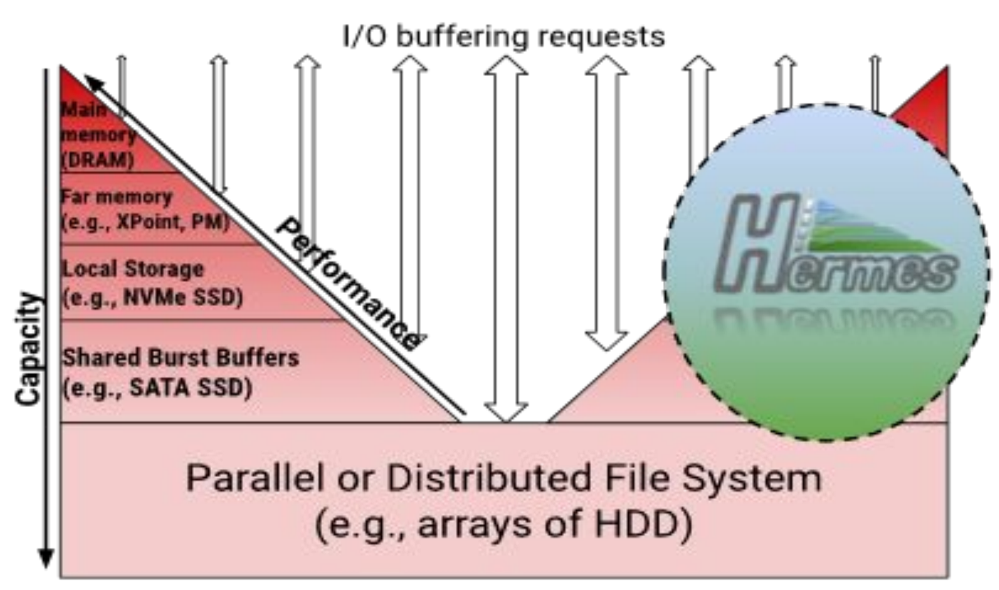
Accelerating I/O-Intensive Applications Through Multi-Tiered Buffering with Hermes

SCALABLE COMPUTING SOFTWARE LABORATORY
Luke Logan, Xian-He Sun, Anthony Kougkas
llogan@hawk.iit.edu, {sun, akougkas}@iit.edu
ILLINOIS INSTITUTE OF TECHNOLOGY
The HDF Group

1. Multi-Tiered Storage



* Many HPC sites have non-volatile burst buffers between memory and disk
* **Deep Memory and Storage Hierarchy (DMSH)**

