Gnosis Research Center @ ILLINOIS TECH

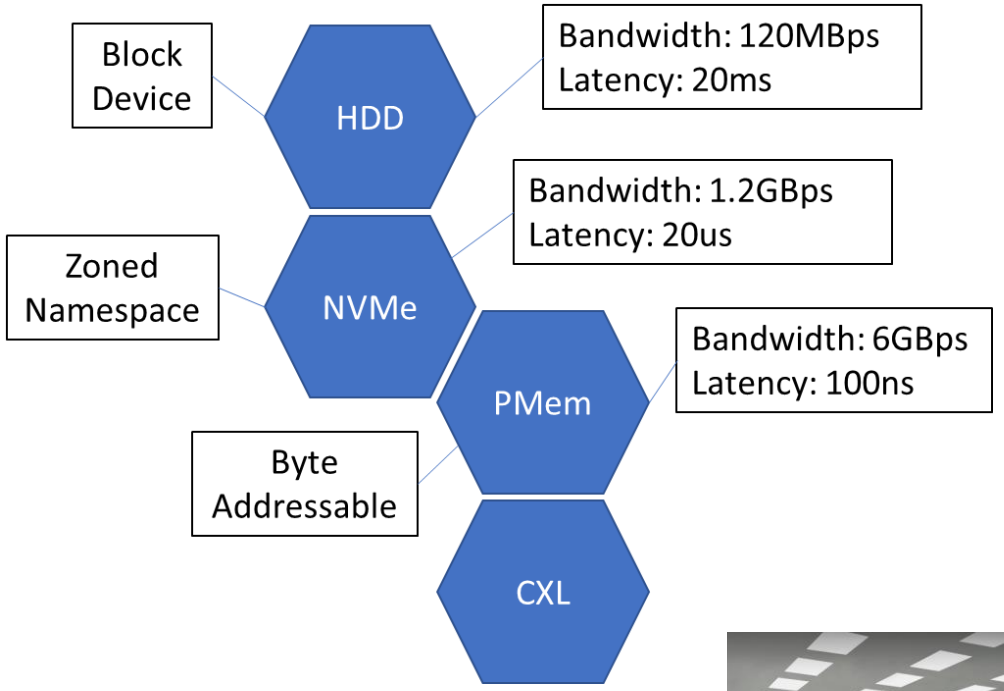
Hermes: A Multi-Tiered Distributed I/O Buffering System

Luke Logan, Anthony Kougkas, Xian-He Sun

llogan@hawk.iit.edu, akougkas@iit.edu, sun@iit.edu

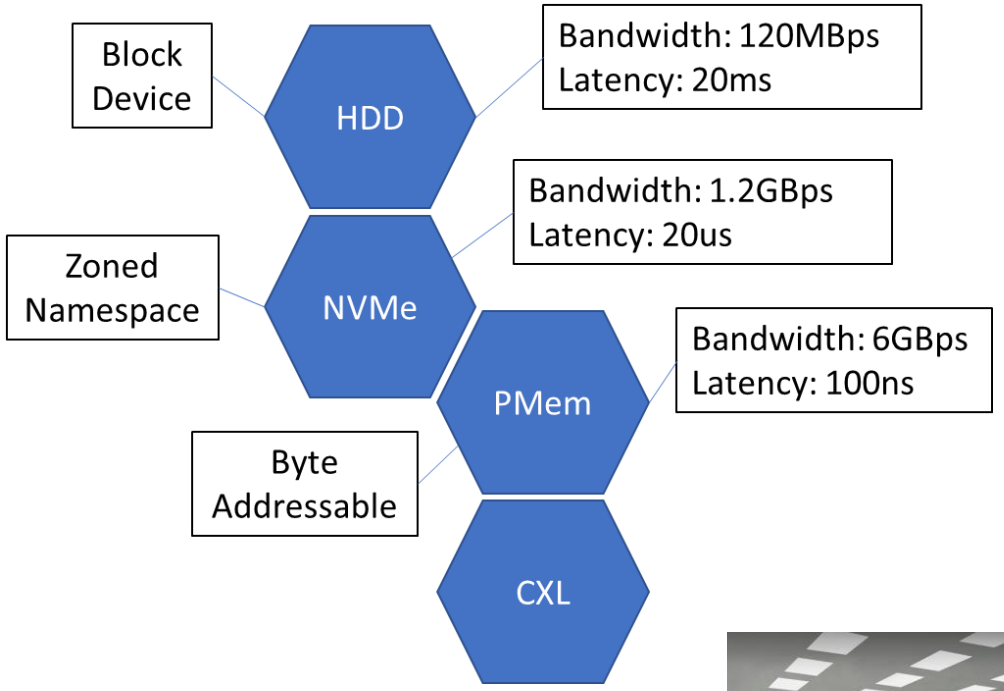
Gnosis Research Center @ Illinois Tech

Rapid Hardware Evolution



- Storage has become **complex**
- **Many different types of hardware** emerged or currently emerging
- Machines integrating **several at once**
  - E.g., El Capitan & Aurora

Deciding Where To Put Data is a Challenge!

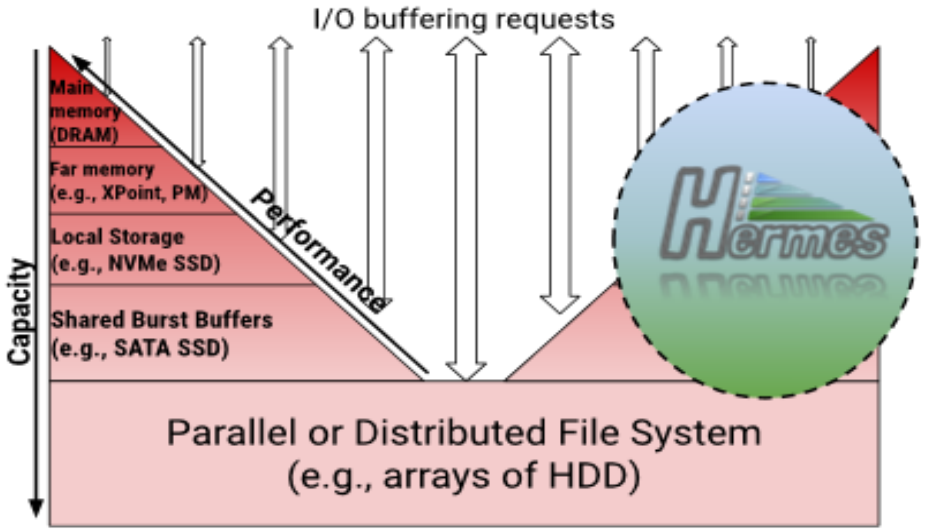


1. HDF5 does not support I/O buffering natively
2. Data placement is left to the user
3. Domain scientists are not I/O experts
4. Poor data placements lead to I/O stalls

Tiered I/O Must Become Simpler

Remove the responsibility of tiered data placement from users!

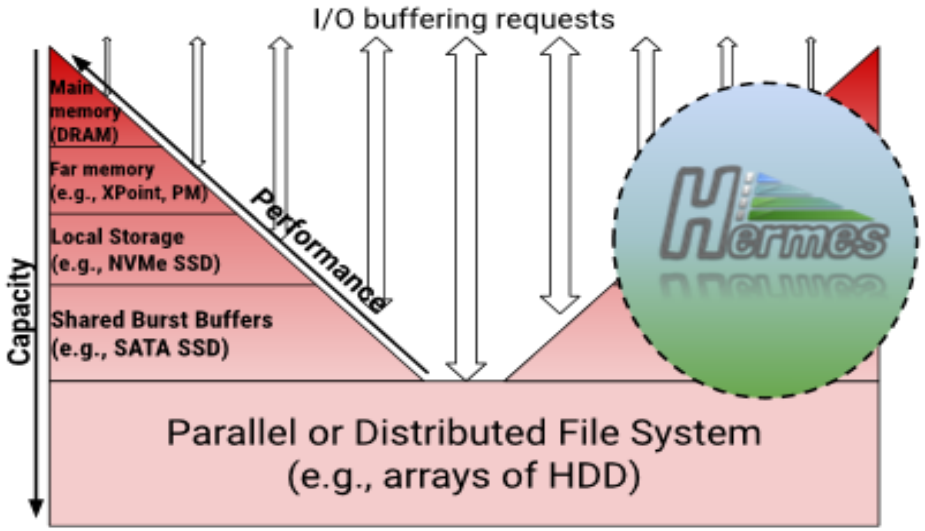
1. Intelligently decide where to place data based on device & application characteristics
2. Correct the placement of data dynamically to adapt to application behavior
3. Support a variety of applications without requiring code changes



Hermes: A Multi-Tiered Buffering System

Remove the responsibility of tiered data placement from users!

