Hewlett Packard
Enterprise

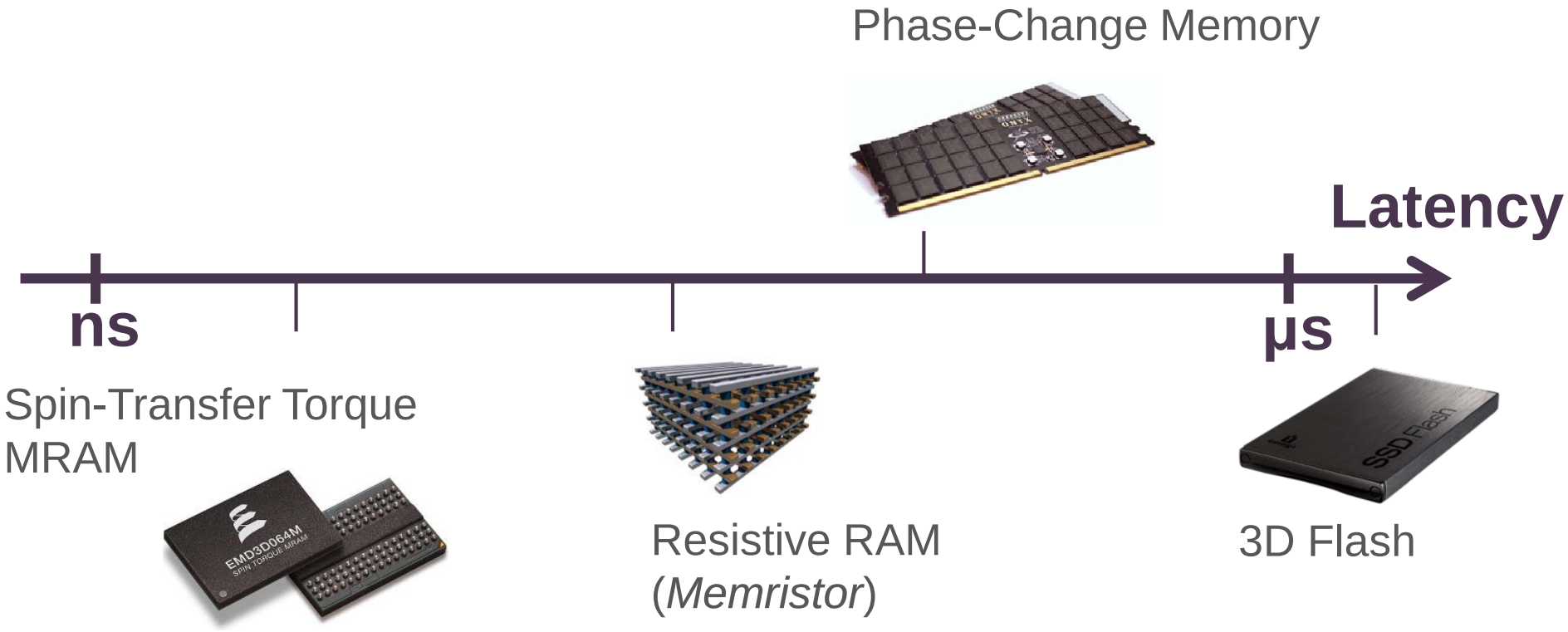
OpenFAM API: programming model for disaggregated persistent memory

Kimberly Keeton, Sharad Singhal, Haris Volos
{kimberly.keeton,sharad.singhal,haris.volos}@hpe.com

MSST 2018

Hewlett Packard
Labs

Non-Volatile Memory (NVM)



– Persistently stores data
– Access latencies comparable to DRAM
– Byte addressable (load/store) rather than block addressable (read/write)
– 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

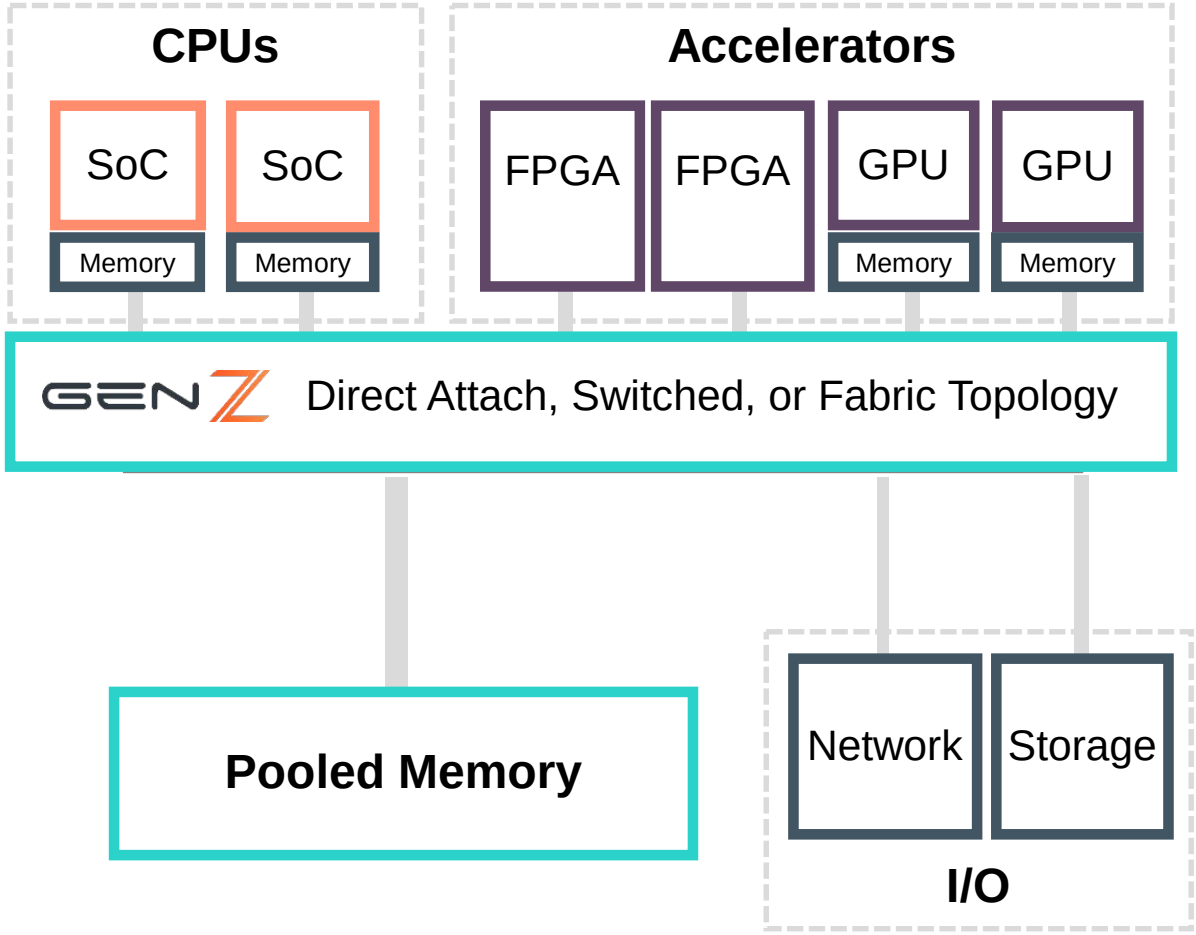
– Members: 40+ companies covering SoC, memory, I/O, networking, mechanical, system software, etc.

