The HDF Group

ILLINOIS INSTITUTE OF TECHNOLOGY

Webinar

Friday, Aug 5th 2022

Agenda

- Hermes Project Overview
- Hermes Architecture recap
- Buffer Organizer
- Future work
- Frequently asked questions

Project Overview

Hermes Project

The team

Collaborative project
funded by NSF
Award: CSSI 1835764

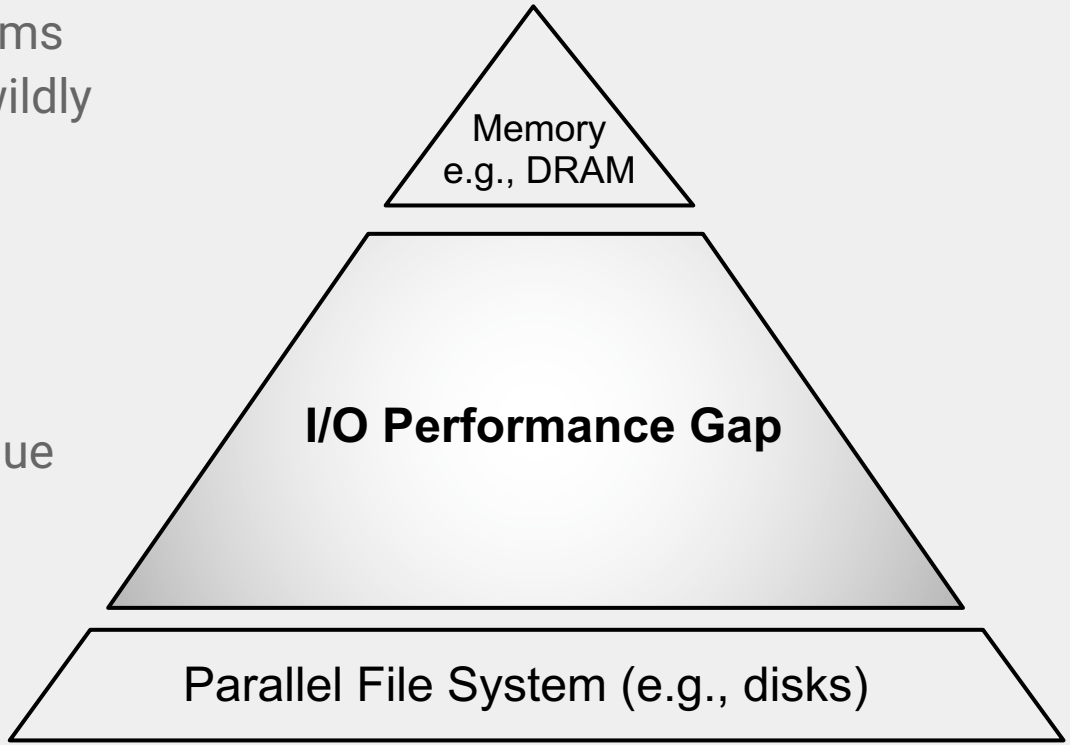
ILLINOIS INSTITUTE OF TECHNOLOGY

The HDF Group

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

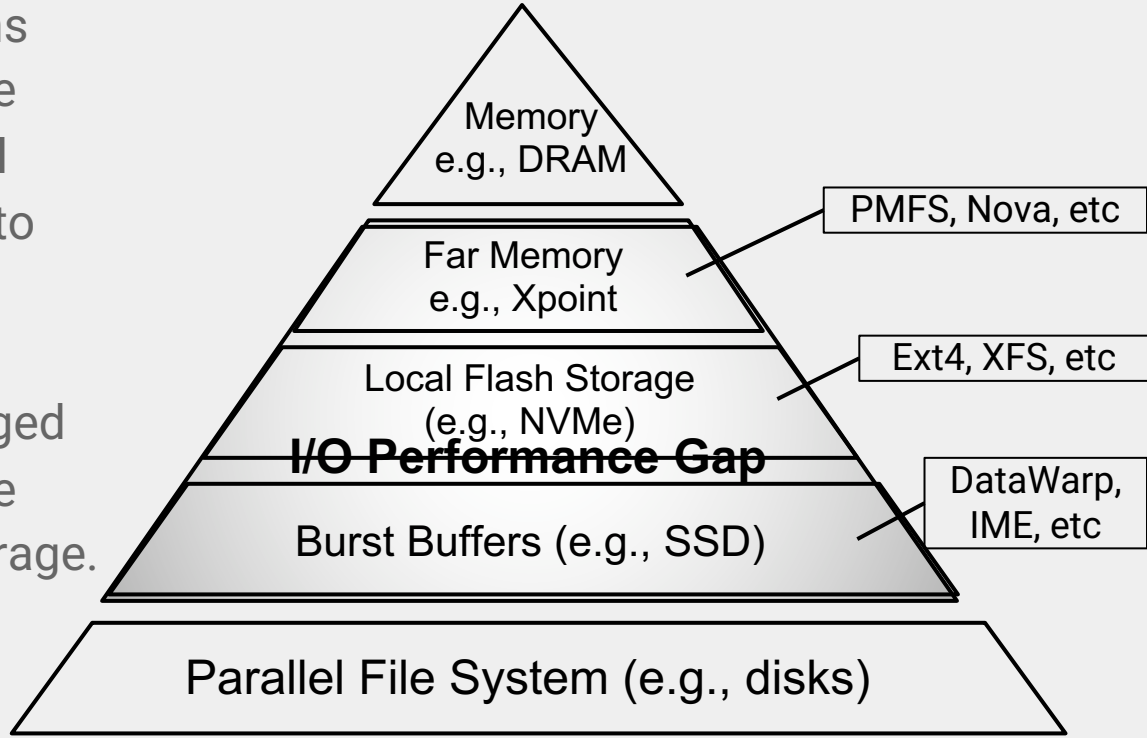
I/O Performance Gap

- Traditionally memory systems and storage demonstrate wildly different performance.
  - Access latency
  - Bandwidth
  - Data representation
- Applications experience performance degradation due to slow remote access to storage.



Hardware Answer

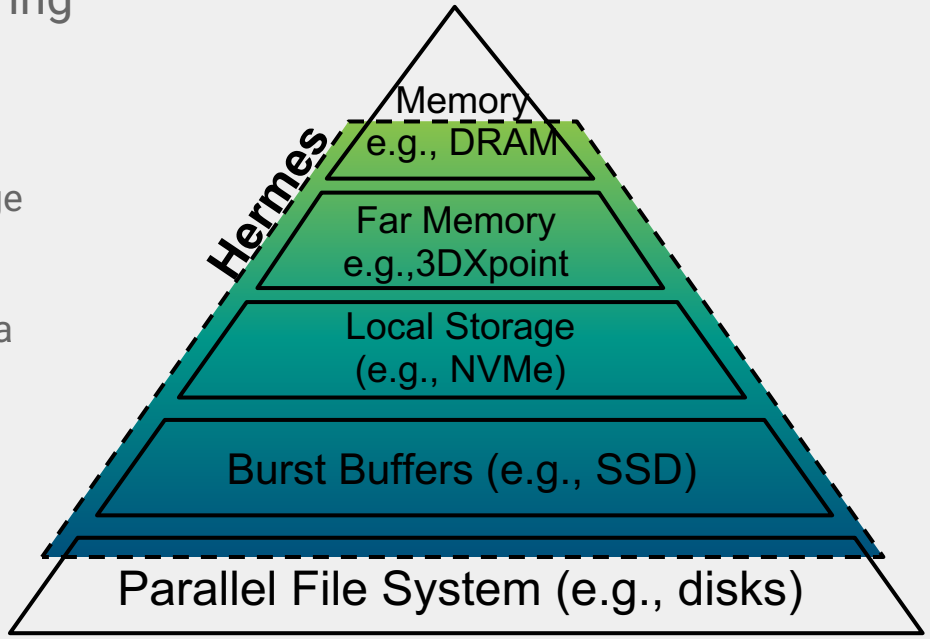
