Hewlett Packard
Enterprise

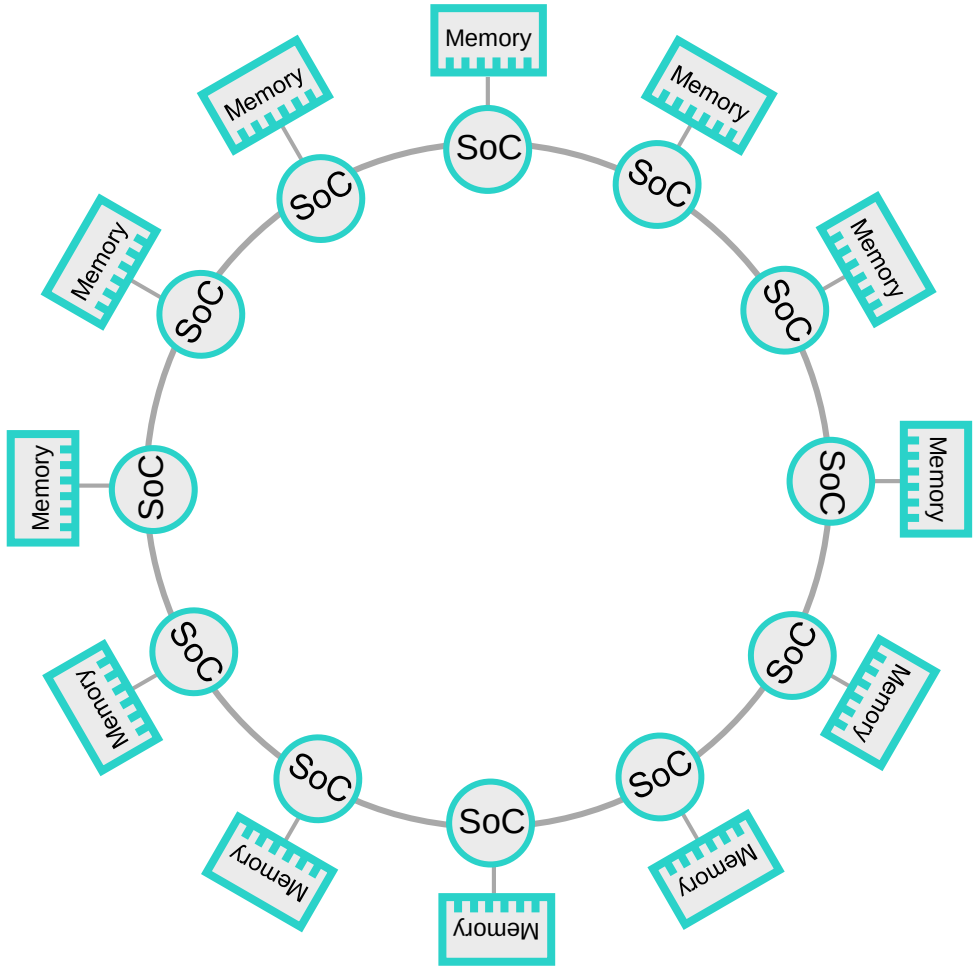
Memory-Driven Computing

Kimberly Keeton

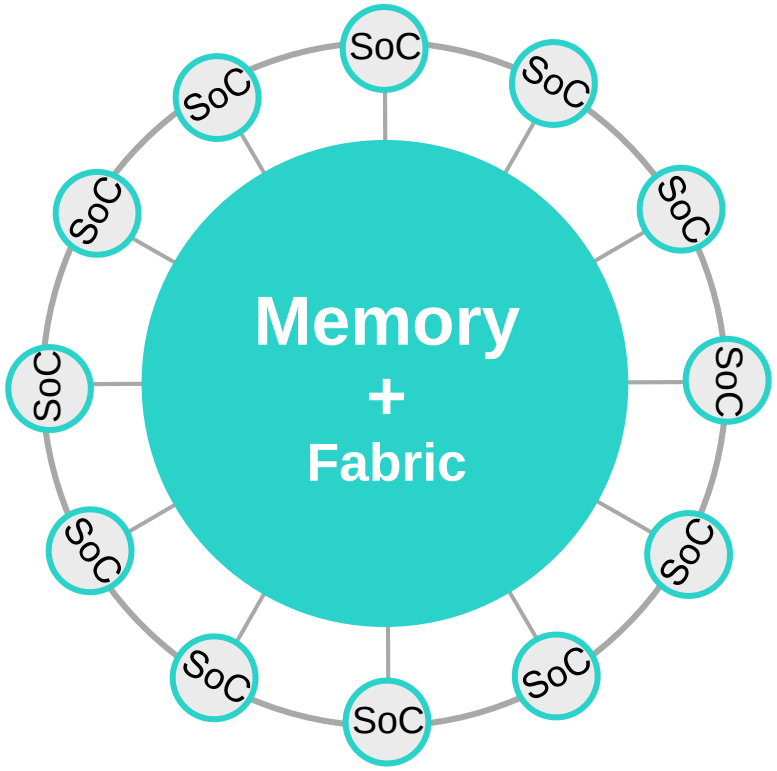
Distinguished Technologist
kimberly.keeton@hpe.com

October 20, 2016

Hewlett Packard
Labs



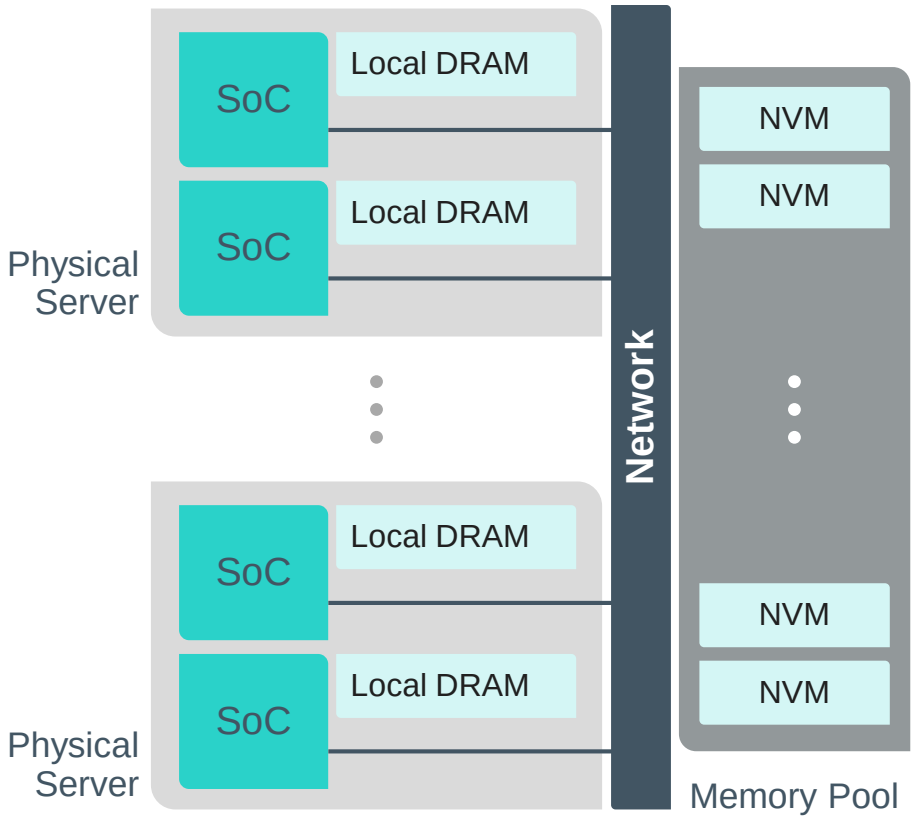
From Processor-Centric Computing…



…to Memory-Driven Computing

Hewlett Packard Enterprise

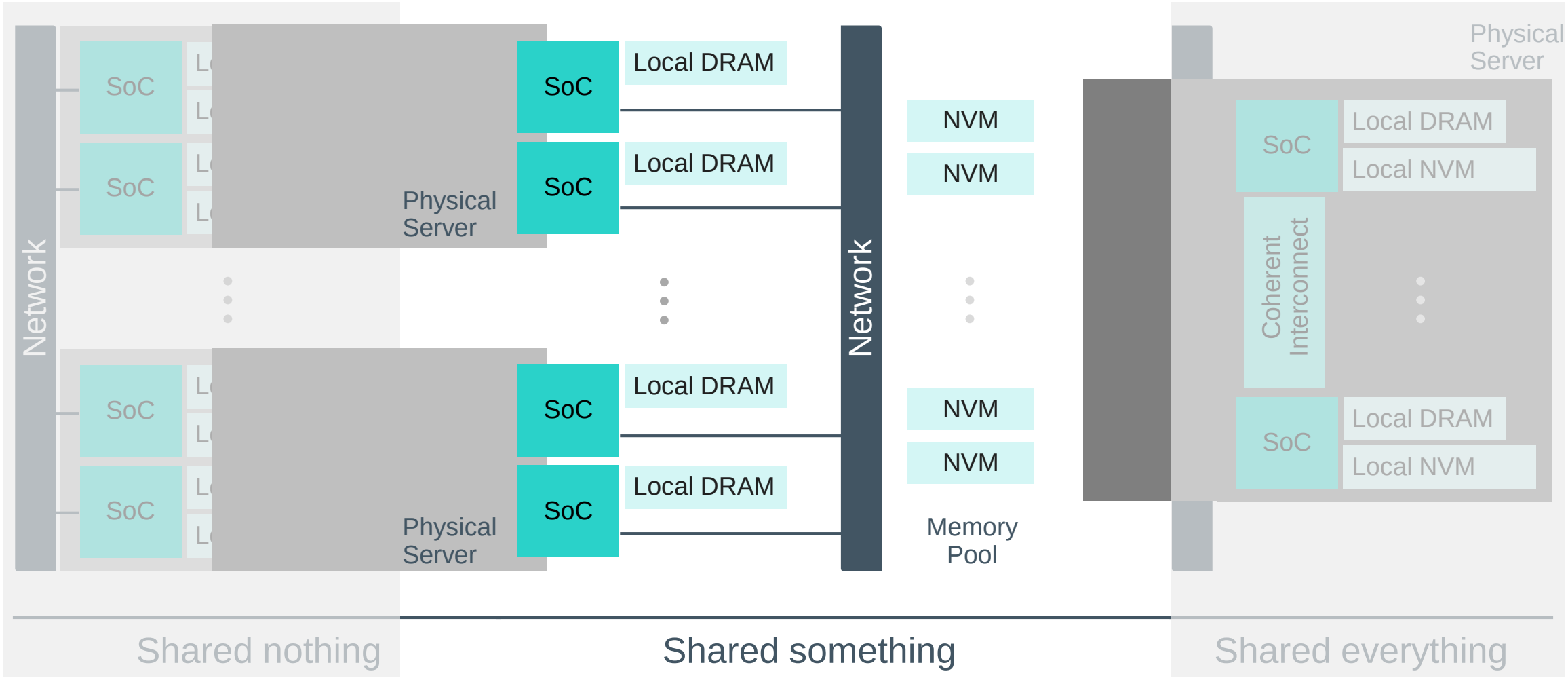
Technology trends enabling Memory-Driven Computing



- **Converging memory and storage**
  - Byte-addressable persistent memory (NVM) replaces hard drives and SSDs
- **Resource disaggregation leads to shared memory pool**
  - Fabric-attached memory pool is accessible by all compute resources
  - Optical networking provides near-uniform latency
  - Local volatile memory provides lower latency, high performance tier
- **Distributed heterogeneous compute resources**
  - Moving compute closer to data
- **Software**
  - Memory-speed persistence
  - Low-latency, high BW memory/storage access
  - Globally addressable, low latency access to all PM across the memory fabric