– 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)

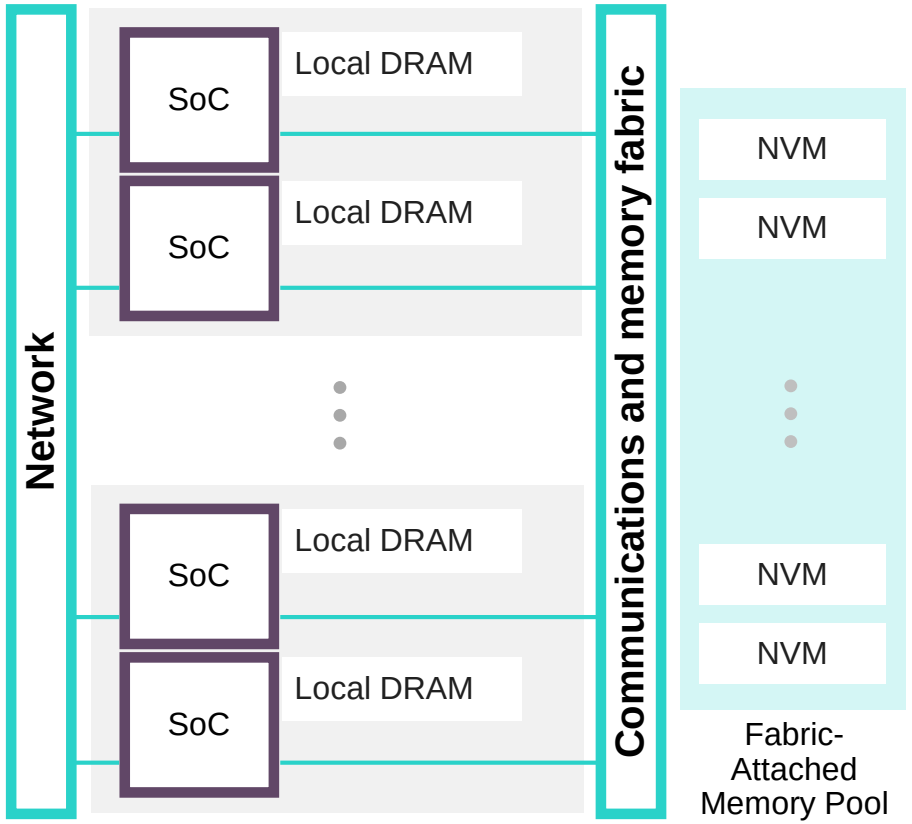
– High performance
  – Tens to hundreds GB/s bandwidth
  – Sub-microsecond load-to-use memory latency

– *Spec available for public download*



Hewlett Packard Enterprise

Fabric-attached (persistent) memory

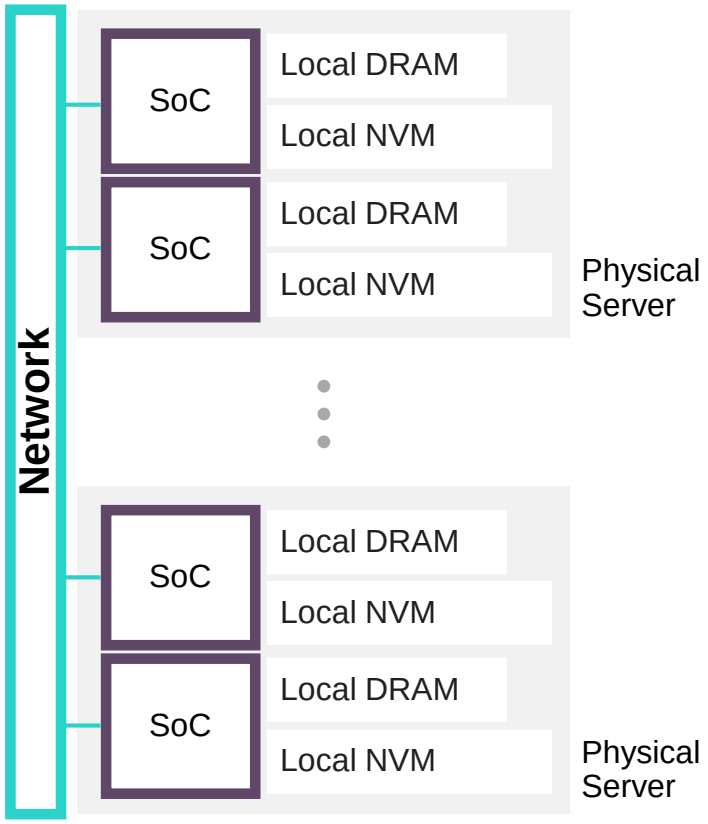


- **Converging memory and storage**
  - Byte-addressable NVM replaces hard drives and SSDs
- **Resource disaggregation leads to high capacity shared memory pool**
  - Fabric-attached memory pool is accessible by all compute resources
  - Low diameter networks provide near-uniform low latency
- **Local 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