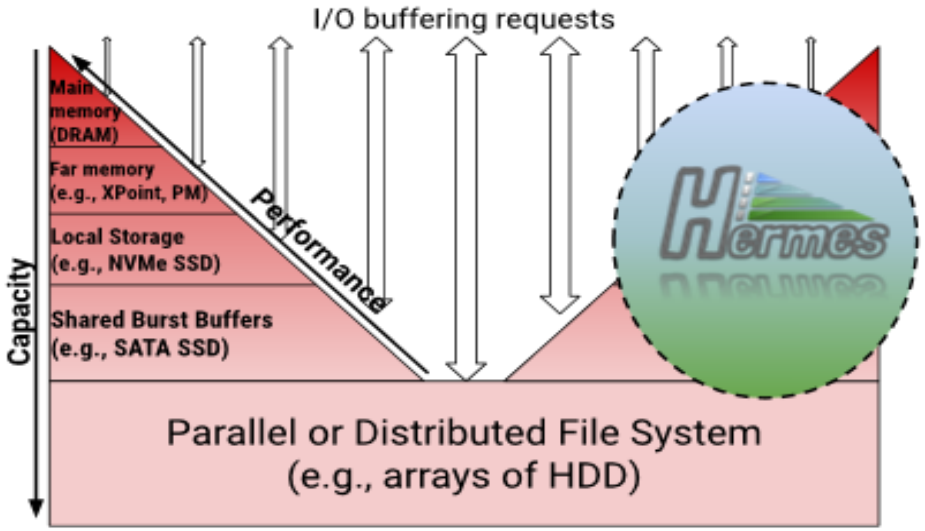
1. Intelligently decide where to place data based on device & application characteristics
   - Data Placement Engines
2. Correct the placement of data dynamically to adapt to application behavior
   - Buffer Organization Polices
3. Support a variety of applications without requiring code changes
   - Adapters



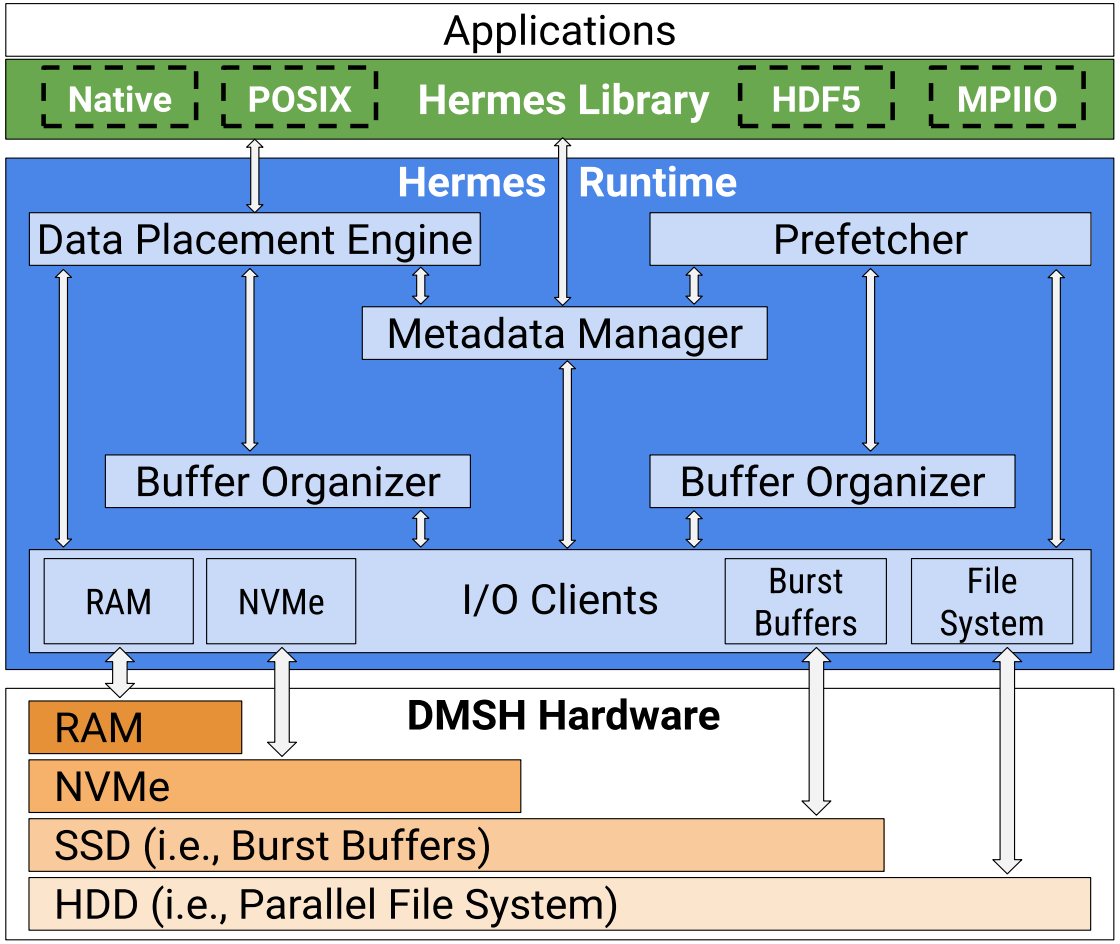
Hermes: A Multi-Tiered Buffering System

Remove the responsibility of tiered data placement from users!

1. Hermes is a multi-million dollar NSF project between HDFGroup & Gnosis@Illinois Tech
2. Hermes version 1.2 is available
3. This talk will discuss the core design of Hermes, recent feature additions, and various use-cases

