How to leverage multi-tiered storage to accelerate I/O

CS550 – Spring 24

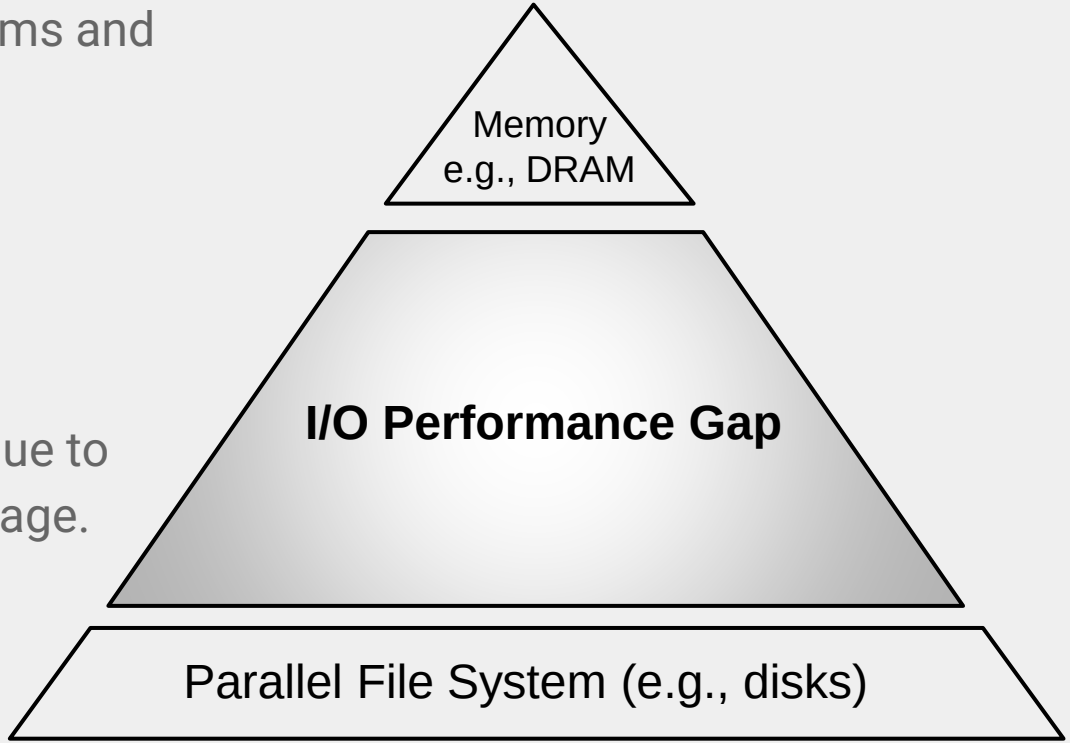
Anthony Kougkas - akougkas@iit.edu

Agenda

- Modern systems and the I/O bottleneck
- Hermes project overview (lecture 1)
- ChronoLog project overview (lecture 2)

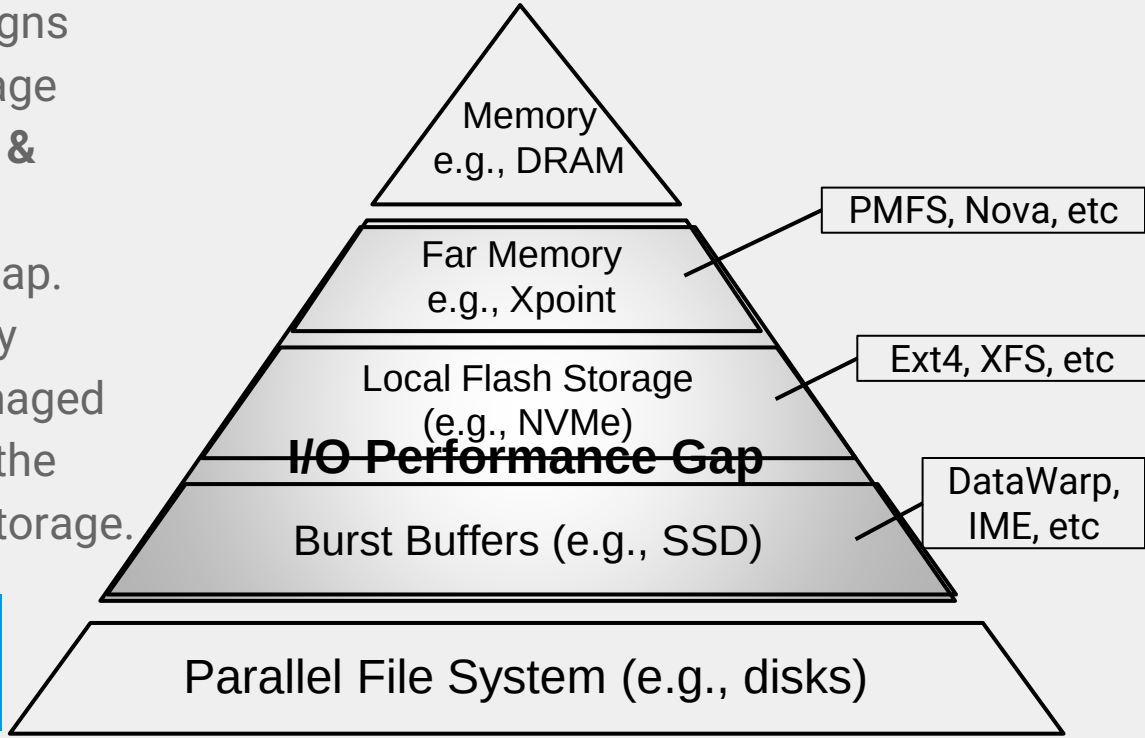
I/O Performance Gap

- 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.



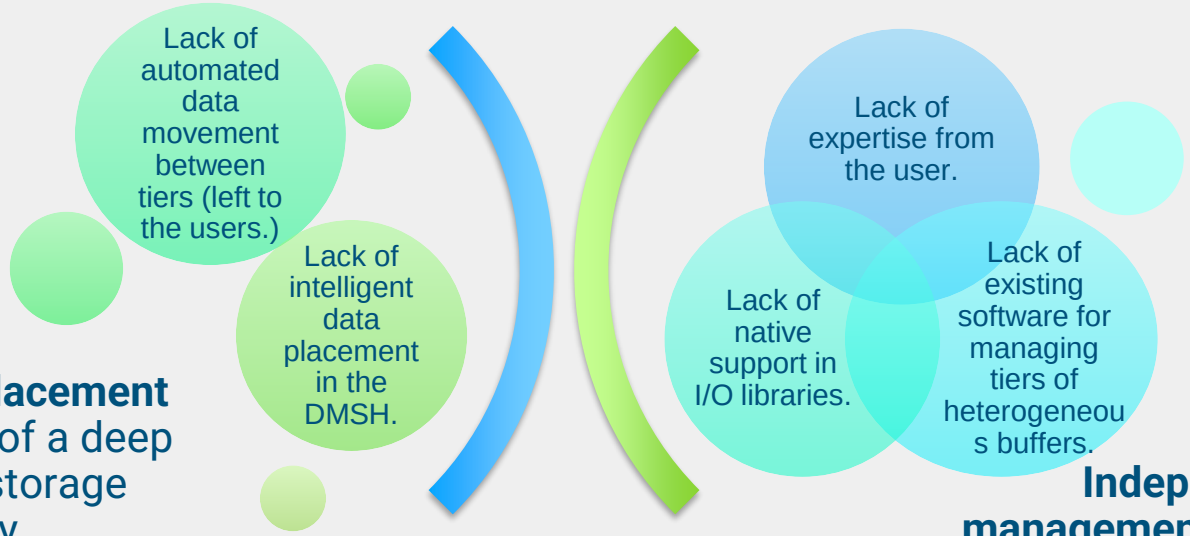
Hardware Answer

- 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**.

