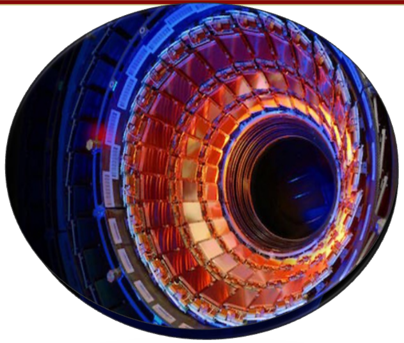
Tuning HDF5 Subfiling Performance on Parallel File Systems

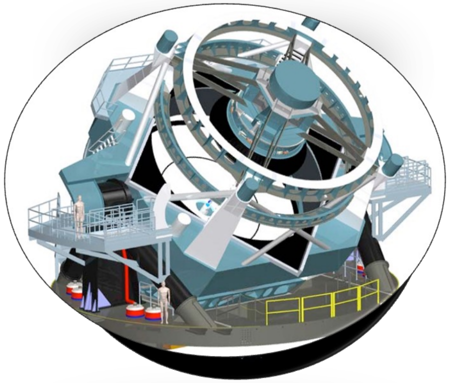
Suren Byna et al., “Tuning HDF5 subfiling performance on parallel file systems”, Cray User Group Conference 2017 (CUG 2017) [Preprint]

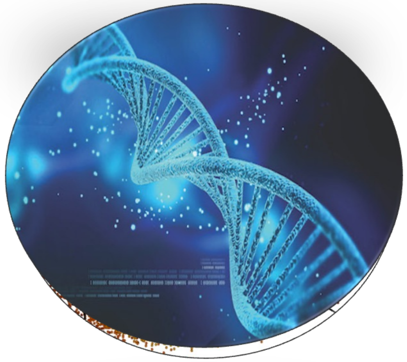
Initial version of subfiling in HDF5, which is different from the current implementation.
New implementation

Scientific applications and massive data

- Simulations
  - Multi-physics (FLASH) – 10 PB
  - Cosmology (NyX) – 10 PB
  - Plasma physics (VPIC) – 1 PB
- Experimental and Observational data
  - High energy physics (LHC) – 100 PB
  - Cosmology (LSST) – 60 PB
  - Genomics – 100 TB to 1 PB
- Scientific applications rely on efficient access to data
  - Storage and I/O are critical requirements of HPC



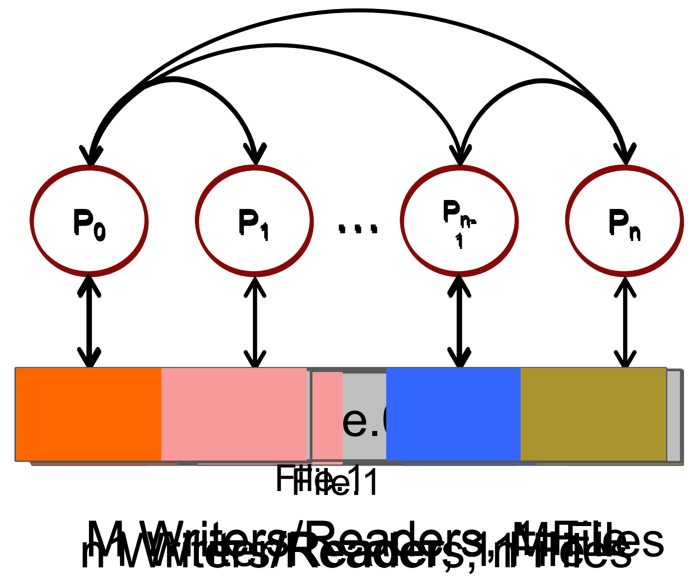




Parallel I/O – Application view

Types of parallel I/O

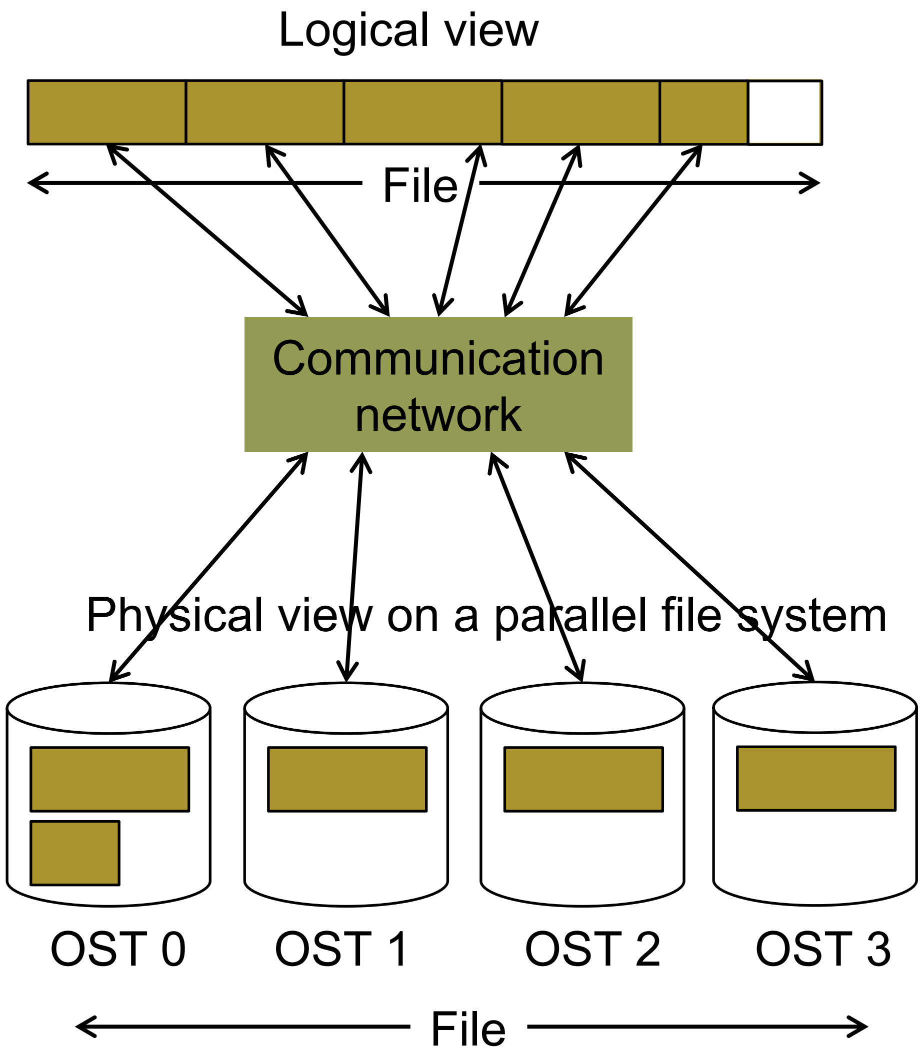
- 1 writer/reader, 1 file
- N writers/readers, N files (File-per-process)
- N writers/readers, 1 file
- M writers/readers, 1 file
  - Aggregators
  - Two-phase I/O
- M aggregators, M files (file-per-aggregator)
  - Variations of this mode



Source: http://www.erdc.hpc.mil/docs/Tips/garnet-lustre-adios.pdf

Parallel I/O – System view

- Parallel file systems
  - Lustre and GPFS
- Typical building blocks of parallel file systems
  - Storage hardware – HDD or SSD RAID
  - Storage servers
  - Metadata servers
  - Client-side processes and interfaces
- Management
  - Stripe files for parallelism
  - Tolerate failures



