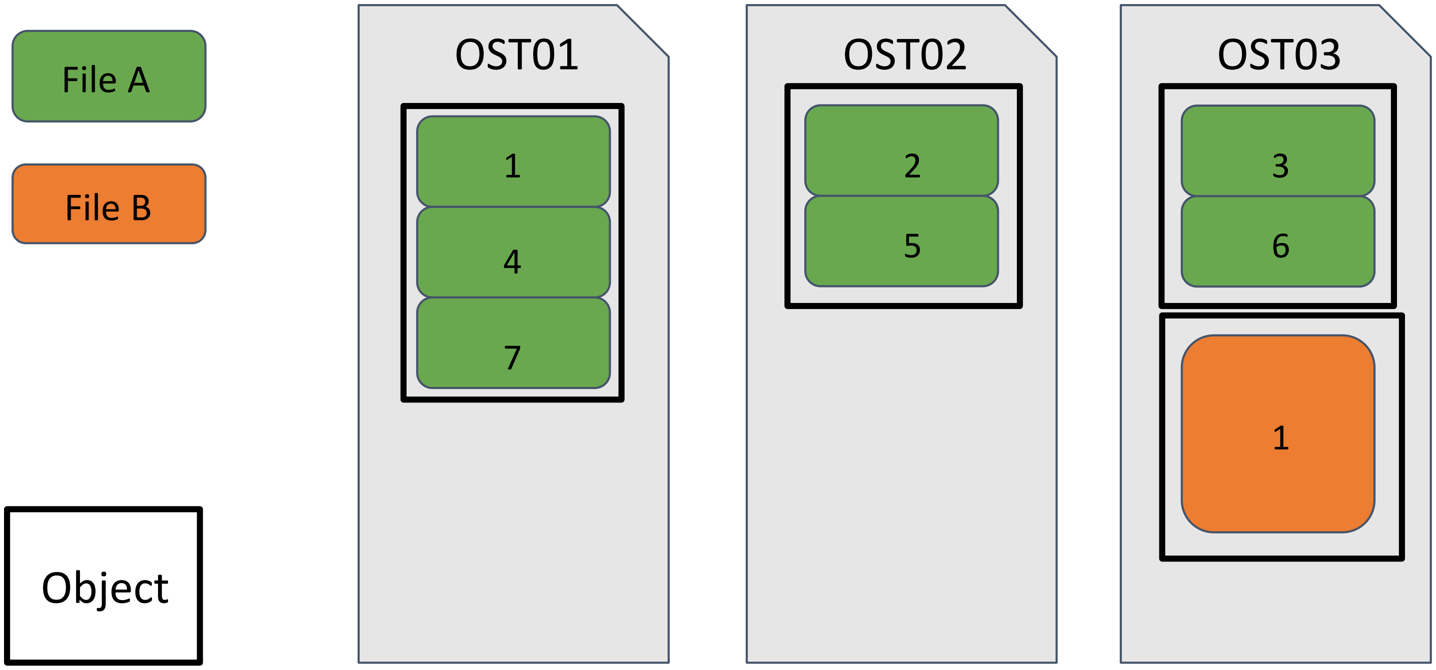
- Allows for significant increases in performance of read/write operations
  - reading from and writing to disks in parallel
- Security and high availability (HA)
  - HA stack reduces risks origination from single point of failure
  - storage consistency verification and security mechanisms
- Simple and well-known user interface
  - operations on files via standard Linux/Unix commands and functions supplied by implementing POSIX

Lustre file stripping

- Stripe count
  - number of objects that make up a file hosted by Lustre
- Stripe size
  - amount of sequential data that is written to an object before moving on to another object