Hewlett Packard Enterprise

Memory-Driven Computing in context



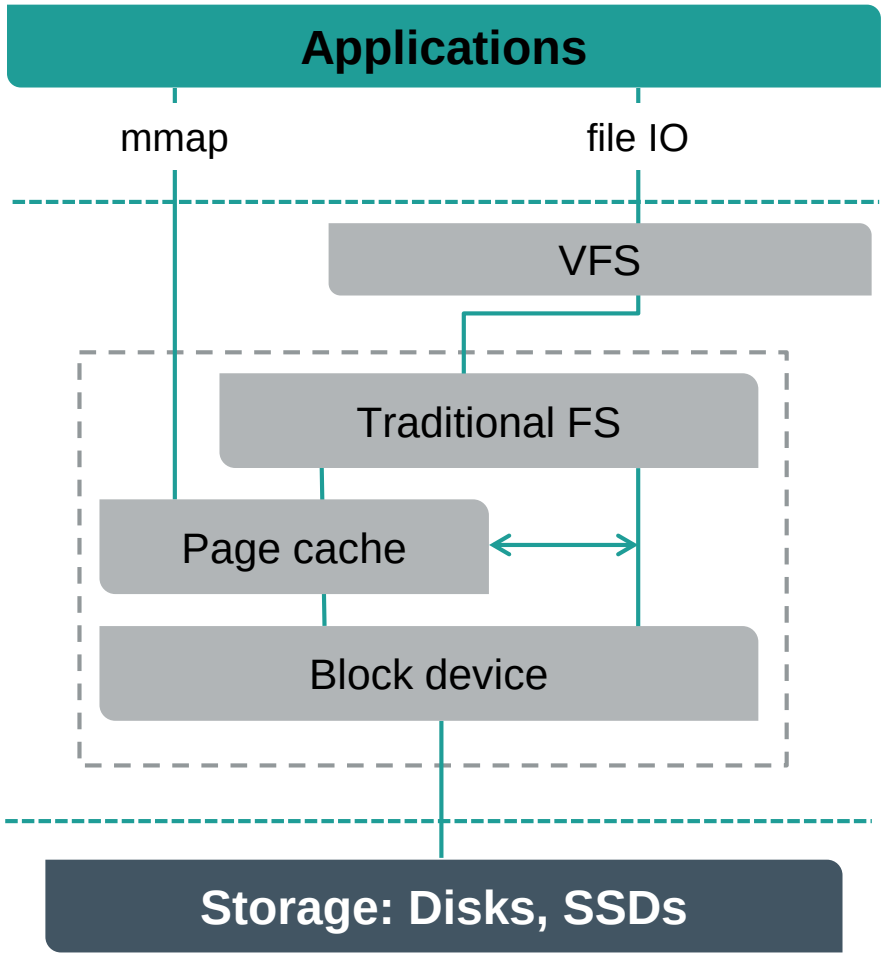
Hewlett Packard Enterprise

Outline

- Overview
- Memory driven systems software: OS and data management
- Memory driven data analytics: Spark for The Machine
- Memory driven programming models
- Prototypes and emulators
- Commercial memory driven computing
- Summary

Memory driven systems software

Traditional file systems



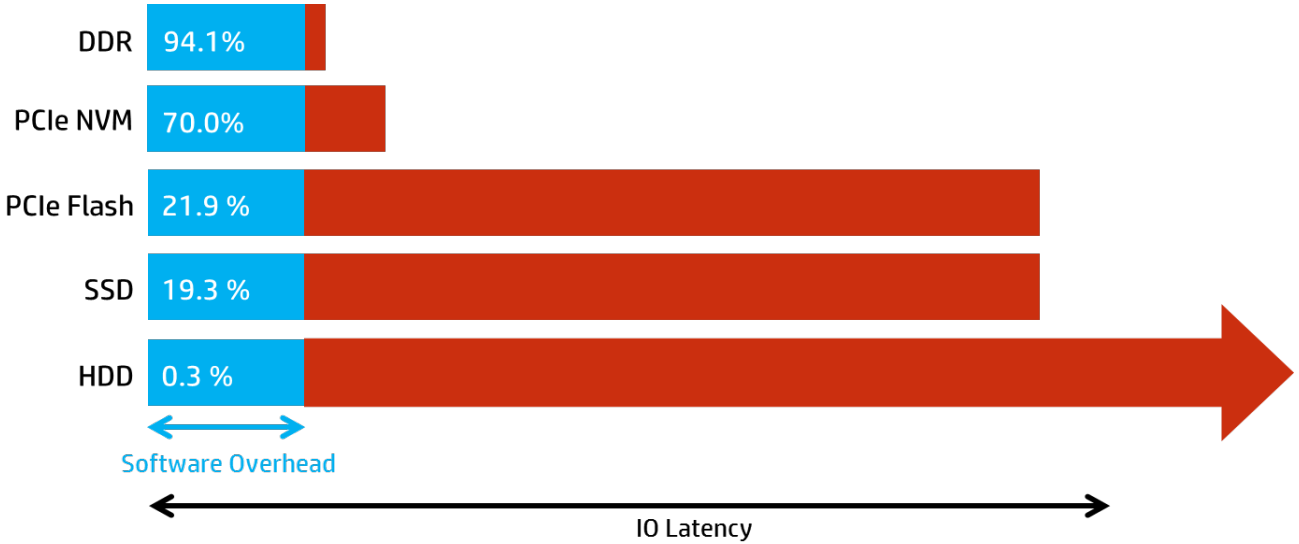
Separate storage address space

- Data is copied between storage and DRAM
- Block-level abstraction leads to inefficiencies

Use of page cache leads to extra copies

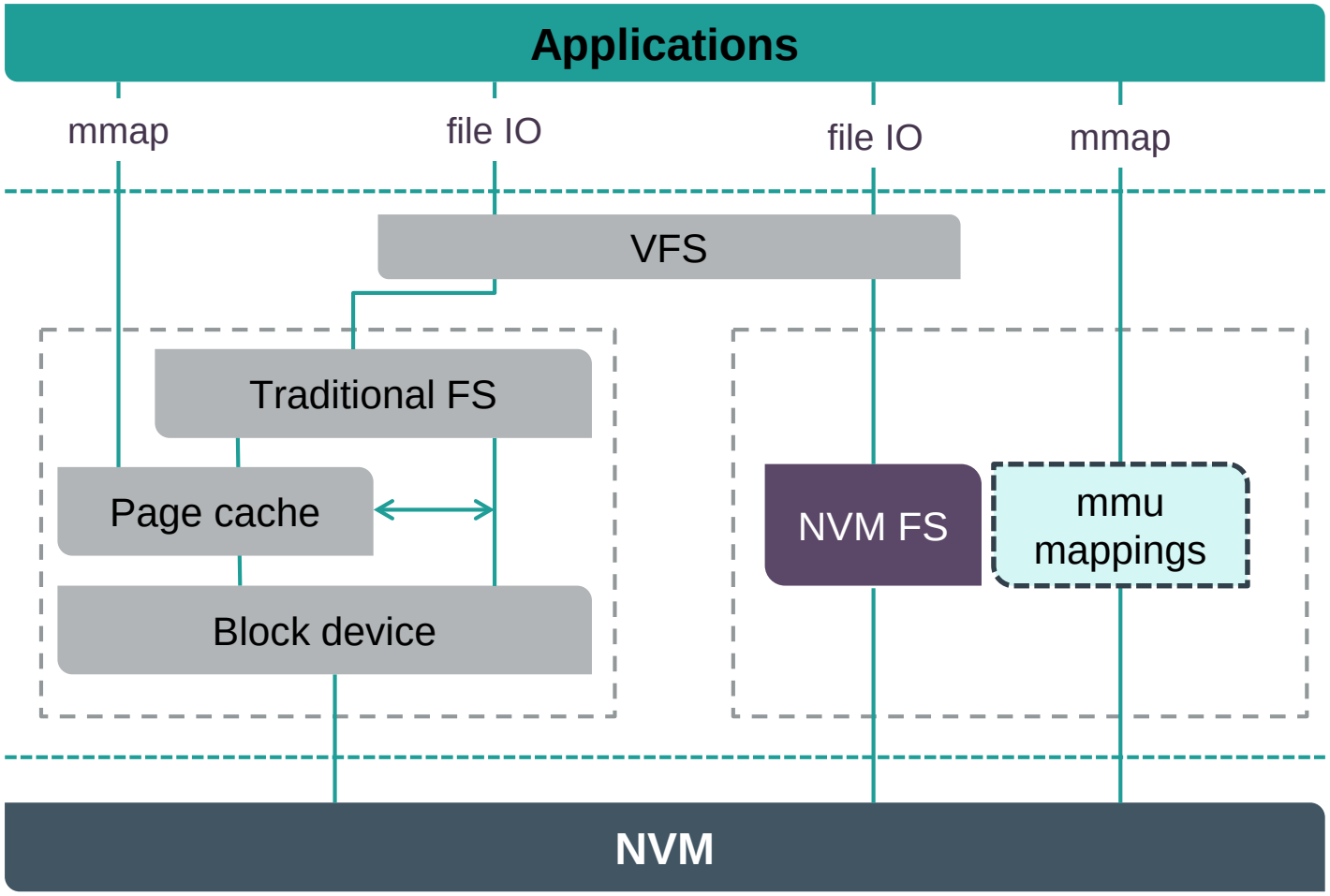
- True even for memory-mapped I/O

Software layers add overhead



Hewlett Packard Enterprise

Non-volatile memory aware file systems



Source: S. R Dulloor, et al. "System Software for Persistent Memory," *Proc. EuroSys*, 2014.

Examples

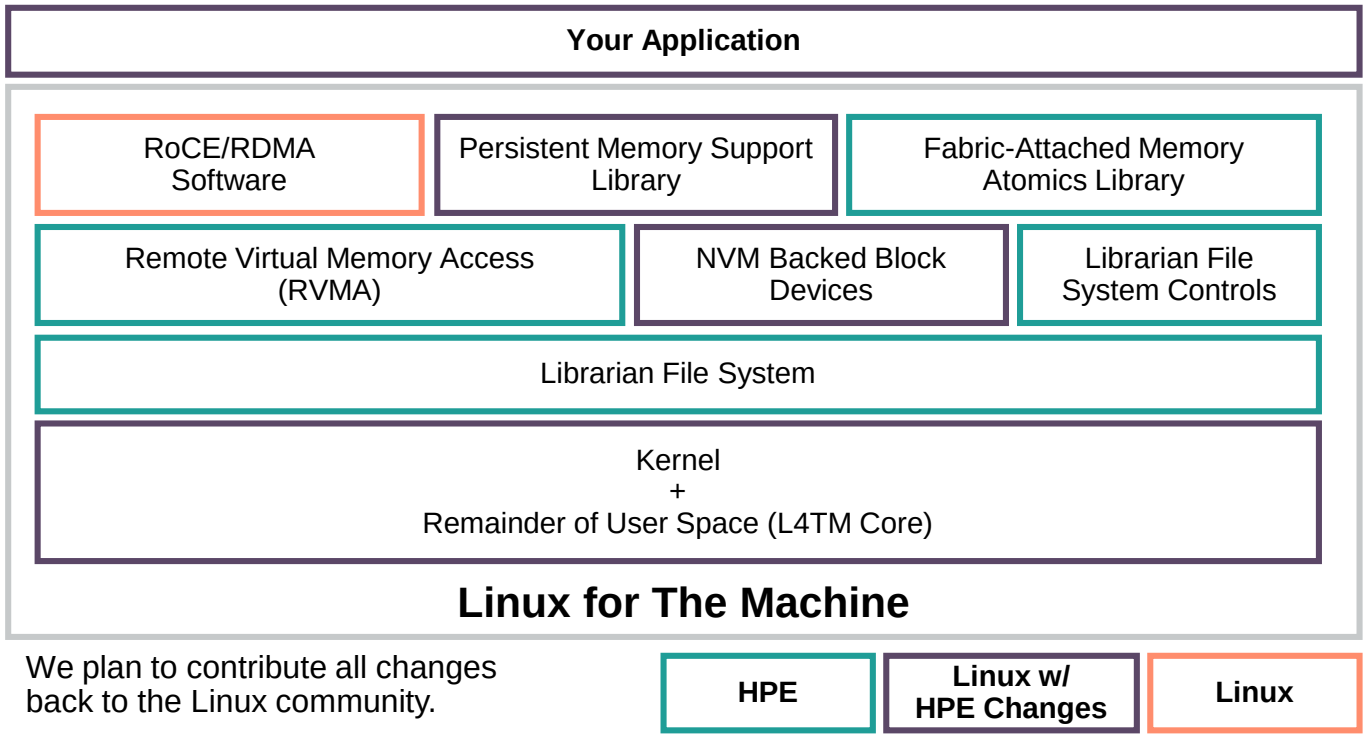
- Direct access (DAX)
- pmem.io/NVML

Low overhead access to persistent memory

- No page cache
- Direct access with mmap

Linux for The Machine

- HPE's modifications to Linux to support
  - Fabric-attached persistent memory
  - Block device abstractions backed by persistent memory
  - Kernel modifications for "flush on failure"
- Additional support for
  - Fabric-attached memory atomics primitives to handle sharing across SoCs
  - "Librarian File System" for naming and giving permissions to persistent memory
  - Remote virtual memory access
  - Configuration and management utilities

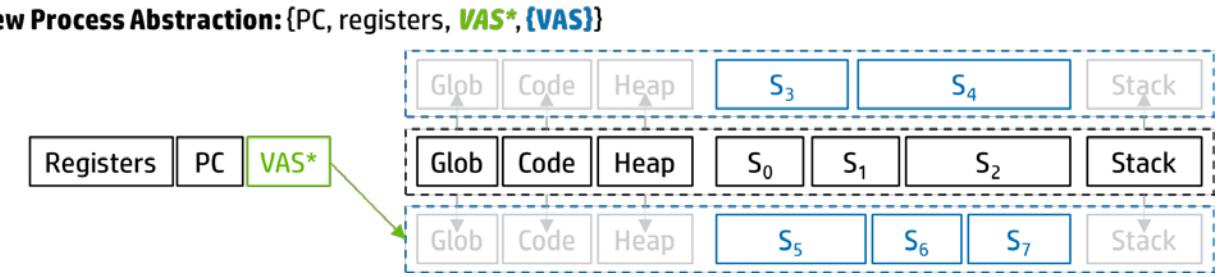


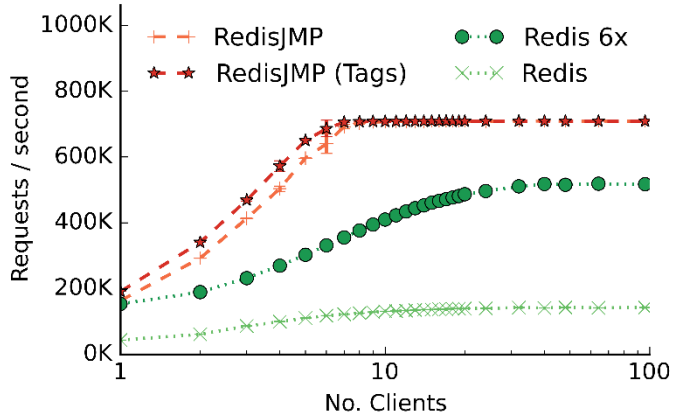
We plan to contribute all changes back to the Linux community.

