SCALABLE COMPUTING
SOFTWARE LABORATORY

ILLINOIS INSTITUTE OF TECHNOLOGY

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

HPDC'18

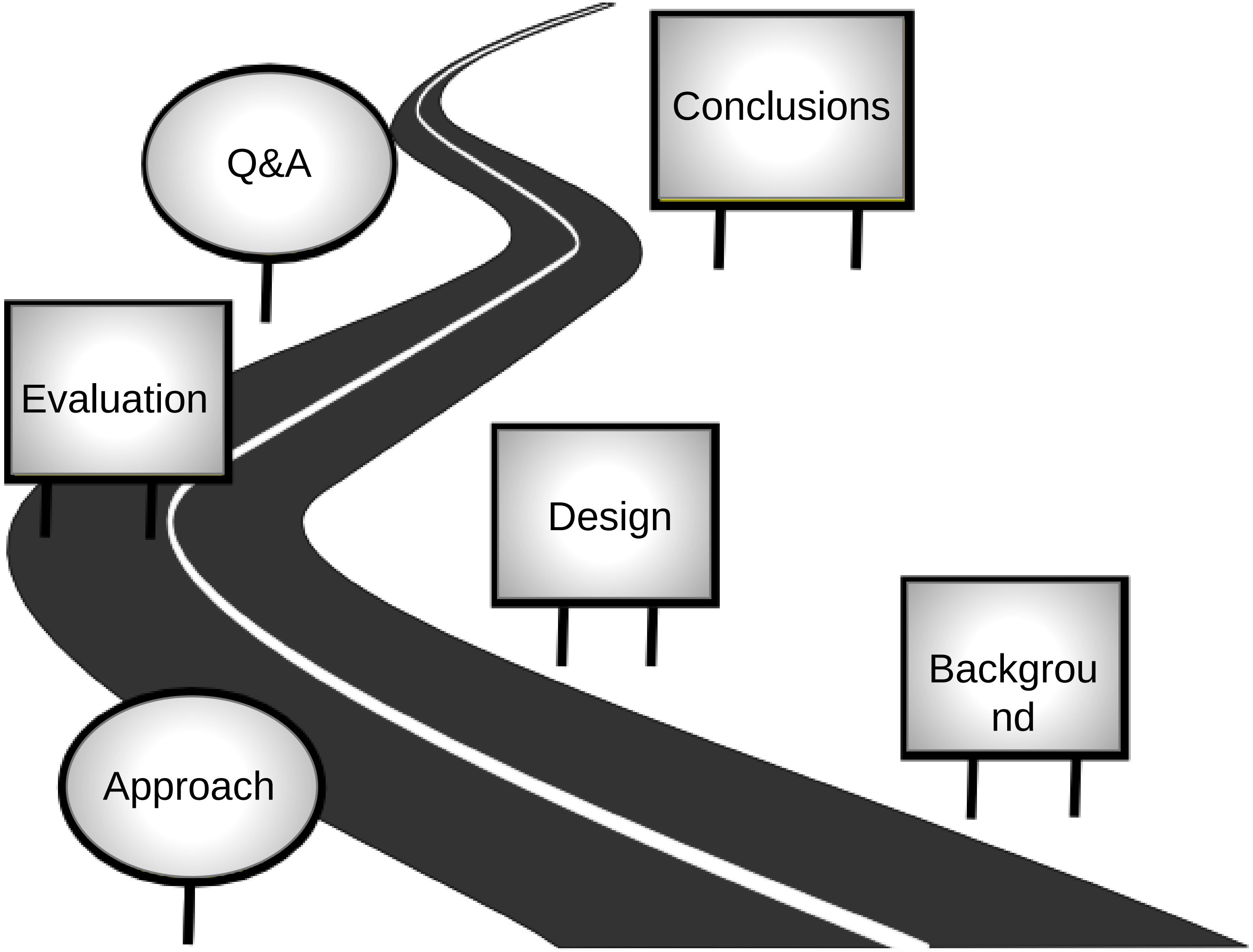
Anthony Kougkas, Hariharan Devarajan, Xian-He Sun
akougkas@hawk.iit.edu

Department of Computer Science
Illinois Institute of Technology

Tempe, AZ, USA

June 15th, 2018

Agenda



Talk roadmap

Highlights of this work

- **Hermes multi-tiered I/O buffering**
  - Heterogeneous Aware
  - Selective buffering
- **Data placement policies**
  - Deep memory and storage hierarchy
  - Transparent to the user
- **Hermes can offer**
  - 5-7x performance gains over baseline
  - 2x better performance compared to other buffering

Storage in HPC

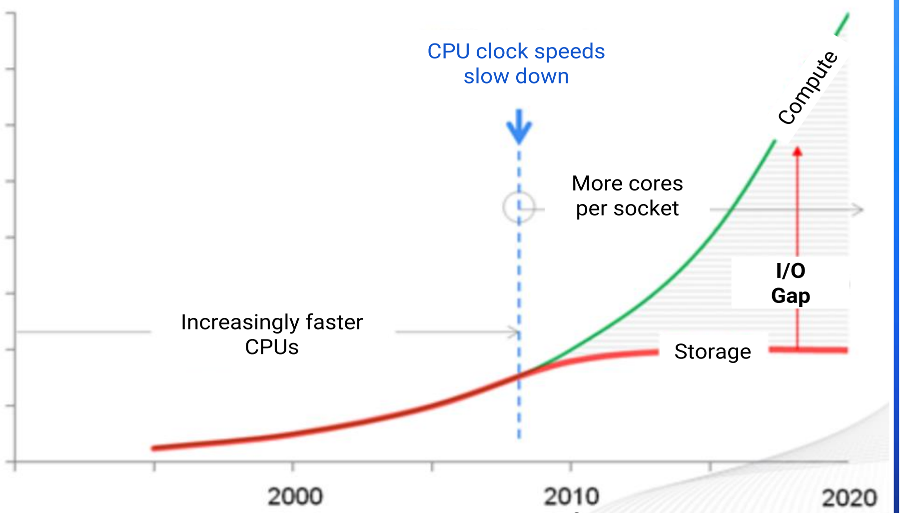
- Parallel File Systems (PFS):
  - Peak performance: ~2000GiB/s
  - Capacity: >70PiB
- Interfaces:
  - POSIX, MPI-IO, HDF5, etc.,
- Limitations:
  - Scalability, complexity, metadata services
  - Small file access, data synchronization, etc.,

#	site.name	site.storage system.net capacity	site.storage system.peak read	site.storage system.peak write	site.storage system.software
		in PiB	*in GiB/s*	*in GiB/s*	
1	kaust	16.96	1955.78	1955.78	Lustre,Cray Tiered Adaptive Storage (TAS)
2	jcahpc	24.10	1918.52	1918.52	Lustre,Lustre
3	riken	39.77	3187.23	1510.85	FEFS,Lustre,FEFS,staging
4	ncsa	27.60	1158.00	1158.00	Lustre,HPSS
5	nscg	14.40	1100.00	1100.00	H2FS
6	llnl	48.85	1000.00	1000.00	Lustre,ZFS
7	ornl	28.00	1000.00	1000.00	Lustre
8	nersc	30.00	700.00	700.00	Lustre,DataWarp
9	jamstec	19.62	407.92	407.92	ScaTeFS,NFS,ScaTeFS,NFS
10	nscc	17.76	288.00	288.00	