2. Current Situation

Placing data in DMSH is currently left to users

Poor data placement decisions lead to I/O stalls

* A need for automatic and intelligent data placement

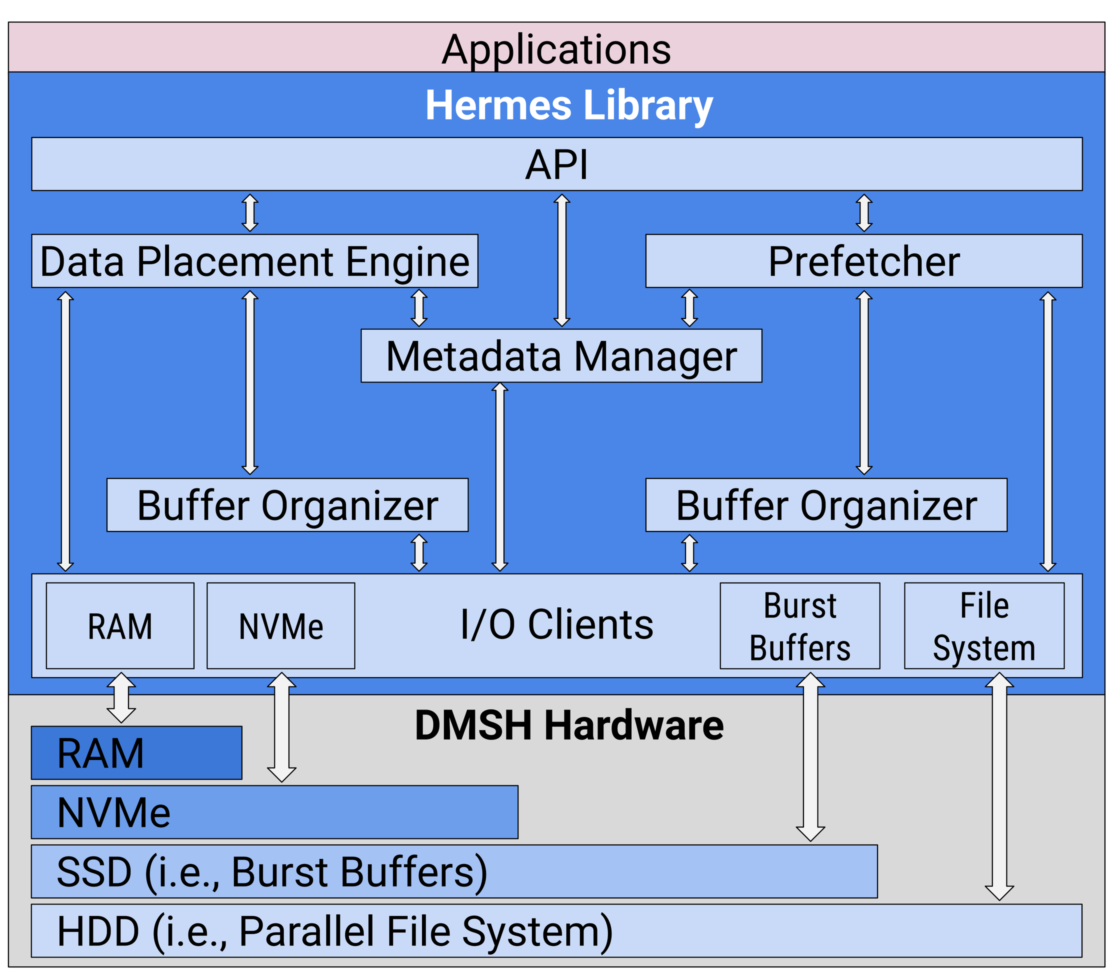
Domain scientists are not I/O experts

HDF5 does not support I/O buffering natively

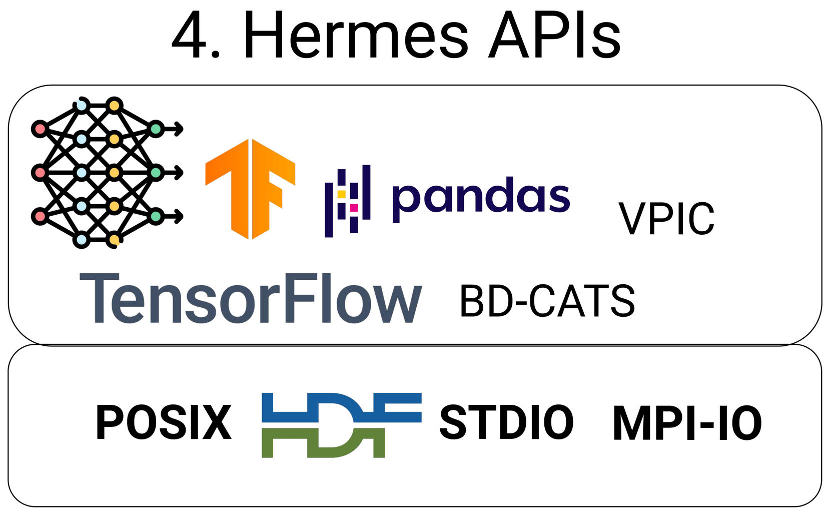
Each tier has different characteristics and interfaces

* Should transparently manage data placement for each independent tier

3. Overview



4. Hermes APIs



* Hermes exposes a Put / Get API to store data "blobs"
* Various adapters transparently convert I/O into blobs
* Supports HDF5, POSIX, STDIO, MPI-IO
* No application changes

5. Data Placement Engine

* Decide where to initially place data
* Can be used to improve write performance
* Three policies currently implemented
* Custom policies can be built using buffer schema

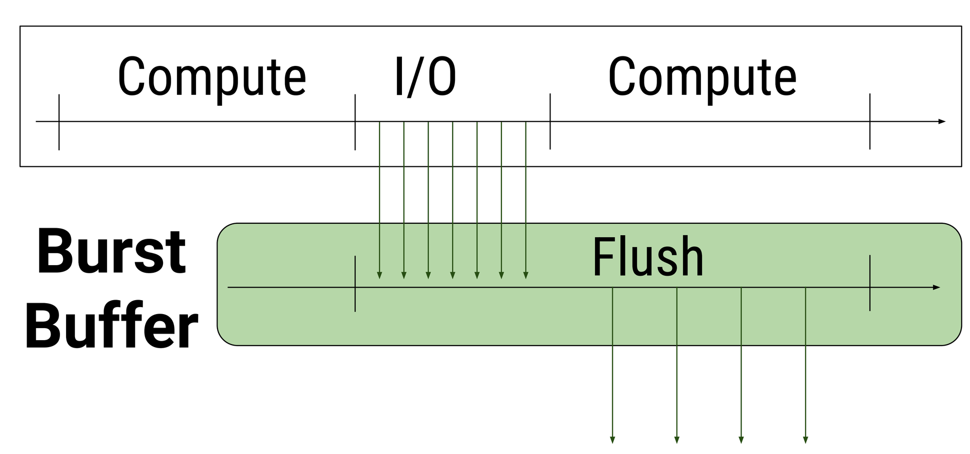
MaxBandwidth
Data placed in fastest tier with enough capacity

RoundRobin
Data is distributed evenly among tiers with enough capacity

Random
Data is randomly sent to tiers with enough capacity

6. Buffer Organizer

Adjusts the position of blobs in the hierarchy asynchronously based on the blob's score