HPE | **Linux w/ HPE Changes** | **Linux**

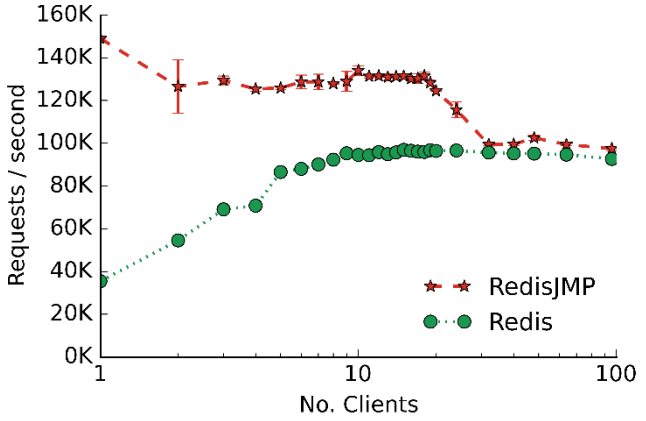
SpaceJMP: Programming with Multiple Virtual Address Spaces

- Process has multiple virtual address spaces





- Efficient safe programming and sharing for huge memories
- Data sharing and communication between processes
- Versioning and checkpointing
- Co-design between OS, programming languages, compilers, and runtimes
- Prototype implementations in BSD, Linux, and Barrelfish



I. El Hajj, et al. “SpaceJMP: Programming with Multiple Virtual Address Spaces,” *Proc. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016.

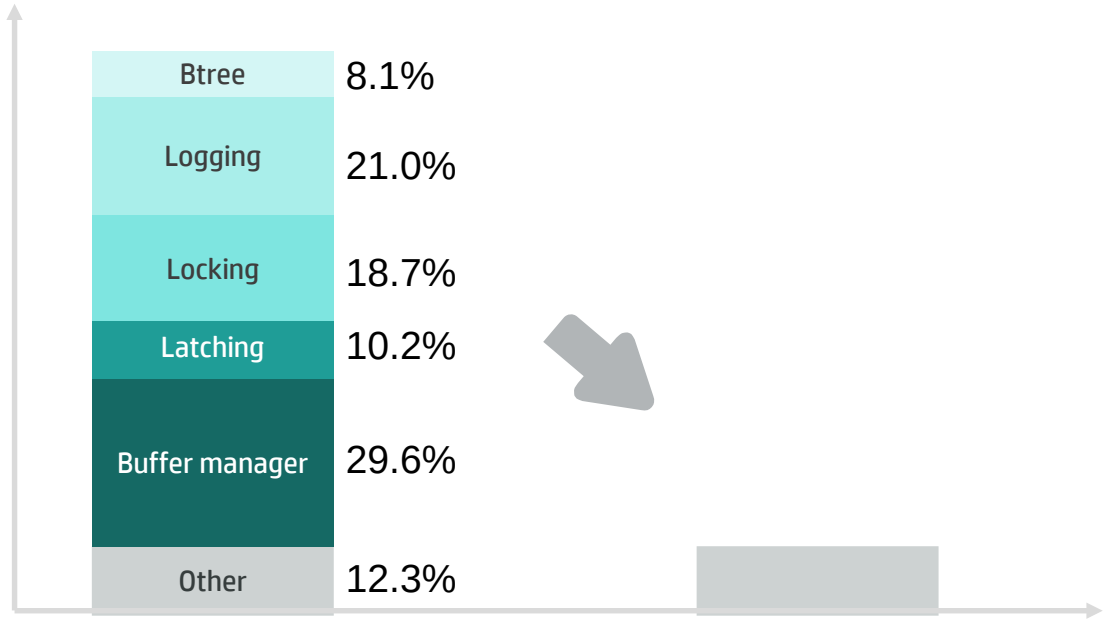
Hewlett Packard Enterprise

Memory-driven data management

Traditional databases

–Example: A database (write) transaction

- Traditional databases struggle with big & fast data
- 90% of a database transaction is overhead
- Memory-semantics non-volatile memory: up to 10x improvement



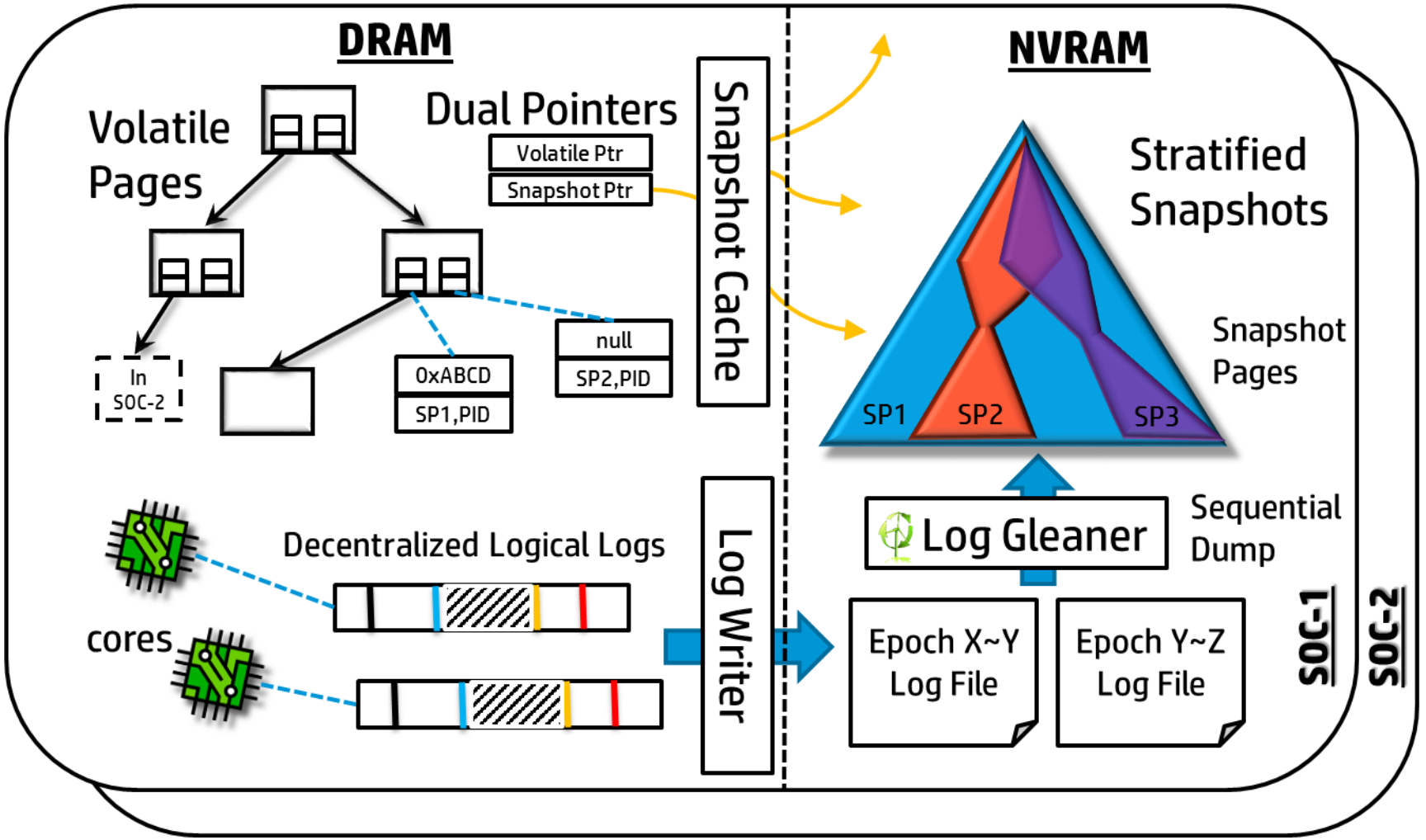
S. Harizopoulos, D. Abadi, S. Madden, and M. Stonebraker, “OLTP Through the Looking Glass, and What We Found There,” *Proc. SIGMOD*, 2008.

Hewlett Packard Enterprise

FOEDUS: Fast optimistic engine for data unification services

- Open-source, from-scratch database engine designed to
  - Manipulate data both in DRAM and NVM
  - Take advantage of large multi-core machines
- Fully ACID, serializable database kernel in C++
  - Can be embedded in applications as a library
  - Simplified in-memory applications
- Designed to eliminate scalability bottlenecks
  - Lightweight optimistic concurrency control
  - Decentralized logs are SoC-friendly
  - Design maximizes NVM bandwidth and endurance

FOEDUS: Fast optimistic engine for data unification services

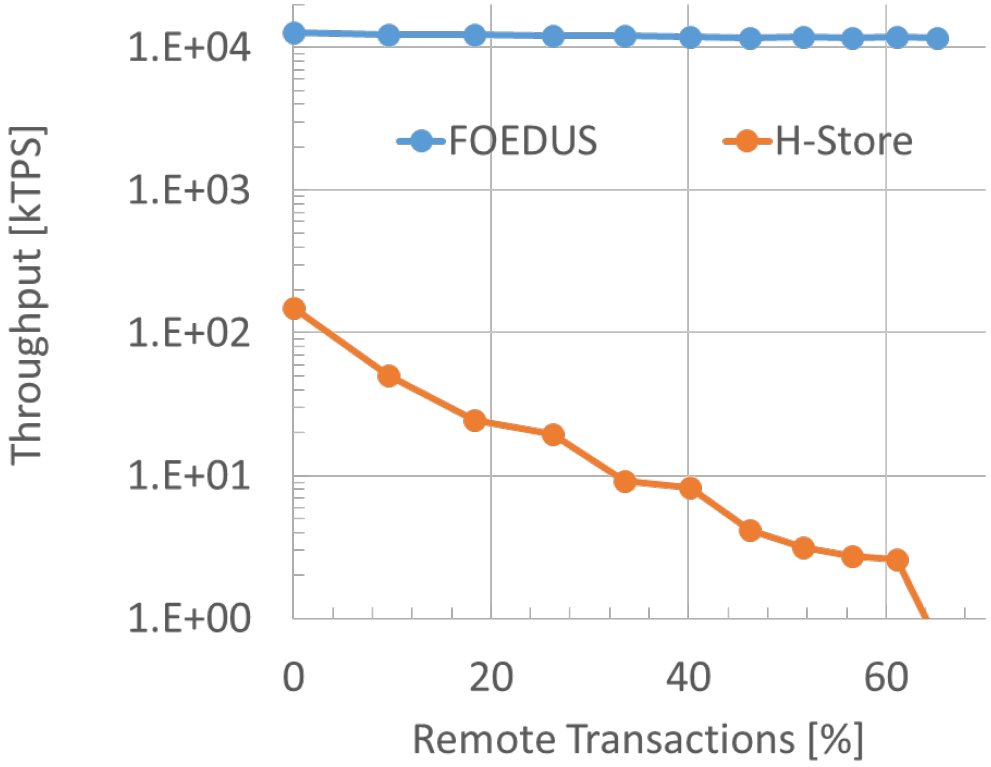


Hewlett Packard Enterprise

FOEDUS: Open source embedded database

– Scalable up to tens of SoCs
  – Tested scale: Superdome X: 12 TB DRAM, 240 cores
– Efficiently handles datasets larger than DRAM
– Orders of magnitude faster when compared to state-of-the-art in-memory engines