- Modern storage system designs include multiple tiers of storage organized in a **deep distributed storage hierarchy**. The goal is to mask the I/O gap.
- Each system is independently designed, deployed, and managed making very difficult to reap the benefits of the hierarchical storage.



Ideally, the presence of multiple tiers of storage should be **transparent** to applications without having to sacrifice **I/O performance**.

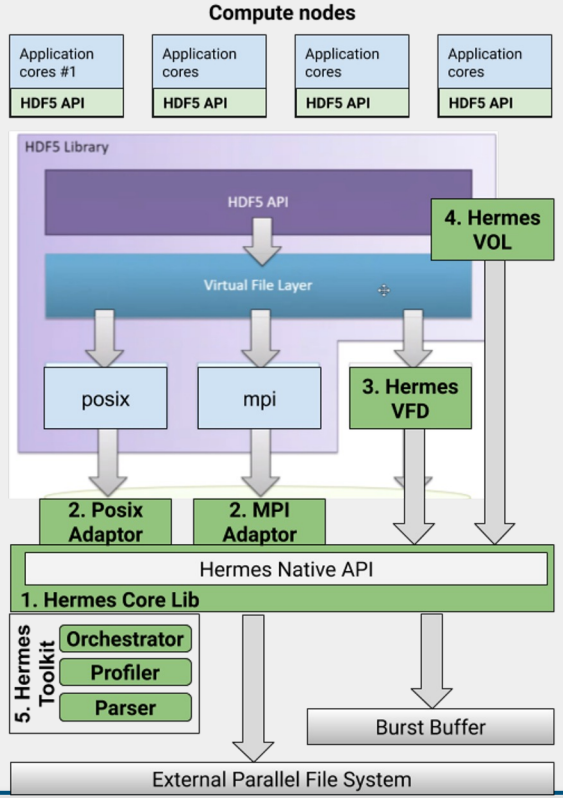
Hermes Overview

- A new, multi-tiered, distributed buffering system that:
  - Enables, manages, and supervises I/O operations in the Deep Distributed Storage Hierarchy (DDSH).
  - Offers selective and dynamic layered data placement.
  - Is modular, extensible, and performance-oriented.
  - Supports a wide variety of applications (scientific, BigData, etc.,).