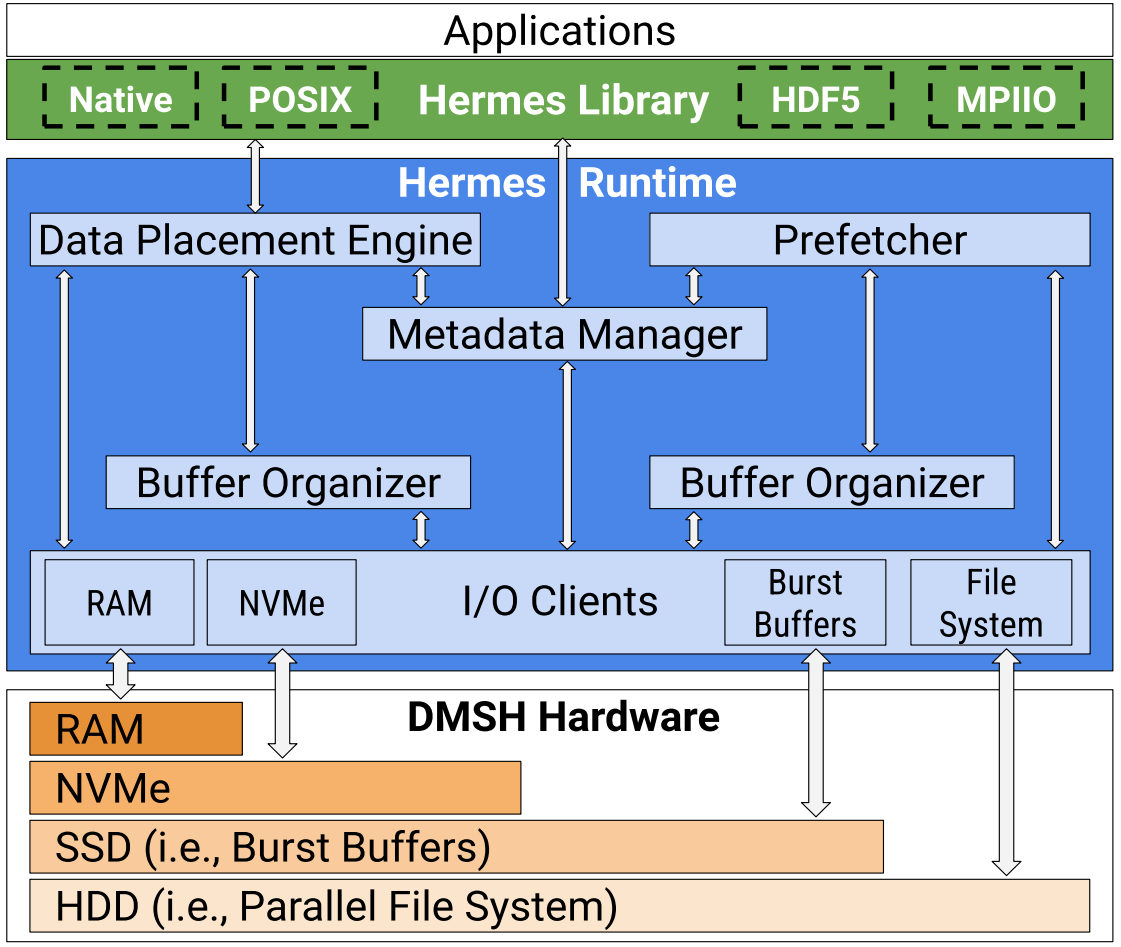


Design Overview



General Use Case

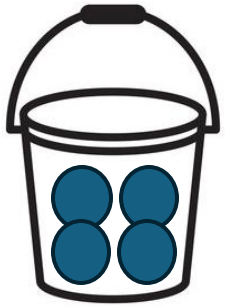
- Hermes runs within the context of an HPC job
- Applications can buffer data within Hermes during the job
- At the end of the job, Hermes flushes all data to the PFS
- During the job, Hermes asynchronously moves data to the PFS to make this flushing faster



Native API

- Applications can call the native Hermes API to interact with data
- Hermes has two primary data types:

Blob
An array of bytes



Bucket
A named collection of blobs

Analogous to a key-value store API (PutBlob & GetBlob)

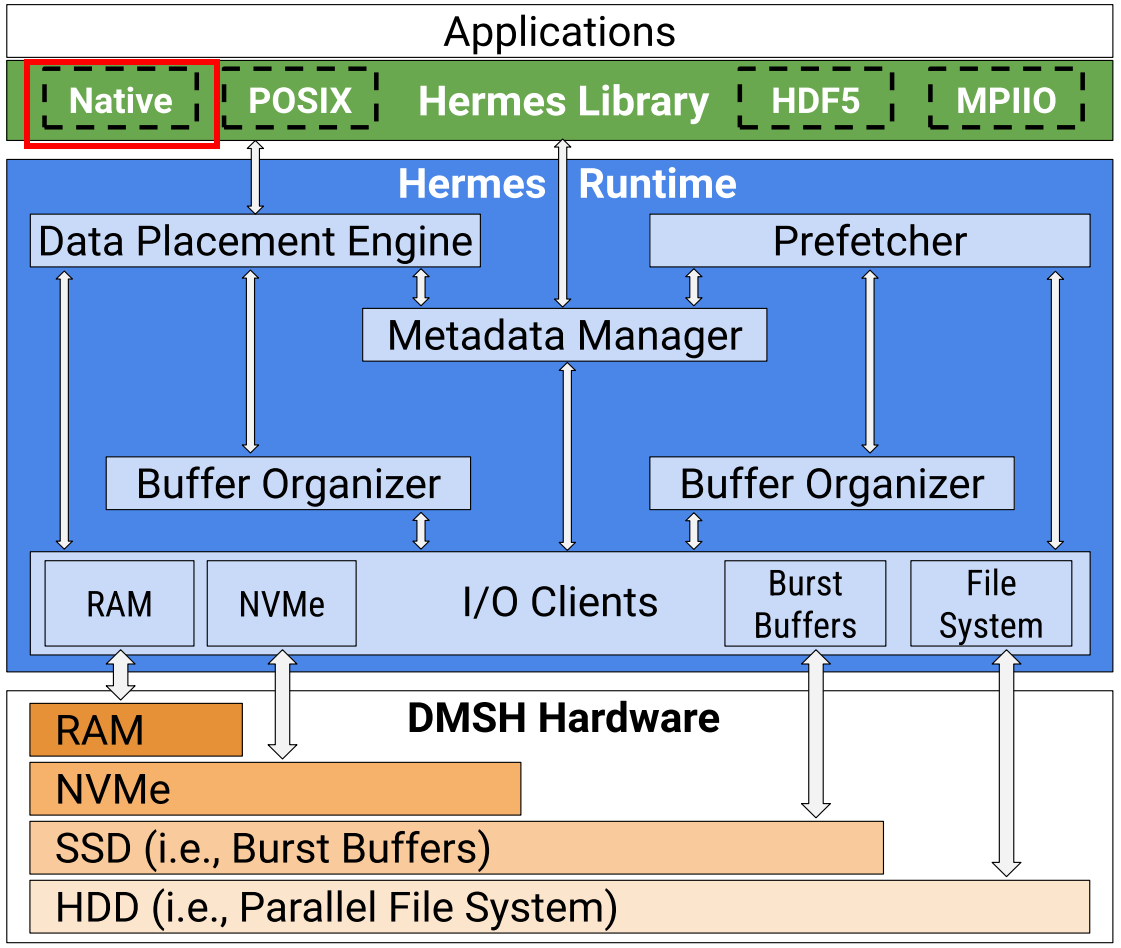
Bucket: USDA Nutrition Data

Key	Calories	Sat. Fat	Carbs
Butter	714	50g	0g
Olive Oil	800	13.3g	0g
Chicken	188	2.94g	1.18g

Blob 1: Butter

Blob 2: Olive Oil

Blob 3: Chicken

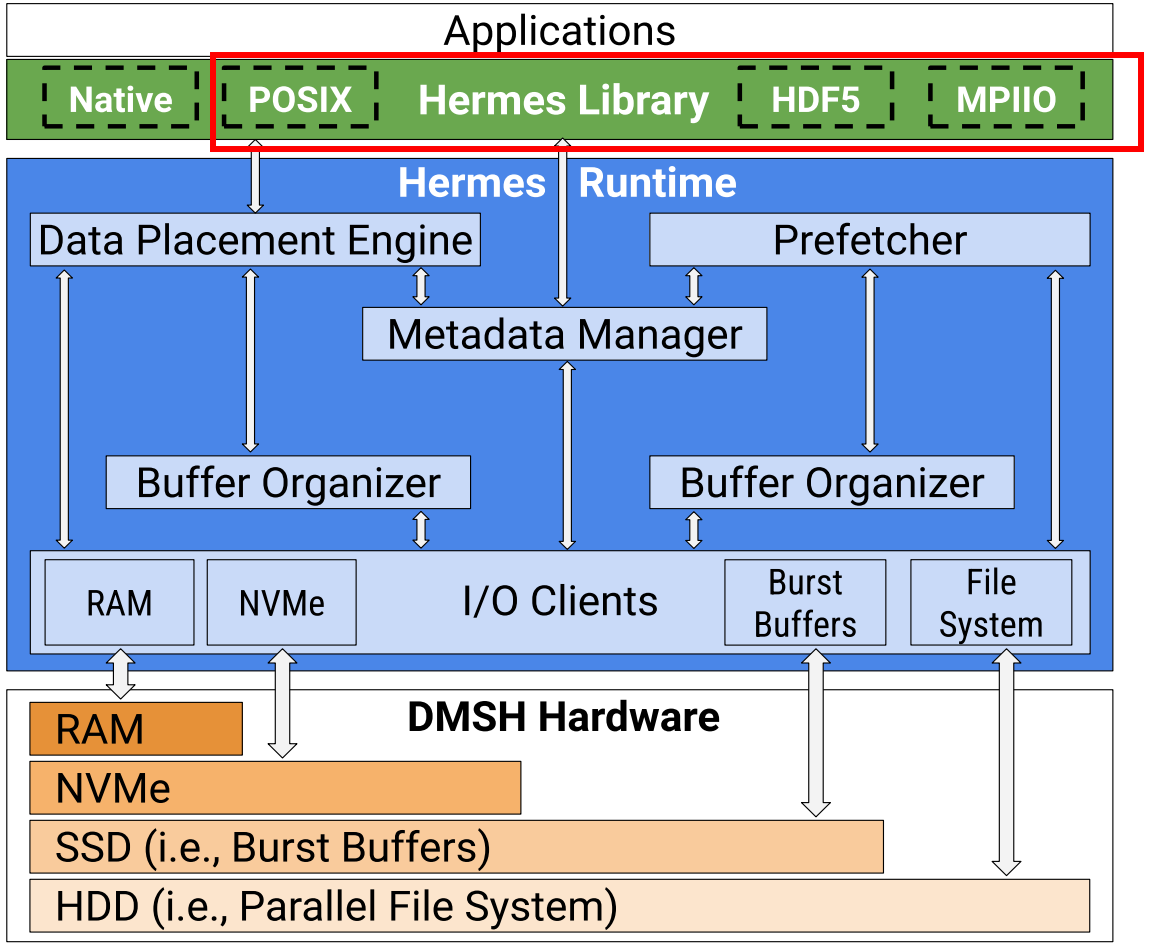


Adapters

- Intercept I/O and convert them into Hermes Native API calls
- **Avoids application changes to use Hermes!**
- Various APIs are supported