– H. Kimura, “FOEDUS: OLTP engine for a thousand cores and NVRAM,” *Proc. SIGMOD*, 2015.
– Open source code, documentation and papers at http://github.com/hkimura/foedus



Hewlett Packard Enterprise

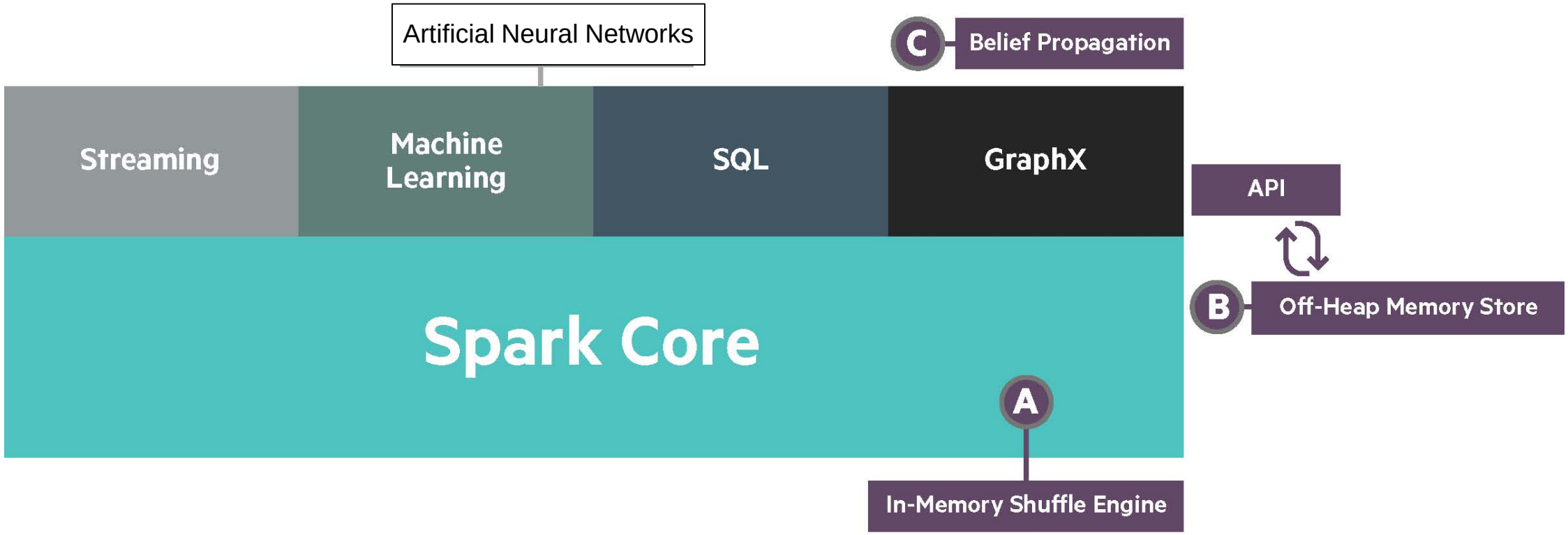
Memory driven data analytics: Spark for The Machine

Hewlett Packard Enterprise

Spark contributions

Maximize advantages of large in-memory processing

HPE Components



Hewlett Packard Enterprise

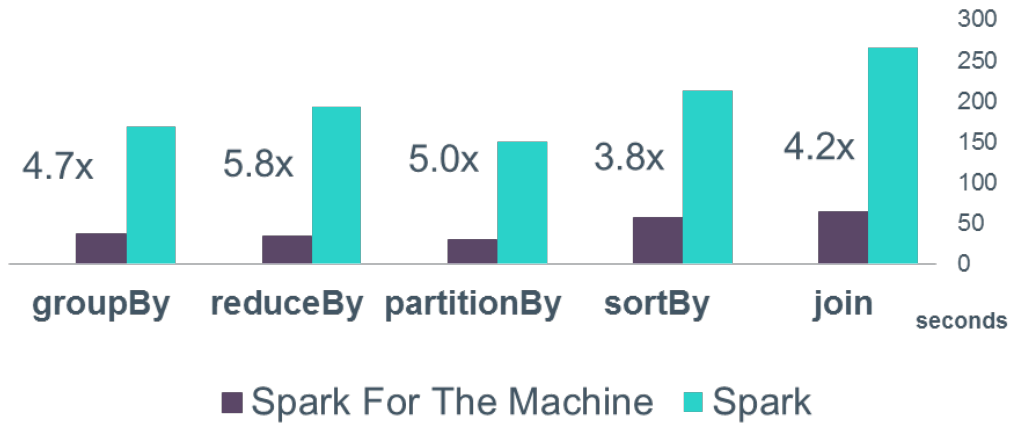
In-memory shuffle engine

Provides efficient access to The Machine's shared-memory architecture and NVM pool

Our approach

– Non-volatile memory based
– In-memory sort/merge
– Optimized CPU cache access

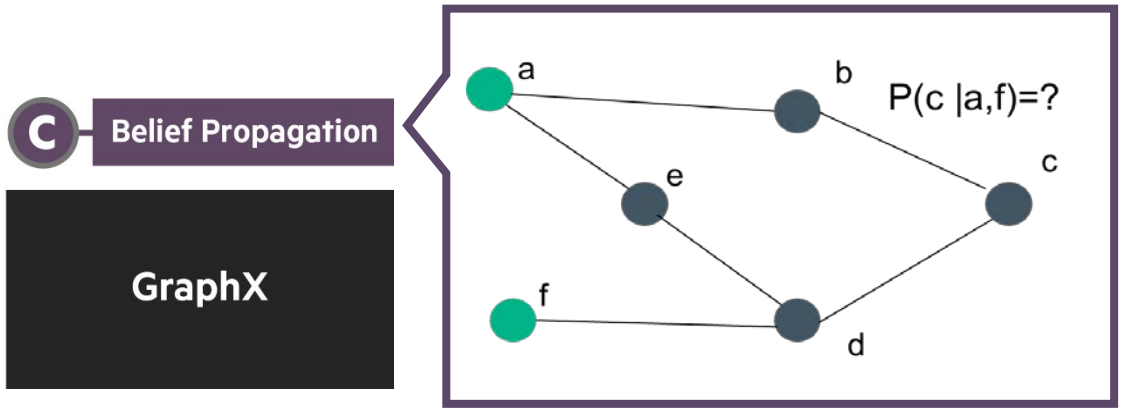
Performance Evaluation for RDD Operators



Hewlett Packard Enterprise

Predictive Analytics

Evaluating Spark for The Machine for fast accurate prediction



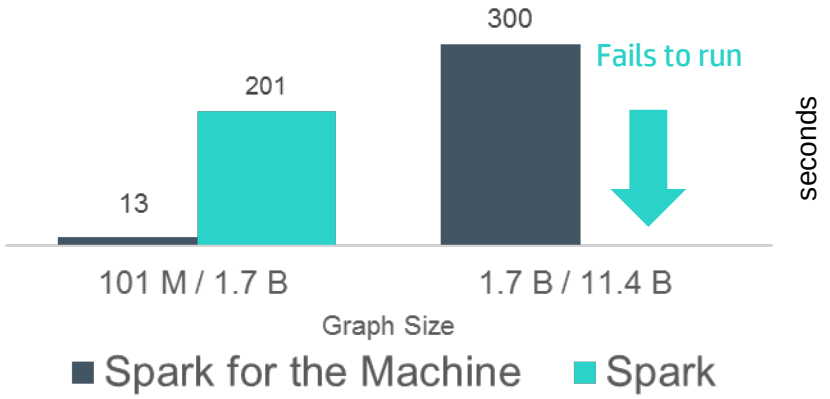
Our approach

– Exploit large NVM pool for data caching

– Leverage computationally intensive belief propagation

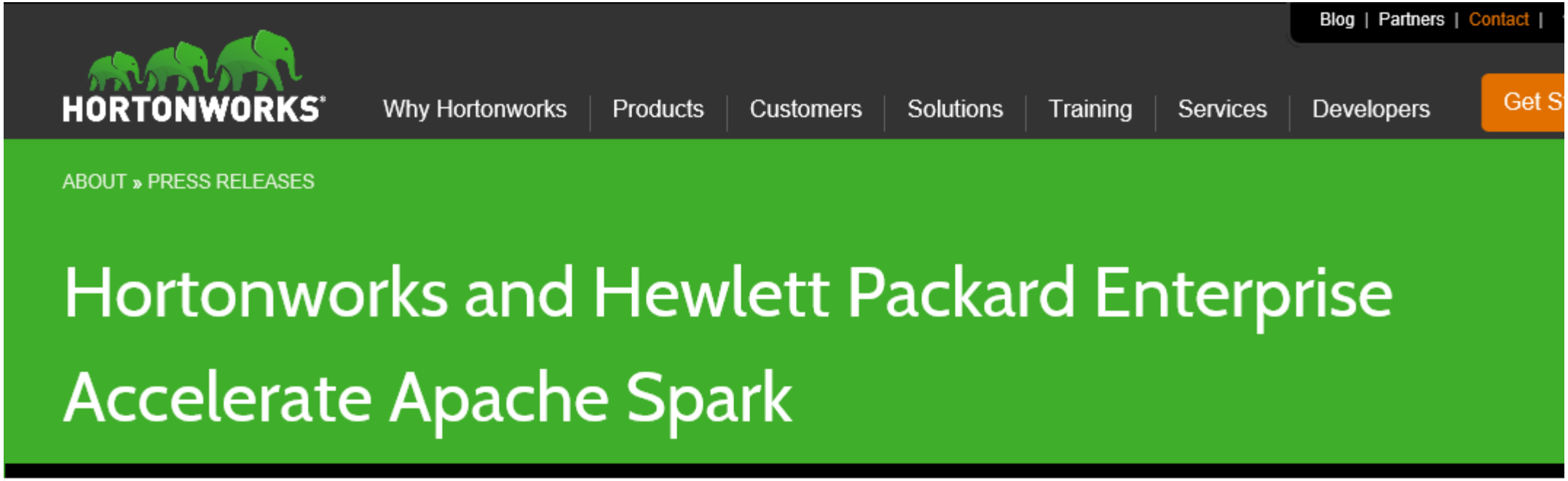
– Decrease communication cost

15x on HPE Superdome X



Hewlett Packard Enterprise

http://hortonworks.com/press-releases/hortonworks-hpe-accelerate-spark/



Blog | Partners | Contact |

HORTONWORKS

Why Hortonworks | Products | Customers | Solutions | Training | Services | Developers | Get S

ABOUT » PRESS RELEASES

Hortonworks and Hewlett Packard Enterprise Accelerate Apache Spark

SAN FRANCISCO, Calif., — March 1, 2016 — Hortonworks, Inc.® (NASDAQ: HDP) and Hewlett Packard Labs, the central research organization of Hewlett Packard Enterprise (NYSE: HPE), today announced a new collaboration to enhance Apache Spark, one of the most active Apache big data projects. The collaboration will center around an entirely new class of analytic workloads that benefit from large pools of shared memory.

Early results of the collaboration include the following:

- Enhanced shuffle engine technologies: Faster sorting and in-memory computations, which has the potential to dramatically improve Spark performance.
- Better memory utilization: Improved performance and usage for broader scalability, which will help enable new large-scale use cases.

For more information :

Michelle Lazzar
408-828-9681
comms@hortonworks.com

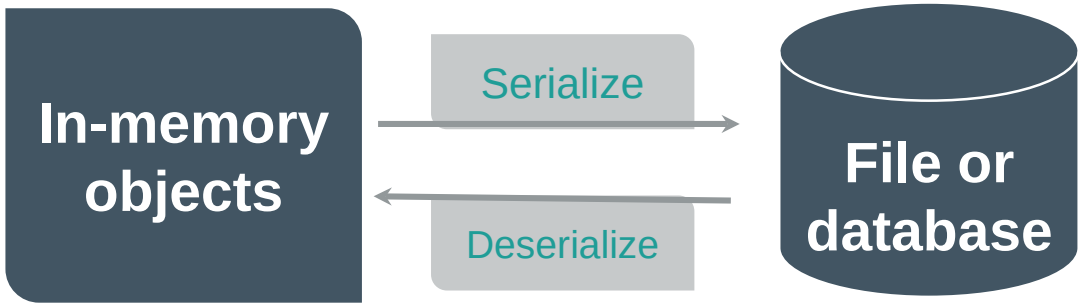
Hewle
Enterp

Memory driven programming models

Do we need separate data representations?

