

NATIVE OS SUPPORT FOR PERSISTENT MEMORY WITH REGIONS

Mohammad Chowdhury (mchow017@fiu.edu)

Raju Rangaswami (raju@cs.fiu.edu)

Florida International University

PERSISTENT MEMORY (PM)

❖ Hybrid characteristics of memory and storage

Memory
<ul style="list-style-type: none">• Volatile• Byte-addressable access• Fast

Storage
<ul style="list-style-type: none">• Non-volatile/Persistent• Block I/O access• Slow

**Read/Write
latency:
4X-10X
of memory**



Persistent Memory
<ul style="list-style-type: none">• Non-Volatile/Persistent• Byte-addressable access• Fast

PM CHALLENGES

- ❖ PM is directly accessible by CPU

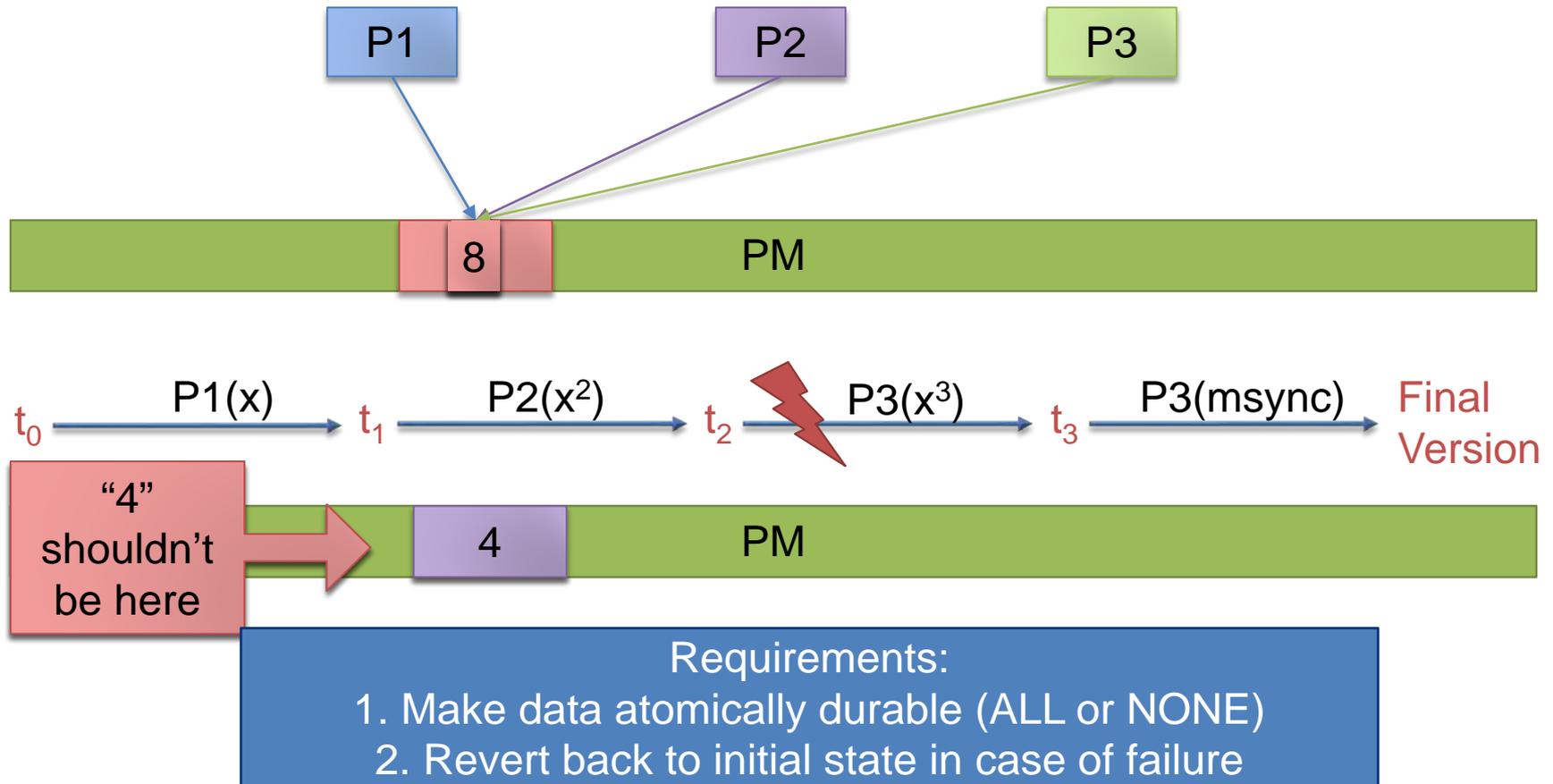
- ❖ **BUT ...**

PM resident data can be corrupted after a system failure if ordering of updates is violated