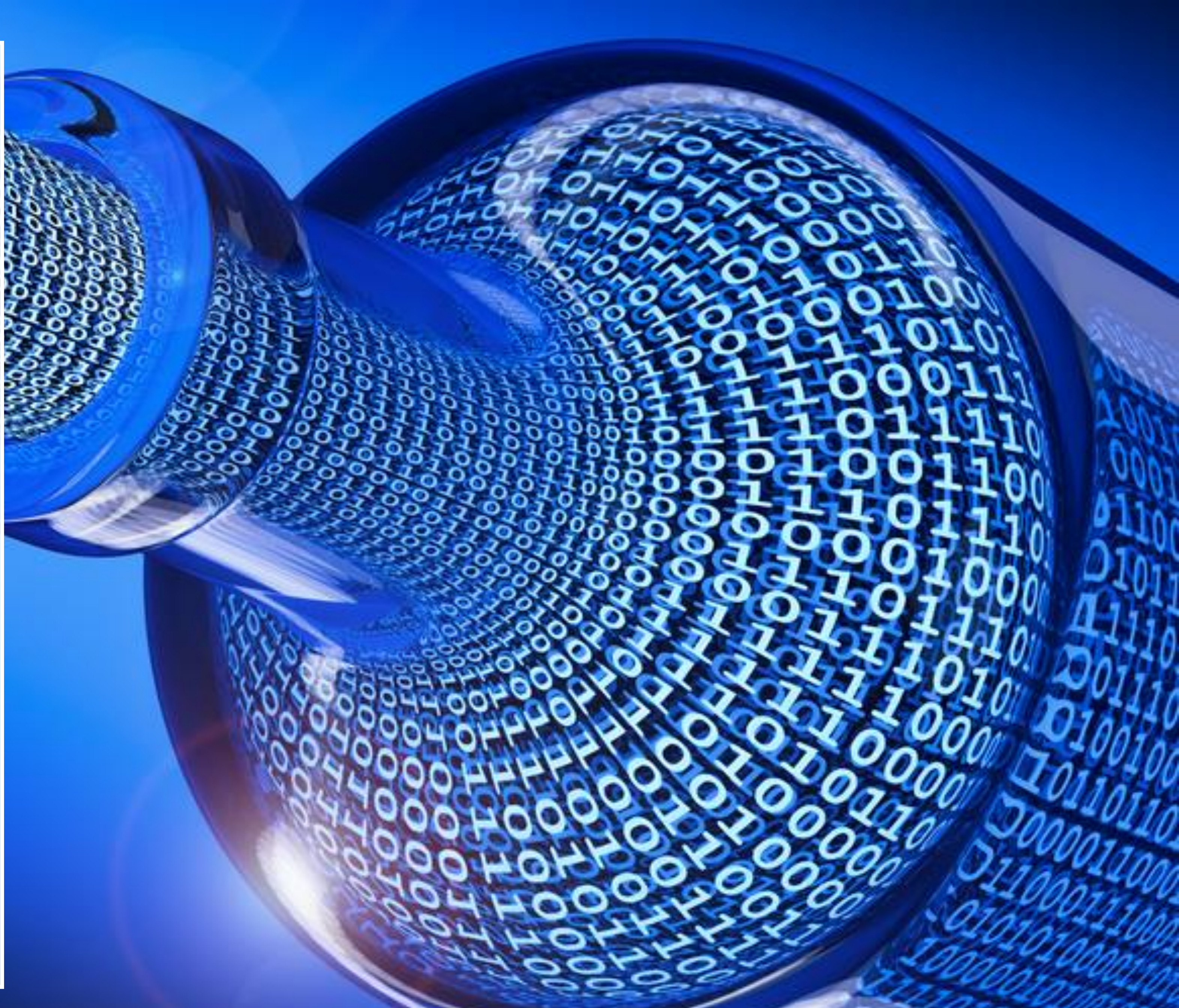
I/O 500 List (Nov 2017)

I/O Bottleneck

- I/O subsystems struggle to deal with the degree of parallelism.
- Any computer system's performance is limited by its slowest component. (Amdahl's "well-balanced" law)
- I/O performance bottlenecks, already a concern at petascale, are likely to become alarmingly narrow as we start the ascent to the exascale.



Source: © 2015 DataCore Software Corporation



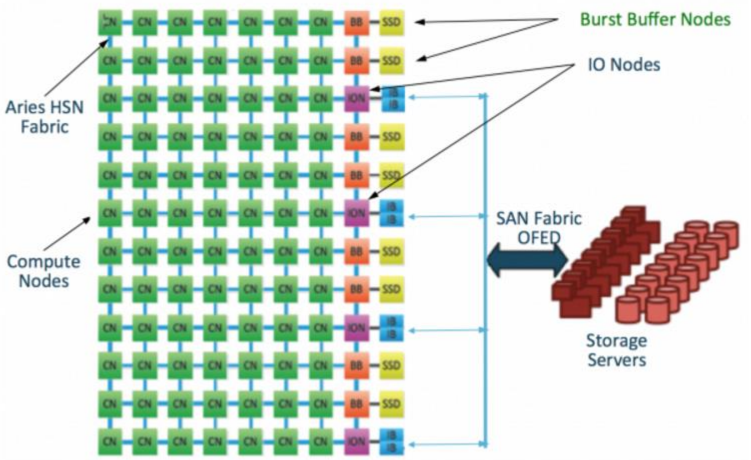
I/O acceleration in HPC

Trends:

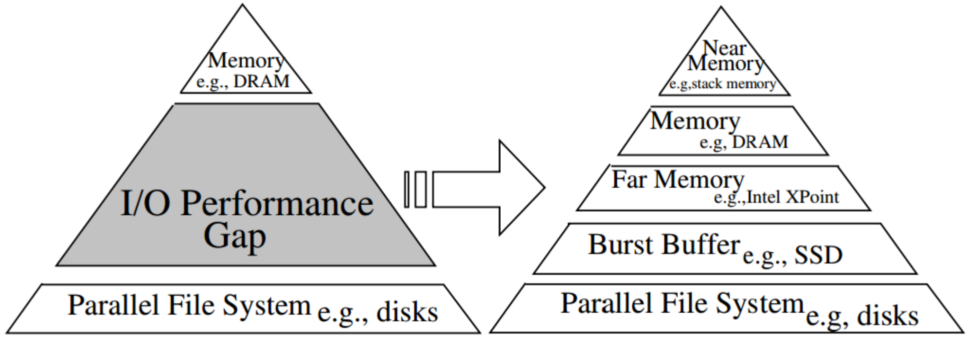
1. Fast storage devices inside compute nodes
2. Shared buffering nodes (a.k.a. burst buffers)

Deep Memory and Storage Hierarchy(DMSH)

- Multiple levels of memory and storage in a hierarchy, called DMSH.
- New storage system designs incorporate non-volatile **burst buffers** between the main memory and the disks.
- HPC hierarchical storage systems with burst buffers (BB) have been installed at several HPC sites.
- Ideally, the presence of multiple layers of storage should be **transparent** to applications without having to sacrifice **I/O performance**.



Cori, a Cray XC40 system at NERSC uses Cray's DataWarp BB technology



Challenges of I/O Buffering in DMSH

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.

Our Approach

DMSH systems require	a scalable, reliable, and high-performance software to efficiently and transparently manage data movement.
DMSH complexity demands	new data placement algorithms, memory and metadata management, and an efficient communication fabric.
We envision a buffering platform that can be	application- and system-aware, and thus, hide lower level details allowing the user to focus on his/her applications.