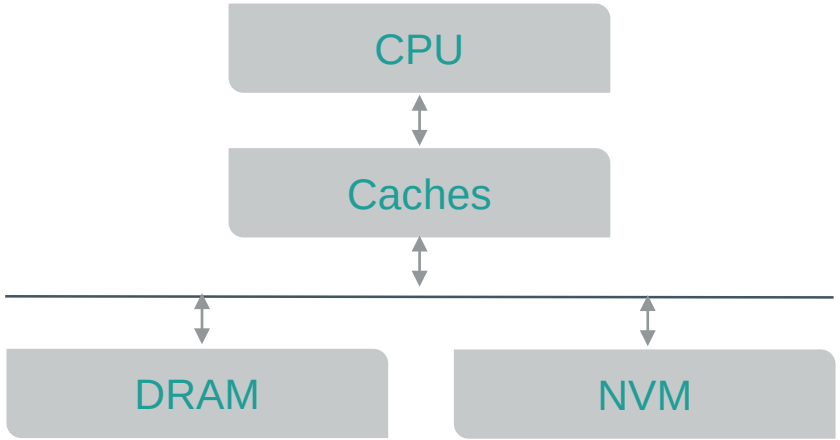
In-storage durability

+ Separate object and persistent formats
  – Programmability and performance issues
  – Translation code error-prone and insecure



In-memory durability

+ In-memory objects are durable throughout
+ Byte-addressability simplifies programmability
+ Low ld/st latencies offer high performance
  – Persistent does not mean consistent!



NVM-aware application programming

Why can’t I just write my program, and have all my data be persistent?

Consider a simple banking program (just two accounts):

```
double accounts[2];
```

I want to transfer money between accounts. Naïve implementation:

```
transfer(int from, int to, double amount) {
    accounts[from] -= amount;
    accounts[to] += amount;
}
```

What if I crash here?

What if I crash here?

Crashes cause corruption, which prevents us from merely restarting the computation

Persistent memory programming – Hewlett Packard Labs

- Manage consistent updates with failure atomicity
- Handle recovery
- Support multi-threaded concurrent access

- Atlas: Persistence for lock-based, multhreaded shared memory programs
  - C/C++11
  - Arbitrary data structures
  - Operate directly on persistent memory within critical sections (lock-based or TM-style transactions)
- Managed Data Structures (MDS) in persistent memory
  - C++11/Java8
  - Specific data structures
  - Library mediated access with ACID transactions with configurable isolation

Manual solution

```
persistent double accounts[2];
transfer(int from, int to, double amount) {
<save old value of accounts[from] in undo log>;
<flush log entry to NVM>
                              accounts[from] -= amount;
<save old value of accounts[to] in undo log>;
<flush log entry to NVM>
                              accounts[to]   += amount;
<flush all other persistent stores to NVRAM>
<clear and flush log>
}
```

- Need code that plays back undo log on restart
- Getting this to work with threads and locks is very hard
- Really want to optimize it
- **Very unlikely application programmers will get it right**

Atlas solution: Consistent sections

Provide a construct that atomically updates NVM

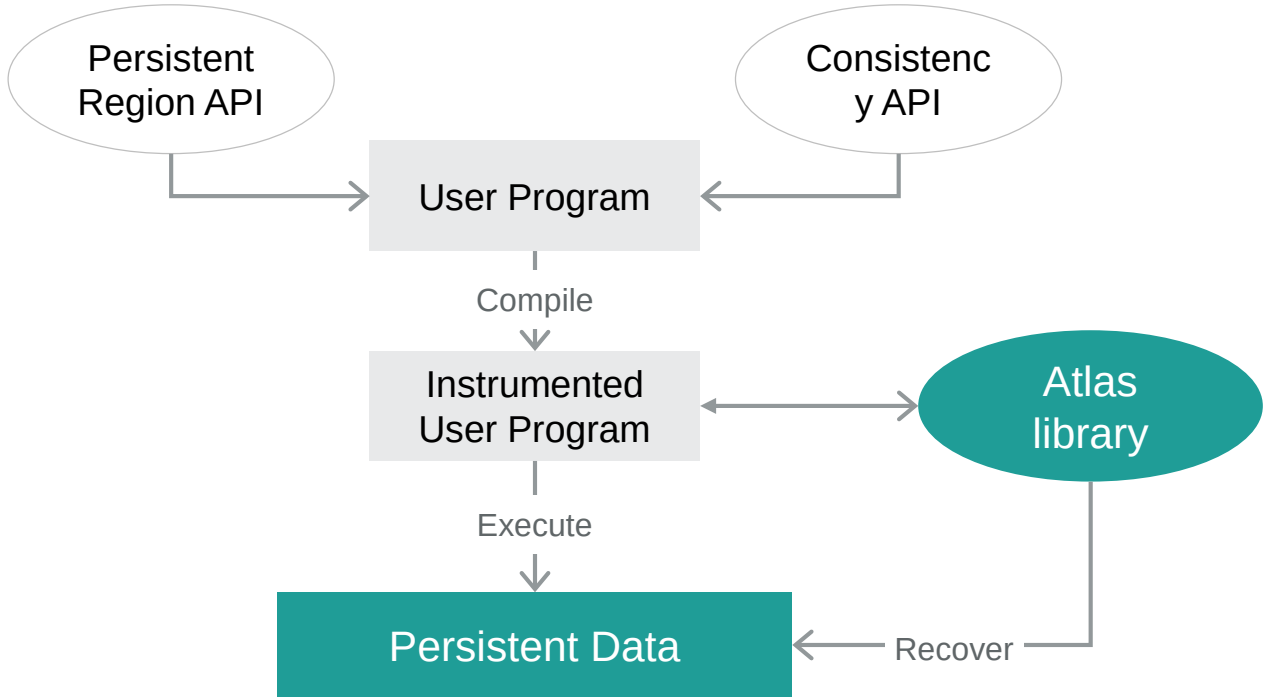
```
persistent double accounts[2];
transfer(int from, int to, double amount) {
   __atomic {
                accounts[from] -= amount;
                accounts[to]   += amount;
    }
}
```

- Updates in **__atomic** block are either completely visible after crash or not at all
- If updates in **__atomic** block are visible, then so are prior updates to persistent memory

D. Chakrabarti, H. Boehm and K. Bhandari. “Atlas: Leveraging Locks for Non-volatile Memory Consistency,” *Proc. OOPSLA*, 2014.

The Atlas programming model
Ensure data consistency in persistent memory

- Programmer distinguishes persistent and transient data
- Persistent data lives in a “persistent region”
  - E.g., in pseudo-file-system in NVM
  - Directly mapped into process address space (no DRAM buffers)
  - Accessed via CPU loads and stores
- Programmer writes ordinary multithreaded code
  - Automatic durability support at a fine granularity, complete with recovery code
  - Supports consistency of durable data derived from concurrency constructs
- Open source code available at https://github.com/HewlettPackard/Atlas

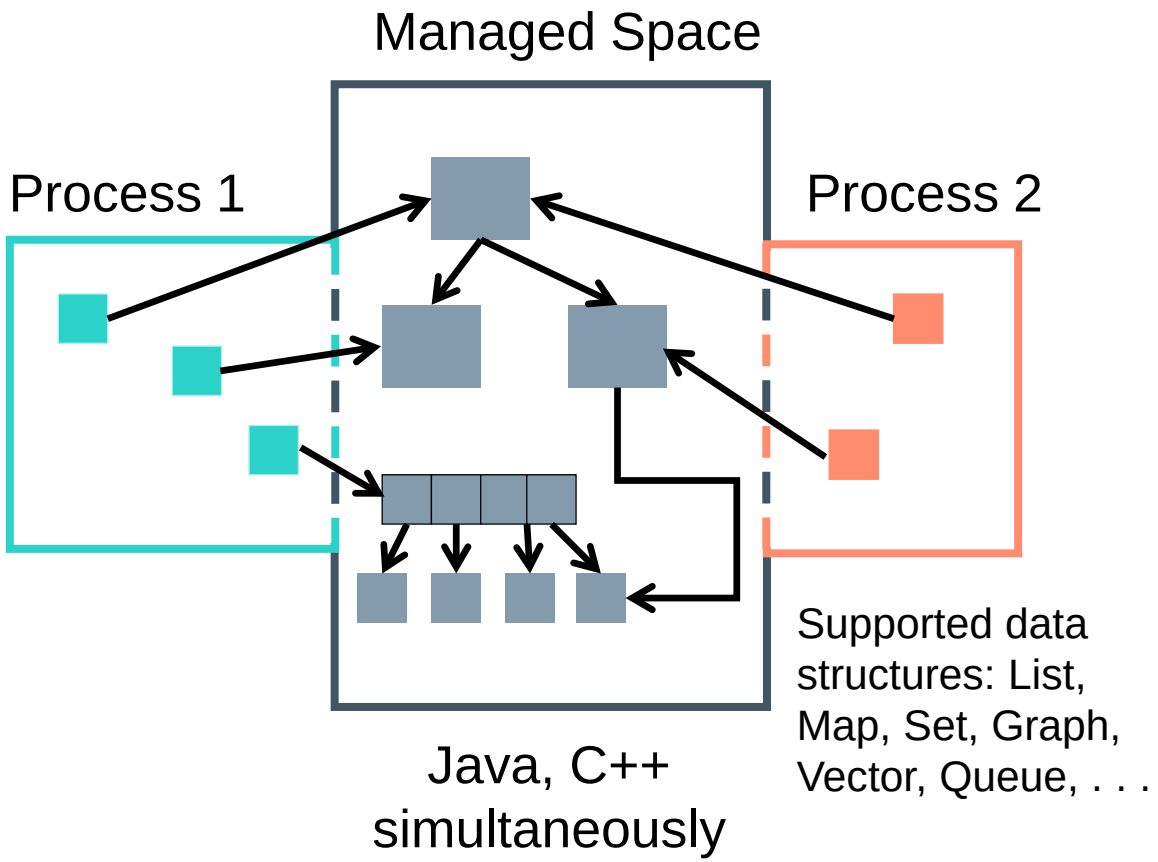


D. Chakrabarti, H. Boehm and K. Bhandari. “Atlas: Leveraging Locks for Non-volatile Memory Consistency,” *Proc. Object-Oriented Programming, Systems, Languages & Applications* (*OOPSLA)*, 2014.

Managed Data Structures (MDS)

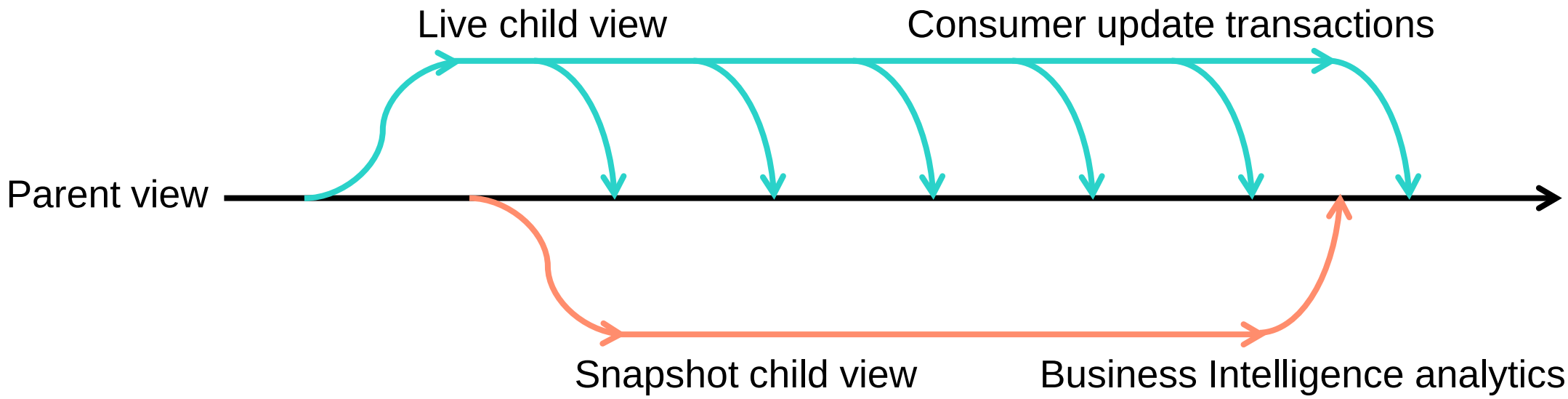
Simplify programming on persistent in-memory data

- Ease of Programming
  - Programmer manages only application-level data structures
    - 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 if it was local to the program
  - High-level concurrency controls
    - Ensure consistent data in the face of data sharing by multiple threads/processes



