Hewlett Packard
Enterprise

Persistent memory: new tier or storage replacement?

Kimberly Keeton and Susan Spence

kimberly.keeton@hpe.com and susan.spence@hpe.com

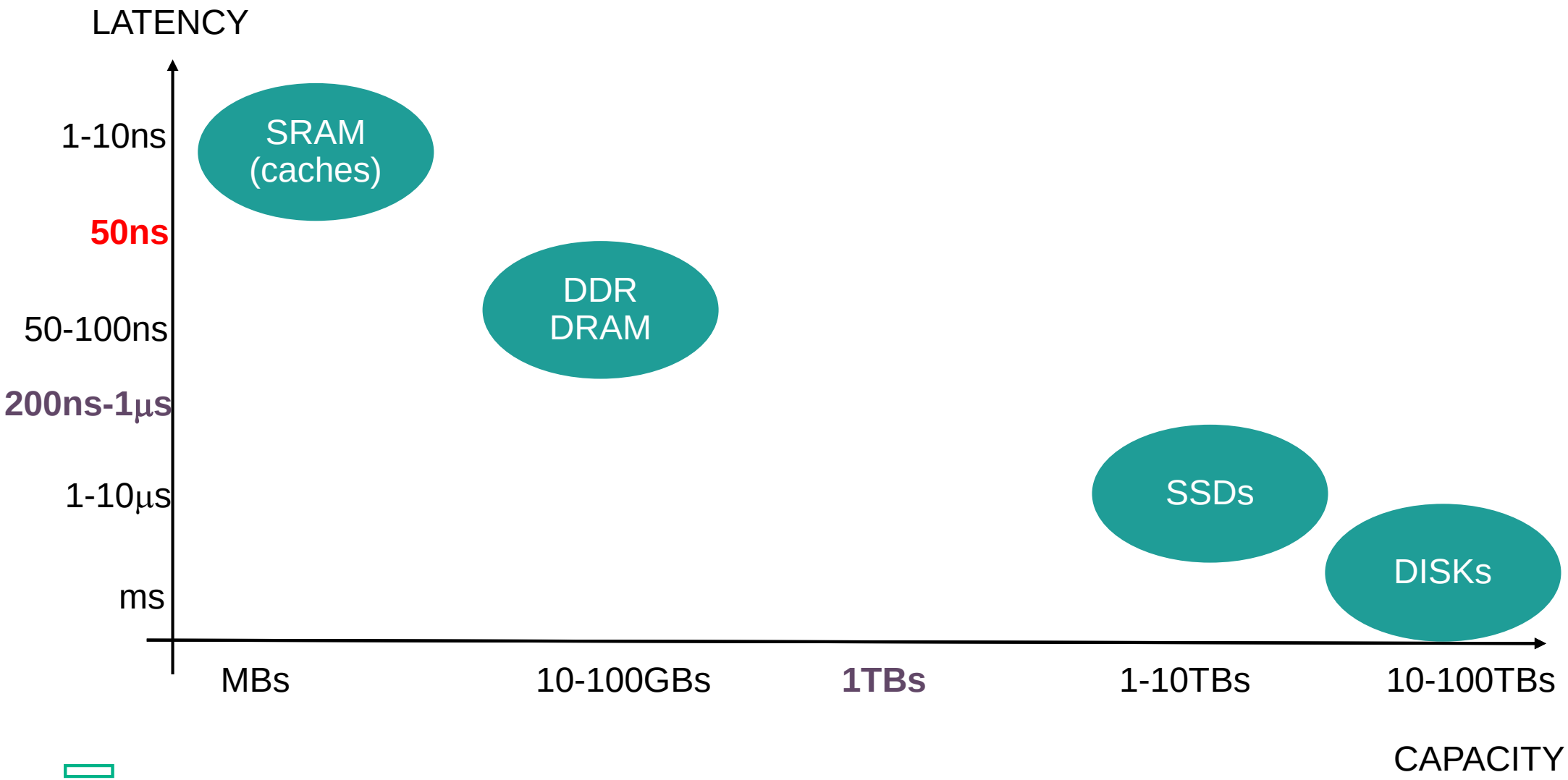
SNIA Storage Developer Conference
September 2017

Hewlett Packard
Labs

Outline

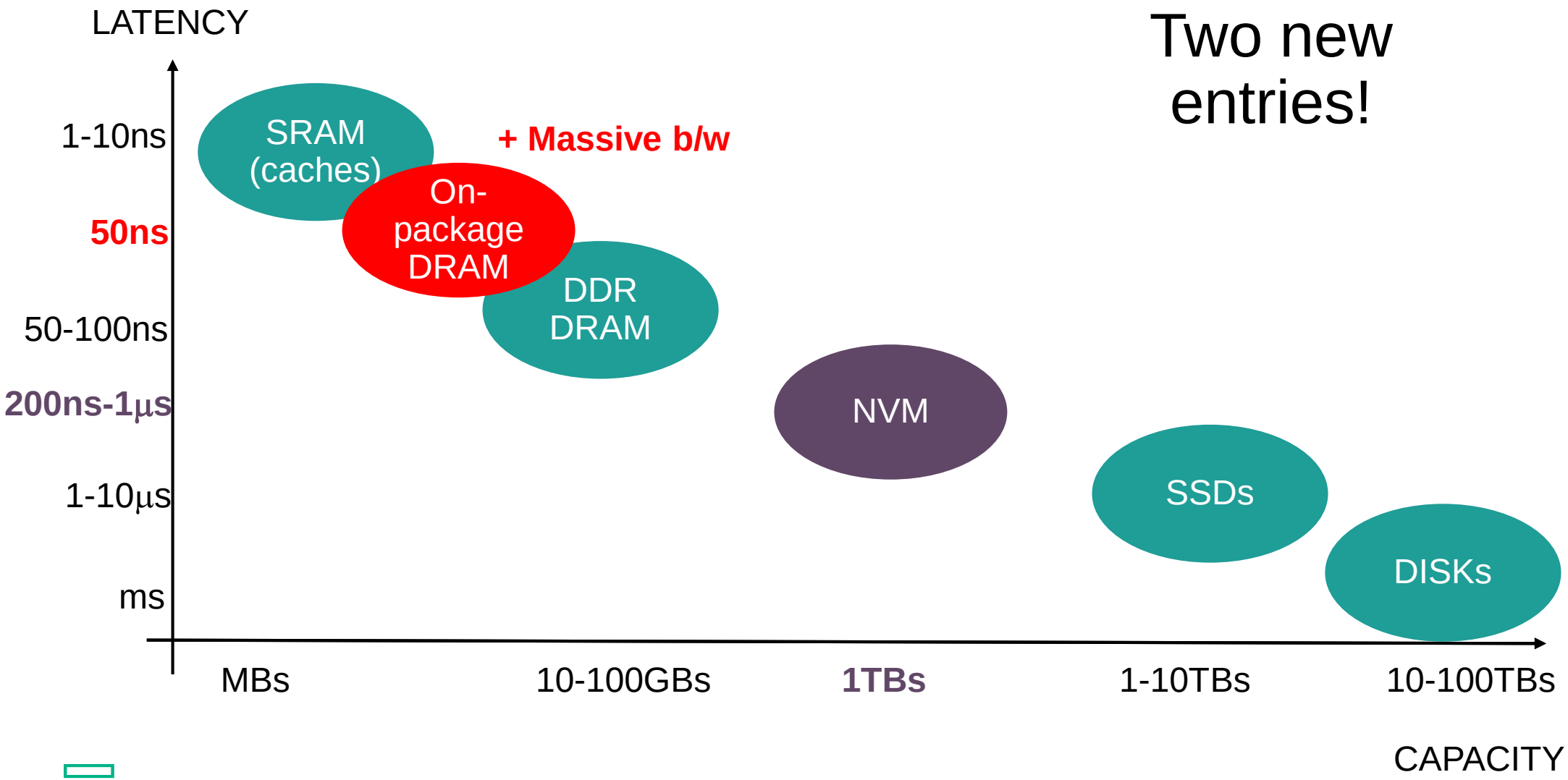
- Persistent memory blurs boundaries between memory and storage
- Memory-Driven Computing (MDC) – context for our work with persistent memory
- Memory-Driven Computing software
- Challenges for persistent memory as storage
- Summary

Memory + storage hierarchy technologies



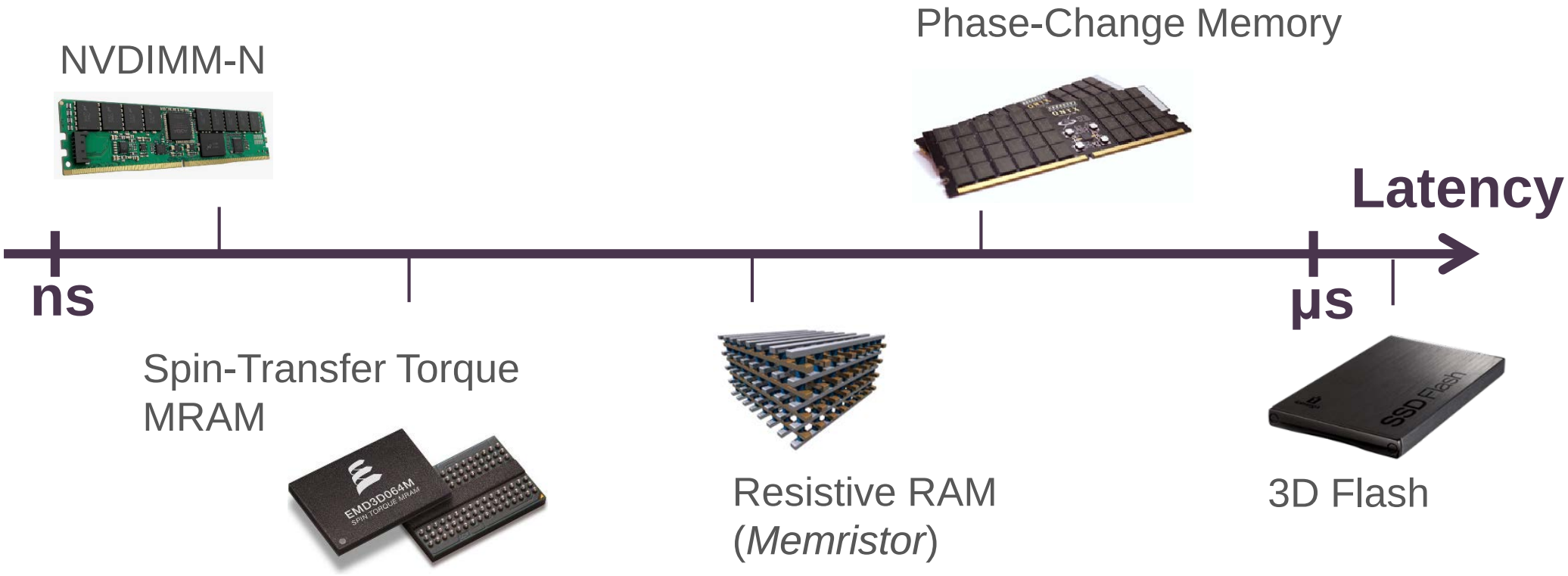
Hewlett Packard Enterprise

Memory + storage hierarchy technologies



Hewlett Packard Enterprise

Non-Volatile Memory (NVM)



– Persistently stores data
– Access latencies comparable to DRAM
– Byte addressable (load/store) rather than block addressable (read/write)
– Some NVM technologies more energy efficient and denser than DRAM

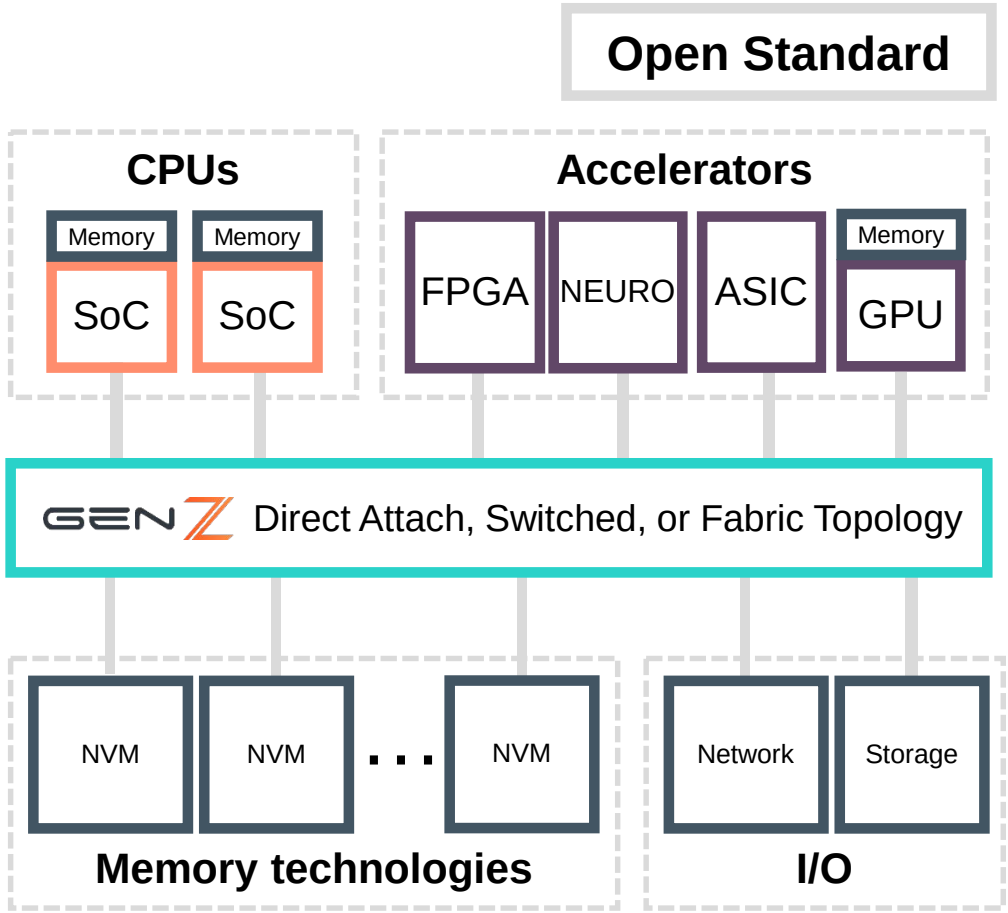
Haris Volos, et al. "Aerie: Flexible File-System Interfaces to Storage-Class Memory," *Proc. EuroSys 2014*.