HermeS

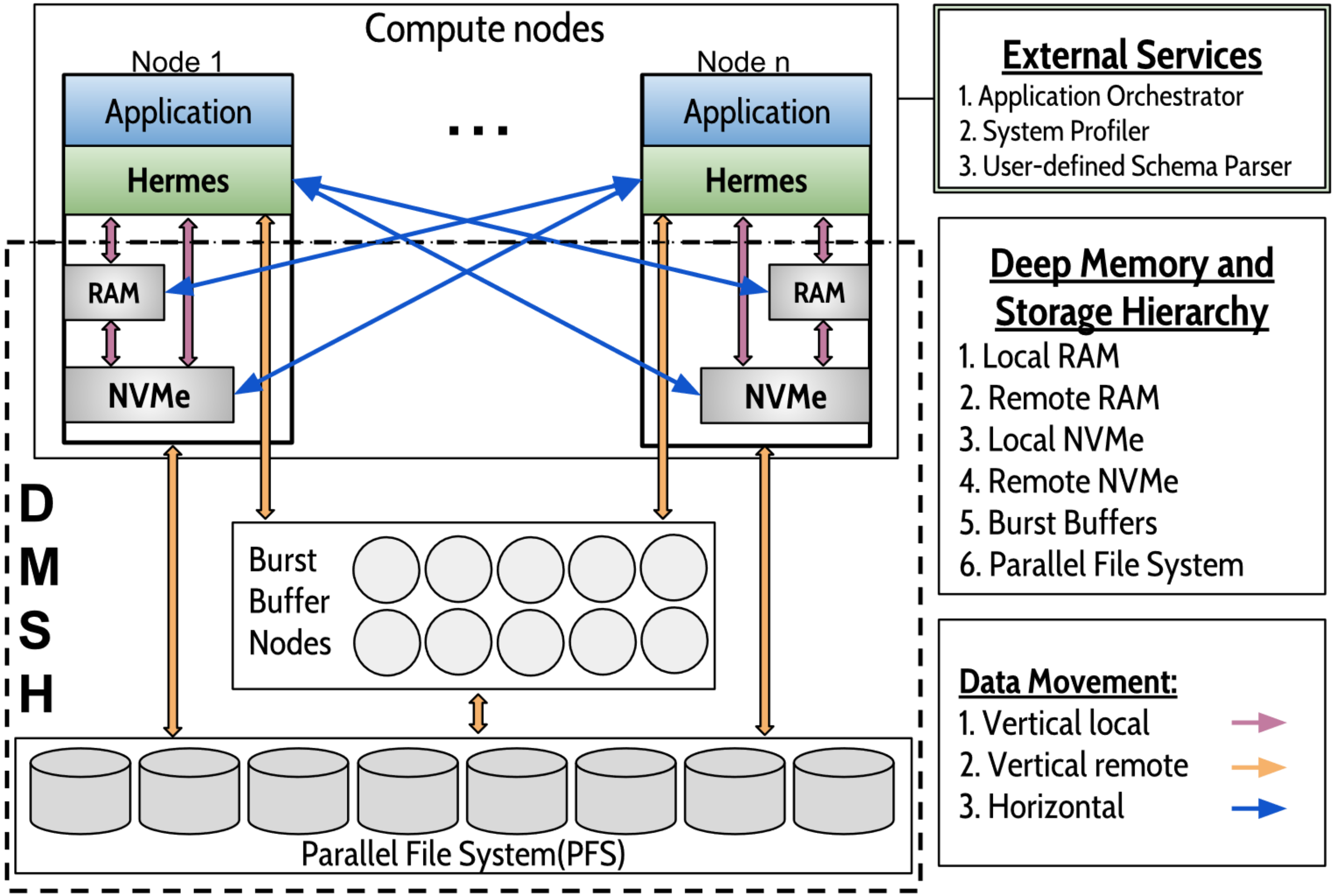
A heterogeneous-aware, multi-tiered distributed I/O buffering system

---

- Vertical and horizontal distributed buffering in DMSH
- Selective layered data placement
- Dynamic buffering via system profiling

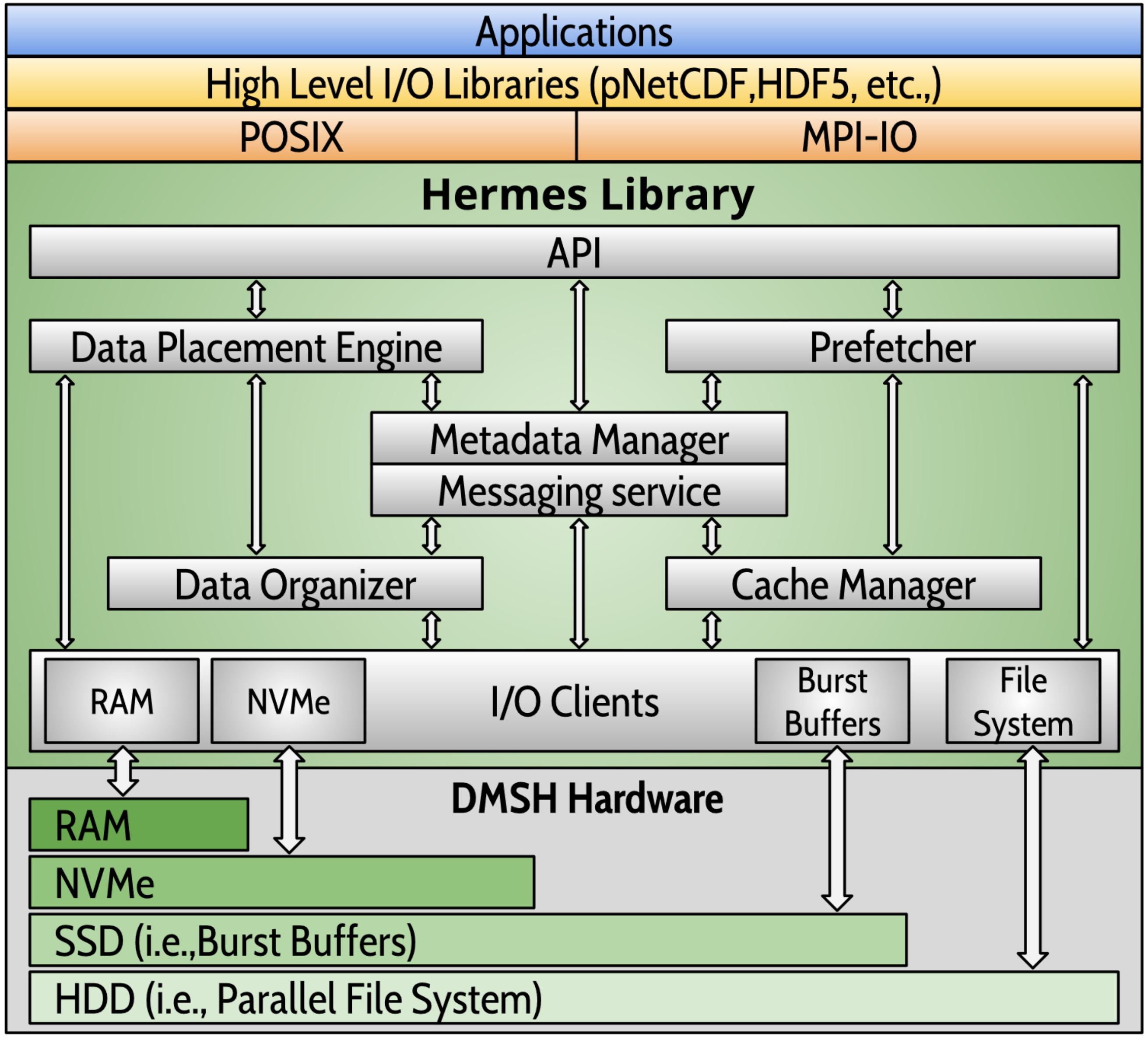
Architecture

- Each compute node has access to
  - Local NVMe or SSD device
  - Shared Burst Buffers
  - Remote disk-based PFS
- Hierarchy based on speed and capacity (numbered in figure)
- Two data paths:
  - Vertical (within node)
  - Horizontal (across nodes)