An example of flushing blobs during compute

Promote — Higher score -> faster tier

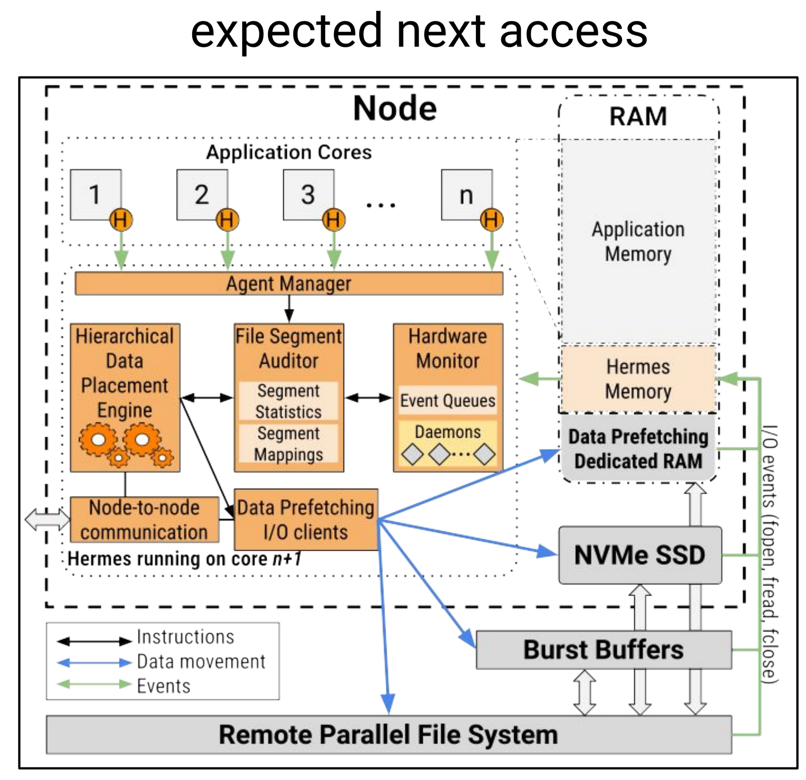
Demote — Lower score -> slower tier

7. Buffer Organizer Blob Scoring

Access Frequency | Access Recency | Blob Size | Blob Tags

8. Prefetcher

Changes the scores of blobs depending on their expected next access



* Many workloads are predictable in their I/O patterns (e.g., deep learning randomness seeds)
* Prefetcher thread is periodically called to update blob scores
* Hermes I/O events are stored in a log, which the prefetchers can analyze

8. Metadata Manager

* Adapter-specific information (e.g., what files should Hermes flush data to before exiting?)
* Internal metadata (e.g., map blobs to hardware locations)

9. Data Staging

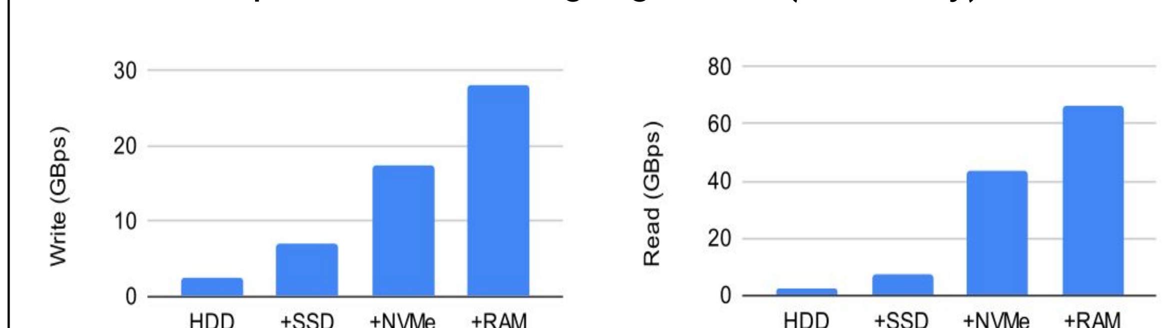
* Import large datasets into Hermes
* Export large datasets from Hermes to a backend
* Shifts the burden of synchronization, aggregation, and processing from the PFS

10. Testbed

CPU	Nodes	Network	Memory + Storage
2.2GHz Xeon Scalable Silver 4114, 48 cores per node	16 nodes	40Gbps Ethernet with RoCE support	* 48G DDR4-2400 * 200GB NVMe * 200GB SSD * 600GB HDD

11. Scientific Simulation Workflow

* VPIC: particle-in-cell simulation code for modeling 3D kinetic plasmas (write-only)
* BD-CATS: particle clustering algorithm (read-only)



Setup
* VPIC writes data
* Data kept in Hermes
* BD-CATS then runs clustering
* 128GB of data per checkpoint
* 8 checkpoints
* 16 nodes, 768 processes
* MaxBandwidth DPE

Analysis
* Adding RAM + NVMe 30-50x faster than using only HDD
* Data effectively buffered
* Can utilize burst buffers to optimize data-intensive workflow stages

12. Conclusion

* Designed / implemented Hermes, an intelligent and transparent I/O buffering system
* Demonstrated the importance of intelligent buffering on scientific workflows

13. Ongoing

* Currently working with application domain scientists to evaluate Hermes for more workloads
* Large-scale evaluations
* Identify opportunities for workload-specific optimization

GitHub

Website