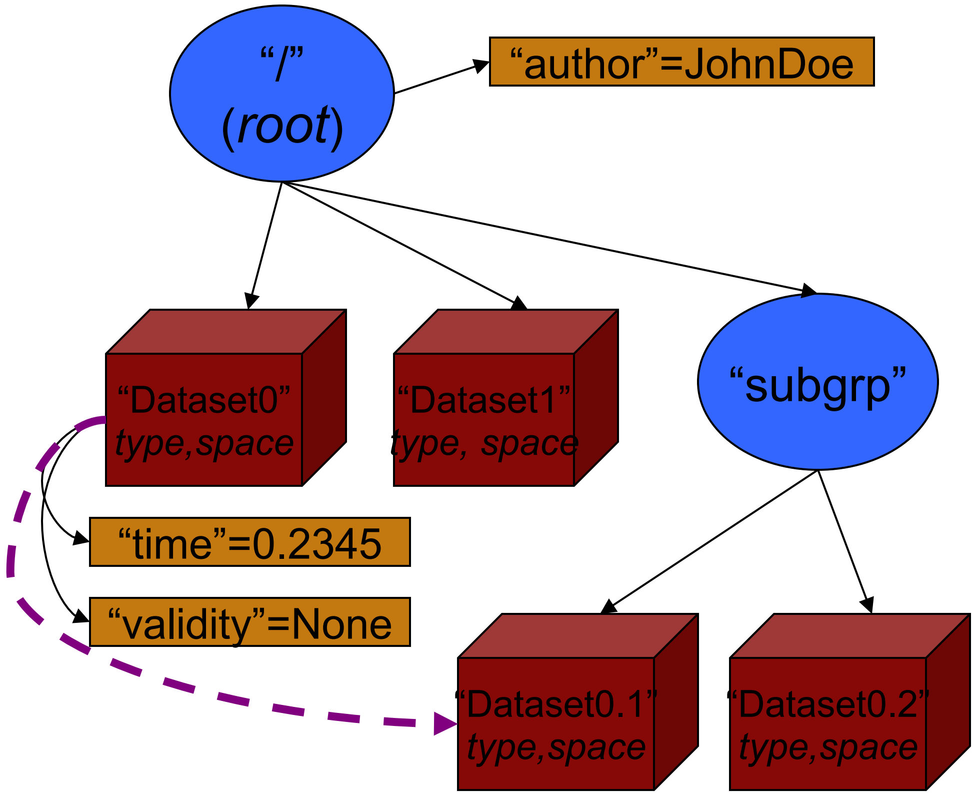
Source: https://www.nersc.gov/users/data-and-file-systems/i-o-resources-for-scientific-applications/
https://fs.hlrs.de/projects/craydoc/docs/books/S-2490-40/html-S-2490-40/chapter-4m6xftd3-oswald-basicsoflustreparallelio.html

Hierarchical Data Format v5 (HDF5)

HDF5 is a data model, file format, and I/O library

- Groups
  - Arranged in directory hierarchy
  - root group is always '/'
- Datasets
  - Dataspace
  - Datatype
- Attributes
  - Bind to Group & Dataset
- References

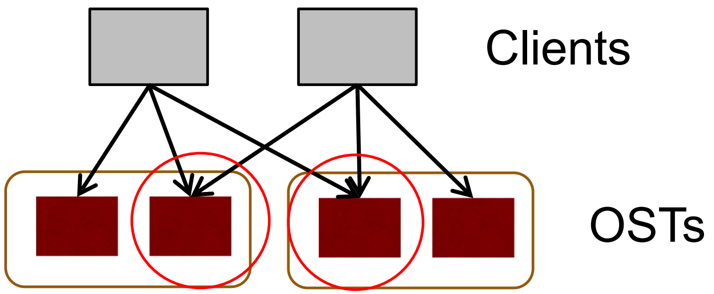
- Flexibility to design and implement data models



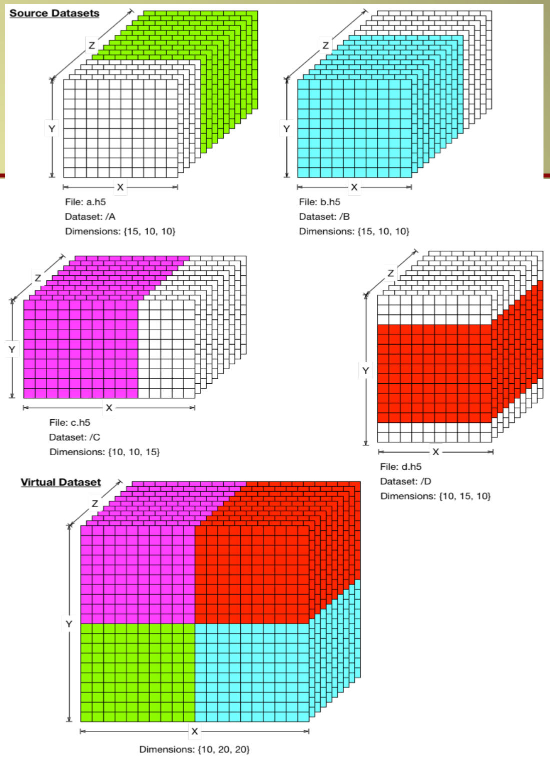
Slide courtesy of John Shalf

Subfiling

- Writing to single shared file may be slow due to:
  - Locking contention
  - Complications in moving large files
- A solution: Subfiling
  - Multiple small files
  - A metadata file stitching the small files together
- Benefits
  - Better use of parallel I/O subsystem
  - Reduced locking and contention issues improve performance
- Related work
  - File system: PLFS
  - I/O middleware: PnetCDF, ADIOS
  - Application libraries: BoxLib

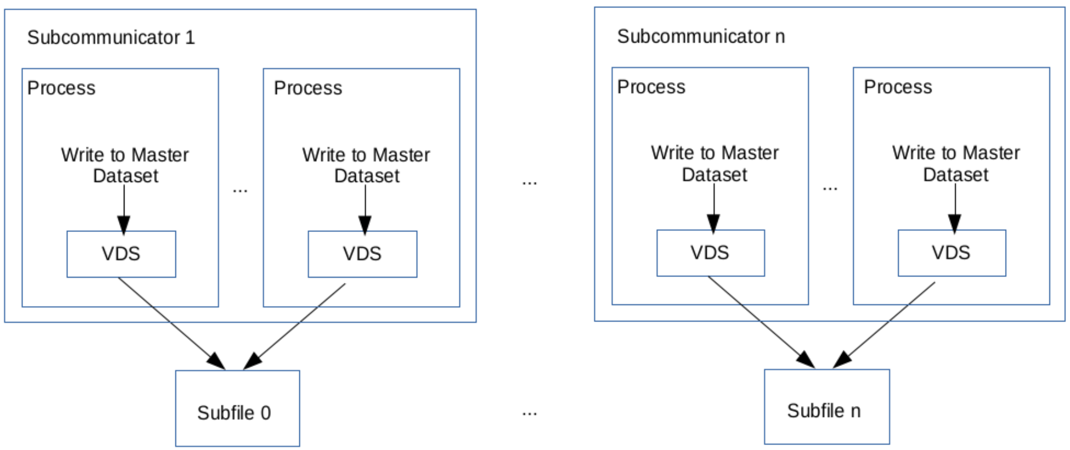


Subfiling in HDF5



- Virtual Datasets
  - Introduced in HDF5 1.10.0
  - Allows for pieces of a dataset to be stored in separate files, but would be viewed as a single dataset from a master file

- Creating subfiles
  - Split the MPI_COMM_WORLD communicator into multiple subcommunicators
  - One subfile per subcommunicator



Using HDF5 subfiling

- Split the MPI_COMM_WORLD communicator into multiple subcommunicators

```
int color = mpi_rank % subfile;
if (n_nodes > subfile)
color = (mpi_rank % n_nodes) % subfile;
MPI_Comm_split (..., &subfile_comm);
```

- Writing subfiles

```
sprintf (subfile_name, "Subfile_%d.h5", mpi_rank);
H5Pset_subfiling_access (fapl_id, subfile_name, MPI_COMM_SELF, MPI_INFO_NULL);
fid = H5Fcreate (filename, ..., fapl_id);
H5Sselect_hyperslab (sid, H5S_SELECT_SET, start, stride, count, block);
dapl_id = H5Pcreate (H5P_DATASET_ACCESS);
H5Pset_subfiling_selection(dapl_id, sid);
did = H5Dcreate (fid, DATASET, ..., sid, ..., dapl_id);
H5Dwrite (did, H5T_NATIVE_INT, mem_sid, sid, H5P_DEFAULT, wbuf);
```

Using HDF5 subfiling

- Split the MPI_COMM_WORLD communicator into multiple subcommunicators

```
int color = mpi_rank % subfile;
if (n_nodes > subfile)
color = (mpi_rank % n_nodes) % subfile;
MPI_Comm_split (..., &subfile_comm);
```

