CSE 5449: Intermediate Studies in Scientific Data Management

Lecture 19: Caching and prefetching in HDF5

Dr. Suren Byna

The Ohio State University

E-mail: byna.1@osu.edu

https://sbyna.github.io

03/28/2023

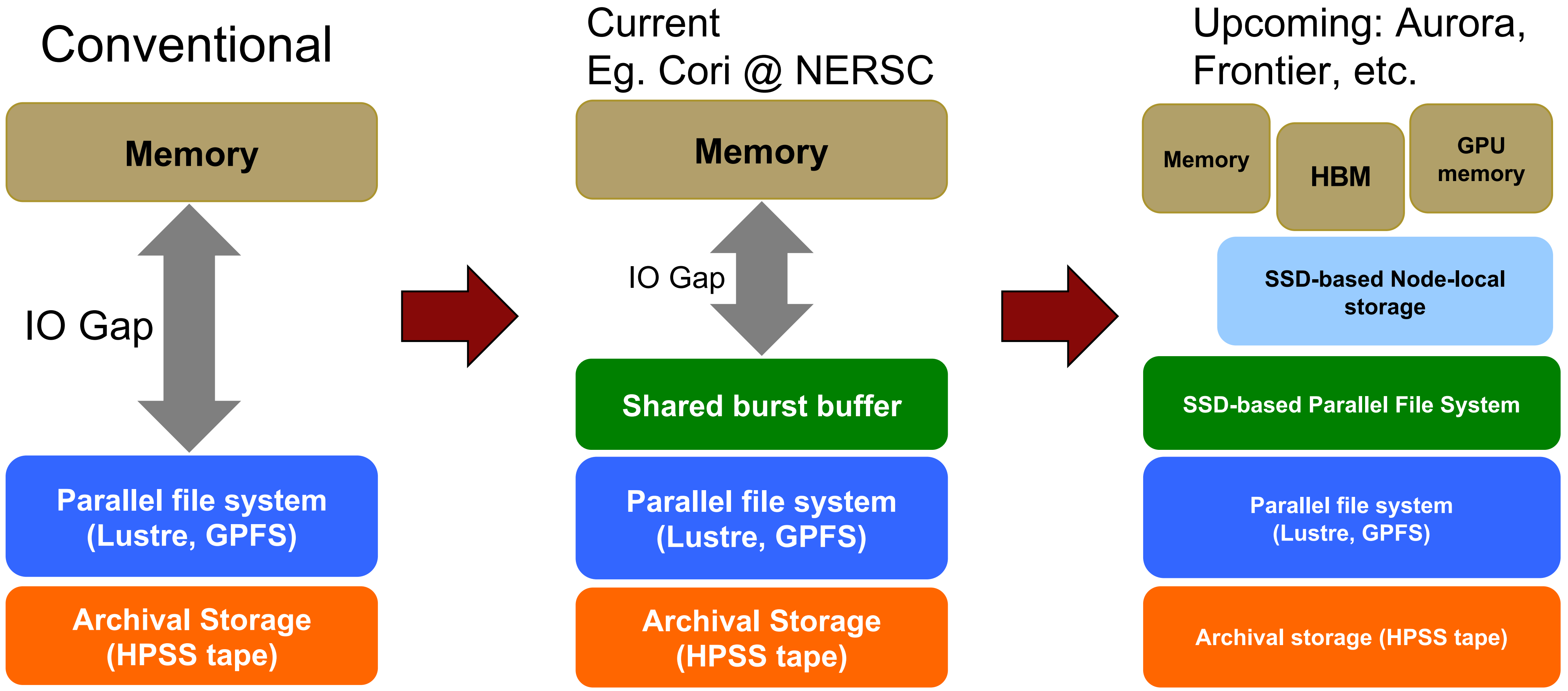
Today’s class

- Any questions?