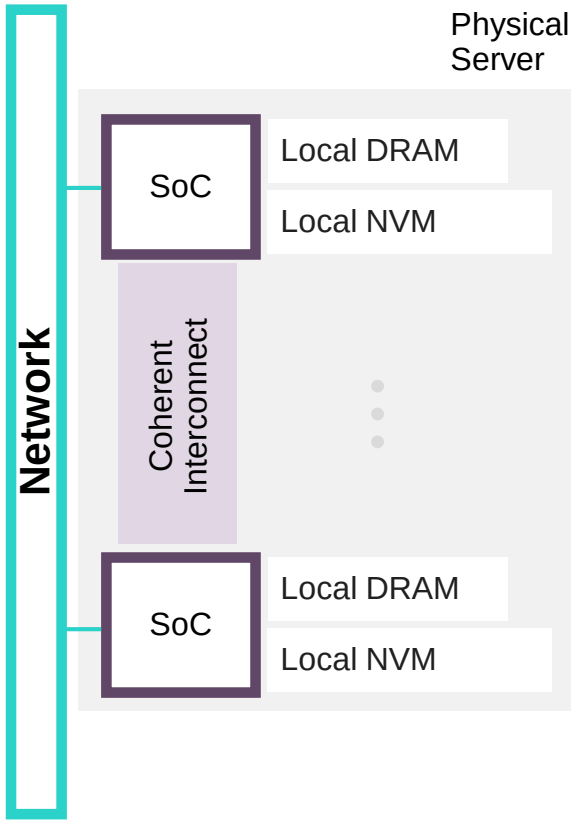
- **Enables Memory-Driven Computing**

Hewlett Packard Enterprise

Memory-Driven Computing in context



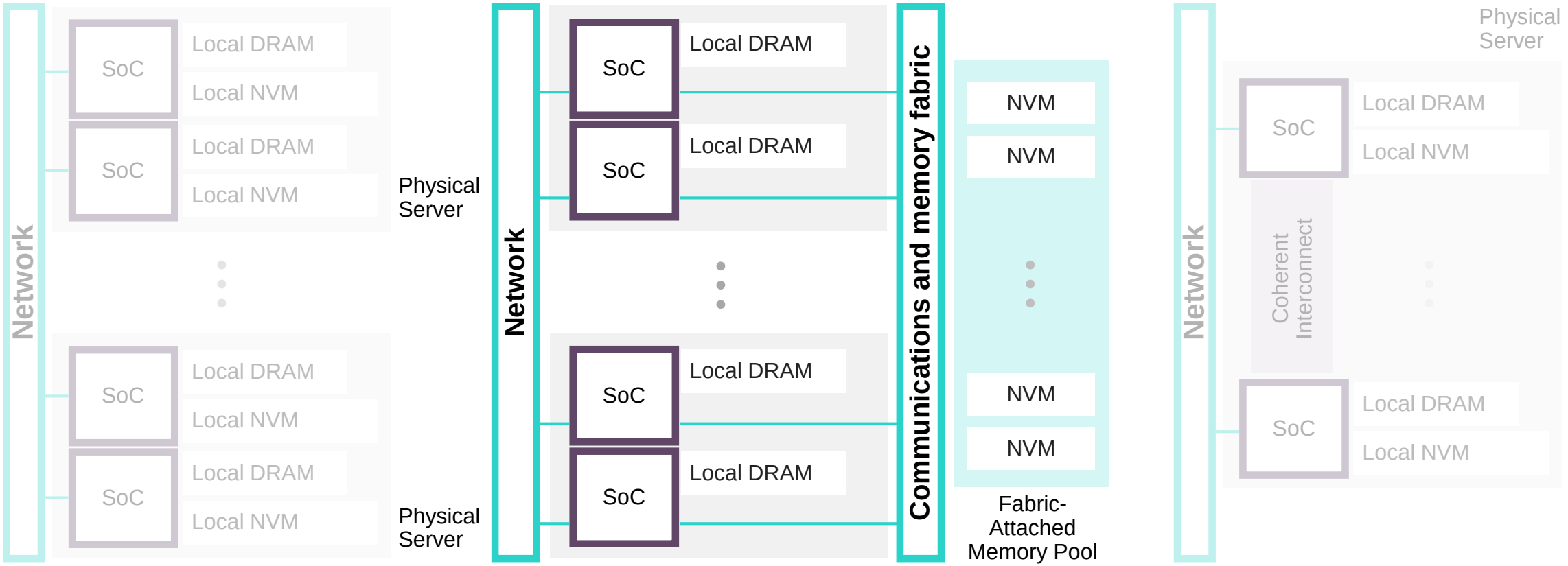
Shared nothing



Shared everything

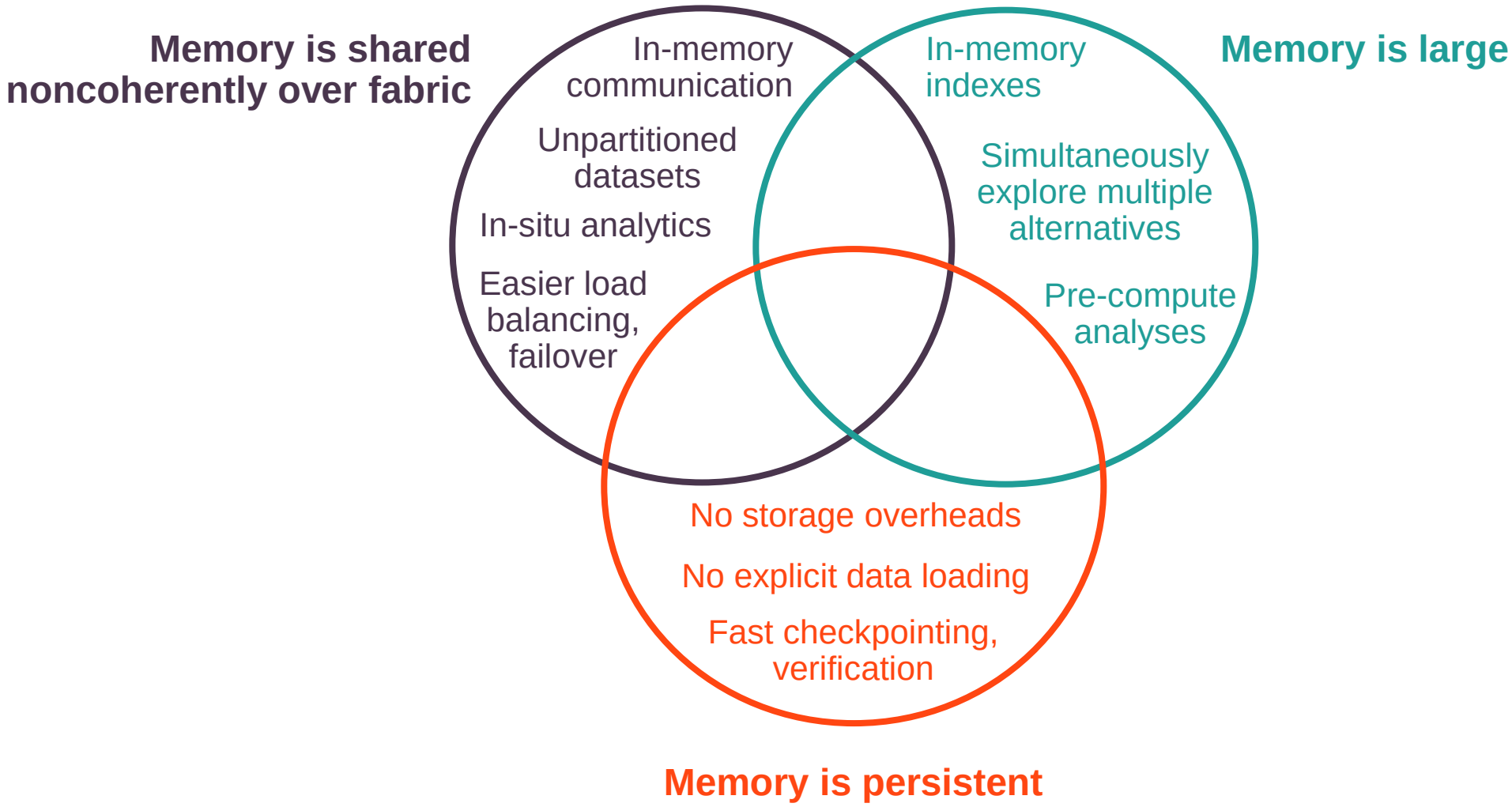
Hewlett Packard Enterprise

Memory-Driven Computing in context



Hewlett Packard Enterprise

Memory-Driven Computing benefits applications



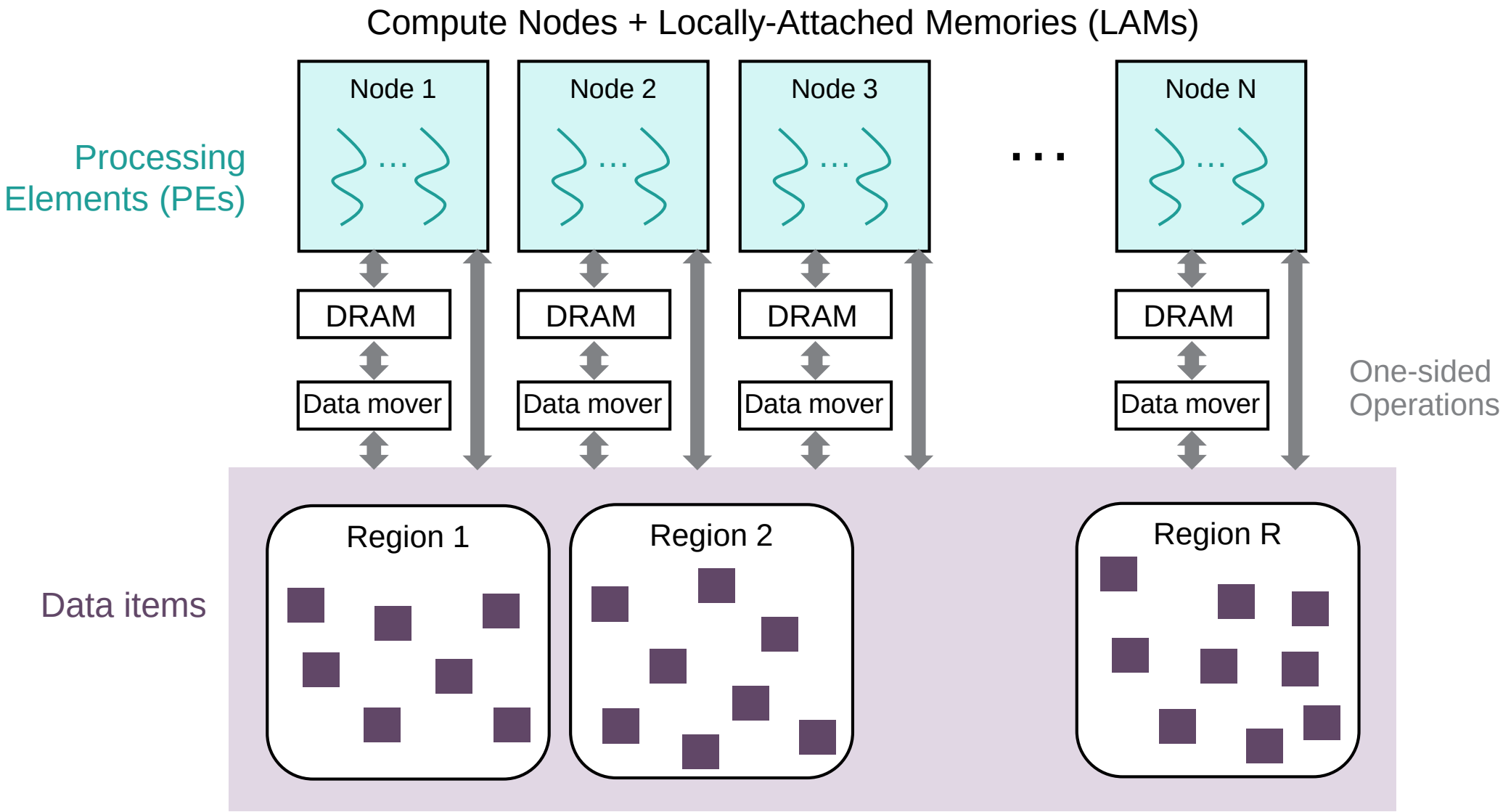
Note: work in progress

OpenFAM: programming model for fabric-attached memory

– Inspired by
  – OpenSHMEM (http://openshmem.org): open source partitioned global address space (PGAS) library with one-sided communication, atomic and collective operations
  – HPE's work on The Machine program (https://www.labs.hpe.com/the-machine)

– Difference #1: Fabric-attached memory (FAM) instead of remote processing element (PE) memory
  – We assume that node memory is treated as private and that FAM is shared
