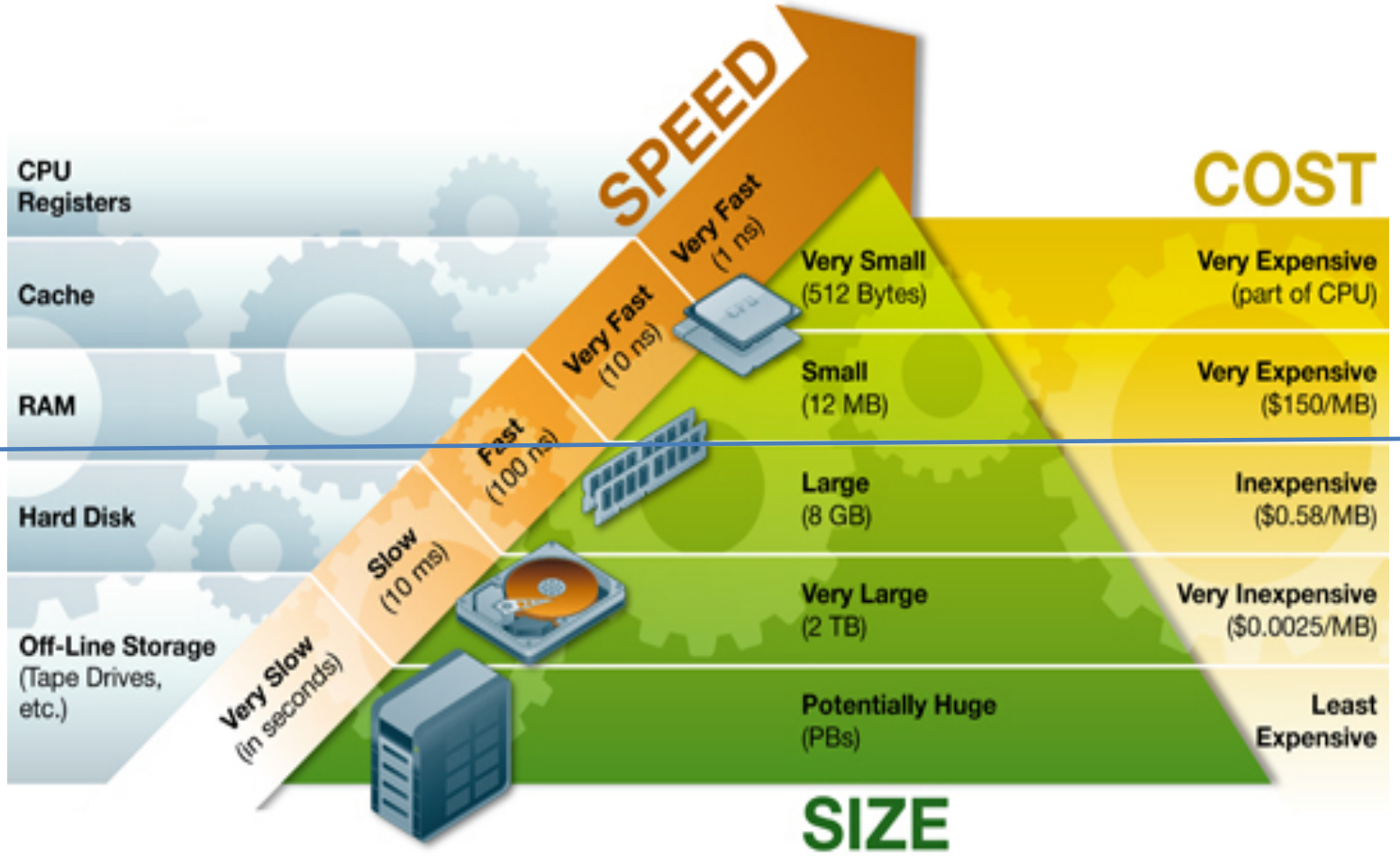
Non-volatile Memory

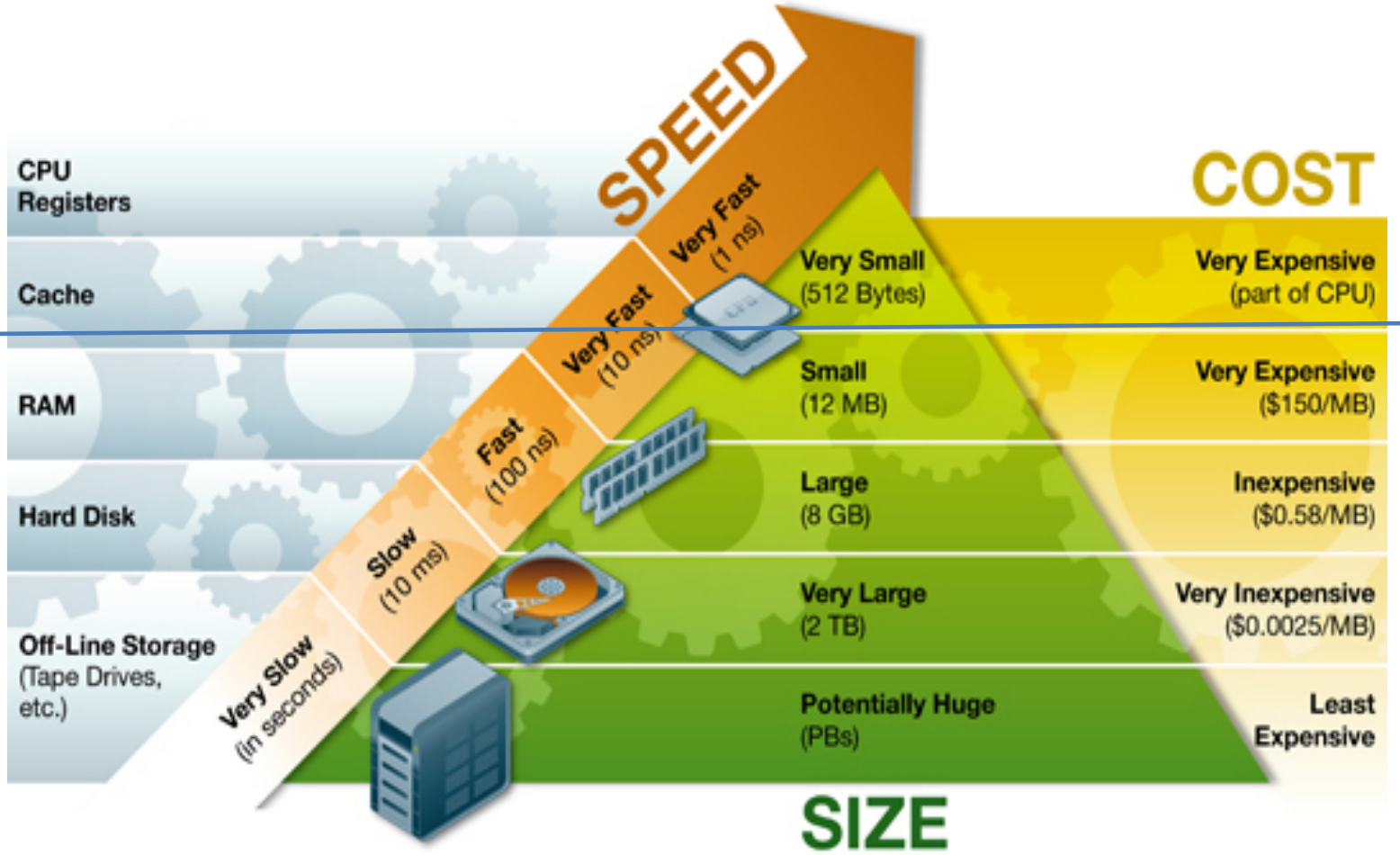
Rong Chen

ACK: Pcommit slides adapted from http://research.cs.wisc.edu/sonar/tutorial/03-hardware.pdf
PMFS Slides adapted from Makoto Shimazu@Reading Circle