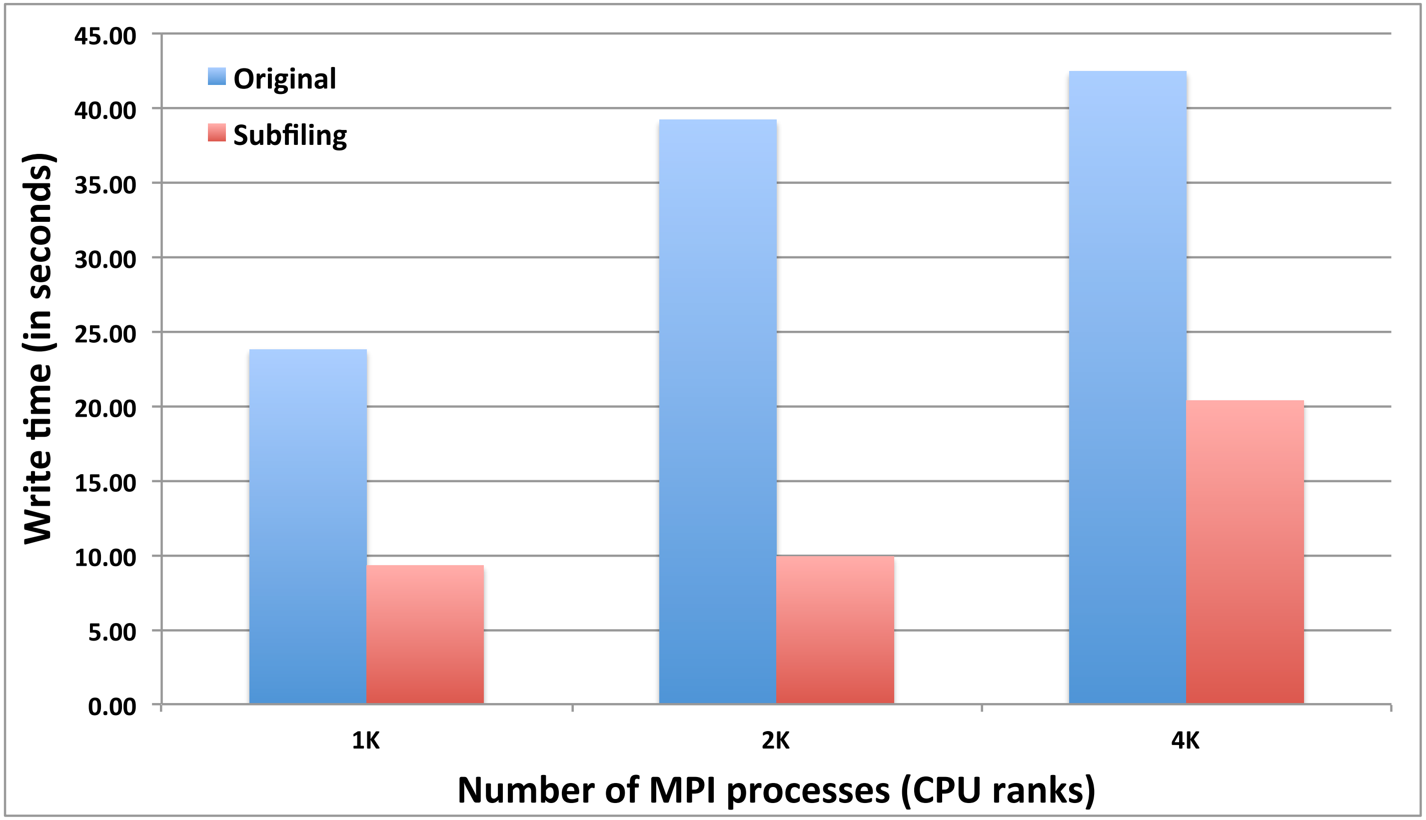
- Reading subfiles

```
sprintf (subfile_name, "Subfile_%d.h5", mpi_rank);
H5Pset_subfiling_access (fapl_id, subfile_name, MPI_COMM_SELF,
MPI_INFO_NULL);
H5Sselect_hyperslab (sid, H5S_SELECT_SET, start, stride, count, block);
H5Dread (did, H5T_NATIVE_INT, mem_sid, sid, H5P_DEFAULT, rbuf);
```

Experimental setup

- Systems
  - Cori
    - Haswell partition → 2388 nodes, 32-core Intel Xeon E5-2698 CPUs
    - File systems: cscratch (Lustre, 248 OSS, 248 OSTs), SSD-based burst buffer
  - Edison
    - 5586 compute nodes, 24-core Ivy Bridge processors
    - File system: scratch3 (Lustre, 36 OSTs), cscratch (Lustre, 248 OSS, 248 OSTs)
- Benchmarks
  - VPIC-IO
    - I/O kernel from a plasma physics simulation
    - 8 million particles per MPI process, 8 variables per particle
  - BD-CATS-IO
    - I/O kernel from Big Data Clustering application to run DBSCAN
    - Reads VPIC-IO data
- IO Measurements
  - IO time and IO rate
  - Each job was run at least three times

Scalability tests – Edison scratch3 – IO time

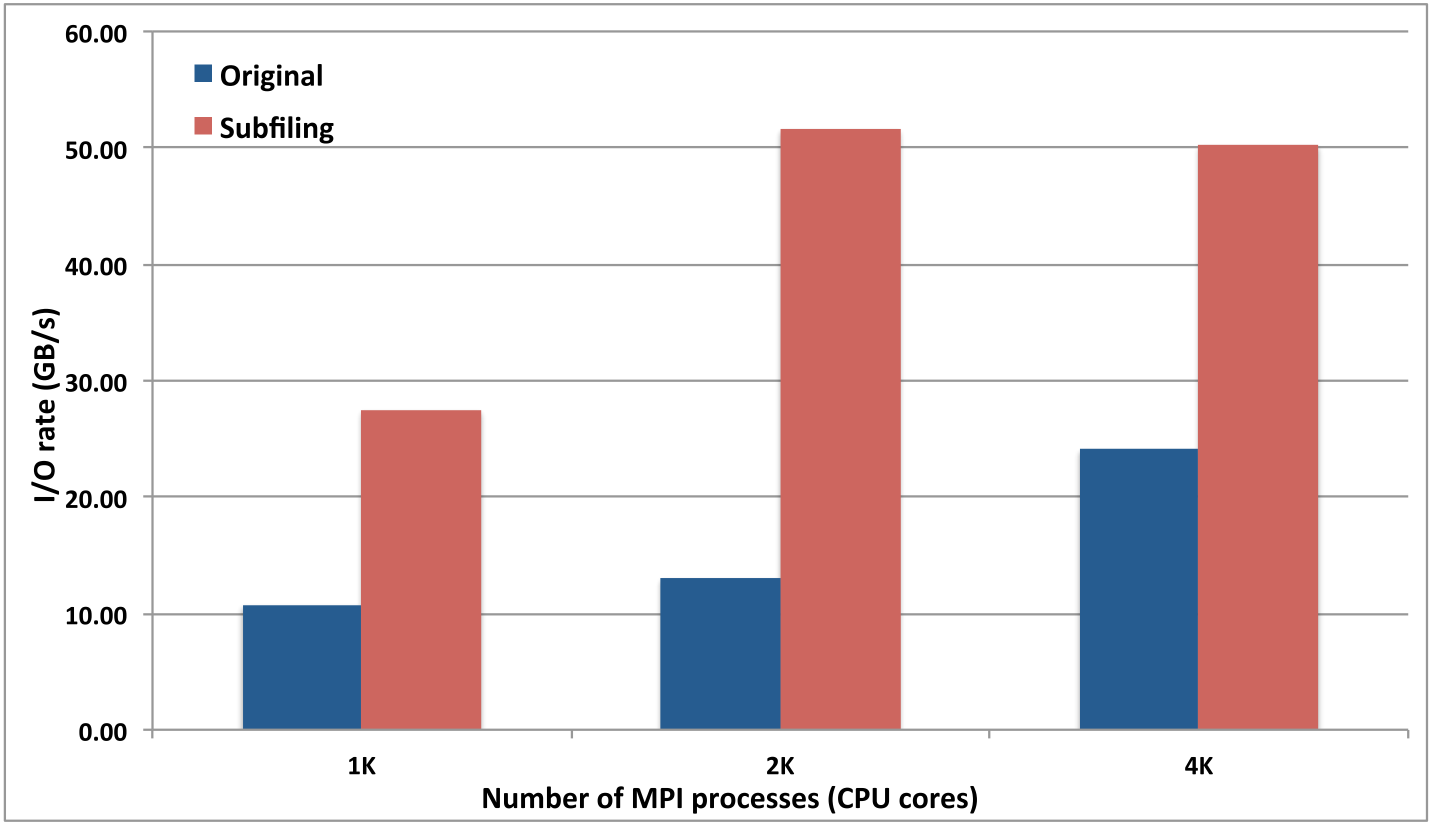


Configuration

- /scratch3 Lustre file system on Edison
- 36 OST, 32MB stripe size
- 72 GB/s peak BW
- Subfiling factor: 32
- # of subfiles:
  - 1K → 32
  - 2K → 64
  - 4K → 128

