The HDF Group

ILLINOIS INSTITUTE OF TECHNOLOGY

July 7th 2021

ESRF/HUG

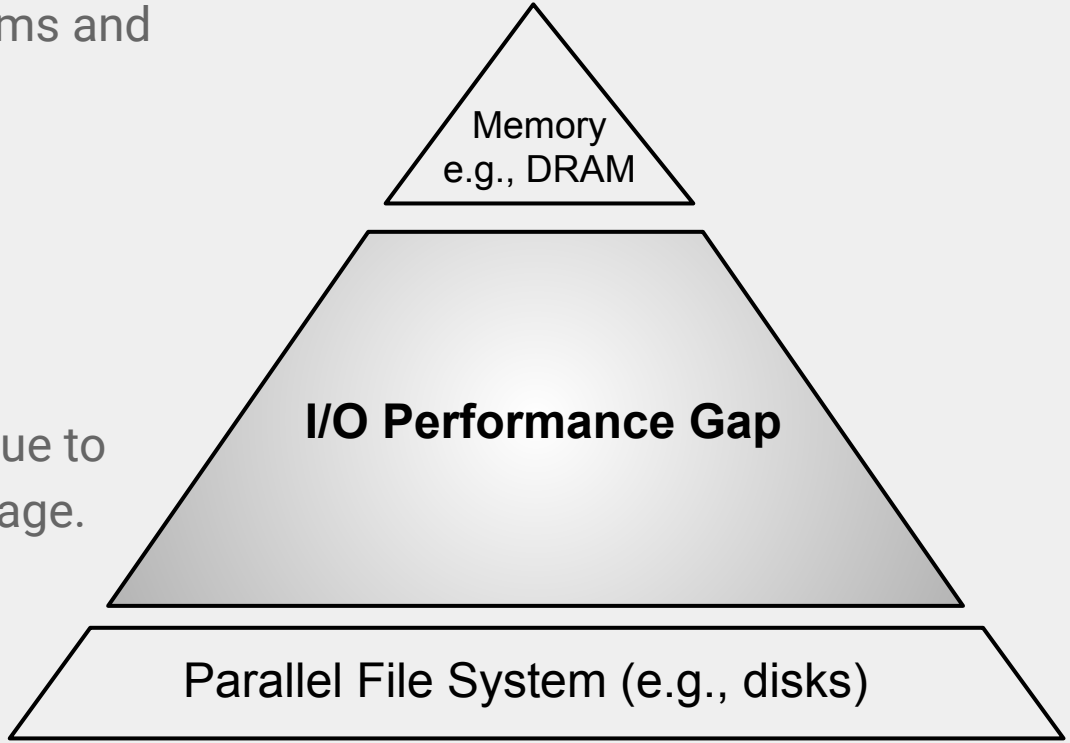
Anthony Kougkas - akougkas@iit.edu

Agenda

- Modern systems and the I/O bottleneck
- Hermes project overview
- Hermes ecosystem and architecture
- How to use Hermes