Hewlett Packard Enterprise

Supporting safe data sharing



Non-blocking transactions

Zero-copy snapshots

Conflict resolution

Hewlett Packard Enterprise

Simplified data management – Benefits to application developer

Conventional Data Formats

- Data structures
- Data format conversion
- Serialisation/deserialisation
- RPC, HTTP, message passing
- Disk communication latency
- Server
- Database
- Filesystem
- Disk

MDS Data Formats

- Data structures
- Local function calls
- Shared non-volatile memory

Shorter path to persistence

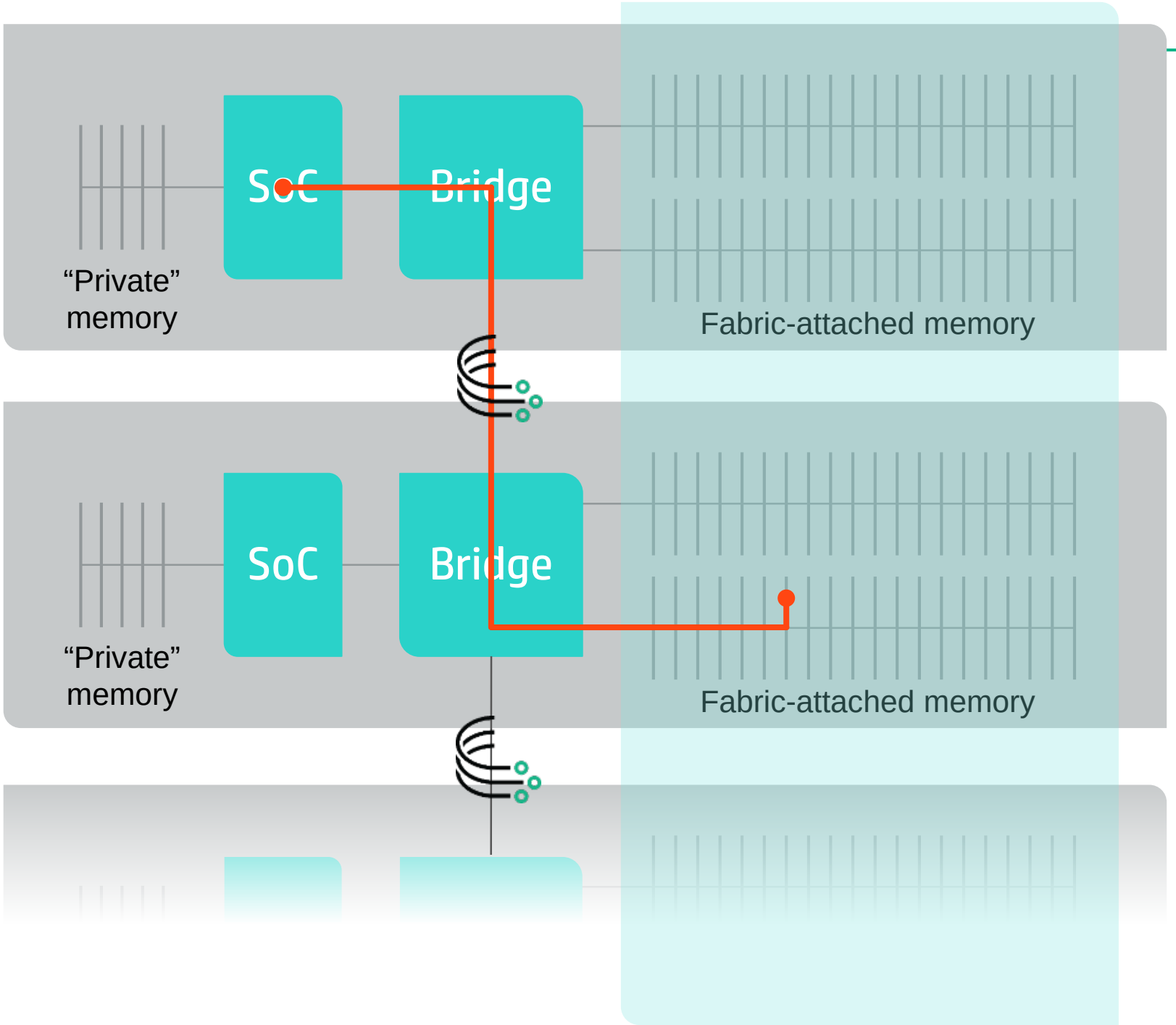
Less code

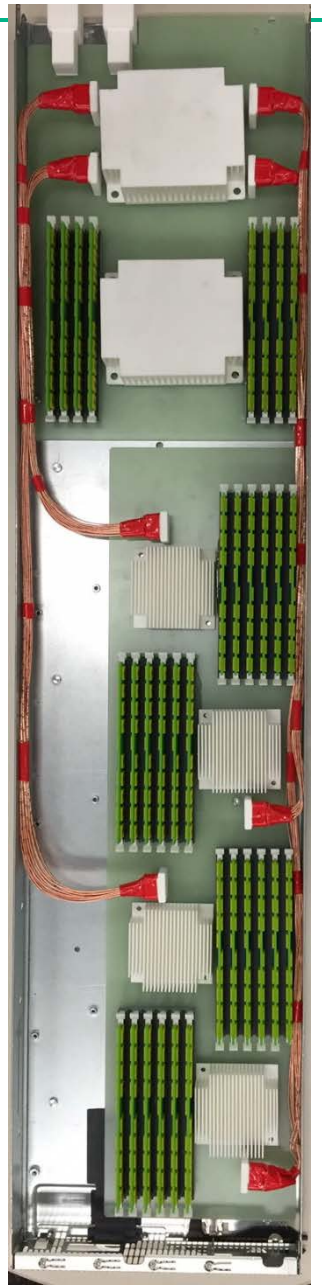
Fewer errors

Faster development

Hewlett Packard Enterprise

Prototypes and emulators



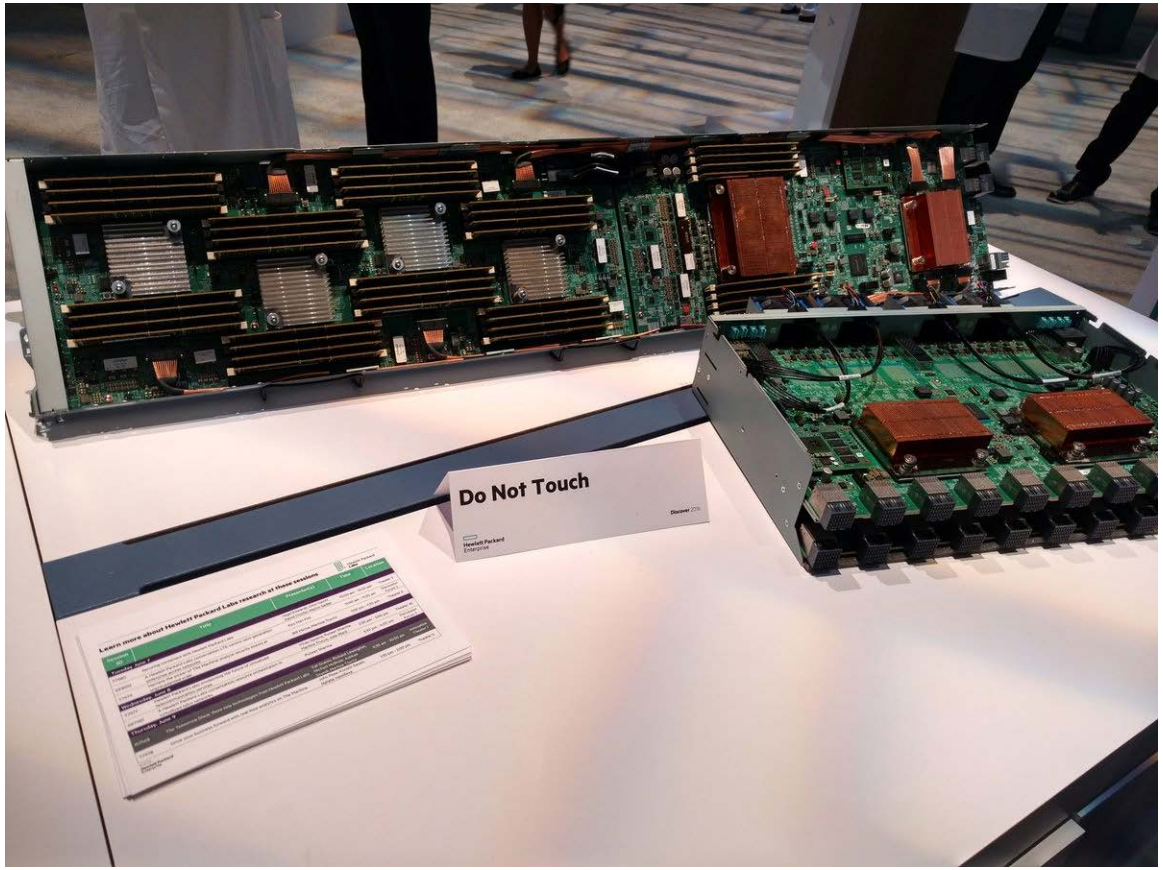


Bridge

SoC + private memory

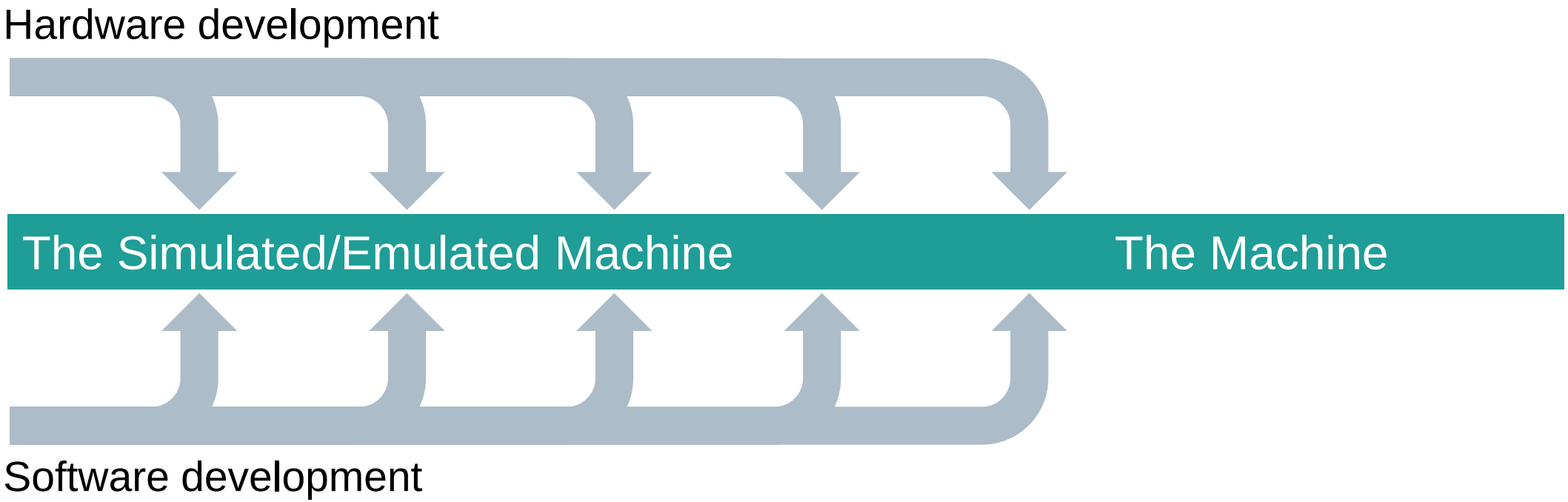
Fabric-attached memory (FAM)