Multi-tiered systems today



Complex data placement among the tiers of a deep memory and storage hierarchy

Independent management of each tier of the DMSH

DMSH systems require

efficient and transparent **data movement** through the hierarchy

new data placement algorithms

effective metadata management

an efficient communication fabric

Leveraging multi-tiered storage

NSF OCI-1835764

Hermes:
A Multi-Tiered Distributed I/O Buffering System

NSF CSSI 2104013

ChronoLog:
A Distributed Tiered Shared Log Store

Learn more

- https://grc.iit.edu/research/projects/hermes
- https://grc.iit.edu/research/projects/chronolog
- https://github.com/HDFGroup/hermes
- https://github.com/HDFGroup/hcl
- https://github.com/scs-lab/ChronoLog

Contact us

akougkas@iit.edu

A Multi-Tiered Distributed I/O Buffering System

Anthony Kougkas
akougkas@iit.edu

Hermes Project

The team

Collaborative project
funded by NSF

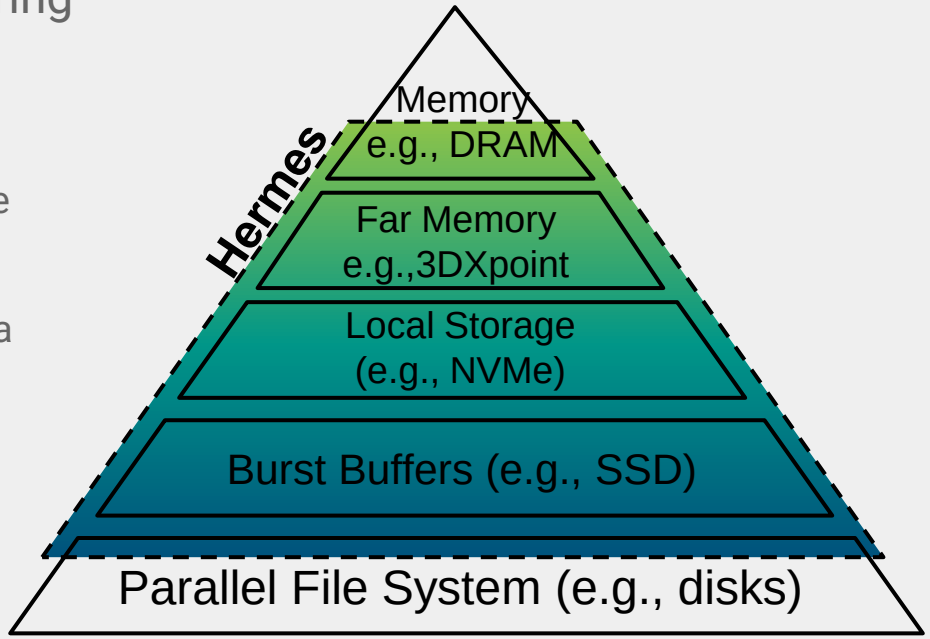
ILLINOIS INSTITUTE OF TECHNOLOGY

The HDF Group

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

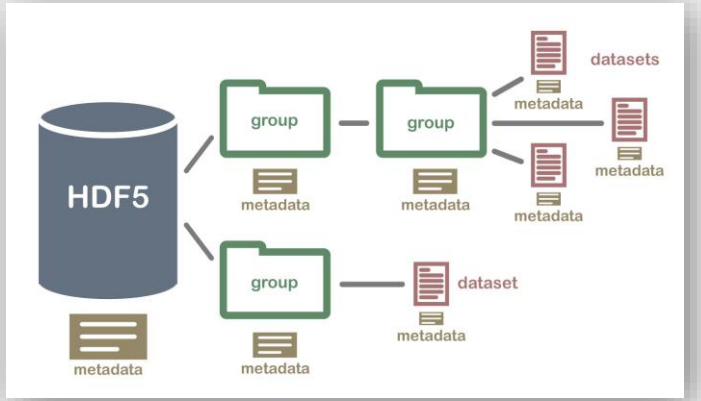
Hermes Overview

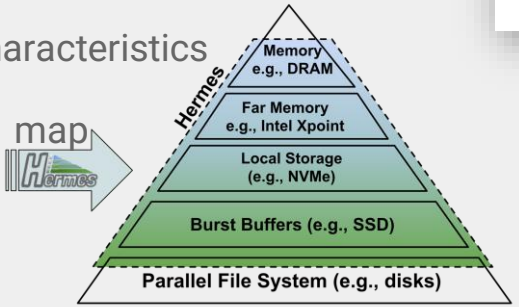
- A new, multi-tiered, distributed buffering system that:
  - Enables, manages, and supervises I/O operations in the Deep Memory & Storage Hierarchy (DMSH).
  - Offers selective and dynamic layered data placement.
  - Is modular, extensible, and performance-oriented.
  - Supports a wide variety of applications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



Hermes + HDF5

- HDF5 is a self-describing hierarchical data format which makes it ideal for Hermes
  - Utilize the rich metadata offered by HDF5 to efficiently place data in the hierarchy.
  - Leverage HDF5 characteristics
    - files,
    - groups,
    - datasets,
    - chunked I/O





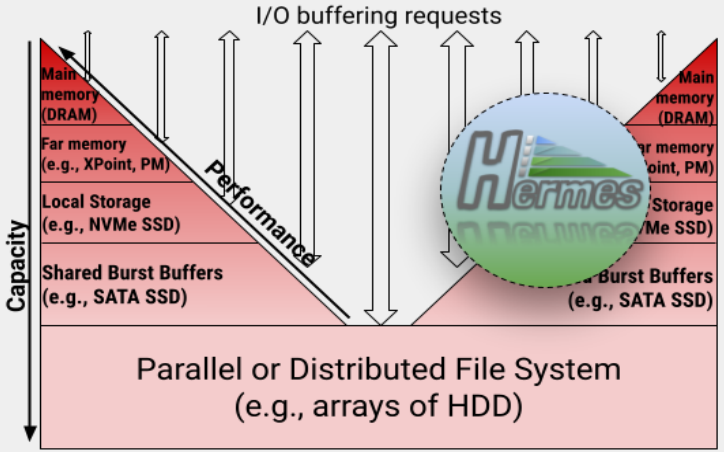
Objectives

- **Seamless Integration**: Ensure Hermes can be easily and transparently integrated into existing scientific workflows, minimizing the need for extensive code modifications.
- **Extended Buffering API**: Provide a rich, yet simple API that offers extensive buffering capabilities, allowing applications to efficiently interact with deep memory and storage hierarchies.
- **Efficient Access to Storage Tiers**: Enable efficient access to all layers of the Deep Memory and Storage Hierarchy (DMSH), ensuring optimal data placement and retrieval.
- **Lightweight and Feature-rich Library**: Develop Hermes to be feature-rich yet maintain a lightweight footprint to ensure high performance without imposing significant overhead on the system.
- **Scalability and Performance**: Demonstrate Hermes' ability to scale efficiently and improve performance in large-scale systems, handling massive data volumes and concurrent I/O operations.
- **Community Engagement and Open Source**: Release Hermes as open-source software under the BSD 3-Clause license to foster collaboration, innovation, and widespread adoption.