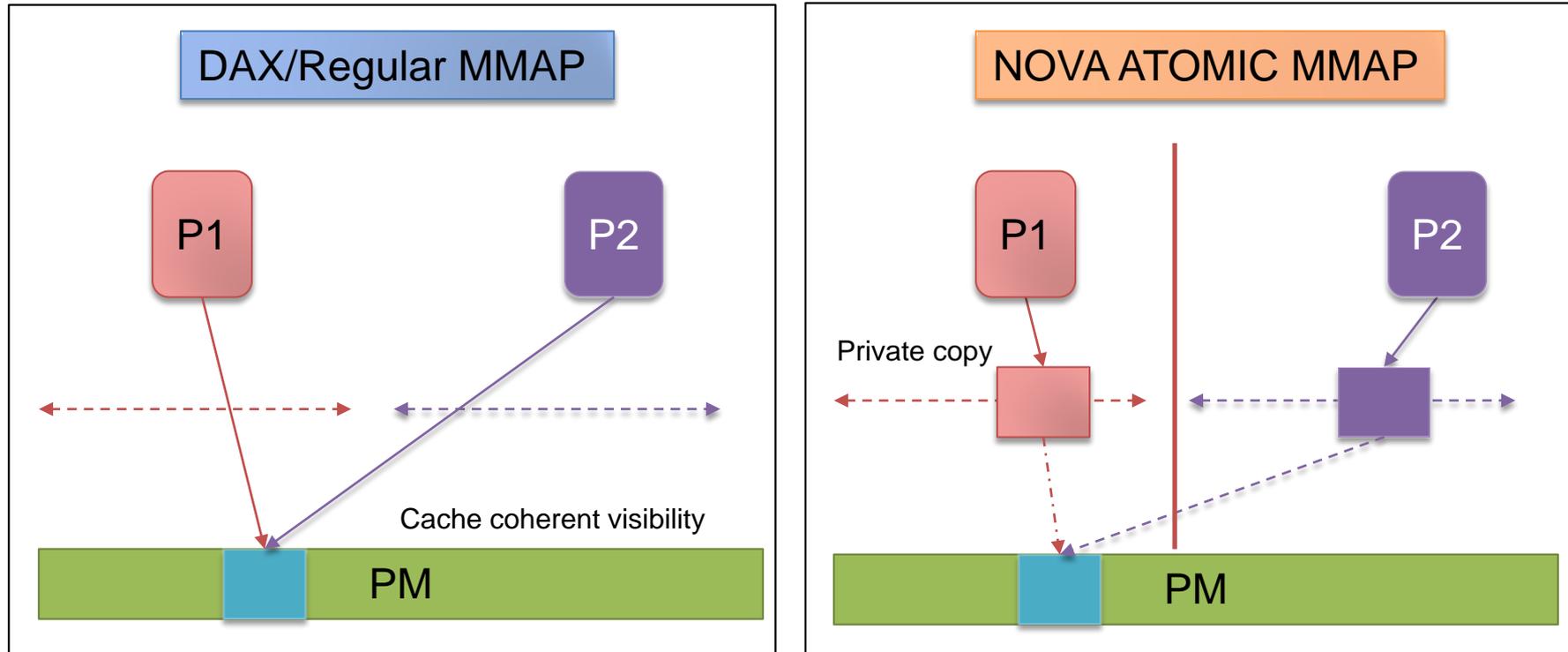
PM CHALLENGES: THE COSTS OF ORDERING

- Ordering requires cache line flushes, barriers, and ADR (asynchronous DRAM refresh)
 - Increased cost of operations
- More redundant metadata → More ordering required
- GOAL →
 - Reduce ordering requirements