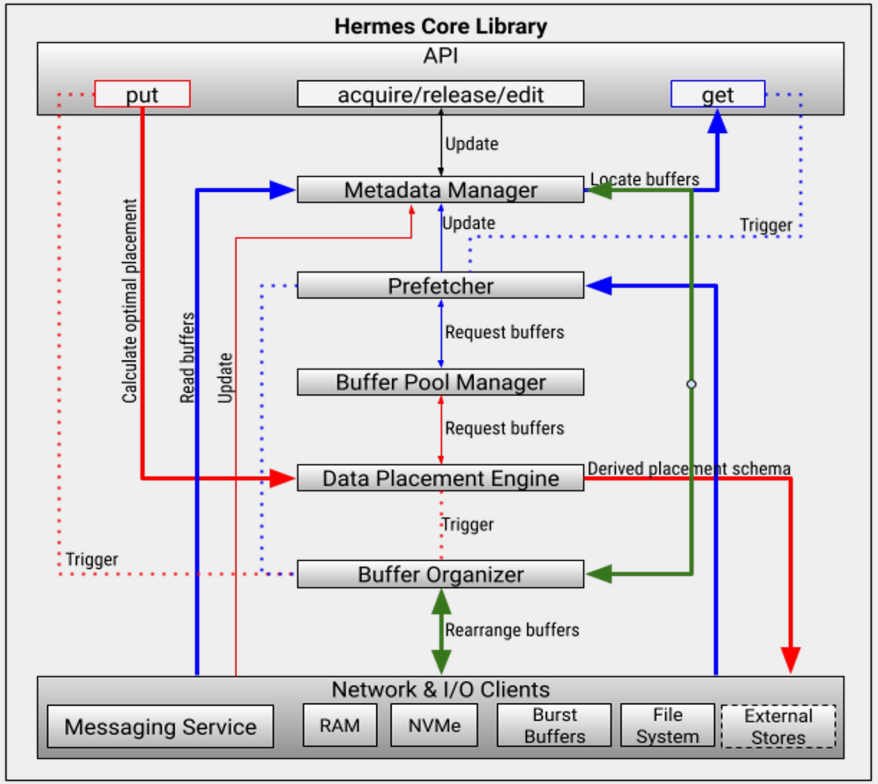
Hermes ecosystem

1. Hermes core library
   a. Manages tiers transparently
   b. Facilitates data movement in the hierarchy
   c. Provides native buffering API
2. Hermes Adapters
   a. POSIX, MPI-IO, Pub-Sub, etc
      i. Intercept I/O calls to Hermes
      ii. Boosts legacy app support
3. Hermes VFD
   a. Directs HDF5 I/O to Hermes native API
4. Hermes VOL
   a. Captures application's behavior and provides hints to Hermes core lib
5. Hermes Toolkit



High-Level Architecture

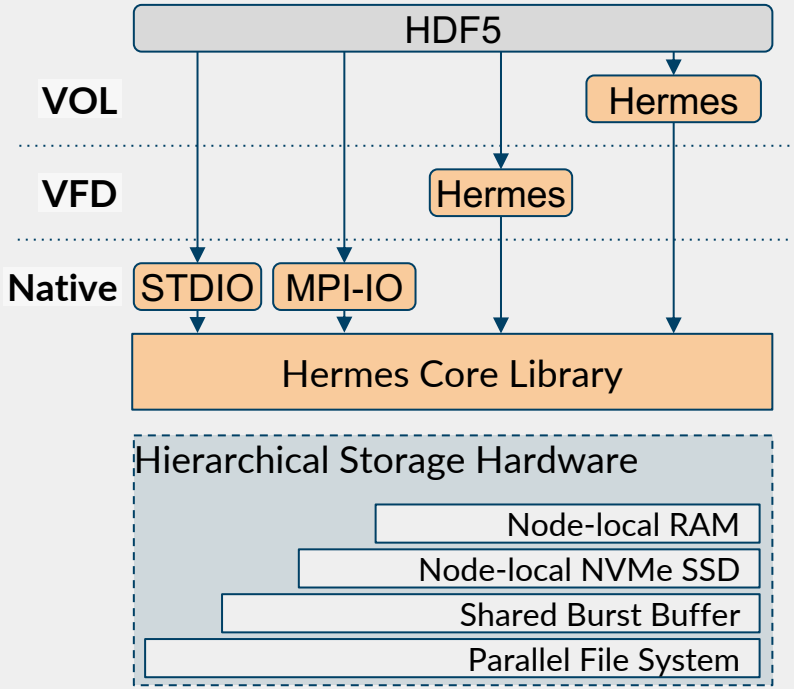
- API
- Metadata Manager
- Prefetcher
- Buffer Pool Manager
- Data Placement Engine
- Buffer Organizer
- I/O Clients