(MPICH)

*In Progress

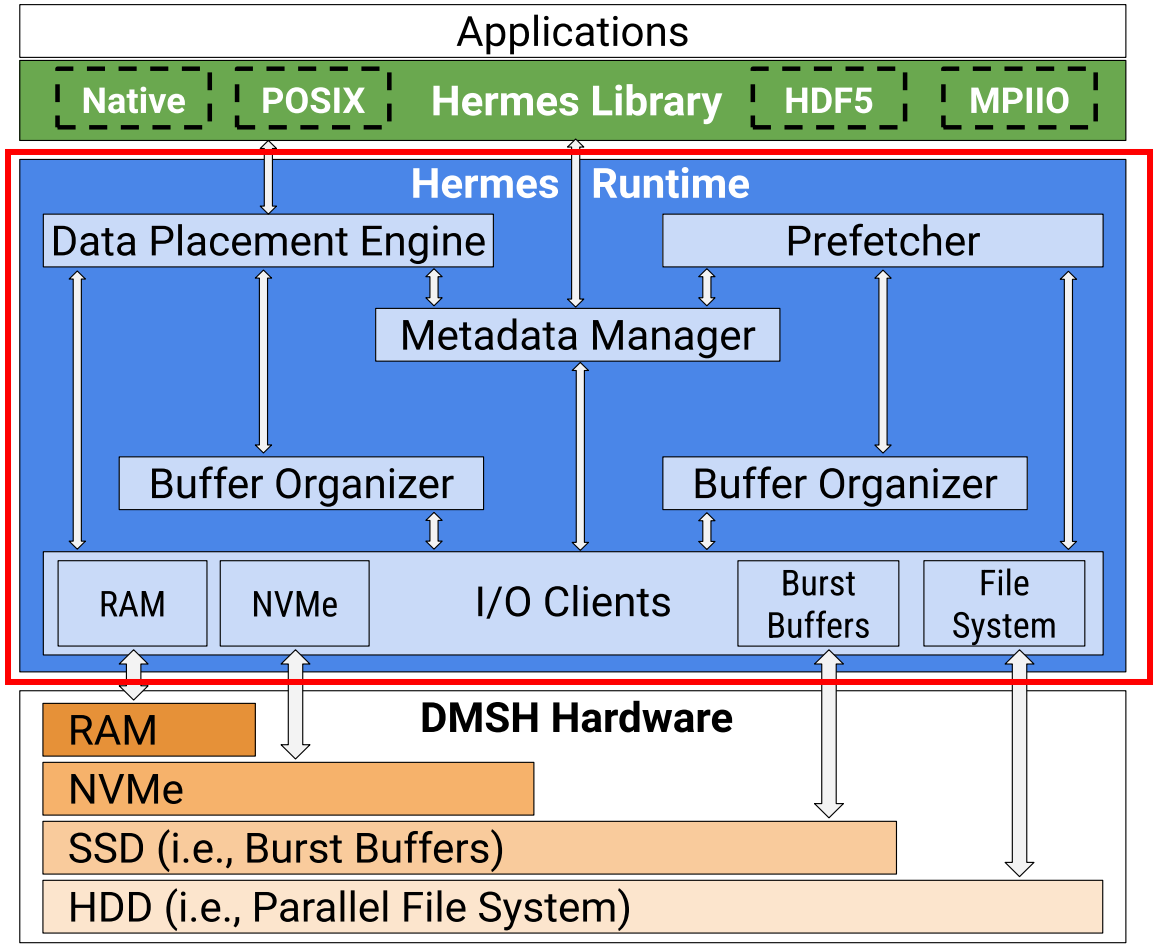


Hermes Runtime

- The Hermes Library converts Native API calls into messages, which are executed by the Hermes Runtime
- Messages sent through shared-memory, lock-free queues
- Two main benefits:

Hermes can run longer than a single application

Many applications can use Hermes simultaneously



Data Placement Engine

- Decides the initial placement of data during a PutBlob operation
- Three given algorithms:

MinIo Time

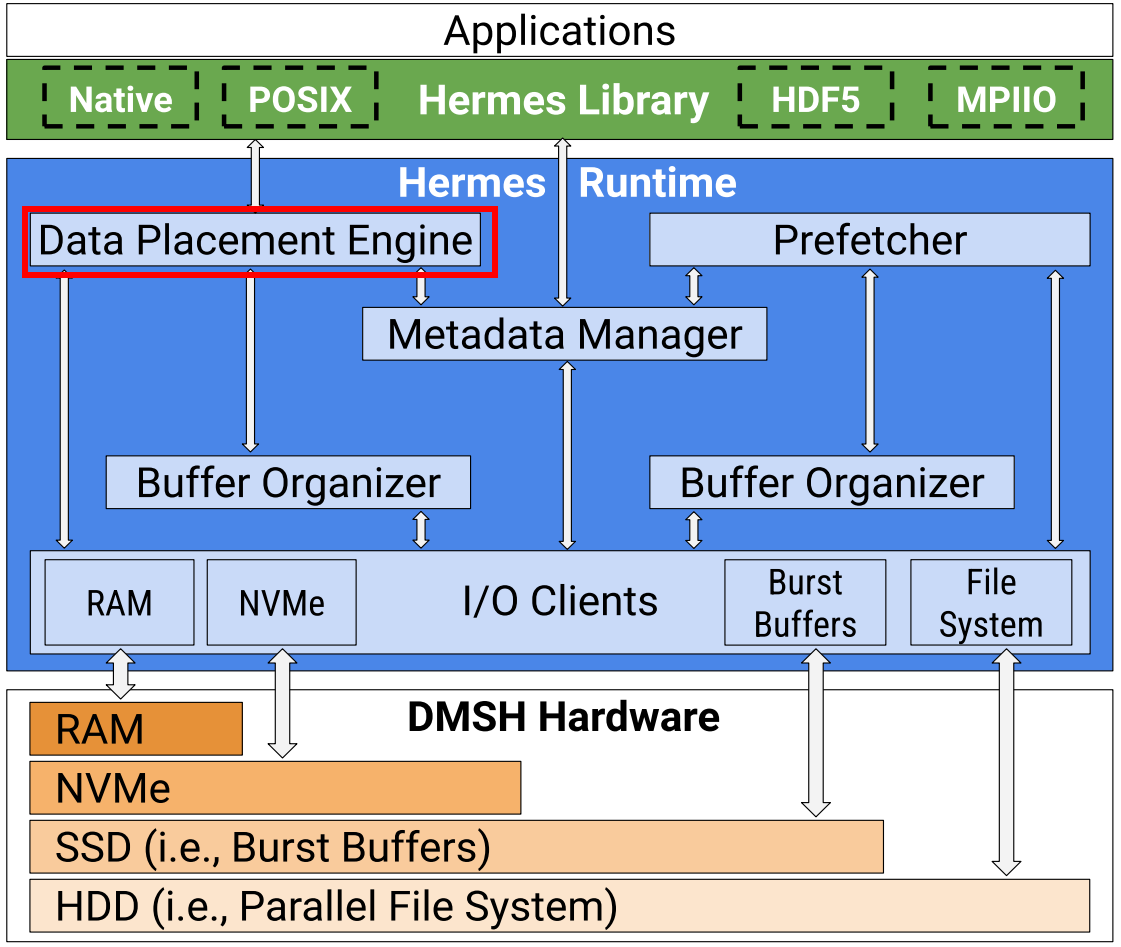
Place data in fastest tier with space

Round Robin

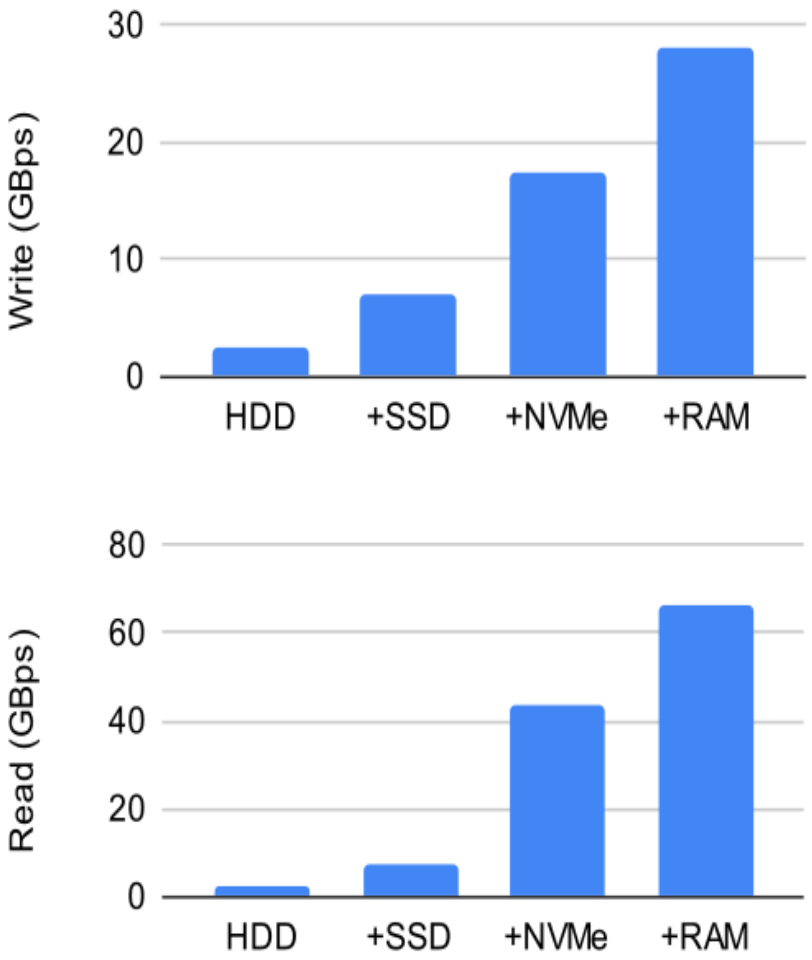
Each tier assigned an equal number of blobs

Rand

Randomly choose tier

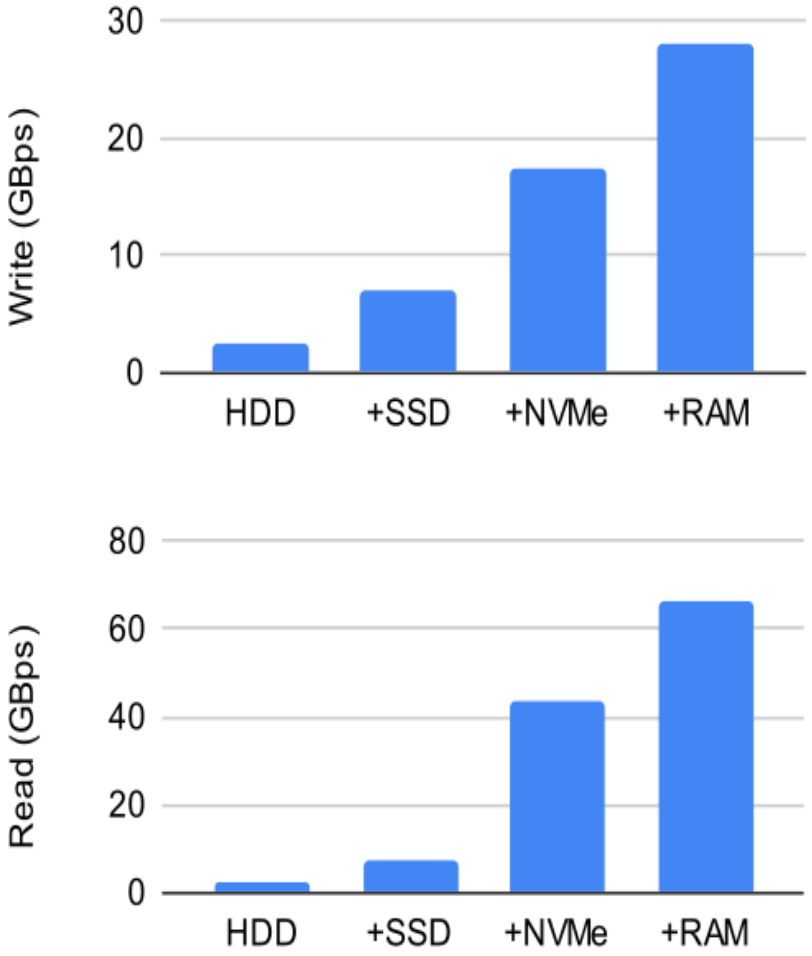


Impact of Tiering on Application Performance



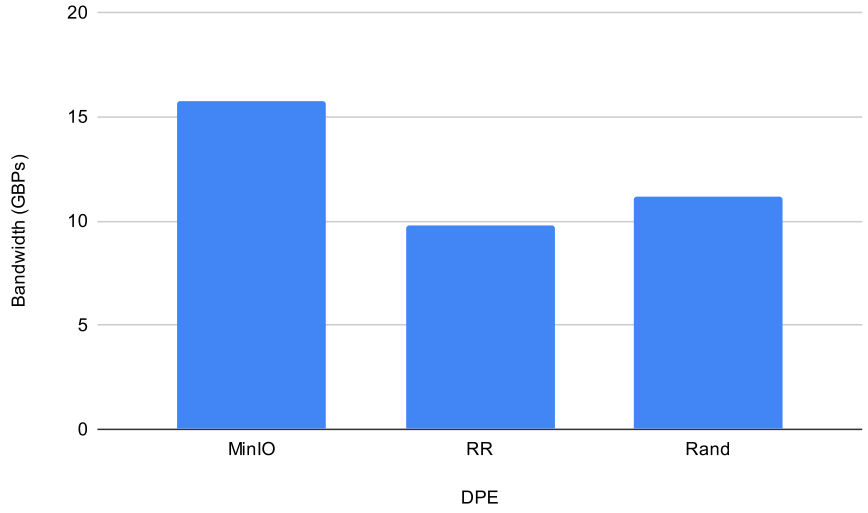
- Compare tiering for two workloads:
- VPIC: particle-in-cell simulation code for modeling 3D kinetic plasmas (write-only)*
- BD-CATS: particle clustering algorithm (read-only)
- 16 nodes, 16 processes per node
- MinIOTime DPE

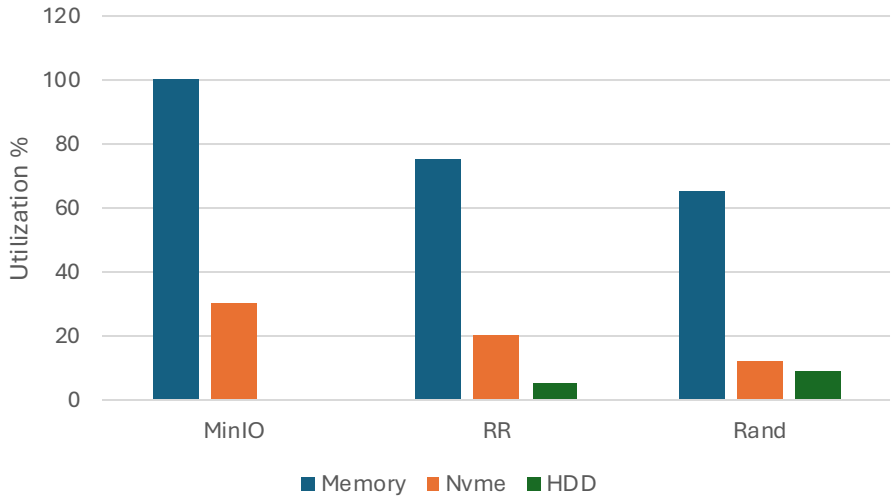
Impact of Tiering on Application Performance