PM CHALLENGES: ATOMIC DATA DURABILITY



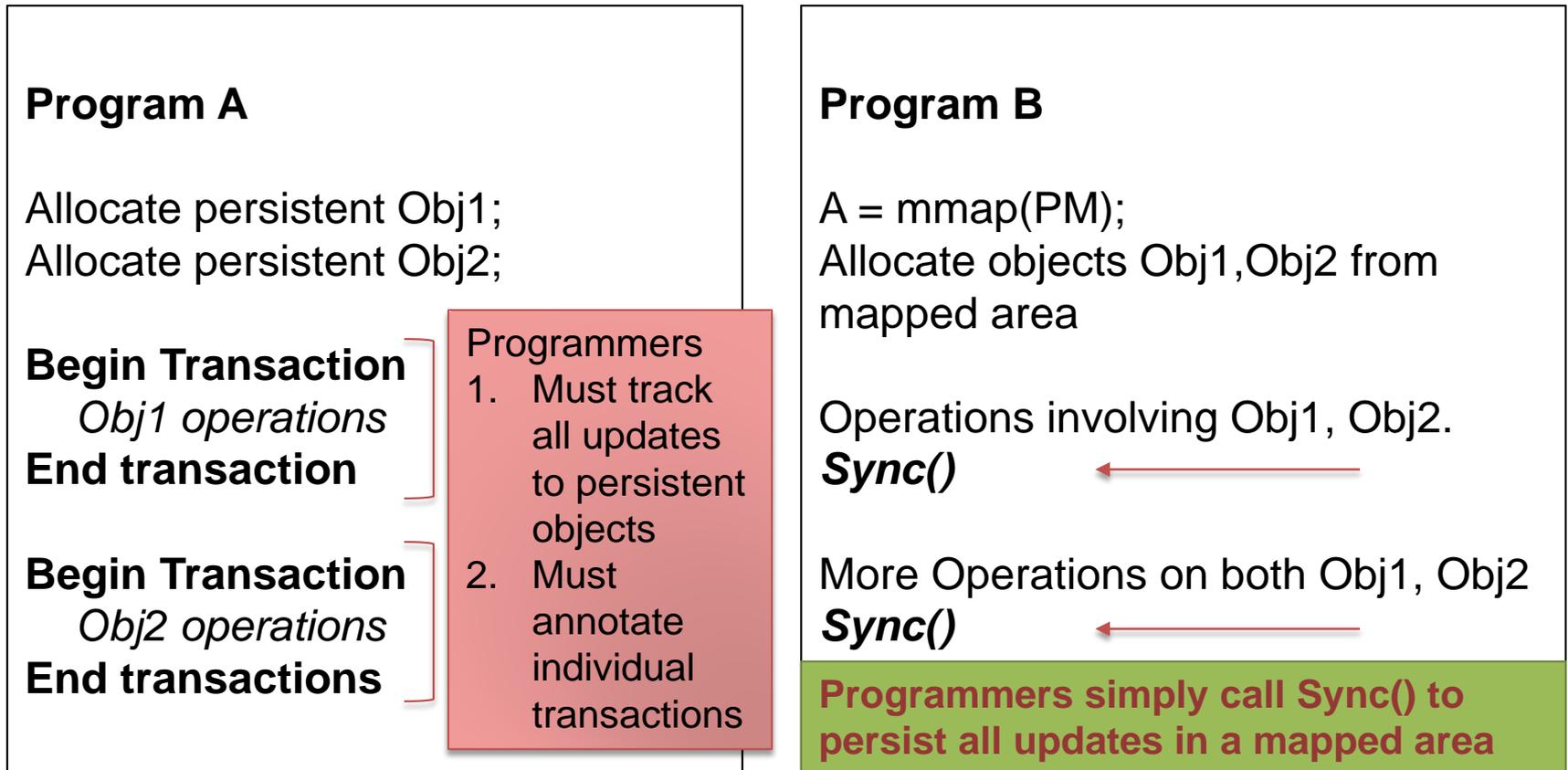
PM OPPORTUNITIES: SHARED CONSISTENCY



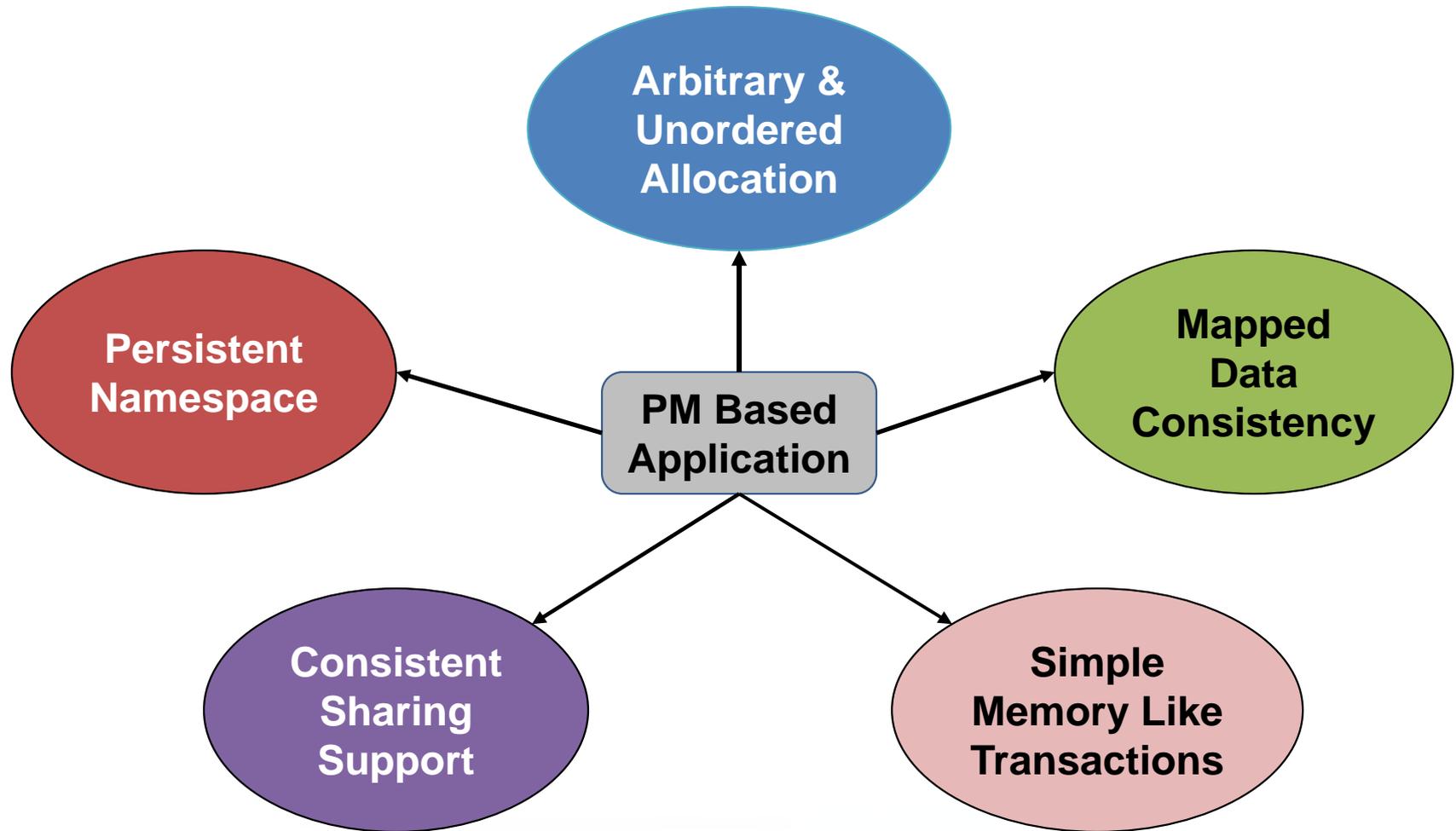
Requirements:

1. Updates should be visible to all the shared processes
2. Should support atomic durability of all updates across a shared region

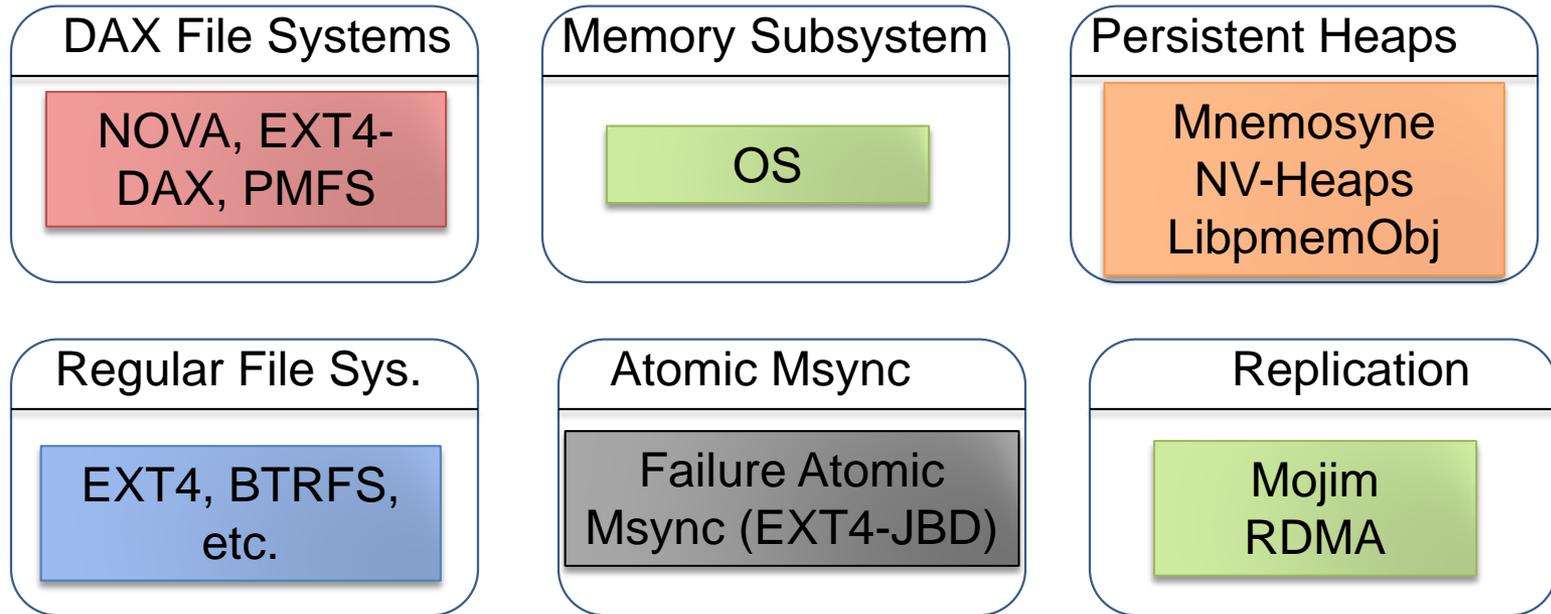
PM OPPORTUNITIES: SIMPLE MEMORY-LIKE TRANSACTIONS