Design

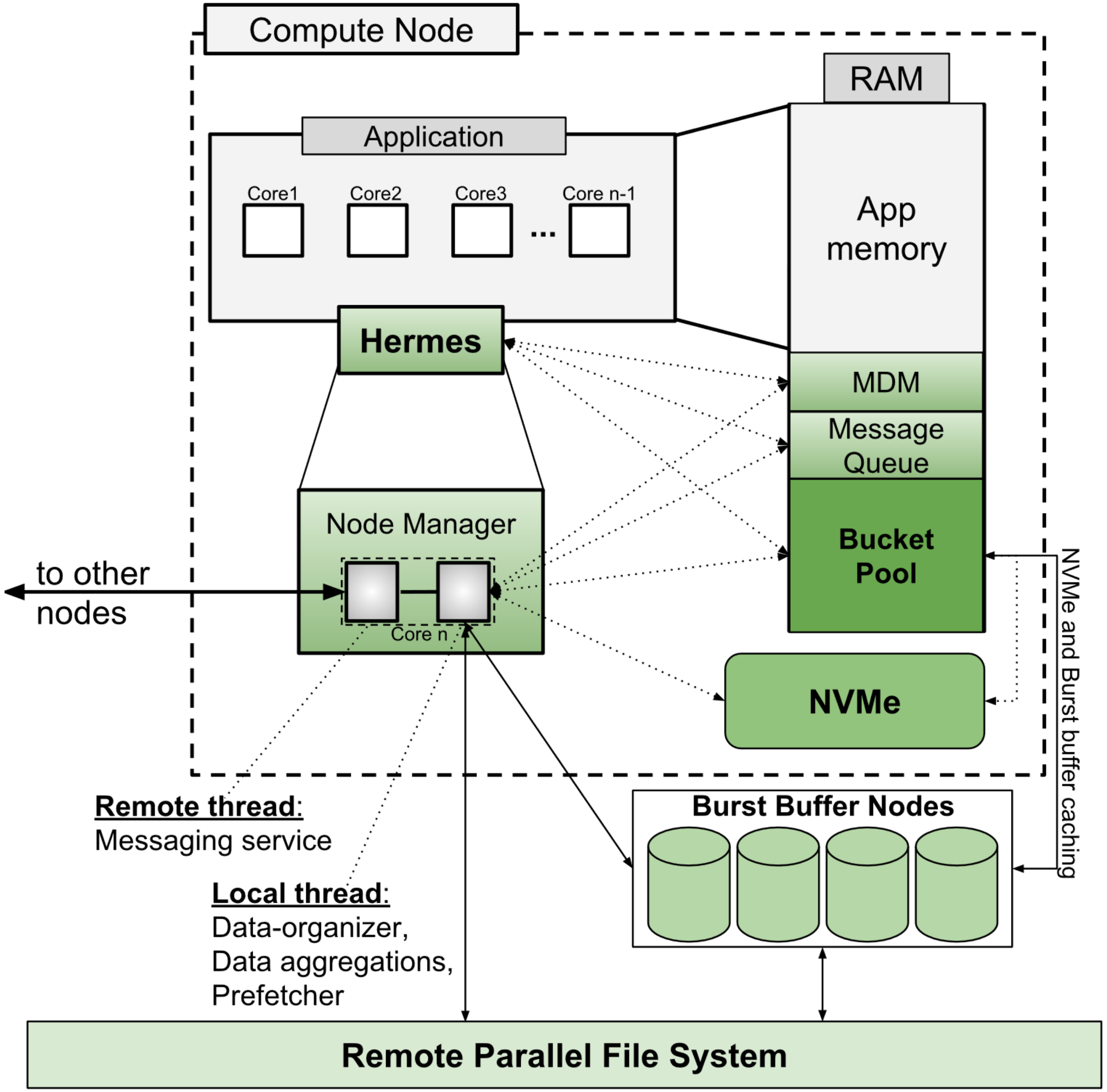
- Middle-ware library, written in C++,
- Link with applications (i.e., re-compile or LD_PRELOAD)
- Wrap-around I/O calls
- Modular, extensible, performance-oriented
- Supports:
  - POSIX
  - HDF5
  - MPI-IO (ongoing)
- Hinting mechanism to pass operations



Anthony Kougkas, akougkas@hawk.iit.edu

Node Design

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



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

Data Placement Policies

Maximum Bandwidth

- Maximize performance applications experience
- Top-down approach, place data higher up and trigger down
- Balance between bandwidth, latency, and capacity of layer
- Default in Hermes

Maximum Data Locality

- Maximize buffer utilization
- Side-ways approach, place data in all layers based on dispersion unit
- Balance between capacity and data's spatial locality
- Ideal for workflows that encapsulate partitioned I/O

Hot Data

- Offer applications a fast cache for frequently accessed data
- Hotness score based on file access frequency
- Place hot data higher up in the hierarchy
- Ideal for workflows with a spectrum of hot-cold data

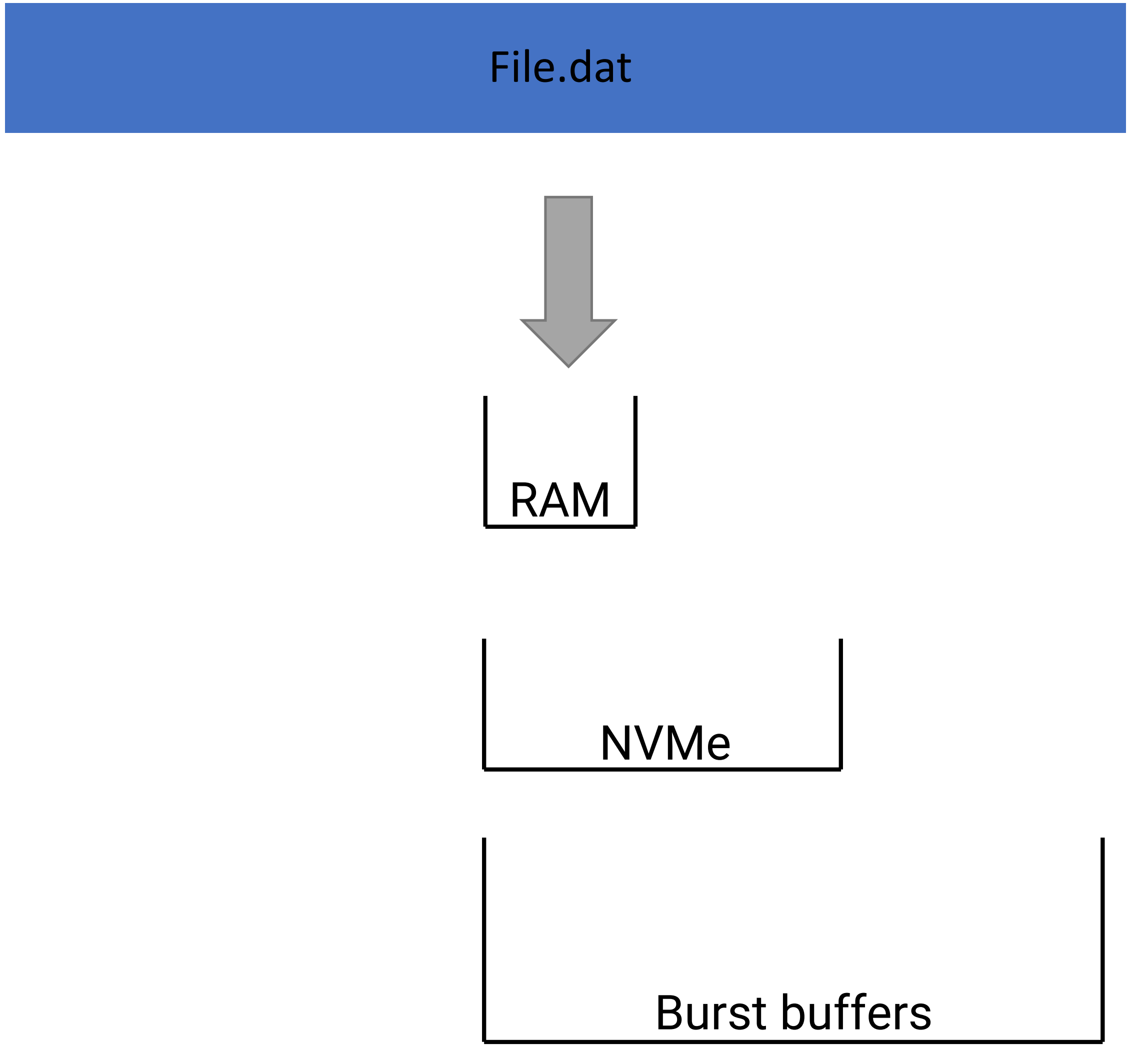
User-defined

- Supports user-defined buffering schemas
- Users submit an XML with requirements
- Parsed during initialization