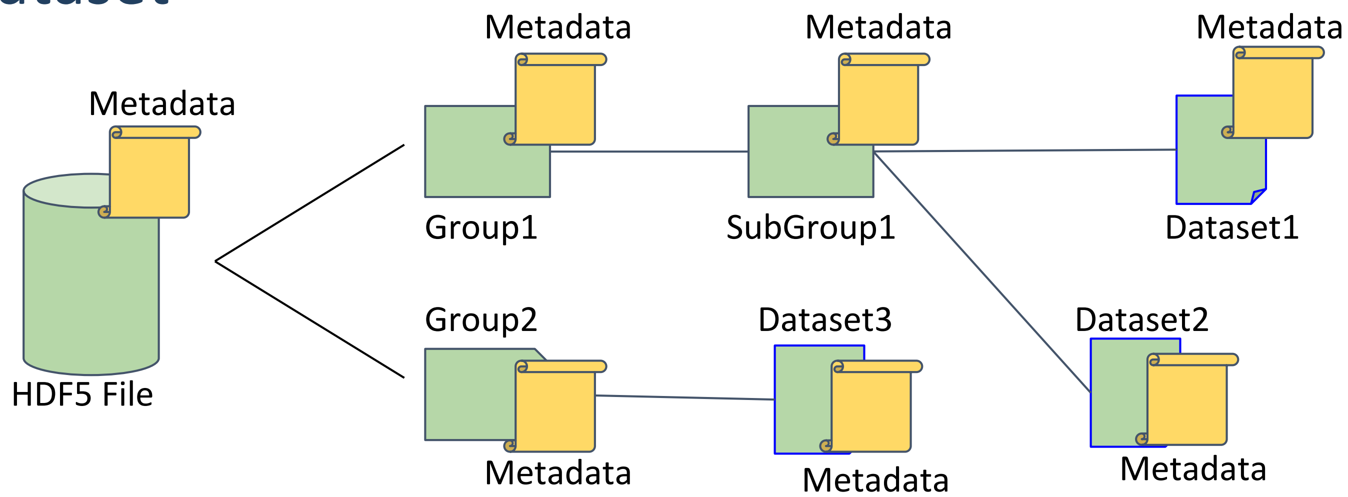
Lustre file stripping

- Parallel reading/writing across multiple OSTs
- Increase aggregate bandwidth linearly
- File size not limited to constraints of single OST



HDF (HDF5): Hierarchical Data Format, set of libraries and tools

- Data stored in a hierarchical and structured way
  - Groups and subgroups - container structures, correspond to directories
  - Datasets - form of a multidimensional arrays with fixed dimensions of a homogenous type, correspond to files
  - Metadata - attributes allowing to describe a group or a dataset



HDF benefits

- Hierarchical data storage system
  - data organization
  - self descriptive data
- Implements functions for parallel access to data
- One big file instead of many small files
  - heterogeneous data (+ metadata)
  - great for Lustre!
- Portable software library
  - high-level API with C, C++, Fortran 90, Java, Python (and many more) interfaces

HDF access to data

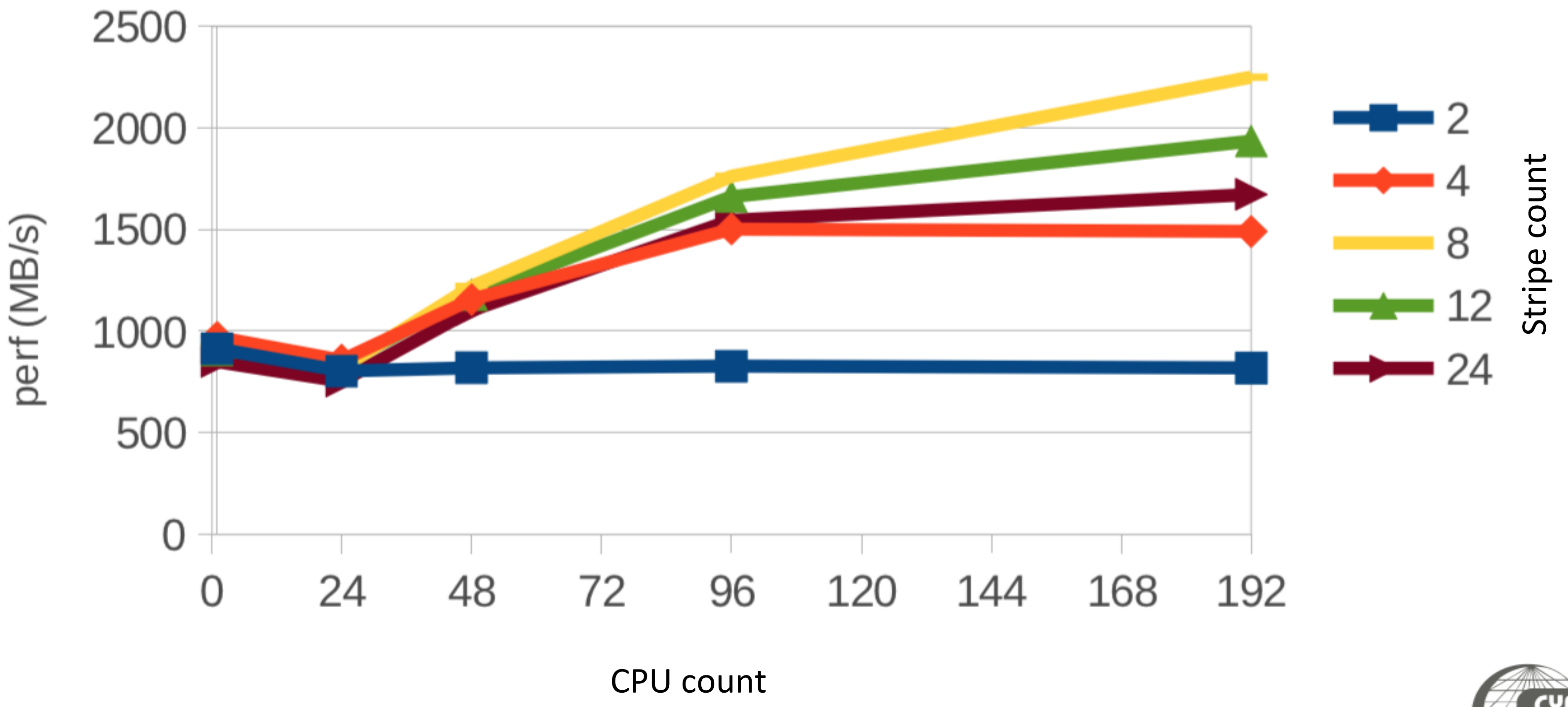
- Parallel access with multiple approaches:
  - all CPUs write to one file
  - one node writes to one file; each node has its own file
  - each CPU writes to one file
- Modes:
  - Collective (additional optimization)
  - Independent