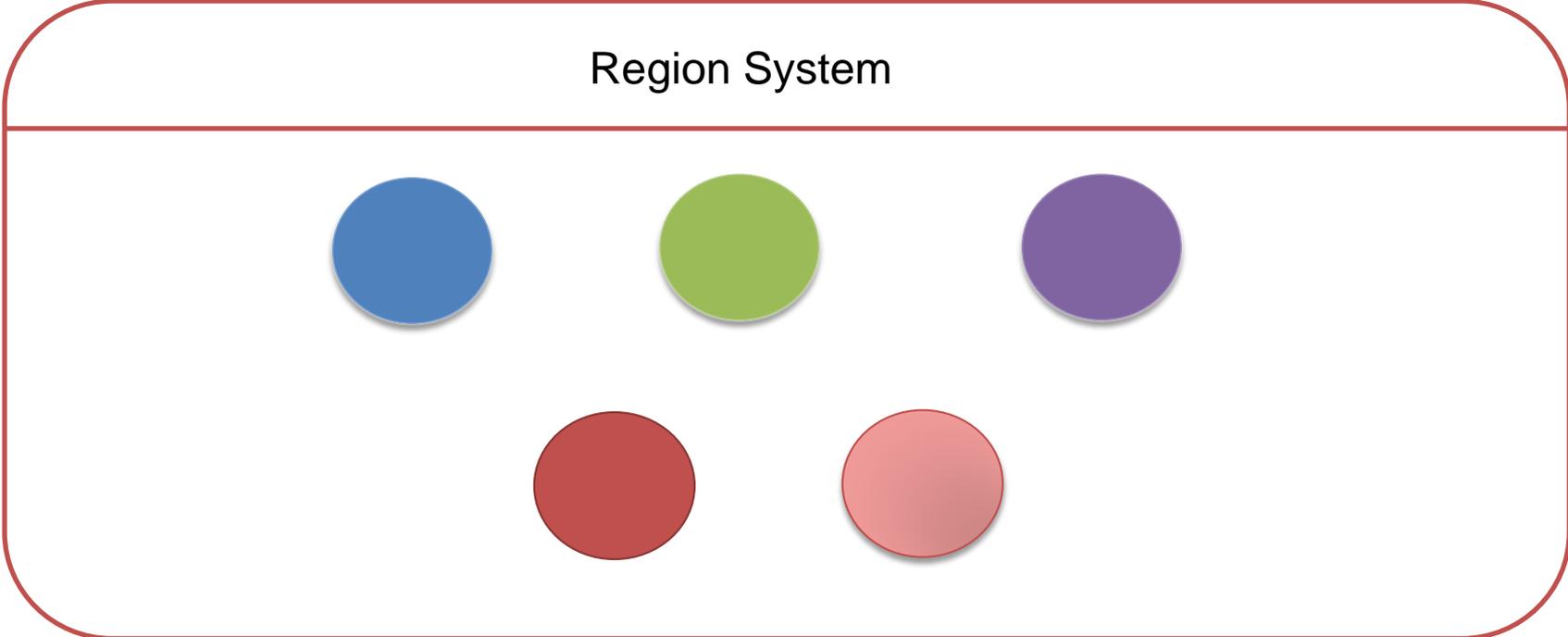
APPLICATIONS REQUIREMENTS FOR PM



CONTEMPORARY SOLUTIONS



CONTEMPORARY SOLUTIONS



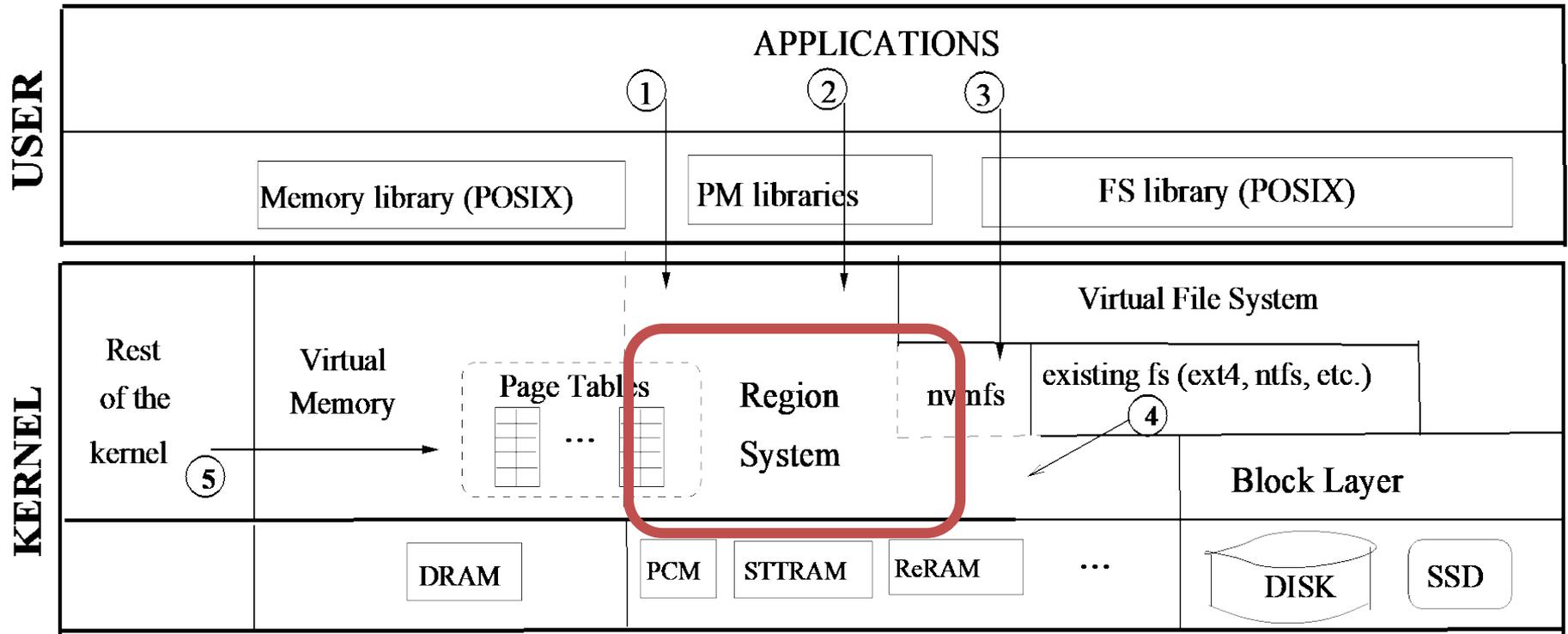
-  **Arbitrary and Unordered Allocation**
-  **Consistent Sharing Support**
-  **Simple Memory Like Transactions**
-  **Mapped Data Consistency**
-  **Persistent Namespace**
-  **Mapped Data Consistency (Partial)**

REGION SYSTEM

We present “Region System”, a kernel subsystem, to support persistent memory to achieve the following goals:

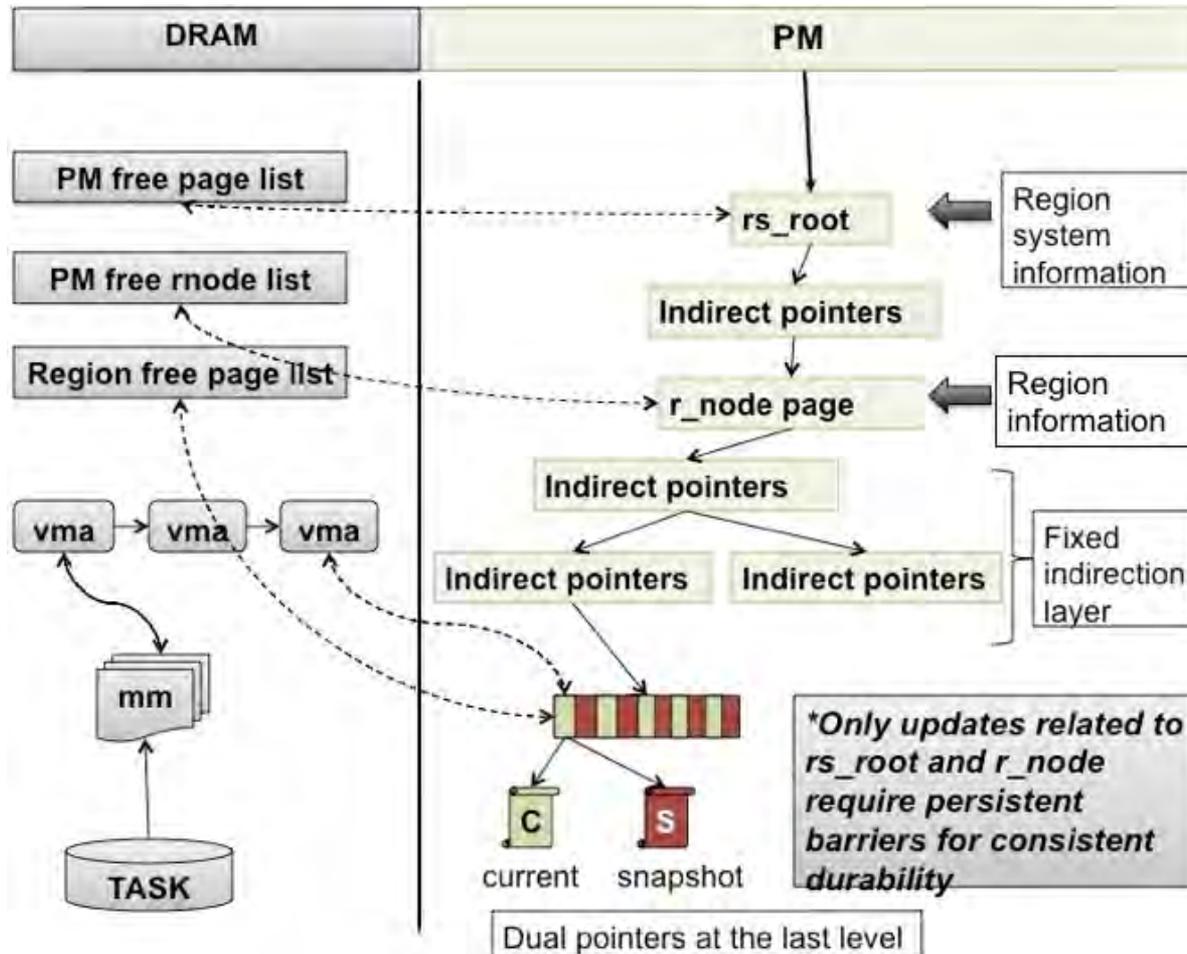
- Minimize unwanted latency in the persistent memory access path;
- Provide users with direct and consistent access to shared persistent memory; and
- Demonstrate modifications of the existing applications for optimized usage.

REDEFINED OS MEMORY/STORAGE STACK



NOT intended as replacement for **File Systems** or **Memory Subsystem**
 RS should serve as a core “Persistent Memory Support System” usable
 by **applications**, **file systems**, and **other kernel subsystems**.

ARCHITECTURE



Region:
Collection of persistent pages

PPAGES: 4KB
PM pages

CONSISTENCY STATES

Current	Snapshot	State
0	0	No Ppage
0	y	Invalid – There can not be a snapshot without current
x	0	Un-synced page, mapped to the address space
x	y	x == y, page in synced state
		x != y, page in unsynced state, “y” is the consistent version

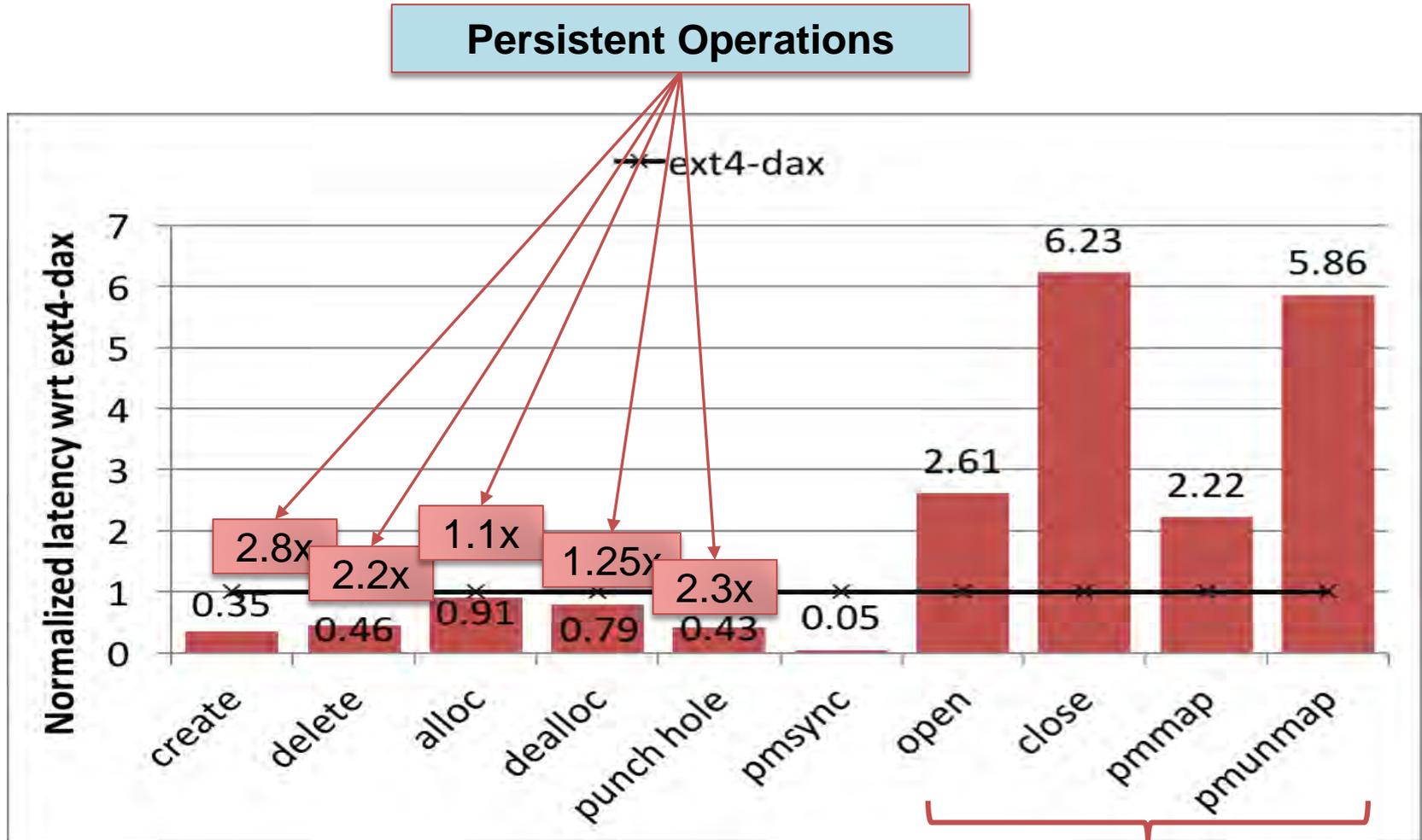
REGION SYSTEM (RS) INTERFACE

Class	System Call
Namespace	region_d open (char region_name, flags f)
	int close (region_d rd)
	int delete (region_d rd)
Allocation	ppage_no alloc_ppage (region_d rd)
	int free_ppage (region_d rd, ppage_no ppn)
Mapping & Consistency	vaddr pmmmap(vaddr va, region_d rd, ppage_no, int nbytes, flags f)
	int pmunmap(vaddr va)
	pmsync(vaddr va)

METADATA OPERATIONS

- Persistent Operations
 - Modifies persistent metadata
- Volatile Operations
 - No updates to persistent metadata
- Persistent operations are designed to achieve atomic durability

METADATA OPERATION COMPARISON



Volatile Operations

MAPPED DATA CONSISTENCY CHALLENGES

- **Avoid Unwanted Durability**

- Applications want to make updates durable only updates a `msync()` invocation.
- Updates are made durable in PM before a `msync` call.
- In case of a failure, the mapped PM area will contain uncommitted data.