Hewlett Packard Enterprise

Gen-Z: open systems interconnect standard

http://www.genzconsortium.org

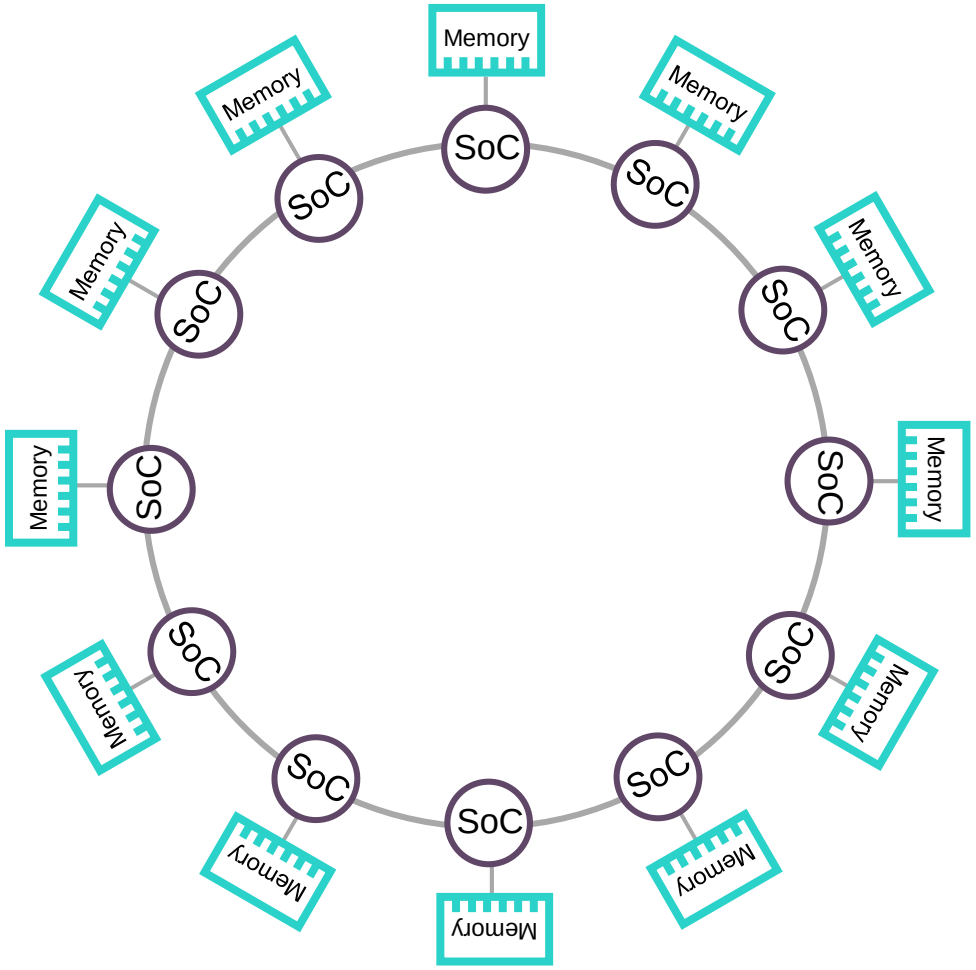
- Open standard for memory-semantic interconnect
- Members: 30+ companies covering SoC, memory, I/O, networking, mechanical, system software, etc.
- Motivation
  - Emergence of low-latency storage class memory
  - Demand for large capacity, rack-scale resource pools and multi-node architectures
- Memory semantics
  - All communication as memory operations (load/store, put/get, atomics)
- High performance
  - Tens to hundreds GB/s bandwidth
  - Sub-microsecond load-to-use memory latency
- *Draft spec available for public download*



Hewlett Packard Enterprise

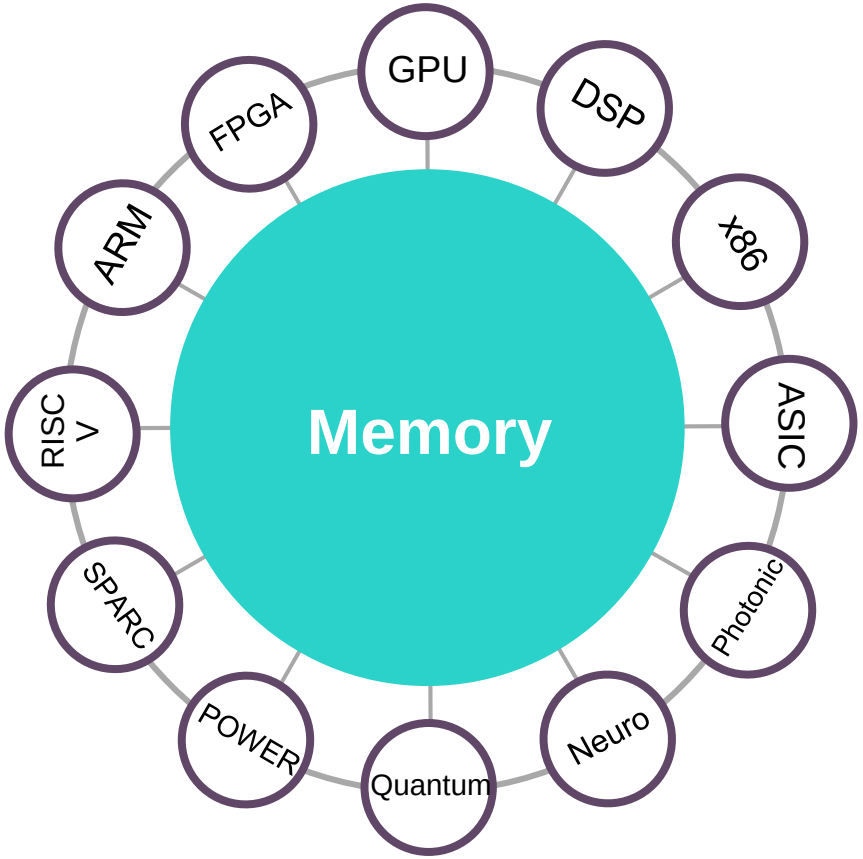
Today's architecture

From processor-centric computing



Future architecture

Memory-Driven Computing



Hewlett Packard Enterprise

Core Memory-Driven Computing components

Fast, persistent memory

Fast memory fabric

Task-specific processing

New and Adapted software

Hewlett Packard Enterprise

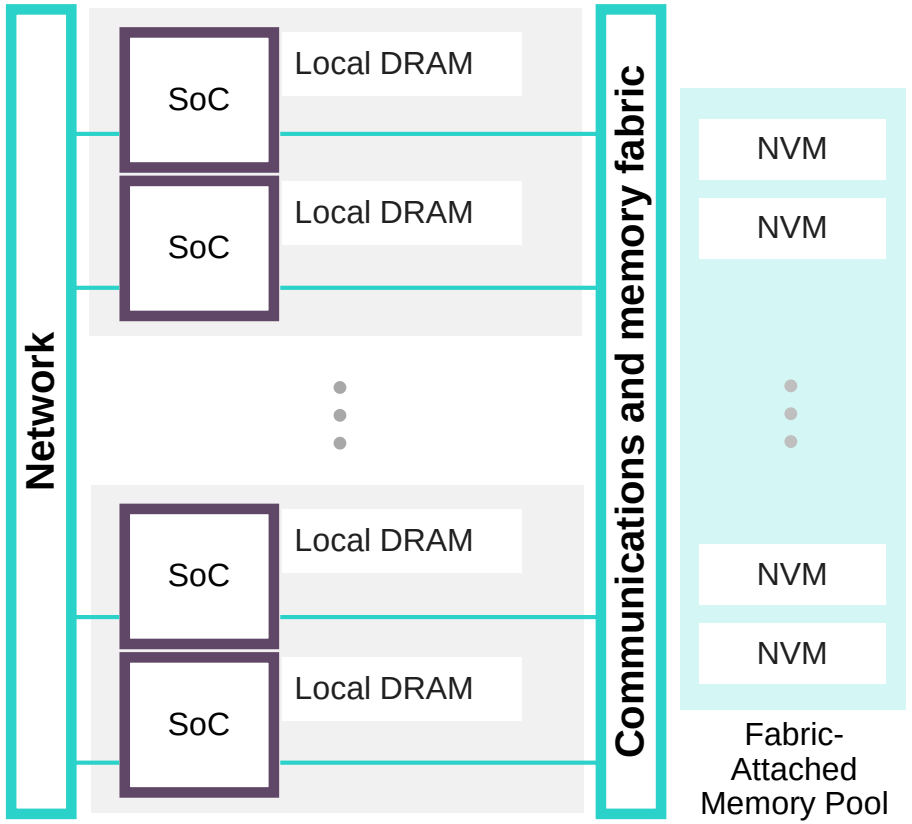
HPE introduces the world's largest single-memory computer

The prototype contains 160 terabytes of fabric-attached memory

- 160 TB of shared memory spread across 40 physical nodes, interconnected using a high-performance fabric protocol
- An optimized Linux-based operating system running on ThunderX2, Cavium's flagship second generation dual socket capable ARMv8-A workload optimized System on a Chip
- Photonics/Optical communication links, including the new X1 photonics module, are online and operational
- Software programming tools designed to take advantage of abundant of persistent memory

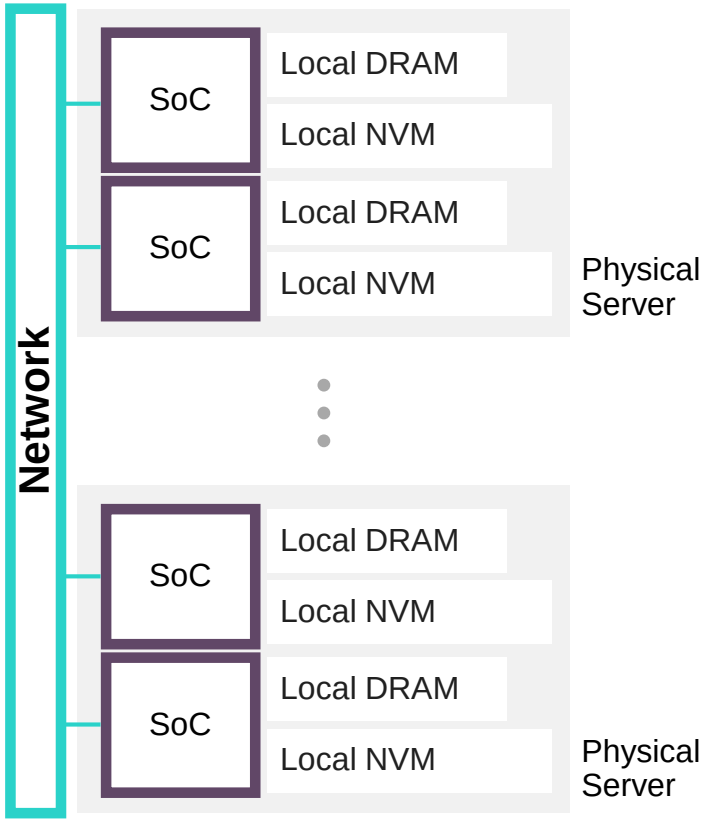
Hewlett Packard Enterprise

Putting it all together: Memory-Driven Computing

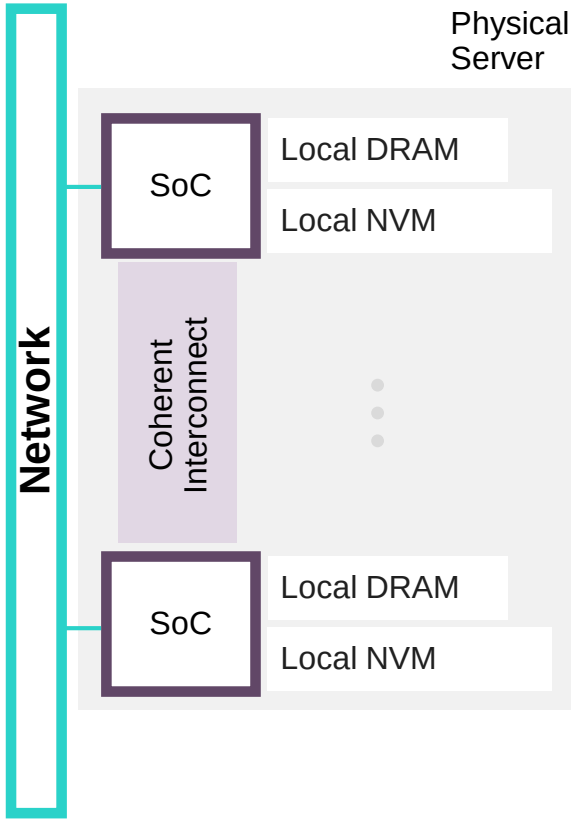


- **Converging memory and storage**
  - Byte-addressable NVM replaces hard drives and SSDs
- **Resource disaggregation leads to high capacity shared memory pool**
  - Fabric-attached memory pool is accessible by all compute resources
  - Low diameter networks provide near-uniform low latency
- **Local volatile memory provides lower latency, high performance tier**
- **Distributed heterogeneous compute resources**
- **Software**
  - Memory-speed persistence
  - Direct, unmediated access to all fabric-attached NVM across the memory fabric
  - Non-coherent accesses between compute nodes