Hermes Adapter Layer

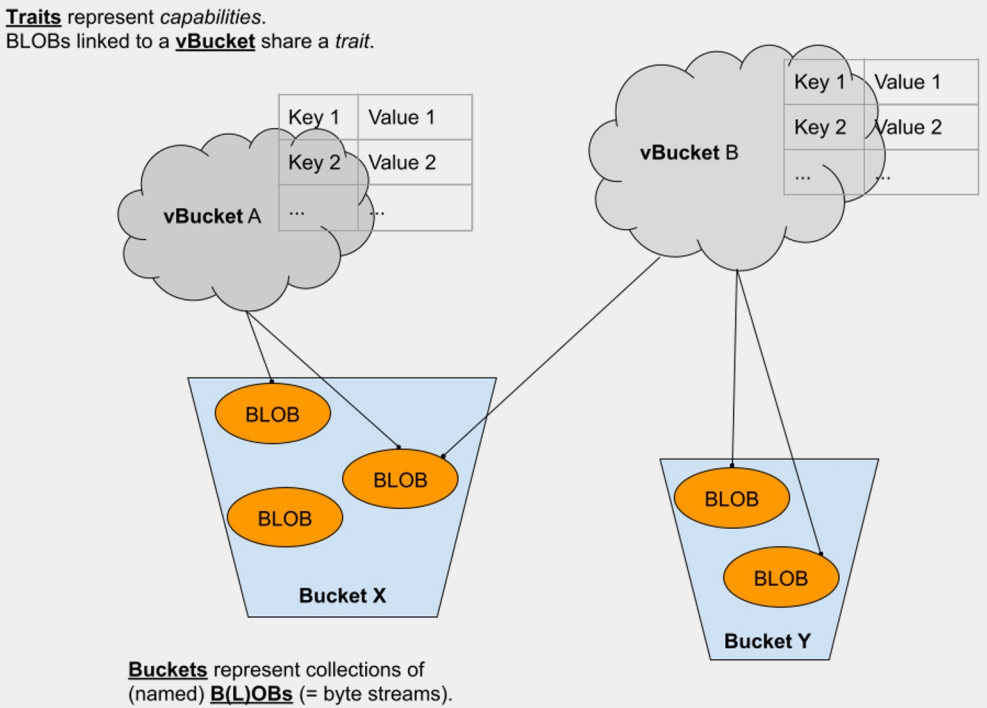
Applications can natively interact with Hermes using existing I/O Interfaces

- Standard Interceptors
  - STDIO
  - POSIX
  - MPI-IO
  - Publish-Subscribe
- HDF5 Level
  - Hermes VFD
  - Hermes VOL



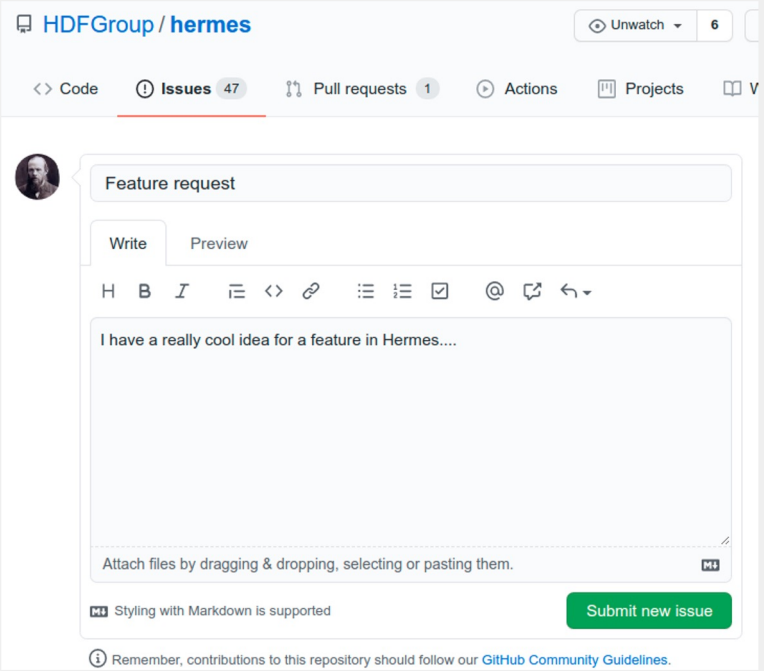
Hermes Data Model

- Blobs
  - Unit of data as key-value pairs
  - Value as uninterpreted byte arrays
  - Stored internally as a collection of buffers across multiple tiers
- Bucket
  - Collection of blobs
  - Flat blob organization
- Virtual Bucket
  - Linked blobs across buckets
  - Attached capabilities
- Traits
  - Ordering, grouping, filtering
  - Compression, deduplication, etc



How Can I Get Involved?

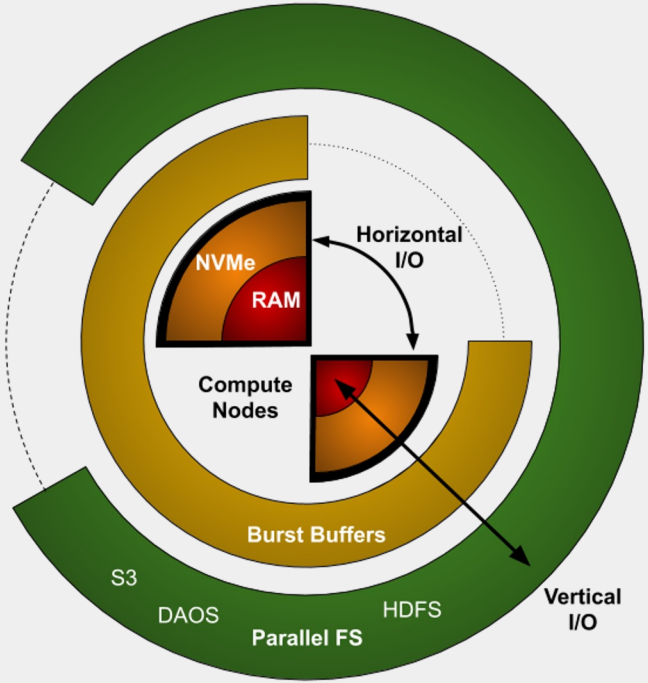
- Github repo: https://github.com/HDFGroup/hermes
- Create an issue to submit feedback, use cases, or feature requests.
- Note: Hermes is still under active development.
- Join us in the Hermes Forum:
  - https://forum.hdfgroup.org/c/hermes