– Difference #2: FAM may be persistent. Data should be able to live beyond program invocation.
– One-sided/unmediated access to fabric-attached memory
  – Find useful intersection between RDMA primitives, Gen-Z primitives and OpenSHMEM APIs
– Keep it as simple as possible: always possible to add on later

OpenFAM concepts

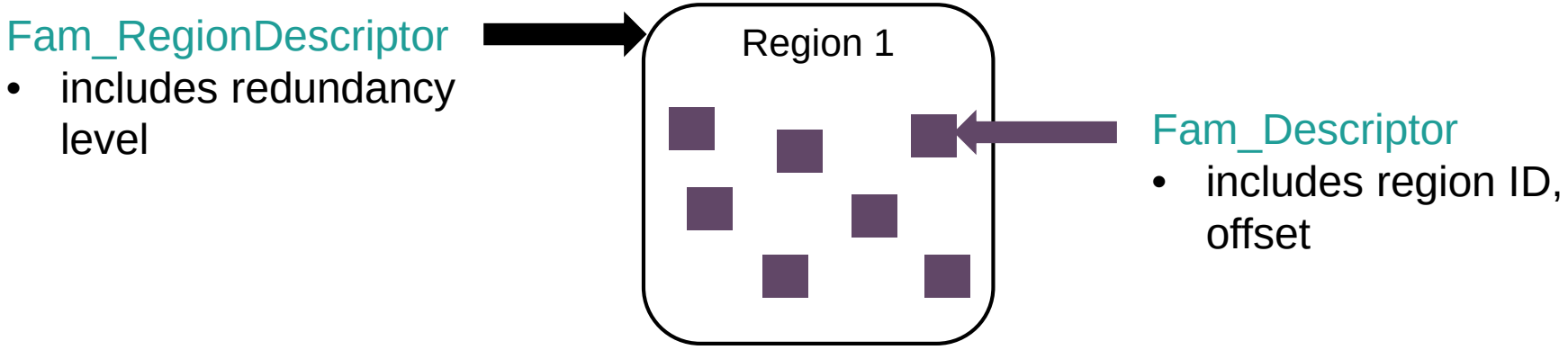


Regions vs. data items

- Regions define sections of FAM with specific characteristics (e.g., persistence, redundancy)
- Useful to permit multiple regions associated with a given job to accommodate different data needs. Ex:
  - No redundancy for communication or scratch space
  - Redundancy for computation results
- Named regions of FAM enable sharing between PEs of a given job and also between jobs of a workflow (for persistent data)
- Region forms basis for heap allocator in memory management routines
  - Data items are allocated using heap allocator

