⬅️ Connect with CASS
https://tinyurl.com/2024-CASS-BOFS

CASS Community BOF Days

The Consortium for the Advancement of Scientific Software

June 11 – 13, 2024

https://cass.community/bofs

Announcing CASS

The Consortium for the Advancement of Scientific Software

CASS Basics

- A newly-formed organization
- Sponsored by DOE Office of Advanced Scientific Computing Research (ASCR)
- Established by DOE Software Stewardship Organizations (SSOs)

CASS Goals

- Forum for SSO collaboration and coordination
- Bigger than the sum of its parts
- Vehicle for advancing the scientific software ecosystem

CASS Status

- Defining governance structure
- Establishing community awareness
- Building a team of teams
- Collaborating on outreach

Software Stewardship Organization (SSO) Basics

- Each SSO represents a specific software ecosystem concern
- **Product SSOs:** Programming systems, performance tools, math packages, data/viz packages
- **Portfolio SSO:** Curating & delivering software stack to the community
- **Community SSOs:** Workforce, partnerships

Engage with CASS

- Participate in June 11-13 CASS Community BOF Days: https://cass.community/bofs
- Visit https://cass.community

8 Software Stewardship Organizations (SSOs)

DOE Office of Advanced Scientific Computing Research (ASCR) Post-ECP Projects

COLABS

Training, workforce development, and building the RSE community

CORSA

Partnering with foundations to provide sustainable pathways for scientific software

FASTMATH

Stewardship, advancement, and integration for math and ML/AI packages

PESO

Stewarding, evolving and integrating a cohesive ecosystem for DOE software

RAPIDS

Stewardship, advancement, and integration for data and viz packages

S4PST

Stewardship, advancement and engagement for programming systems

STEP

Stewardship, advancement of software tools for understanding performance and behavior

SWAS

Stewardship and project support for scientific workflow software and its community

11 JUNE 2024

I/O LIBRARY CHALLENGES AND OPPORTUNITES

ROB LATHAM
Research Software Developer
Math and Computer Science Division

WEI-KENG LIAO
Research Professor
Northwestern University

SCOT BREITENFELD
HPC Lead
HDF Group

YOUR PANELISTS

Rob Latham
- I/O libraries
- Applications
- Tutorials

Wei-keng Liao
- Parallel-NetCDF
- ROMIO
- HDF VOL

Scot Breitenfeld
- HDF5
- HPC
- CGNS

CHALLENGE: “EVERYTHING IS BIGGER”

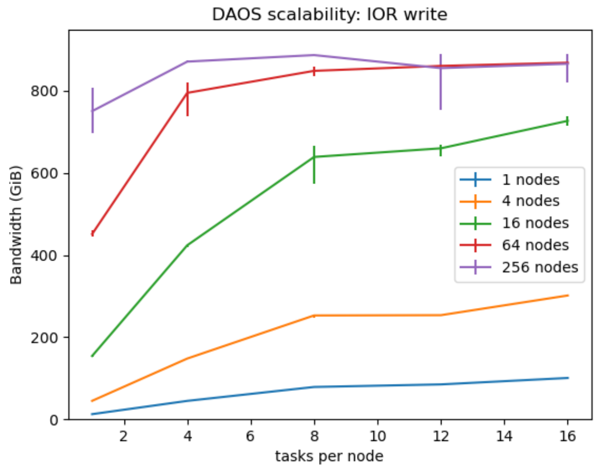
- MPI-4 introduced “large count” methods
- Passing more than 2 billion items into ROMIO caused problems
  - Fixed integer overflows in next MPICH release
  - Are ROMIO’s algorithms ready for billions of items?

```
  size_t count = 1024*1024*1024;
  double *buffer;
  buffer = malloc(count);
  /* 8 GiB transfer:  'count' fits in int */
1 CHECK(MPI_File_write_at_all(fh, count*rank,
         buffer, count, MPI_DOUBLE, &status));
  /* still 8 GiB transfer, but requires a 64-bit MPI_Count */
2 CHECK(MPI_File_write_at_all_c(fh, count*rank,
         buffer, count*sizeof(double), MPI_BYTE, &status))
```

We fixed all the “large transfer” overflows years ago (1); “large count” (2) took a bit more work.

OPPORTUNITY: DAOS

- DAOS (https://daos.io/) is finally here
- Tuning
  — Concurrency, block sizes, transfer sizes
- Better interfaces
  — "scatter-gather"
  — "relaxed mode" consistency semantics
  — (Feels a lot like the old "PVFS" approaches)



This work was done on a pre-production supercomputer with early versions of the Aurora software development kit.