- Subfiling is 4X better at 2K and 2X better at 4K

Scalability tests – Edison scratch3 – IO rate

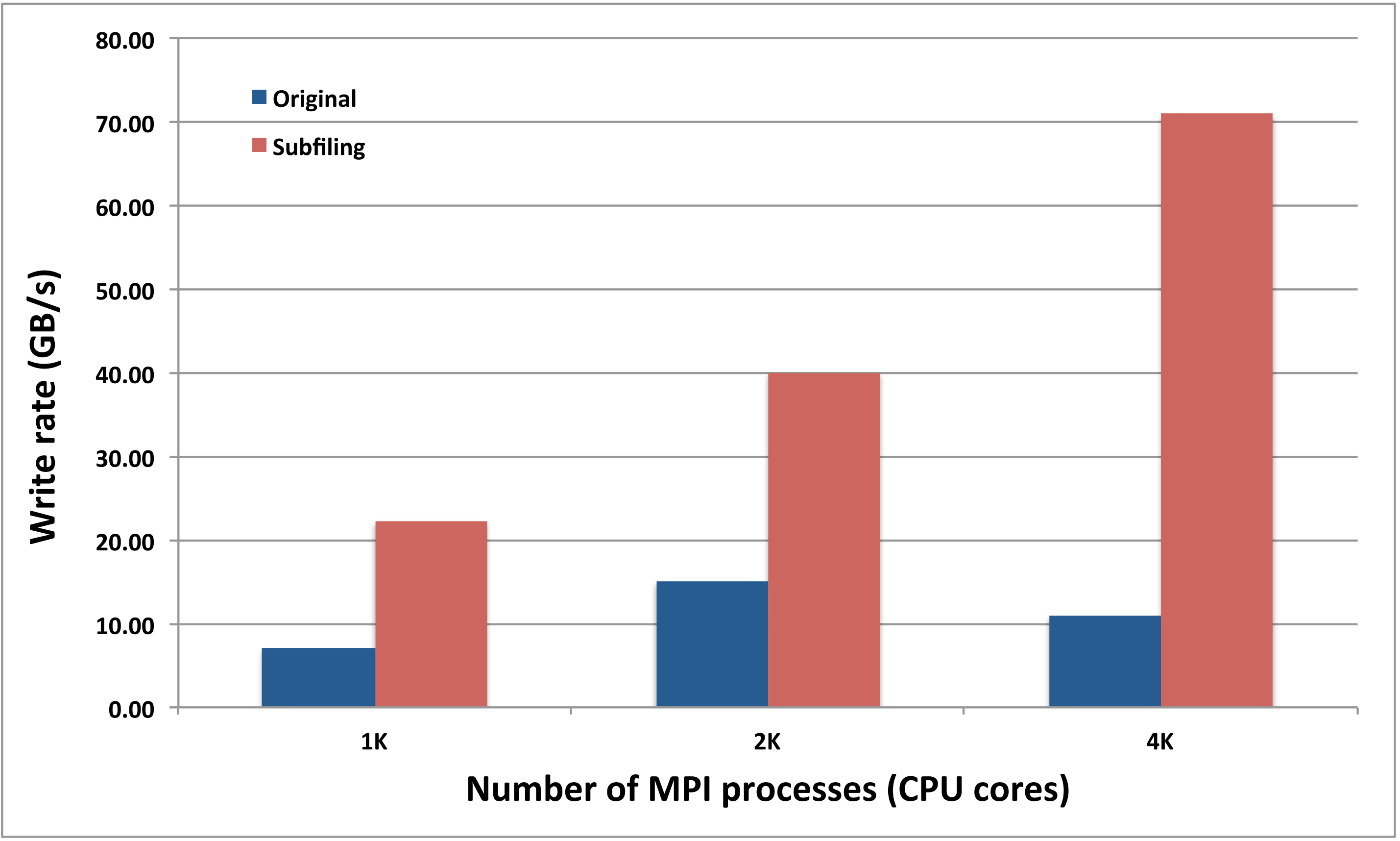


Configuration

- /scratch3 Lustre file system on Edison
- 36 OST, 32MB stripe size
- 72 GB/s peak BW
- Subfiling factor: 32
- # of subfiles:
  - 1K → 32
  - 2K → 64
  - 4K → 128

- 67% of the peak bandwidth with subfiling

Scalability tests – cscratch from Edison

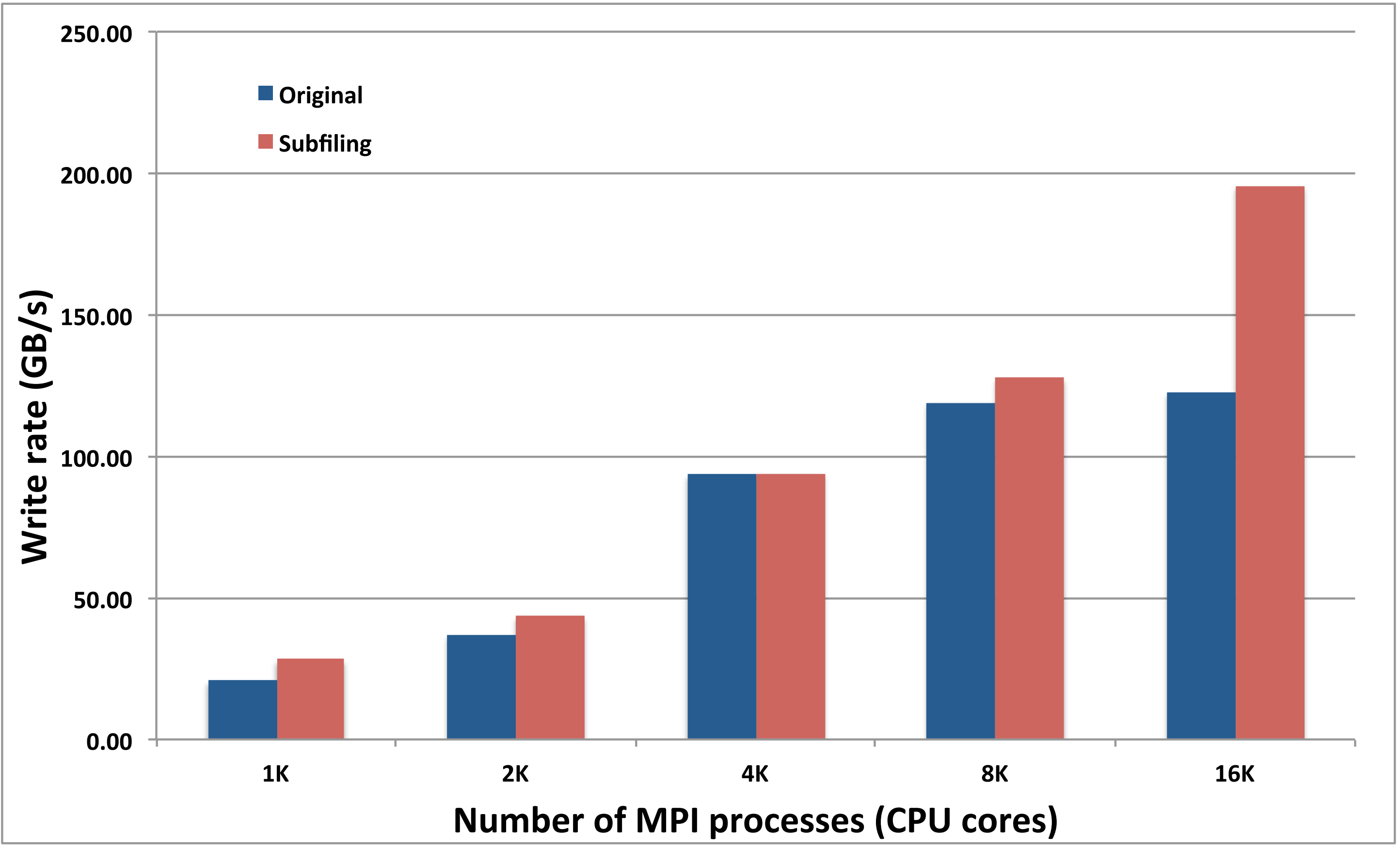


Configuration

- cscratch Lustre file system on Cori
- 128 OSTs out of 248, 32MB stripe size
- >700 GB/s peak BW
- Subfiling factor: 32
- # of subfiles:
  - 1K → 32
  - 2K → 64
  - 4K → 128