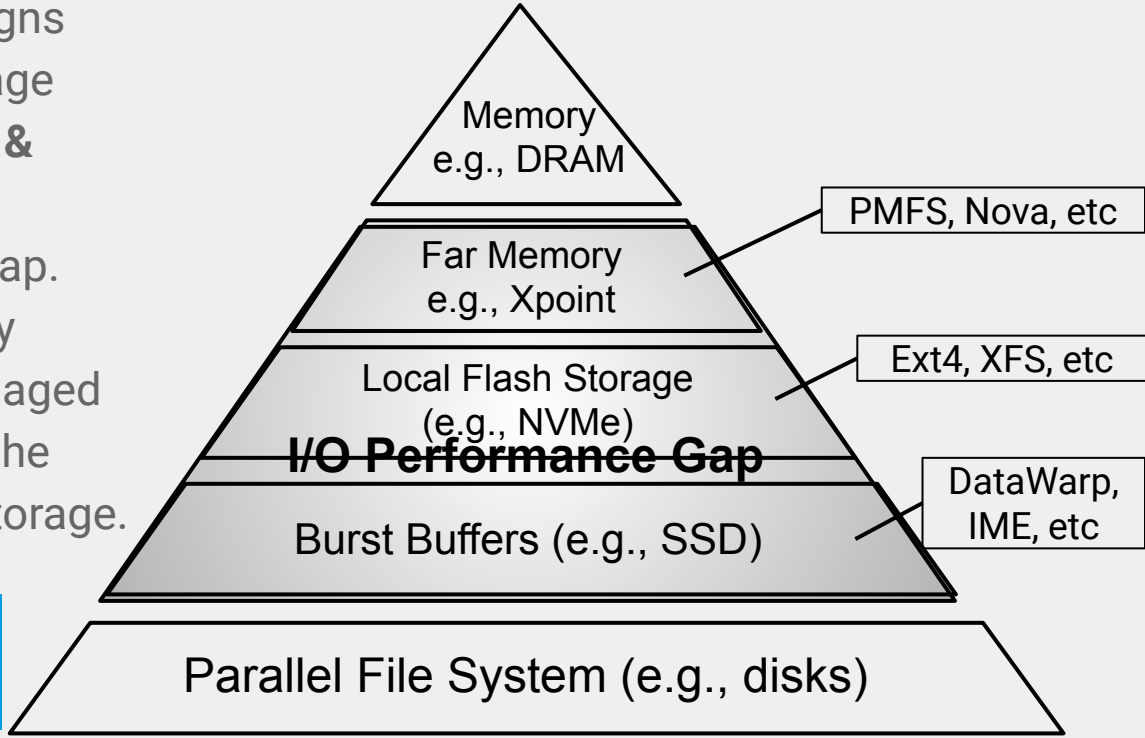
I/O Performance Gap

- 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 memory & storage hierarchy (DMSH)**. 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**.

I/O Challenges in Deep Memory & Storage Hierarchies

Increased **complexity** of data movement between and among the layers of the DMSH.

Each DMSH layer is an **independent** system requiring expertise to manage.

Lack of **automated** DMSH buffering management software.