Memory-Driven Computing in context



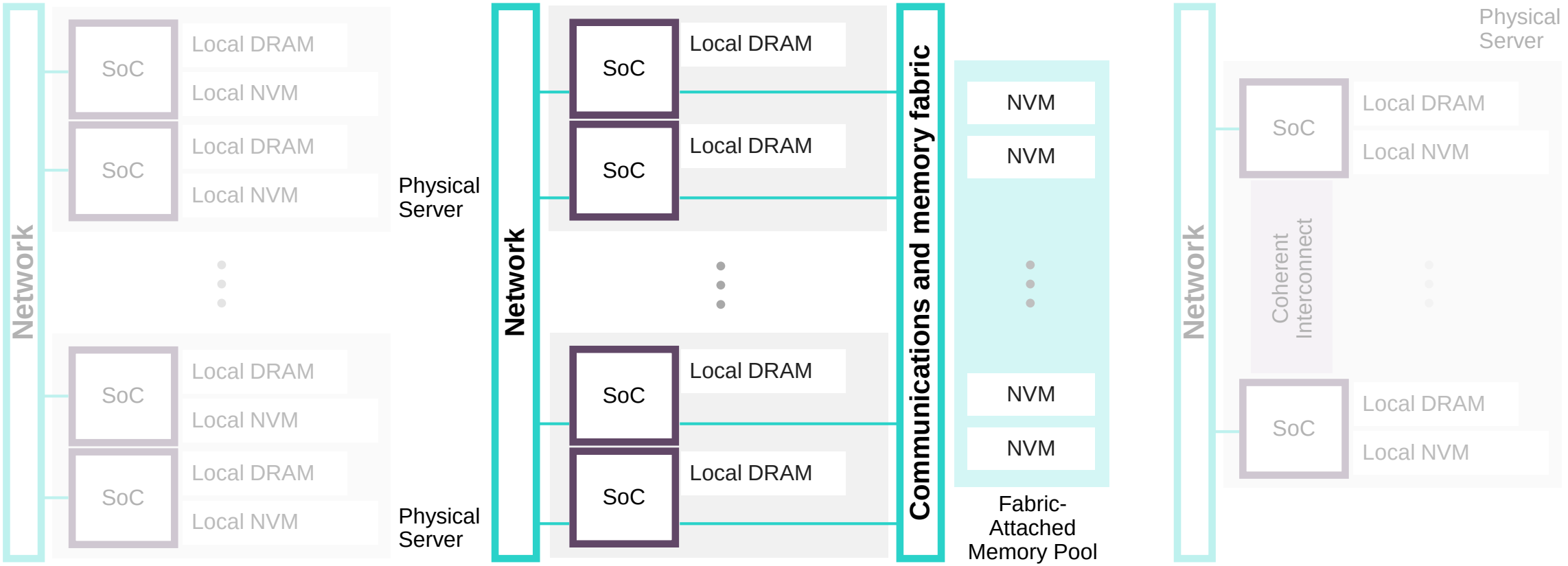
Shared nothing



Shared everything

Hewlett Packard Enterprise

Memory-Driven Computing in context



Hewlett Packard Enterprise

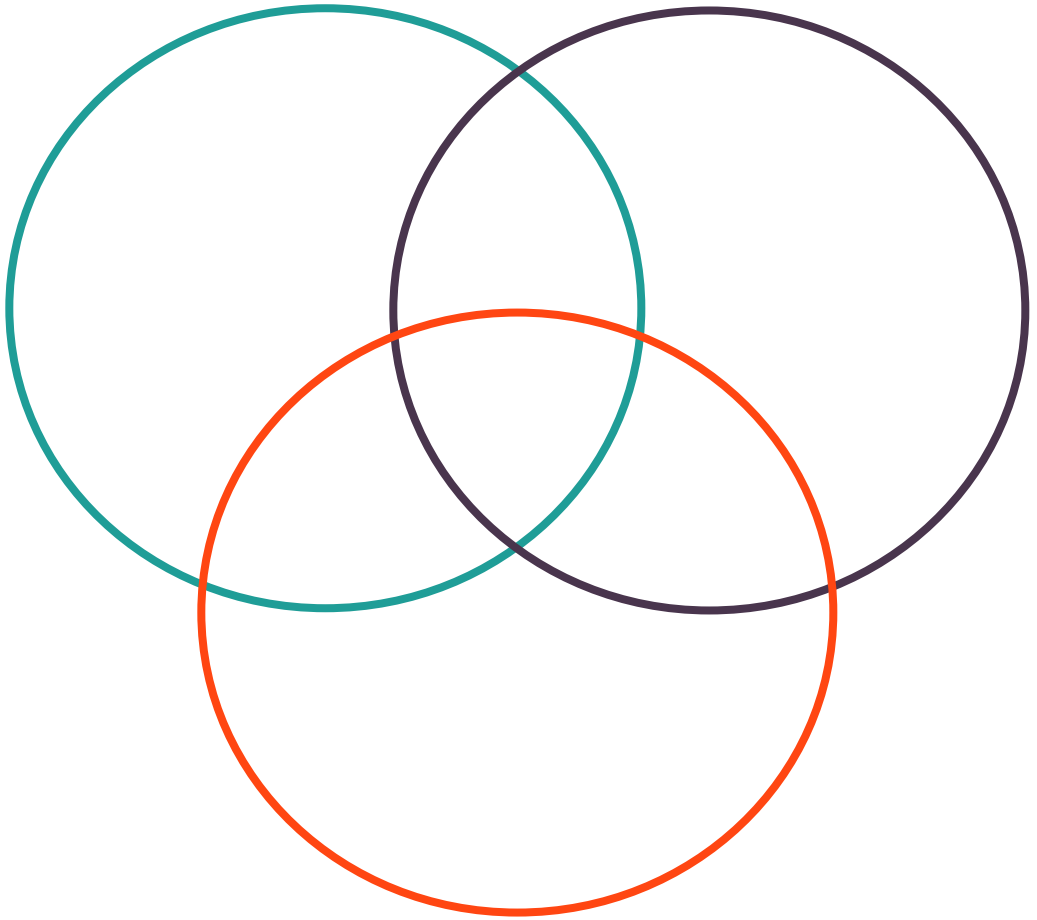
Questions driving work on MDC software

How do I ...

- use persistent memory?
- take full advantage of byte-addressable persistent memory?
- share data safely in a large, globally-addressable pool of memory?
- ensure data consistency in the face of failures and errors?
- scale effectively to use our large-memory, many-core systems?

Memory-Driven Computing delivered with systems software

Memory is large



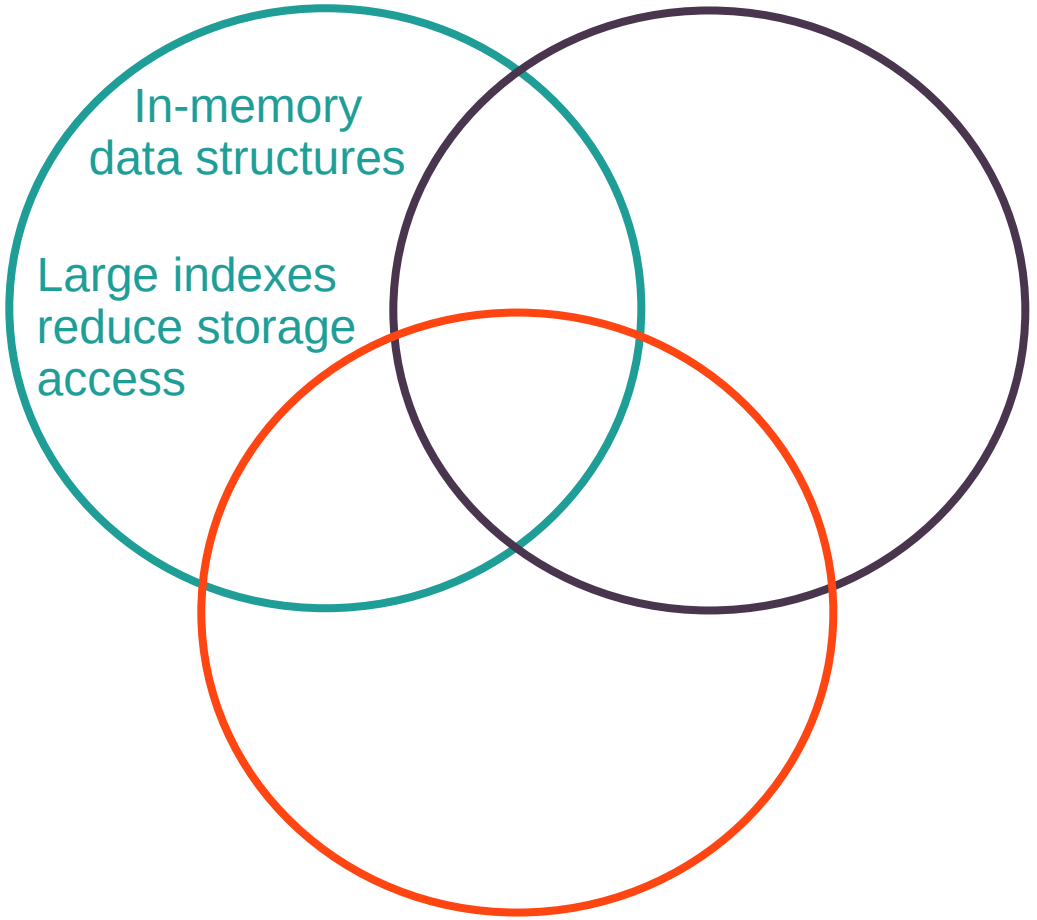
Memory is shared non-coherently over fabric

Memory is persistent

Hewlett Packard Enterprise

Memory-Driven Computing delivered with systems software

Memory is large



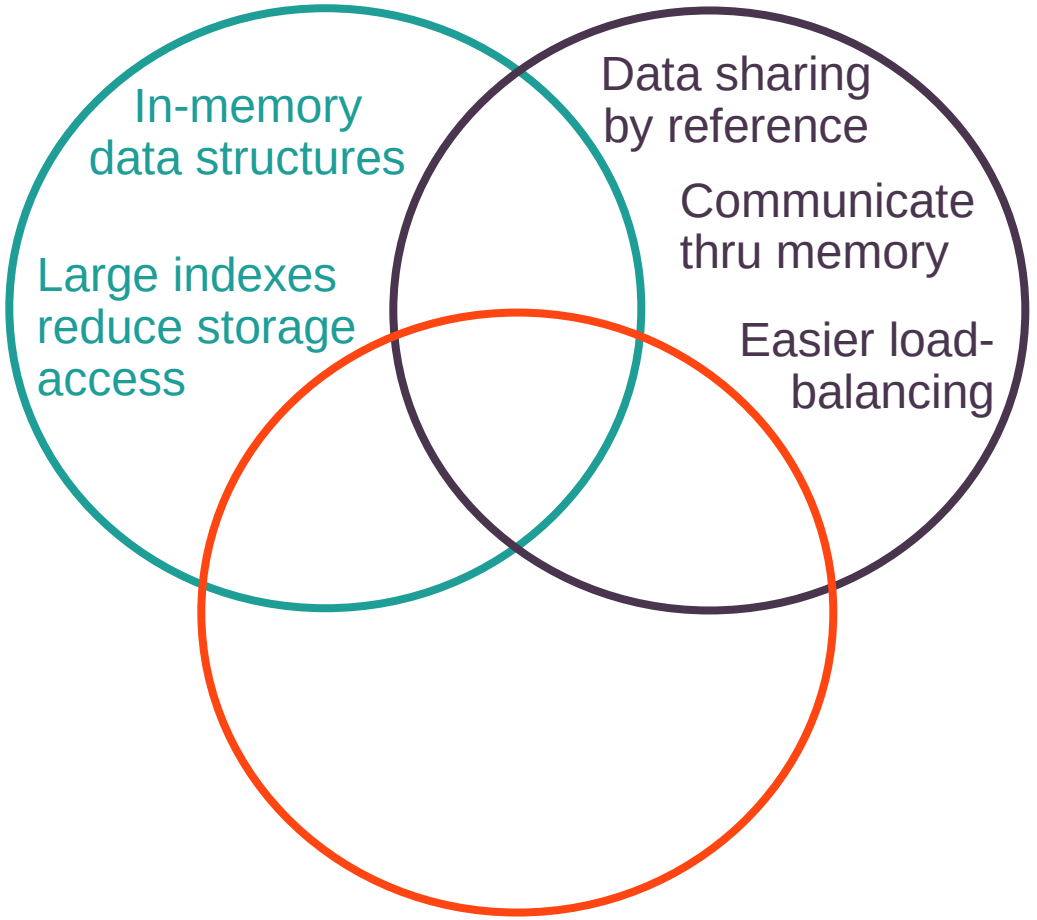
Memory is shared non-coherently over fabric