- Up to 6.5X better performance @ 4K cores

Scalability tests – cscratch from Cori

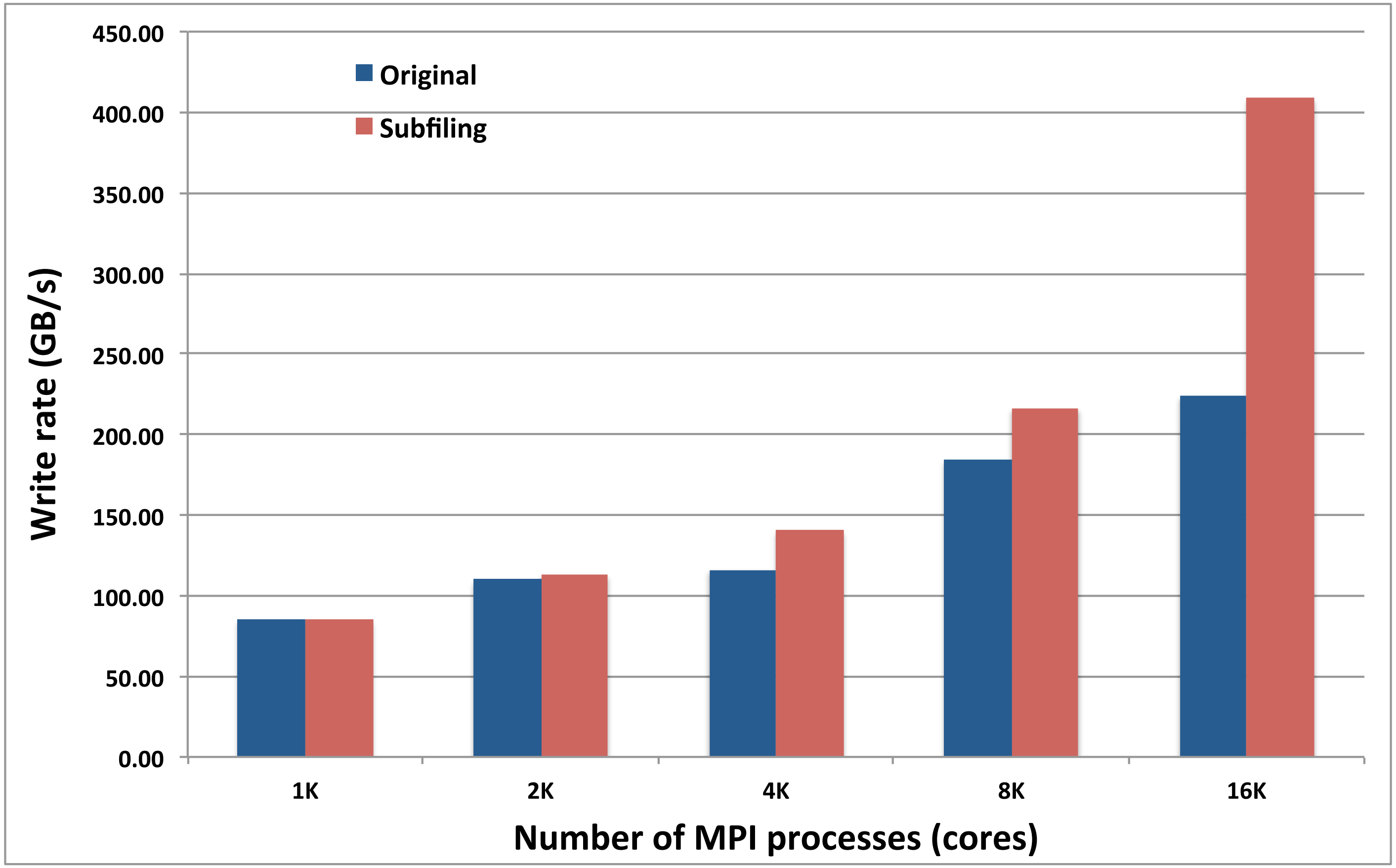


Configuration

- cscratch Lustre file system on Cori
- 128 OSTs out of 248, 32MB stripe size
- >700 GB/s peak BW
- Subfiling factor: 32
- # of subfiles:
  - 1K → 32
  - 2K → 64
  - 4K → 128
  - 8K → 256
  - 16K → 256 (subfiling factor 64)

- Up to 60% better performance @ 16K cores
- 195 GB/s IO rate at 16K processes

Scalability tests – Burst buffer from Cori

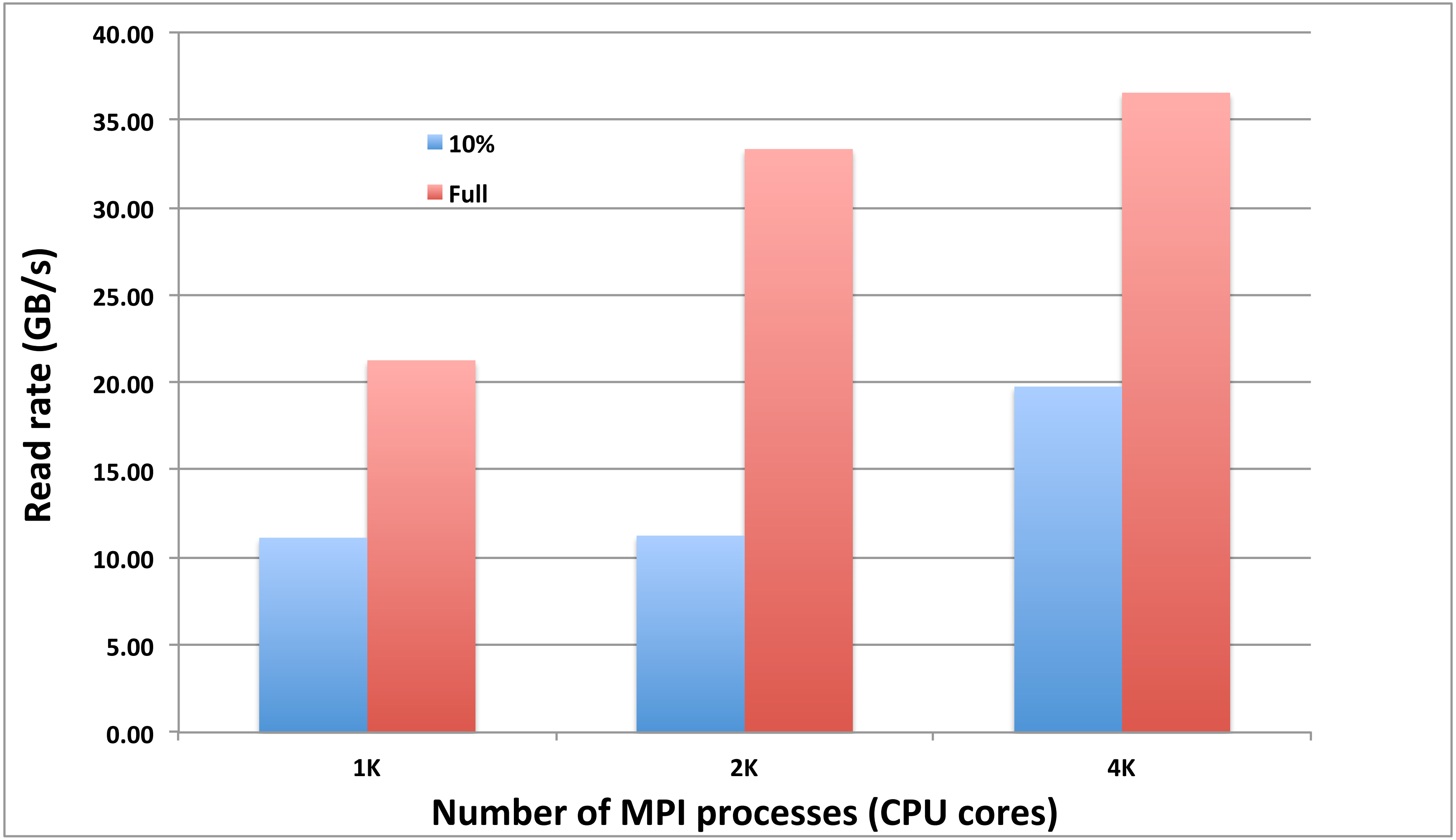


Configuration

- Burst Buffer on Cori
- 1.8 TB/s peak BW
- Subfiling factor: 32
- # of subfiles:
  - 1K → 32
  - 2K → 64
  - 4K → 128
  - 8K → 256
  - 16K → 256 (subfiling factor 64)