Anthony Kougkas, akougkas@hawk.iit.edu

Maximum Bandwidth

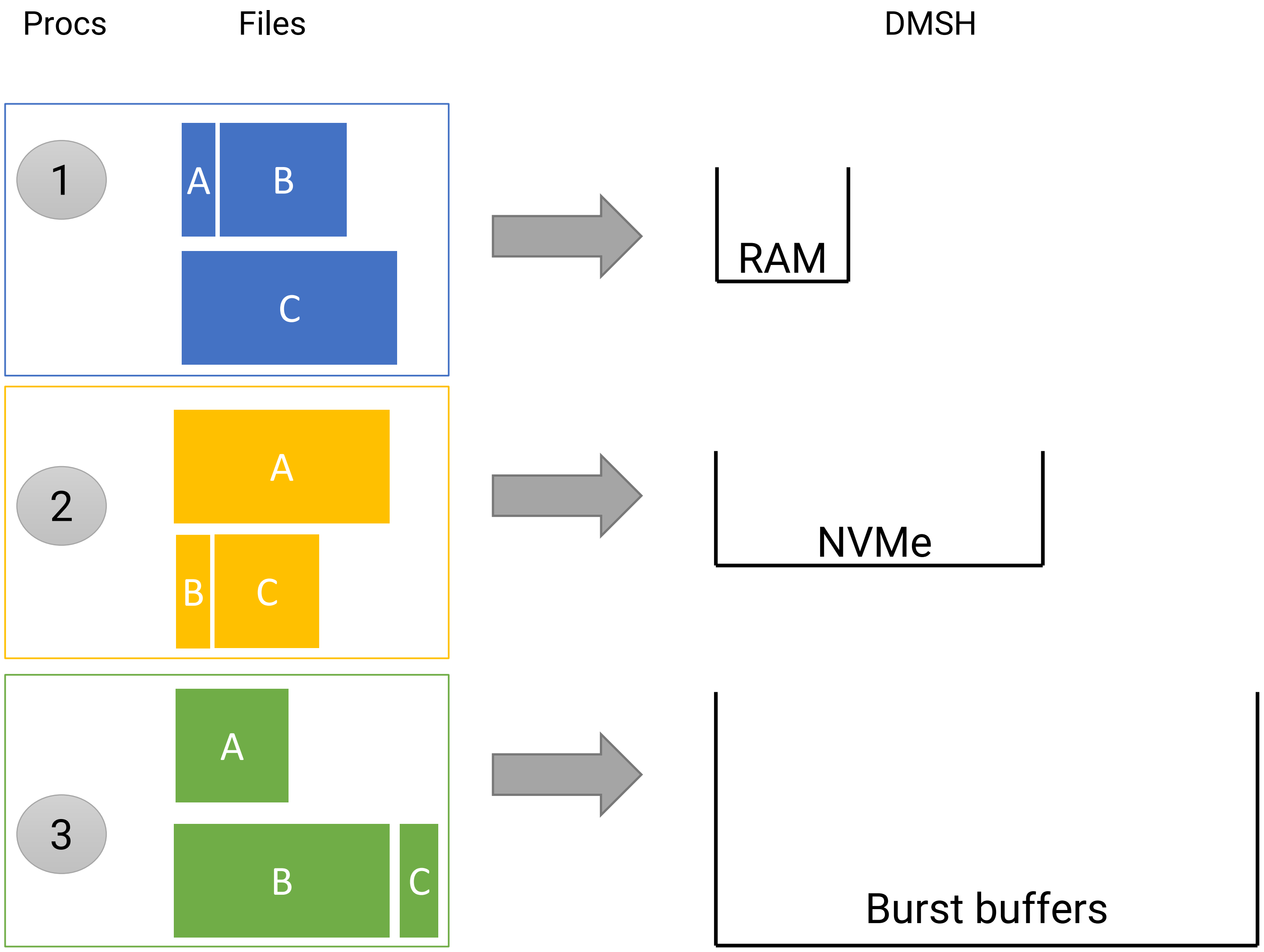
- Start from the top layer
  - If free space > request size
    - place data here
  - If not, choose the best between
    1. Place as much data as possible here and the rest to the next layer
       OR
    2. Skip this layer and place data to the next one
       OR
    3. First flush top layer and then place data
- Recursive process



Anthony Kougkas, akougkas@hawk.iit.edu

Data Locality

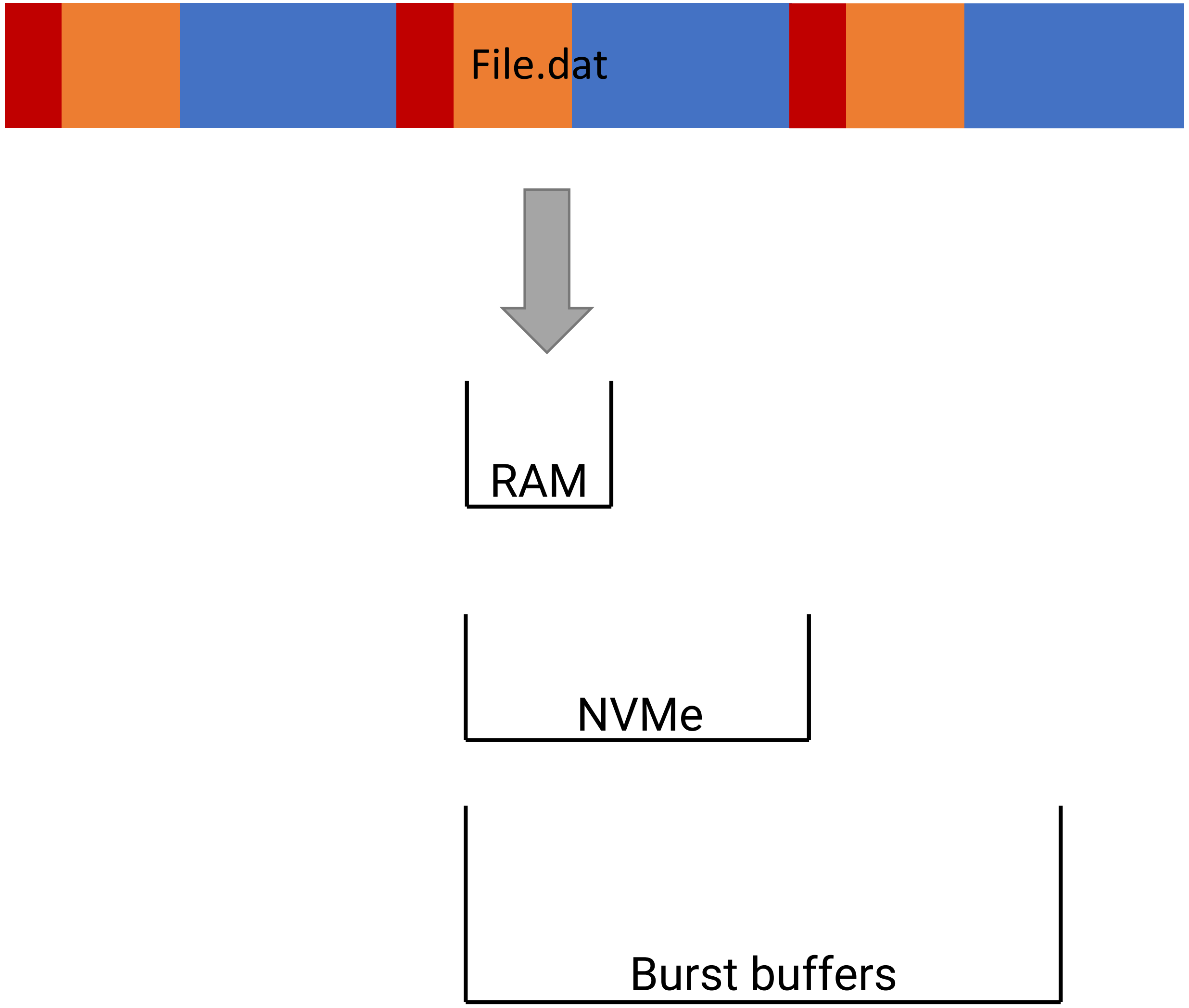
- Data dispersion unit:
  - POSIX files
  - HDF5 datasets
  - Etc.
- Place data based on:
  - Location of previously buffered data
  - Ratio between layers