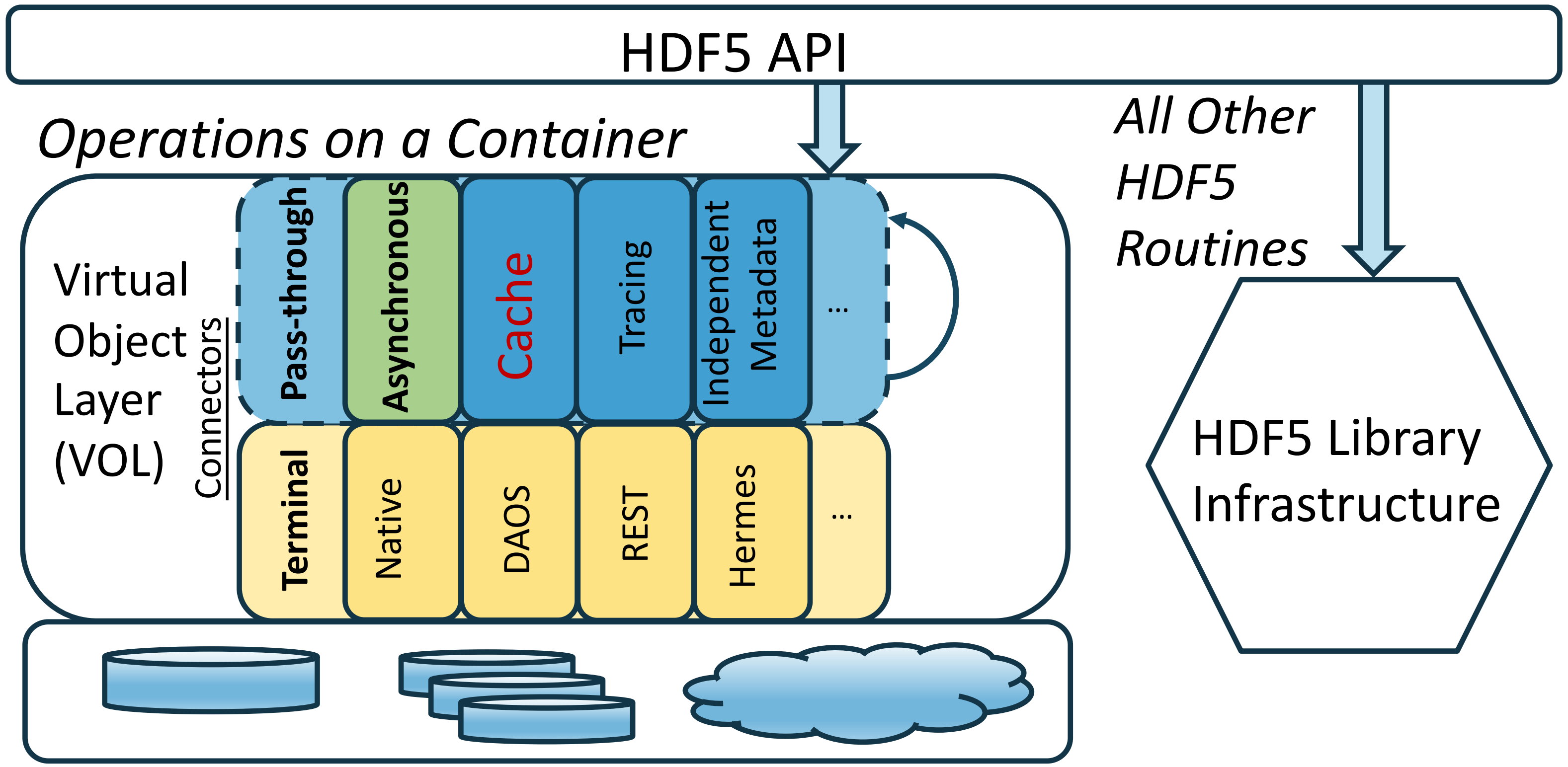
- Class presentation topic

- Today’s class –
  - HDF5 optimizations – Caching and prefetching

Storage systems in high performance compute systems


Conventional
Memory
IO Gap
Parallel file system (Lustre, GPFS)
Archival Storage (HPSS tape)
Current
Eg. Cori @ NERSC
Memory
IO Gap
Shared burst buffer
Parallel file system (Lustre, GPFS)
Archival Storage (HPSS tape)
Upcoming: Aurora, Frontier, etc.
Memory
HBM
GPU memory
SSD-based Node-local storage
SSD-based Parallel File System
Parallel file system (Lustre, GPFS)
Archival storage (HPSS tape)

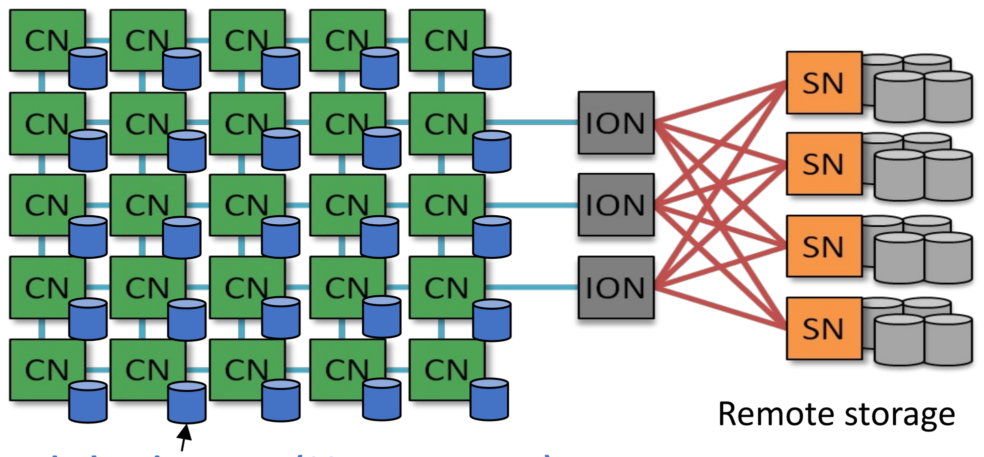
HDF5 Virtual Object Layer (VOL)



Huihuo Zheng, Venkatram Vishwanath, Quincey Koziol, Houjun Tang, John Ravi, John Mainzer and Suren Byna, "HDF5 Cache VOL: Efficient and Scalable Parallel I/O through Caching Data on Node-local Storage", The 22nd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGrid) 2022.

Transparently integrating node-local storage into parallel I/O workflows

Typical HPC storage hierarchy



Node-local storage (SSD, NVMe, etc)

Polaris @ ALCF: NVMe (7.68 TB / node)
Summit @ OLCF: GPFS + NVMe (1.6 TB / node)
Fugaku @ RIKEN: Lustre + NVMe (1.6 TB / 16 nodes)
Frontier @ OLCF: Lustre + NVMe (37PB total)

Node-local storage

- Local & private; no contention or job interference
  → more stable and scalable IO;
- Faster (larger aggregate bandwidth).
  Theta (w) – Lustre: 650 GB/s, SSD: 3TB/s
  Summit (w) – GPFS: 2.5 TB/s, NVMe: 9.7 TB/s

Challenges

- No global namespace;
- Accessible only during job running;
- Limited system software support.

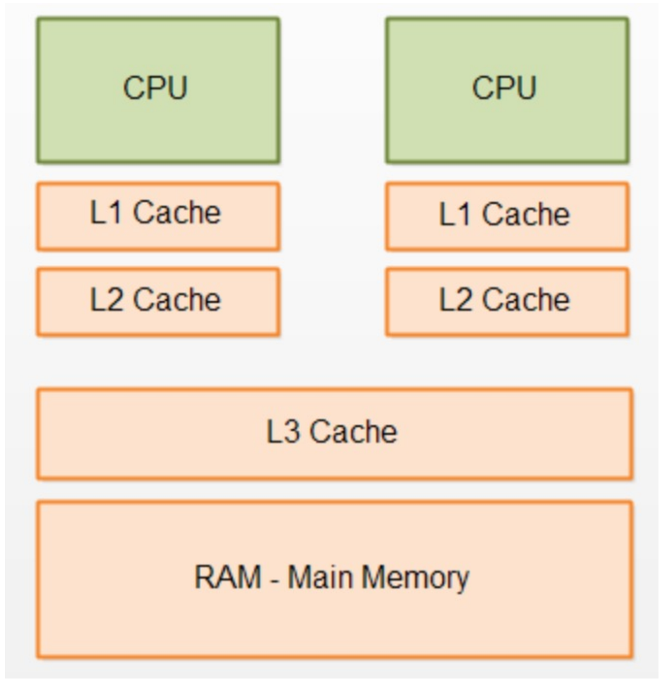
Cache VOL: using node-local storage as a cache