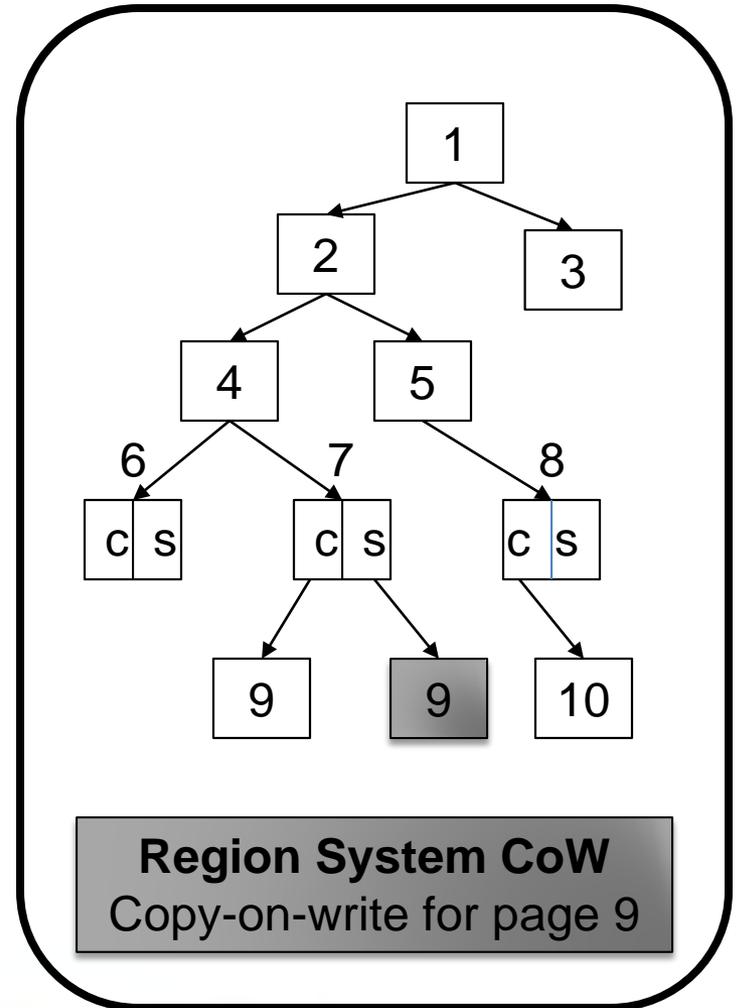
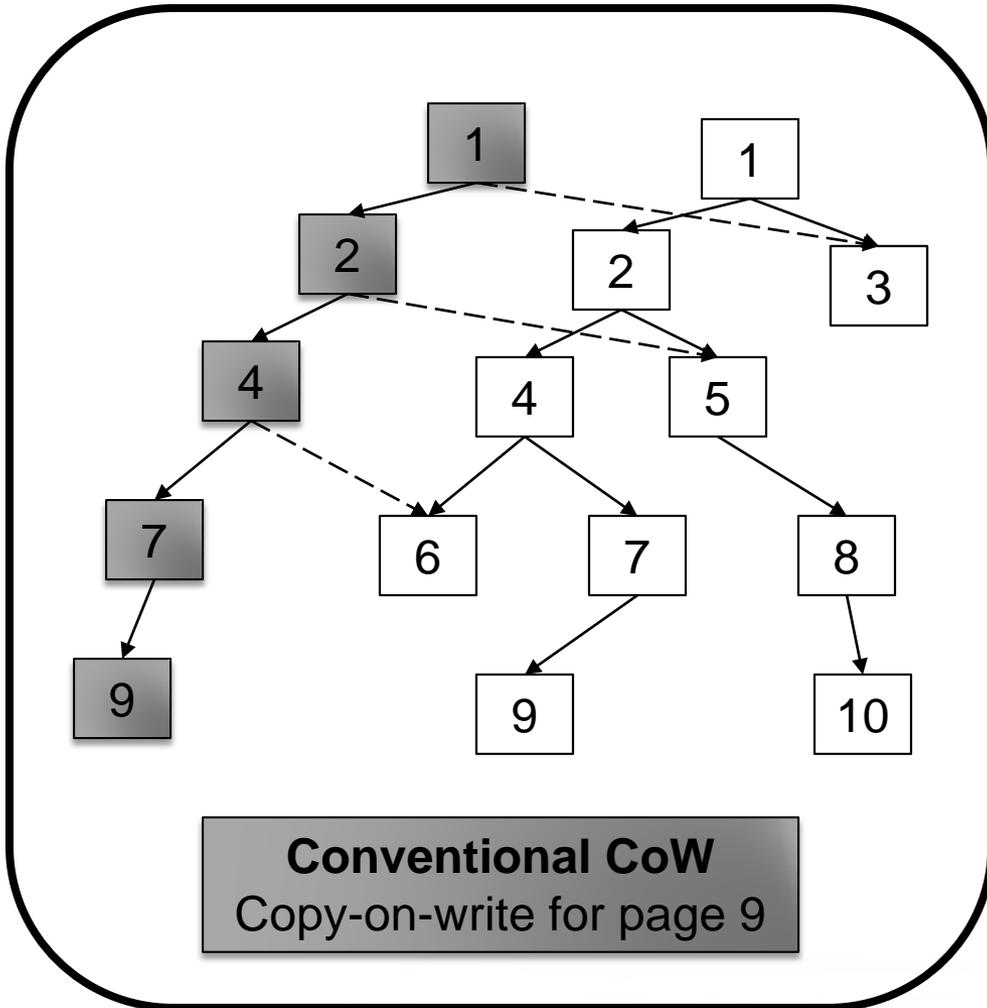
- **Protecting the Sync**

- During sync operation no applications should be allowed to write to mapped PM → difficult to achieve due to direct CPU access.