I/O Request Aggregation

Wei-Keng Liao, ECE Department, Northwestern University

Session: Near-term Challenges and Opportunities for I/O

2024 CASS Community BOF Days, June 11, 2024

High-level I/O libraries

- Application users are moving away from using MPI-IO directly
  - MPI-IO programming deals with file offsets
- High-level I/O libraries
  - PnetCDF and HDF5
  - It is easier to deal with logical data structures, e.g. sub-arrays
  - Self-describing, metadata-rich, portable file format
  - Built on top of MPI-IO
  - PIO @NCAR — I/O libraries built on top of PnetCDF, HDF5, NetCDF4

Common practices used in applications

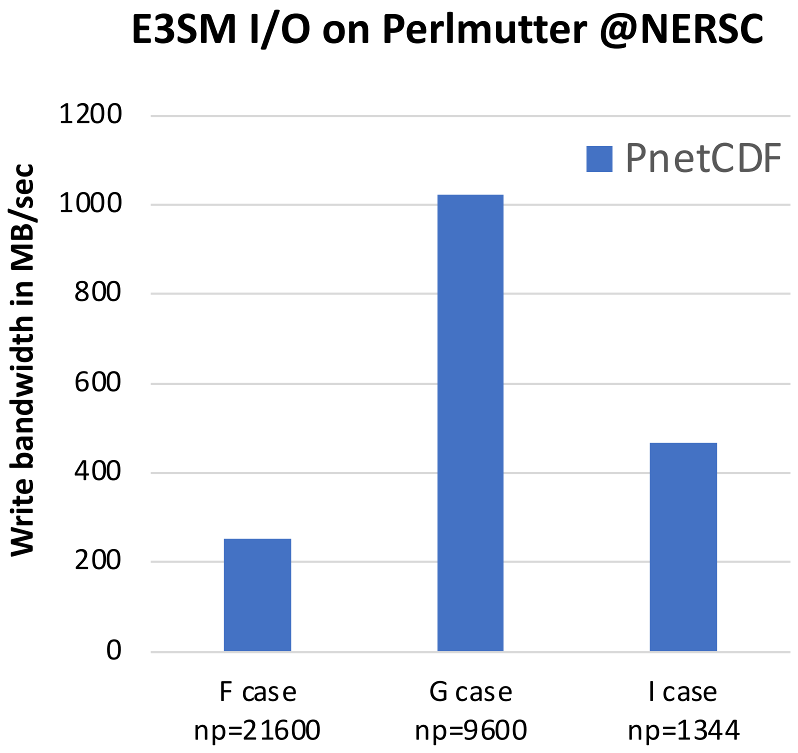
- Computation phase and I/O phase
  - Most applications run a loop of computation and periodically save the intermediate results to files, often referred to as checkpointing.
  - There are often multiple variables to be saved.
  - The same memory buffers may be used for computation and I/O.
  - Asynchronous I/O to overlap the two phases: improves the speed but requires double buffer size.
- User intent for the I/O phase
  - To ensure all variables are safely stored in the file system, before returning to the computation phase.
  - Such intent can be better realized by high-level libraries (requestion aggregation feature in PnetCDF, HDF5, and PIO)