Anthony Kougkas, akougkas@hawk.iit.edu

Hot data

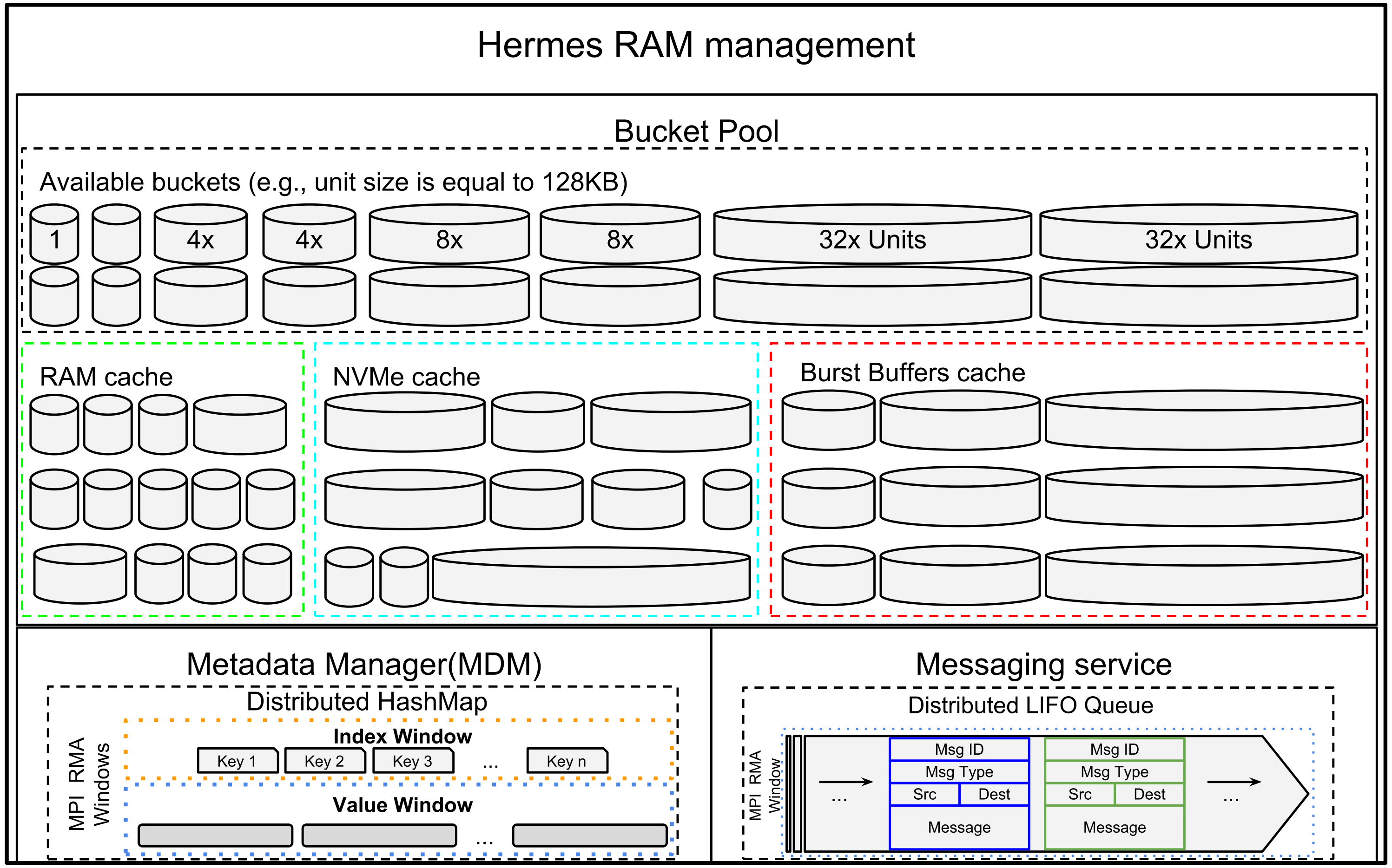
- Place data based on:
  - Spectrum of hot – cold data
- Higher layers hold hotter data



Anthony Kougkas, akougkas@hawk.iit.edu

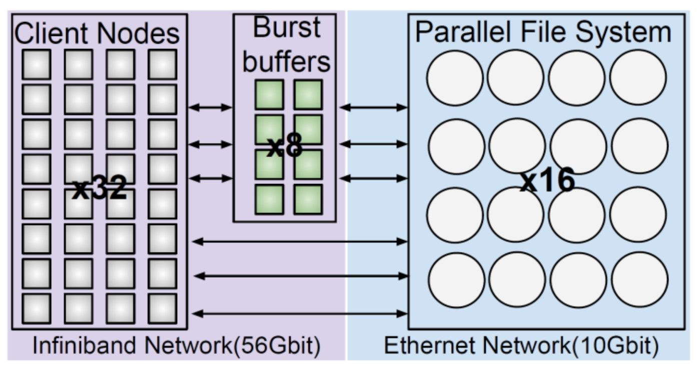
Hermes Critical Components

- RAM management via buckets
- MDM and Messaging via MPI RMA operations



Device	RAM	NVMe	SSD	HDD
Model	M386A4G40DM0	Intel DC P3700	Intel DC S3610	ST9250610NS
Connection	DDR4 2133Mhz	PCIe Gen3 x8	SATA 6Gb/s	SATA 7200rpm
Capacity	128 GB(8GBx16)	1.2 TB	1.6 TB	2.4 TB
Latency	13.5 ns	20 μs	55-66 μs	4.16 ms
Max Read BW	13000 MB/s	2800 MB/s	550 MB/s	115 MB/s
Max Write BW	10000 MB/s	1900 MB/s	500 MB/s	95 MB/s
Test Config	32x client nodes	RamFS emulated	8x burst buffers	16x PFS servers
ReadBW tested	92647 MB/s	38674 MB/s	3326 MB/s	883 MB/s
WriteBW tested	86496 MB/s	33103 MB/s	2762 MB/s	735 MB/s

- Testbed: Chameleon System
- Appliance: Bare Metal
- OS: Centos 7.1
- Storage:
  - OrangeFS 2.9.6
  - Redis 4.0.6
  - Memcached 1.4.36
  - NATS 1.0.4
- MPI: Mpich 3.2
- Programs:
  - Synthetic benchmark
  - VPIC
  - HACC
- Buffering Platforms:
  - Cray’s DataWarp
  - LBNL Data Elevator