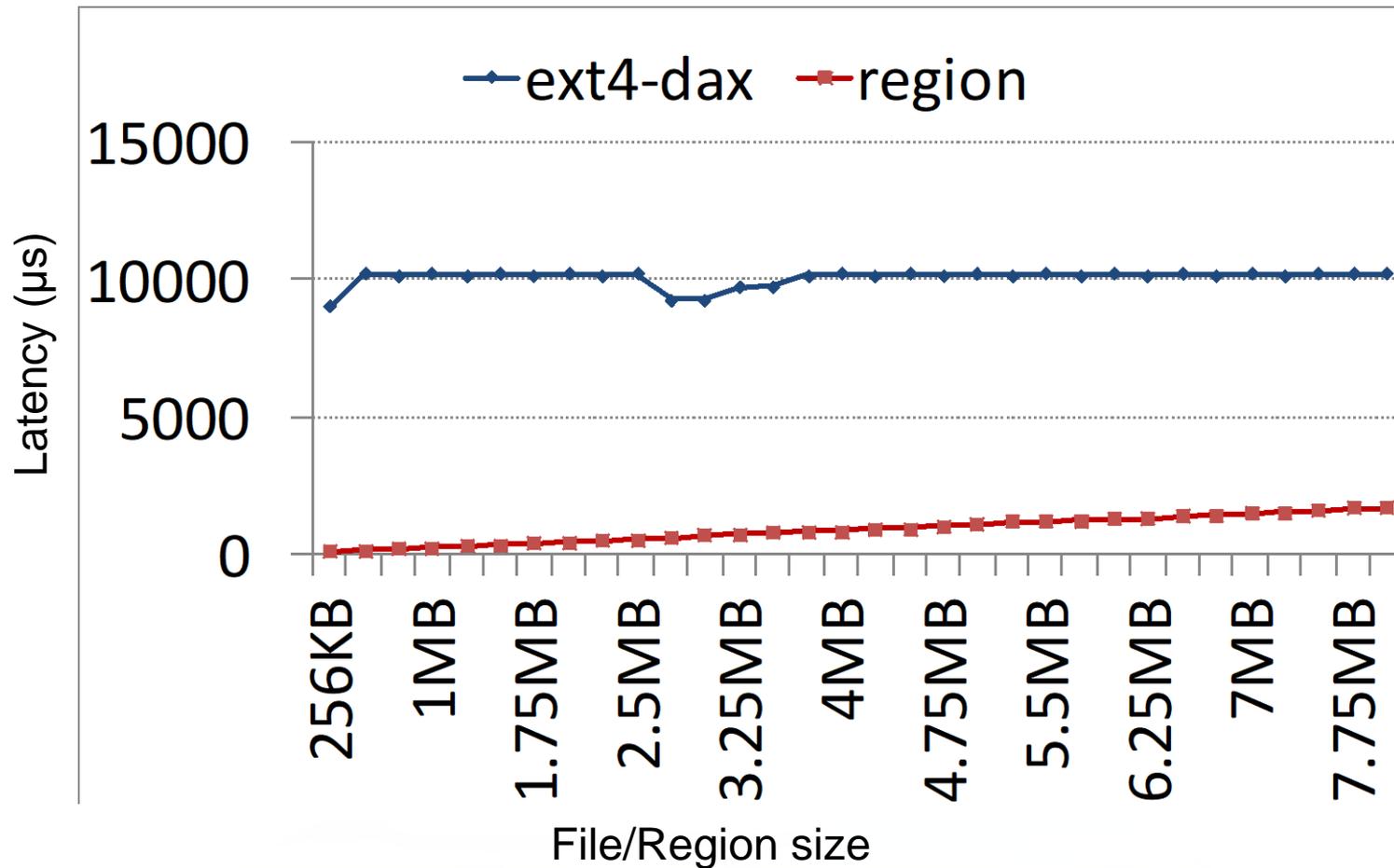
ATOMIC DURABILITY WITH PMSYNC

1. Identify the dirty pages
2. Write protect the pages
3. Flush dirty cache lines
4. Copy-on-write protection for future writes to a sync'ed page