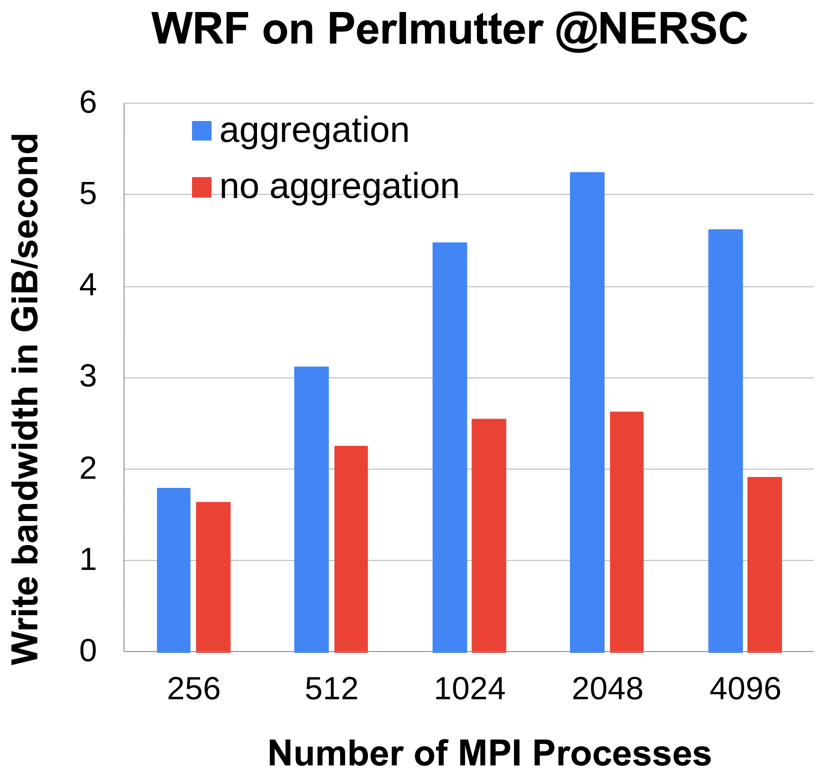
Example applications

- E3SM F case
  - Simulates atmospheric components.
  - 414 variables, 387 are partitioned and 27 are not.
  - 3 data partitioning patterns, along longitude and latitude dimensions, based on the Hilbert space curve algorithm.
  - Each process writes to a large number of non-contiguous file regions (~184K)
- WRF CONUS 2.5km
  - Widely used for weather forecasting and climate research
  - 202 variables, 147 are partitioned and 55 are not.
  - A 2D checker-board partitioning pattern
  - Each process writes to a rectangle subarray per variable

Implementation of I/O request aggregation

- PnetCDF
  - Nonblocking APIs allow users to post multiple requests to the same or different variables
  - A wait_all call to flushed out all pending requests using a single MPI-IO call
- HDF5 multi-dataset APIs
  - H5Dwrite_multi — allows a single call to write multiple variables
- PIO
  - Two aggregation options: subset and box rearrangers
  - A subset of processes are selected to aggregate date from all processes





*Non-aggregation did not complete in a reasonable time.

Challenge for ROMIO — large requests

- Request aggregation increases I/O amount per MPI-IO call
  - Larger requests per MPI process (local aggregation)
  - Larger aggregated amount at each I/O aggregator (at high-level libraries)
- Need large-request support from MPI-IO
  - See Rob Latham's slides

Challenge for ROMIO — memory footprint

- Aggregation increases memory footprints
  - Fileview is flattened into offset-length pairs
  - User buffer datatype is flattened into offset-length pairs
  - Internal memory space required to store these pairs can become significant
- Need a new datatype flattening mechanism
  - MPI collective requests are carried out in multiple rounds of two-phase I/O
  - Each round processes requests of size <= cb_buffer_size
  - Reducing memory footprint by flattening on the fly in each round

Challenge for ROMIO — data sieving

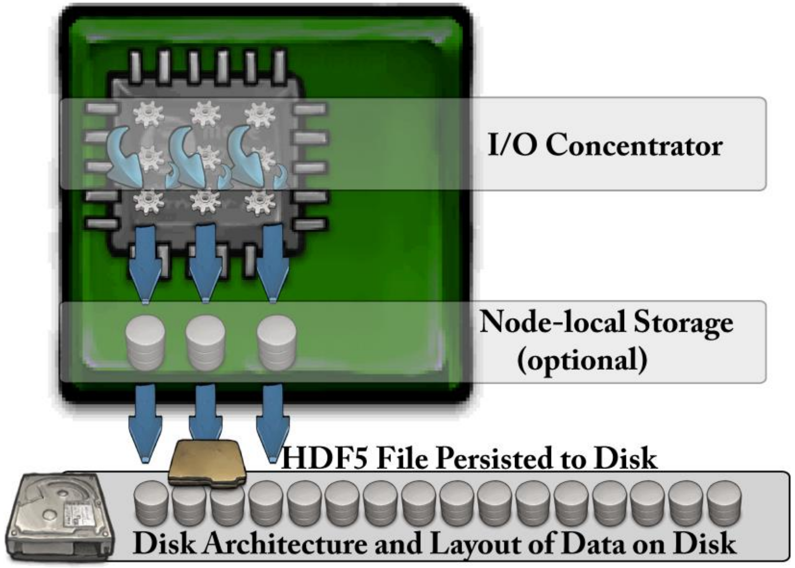
- Data sieving can become expensive
  - I/O aggregator checks “holes” within its file domain from the offset-length pairs received from non-aggregators
  - If holes are found, read-modify-write will perform
  - Sorting and merging offset-length pairs can be expensive, i.e. more than reading the file domain
- Need a threshold for triggering data sieving