Descriptors

- Descriptors are opaque read-only data structures that uniquely identify a location in FAM
- Descriptors are portable across OS instances
  - Use base + offset addressing
  - Can be freely copied and shared across processing nodes by the program



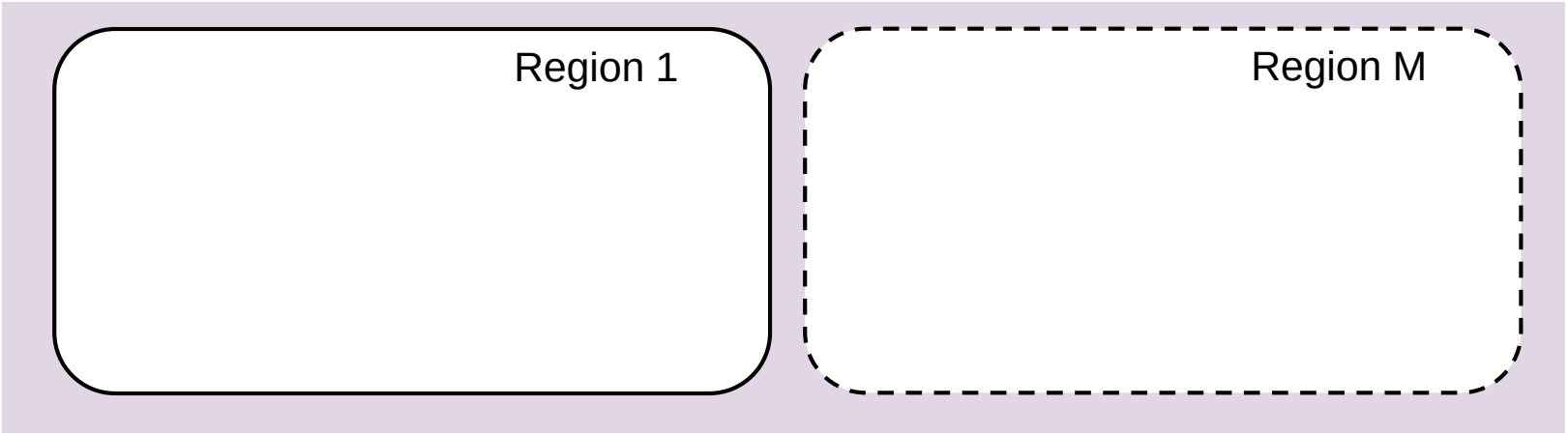
Allocation (region management)

– **fam_create_region:** allocates space for region, with associated options (size, redundancy, persistence, permissions, name)

– **fam_destroy_region:** used to indicate that user is done with allocated region

  – Triggers delayed reclamation to accommodate ongoing users

– **fam_resize_region:** change size of a region allocation



Global Shared Non-volatile Memory (aka Fabric-Attached Memory (FAM))

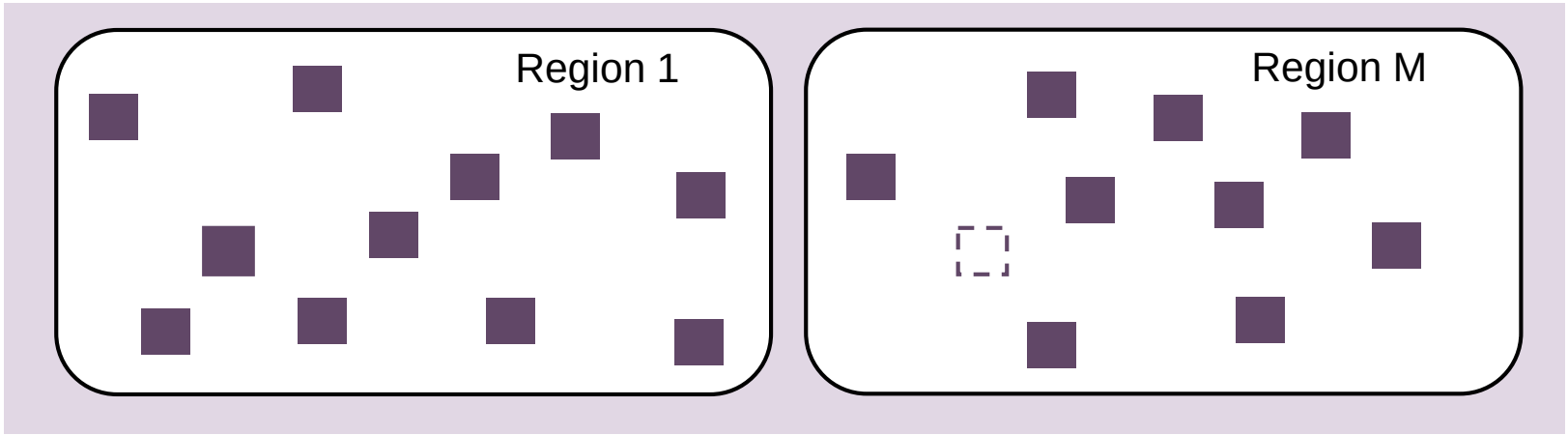
Allocation (data item / heap management)

– **fam_allocate:** allocates space for data item within a region, with associated options (size, permissions, name)

– **fam_deallocate:** used to indicate that user is done with allocated data item