Design and Architecture

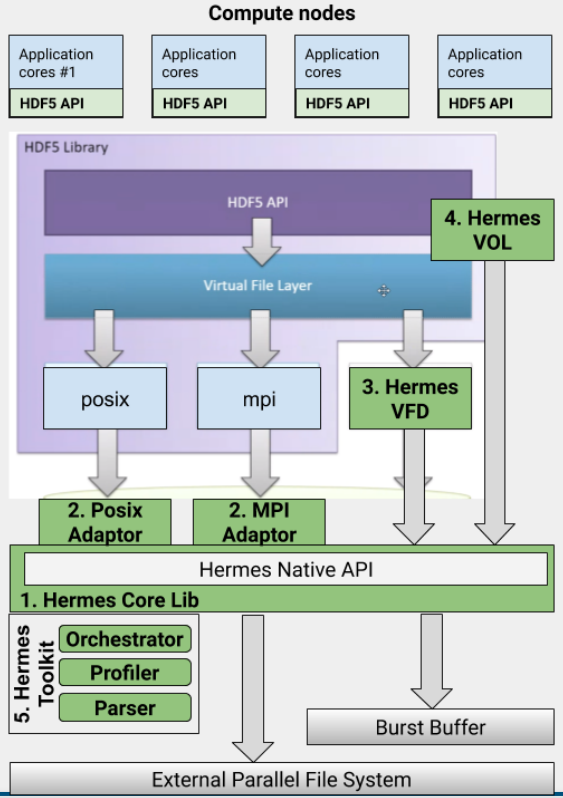
Design Goals

- 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



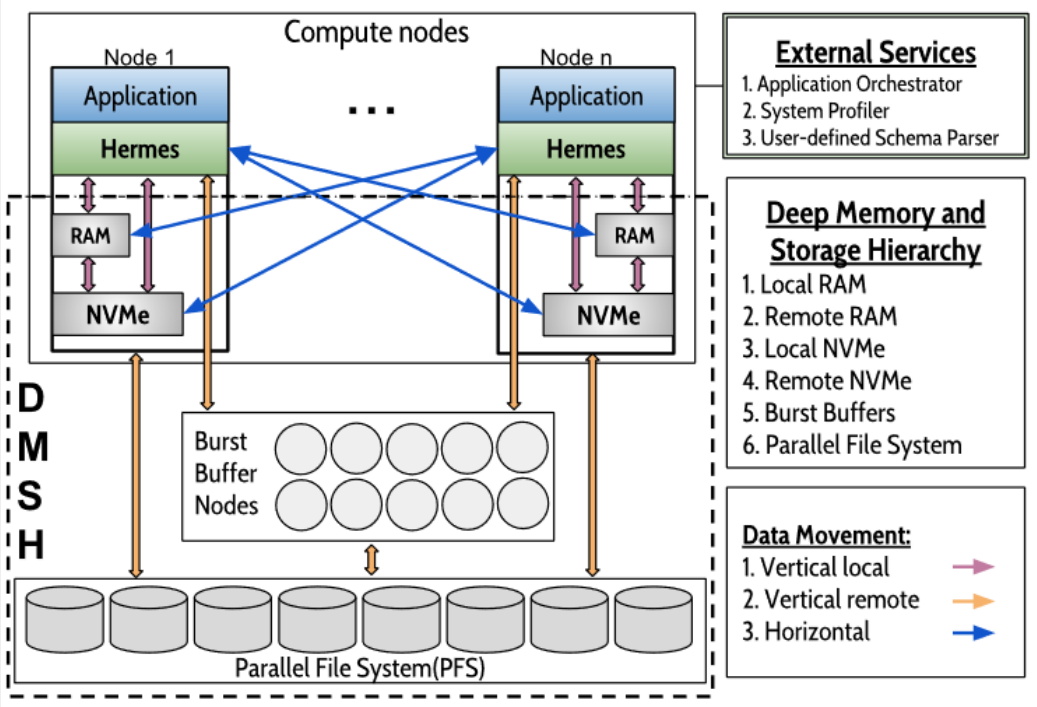
Hermes Ecosystem

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



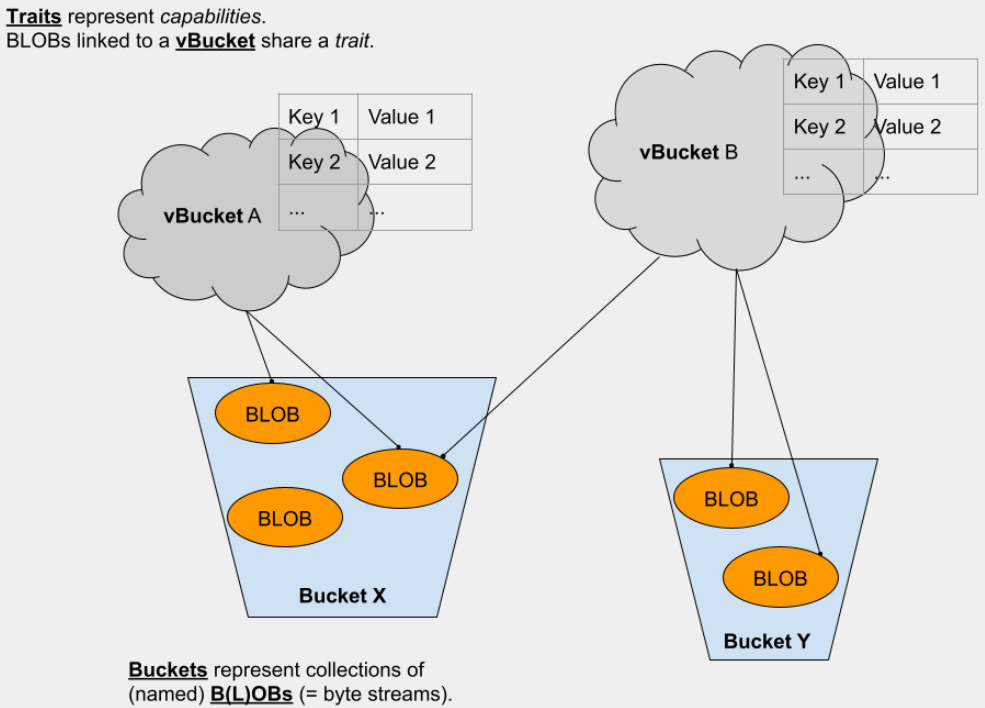
Hermes Architecture

- Hermes machine model:
  - Large amount of RAM
  - Local NVMe and/or SSD device
  - Shared Burst Buffers
  - Remote disk-based PFS
- Hierarchy based on:
  - Access Latency
  - Data Throughput
  - Capacity
- Two data paths:
  - Vertical (within node)
  - Horizontal (across nodes)