Test setup

- Goal: optimize data write from a job with multiple processes
- Gathered data from tests performed on archive space
- Environment:
  - library: Parallel hdf5-1.8.12
  - Data set size: 4394,53MB
  - Stripe size: 1MB
  - Process count: 1/24/48/96/192 (multiples of 24 cores/node)

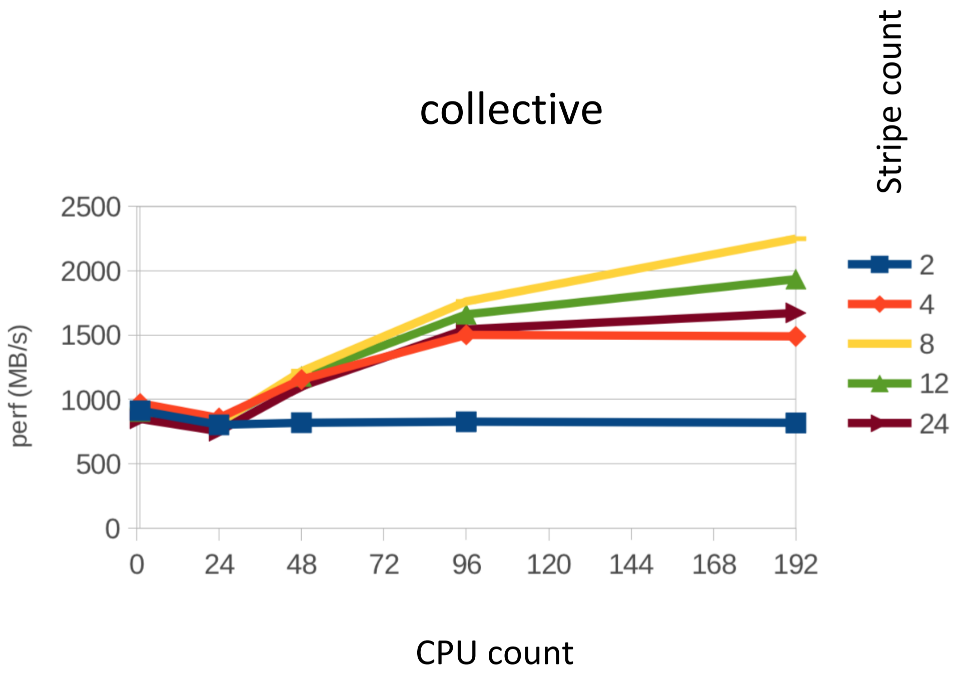
Stripe count

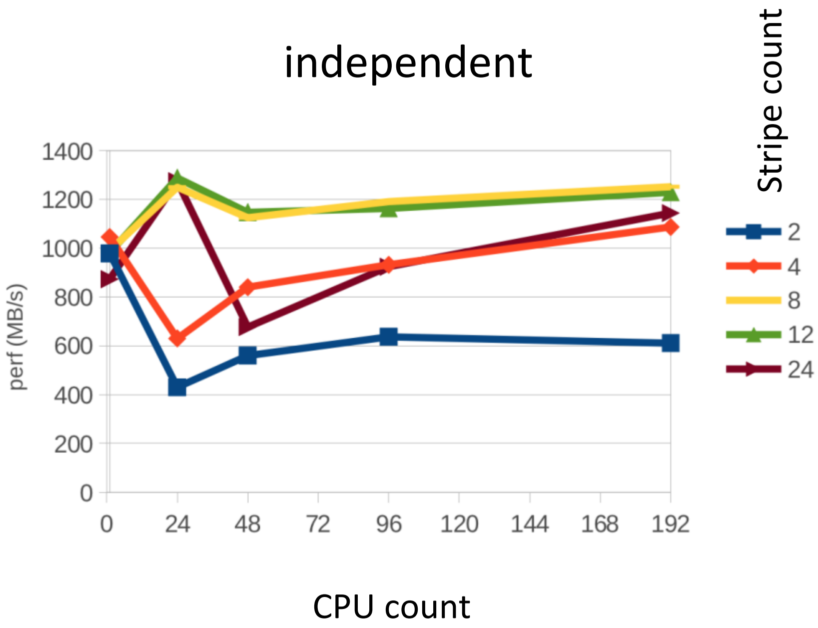
- Setup
  - One output file for all CPUs
  - Collective mode
- Goal: What amount of stripes works best?



Collective vs independent

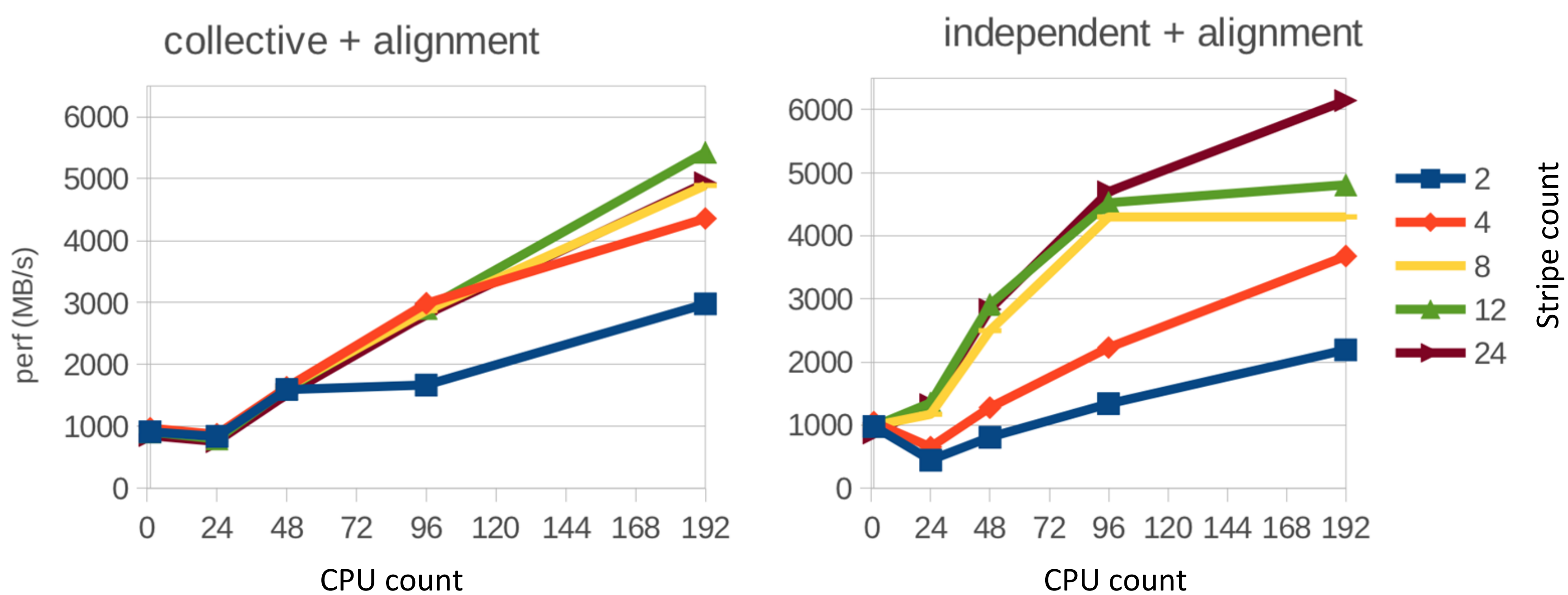
- Setup
  - One output file for all CPUs