AVOIDING COW PROPAGATION



PMSYNC COMPARISON WITH EXT4-DAX

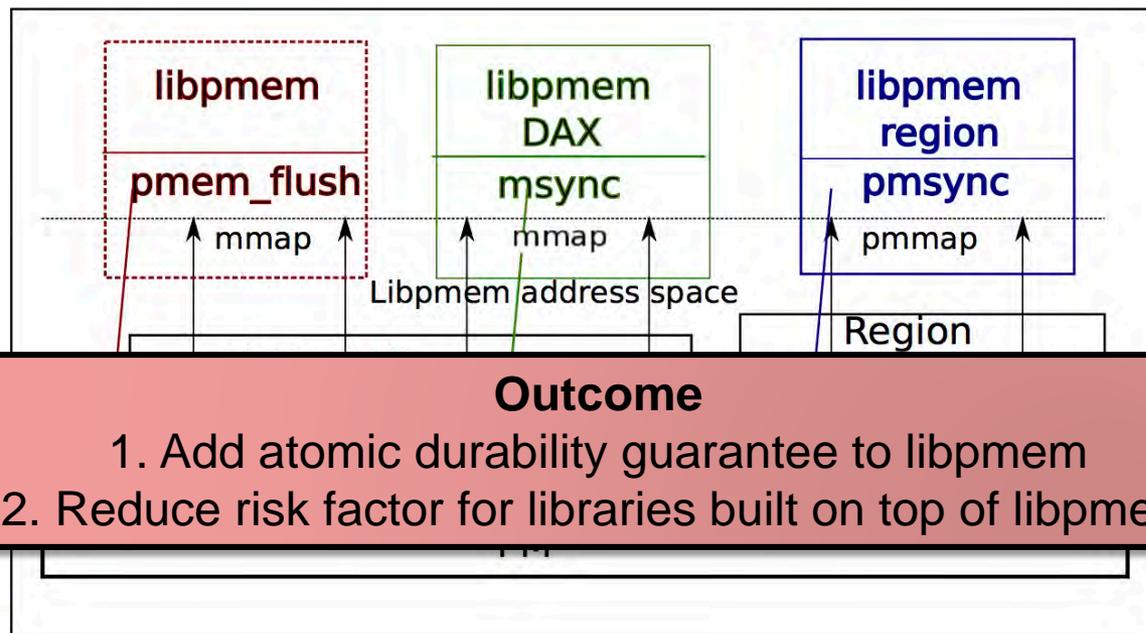


LIBPMEM-REGION

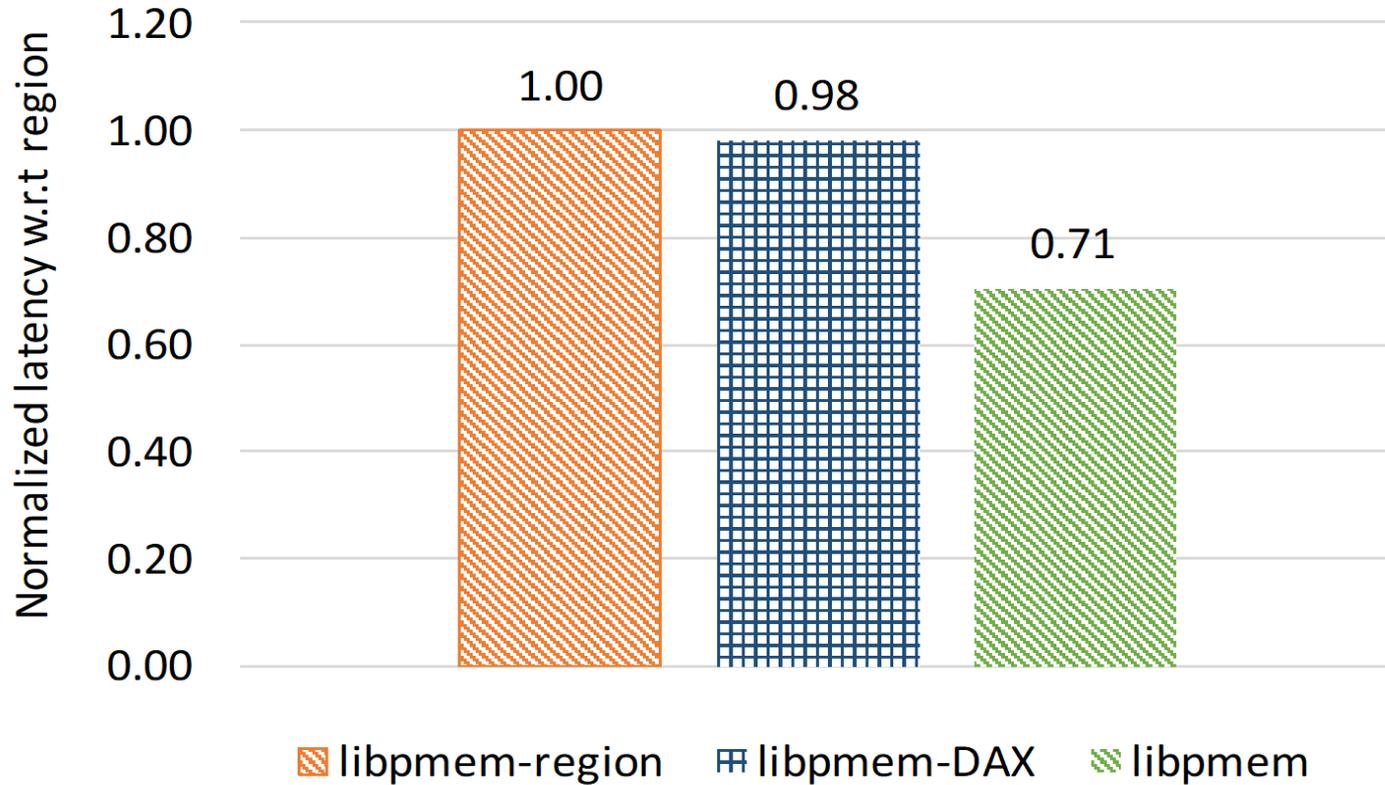
Non-transactional pmem-flush

All or None policy does not work

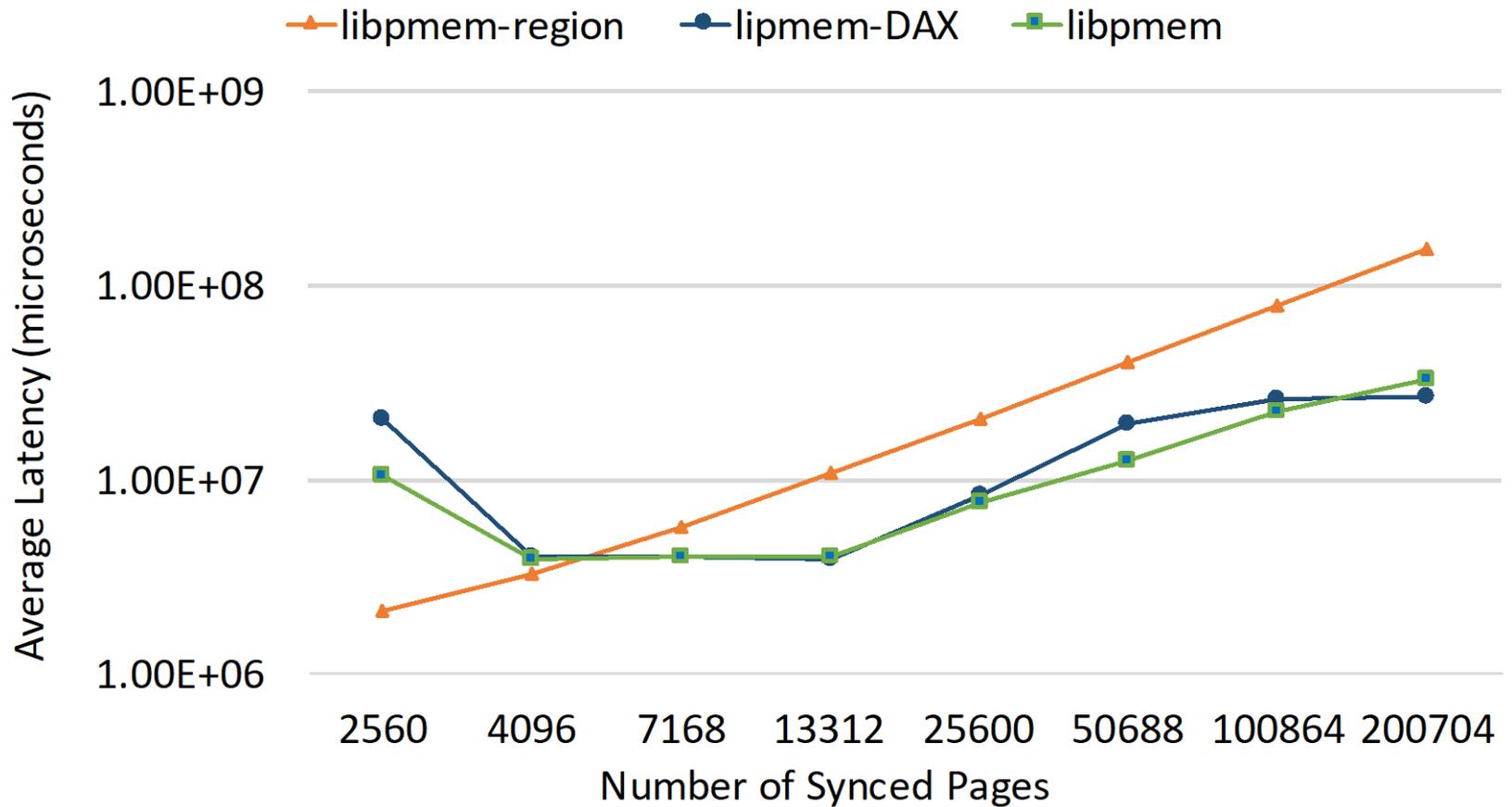
A portion of the updates can be lost



LIBPMEM COMPARISONS



LIBPMEM COMPARISONS



SUMMARY

- Region System Features
 - Provides **arbitrary** and **unordered** allocation and deallocation
 - Minimizes ordering requirements by **eliminating redundancy**
 - Provides **transparent sharing** and **atomic durability** of mapped data with competitive performance
 - Usable by **File systems, Applications, Libraries**, and other kernel subsystems or modules.
 - Source code will be made public soon!

Thanks!

QUESTIONS?