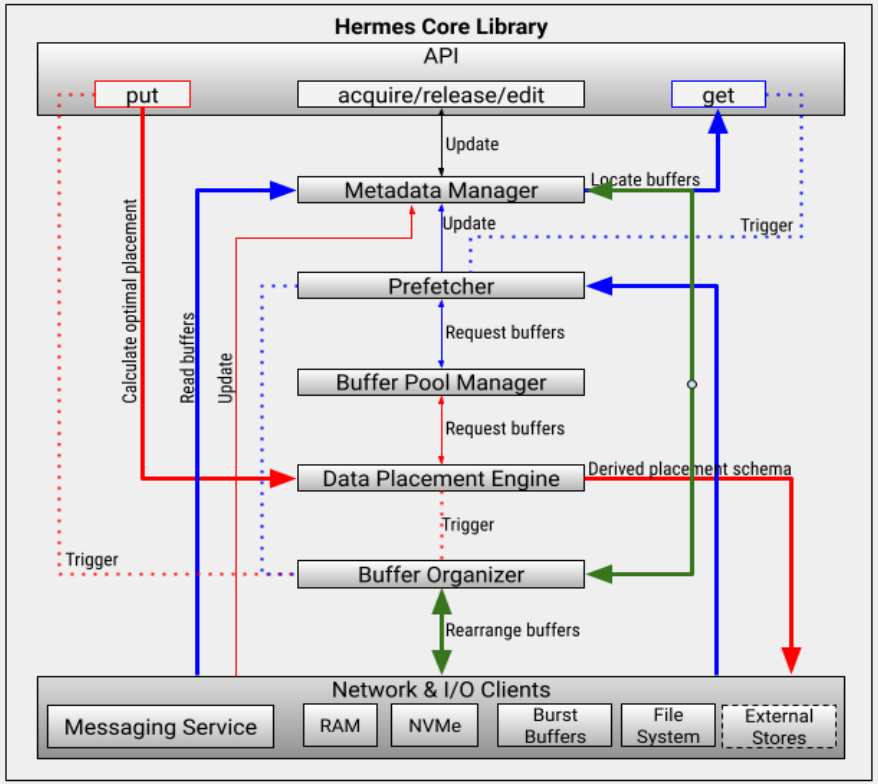
Hermes Data Model

- 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



Hermes Lib Design

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



Hermes Buffering Modes

1

Persistent

- Synchronous
  - write-through cache,
  - stage-in
- Asynchronous
  - write-back cache,
  - stage-out

2

Non-Persistent

- Temporary scratch space
- Intermediate results
- In-situ analysis and visualization

3

Bypass

- Write-around cache

Hermes Data Placement Policies

1

Maximum Application Bandwidth (MaxBW): this policy aims to maximize the bandwidth applications experience when accessing Hermes.

2

Maximum Data Locality: this policy aims to maximize buffer utilization by simultaneously directing I/O to the entire DMSH.

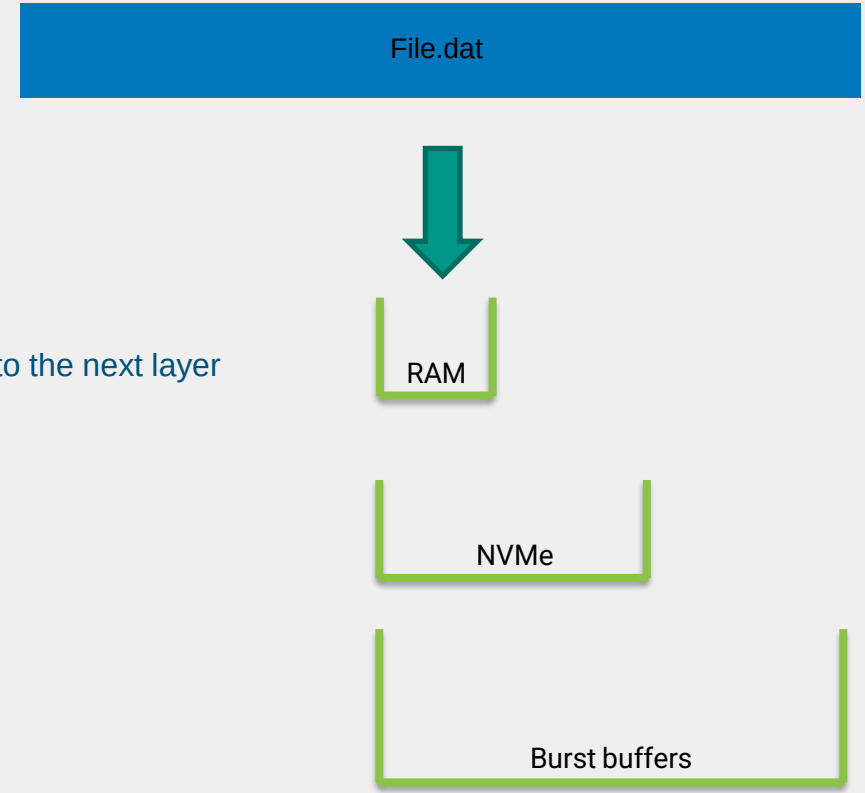
3

Hot-data: this policy aims to offer applications a fast cache for frequently accessed data (i.e., hot-data).

4

User-defined: this policy aims to support user-defined buffering schemas. Users are expected to submit an XML file with their preferred buffering requirements.

1 Maximum Bandwidth



Start from the top layer

If free space > request size

place data here

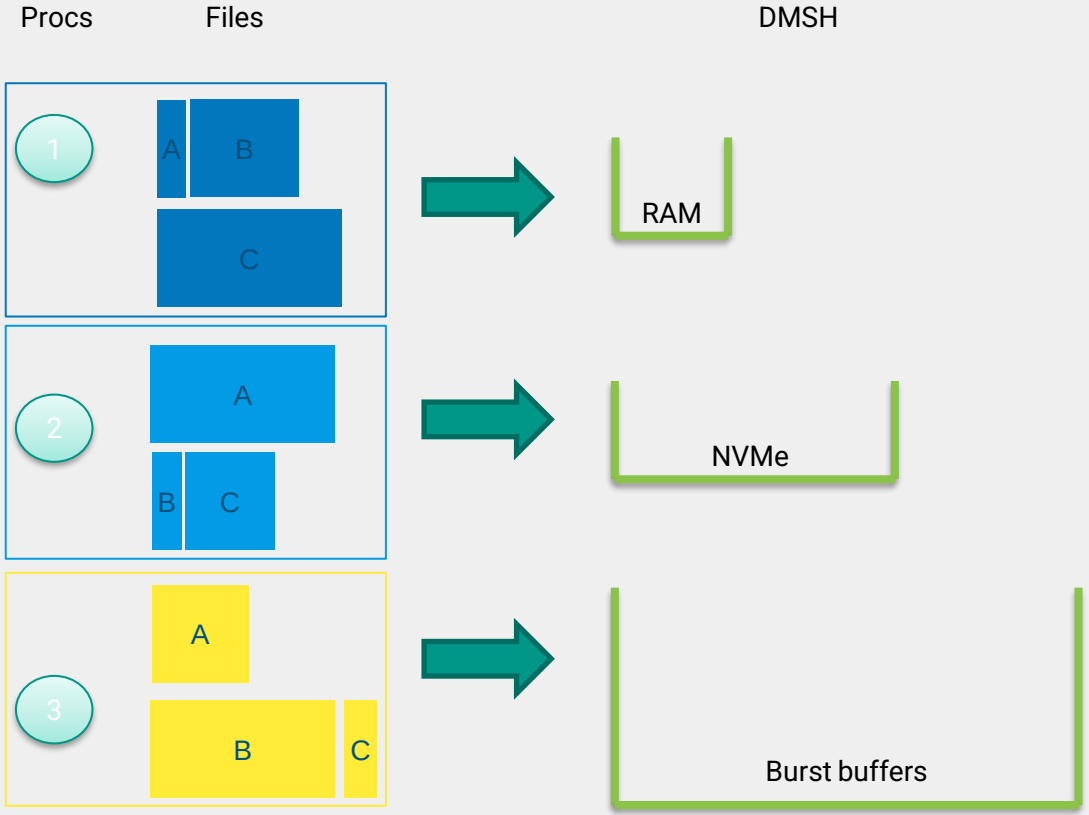
If not, choose the best between

1. Place as much data as possible here and the rest to the next layer OR
2. Skip this layer and place data to the next one OR
3. First flush top layer and then place data