The Buffer Organizer (Borg)

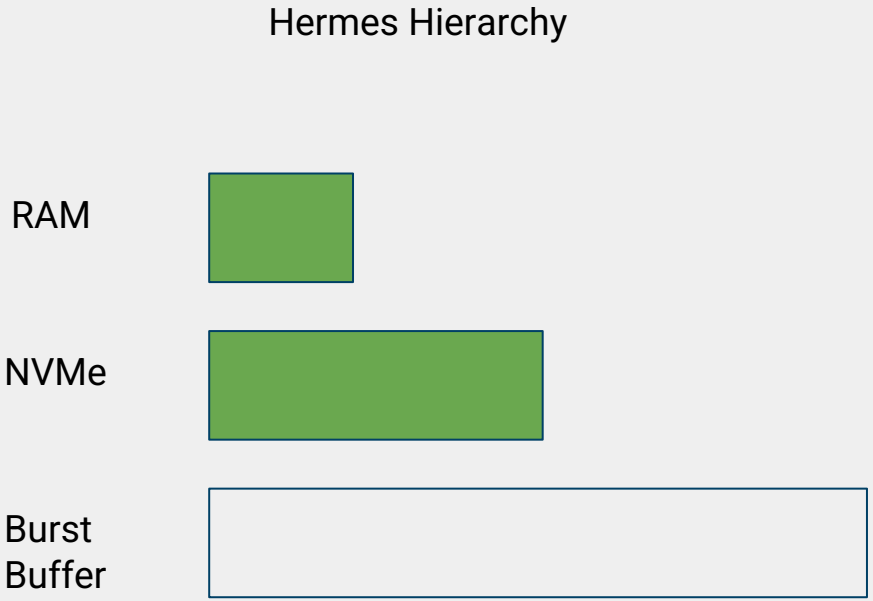
Buffer Organizer Objectives

- Observation:
  - Injection point ≠ resting location OR
  - Write optimization ≠ Read optimization
- DPE - BORG
  - predictor - corrector model
- Responsibilities:
  - Management of hierarchical buffering space
    - Data flushing
    - Read acceleration
  - Manage data life cycle, or journey
    - When is the blob in equilibrium?
    - How to eliminate unnecessary data movement?



Motivating Example - Checkpointing

- Status: Hermes' RAM and NVMe tiers are full, and the Burst Buffer tier is empty.
- Great for reading.
- Not so great for writing.
- Still beats PFS.

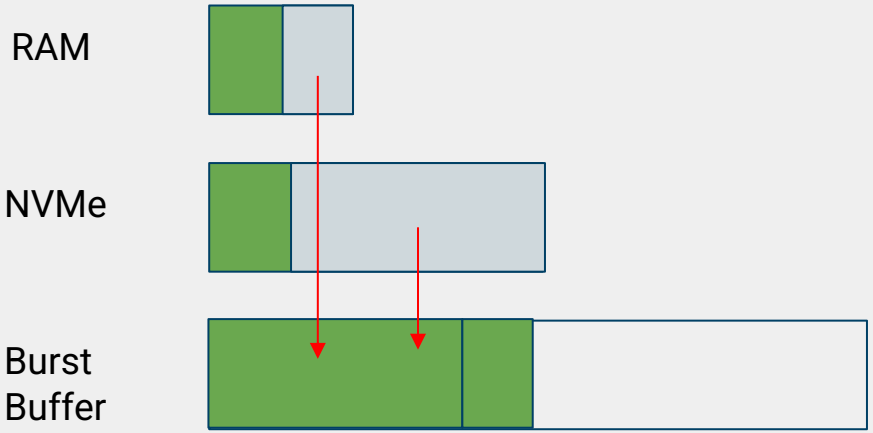


Solution - Checkpointing

Write-heavy workloads

- BORG makes space in faster tiers by moving data to slower tiers.
- Least “important” data is moved first.
- Each Blob has a score based on recency and frequency of access