- Up to 80% better performance @ 16K cores
- 410 GB/s IO rate at 16K processes

Reading HDF5 subfile data – cscratch from Edison



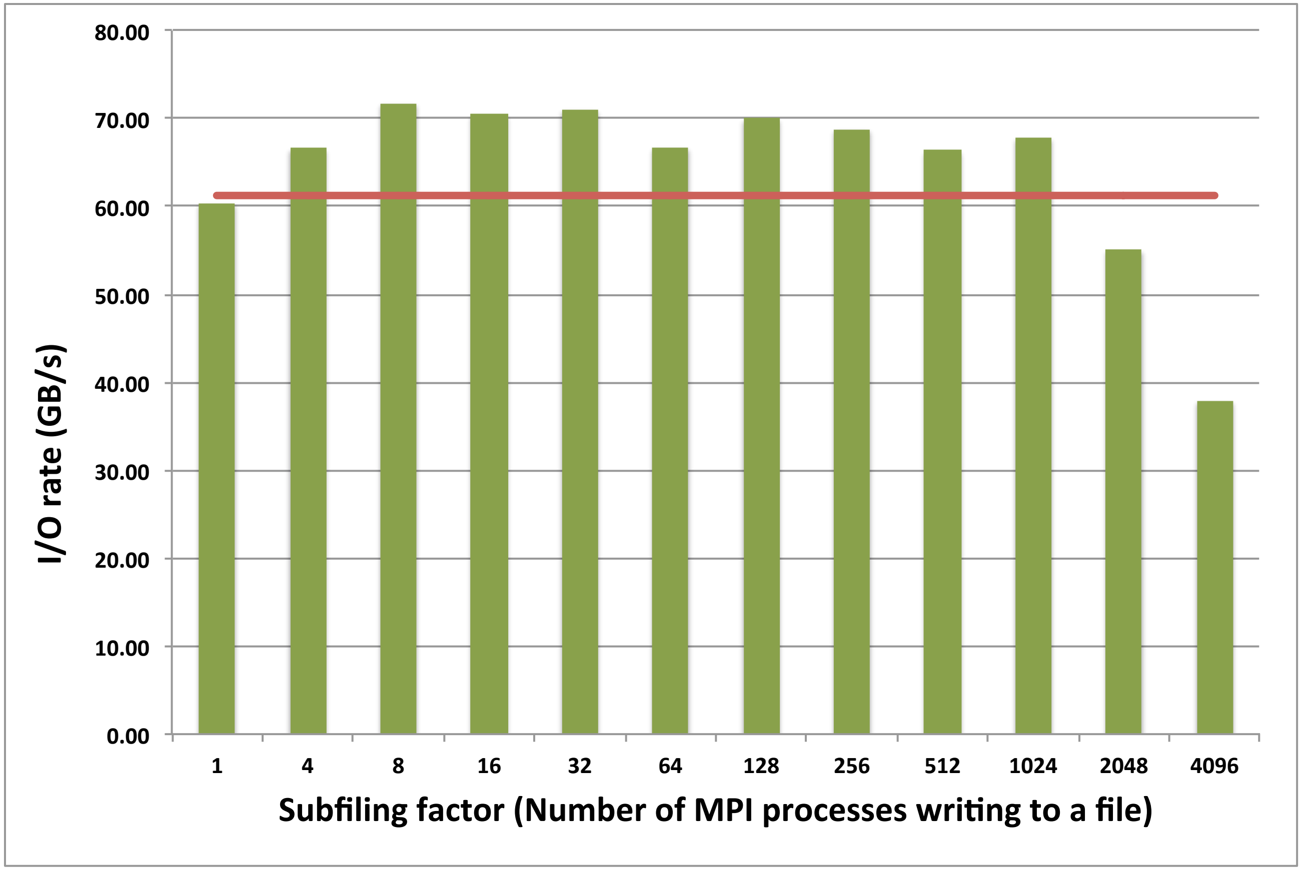
Tuning subfiling factor – Edison scratch3

Configuration

- /scratch3 Lustre file system on Edison
- 36 OSTs, 32MB stripe size
- 72 GB/s peak BW
- 4K cores
- 1TB data
- Varied the number of subfiles

- Subfiling factors of 8 to 32 resulted in good performance
- Subfiling factor of 64 resulted in poor performance consistently; but 128 to 1024 was above average

Tuning subfiling factor – cscratch from Edison

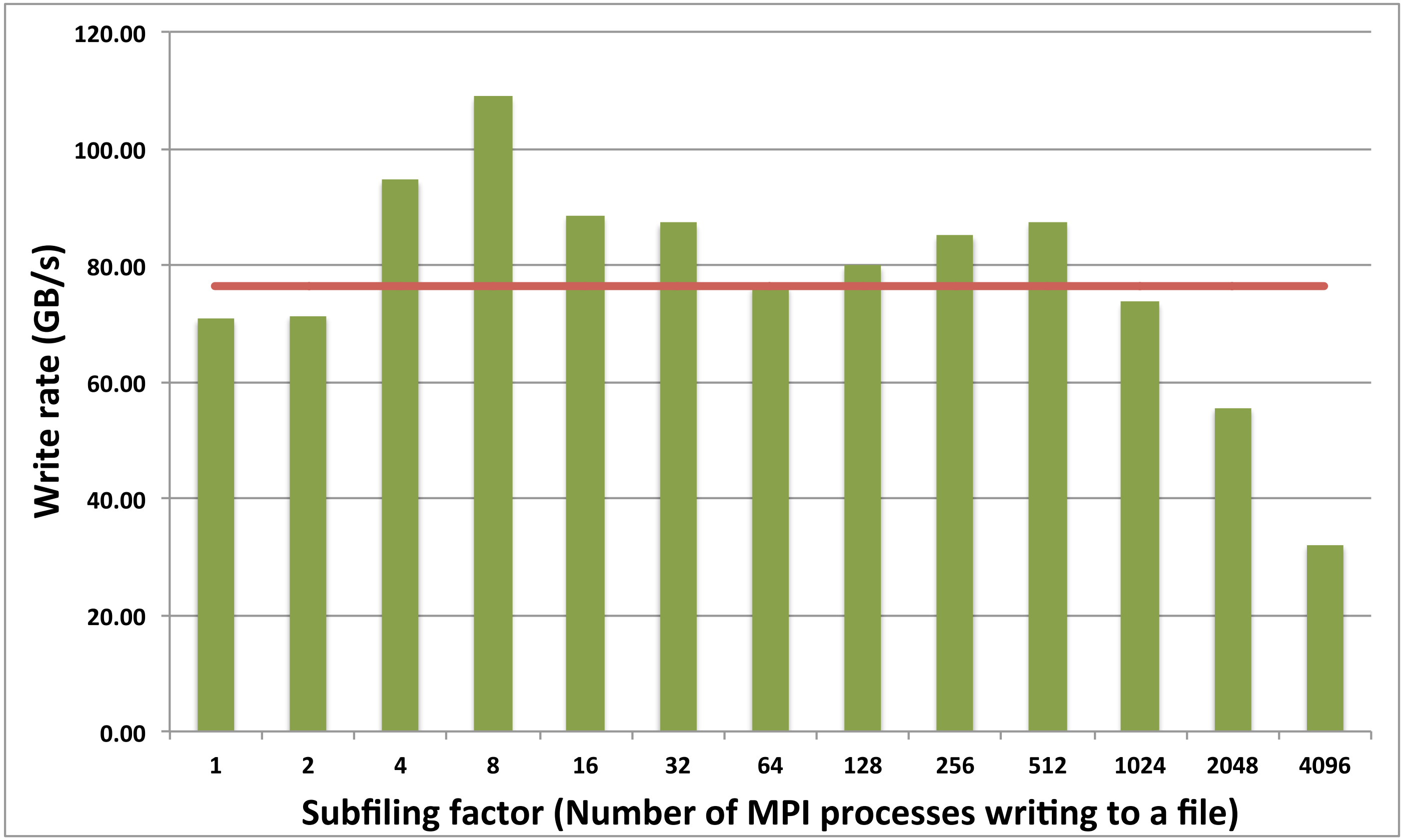


Configuration

- cscratch Lustre file system on Cori
- 128 OSTs, 32MB stripe size
- >700 GB/s peak BW
- 4K cores
- 1TB data
- Varied the number of subfiles

- Subfiling factors of 8 to 32 resulted in good performance
- Subfiling factor of 64 resulted in poor performance consistently; but 128 to 1024 was above average