Recursive process

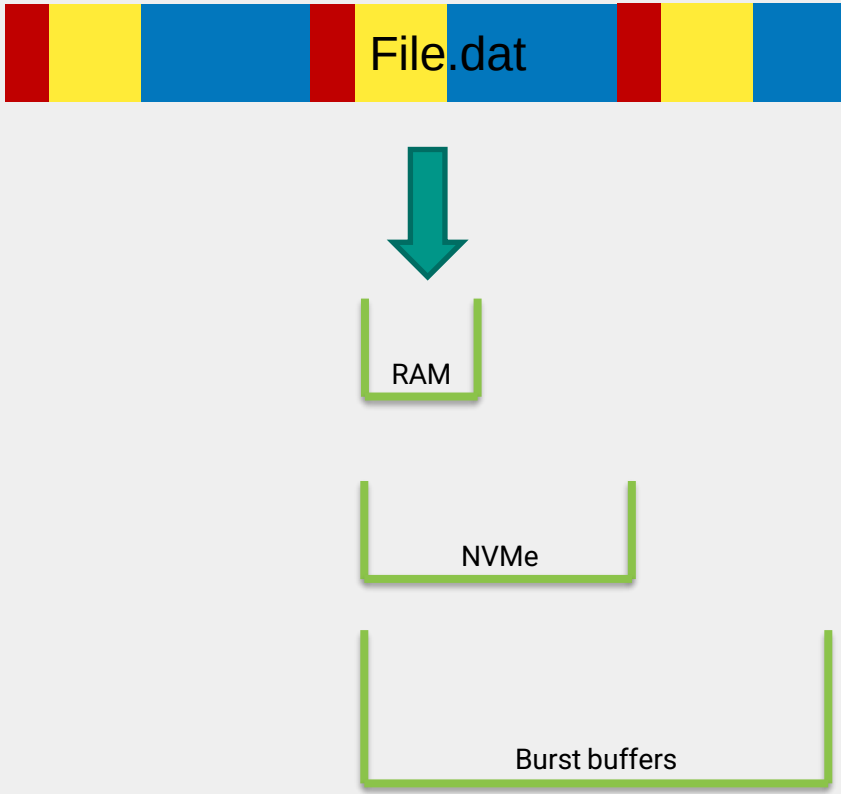
2 Data Locality

- Data dispersion unit:
  - POSIX files
  - HDF5 datasets
  - Etc.
- Place data based on:
  - Location of previously buffered data
  - Ratio between layers



3 Hot data

- Place data based on:
  - Spectrum of hot – cold data
- Higher layers hold hotter data

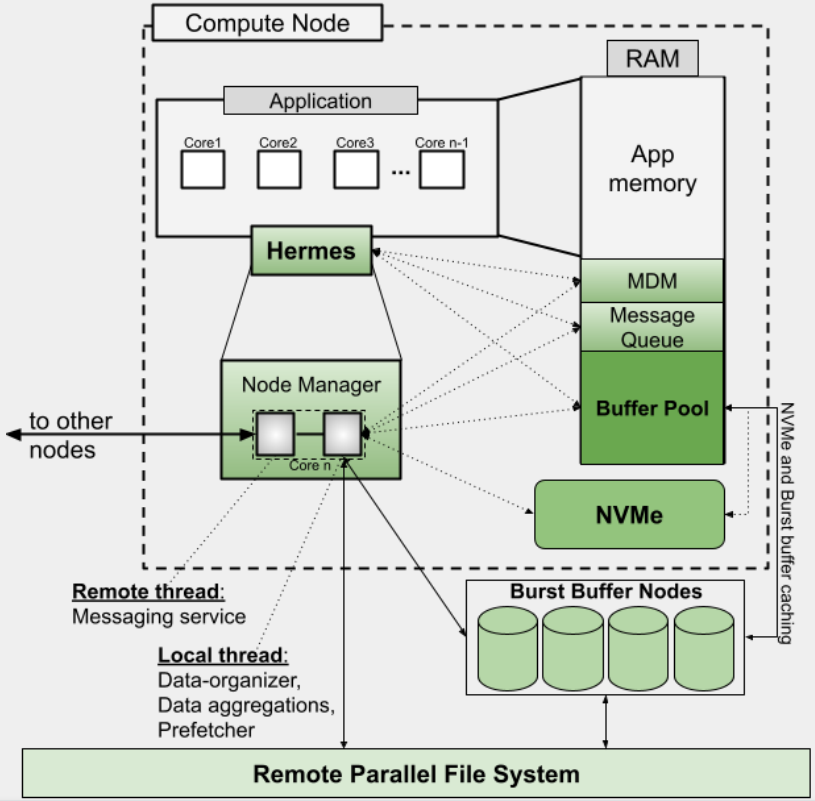


Xian-He Sun, Professor
sun@iit.edu

Deployment

Hermes Node Design

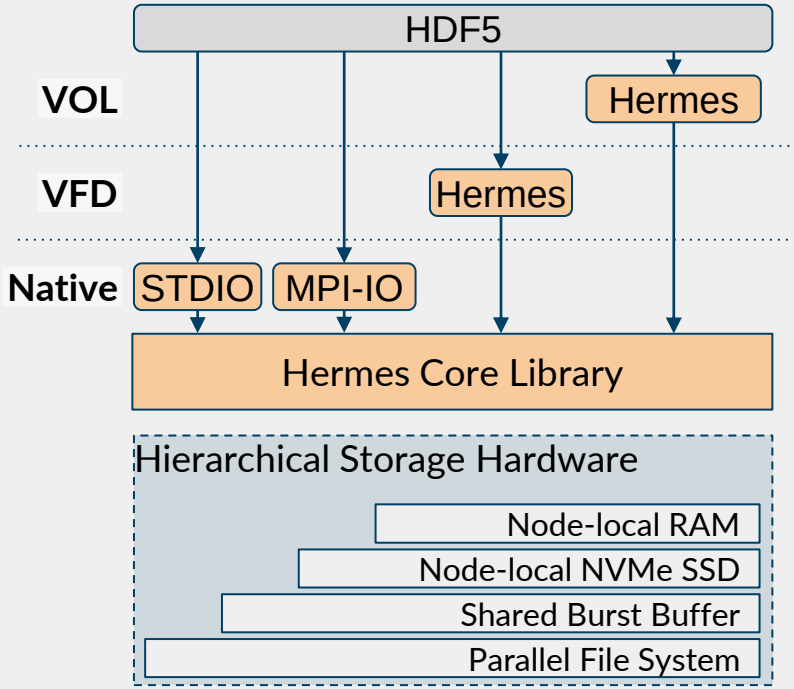
- Dedicated core for Hermes
- Node Manager
  - Dedicated multithreaded core per node
  - MDM
  - Buffer Organizer
  - Messaging Service
  - Memory management
  - Prefetcher
  - Cache manager
- RDMA-capable communication
- Can also be deployed in I/O Forwarding Layer (I/O FL)