https://github.com/hpc-io/vol-cache.git

Slide from Huihuo Zheng, CCGrid 2022 presentation

Using caching to improve data access

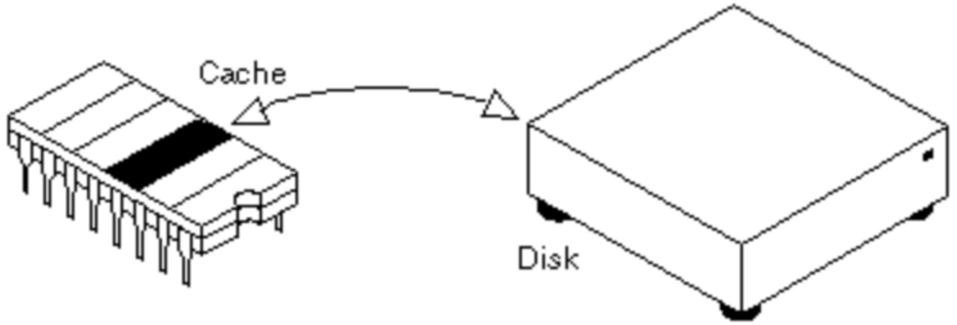
Caching in memory hierarchy



~TB/s

~100GB/s

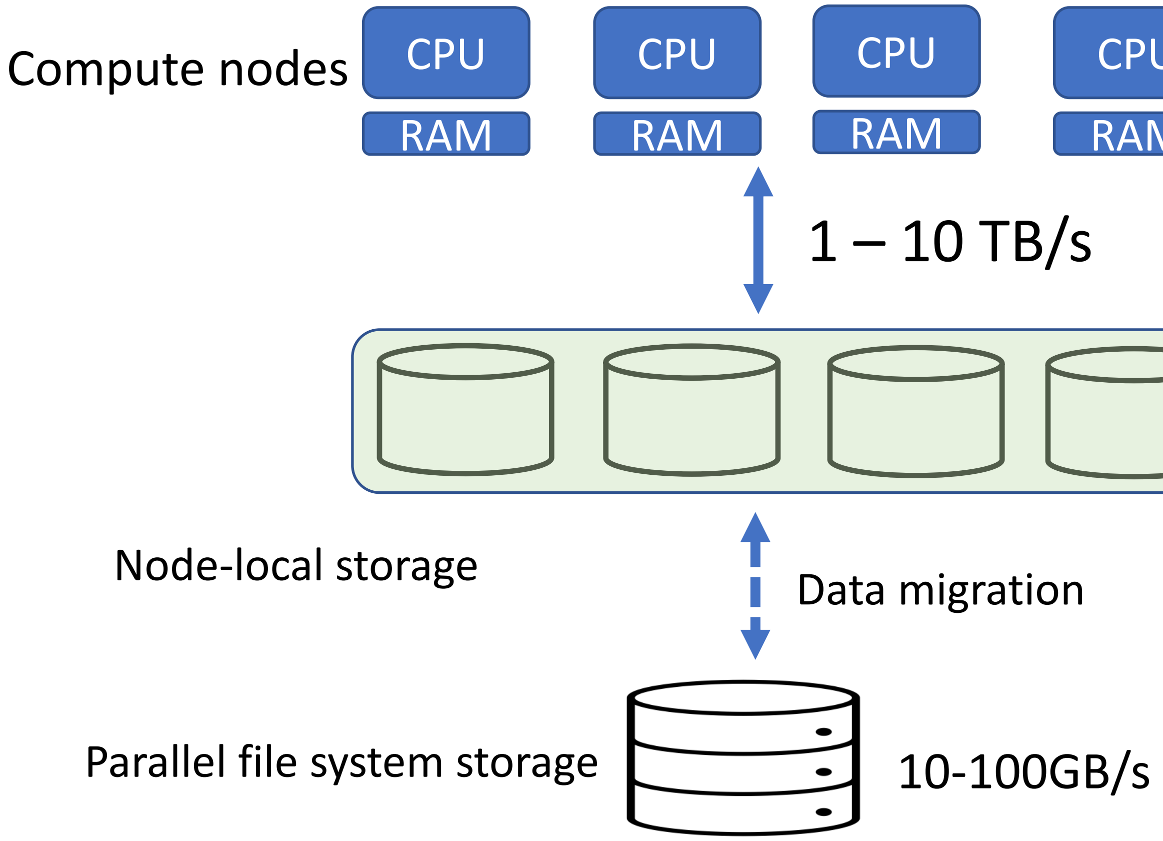
Page caching in I/O



- **Write:** the data is copied from the user's buffer into the page cache in DRAM. The actual writes to disk are done later.
- **Read:** data is read directly from the page cache in DRAM if it is cached there.

Slide from Huihuo Zheng, CCGrid 2022 presentation

Using node-local storage as a cache to the global parallel file systems

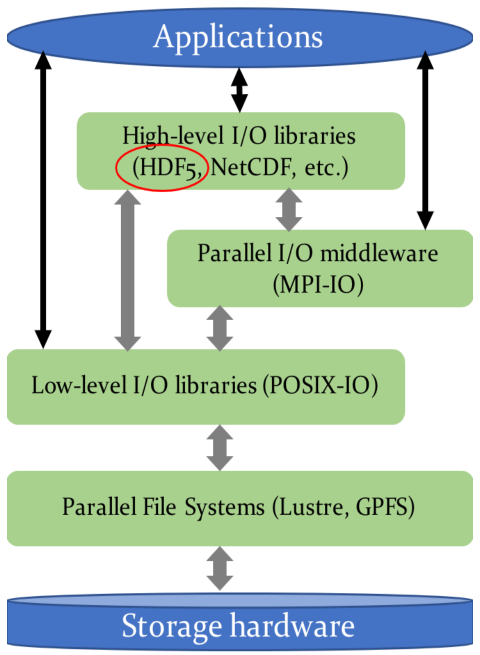


Design consideration

- Managing node-local storage inside the I/O libraries
- Caching and async. data migration in the background;
- Hiding all complexity;
- Easy integration to existing applications with minimal code change.