Tuning subfiling factor – cscratch from Cori

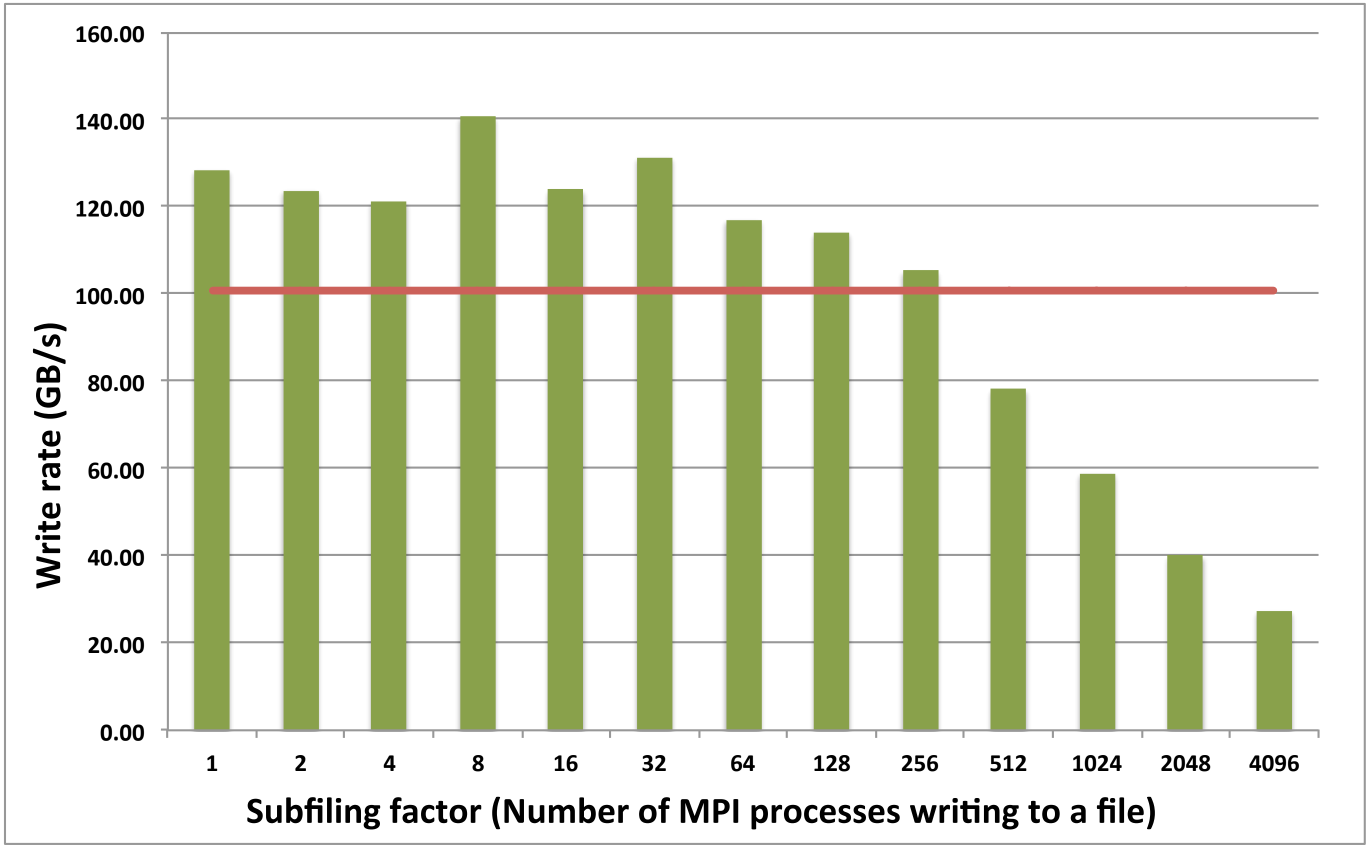


Configuration

- cscratch Lustre file system on Cori
- 128 OSTs, 32MB stripe size
- >700 GB/s peak BW
- 4K cores
- 1TB data
- Varied the number of subfiles

- Subfiling factors of 4 and 8 resulted in good performance
- Subfiling factors between 16 to 512 showed above average I/O rates

Tuning subfiling factor – Burst buffer on Cori

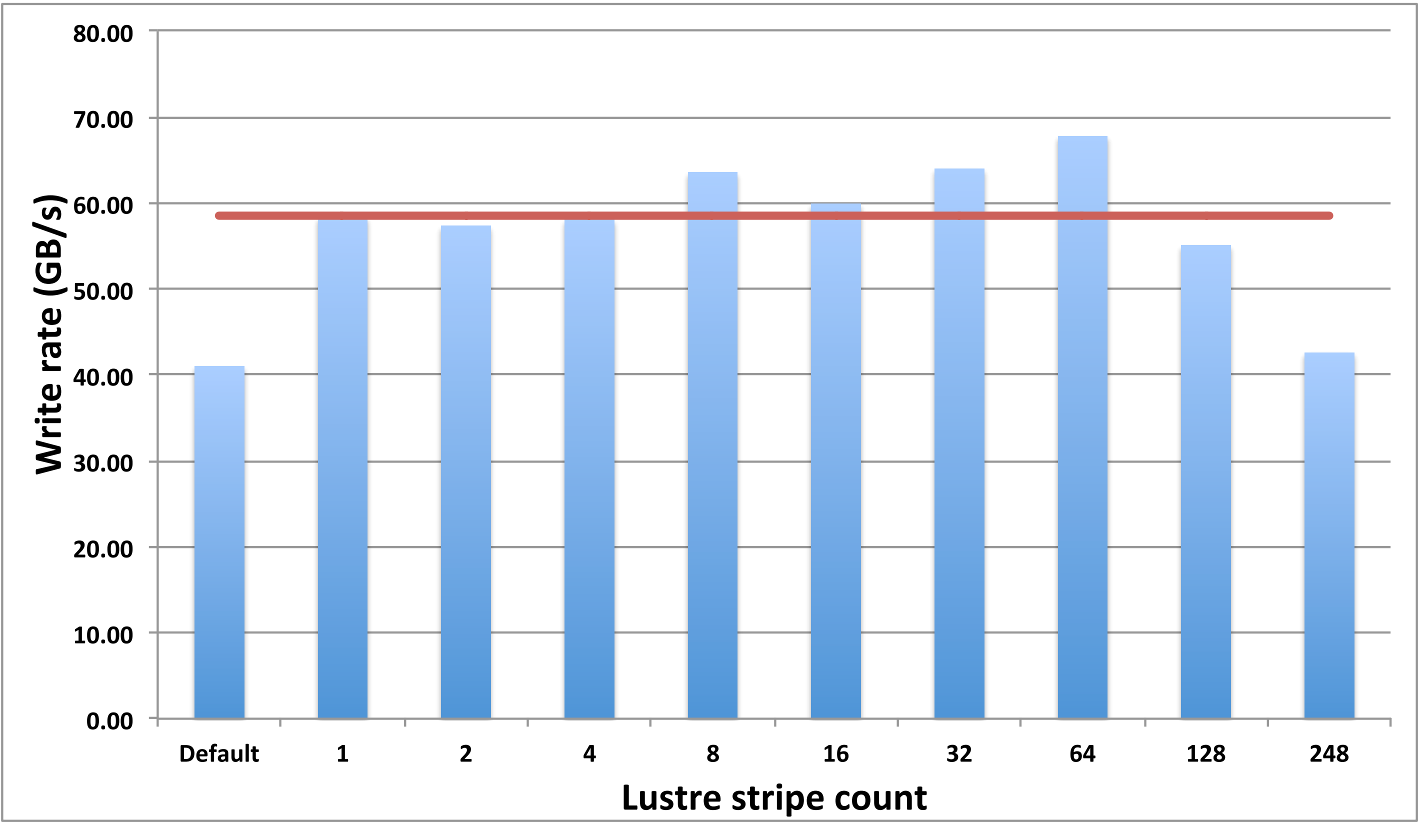


Configuration

- Burst buffer on Cori
- 1.8 TB/s peak BW
- 4K cores
- 1TB data
- Varied the number of subfiles

- Subfiling factors of 1 to 32 resulted in good performance
- Performance degraded with subfiling factors beyond 32 and beyond