- With each additional tier, Hermes performs several times better
- Full tiering yields minimum 15x performance boost for read and write
- DPEs can efficiently remove the burden of data placement from users

Pros and Cons of DPE Choice



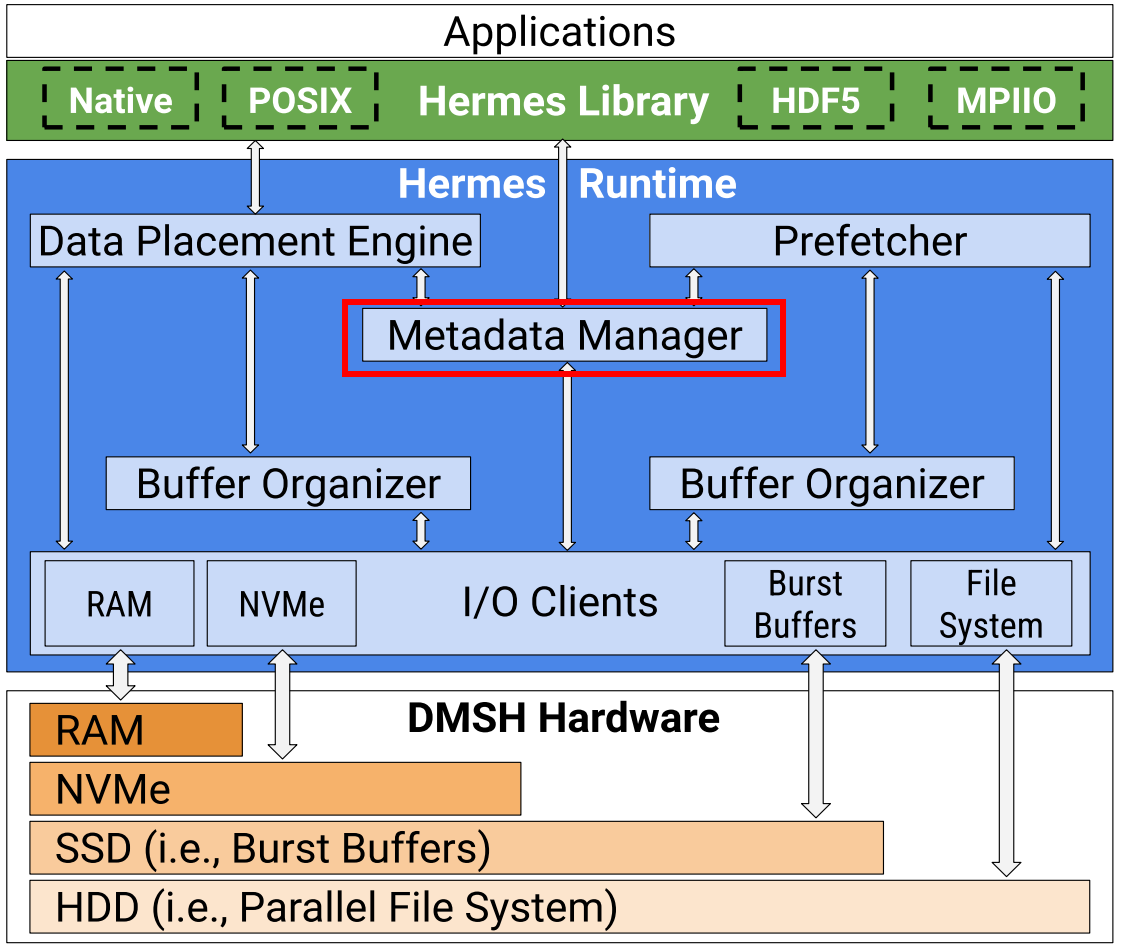


- Synthetic benchmark that PUTs 10GB worth of blobs per node
- 8 nodes and 16 processes per node

- MinIO places data in the fastest tiers, and thus performs the best
- Round-Robin and Rand utilize all resources, reducing RAM pressure

Metadata Manager

- Stores the metadata for blobs and buckets
  - E.g., position of blob in the DMSH
- Can be configured to track blob and bucket ops
  - E.g., track the order with which blobs are created and modified
  - Useful for monitoring

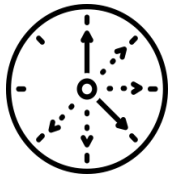


Buffer Organizer

- Corrects the position of the blob after initial placement
- Various factors involved in the decision

I/O Size

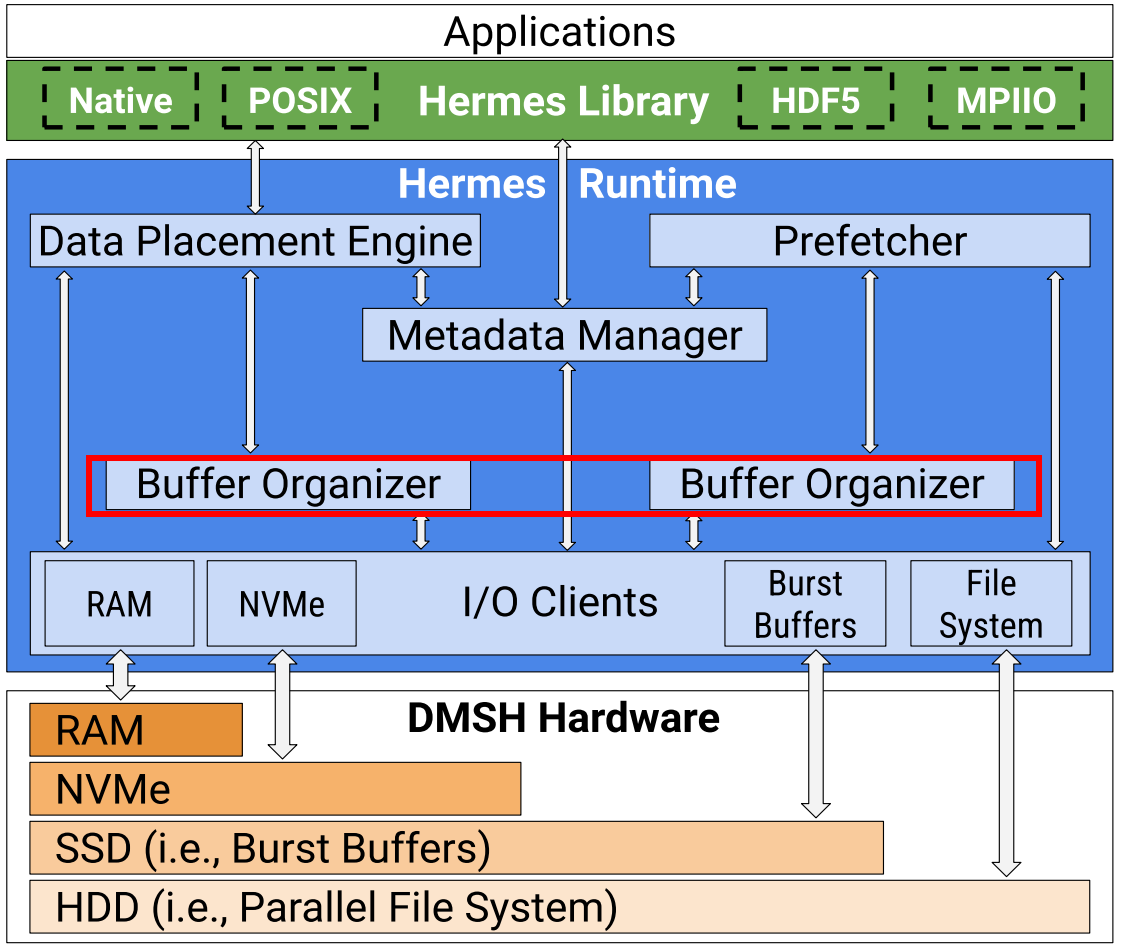
Access Frequency



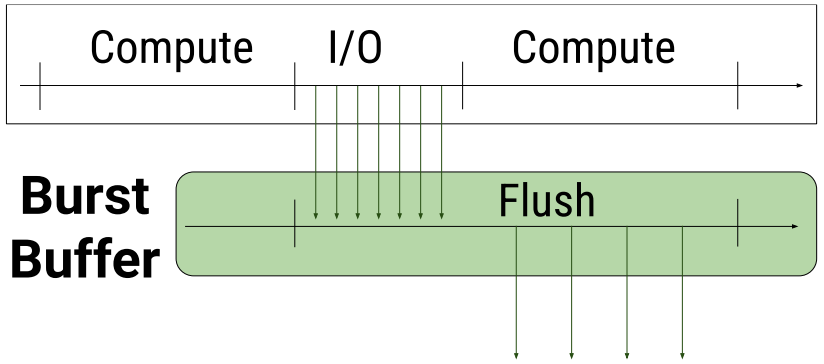
Access Recency

User Hints

(e.g., IS_META)