Project Overview

Hermes Project

The team

Collaborative project

funded by NSF

ILLINOIS INSTITUTE OF TECHNOLOGY

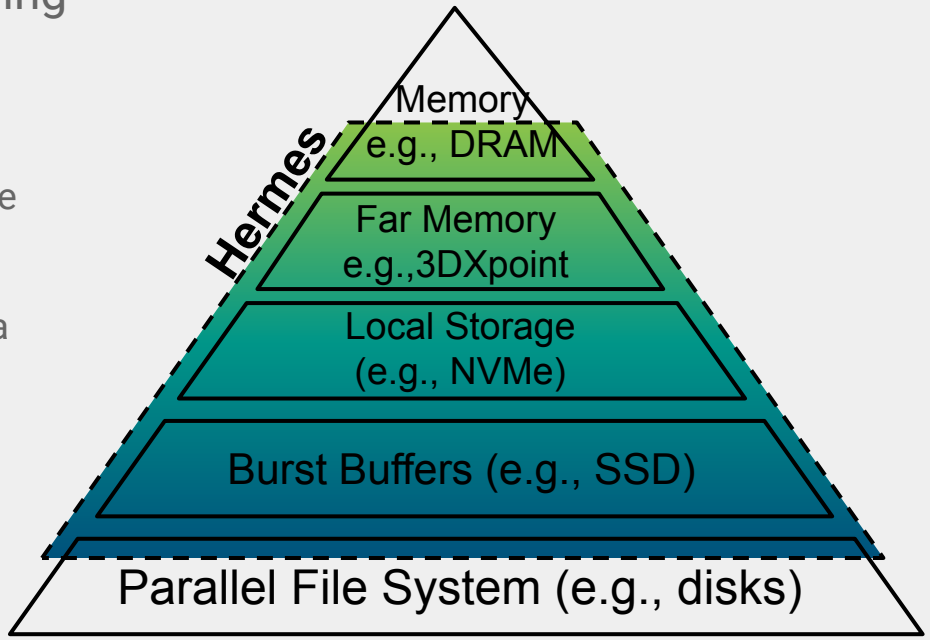
The HDF Group

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

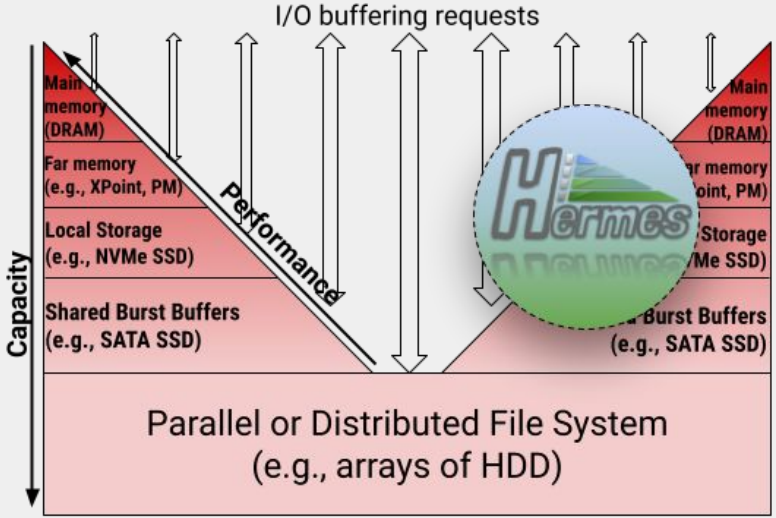
Hermes Overview

- 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.,).



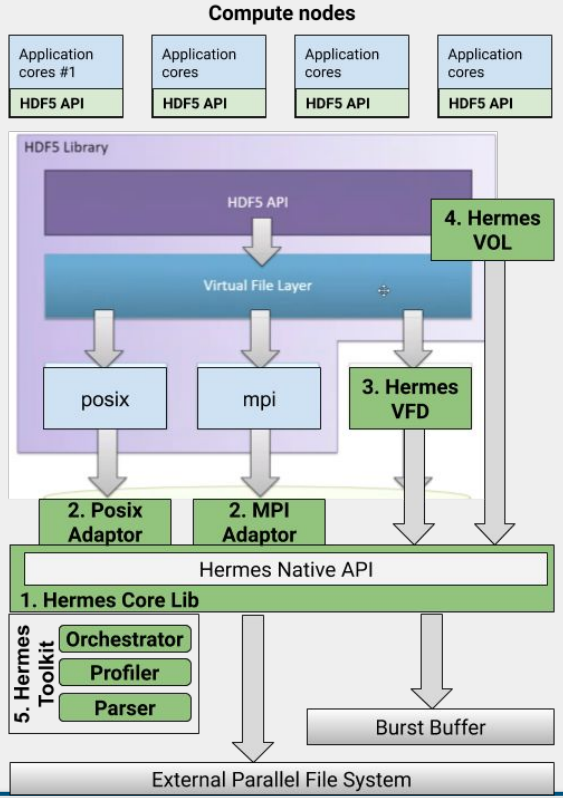
Objectives

- DMSH requires:
  - a scalable, reliable, and high-performance software to efficiently and transparently manage data movement.
  - new data placement algorithms, memory and metadata management, and an efficient communication fabric.
- Hermes strives for:
  - being application- and system-aware,
  - maximizing productivity and path-to-science,
  - increasing resource utilization,
  - abstracting data movement,
  - maximizing performance, and
  - supporting a wide range of applications