Memory Hierarchy



NVM Memory Hierarchy



Volatile

Non-volatile

A New World of Storage

DRAM

+ Fast
+ Byte-addressable
- Volatile

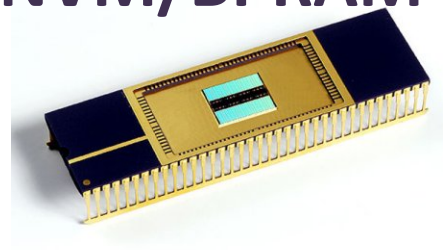
Disk / Flash

+ Non-volatile
- Slow
- Block-addressable

A New World of Storage

Byte-addressable, Persistent RAM

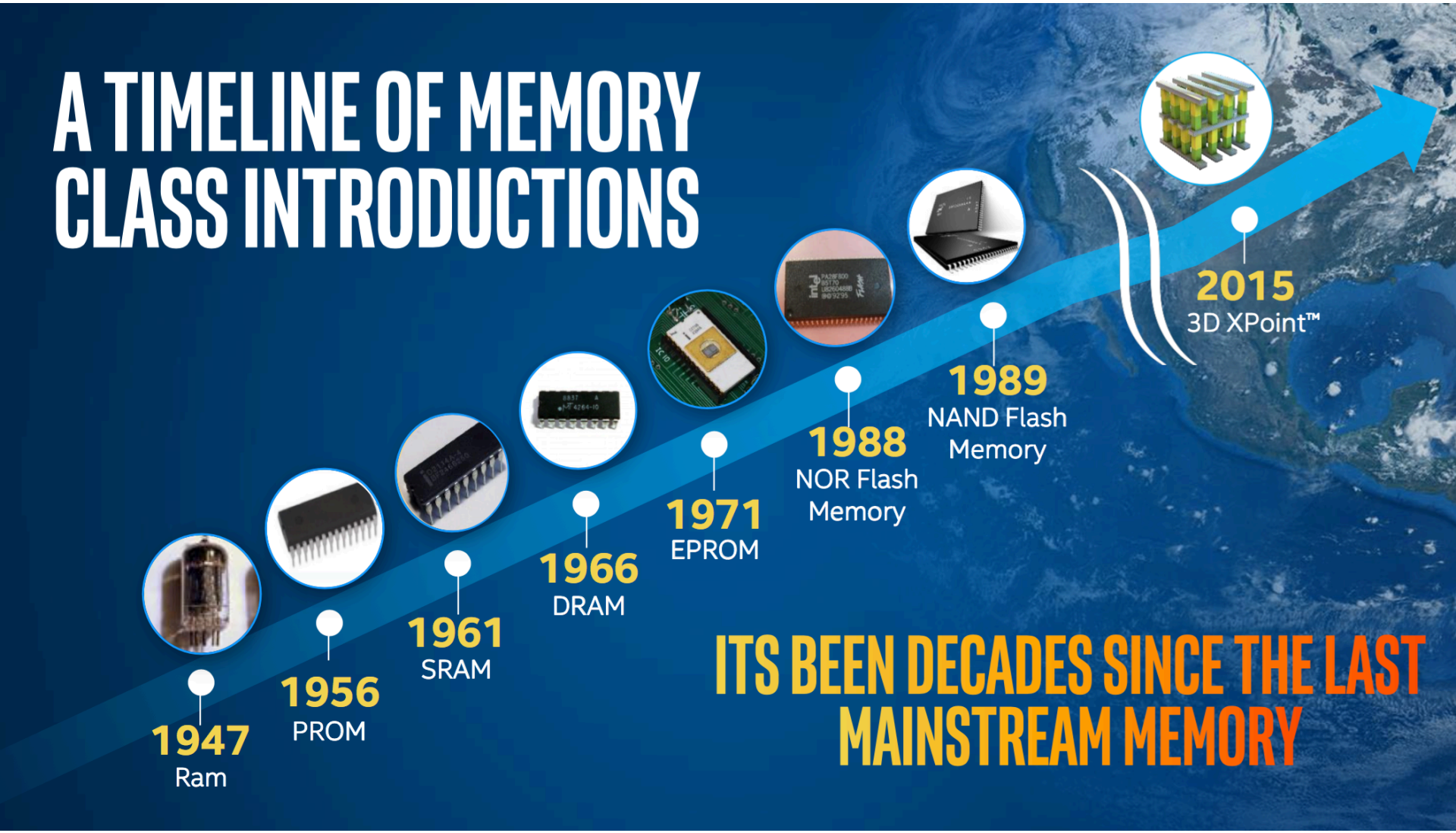
NVM/BPRAM



+ Fast
+ Byte-addressable
+ Non-volatile

EVOLUTION OF NON-VOLATILE MEMORY

Evolution of Memory



Overview of Non-volatile Memory

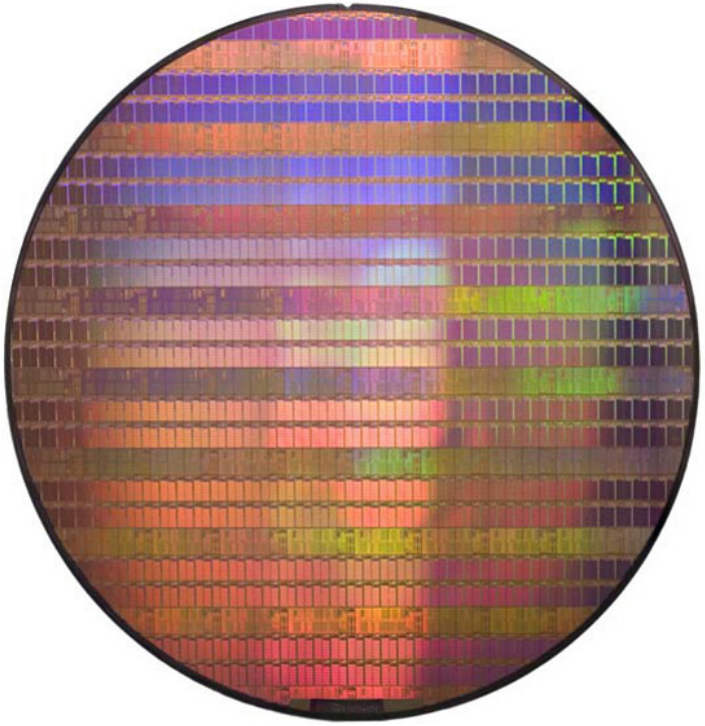
Type of Memories

	Emerging Memories				Established Memories	
	FeRAM *(or FRAM)*	MRAM	ReRAM *(or RRAM)*	PCRAM *(or PRAM, PCM)*	DRAM	Flash NAND
Nonvolatile	YES	YES	YES	YES	NO	YES
Endurance	High (10^{12})	High (10^{15})	Medium (10^{8})	Medium (10^{8})	High (10^{15})	Low (10^{5})