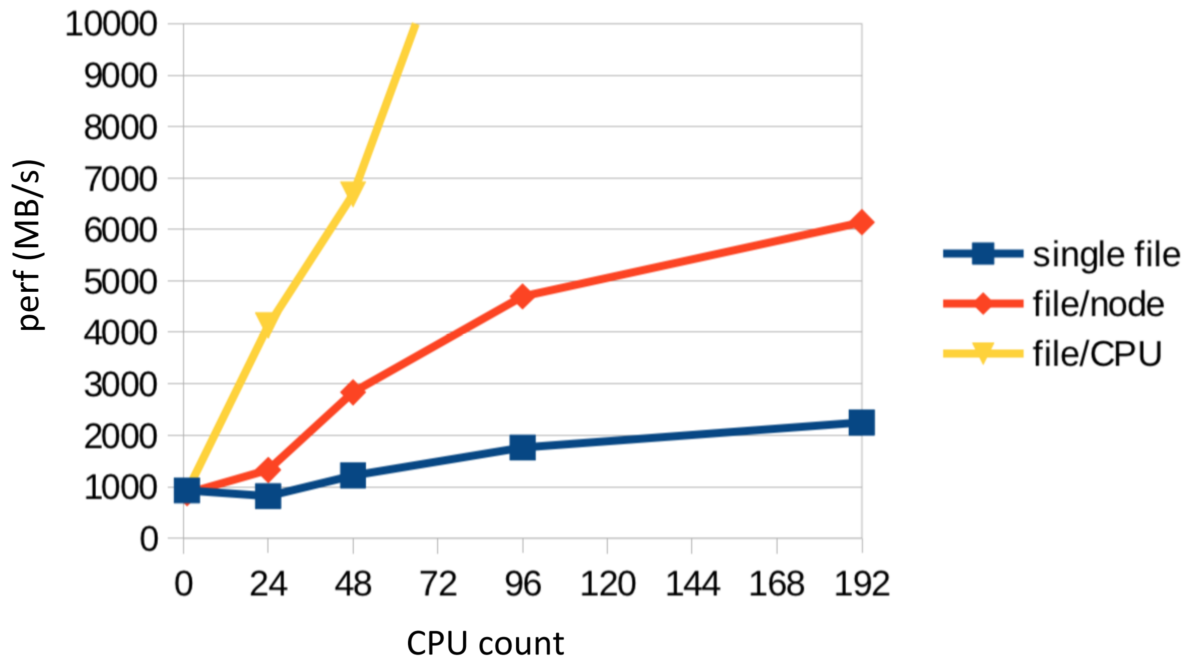
Collective vs independent

- Setup
  - One file per node (24 CPUs)



Write strategy comparison

- Setup
  - all CPUs write to one file
  - one node writes to one file; each node has its own file
  - each CPU writes to one file
- What strategy works best for studied use case?
(best settings for individual strategies)



Conclusions

- It is possible to achieve significant performance improvements by optimizing access to storage
- File per cpu strategy/node/job provides biggest difference
- Many small files suggest collective, few larger ones independent
- Higher stripe count usually provides better performance, for jobs with many CPUs working with few files
- There is no “best approach” to implementing IO operations in HPC applications

References

1. Lockwood, G. K., Hazen, D., Koziol, Q., Canon, R. S., Antypas, K., Balewski, J., et al., "Storage 2020: A Vision for the Future of HPC Storage" (2017).
2. http://lustre.org (accessed January 2019).
3. https://www.hdfgroup.org/solutions/hdf5 (accessed January 2019).
4. Howison, M. "Tuning HDF5 for Lustre File Systems" (2010).
5. http://www.nersc.gov/users/storage-and-file-systems/i-o-resources-for-scientific-applications/optimizing-io-performance-for-lustre
6. http://www.nersc.gov/users/training/online-tutorials/introduction-to-scientific-i-o

Thank you for attention! Questions?

CYFRONET

short url

goo.gl/gxBs1k