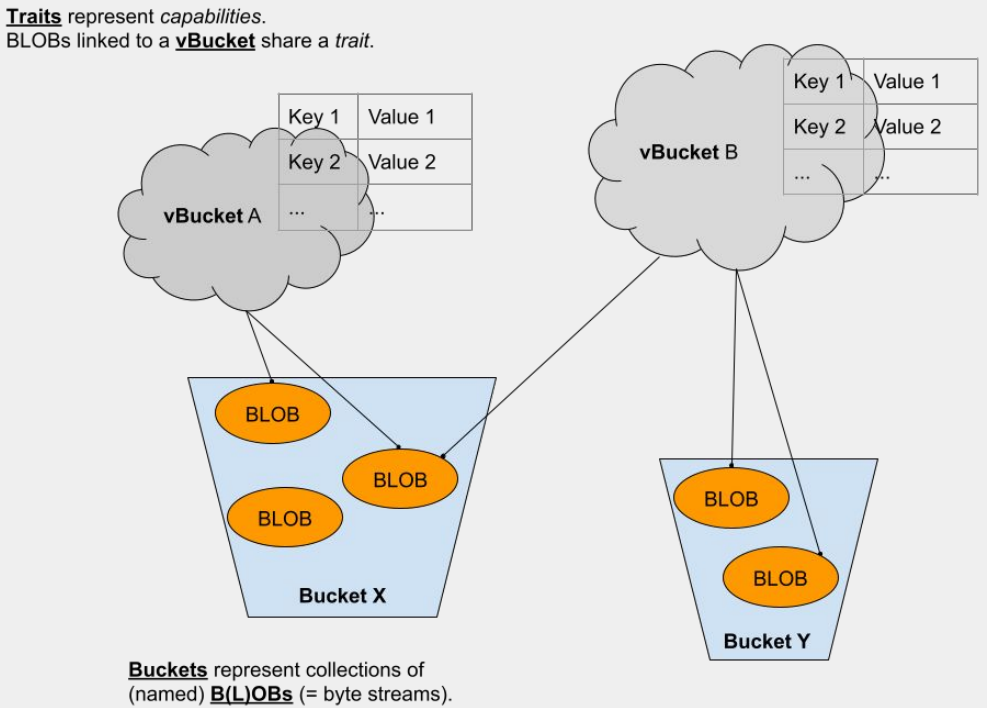
Hermes ecosystem

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



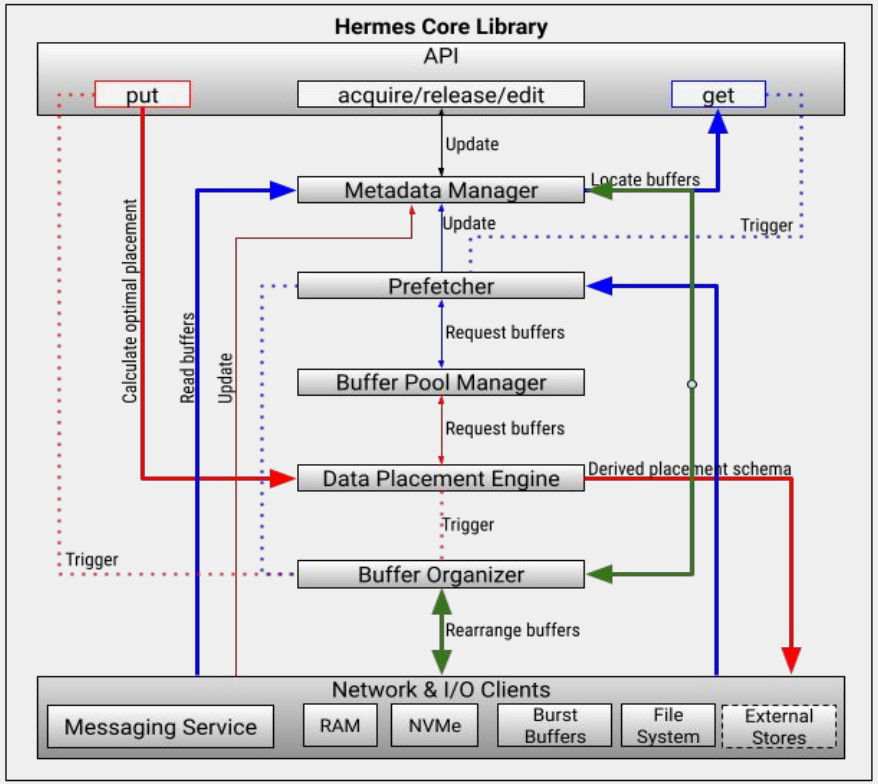
Hermes Data Model

- 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



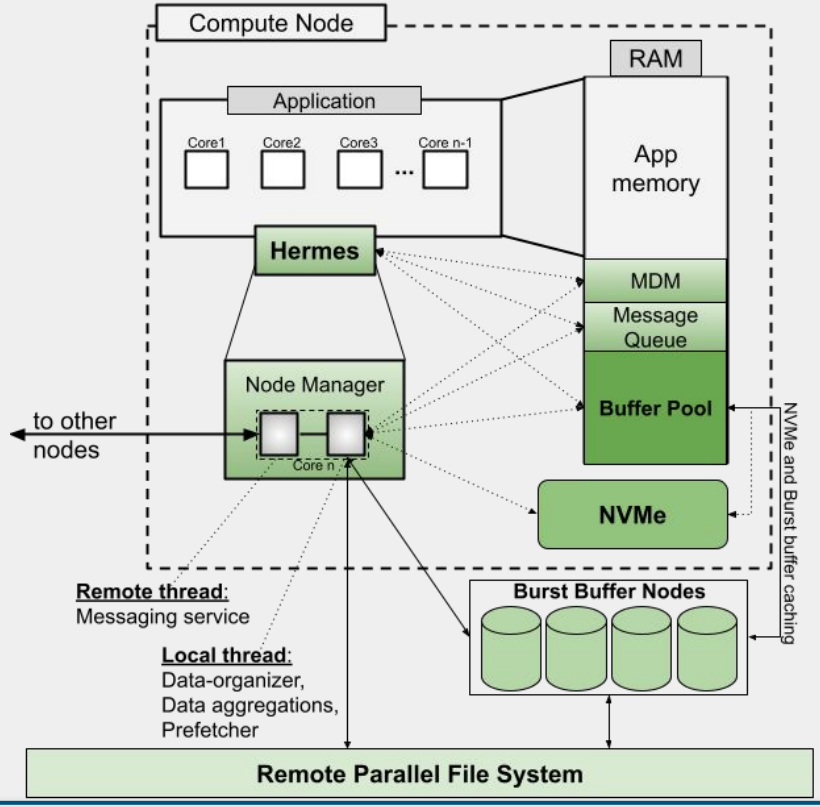
High-Level Architecture

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