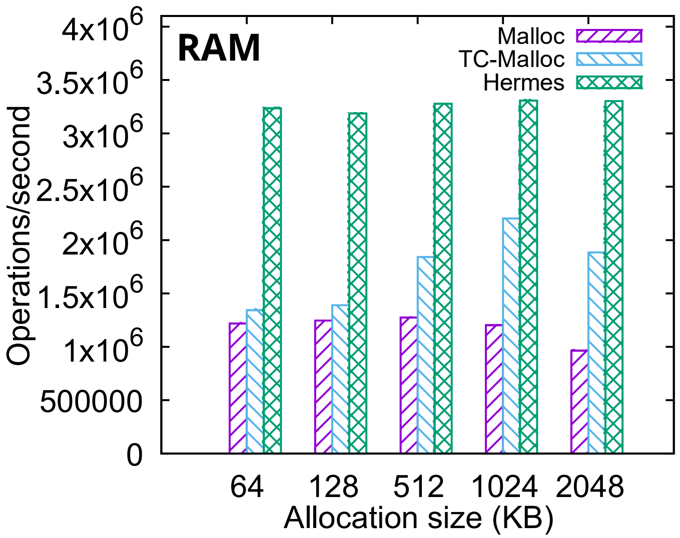
Hermes: A Heterogeneous-Aware Multi-Tiered Distributed I/O Buffering System

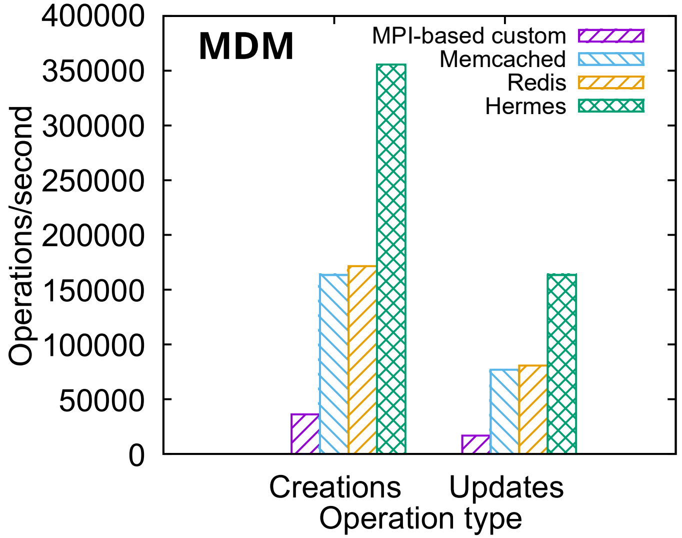
Anthony Kougkas, akougkas@hawk.iit.edu

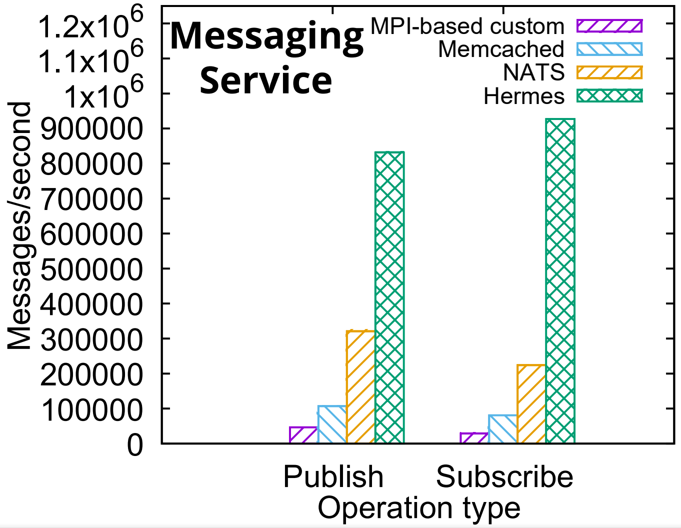
Testbed and Configurations

Benchmarks

- MPI shared dynamic memory window exposed in all nodes
- *MPI_Put(), MPI_Get()*
  - If RDMA is present, MPI uses it
- No need for dedicated server
- Indexing of windows for fast querying
- Complex data structures
- Update operations use MPI_EXCLUSIVE which ensure FIFO consistency
- Entire window with its index is mmap'ed for fault tolerance



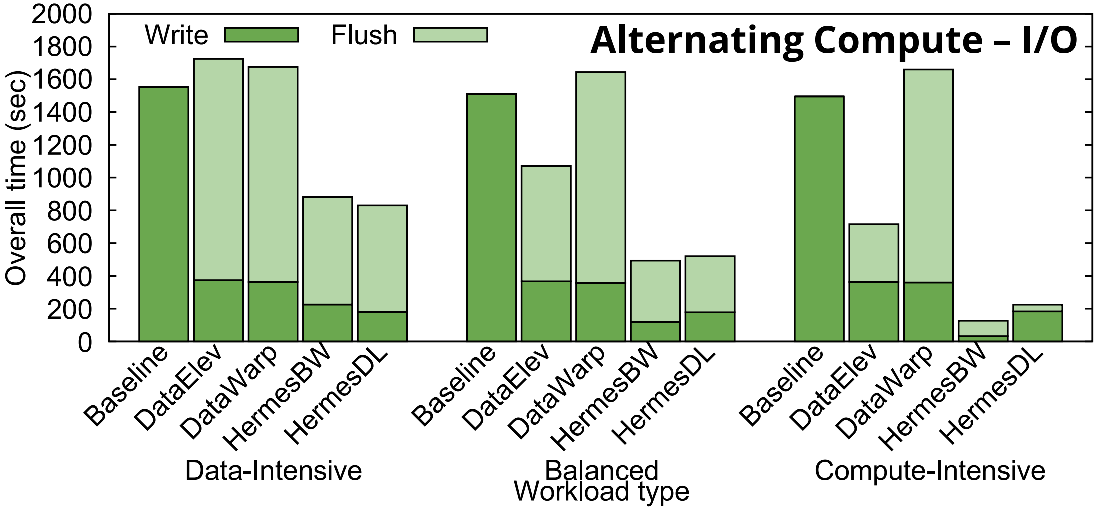




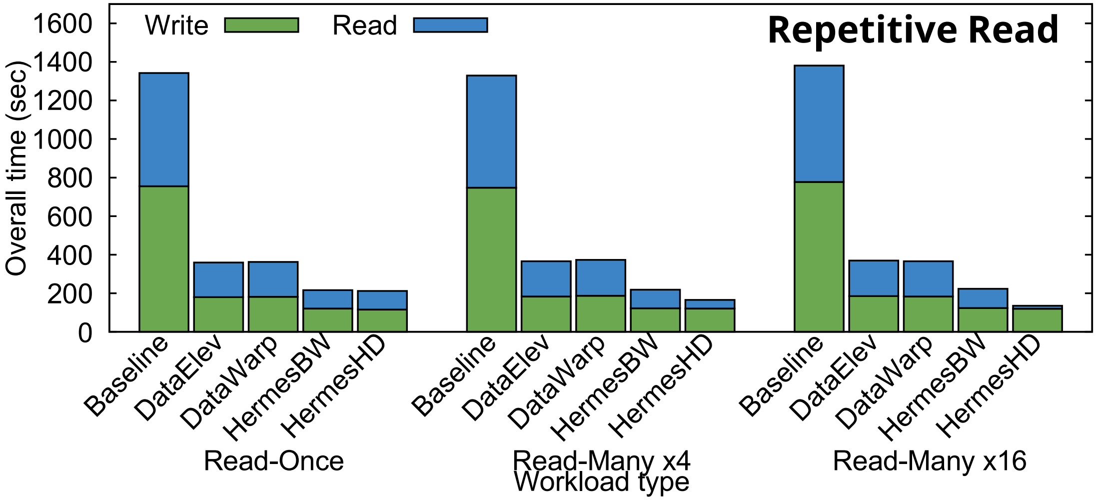
- 1 million fwrite() of various size and measured memory ops/sec
- 1 million metadata operations and measure MDM throughput ops/sec
- 1 million queue operations and measure messaging rate msg/sec