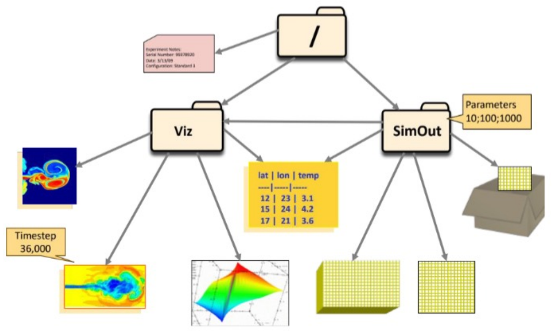
Slide from Huihuo Zheng, CCGrid 2022 presentation

Implementation as a HDF5 pass-through Virtual Object Layer connector



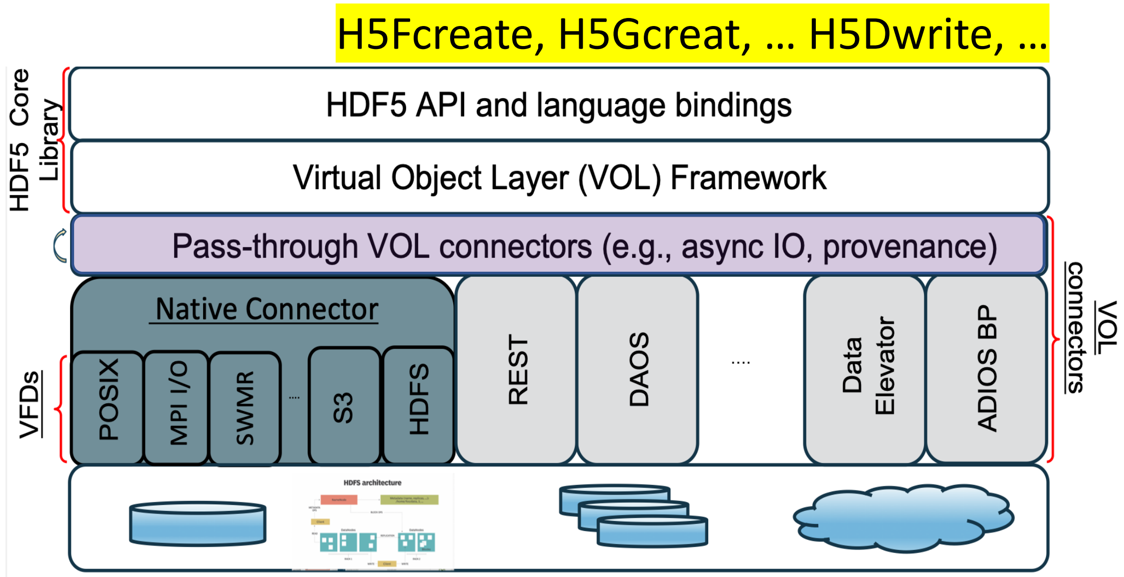
Applications
High-level I/O libraries (HDF5, NetCDF, etc.)
Parallel I/O middleware (MPI-IO)
Low-level I/O libraries (POSIX-IO)
Parallel File Systems (Lustre, GPFS)
Storage hardware

Parallel I/O software layers



HDF5 is a high-level I/O library used for storing scientific data in a hierarchical database-like way.

Virtual Object Layer (VOL) Framework:
an abstraction layer within HDF5 Library.



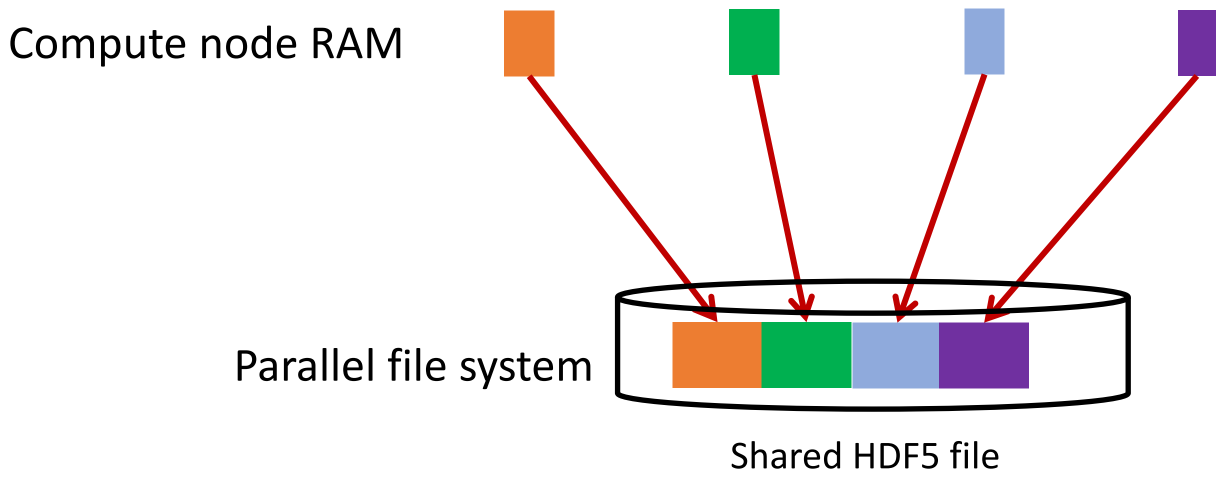
H5Fcreate, H5Gcreat, … H5Dwrite, …
HDF5 Core Library
HDF5 API and language bindings
Virtual Object Layer (VOL) Framework
Pass-through VOL connectors (e.g., async IO, provenance)
Native Connector
VFDs
POSIX
MPI I/O
SWMR
S3
HDFS
REST
DAOS
Data Elevator
ADIOS BP
VOL connectors

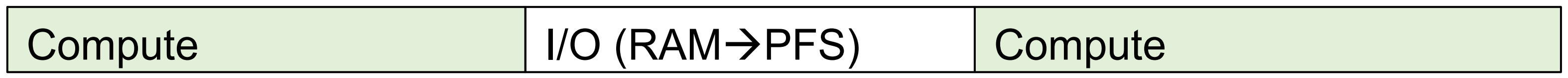
Pass-through VOL – perform operations (e.g., caching) before passing the data on to the next connector.

https://www.hdfgroup.org/wp-content/uploads/2020/10/Virtual-Object-Layer-VOL-Intro.pdf

Slide from Huihuo Zheng, CCGrid 2022 presentation

Parallel Write (H5Dwrite)