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
```

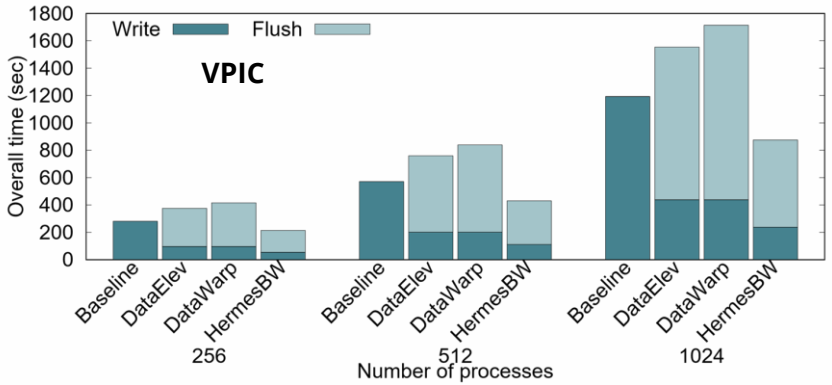
Initial Results

Results Summary

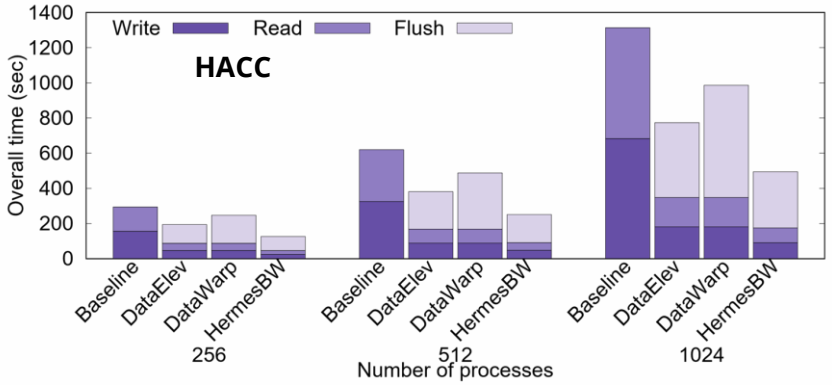
- Hermes has been rigorously tested on large-scale systems, proving its capability to efficiently handle massive data volumes and support high levels of concurrent I/O operations.
  - Demonstrated scalability up to thousands of nodes while maintaining optimal performance and data integrity.
- Comparative benchmarks have shown Hermes to significantly outperform traditional I/O methods across various scientific applications.
  - In the Vector Particle-In-Cell (VPIC) simulation, Hermes achieved up to a 5x increase in write performance and a 7.5x improvement in read performance for repetitive access patterns compared to existing buffering platforms.
- Hermes' intelligent data management strategies, including proactive prefetching and efficient data placement, have drastically reduced I/O latency, making it highly effective for data-intensive computing.
  - Enabled more complex simulations and analyses to be performed in shorter timescales, directly contributing to accelerated scientific discovery.
- The project investigated deep learning (DL) workloads and optimized Hermes to enhance the performance of DL applications in HPC systems.
  - Introduced optimizations such as asynchronous and partial I/O, improving performance by up to 2-10x for applications performing primarily small I/O operations.

Scientific Applications

- Strong scaled up to 1024 ranks
- 16-time steps
- Metric:
  - Total I/O time (write + read + flush)
- Vector Particle-In-Cell (VPIC):
  - Uses HDF5 files
- Hardware Accelerated Cosmology Code (HACC):
  - MPI - I/O Independent



Hermes offers **5x and 2x** higher write performance on average when compared to No Buffering and state-of-the-art buffering platforms



Hermes offers **7.5x and 2x** higher read performance for repetitive patterns when compared to No Buffering and state-of-the-art buffering platforms

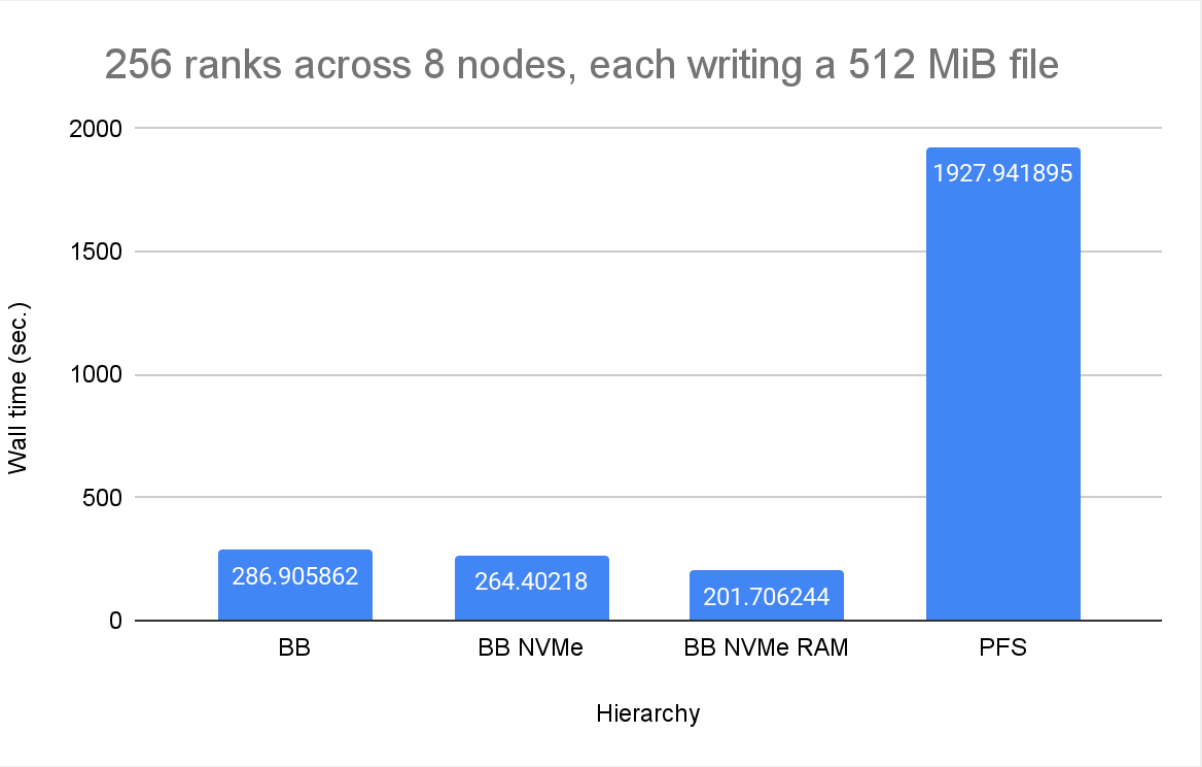
- Hermes hides data movement between tiers behind compute
- Hermes leverages the extra layers of the DMSH to offer higher BW
- Hermes utilizes a concurrent flushing overlapped with compute

Anthony Kougkas, Hariharan Devarajan, and Xian-He Sun. *Hermes: A Heterogeneous-Aware Multi-Tiered Distributed I/O Buffering System,* In Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, pp. 219-230. ACM, 2018.

Hermes Acceleration for VPIC-style Workload

VPIC-IO

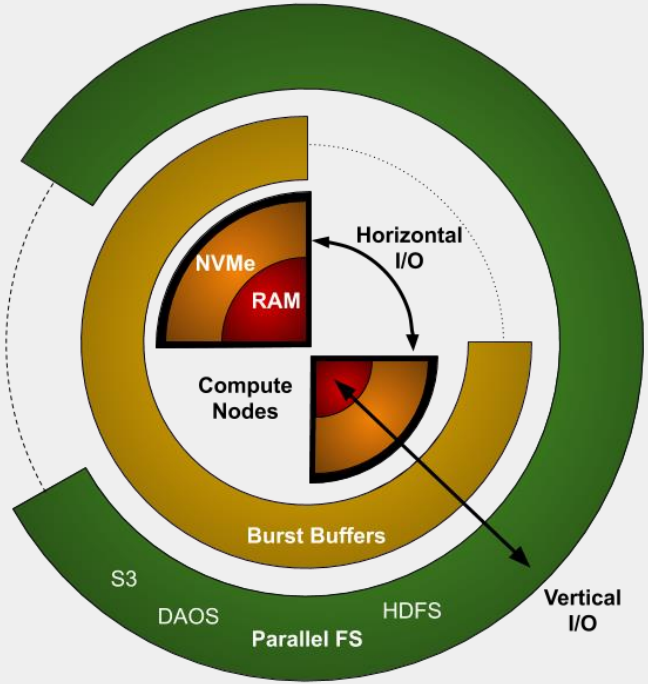
- HDF5 files
- Checkpointing
- File-per-process
- Buffer the intermediate checkpoints and flush at finalize
- Remote global PFS suffers from high latency and low throughput
- Contention across processes



Tools and Services

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.



Hermes External Services

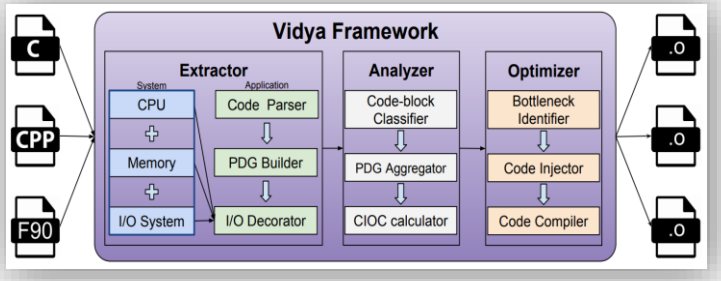
- Application Orchestrator
  - offers support in a multiple-application environment
  - manages access to the shared layers of the hierarchy
  - minimizes interference between different applications sharing a layer
  - coordinates the flushing of the buffers to achieve maximum I/O performance

Anthony Kougkas, Hariharan Devarajan, Xian-He Sun, and Jay Lofstead.
"Harmonia: An Interference-Aware Dynamic I/O Scheduler ",
In Proceedings of the IEEE International Conference on Cluster Computing 2018 (Cluster'18)

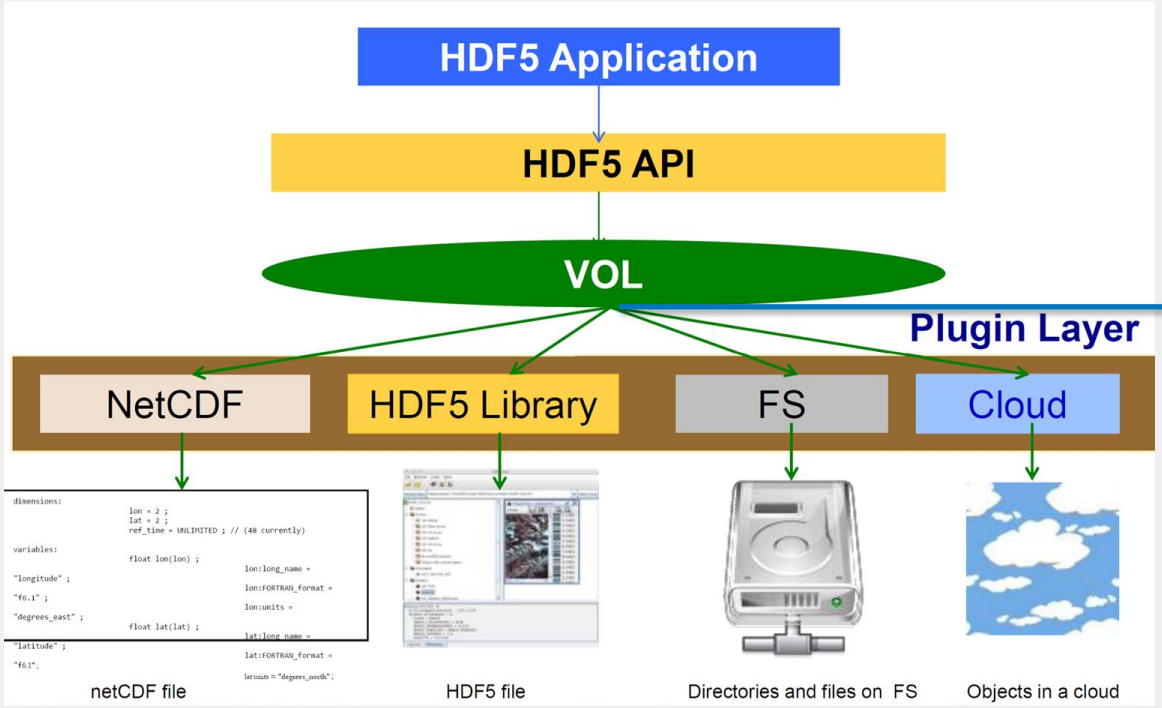