Benchmarks

- File-per-process
- 1024 ranks each 64MB
- 16 phases resulting 1TB total I/O
- Alternating Compute – I/O :
  - Data need to persisted
  - Workloads:
    - Data-intensive
    - Balanced
    - Compute-intensive
  - Metric:
    - Overall I/O time (write + flush)
- Repetitive Read:
  - Temporary data
  - Workloads:
    - Read-once: 32MB read 1x time
    - Readx4: 8MB read 4x times
    - Readx16: 2MB read 16x times
  - Metric:
    - Overall I/O time (write + read)



Hermes offers **8x and 2x** higher write performance on average when compared to No Buffering and state-of-the-art buffering platforms



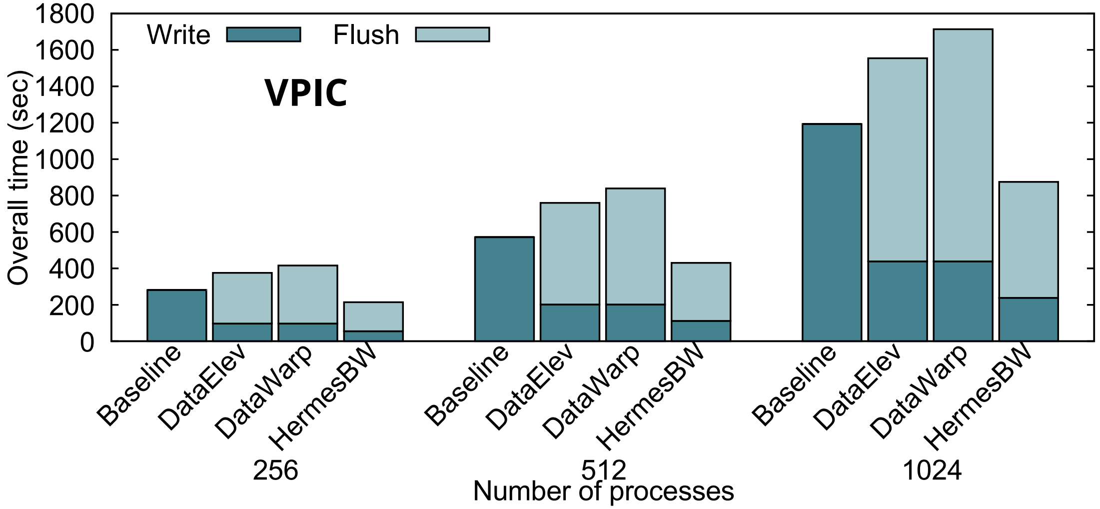
Hermes offers **38x and 11x** higher read performance for repetitive patterns when compared to No Buffering and state-of-the-art buffering platforms

- Hermes hides flushing behind compute (similar to Data Elevator)
- Hermes also hides data movement between layers behind compute
- Hermes leverages the extra layers of the DMSH to offer higher BW

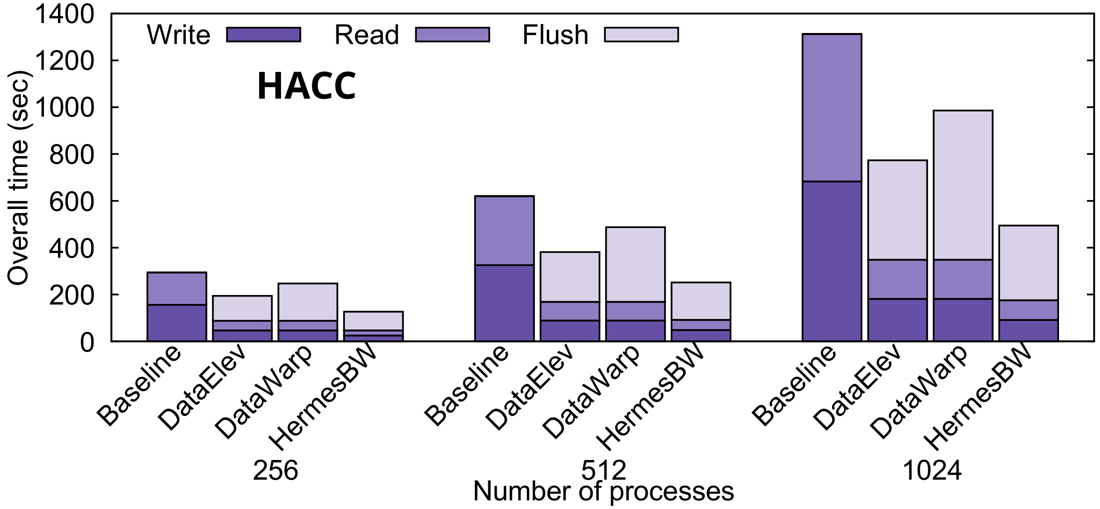
Anthony Kougkas, akougkas@hawk.iit.edu

Benchmarks

- Strong scaled up to 1024 ranks
- 16 time steps
- VPIC:
  - HDF5 files
  - Metric:
    - Overall I/O time (write + flush)
- HACC:
  - MPI - I/O Independent
  - Metric:
    - Overall I/O time (write + read + flush)



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

Q&A

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

A: That is true. We suggest using profiling tools before hand to learn about the application's behavior and tune Hermes. Default policy works great.

Q: How does Hermes integrate to modern HPC environments?

A: As of now, applications link to Hermes (re-compile or dynamic linking). We envision a system scheduler that also incorporates buffering resources.

Q&A

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

A: Hermes' Application Orchestrator was designed for multi-tenant environments. (this work is under review)

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

A: It can be severe but in scenarios where there is some computation in between I/O then it can work nicely to our advantage.

Q&A

Q: What is the metadata size?

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

Q: Is Hermes open source?

A: Hermes will be open sourced by the end of this year. 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.

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: Why not compare jemalloc which is commonly used by data-intensive applications such as LevelDB and Redis?

A: We are looking into it. Great suggestion. We will follow up with this.

Q&A

Q: The authors claim: "We strive for maximizing productivity…" How?

A: We simply believe that the user should not be responsible to manage all layers independently. Hermes can provide a certain level of automation.

Q: What is Hermes API?

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

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.

Q: What is Hermes' DPE complexity?

A: In the order of number of layers

In summary

- **Data movement** among the layers of a deep memory and storage hierarchy is significantly **complex**.
- Each layer of the DMSH is an **independent** system that requires expertise to manage.
- The lack of **automated** data movement between tiers is a burden currently left to the users.
- **Hermes**, is a new, multi-tiered, distributed buffering platform that enables, manages, and supervises I/O buffering into the **DMSH**.
- Hermes boosts I/O performance by more than **8x**.
- Hermes outperforms other buffering platforms by **2x**.

Hermes

A Heterogeneous-Aware Multi-Tiered Distributed I/O Buffering System

Thank you.

Anthony Kougkas

akougkas@hawk.iit.edu

https://sites.google.com/iit.edu/akougkas

National Science Foundation
WHERE DISCOVERIES BEGIN

This work was supported by the National Science Foundation under grants no. CCF-1744317, CNS-1526887, and CNS-0751200.

SCALABLE COMPUTING SOFTWARE LABORATORY

ILLINOIS INSTITUTE OF TECHNOLOGY