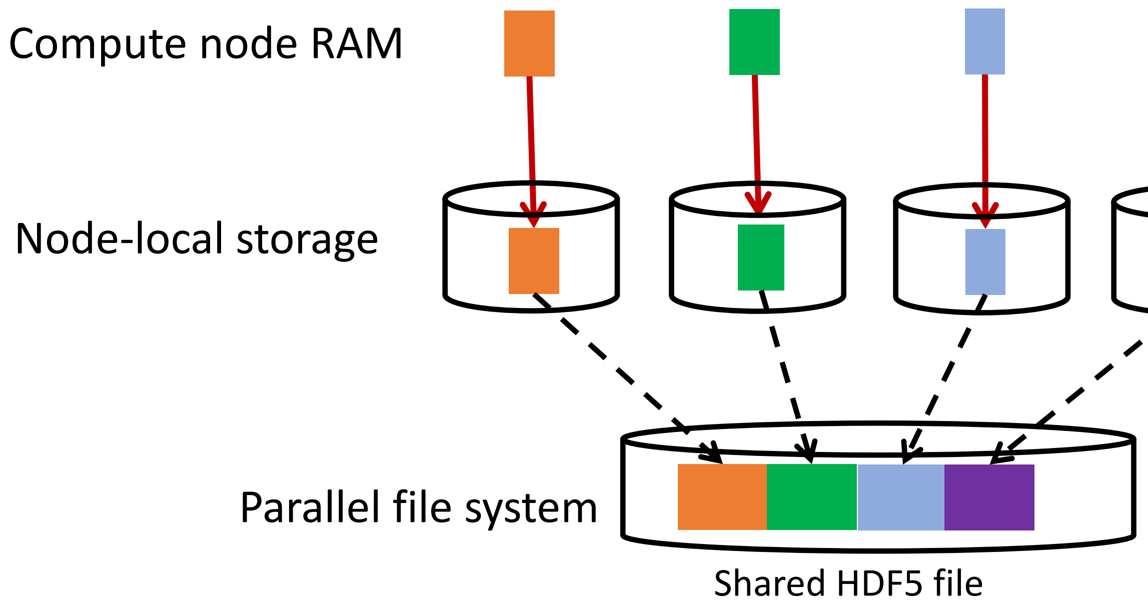




https://github.com/hpc-io/vol-cache.git

Slide from Huihuo Zheng, CCGrid 2022 presentation

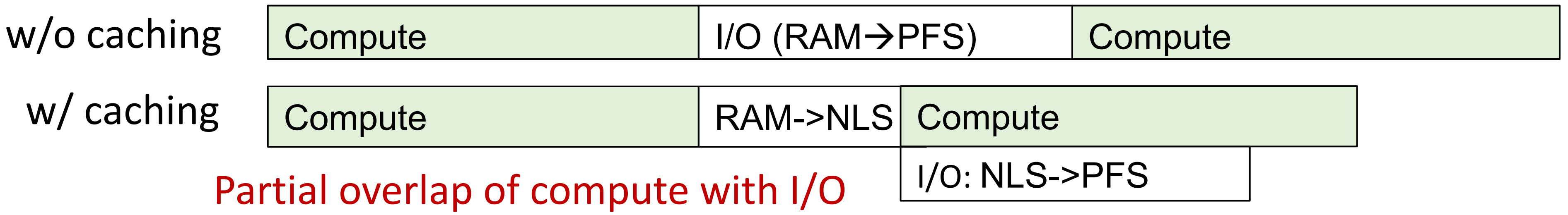
Parallel Write (H5Dwrite)



1. Data is synchronously copied from the memory buffer to memory mapped files on the node-local storage using POSIX I/O.

2. Move data from memory mapped file to the parallel file system asynchronously through *Async VOL* stacked below the Cache VOL

3. Wait for all the tasks to finish in H5Dclose() / H5Fclose()



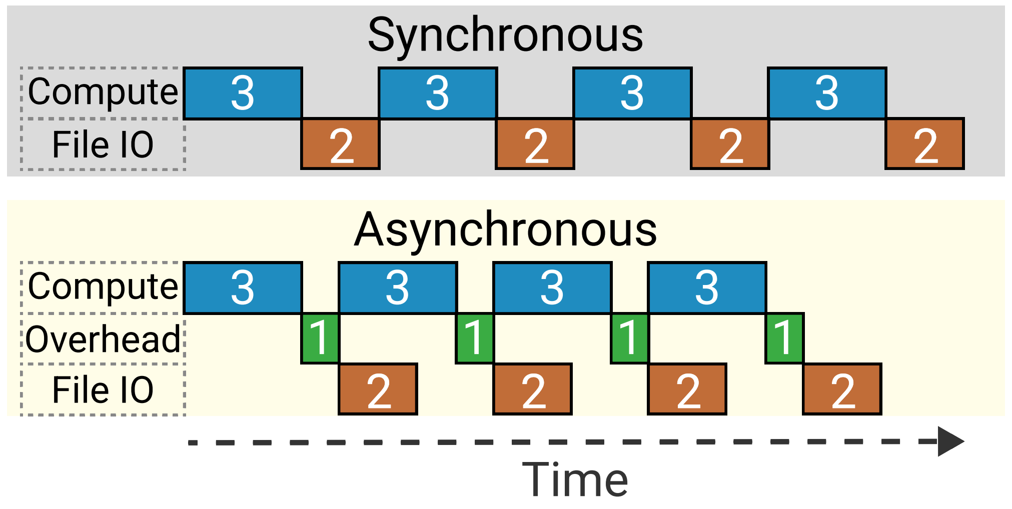
Partial overlap of compute with I/O

Details are hidden from the application developers.