Buffer Organizer



An example of flushing blobs during compute

Promote

Higher score -> faster tier

Demote

Lower score -> slower tier

- Corrects the position of the blob after initial placement
- Various factors involved in the decision

I/O Size

Access Frequency

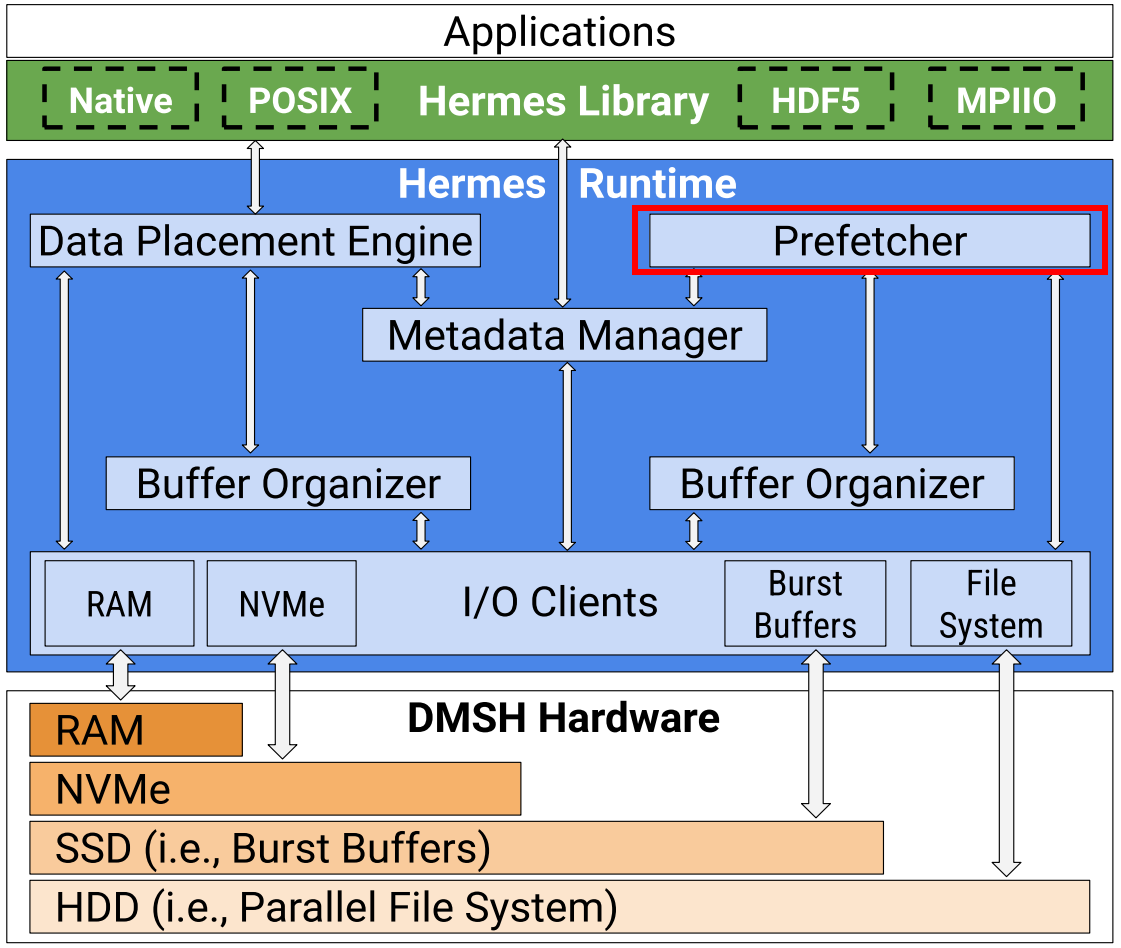
Access Recency

User Hints

(e.g., IS_META)

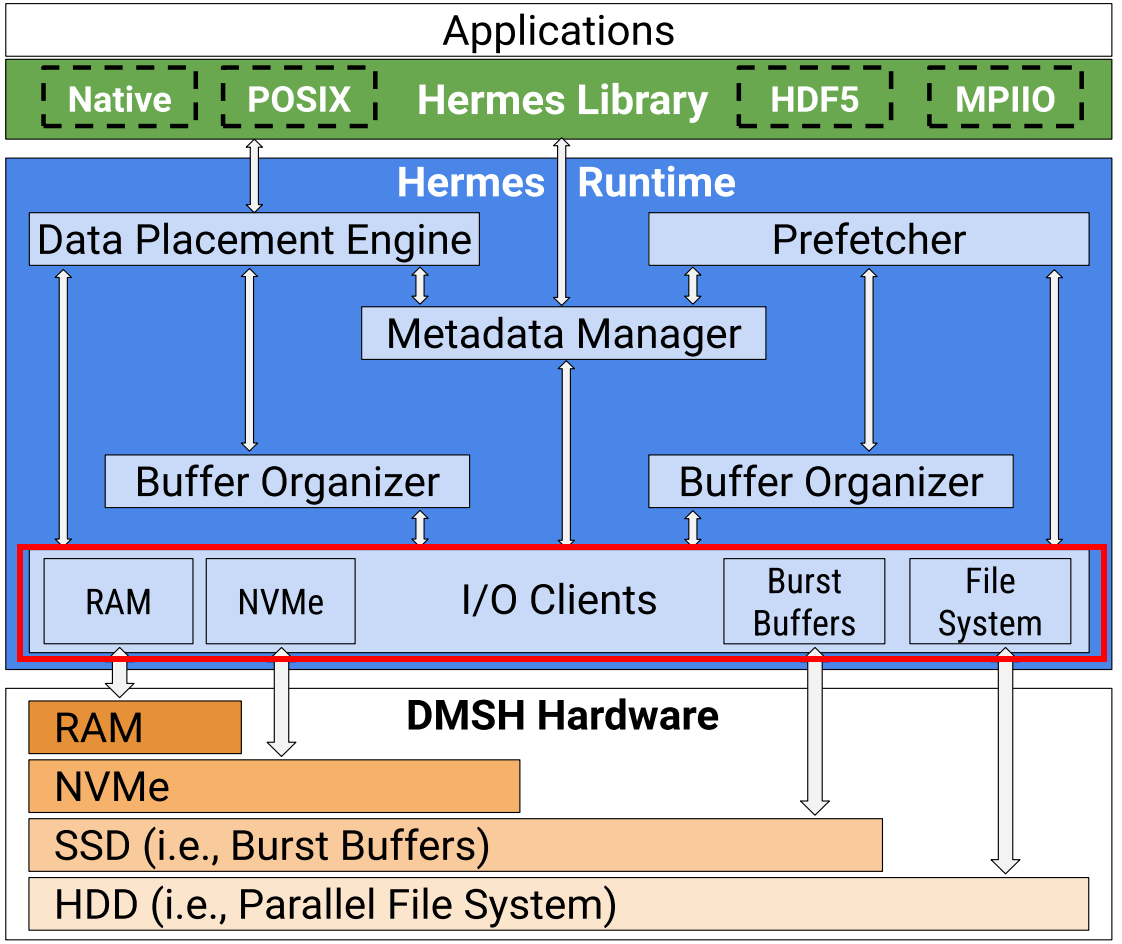
Prefetcher

- Changes the score of a blob based by predicting access pattern
- Can access I/O pattern logs from the Metadata Manager to analyze access patterns dynamically