Tuning Lustre striping – cscratch from Edison

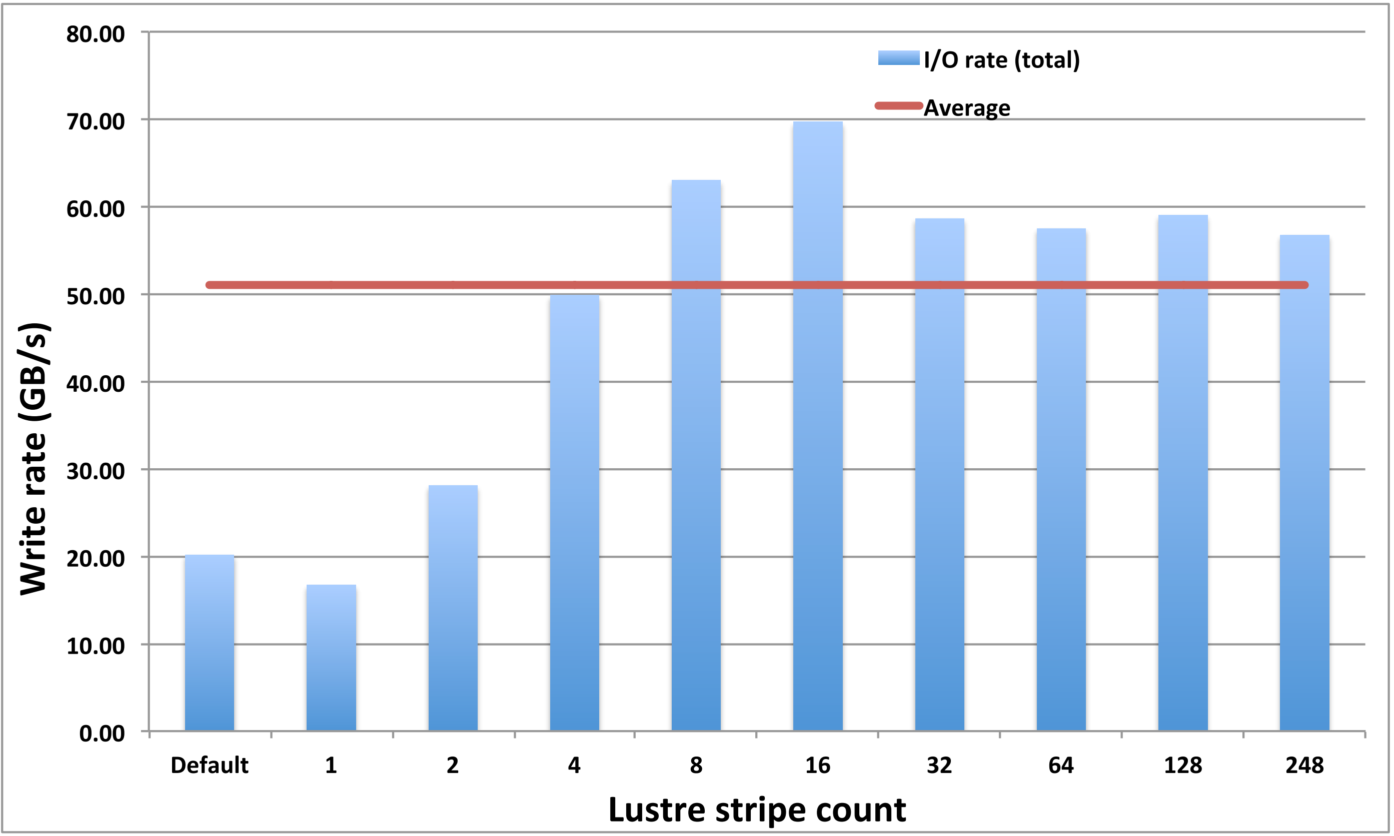


Configuration

- cscratch Lustre file system on Cori
- >700 GB/s peak BW
- 4K cores
- 1TB data
- Subfiling factor of 128
- Varied the number of OSTs
- Stripe size: 32MB
- Default: 1 OST, 1 MB stripe size

- Using 8 to 64 stripes resulted in more than average performance
- 70% better performance than default stripe settings

Tuning Lustre striping – cscratch from Cori – 4K

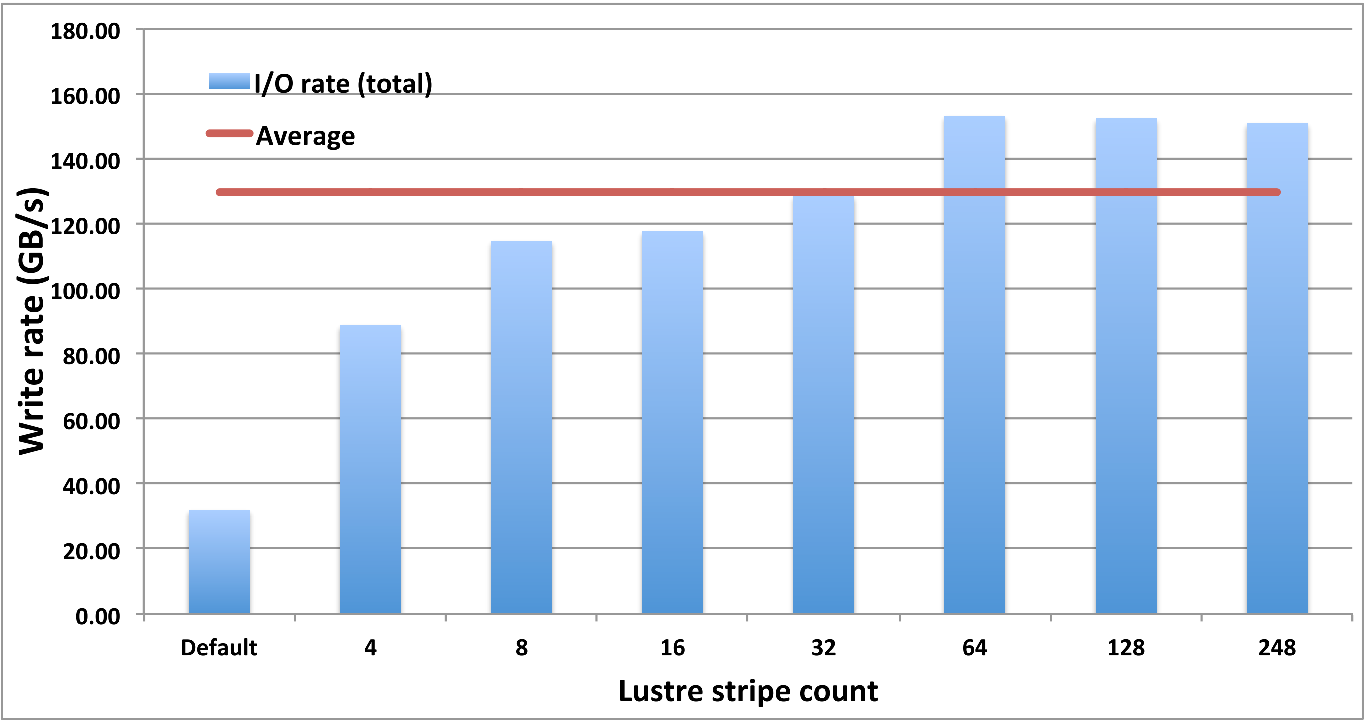


Configuration

- cscratch Lustre file system on Cori
- >700 GB/s peak BW
- 4K cores
- 1TB data
- Subfiling factor of 128
- Varied the number of OSTs
- Stripe size: 32MB
- Default: 1 OST, 1 MB stripe size

- Using 8 to 248 stripes resulted in good I/O performance
- 3.5X faster performance than default stripe settings @ 16 OSTs

Tuning Lustre striping – cscratch from Cori – 16K



Configuration

- cscratch Lustre file system on Cori
- >700 GB/s peak BW
- 16K cores
- 1TB data
- Subfiling factor of 64
- Varied the number of OSTs
- Stripe size: 32MB
- Default: 1 OST, 1 MB stripe size

- Using 64 stripes resulted in the best performance
- 4.8X faster performance than default stripe settings @ 64 OSTs

Conclusions

- Recommendations for obtaining good I/O rate
  - Subfiling factor of 8 to 64 is reasonable
  - Striping 16 at smaller scales, and 64 at larger scales
- Limitations
  - Using subfiling at 32K MPI processes failed
  - Failure observed for region sizes > 2GB (probably an MPI limitation)
  - Number of readers have to be equal to the number of writers
- Subfiling is showing better performance than writing to a single shared file
  - Up to 6.5X performance advantage
- Reading with an arbitrary number of MPI processes, without matching the number of readers will be useful