Data Aggregation Challenges and Benefits

- Benefits (usually are large node counts)
  - Better use of parallel I/O subsystems, such as node-local storage
  - Reduces the complexity of *file-per-process*
  - By leveraging parallel I/O subsystems, we can effectively mitigate locking and contention issues, leading to substantial performance enhancements, especially at larger processor counts compared to a single-file approach
  - It should be relatively easy for applications to use

- *Challenges*
  - It may still be burdensome working with many subfiles
    - Do the readers understand the data layout and organization
    - May need to combine the files into a valid format
      - can be expensive and negate any benefits from aggregation
    - Hiding data processing during computation to avoid with-out impacting compute performance
  - Unknown at what node count does aggregation start to benefit

Subfiling VFD

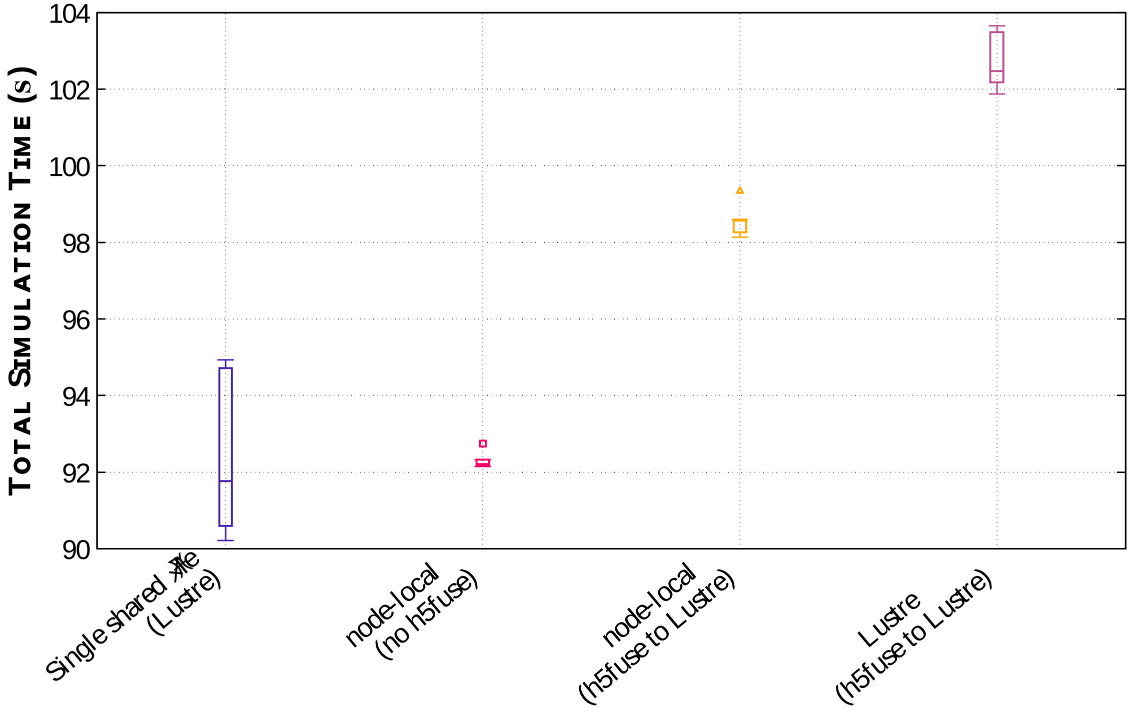

I/O Concentrator
Node-local Storage (optional)
HDF5 File Persisted to Disk
Disk Architecture and Layout of Data on Disk

The resulting subfiles can be read using the Subfiling VFD or coalesced via a post-processing step into a single HDF5 file

- HDF5's ***h5fuse*** is a tool to recombine subfiles into a single HDF5 file

a. I/O Concentrators are implemented as independent threads attached to a normal HDF5 process.
b. MPI is utilized for communicating between HDF5 processes and the set of I/O Concentrators.
c. Because of (b), applications need to use *MPI_Init_thread* to initialize the MPI library.

Challenges of data aggregation and node-local storage



- Cabana/ExaMPM
- Outputs data every 100 timesteps (5 total)
- Frontier, 256 nodes