Hermes node design

- Dedicated core for Hermes
- Node Manager
  - Dedicated multithreaded core per node
  - MDM
  - Data Organizer
  - Messaging Service
  - Memory management
  - Prefetcher
  - Cache manager
- RDMA-capable communication
- Can also be deployed in I/O Forwarding Layer (I/O FL)

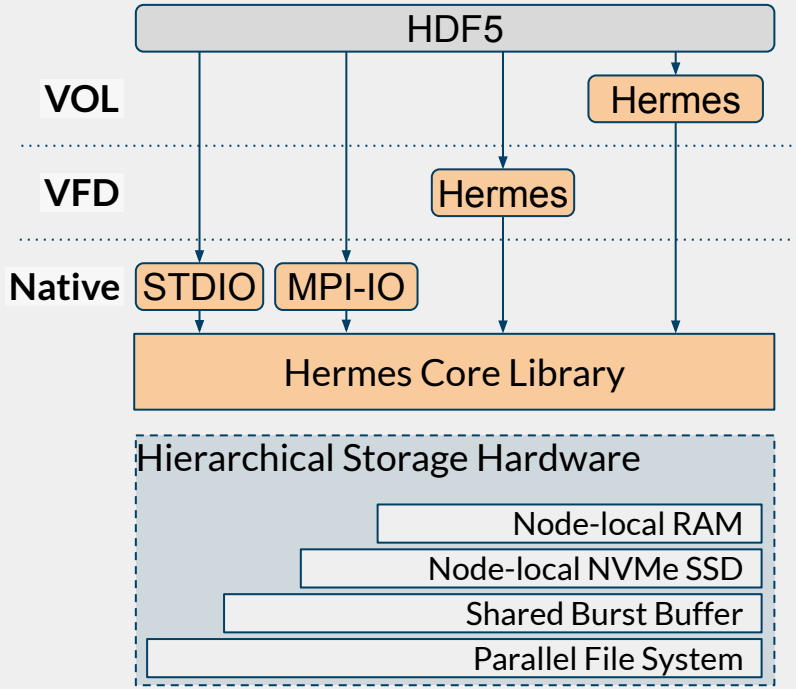


How can I use Hermes?