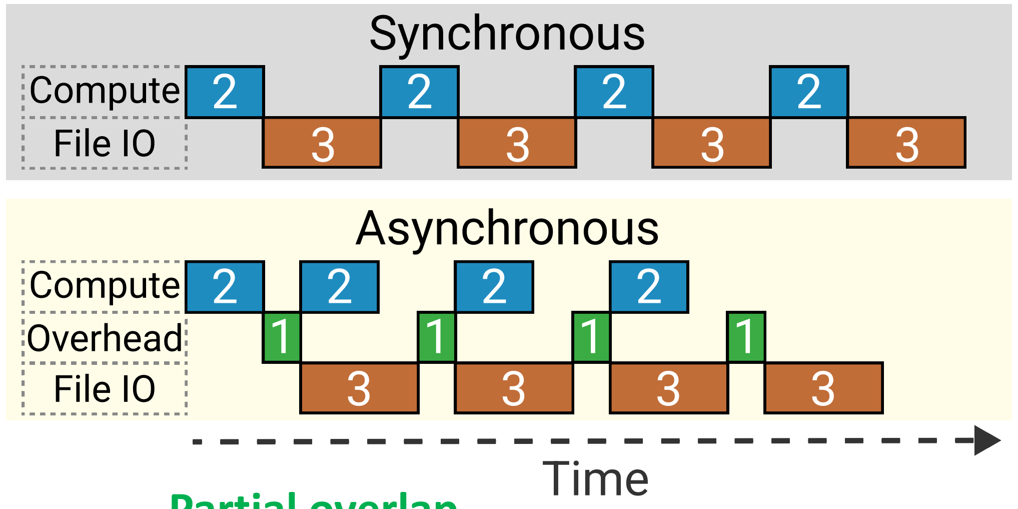
https://github.com/hpc-io/vol-cache.git

Slide from Huihuo Zheng, CCGrid 2022 presentation

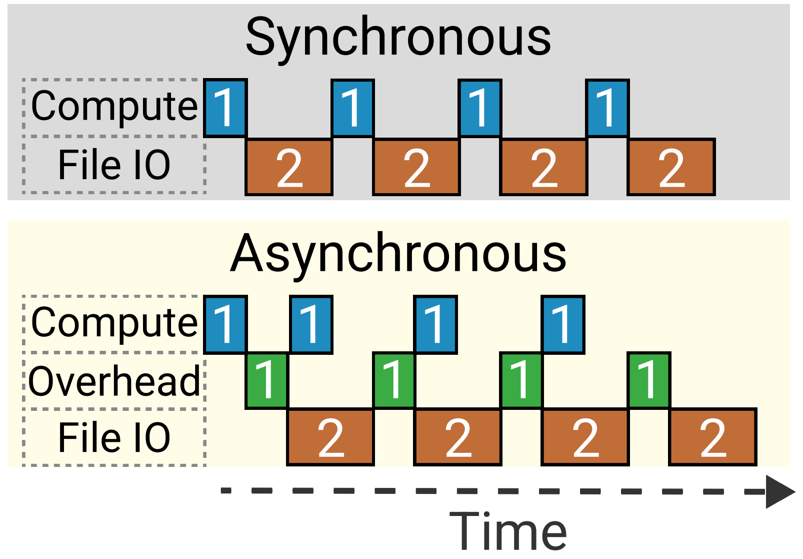
Asynchronous I/O scenarios



Computation is longer than I/O time



Partial overlap



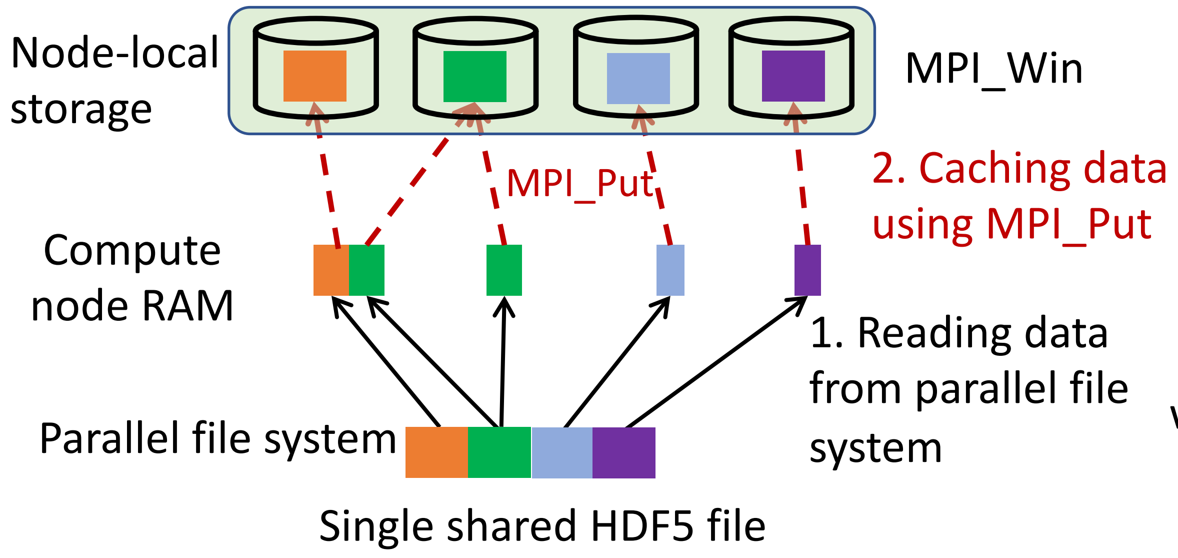
Slow down

John Ravi, Suren Byna, Quincey Koziol, Houjun Tang, and Michela Becchi, "Evaluating Asynchronous Parallel I/O on HPC Systems", 37th IEEE International Parallel and Distributed Processing Symposium (IPDPS) 2023.

Parallel Read (H5Dread)

Targeting workloads with repeatedly reading the same dataset multiple times.