Memory is persistent

Hewlett Packard Enterprise

Memory-Driven Computing delivered with systems software

Memory is large



Memory is shared non-coherently over fabric

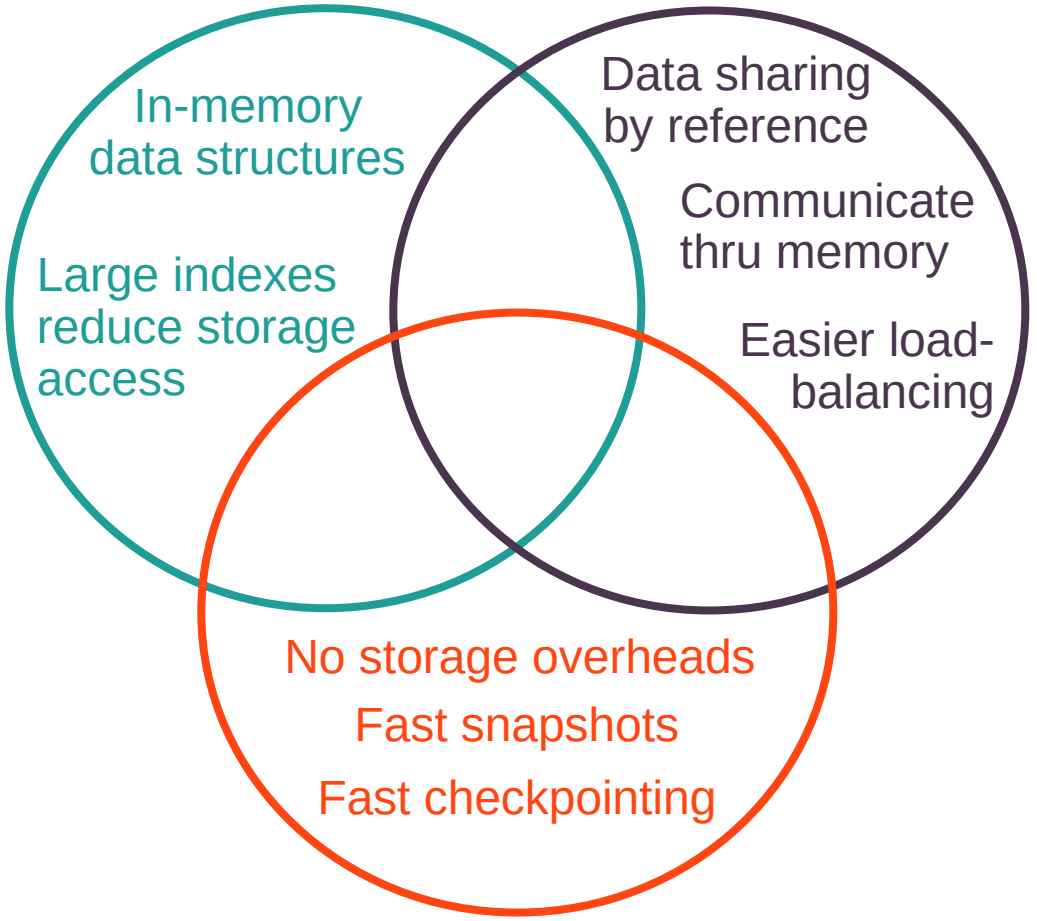
Memory is persistent

Hewlett Packard Enterprise

Memory-Driven Computing delivered with systems software

Memory is large

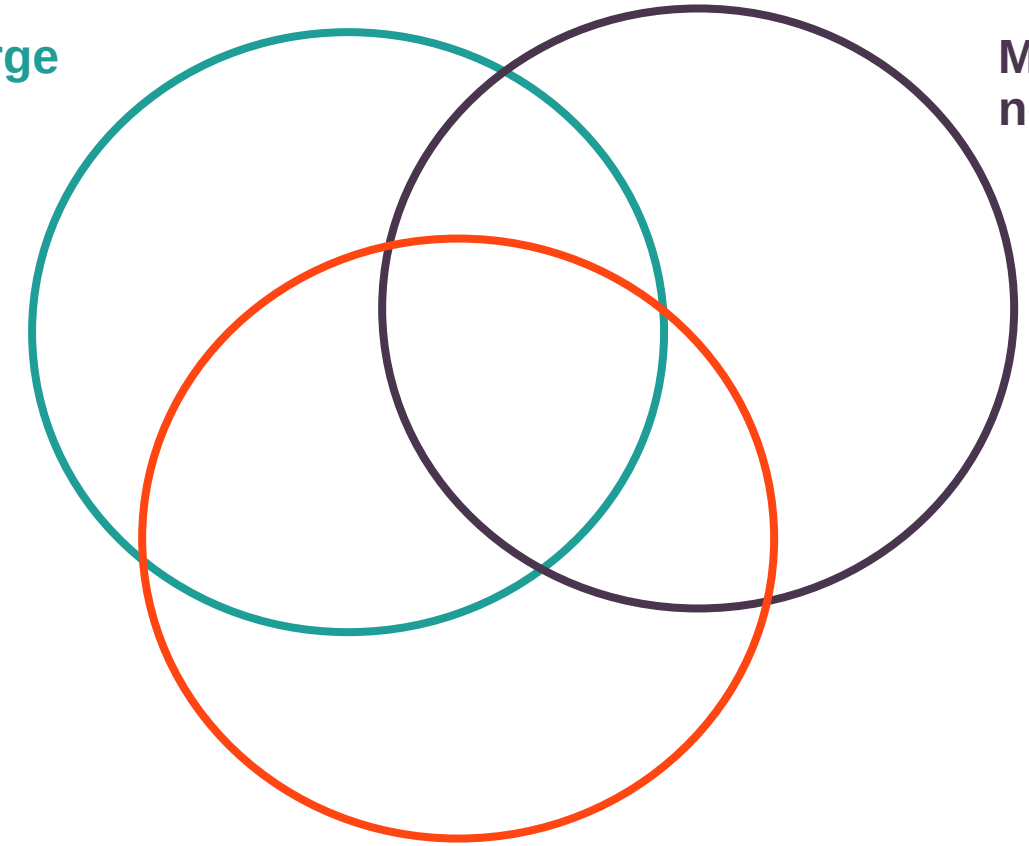
Memory is shared non-coherently over fabric



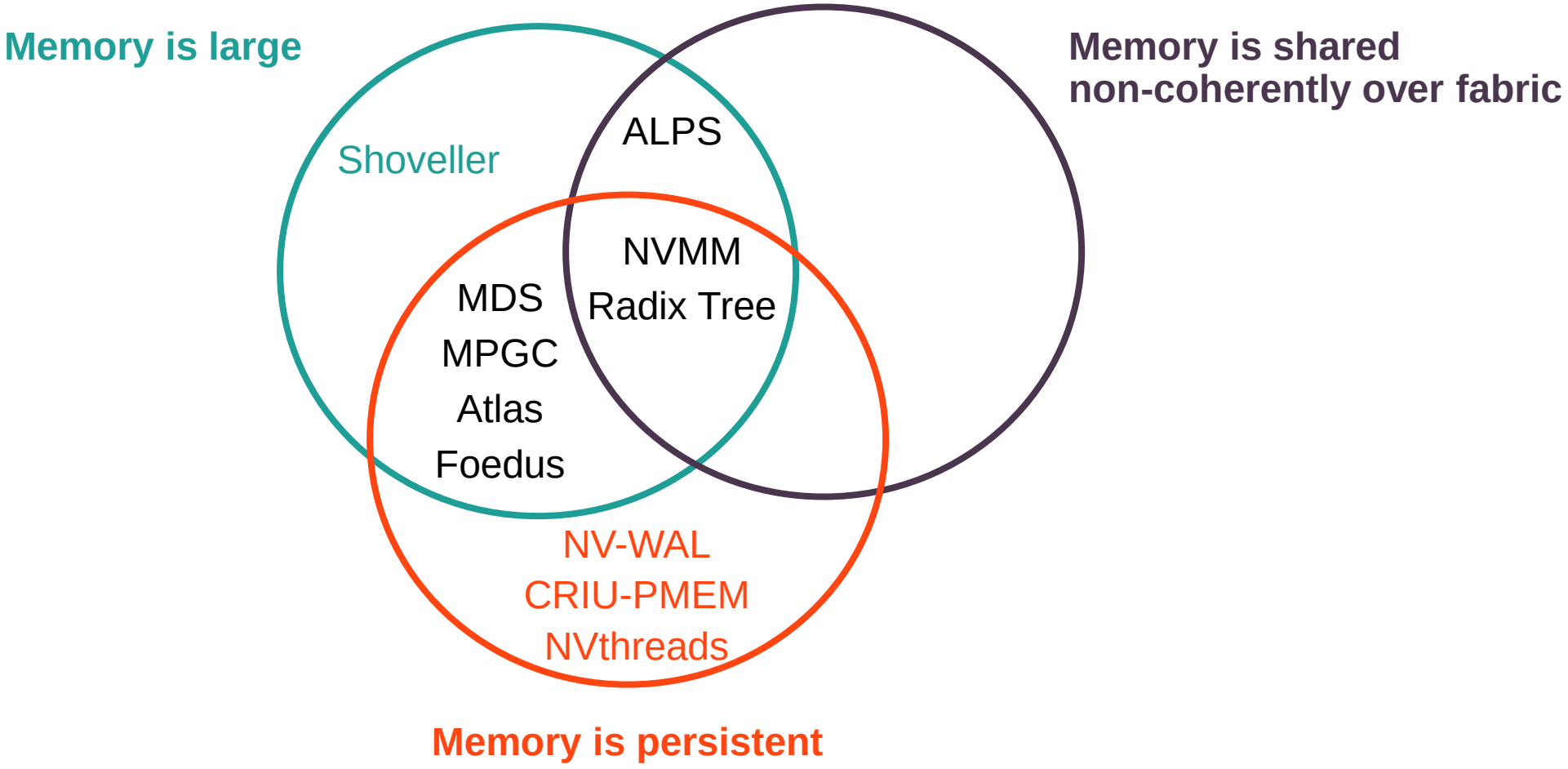
Memory is persistent

Hewlett Packard Enterprise

Memory-Driven Computing Developer Toolkit



Memory-Driven Computing Developer Toolkit

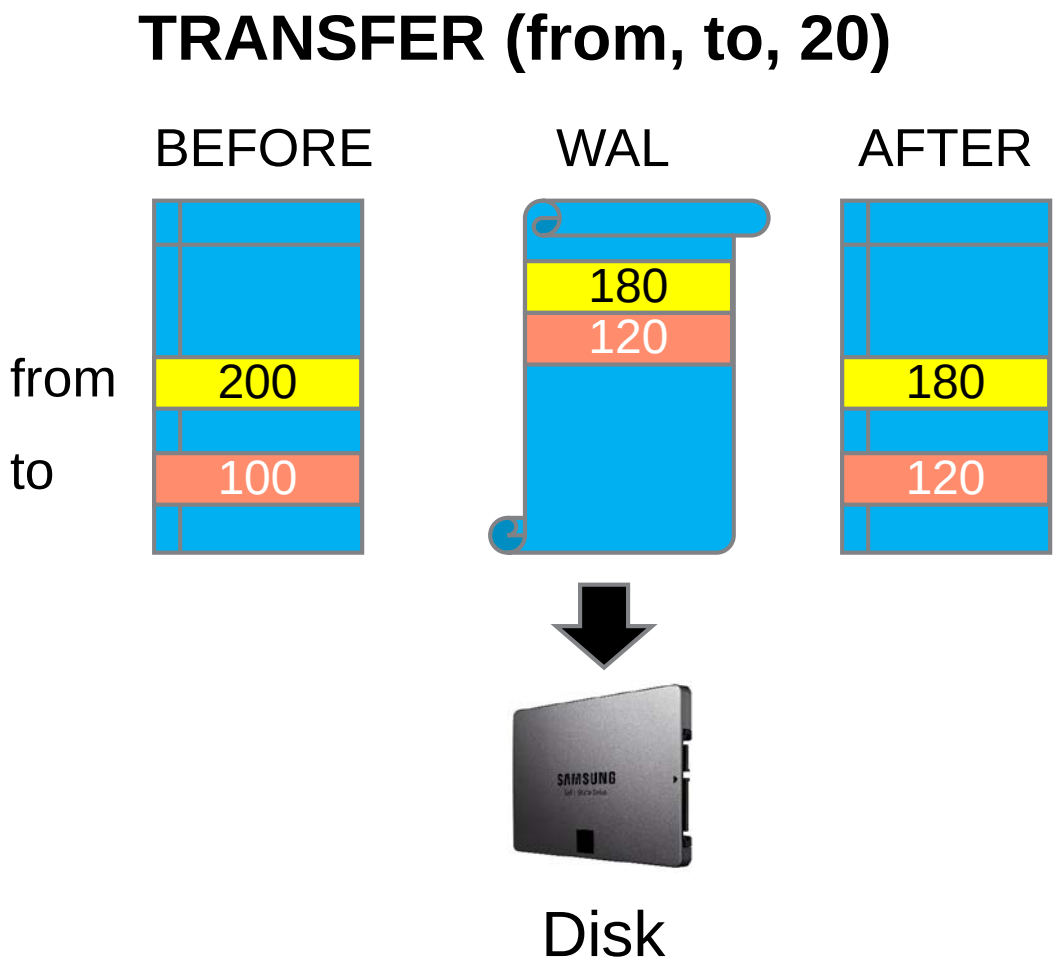


Libnvwal: Write-Ahead Logging Library

– Key mechanism for atomicity and durability
  – Atomicity: Record all modifications in WAL, and
  – Durability: Persist WAL before applying

```
TRANSFER (from, to, amount)
  transaction {
    from = from – amount
    to = to + amount
  }
```

– Persistence latency critical to transaction performance



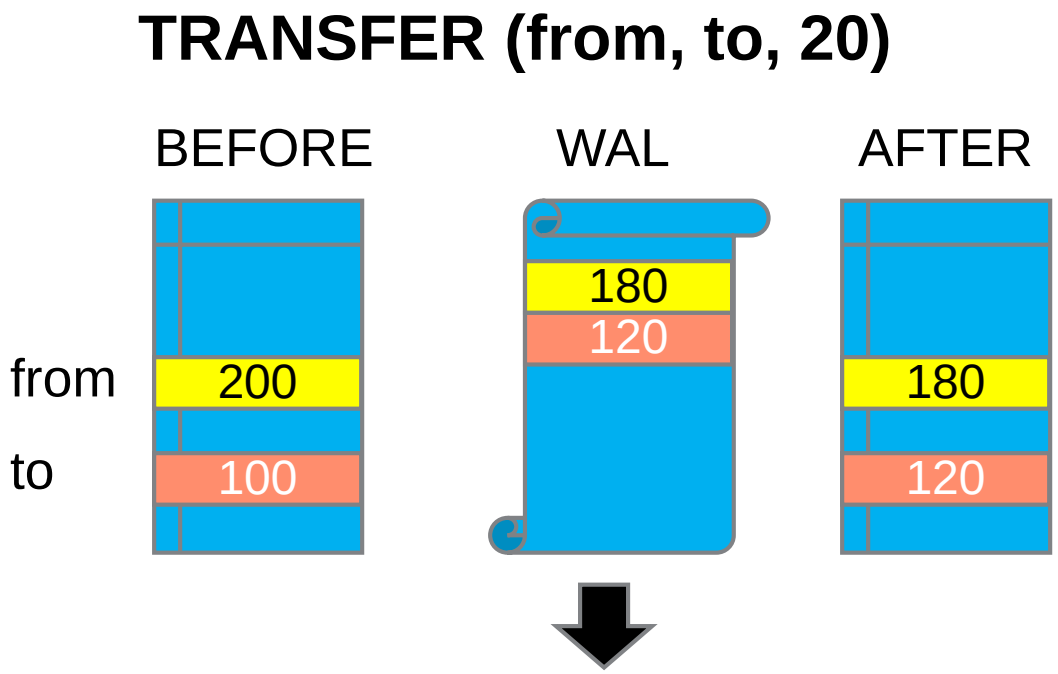
Libnvwal: Write-Ahead Logging Library

– Key mechanism for atomicity and durability
  – Atomicity: Record all modifications in WAL, and
  – Durability: Persist WAL before applying

```
TRANSFER (from, to, amount)
  transaction {
    from = from – amount
    to = to + amount
  }
```

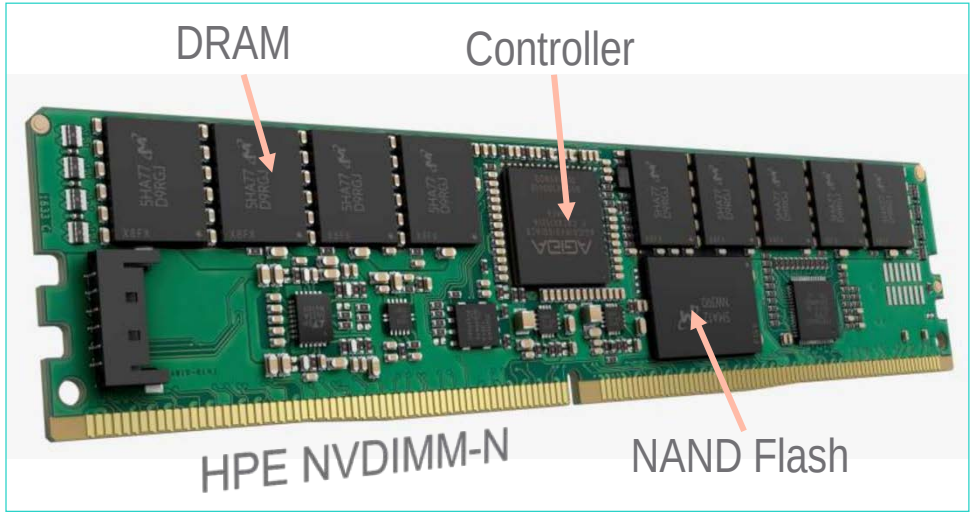
– Persistence latency critical to transaction performance

– Non-volatile memory DIMM (NVDIMM) enables low latency logging



Libnvwal: Hybrid NVDIMM/Disk Write-Ahead Logging

– NVDIMM enables low latency persistence
  – DRAM: Best performance/lowest latency for fast data access
  – NAND Flash: Persistent store for the NVDIMM
– But limited capacity (8GB NVDIMM today) requires truncating log
  – Requires checkpointing database
  – Precludes archiving log for disaster recovery