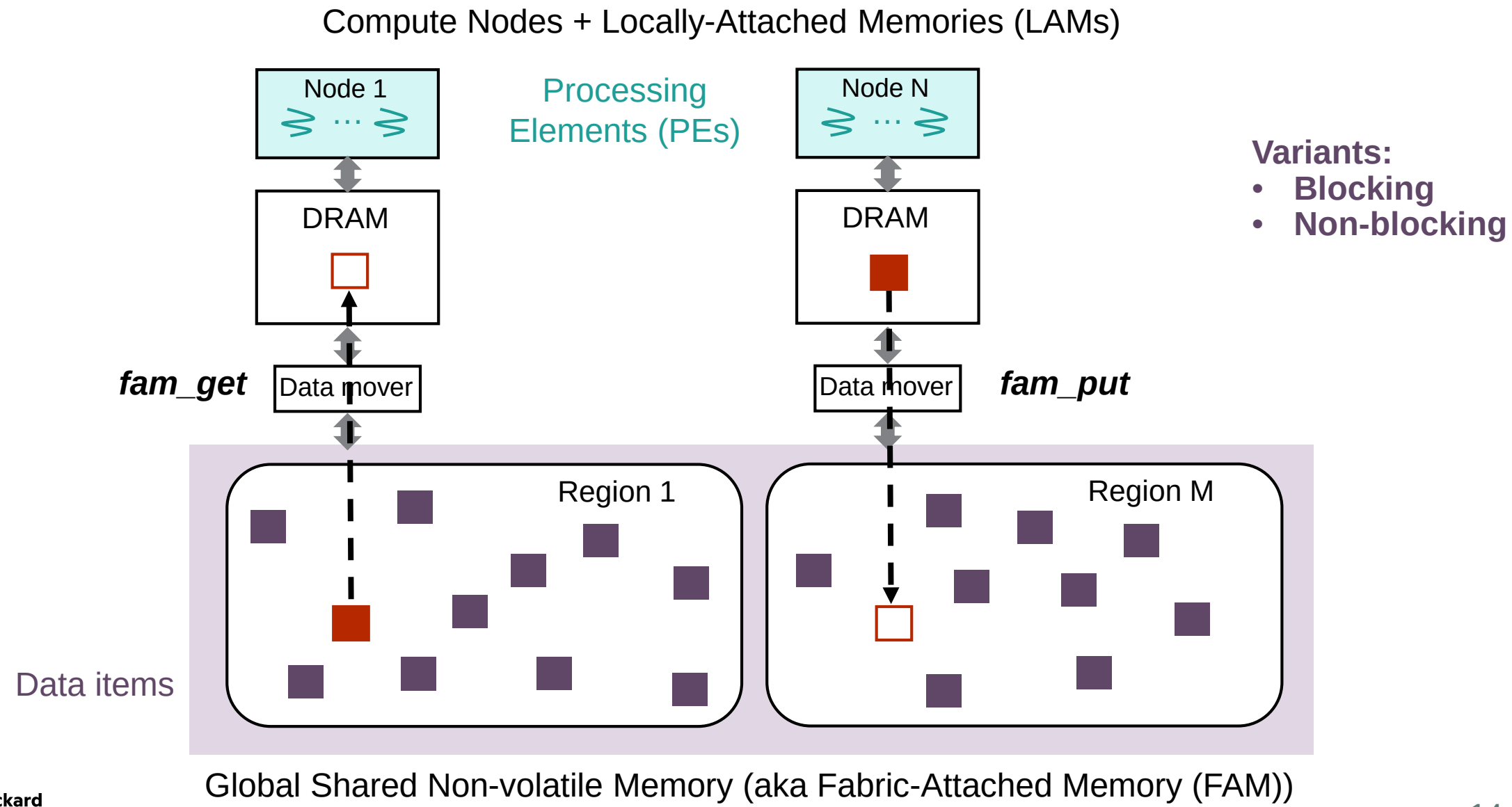
  – Triggers delayed reclamation to accommodate ongoing users

– **fam_change_permissions:** change permissions of a data item or region in FAM



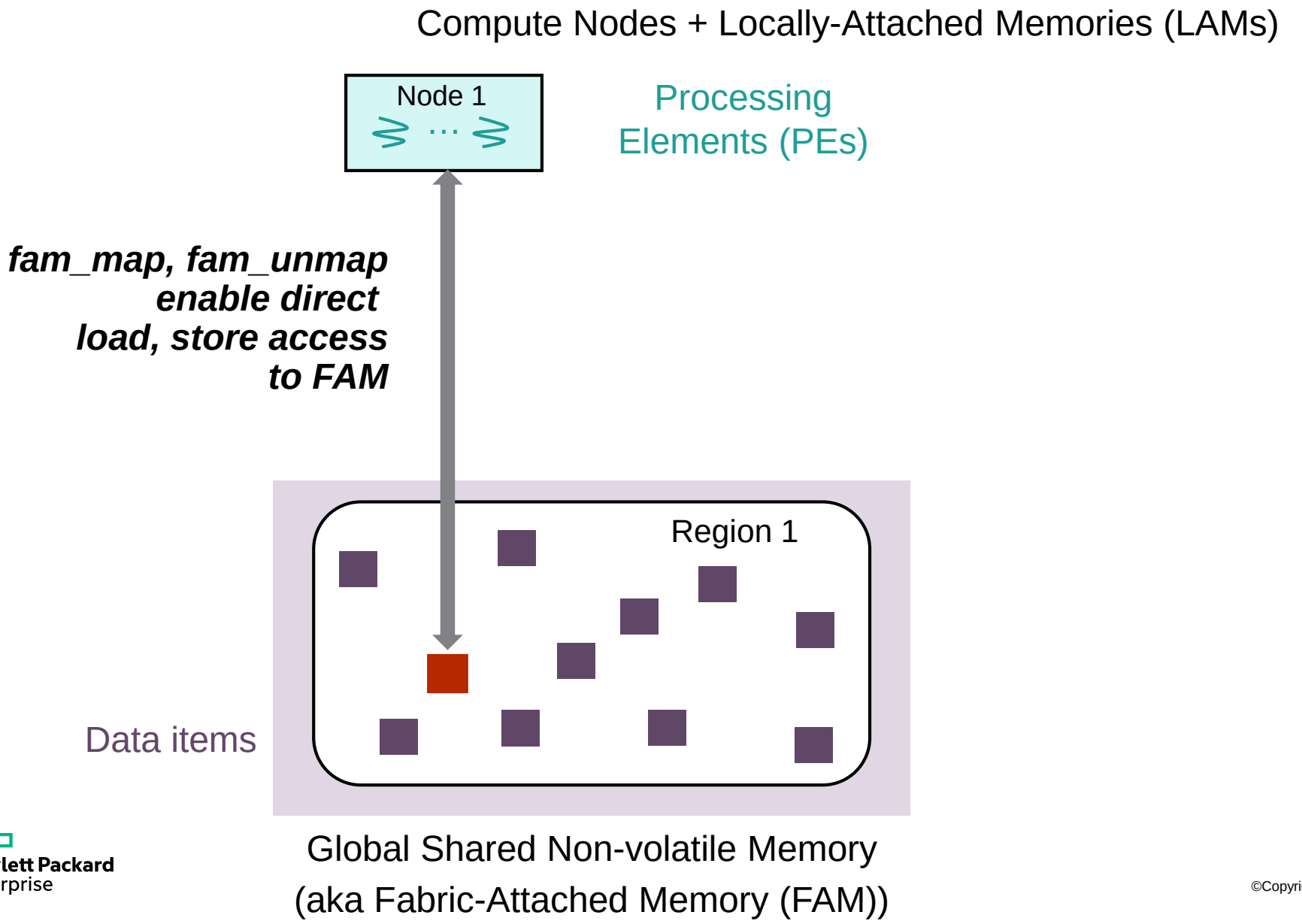
Global Shared Non-volatile Memory (aka Fabric-Attached Memory (FAM))