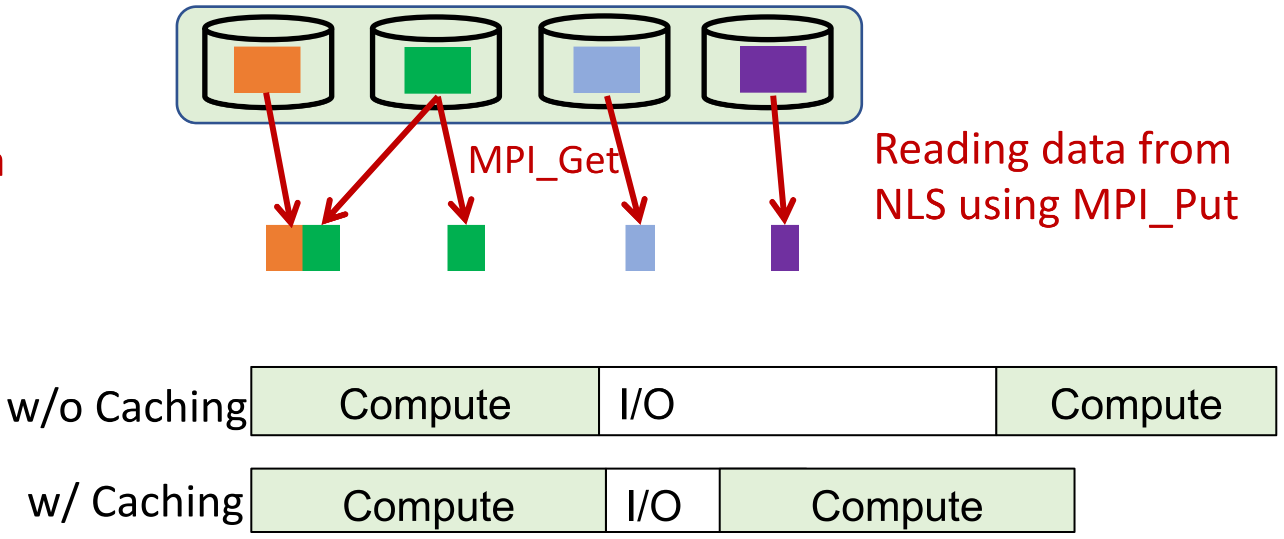
Create memory mapped files and attached virtual memory pointer to an MPI window



First time reading the data

Memory-mapped shared file system

- Each process exposes a portion of its storage to other processes through MPI Window
- Other processes read from or write to this shared storage space through MPI_Put, MPI_Get.



Reading the data directly from node-local storage

https://github.com/hpc-io/vol-cache.git

Slide from Huihuo Zheng, CCGrid 2022 presentation

Easy to adopt in the applications

1) Setting VOL connectors

```
export HDF5_PLUGIN_PATH=$HDF5_VOL_DIR/lib
export HDF5_VOL_CONNECTOR="cache_ext
config=SSD.cfg;under_vol=518;under_info={under_vol=0;under_info={}}"
export LD_LIBRARY_PATH=$HDF5_PLUGIN_PATH:$LD_LIBRARY_PATH
```

```
#contents of SSD.cfg
HDF5_CACHE_STORAGE_SIZE                 137438953472
HDF5_CACHE_STORAGE_TYPE                  SSD
HDF5_CACHE_STORAGE_PATH                 /local/scratch/
HDF5_CACHE_STORAGE_SCOPE                LOCAL
HDF5_CACHE_WRITE_BUFFER_SIZE            102457690
HDF5_CACHE_REPLACEMENT_POLICY            LRU
```

2) Enabling cache VOL

Opt. 1 Through global environment variables (**HDF5_CACHE_RD / HDF5_CACHE_WR [yes|no]**)

Opt. 2 Through setting file access property: `H5Pset_fapl_plist('HDF5_CACHE_RD', true)`

3) Initializing MPI with MPI_Init_thread(…, MPI_THREAD_MULTIPLE…)

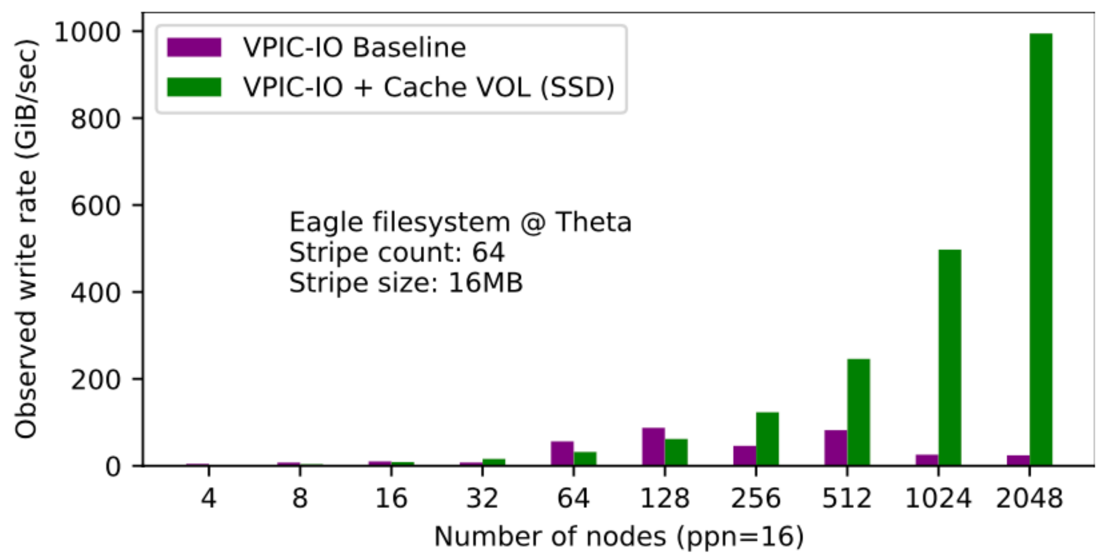
4) In some cases, rearranging the function calls to allow the overlap of computation with data migration (check our github repo for the examples and best practices**)**