Hermes External Services

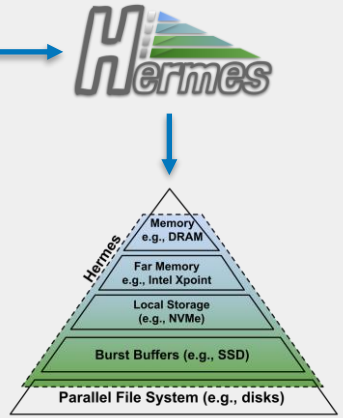
- System Profiler
  - runs once during the initialization
  - performs a profiling of the underlying system in terms of hardware resources
  - detects the availability of DMSH and measures each layer's respective performance
  - profiles the applications and identifies incoming I/O phases
  - works together with the application coordinator (Harmonia) to detect access conflicts

Hariharan Devarajan, Anthony Kougkas, P. Challa, Xian-He Sun
"Vidya: Performing Code-Block I/O Characterization for Data Access Optimization"
In Proceedings of the IEEE International Conference on High Performance Computing, Data, and Analytics 2018 (HiPC'18), Bengaluru, India



Hermes VOL plugin for HDF5 coming…





Lessons Learned

Some Lessons Learned

- Throughout the development and implementation of the Hermes project, several key lessons have been learned that inform future research and development in high-performance computing and data management:
  - **Understanding Workloads**: Deep investigation into deep learning (DL) workloads and their I/O characteristics is crucial. The development of the DLIO benchmark has been instrumental in testing and refining Hermes' capabilities to accelerate such workloads.
  - **The Importance of Resource Monitoring**: Accurate and timely resource monitoring, especially for hierarchical storage resources, is essential for optimizing data placement, prefetching, and tier management. The introduction of Apollo, a scalable resource monitoring service, highlights this need.
  - **Leveraging Modern Hardware**: Utilizing modern hardware APIs can lead to significant performance improvements. Hermes' optimizations, such as Asynchronous and Partial I/O, have showcased performance boosts of up to 2-10x for latency-sensitive applications.
  - **Community Engagement**: Engaging with the broader scientific and research communities through open-source release and active collaboration has amplified Hermes' impact and facilitated real-world application testing and feedback.
- These insights reflect the dynamic interplay between technological innovation and user-centric design, guiding the development team in designing a platform that not only meets current computational demands but is also poised to adapt to future challenges.

Q&A

Q&A

Q: The Data Placement Engine (DPE) policies rely on the fact that users know the behavior of their application in advance which can be a bold assumption.

A: Hermes uses profiling tools beforehand to learn about the application's behavior and thus, tune itself. Our work, Vidya, further solves this issue by automating the whole process analyzing the source code.

Q: How does Hermes integrate to modern HPC environments?

A: As of now, applications link to Hermes (re-compile or dynamic linking). An HDF5 VOL plugin is underway. We also intend to incorporate Hermes to the system scheduler and thus, include buffering resources into batch scheduling.

Q&A

Q: How are Hermes' policies applied in multi-user environments?

A: Hermes' Application Orchestrator is designed for multi-tenant environments. Our work, Harmonia, has been tested and proven it can mitigate the contention between competing applications.

Q: What is the impact of the asynchronous data reorganization?

A: In scenarios where there is some computation in between I/O (i.e., most realistic application workloads), asynchronicity works great.