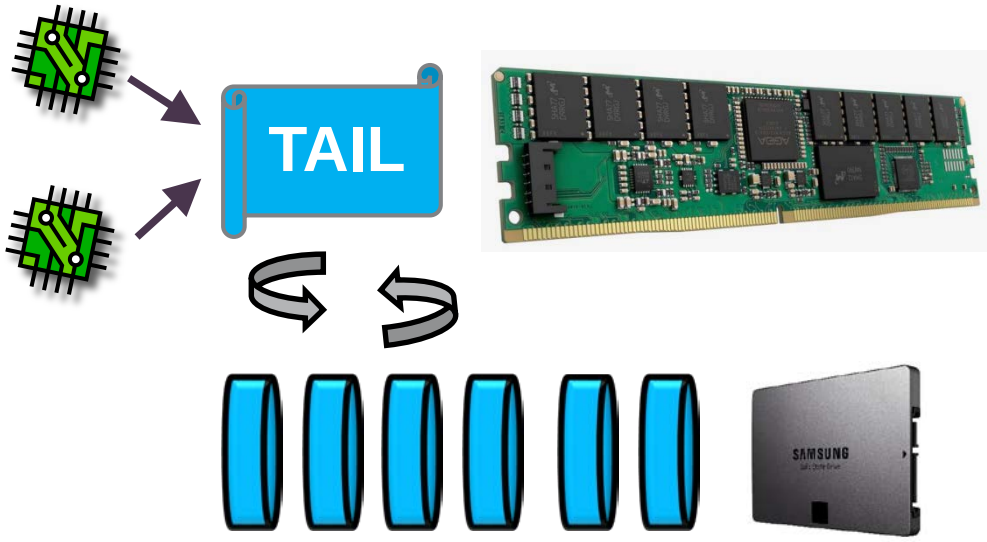


Libnvwal: Hybrid NVDIMM/Disk Write-Ahead Logging

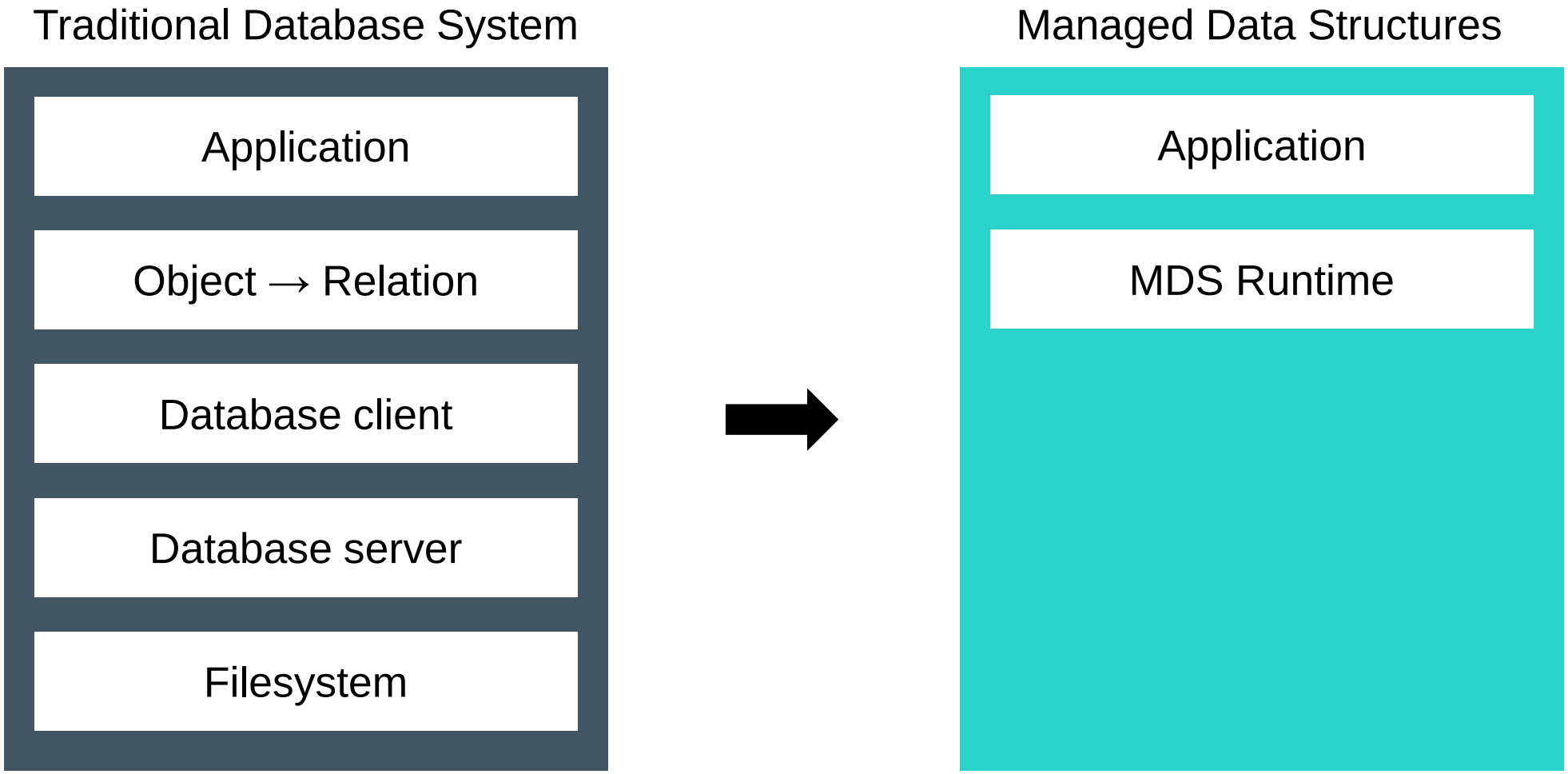
– NVDIMM enables low latency persistence
  – DRAM: Best performance/lowest latency for fast data access
  – NAND Flash: Persistent store for the NVDIMM
– But limited capacity (8GB NVDIMM today) requires truncating log
  – Requires checkpointing database
  – Precludes archiving log for disaster recovery

Libnvwal

– Stores log tail in NVDIMM for low-latency and de-stages log to disk for capacity
– Relieves application from implementing complex rotation logic
  – Non-trivial interaction between NVM buffers, file system, log metadata durability, and flushing threads
– Improves MySQL OLTP performance by up to 106% (sysbench)



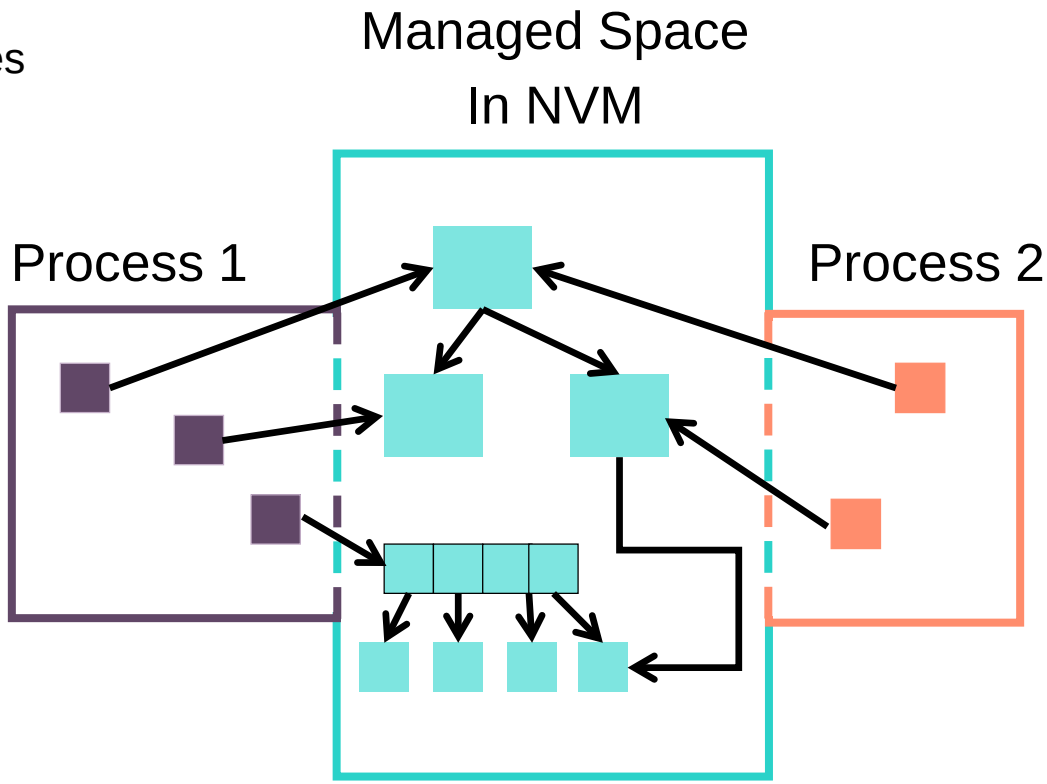
Fewer software layers



Managed Data Structures (MDS)

Simplify programming on persistent in-memory data

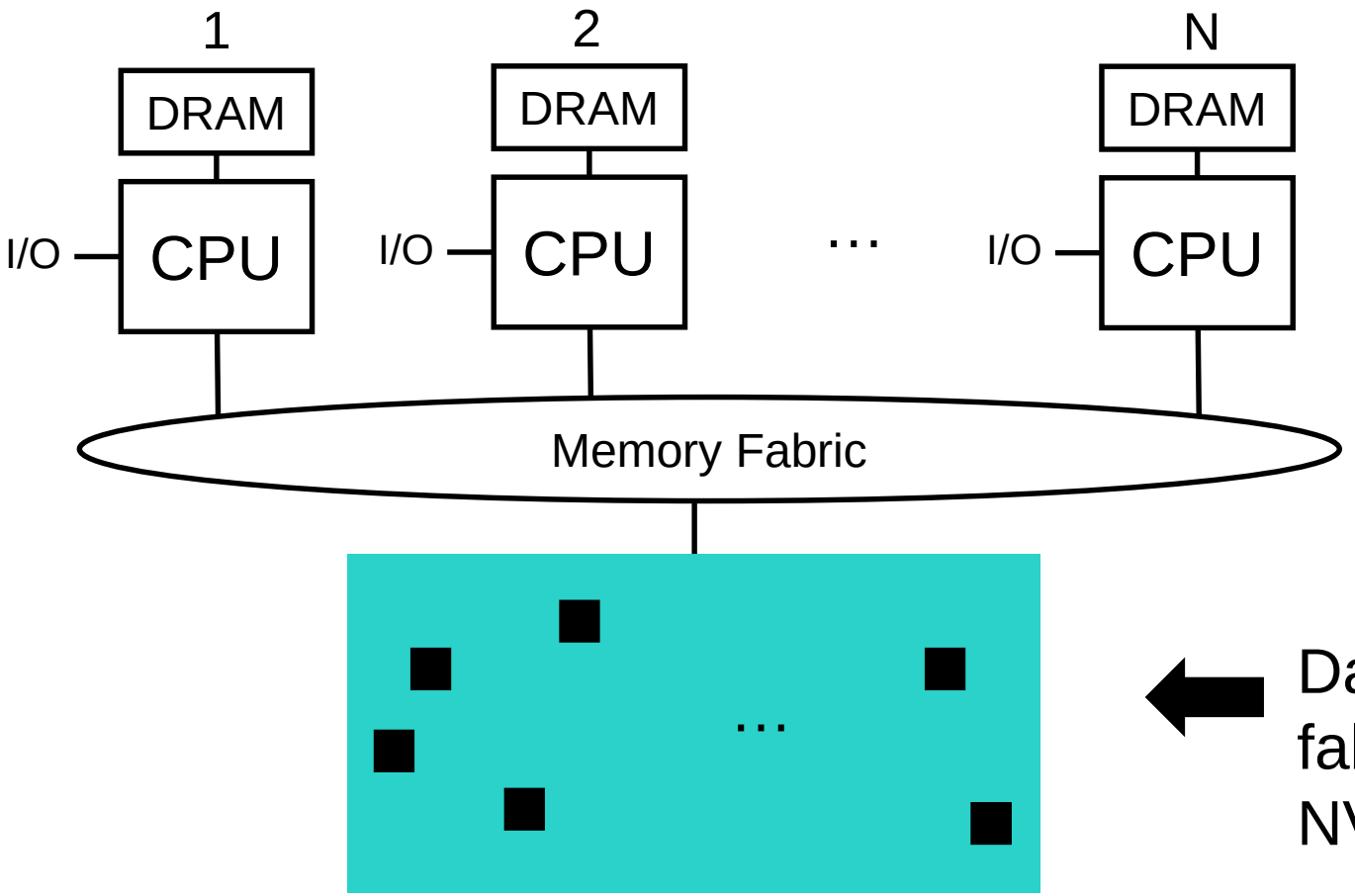
- Ease of Programming
  - Programmer manages only application-level data structures
    - Record, Array, Map, Set, Graph, List, Queue, …
    - MDS data structures are automatically persisted in NVM
  - APIs in multiple programming languages: Java, C++
    - Programmer access through references to data
    - Direct reads and writes
- Ease of Data Sharing
  - Just pass a reference
    - Each program treats the data as local to the program
  - High-level concurrency controls
    - Ensure consistent data in the face of data sharing by multiple threads/processes



Hewlett Packard Enterprise

“Shared something” Key-Value Store (KVS)

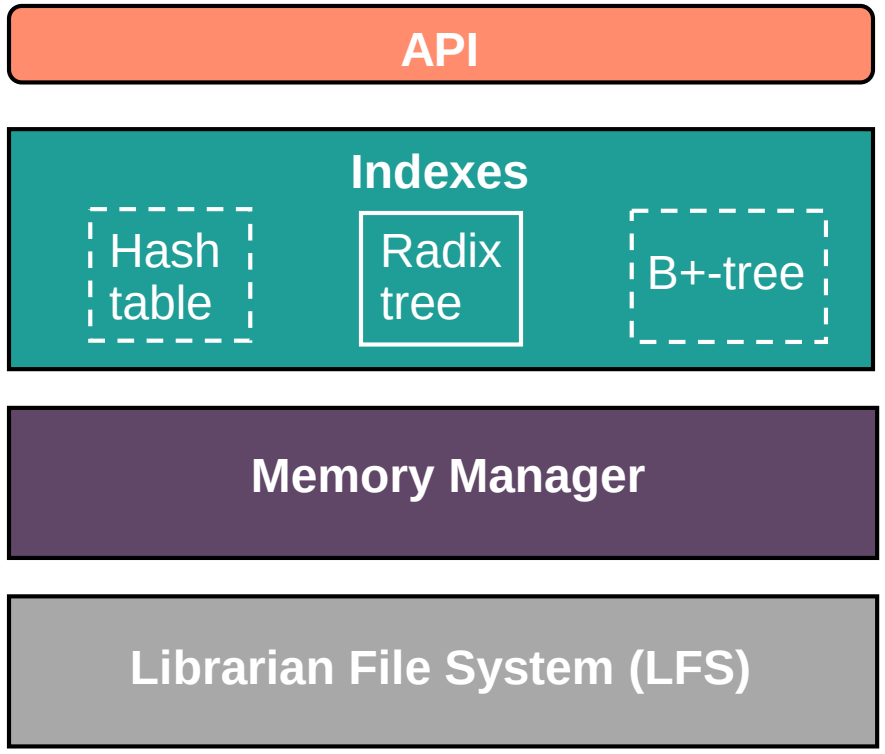
Leverage large pool of fabric-attached memory and fabric atomics