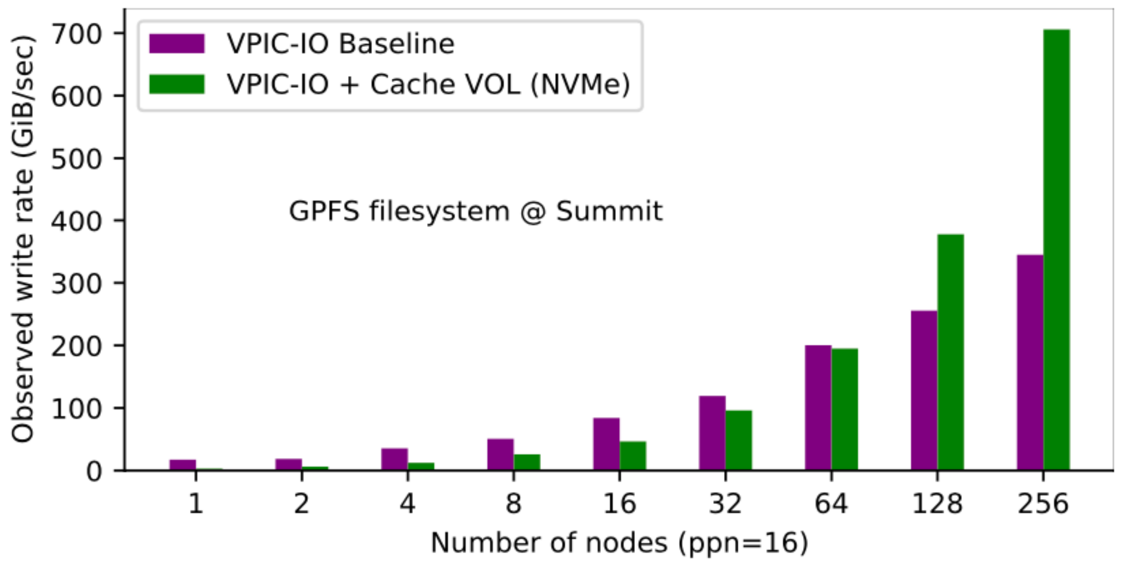
https://github.com/hpc-io/vol-cache.git

Slide from Huihuo Zheng, CCGrid 2022 presentation

Write performance (VPIC-IO)

VPIC: plasma physics application for simulating the dynamics of plasma particle.
I/O pattern: each process writes check-points data (32MB x 8) to a shared file at each timestep.





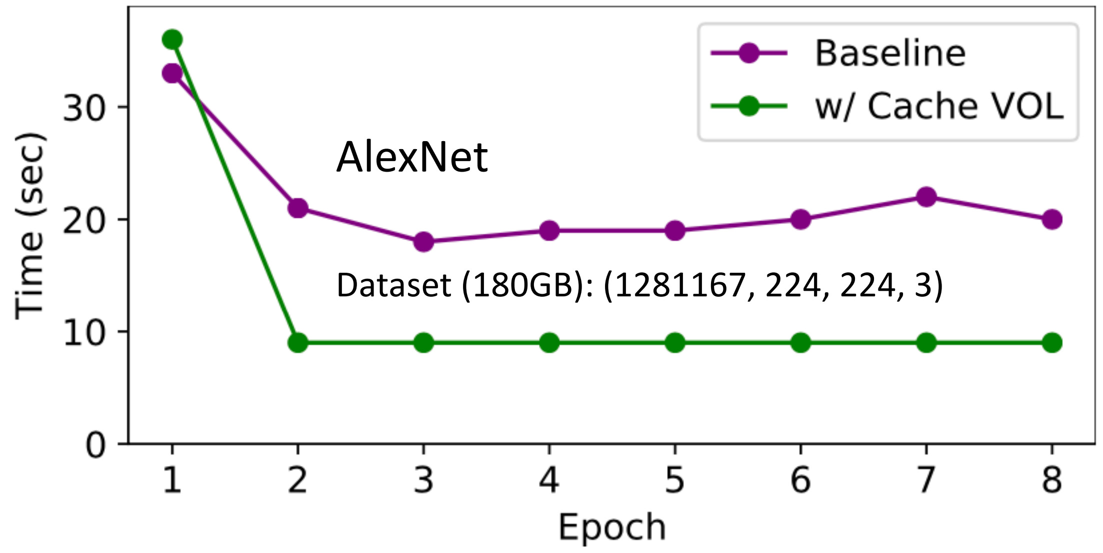
Observed write rate on (Left) Theta and (Right) Summit. The number of time steps is 20. The write rate reported here is the average over the 20 timesteps. The emulated time is 20 seconds per time step on Summit and 200 seconds per time step on Theta.

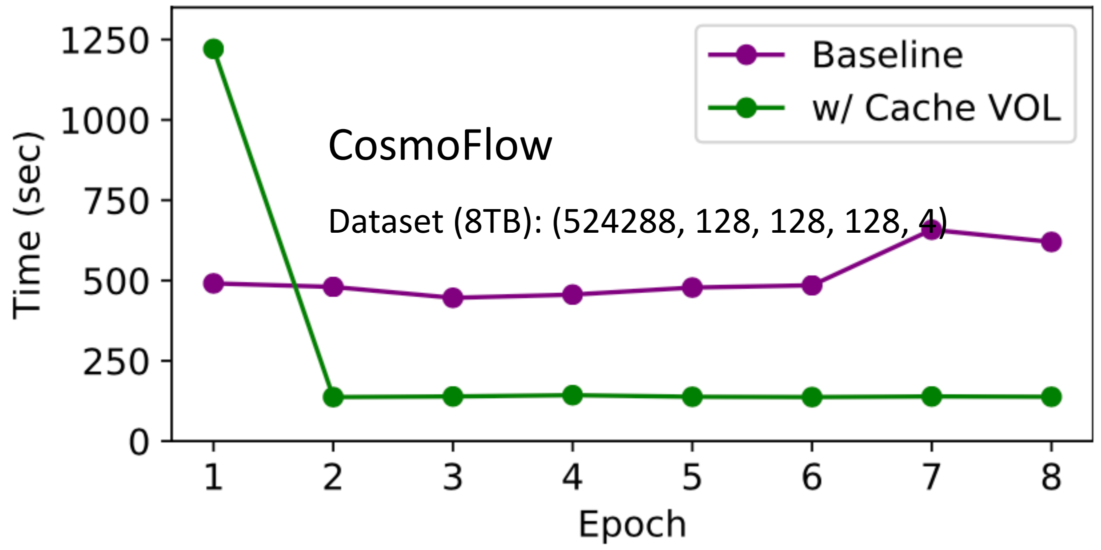
https://github.com/hpc-io/vol-cache.git

Slide from Huihuo Zheng, CCGrid 2022 presentation

Read performance (Deep Learning)

AlexNet: 2D CNN model for image classification.
CosmoFlow: 3D CNN model for predicting universe cosmology parameters.
I/O pattern: each training step randomly read a minibatch of samples from a shared HDF5 file





Improvement of training throughput by caching data on the node-local storage: (Left) AlexNet and (Right) CosmoFlow. The training were done on 16 DGX nodes with 128 Nvidia A100 GPUs on ThetaGPU.

Slide from Huihuo Zheng, CCGrid 2022 presentation

Summary of today's class

- Today's class: HDF5 cache VOL

- Next Class – Proactive Data Containers (PDC)

- Class project –
    - Status update on Apr 4th
    - Final presentation on Apr 20th

- Final exam on Apr 25th