Hermes Adapter Layer

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

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



Deployment Models

Collocated

- Hermes Core is part of the application.
- Synchronization is managed internally by hermes lib.
- Isolates buffering data across applications.

```
mpirun -n 1280 -f app_hf ./application
```

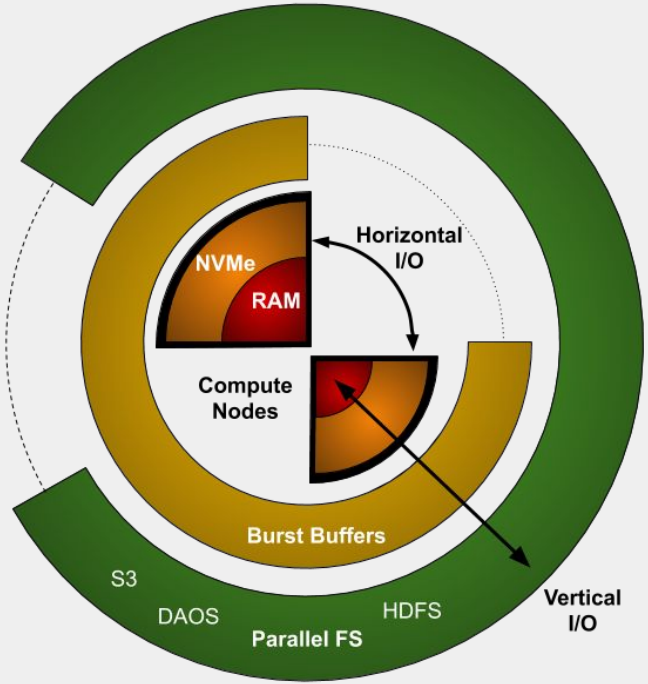
Decoupled

- Hermes Core is separate from the application.
- The Hermes core needs to be running before the application.
  - Manually, or
  - as a service
- Can share buffering data across applications.

```
mpirun -n 32 -f hermes_core_hf ./hermes_core
mpirun -n 1248 -f app_hf ./application
```

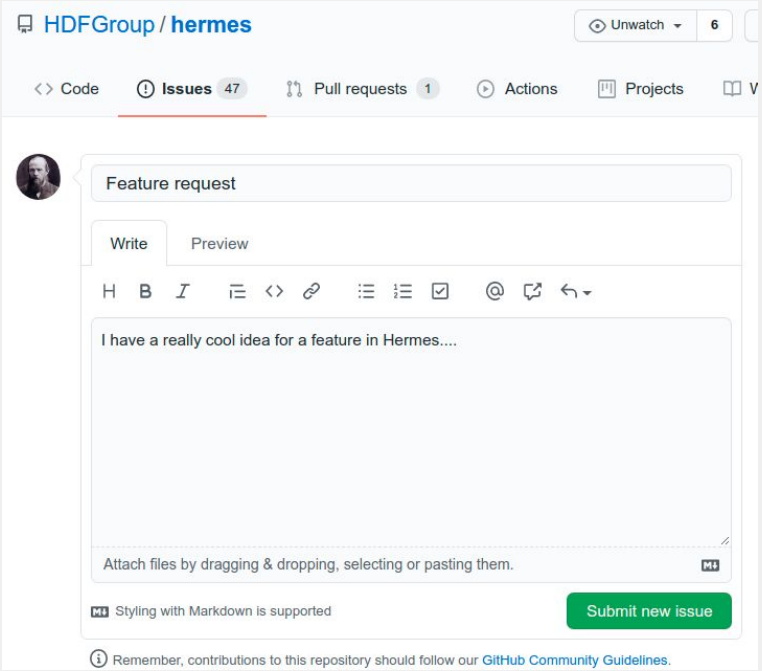
Hermes Container Library (HCL)

- Deep Storage Hierarchy
  - Spans multiple tiers within a node...
  - ...but also multiple nodes in the cluster
- Applications need to distribute data structures across multiple nodes.
  - Hermes Container Library (HCL)
    - H. Devarajan, A. Kougkas, K. Bateman, and X-H Sun. "HCL: Distributing parallel data structures in extreme scales." In 2020 IEEE International Conference on Cluster Computing (CLUSTER).
    - https://github.com/HDFGroup/hcl
  - We invite the community to try it out
    - And please give us feedback.