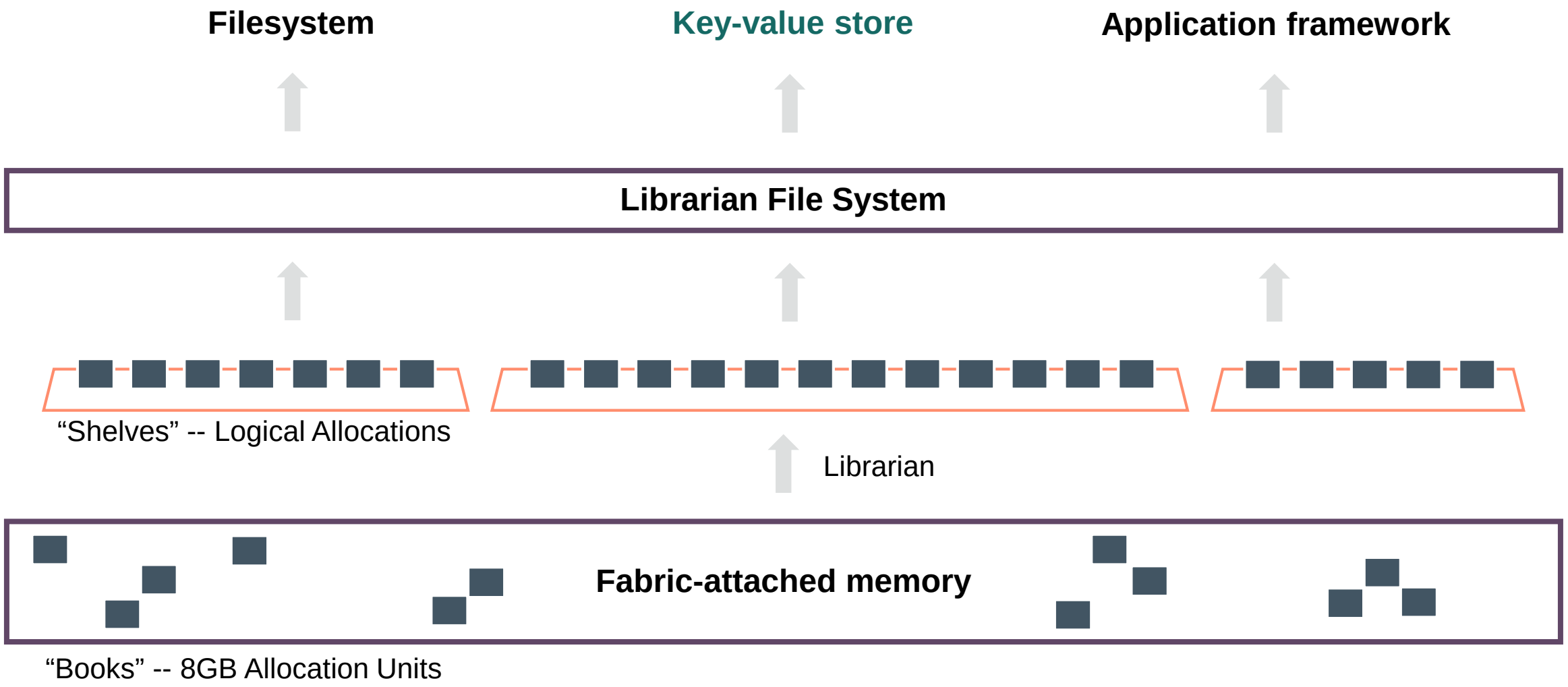


Data stored in fabric-attached NVM

High-level architecture for shared something KVS

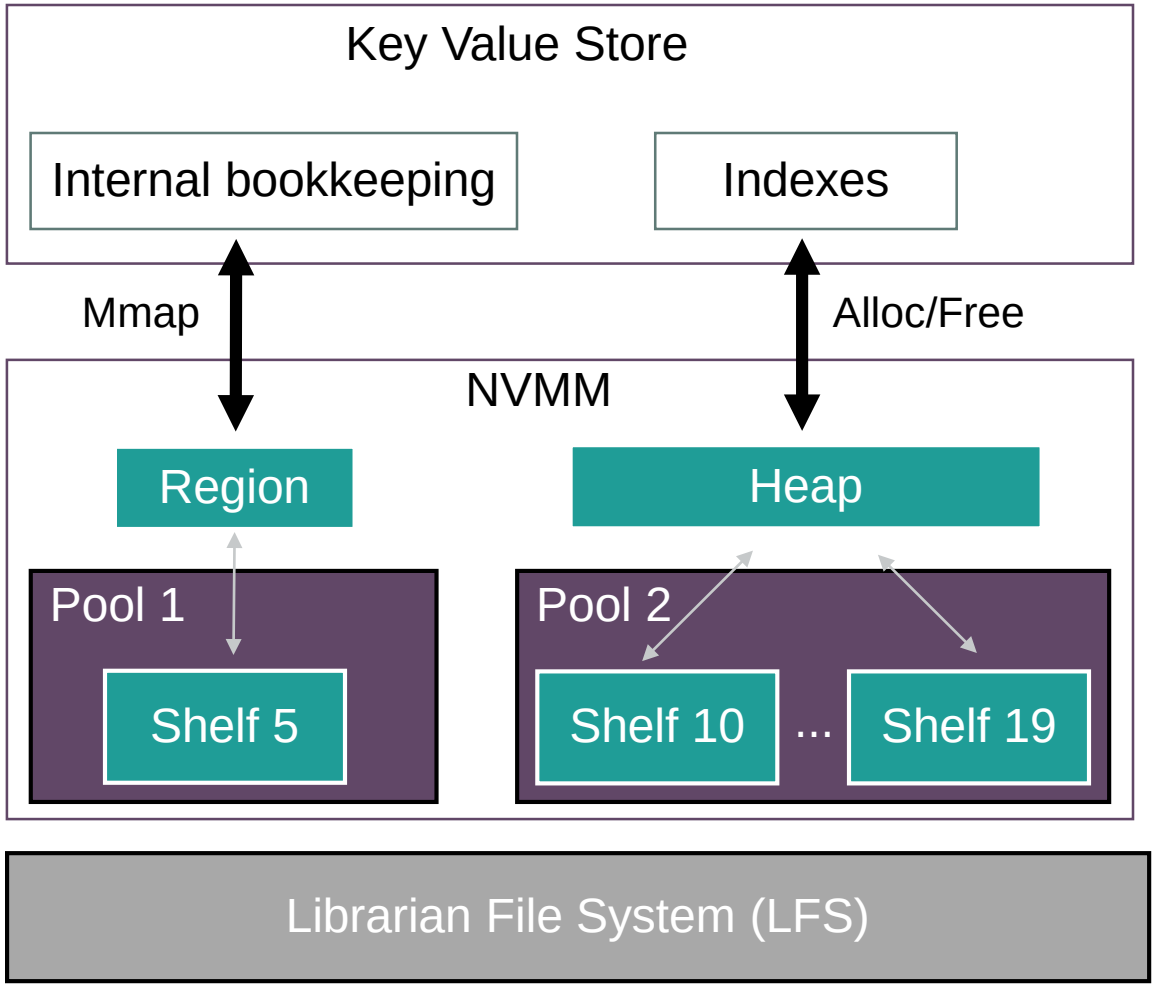
- What's different in fabric-attached memory (FAM)?
  - Memory semantic accesses (including atomics)
  - No hardware-based cache coherence between nodes
  - Separate fault domains between FAM and nodes
- Goal: multi-node, FAM-aware KVS
- How: exploit FAM and FAM atomics for
  - Concurrency control
  - Fault tolerance
  - Load balancing
- Open sourced components
  - Memory manager: MDC toolkit NVMM
  - Indexes: MDC toolkit radix tree





Non-volatile Memory Manager (NVMM)

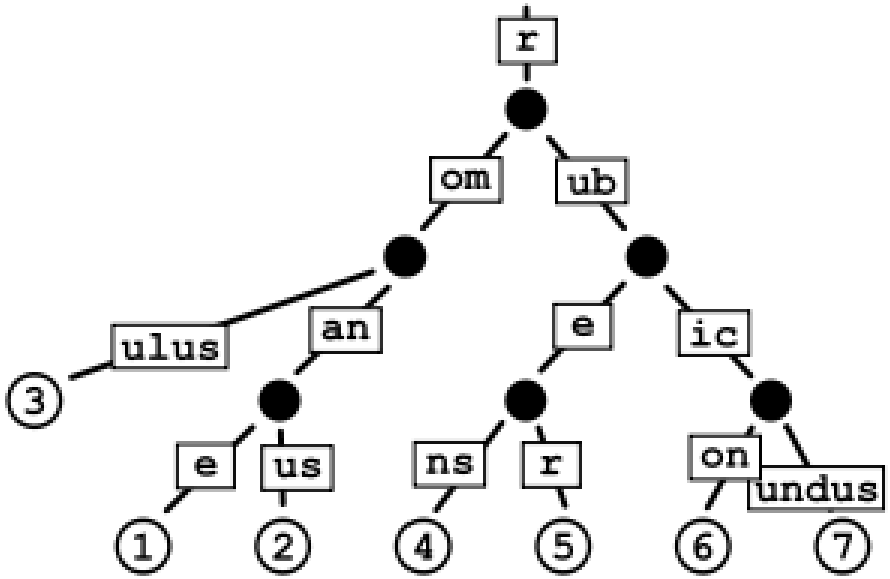
- Goals
  - Provide finer-grained memory allocation than LFS
  - Enable sharing across different processes and nodes
  - Tolerate failures of compute nodes and processes
- Manage shelves provided by LFS
  - Allocate, free, and name shelves (multiples of 8GBs)
  - Expose a global address space: {shelf ID, shelf offset}
- Memory access abstractions
  - *Region* APIs for direct memory-map access
  - *Heap* APIs for alloc/free style access
- Heap: allow any process from any node to allocate and free globally shared NVM transparently



Radix trees

- Also known as compact prefix tries
  - "Compress" common prefixes to improve space efficiency
- Maintain keys in sorted order
  - Support range (multi-key) lookups, unlike hash tables
- More amenable to FAM than B+-trees
  - Require fewer structural changes (e.g., node splits) as tree grows
  - Atomic operations used to insert or delete key and leave tree in consistent state

```
romane 1
romanus 2
romulus 3
rubens 4
ruber 5
rubicon 6
rubicundus 7
```

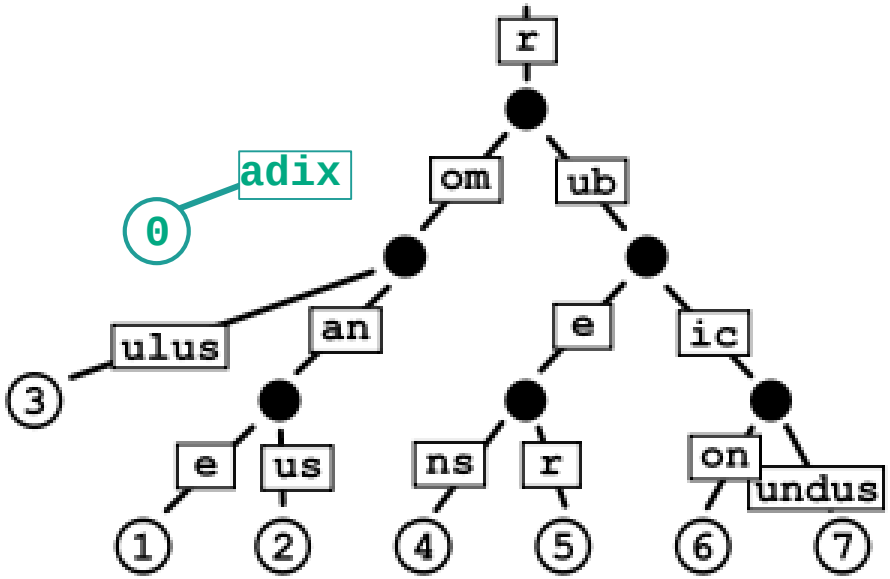


https://en.wikipedia.org/wiki/Radix_tree

Radix trees

– Also known as compact prefix tries
  – “Compress” common prefixes to improve space efficiency
– Maintain keys in sorted order
  – Support range (multi-key) lookups, unlike hash tables
– More amenable to FAM than B+-trees
  – Require fewer structural changes (e.g., node splits) as tree grows
  – Atomic operations used to insert or delete key and leave tree in consistent state

```
radix 0
romane 1
romanus 2
romulus 3
rubens 4
ruber 5
rubicon 6
rubicundus 7
```

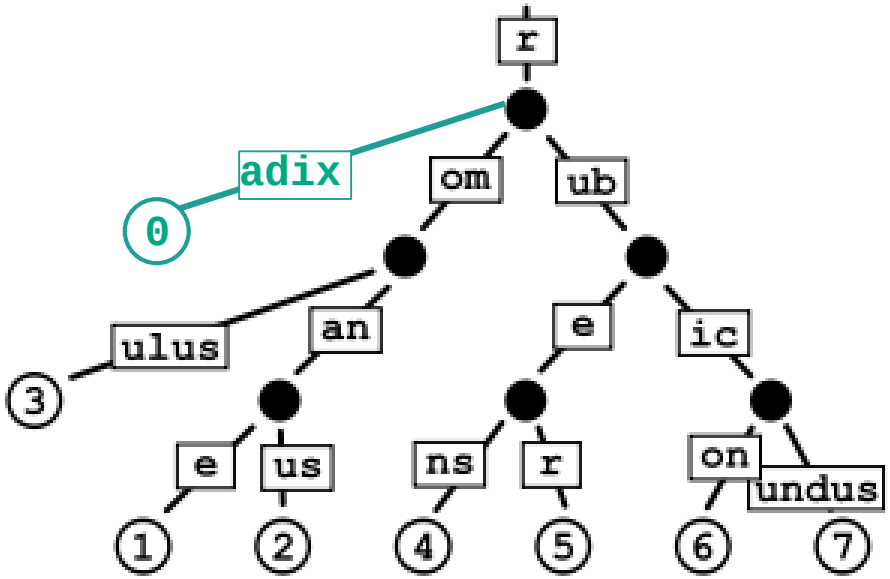


https://en.wikipedia.org/wiki/Radix_tree

Radix trees

– Also known as compact prefix tries
  – “Compress” common prefixes to improve space efficiency
– Maintain keys in sorted order
  – Support range (multi-key) lookups, unlike hash tables
– More amenable to FAM than B+-trees
  – Require fewer structural changes (e.g., node splits) as tree grows
  – Atomic operations used to insert or delete key and leave tree in consistent state