I/O Clients

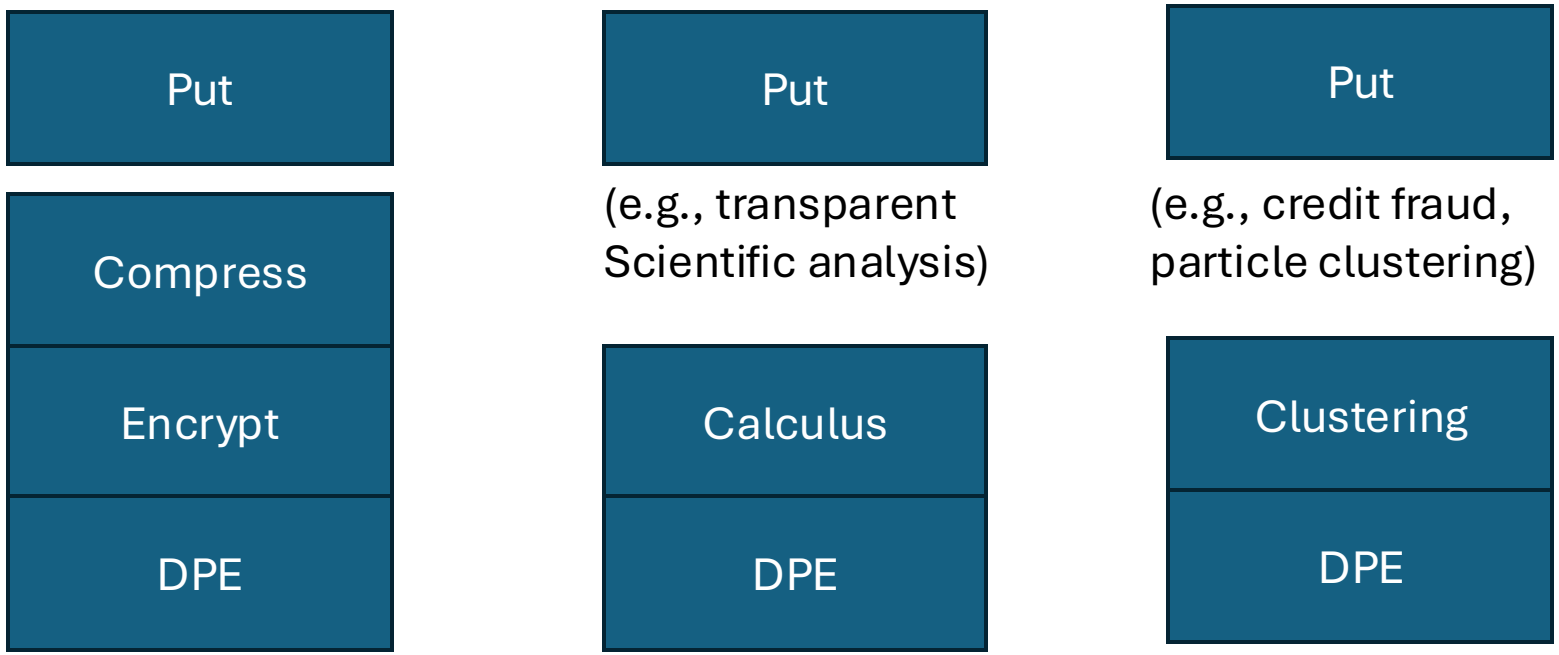
- Interact with a variety of different storage hardware
- Many different storage APIs available
  - POSIX
  - SPDK for NVMe
  - LighNVM for NVMe
  - Memory Map for memory, PM, & CXL
- This class unifies the different APIs to allow Buffer Organizer to place data



API Additions

Composable, Active Storage through Traits

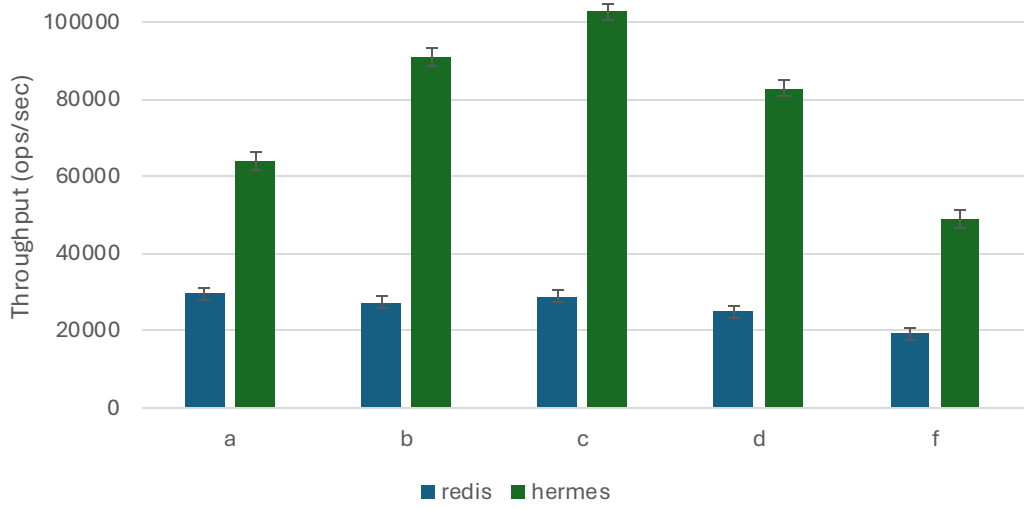
- Many applications desire the ability to apply operators near data
- Hermes allows for traits to be added to the I/O stack
  - *Evaluation omitted due to time



Hermes For the Cloud

- Hermes can also be used for Cloud applications
- Many cloud applications could be built using the Hermes API
- Key-value stores and databases are an example

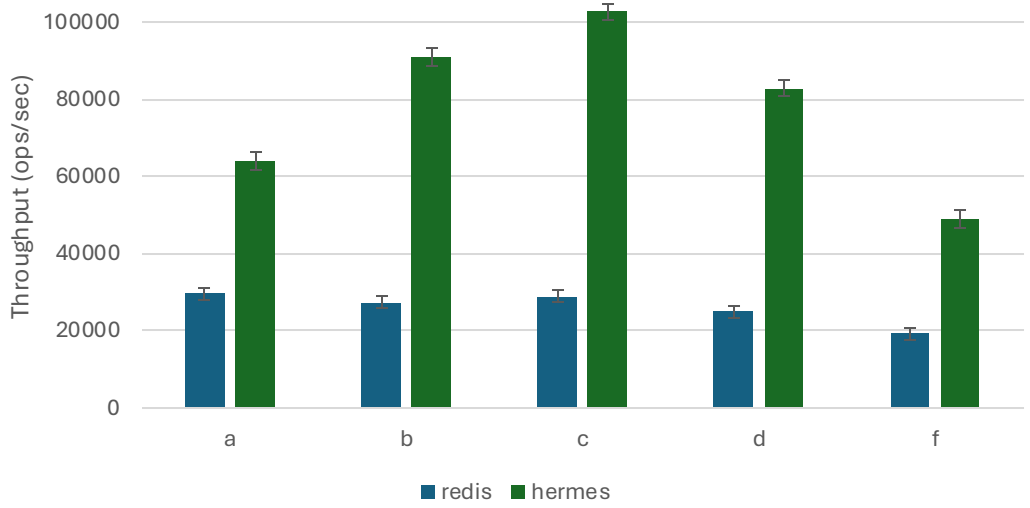
Evaluation: Yahoo Cloud Service Benchmark



- YCSB benchmarks various web workloads
- Latency-sensitive
- We compare against Redis, a widely-used key-value store
  - Hermes & Redis have analogous APIs

Workload	%Read	%Update	%Insert	%RMW	I/O Size	Description
a	50	50	0	0	1KB	Session store recording recent actions
b	95	5	0	0	1KB	Read/update photo tags
c	100	0	0	0	1KB	User profile cache
d	95	0	5	0	1KB	User status updates
f	50	0	0	50	1KB	A user database

Evaluation: Yahoo Cloud Service Benchmark



Workload	%Read	%Update	%Insert	%RMW	I/O Size
a	50	50	0	0	1KB
b	95	5	0	0	1KB
c	100	0	0	0	1KB
d	95	0	5	0	1KB
f	50	0	0	50	1KB

- Hermes performs 2-3x faster
- Redis requires several copies to get data to the Redis daemon
- Hermes uses shared memory, reducing the number of data copies
- Additionally, writes are asynchronous, further improving latency

Conclusion

Conclusion

- Demonstrated that Hermes can yield as much as 15x performance boost for applications by leveraging tiering
- Showed that Hermes can be adapted to a wide variety of applications spanning HPC and Cloud
- Described various use-cases that can benefit from tiered, active storage