How Can I Get Involved?

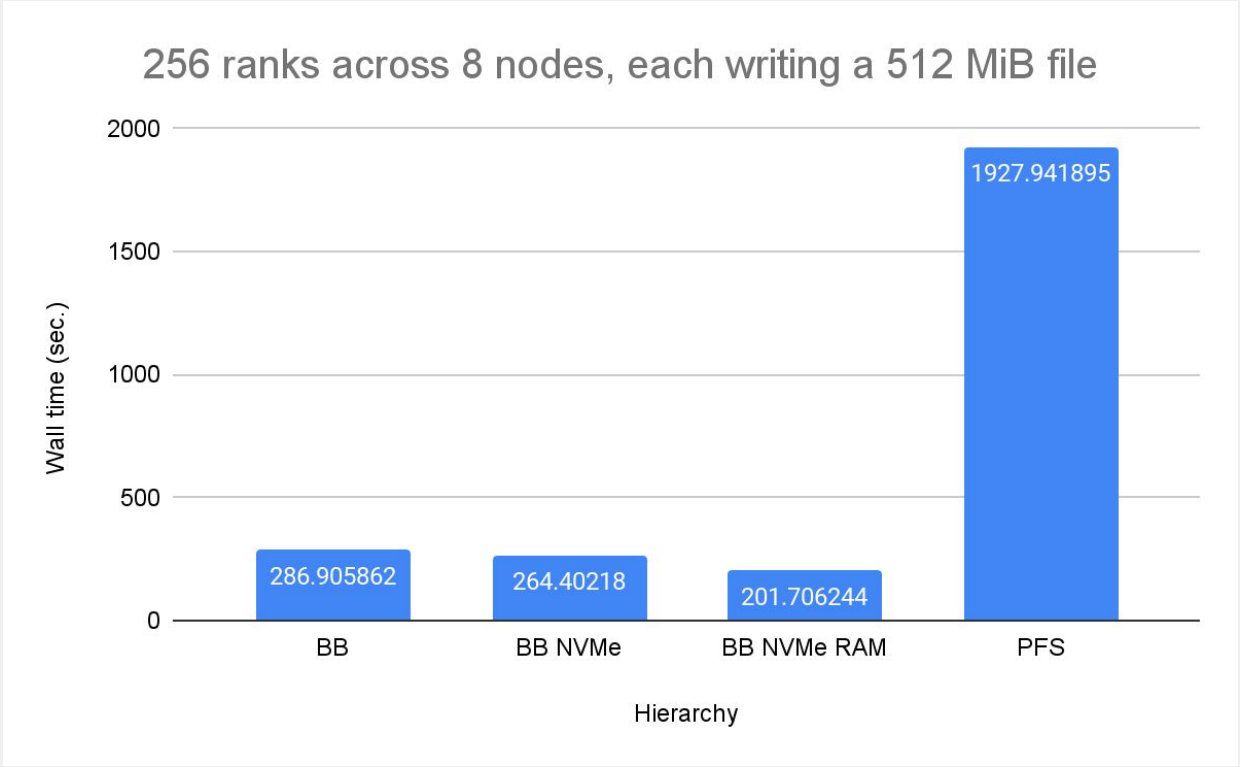
- Github repo: https://github.com/HDFGroup/hermes
- Create an issue to submit feedback, use cases, or feature requests.
- Note: Hermes is still under active development with target beta release this November



Hermes Acceleration for VPIC-style Workload

VPIC-IO

- HDF5 files
- Checkpointing
- File-per-process
- Buffer the intermediate checkpoints and flush at finalize
- Remote global PFS suffers from high latency and low throughput
- Contention across processes



Multi-Tiered
Distributed I/O
Buffering System

Thank you.

Learn more

http://www.cs.iit.edu/~scs/assets/projects/Hermes/Hermes.html

https://github.com/HDFGroup/hermes

https://github.com/HDFGroup/hcl

Contact us

akougkas@iit.edu

gheber@hdfgroup.org

The HDF Group

ILLINOIS INSTITUTE OF TECHNOLOGY