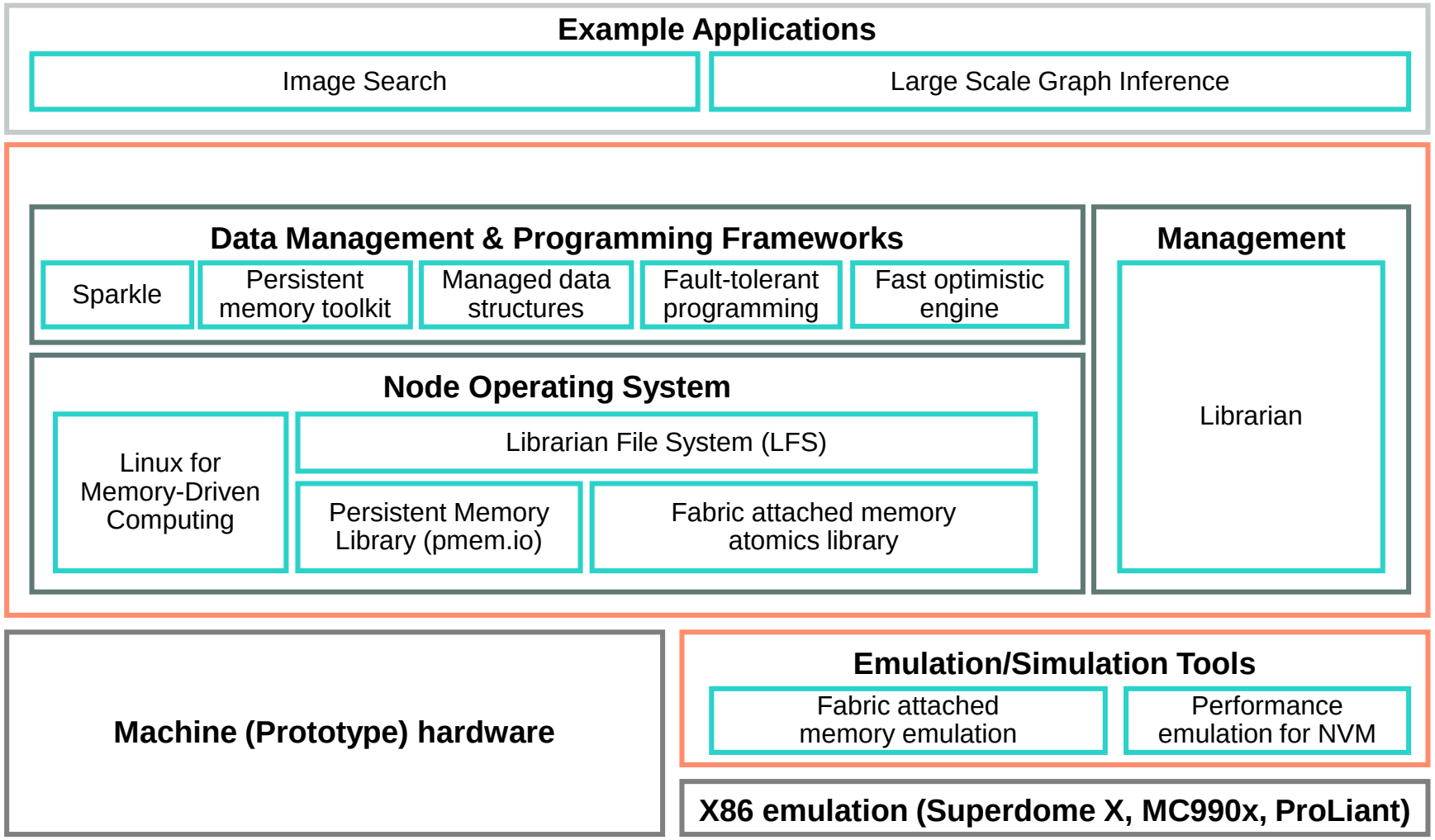
```
radix 0
romane 1
romanus 2
romulus 3
rubens 4
ruber 5
rubicon 6
rubicundus 7
```



https://en.wikipedia.org/wiki/Radix_tree

Memory-Driven Computing Developer Toolkit

Software already available to you



- Example applications
- Programming and analytics tools
- Operating system support
- Emulation/simulation tools

Get access to the toolkit:

https://www.labs.hpe.com/the-machine/developer-toolkit

Hewlett Packard Enterprise

Open source components

Persistent memory as storage?

–If persistent memory is the new storage…

it must safely remember persistent data.

–Persistent data should be stored:

- Reliably, in the face of failures
- Securely, in the face of exploits
- In a cost-effective manner
- Using a data access API that's most natural for the device characteristics

Storing data reliably, securely and cost-effectively

The problem

- Potential concerns about using persistent memory to safely store persistent data:
  - NVM failures may result in loss of persistent data
  - Persistent data may be stolen

- Time to revisit traditional storage services
  - Ex: replication, erasure codes, encryption, compression, deduplication, wear leveling, snapshots

- New challenges:
  - Need to operate at *memory* speeds, not storage speeds
  - Traditional solutions (e.g., encryption, compression) complicate direct access
  - Space-efficient redundancy for NVM

Storing data reliably, securely and cost-effectively
Potential solutions

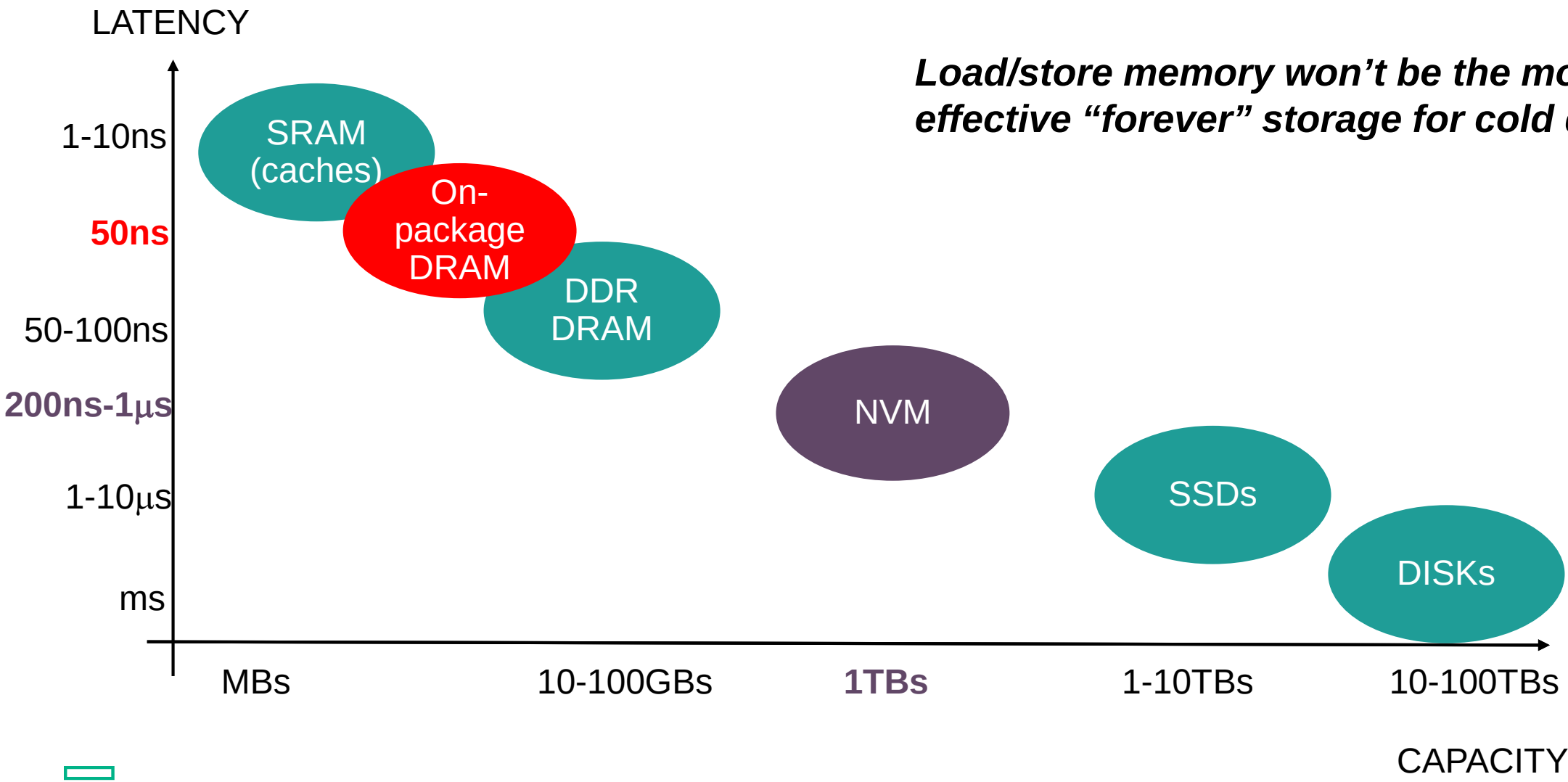
- Software implementations can trade performance for reliability, security and cost-effectiveness
  - But will diminish benefits from faster technologies

- Memory-side hardware acceleration
  - Memory speeds may demand acceleration (e.g., DMA-style data movement, memset, encryption, compression)
  - What functions are ripe for memory-side acceleration?

- Wear leveling for fabric-attached non-volatile memory
  - Repeated NVM writes may exacerbate device wear issues
  - What's the right balance between hardware-assisted wear leveling and software techniques?

- Fabric-attached non-volatile memory diagnostics
  - What is the equivalent of Self-Monitoring, Analysis and Reporting Technology (SMART) for NVM?

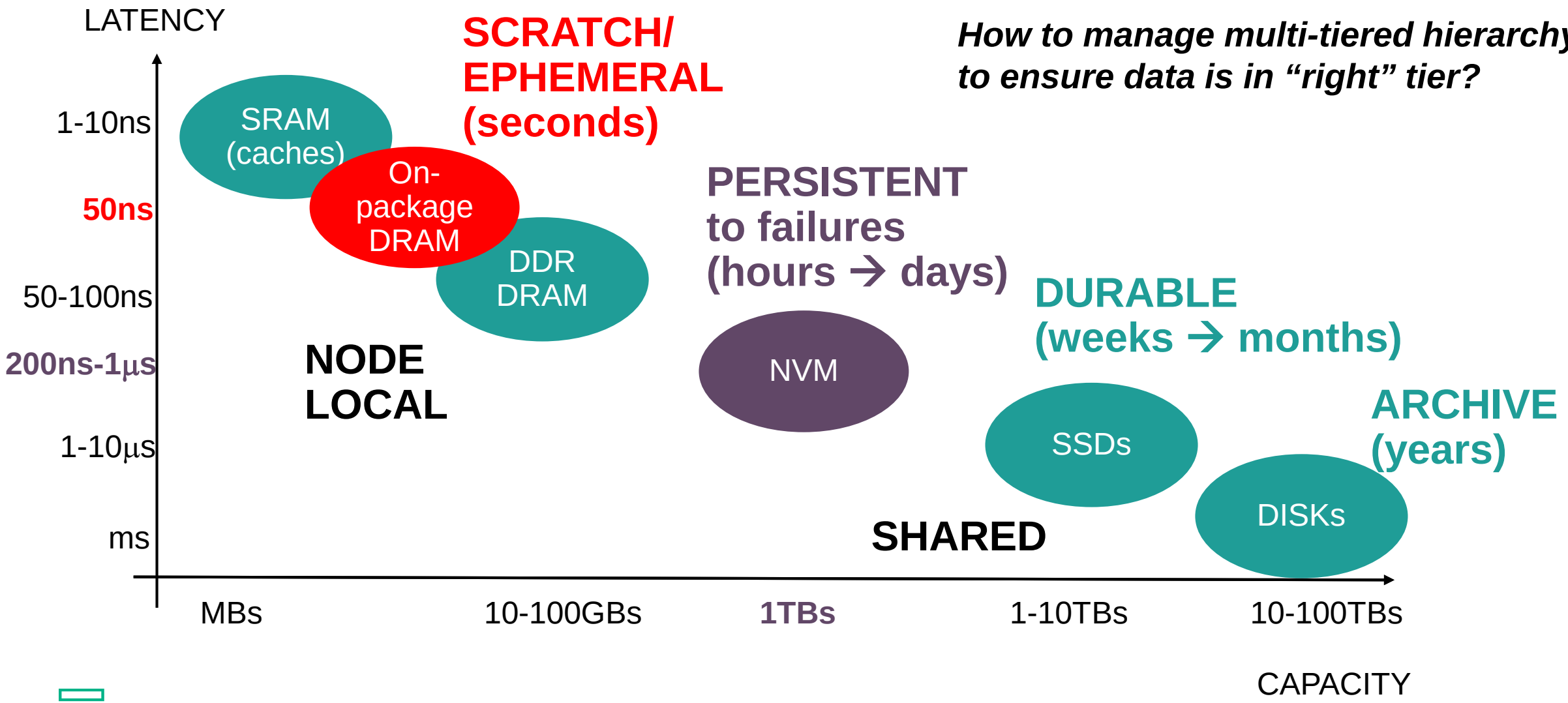
Memory + storage hierarchy technologies



Load/store memory won't be the most cost-effective "forever" storage for cold data