Hermes Hierarchy

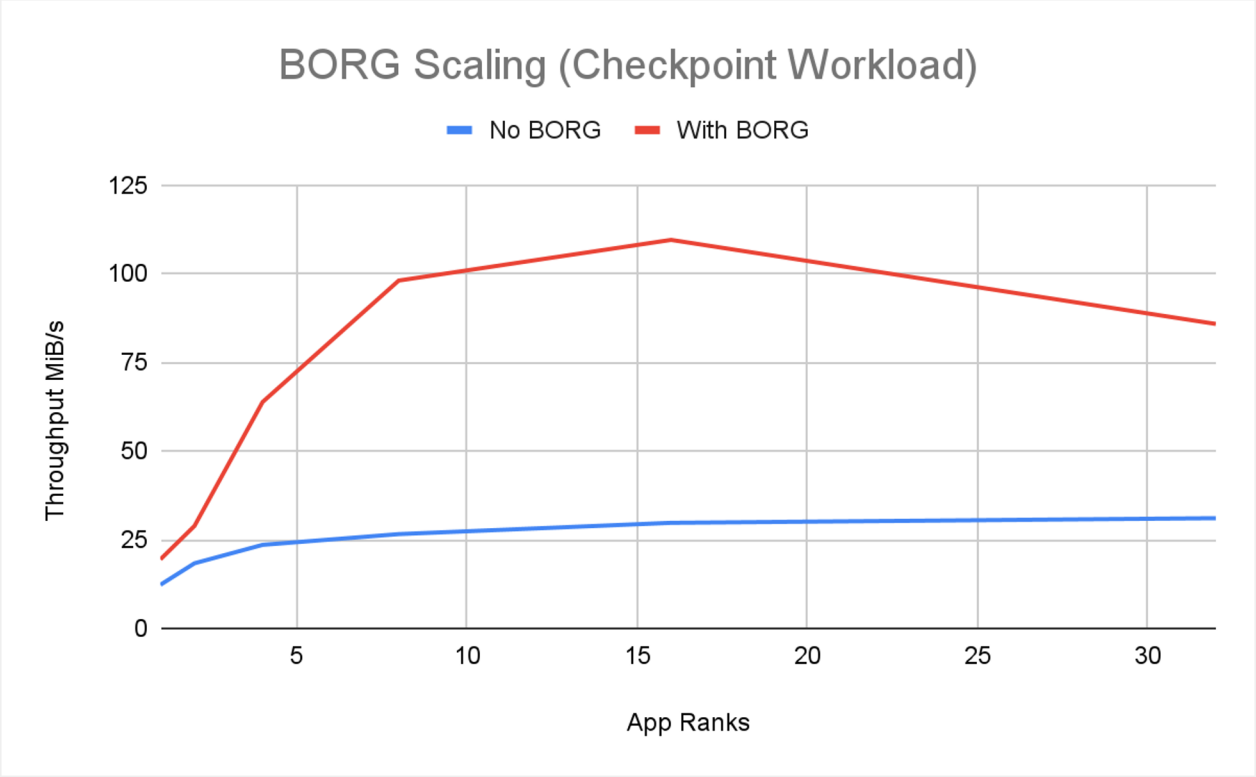


Results - Checkpointing

1.78x speedup with BORG

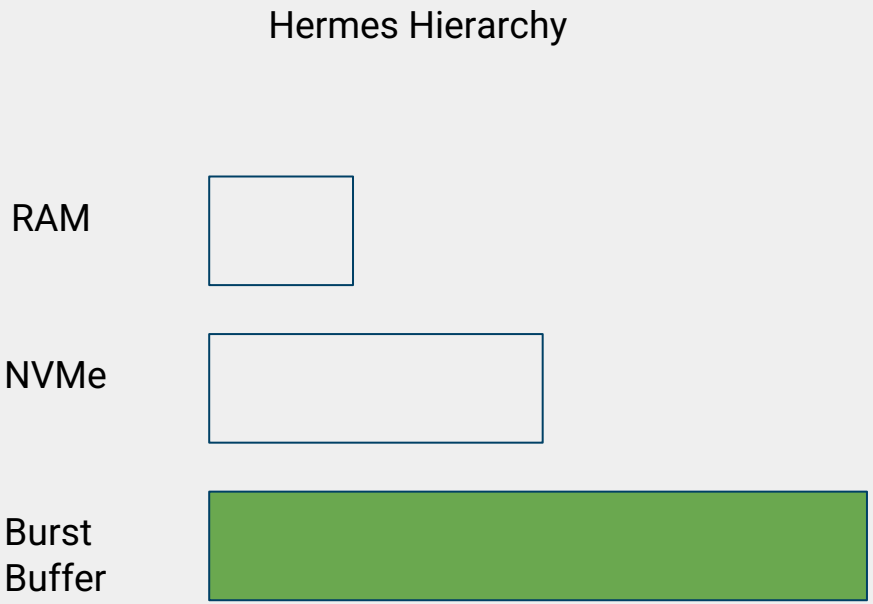
Results - Checkpointing

4x speedup with BORG



Motivating Example - Read Heavy

- Status: Hermes' RAM and NVMe tiers are empty, and the Burst Buffer tier is full
- Great for incoming writes
- Not so great for reads
- Still beats PFS