Q&A

Q: What is Hermes' metadata size?

A: In our evaluation, for 1 million user files, the metadata created by Hermes were 1.1GB.

Q: Is Hermes open source?

A: Yes! The 1st public beta release is scheduled for Nov 1st. We are currently improving the quality of the code and writing documentation.

Q&A

Q: How to balance the data distribution across different compute nodes especially when the I/O load is imbalanced across nodes?

A: Hermes' System Profiler provides the current status of the system (i.e., remaining capacity, etc) and DPE is aware of this before it places data in the DMSH. It is up to Hermes' Engine to balance the load.

Q: How to minimize extra network traffic caused by horizontal data movement?

A: Horizontal data movement can be in the way of the normal compute traffic. RDMA capable machines can help. We also suggest using the "*service class*" of the Infiniband network to apply priorities in the network.

Q&A

Q: How is the limited RAM space partitioned between applications and Hermes?

A: Totally configurable by the user. Typical trade-off. More RAM to Hermes can lead to higher performance. No RAM means skip the layer.

Q: What is Hermes' DPE complexity?

A: In the order of number of tiers.

Q&A

Q: How difficult is to tune Hermes' configuration parameters?

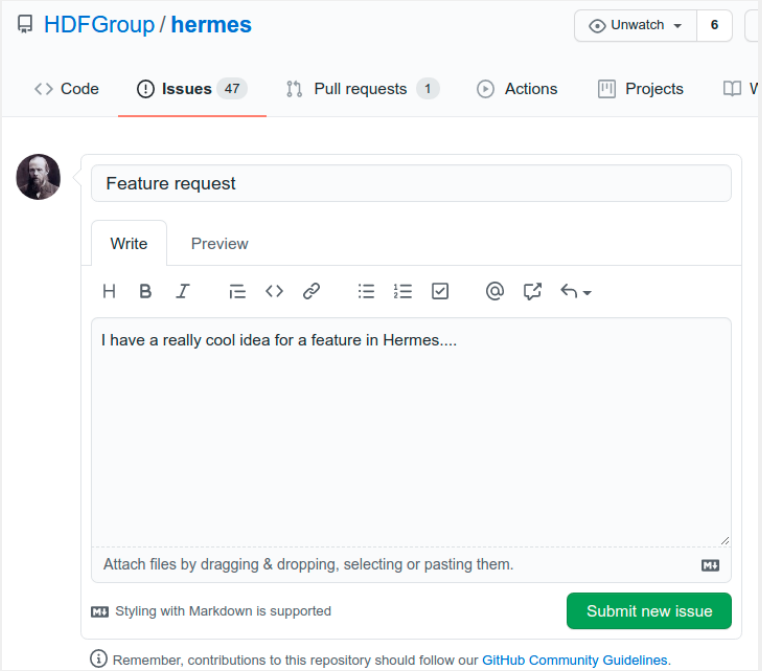
A: We expose a configuration_manager class which is used to pass several Hermes' configuration parameters. ML-assisted tuner is planned to be added.

Q: What is Hermes API?

A: Hermes captures existing I/O calls. Our own API is really simple consisting of hermes::get(…, flags) and hermes::put(…,flags). Flag system implements active buffering semantics (currently only for the burst buffer nodes).

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



Publications

Publications

- Conferences:
  - Kougkas, A; Devarajan, H; and Sun X-H. "*Hermes: a heterogeneous-aware multi-tiered distributed I/O buffering system*." In Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, pp. 219-230. 2018.
  - Devarajan, H; Kougkas, A; Bateman, K; Sun, X.-H. "*HCL: Distributing Parallel Data Structures in Extreme Scales*," 2020 IEEE International Conference on Cluster Computing (CLUSTER),
  - Devarajan, H; Kougkas, A; Logan, L; and Sun, X.-H. "*HCompress: Hierarchical Data Compression for Multi-Tiered Storage Environments,*" 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, 2020,
  - Devarajan, H; Kougkas, A; Sun, X.-H. "*HFetch: Hierarchical Data Prefetching for Scientific Workflows in Multi-Tiered Storage Environments*," 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, 2020,
  - Devarajan, H; Kougkas, A; Sun, X.-H. "*HReplica: A Dynamic Data Replication Engine with Adaptive Compression for Multi-Tiered Storage*," 2020 IEEE International Conference on Big Data (Big Data)

Publications

- Journals:
  - Kougkas, A; Devarajan, H; Sun, X.-H., "*I/O Acceleration via Multi-Tiered Data Buffering and Prefetching*", In Journal of Computer Science and Technology (JCST) 2020. vol 35. no 1. pp 92-120. 10.1007/s11390-020-9781-1
  - Kougkas, A; Devarajan, H; Sun, X.-H., "*Bridging Storage Semantics using Data Labels and Asynchronous I/O*", in Transactions on Storage (TOS), Vol 16, No 4, Article 22, 2020. DOI:https://doi.org/10.1145/3415579

Multi-Tiered
Distributed I/O
Buffering System

Questions?

Anthony Kougkas
akougkas@iit.edu

SCALABLE COMPUTING
SOFTWARE LABORATORY

ILLINOIS INSTITUTE
OF TECHNOLOGY