Hewlett Packard Enterprise

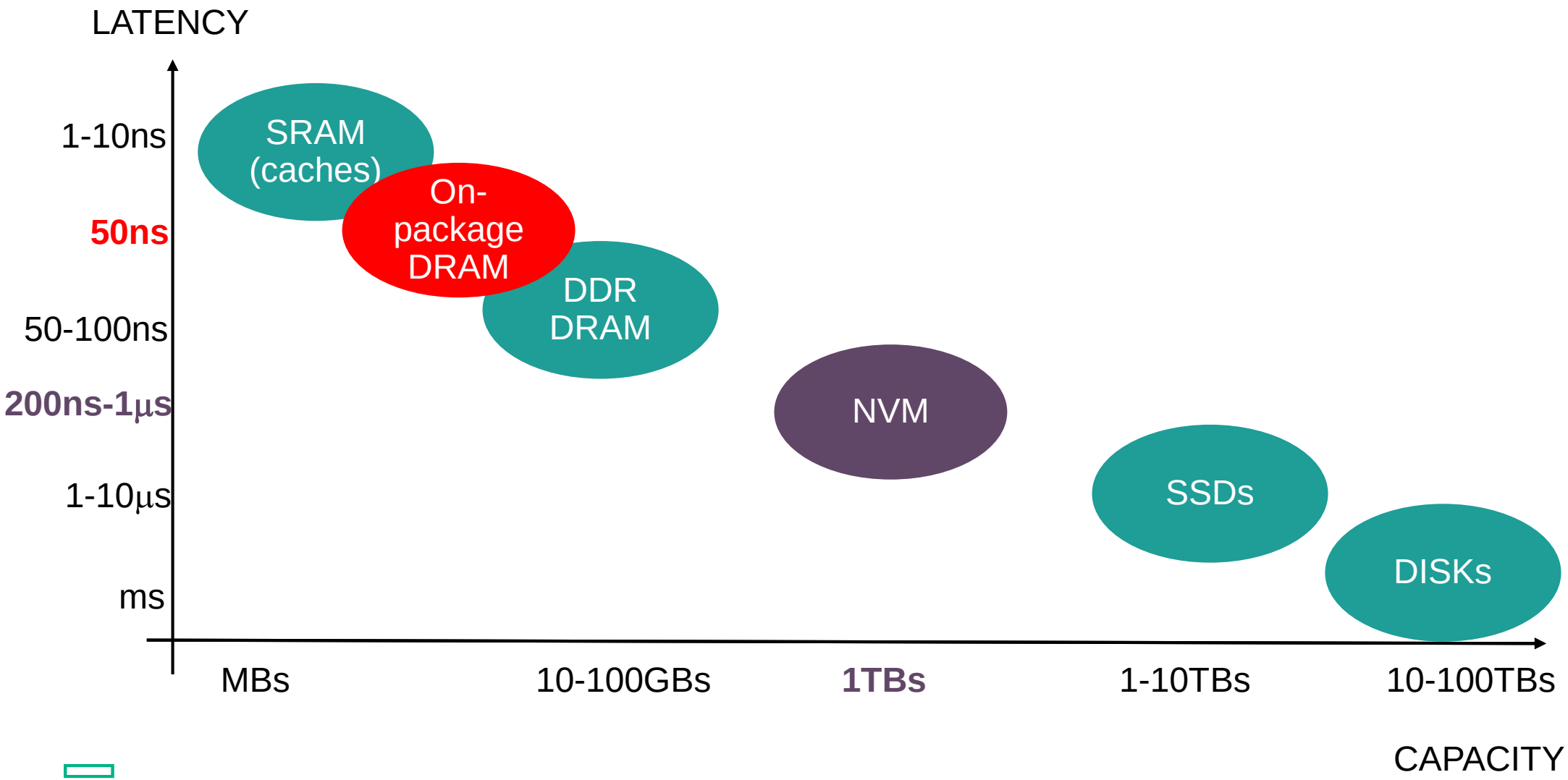
Memory + storage hierarchy – a usage view (ownership)



How to manage multi-tiered hierarchy, to ensure data is in "right" tier?

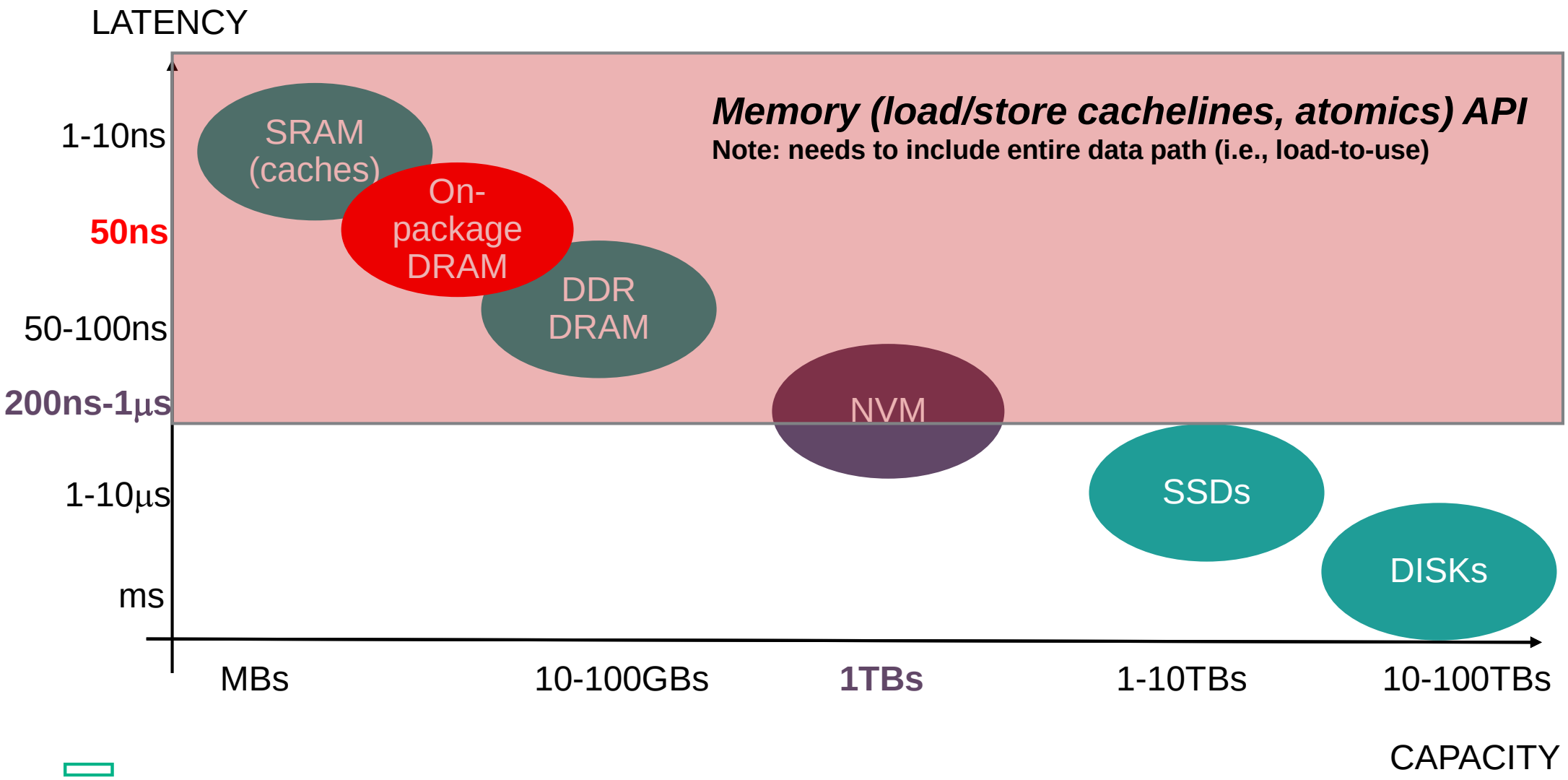
Hewlett Packard Enterprise

The appropriate API for the technology



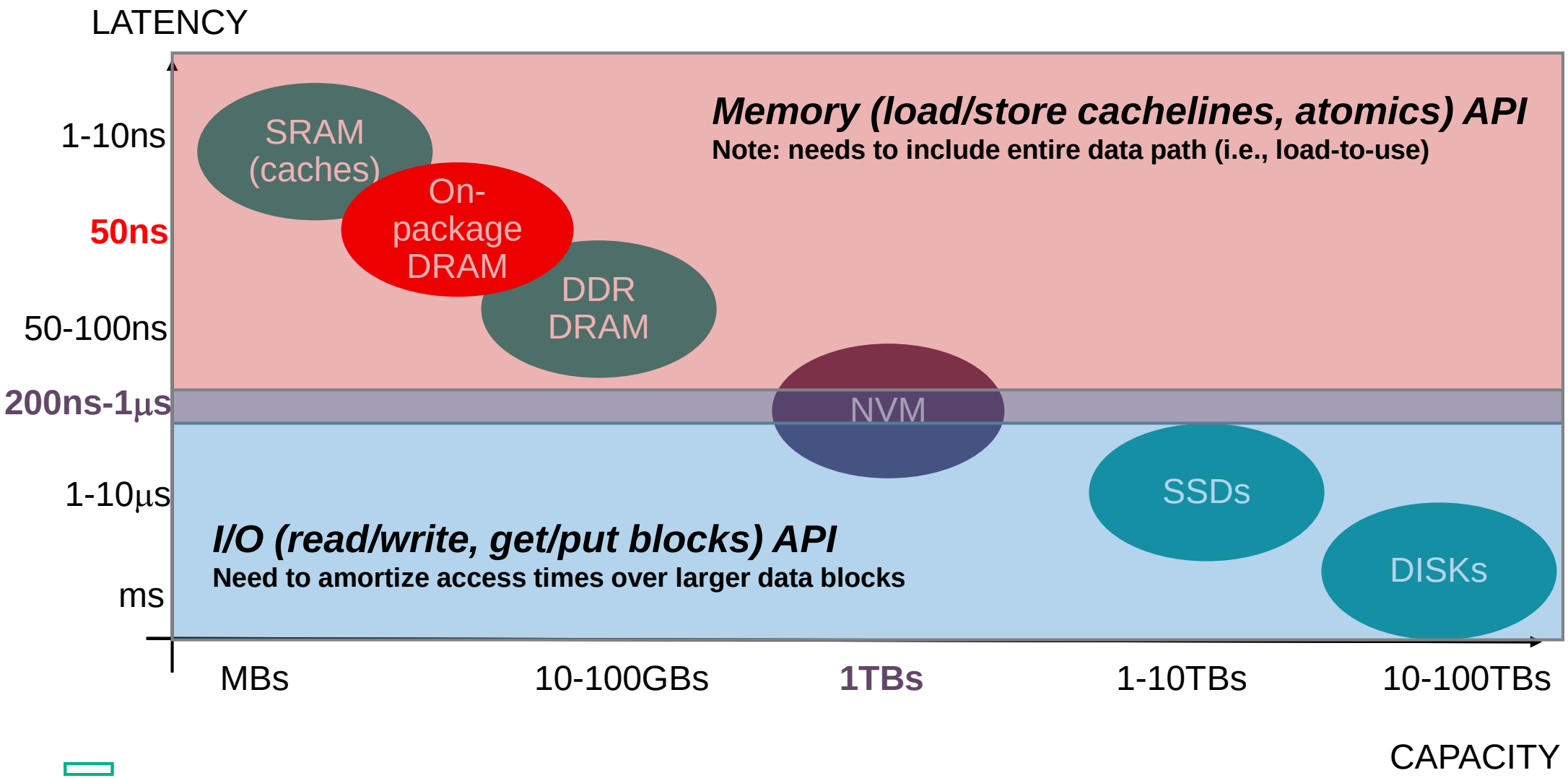
Hewlett Packard Enterprise

The appropriate API for the technology



Hewlett Packard Enterprise

The appropriate API for the technology



Hewlett Packard Enterprise

How to get started with Memory-Driven Computing

- Latest updates on The Machine project and Memory-Driven Computing: www.hpe.com/themachine

- Join The Machine User Group at https://www.labs.hpe.com/the-machine/user-group
  - Join community discussions on Slack https://www.labs.hpe.com/slack #themachineusergroup
  - Subscribe to “The Machine User Group” tab in the “Behind the scenes @ Labs” blog https://community.hpe.com/t5/Behind-the-scenes-Labs/bg-p/BehindthescenesatLabs/label-name/The%20Machine%20User%20Group#.WXZGN4jyscE.
    Register and click “subscribe to this label”.
  - Questions? Contact themachineusergroup@hpe.com

- Follow us on our Hewlett Packard Labs social handles:
  - Twitter: @HPE_Labs
  - LinkedIn: “Hewlett Packard Labs”
  - “Hewlett Packard Enterprise” YouTube page – The Machine and Hewlett Packard Labs channels
  - Instagram: HPE
  - Facebook: Hewlett Packard Enterprise

Wrapping up

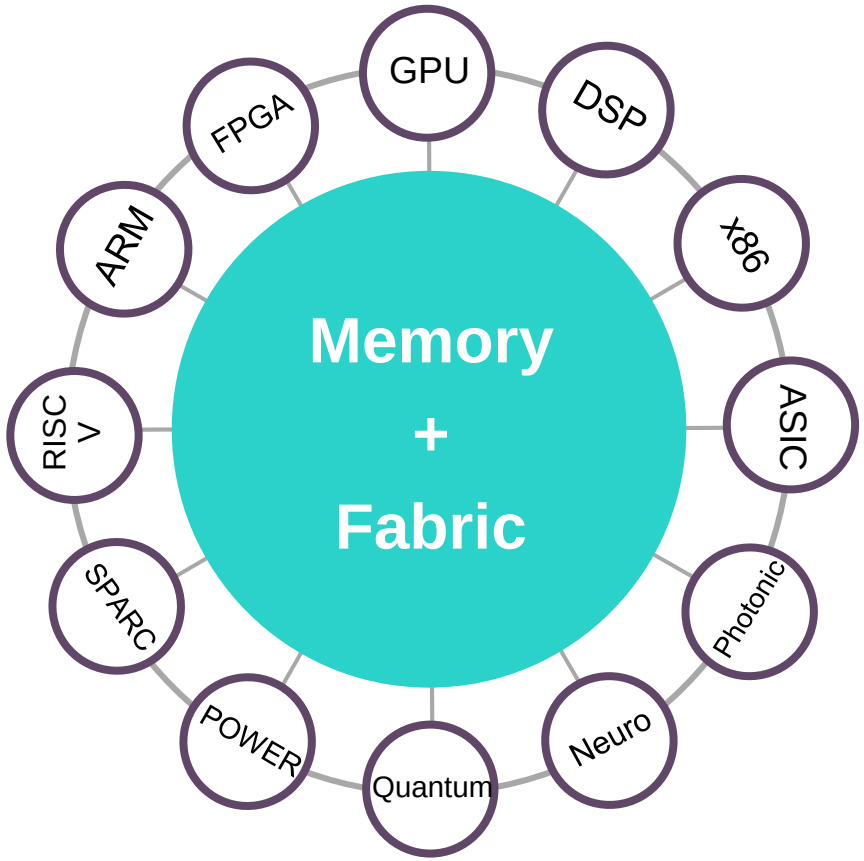
– Persistent memory blurs the line between memory and storage

– Gen-Z fabric enables us to shift system architecture towards Memory-Driven Computing

– Combination of technologies enables us to rethink the programming model
  – Simplify software stack
  – Operate directly on memory-format persistent data

– Many opportunities for software innovation

– How would *you* exploit Memory-Driven Computing?



https://www.labs.hpe.com/the-machine

https://www.labs.hpe.com/the-machine/developer-toolkit

https://www.labs.hpe.com/next-next/mdc