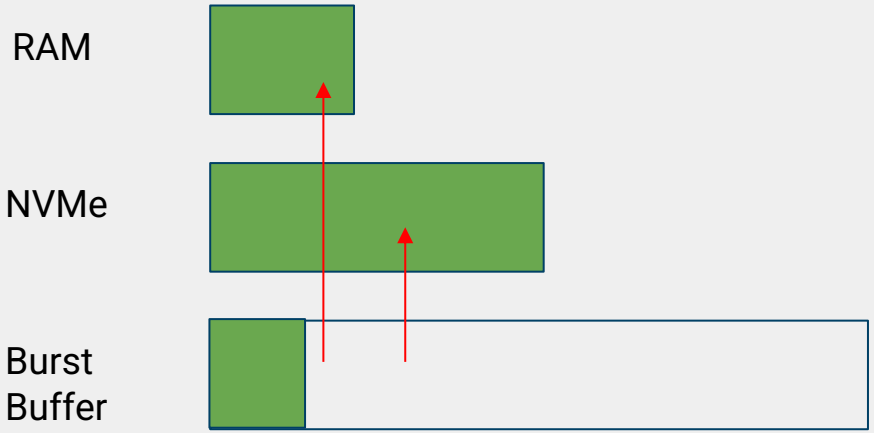


Solution - Read Heavy

Read-heavy workloads

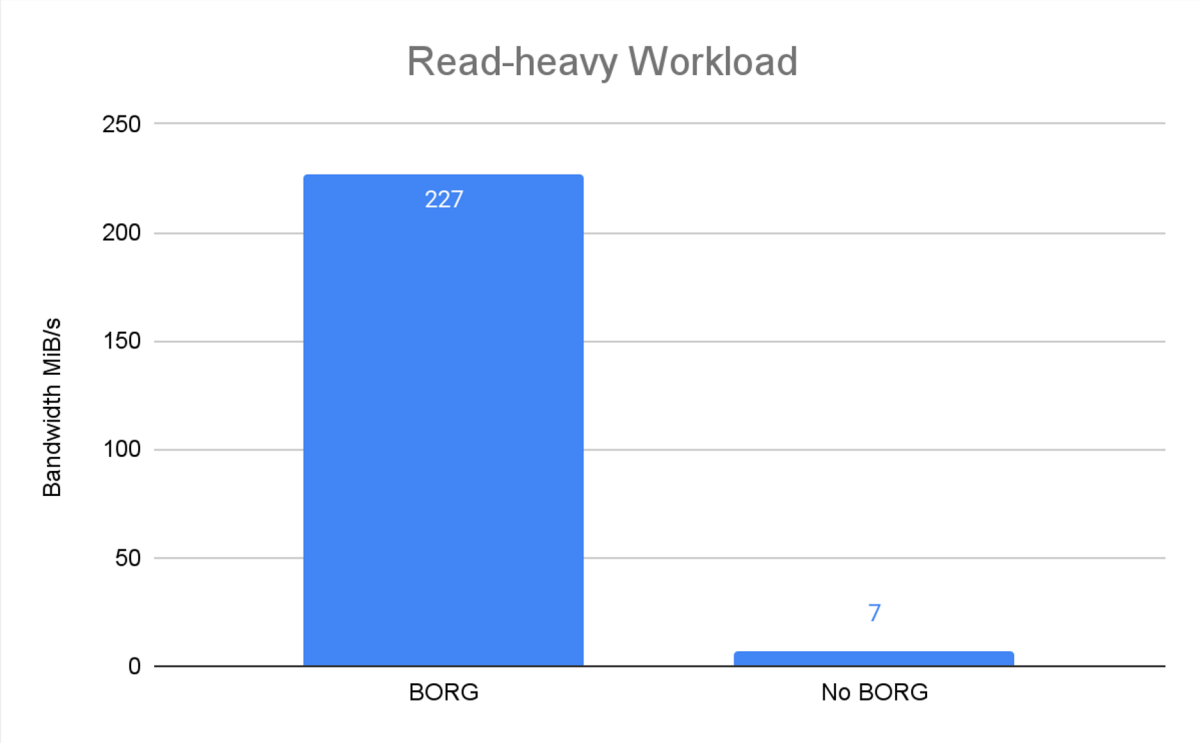
- BORG moves data to faster tiers from slower tiers.
- Data elevated based on importance score
- Prefetching (coming soon)

Hermes Hierarchy



Results - Read Heavy

32x speedup with BORG



User Control

- What about mixed workloads?
- Hermes configuration file provides the `bo_capacity_thresholds` option.
- For each tier, specify the minimum and maximum capacity threshold.
- Number of BORG threads also configurable

```
# BORG disabled
bo_capacity_thresholds = {
  {0.0, 1.0},
  {0.0, 1.0},
  {0.0, 1.0},
};
```

```
# Write heavy
bo_capacity_thresholds = {
  {0.0, 0.70},
  {0.0, 0.70},
  {0.0, 1.0},
};
```

```
# Read heavy
bo_capacity_thresholds = {
  {0.80, 1.0},
  {0.80, 1.0},
  {0.0, 1.0},
};
```

Roadmap

Roadmap

- BORG will be released in Hermes v0.8.0-beta (any day now)
- Hermes is currently in Beta.
- Monthly release schedule
  - Alternating bug-fix releases with feature releases
- Activities to complete before v1.0
  - Stage-in/stage-out
  - Prefetching
  - Applications and scalability testing
- Version 1.0 in October.
- Github repo: https://github.com/HDFGroup/hermes
- Getting started guide: https://github.com/HDFGroup/hermes/wiki/1.-Getting-Started

FAQ

FAQ (1)

How does Hermes integrate into modern HPC environments?