Data path (get / put)

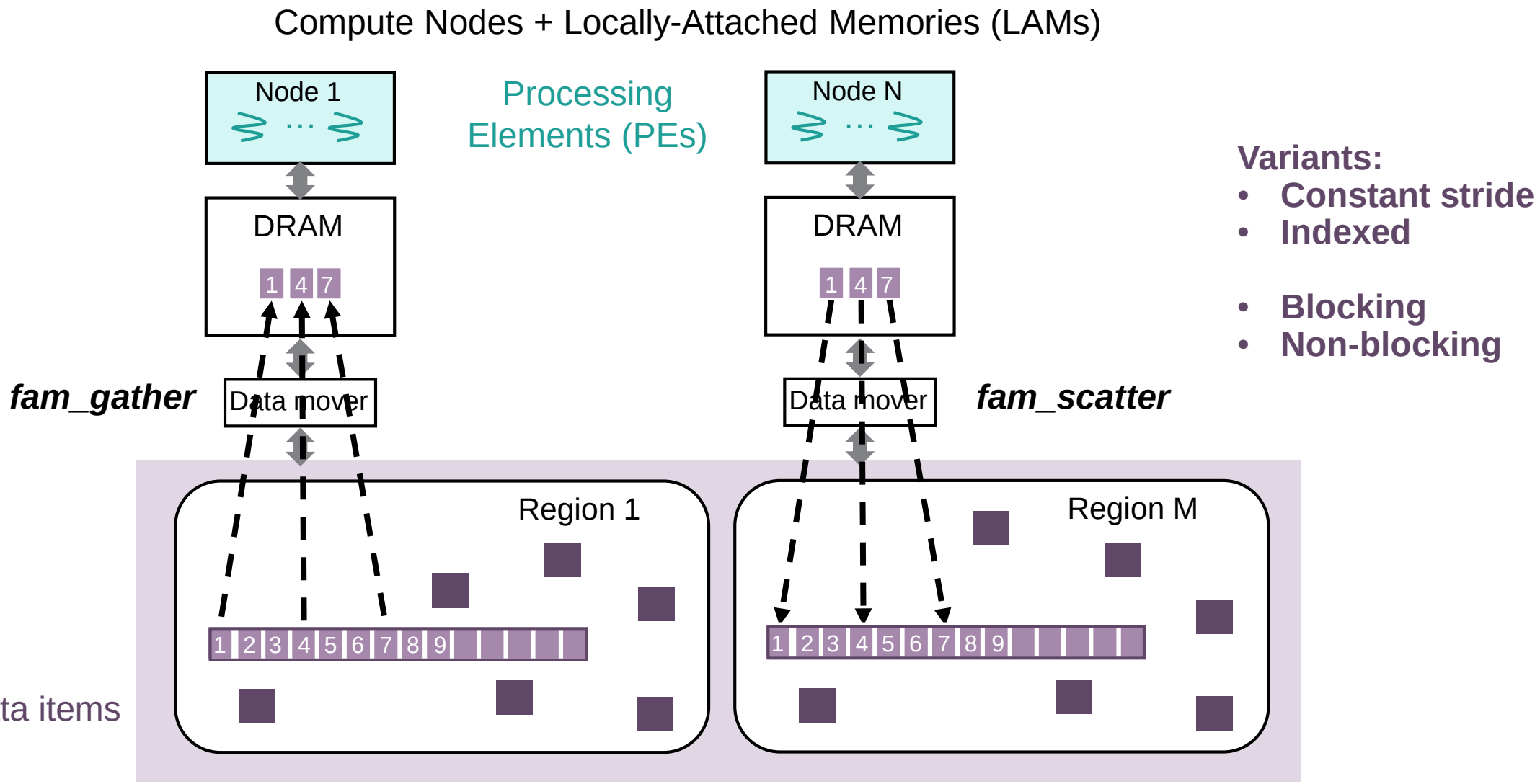


Variants:
- **Blocking**
- **Non-blocking**

Data path (direct access)



Data path (gather / scatter)



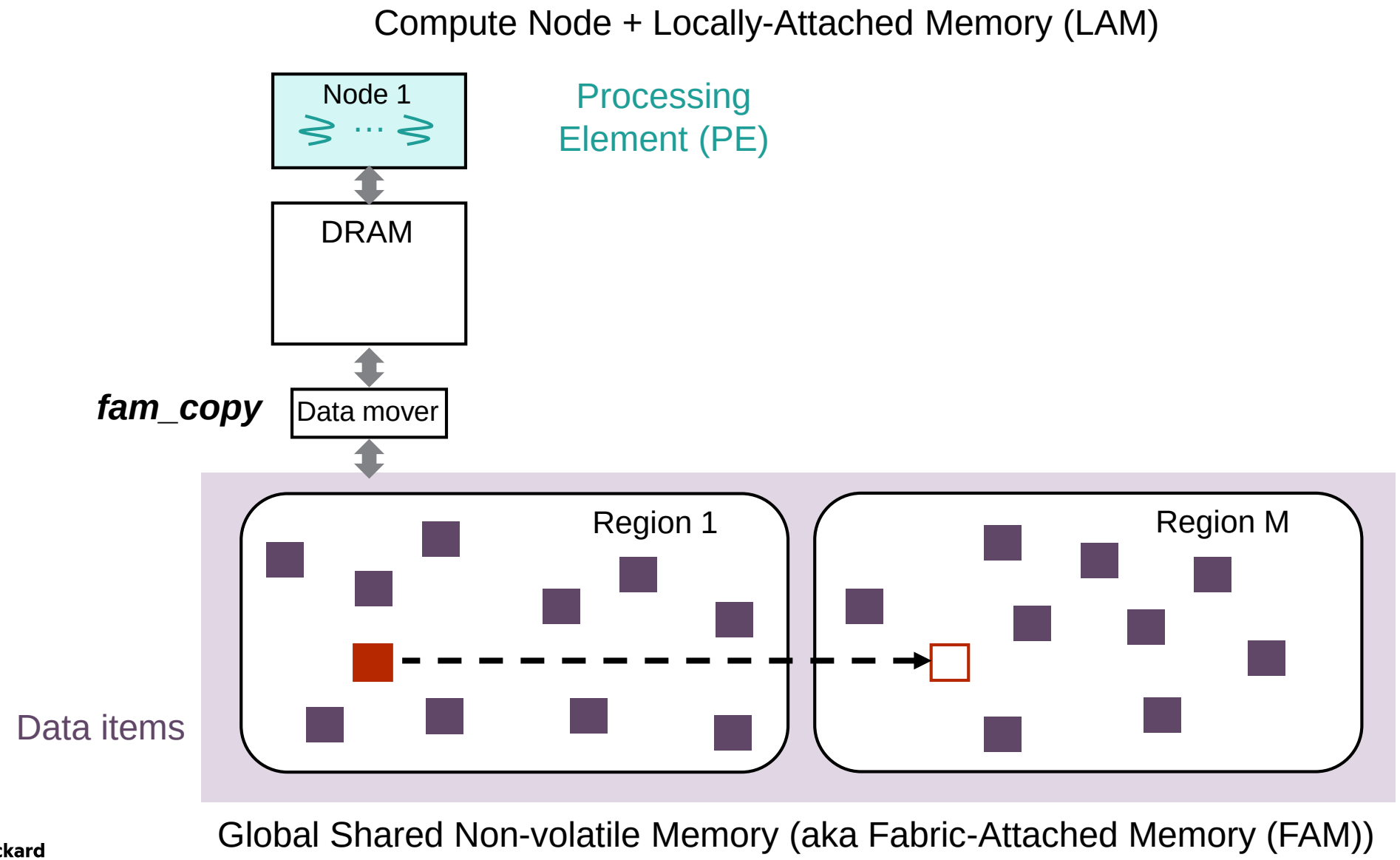
Variants:

- **Constant stride**
- **Indexed**

- **Blocking**
- **Non-blocking**

Hewlett Packard Enterprise

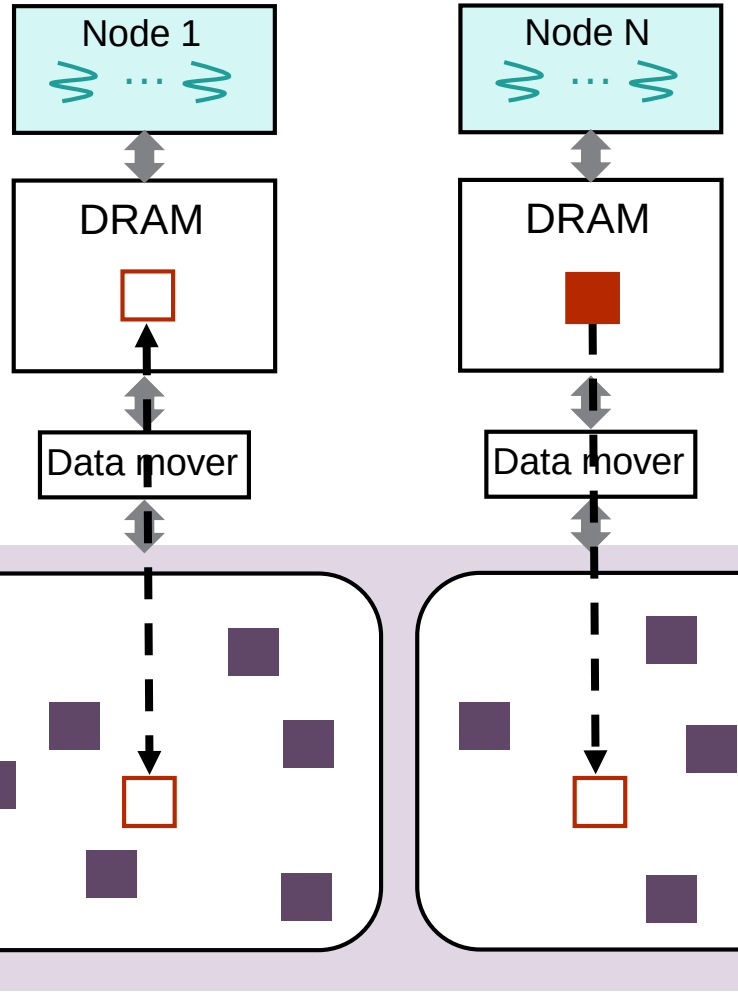
Data management (copy)



Atomics

Compute Nodes + Locally-Attached Memories (LAMs)

Processing Elements (PEs)



OpenFAM fetching routines:

Fetch, swap, add, subtract, min, max: signed and unsigned 32b and 64b integer, float, double

Compare-and-swap: signed and unsigned 32b and 64b integer, 128b integer

And, or, xor: 32b and 64b unsigned integer

OpenFAM non-fetching routines:

Set, add, subtract, min, max: signed and unsigned 32b and 64b integer, float, double

And, or, xor: 32b and 64b unsigned integer

Global Shared Non-volatile Memory (aka Fabric-Attached Memory (FAM))

Hewlett Packard Enterprise

Memory ordering (fence)

Pre-fence fabric memory operations

Post-fence fabric memory operations

Memory fence

– **fam_fence** ensures that FAM operations issued by the calling PE before the fence are completed before FAM operations issued after the fence are dispatched (non-blocking)

– **fam_quiet** waits for completion of all outstanding requests to global fabric-attached memory (puts, atomics, stores) (blocking)

Alternatives for handling OpenFAM call failures

Failure-reporting (app control)

- OpenFAM failures report their errors
  - Application can determine how to handle, based on severity of error, application logic, etc.
- Advantages:
  - Permits applications to implement their own redundancy (e.g., app-directed replication)
  - Stronger guarantees about semantics may be possible (e.g., all-or-nothing completion), albeit at greater overheads
- Disadvantages: Performance limitations
  - Attributable failures can only be delivered for blocking calls
  - All-or-nothing completion implementations must enforce stricter data update mechanisms (e.g., begin/end transaction, logging or non-in-place updates)

Fail-fast (performance)

- OpenFAM failures cause application to be killed
  - Similar to OpenSHMEM semantics
  - No guaranteed atomicity of operations; partial completions may occur
  - Applications should be structured to perform appropriate application-specific recovery if needed
- Advantages:
  - Potential for improved performance (non-blocking operations possible)
  - Guarantees better matched to direct access (load/store) mode
- Disadvantages:
  - Fabric or FAM failures may result in corrupted persistent data

Summary

– OpenFAM API is a programming model for disaggregated persistent fabric-attached memory

– Status
  – “Ported” example applications to validate OpenFAM API
    – Ex: sparse matrix-vector multiplication, large-scale graph inference, PageRank pipeline, sort, random access
  – Presented OpenFAM to OpenSHMEM steering committee
    – Interest in including ideas for disaggregated and persistent memory in OpenSHMEM 2.0 (due late 2020)

– We’re interested in your feedback
  – Draft of OpenFAM API spec available for review: https://github.com/OpenFAM/API
  – Email us at openfam@groups.ext.hpe.com