Show and tell at HPE Discover 2016



Hewlett Packard Enterprise

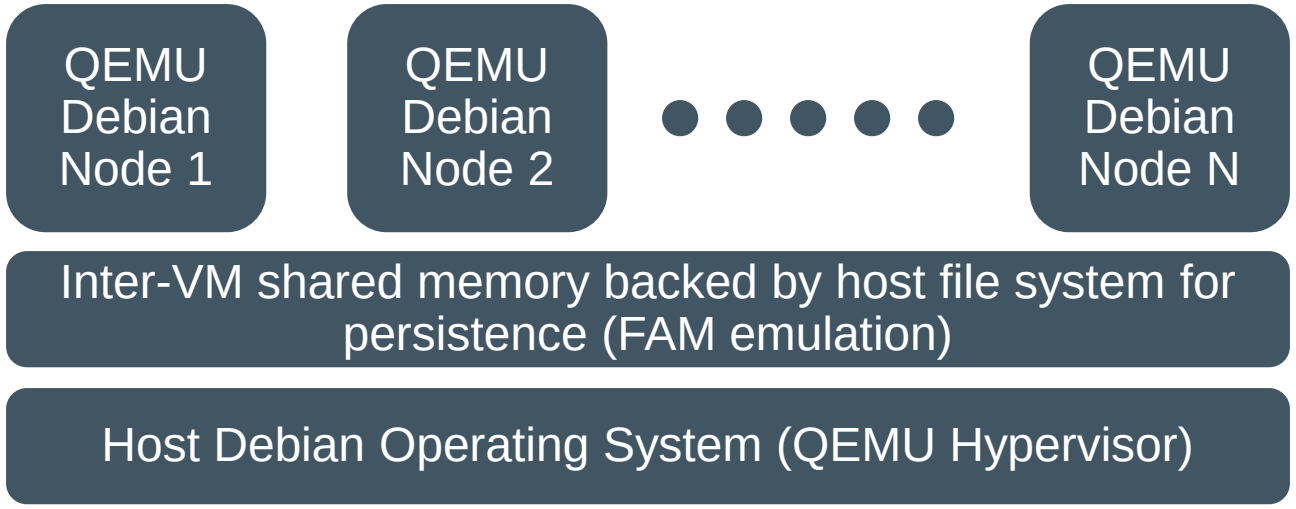
Hardware/software co-development



Fabric-Attached Memory Emulation for The Machine

Code for memory driven architecture

- Provides a programmer's view of fabric attached memory
  - QEMU virtual machines running Debian mimic compute nodes
  - Shared memory on the host emulates fabric attached memory (FAM)
  - IVSHMEM links guests and host
- Performant environment allows developers to
  - Create code for The Machine architecture
  - Modify legacy code to take advantage of The Machine architecture
- Open source code available at https://github.com/FabricAttachedMemory



Hewlett Packard Enterprise

Quartz: NVM Performance Emulator

- Quartz: *performance emulator* for NVM based on commodity hardware (DRAM)
- Focus: modeling primary *performance characteristics of NVM* that affect application end-to-end performance
  - Two performance knobs for NVM emulation: *bandwidth* and *latency*
  - *Non-goals:* accurate simulation of NVM features, NVM functionality, and NVM devices…
- *Quartz aims to support:*
  - *Sensitivity analysis* of complex applications on future hardware
    - Which ranges of latencies and bandwidth are critical for achieving good performance and scalability?
  - Design questions for future machines with both DRAM and NVM
    - DRAM as cache for NVM vs. DRAM and NVM as peers
    - Strategies for efficient data placement

Open source code available at
https://github.com/HewlettPackard/Quartz
H. Volos, G. Magalhaes, L. Cherkasova, J. Li. Quartz: A Lightweight Performance Emulator for Persistent Memory Software. *Proc. of ACM/Usenix/IFIP Conference on Middleware, 2015.*

Commercial Memory-Driven Computing

Hewlett Packard Enterprise

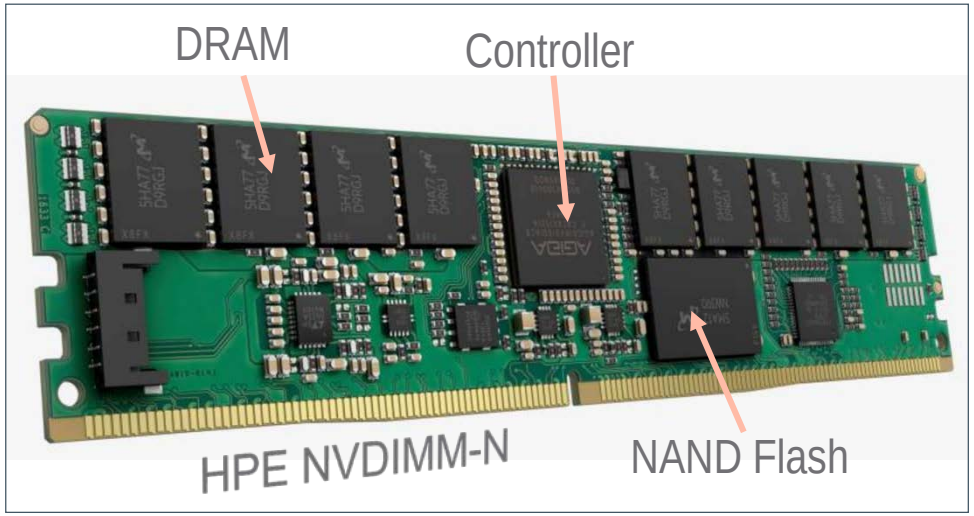
HPE Integrity Superdome X

Large-scale shared memory multiprocessor

– Up to 16 processors, 384 cores
  – Intel® Xeon® Processor E7 v4 and E7 v3 family
– Up to 24 TB DDR4 memory (Gen9)
  – 384 DIMM slots
– 24 Mezz PCIe gen3 slots (3 per blade) for IO connectivity to LAN, SAN, and InfiniBand
– 18U enclosure
– https://www.hpe.com/us/en/servers/superdome.html

Hewlett Packard Enterprise

HPE Persistent Memory

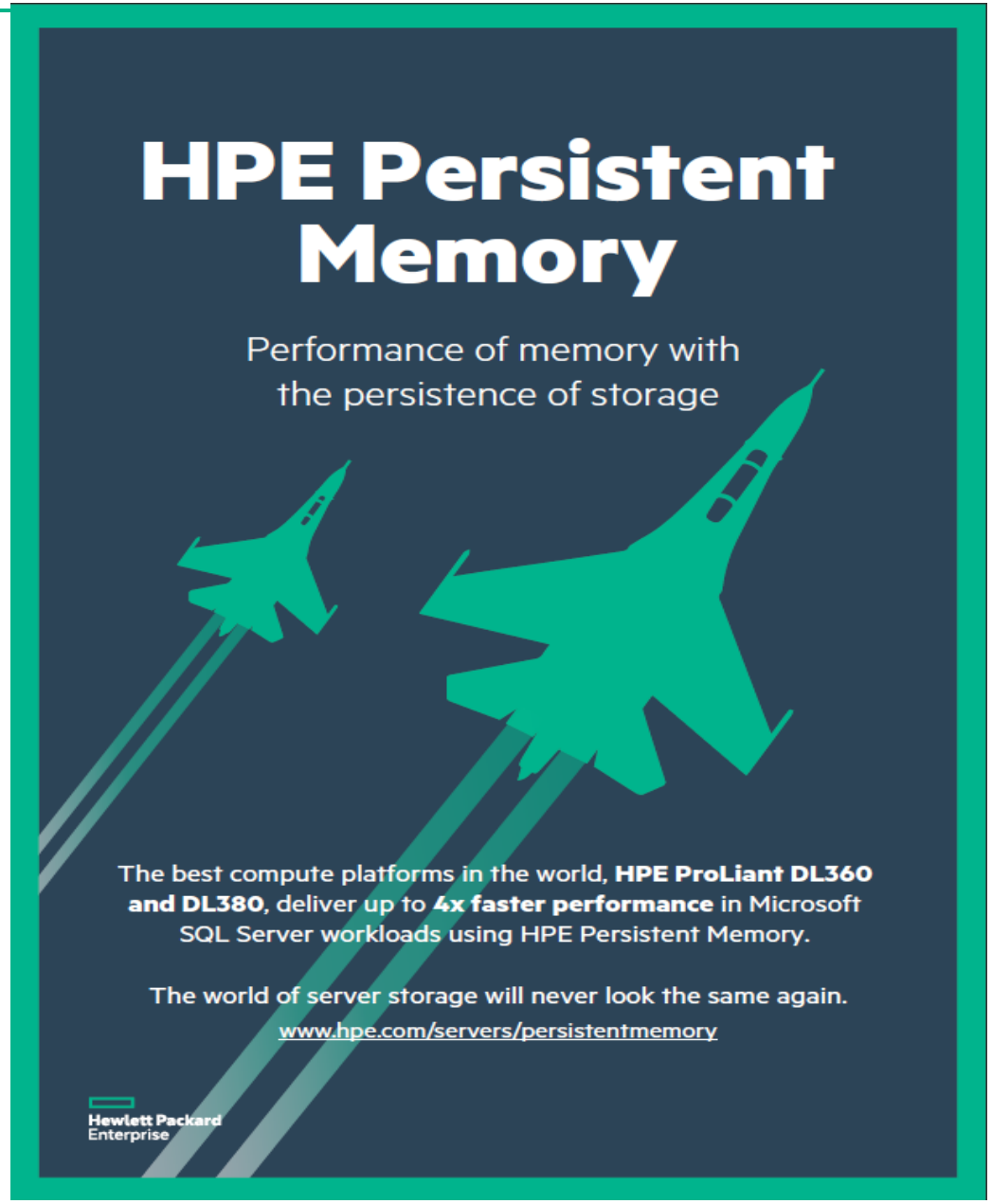


– HPE 8GB NVDIMM Module (782692-B21)

– HPE ProLiant Gen9 Servers Supported and Configurations
  – DL360 Gen9 and DL380 Gen9 E5-2600v4

– Ideal for accelerating databases and analytics workloads

– https://www.hpe.com/us/en/servers/persistent-memory.html



Hewlett Packard Enterprise

Gen-Z Consortium

- Industry Leaders developing a memory-semantic interconnect

Memory-Semantic Fabric

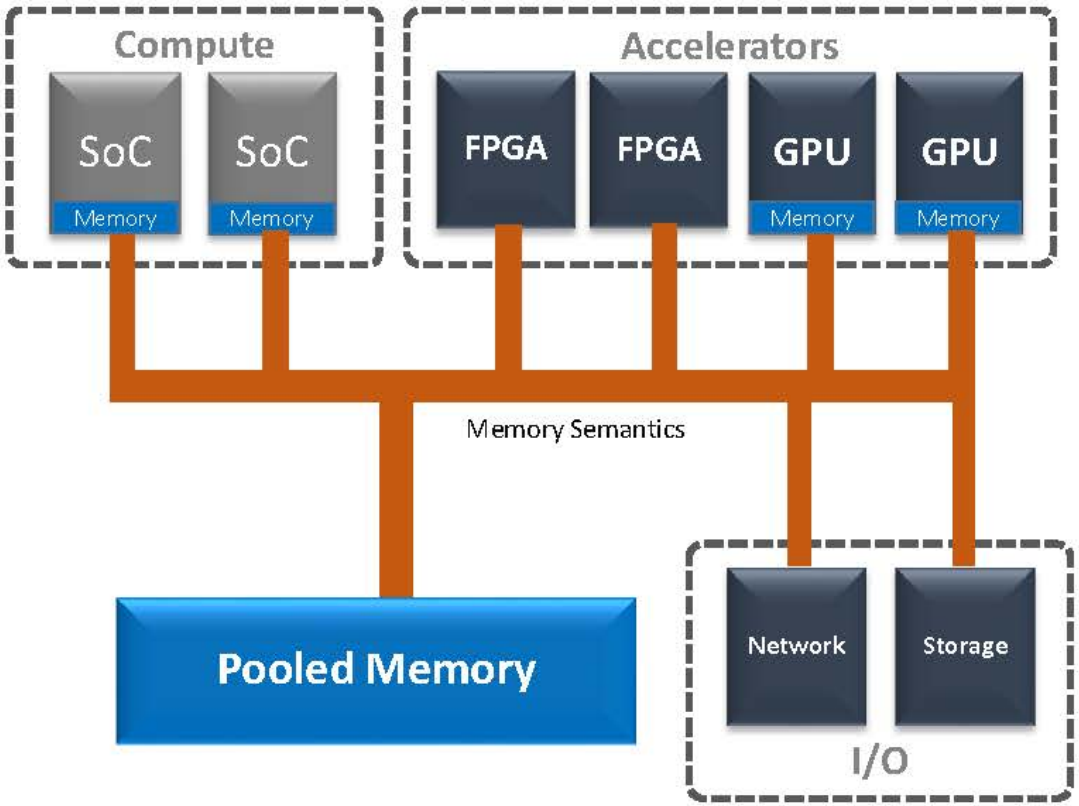
- Communication at the speed of Memory

What is a Memory Semantic Fabric?

- Handles all communication as memory operations such as load/store, put/get and atomic operations typically used by a processor
- Memory semantics are optimal at sub-microsecond latencies from CPU load command to register store
  - Unlike, storage accesses which are block based and managed by complex, code intensive, software stacks

Why Now?