- As of now, applications link to Hermes (re-compile or dynamic linking). We plan to integrate Hermes with the system scheduler so to provision buffering resources.
- Hermes can also be run as a service by spawning the Hermes daemons before the application starts. This way, workflows are supported where one job (typically a simulation) produces data and another job (typically analytics) consumes the data directly from the buffering system.
- Lastly, for HPC environments that support containers, we have bundled Hermes into Docker images. Singularity containers may also be supported if requested.

FAQ (2)

What are Hermes build dependencies?

- A C++ compiler that supports C++ 17.
- Thallium - RPC library for HPC.
- GLOG - The Google logging library (v0.4.0).
- Google ORTOOLS for constraint optimization.
- MPI (tested with MPICH 3.3.2 and OpenMPI 4.0.3).
- The Catch2 testing framework (only required if built with -DBUILD_TESTING=ON).
- Hermes README

FAQ (3)

Does running Hermes require any elevated privileges?

- No. Hermes does not require administrative privileges. Hermes is designed to run in user space as an application extension.

FAQ (4)

How to get started with Hermes? Do I need to make changes to my application?

- Start with provided adapters, currently, POSIX, STDIO, MPI-IO
  - No code changes, just `LD_PRELOAD`
- Use the HDF5 Hermes VFD; required code change: a new HDF5 file access property or no code change: default HDF5 file access property
- Use Hermes API directly; mainly for new development
  - Benefits: maximum flexibility and greatest potential performance gains; the Hermes native API is simple and minimalistic.
    - Buckets and Blobs
    - Virtual Buckets and Traits

FAQ (5)

How scalable is Hermes?

- On paper, Hermes should be as scalable as the RPC components of Mochi.
- Currently, our development testbed consists of only 64 nodes, but Exascale is the target, clearly.
- Scalability testing will be an important activity over the next year.
- Help us to explore more :)

FAQ (6)

Which MPI implementations are supported?

- For the native Hermes API, we support OpenMPI, MPICH and Intel MPI.
- Currently, at the MPI-IO adapter supports only MPICH. The following updates to MPI_Status happen in 3 places in mpiio.cc, and won't compile with OpenMPI.

```
mpi_status->count_hi_and_cancelled = 0;
mpi_status->count_lo = ret;
```

- We intend to remove the specifics of the MPICH implementation and support more MPI implementations.

FAQ (7)

What data placement strategies are available in Hermes? Can I bring my own?

- Data placement in the hierarchical space is configurable and extensible.
- We currently provide three strategies: Random, Round-Robin and MinimizeIoTime.
  - **Random**, each blob is distributed randomly between buffering targets favoring load balancing.
  - **Round-Robin**, each blob is placed to the next buffering target in order (mimics PFSs)
  - **Linear Optimization**, each blob is decomposed and distributed to a subset of the best available buffering targets. This is the default engine and is customizable through the set of constraints the algorithms receives. For example, the user can request targets with limited remaining capacity to not be considered for placement.
- We will provide an interface for user-defined data placement strategies.

FAQ (8)

How do I prevent specific paths from being intercepted by an adapter?

- If you want to exclude a Hermes adapter from intercepting certain paths, you currently add the directory to `hermes::adapter::kPathExclusions` (in `interceptor.h`) and recompile.
- We will provide an environment variable to facilitate that.

FAQ (9)

Are there examples of how to use the Hermes native API?

- Several examples are available here.
- Start with end_to_end_test.

FAQ (10)

How much RAM should I set aside for Hermes buffering space?

- Configurable by the user.
- 10-20% of the available RAM is recommended.

FAQ (11)

What is the RAM trade-off between applications and Hermes?

- More RAM for Hermes can lead to higher performance (e.g., data-intensive science).
- No RAM for Hermes means skipping the DRAM tier entirely (e.g., older servers)

FAQ (12)

What is the RAM footprint of Hermes metadata management?

- Total bytes for metadata =
  280 +
  (192 * num_targets) +
  (288 * max_buckets_per_node) +
  (352 * max_vbuckets_per_node) +
  (64 * max_blobs_per_node) +
  (64 * max_blobs_per_node * avg_buffers_per_blob)

- We plan to implement a technique where unused metadata information is swapped into NVMes to relieve DRAM pressure.

FAQ (13)

Is there any interference between Hermes and the OS page cache?

- There is interference.
- It’s a long story.

FAQ (14)

Is prefetching currently supported in Hermes?

- We are going to support it, just not in this release.
  - We have an initial implementation and test results, but more polishing is needed.
- The prefetching decision is based upon the importance score of a blob.
  - If a blob is deemed important by either the user or by the system (using collected statistics reflecting the access characteristics), the prefetcher will kick in and elevate the blob to upper tiers of the hierarchy.
    - Less important blobs will move in the opposite direction and start moving down in the hierarchy.

FAQ (15)

How is Hermes configured?

- Currently, through a configuration file; better tooling will be forthcoming.
- Configuration examples can be found here.
- Adjustments for system *and* application might be needed. See this Hermes Wiki entry.

Multi-Tiered
Distributed I/O
Buffering System

Learn more
tinyurl.com/hermes-buffering

https://github.com/HDFGroup/hermes

Thank you.

Contact us
akougkas@iit.edu
gheber@hdfgroup.org

The HDF Group

ILLINOIS INSTITUTE OF TECHNOLOGY