- The emergence of low latency, Storage Class Memory (SCM) and the demand for large capacity, rack scale resource pools, and multi node architectures



Gen-Z: A New Data Access Technology

High Bandwidth Low Latency

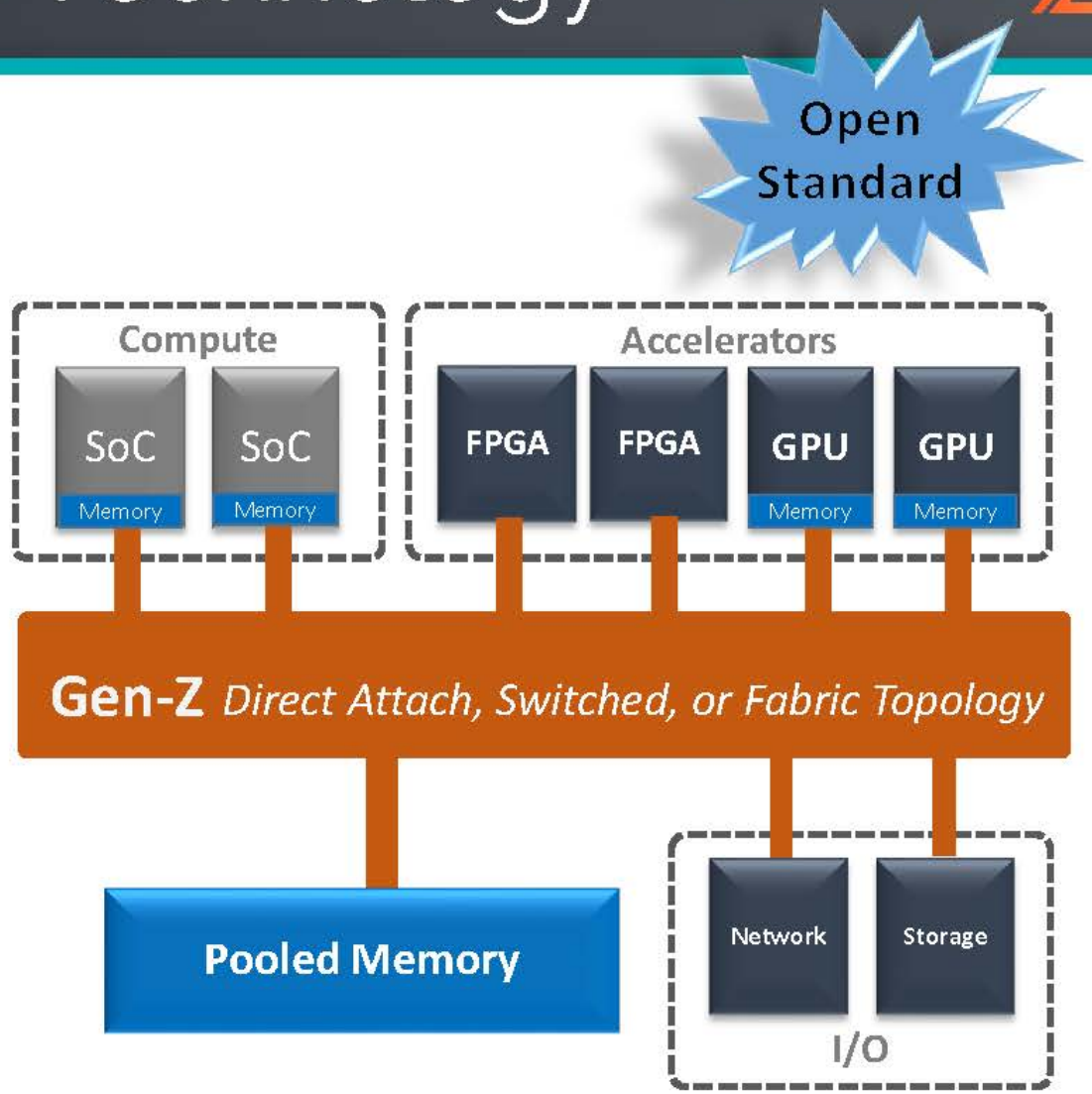
- Memory Semantics – simple Reads and Writes
- From tens to several hundred GB/s of bandwidth
- Sub-100 ns load-to-use memory latency

Advanced Workloads & Technologies

- Real time analytics
- Enables data centric and hybrid computing
- Scalable memory pools for in memory applications
- Abstracts media interface from SoC to unlock new media innovation

Secure Compatible Economical

- Provides end-to-end secure connectivity from node level to rack scale
- Supports unmodified OS for SW compatibility
- Graduated implementation from simple, low cost to highly capable and robust
- Leverages high-volume IEEE physical layers and broad, deep industry ecosystem



Rack Scale

For more information...

http://www.labs.hpe.com/research/themachine/

- D. Chakrabarti, H. Volos, I. Roy, and M. Swift. “How Should We Program Non-volatile Memory?”, tutorial at *ACM Conf. on Programming Language Design and Implementation (PLDI)*, 2016.
- I. El Hajj, A. Merritt, G. Zellweger, D. Milojicic, W. Hwu, K. Schwan, T. Roscoe, R. Achermann, and P. Faraboschi. “SpaceJMP: Programming with multiple virtual address spaces,” *Proc. ACM Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016.
- J. Izraelevitz, T. Kelly, A. Kolli, “Failure-atomic persistent memory updates via JUSTDO logging,” *Proc. ACM Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2016.
- K. Bresniker, S. Singhal, and S. Williams. “Adapting to thrive in a new economy of memory abundance,” *IEEE Computer*, December 2015.
- H. Volos, G, Magalhaes, L, Cherkasova, J, Li. “Quartz: A lightweight performance emulator for persistent memory software,” *Proc. of ACM/Usenix/IFIP Conference on Middleware*, 2015.
- H. Kimura, “FOEDUS: OLTP engine for a thousand cores and NVRAM,” *Proc. ACM SIGMOD*, 2015.
- P. Faraboschi, K. Keeton, T. Marsland, D. Milojicic, “Beyond processor-centric operating systems,” *Proc. USENIX Workshop on Hot Topics in Operating Systems (HotOS)*, 2015.
- S. Gerber, G. Zellweger, R. Achermann, K. Kourtis, and T. Roscoe, D. Milojicic. “Not your parents’ physical address space,” *Proc. USENIX HotOS,* 2015.
- S. Novakovic, K. Keeton, P. Faraboschi, R. Schreiber, E. Bugnion. “Using shared non-volatile memory in scale-out software,” *Proc. ACM Intl. Workshop on Rack-scale Computing (WRSC)*, 2015.
- M. Swift and H. Volos. “Programming and usage models for non-volatile memory,” Tutorial at *ACM ASPLOS*, 2015.
- D. Chakrabarti, H. Boehm and K. Bhandari. “Atlas: Leveraging locks for non-volatile memory consistency,” *Proc. ACM Conf. on Object-Oriented Programming, Systems, Languages & Applications* (*OOPSLA)*, 2014.
- H. Volos, S. Nalli, S. Panneerselvam, V. Varadarajan, P. Saxena, M. Swift. "Aerie: Flexible file-system interfaces to storage-class memory," *Proc. ACM EuroSys*, 2014.

For open source code...

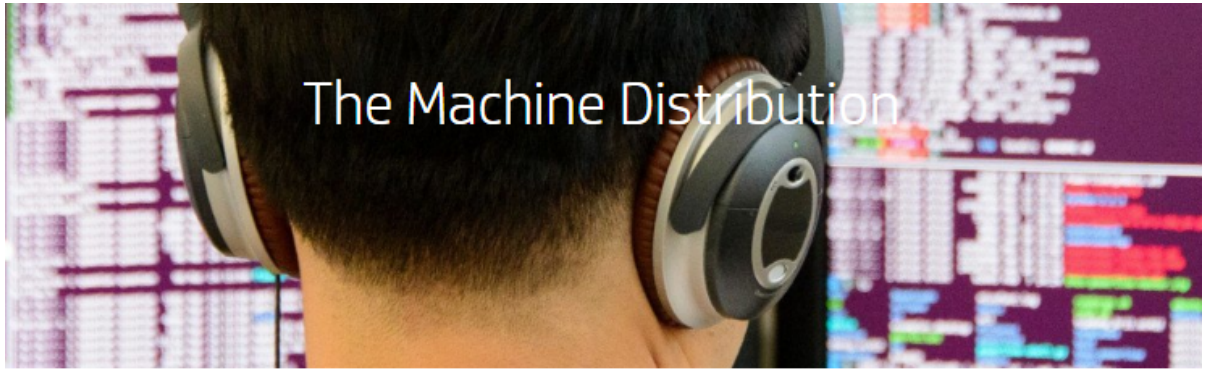
http://www.labs.hpe.com/research/themachine/TheMachineDistribution/

– Fast optimistic engine for data unification services (FOEDUS): https://github.com/hkimura/foedus

– Fault-tolerant programming model for non-volatile memory (Atlas): https://github.com/HewlettPackard/Atlas

– Fabric Attached Memory Emulation: https://github.com/FabricAttachedMemory/Emulation

– Performance emulation for non-volatile memory latency and bandwidth (Quartz): https://github.com/HewlettPackard/Quartz



Research / The Machine / The Machine Distribution

Time to get Machine-ready

With the fundamental shift in how The Machine will work, it's important that we invite open source developers very early in the software development cycle to start familiarizing with new non-volatile Memory-Driven Computing programming models.

As part of HPE's ongoing commitment and active participation in the open source community, The Machine's development team will work transparently and invite developers work on Memory-Driven Computing. We'll start with familiar environments like Linux and Portable Operating System Interface APIs with programming languages like C/C++ and Java and make the performance advantages of massive memory on fabrics available to them in a way that they can be productive quickly.

During the coming months, we will continue to add to the developer tools. The four contributions of code available today are:

1. Fast optimistic engine for data unification services: A completely new database engine that speeds up applications by taking advantage of a large number of CPU cores and non-volatile memory (NVM).
2. Fault-tolerant programming model for non-volatile memory: Adapts existing multi-threaded code to store and use data directly in persistent memory, provides simple, efficient fault-tolerance in the event of power failures or program crashes.
3. Fabric Attached Memory Emulation: An environment designed to allow users to explore the new architectural paradigm of The Machine.
4. Performance emulation for non-volatile memory bandwidth: A DRAM-based performance emulation platform that leverages features available in commodity

Related articles

Hortonworks: HPE and Hortonworks collaborate to bring big-memory Spark to the enterprise

TheNextPlatform: Programming For Persistent Memory Takes Persistence

TheNextPlatform: First Steps In The Program Model For Persistent Memory

TheNextPlatform: Spark on Superdome X Previews In-Memory On The Machine

Hewlett Packard Enterprise

Wrapping up

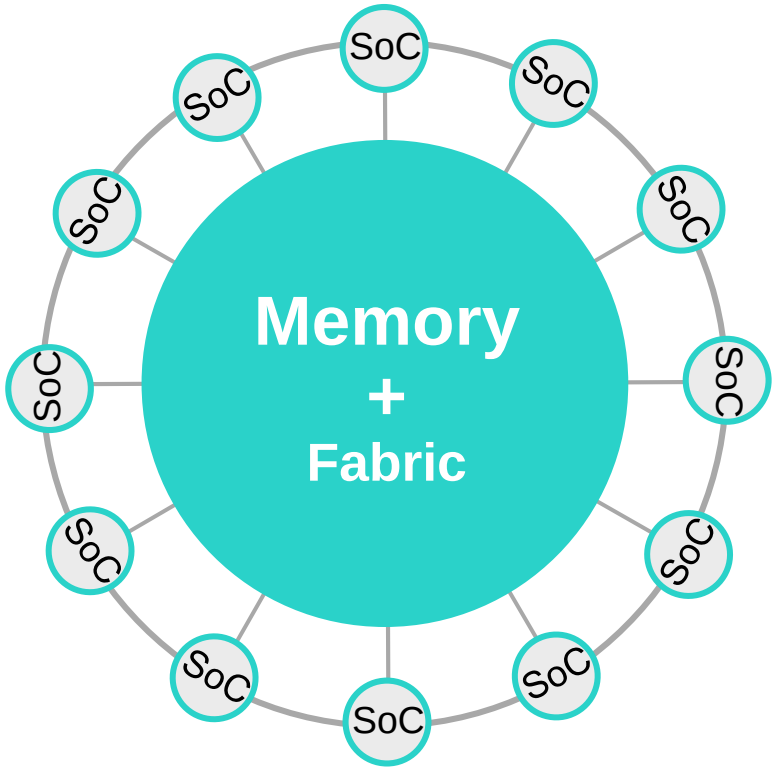
Memory-Driven Computing

- Fast load/store access to large shared pool of fabric-attached non-volatile memory

Many opportunities for software innovation

- Operating systems
- Data stores
- Analytics platforms
- Programming models and tools
- Applications
- Algorithms

How would *you* exploit Memory-Driven Computing?



Hewlett Packard Enterprise