2012 latest technological node produced (nm)	130 nm	130 nm	R&D	45 nm	30 nm	20 nm
Cell Size (cell size in F^2)	Large (15-20)	Large/Medium (6-40)	Medium (6-12)	Medium (6-12)	Small (6-10)	Very small (4)
Write speed	Medium (100ns)	High (10 ns)	Medium (75 ns)	Medium (75 ns)	High (10ns)	Low (10 000 ns)
Power Consumption	Low	High/Low	Low	Low	Low	Very High
Cost ($/Gb)	High ($ 10 000/Gb)	High ($ 1000 – 100 /Gb)	R&D	Medium (few $/ Gb)	Low ($1/Gb)	Very Low ($ 0.1/Gb)

Phase Change Memory

Used to consider as a promising form of BPRAM

“Melting memory chips in mass production”
– *Nature*, 9/25/09

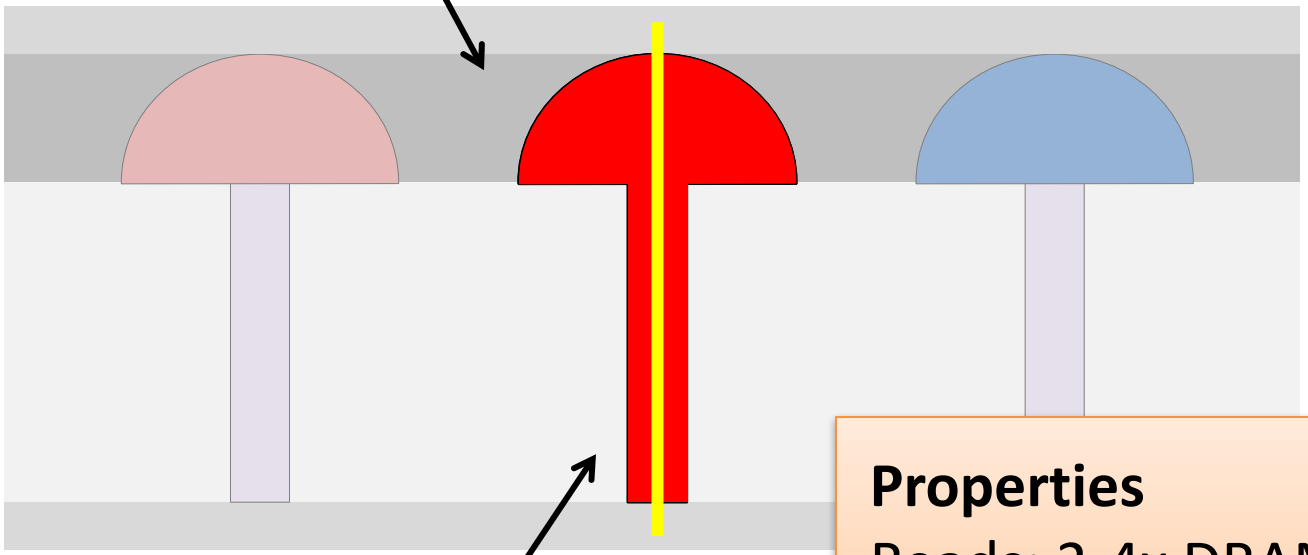


Phase Change Memory

slow cooling -> crystalline state (1)

fast cooling -> amorphous state (0)

phase change material
(chalcogenide)



electrode

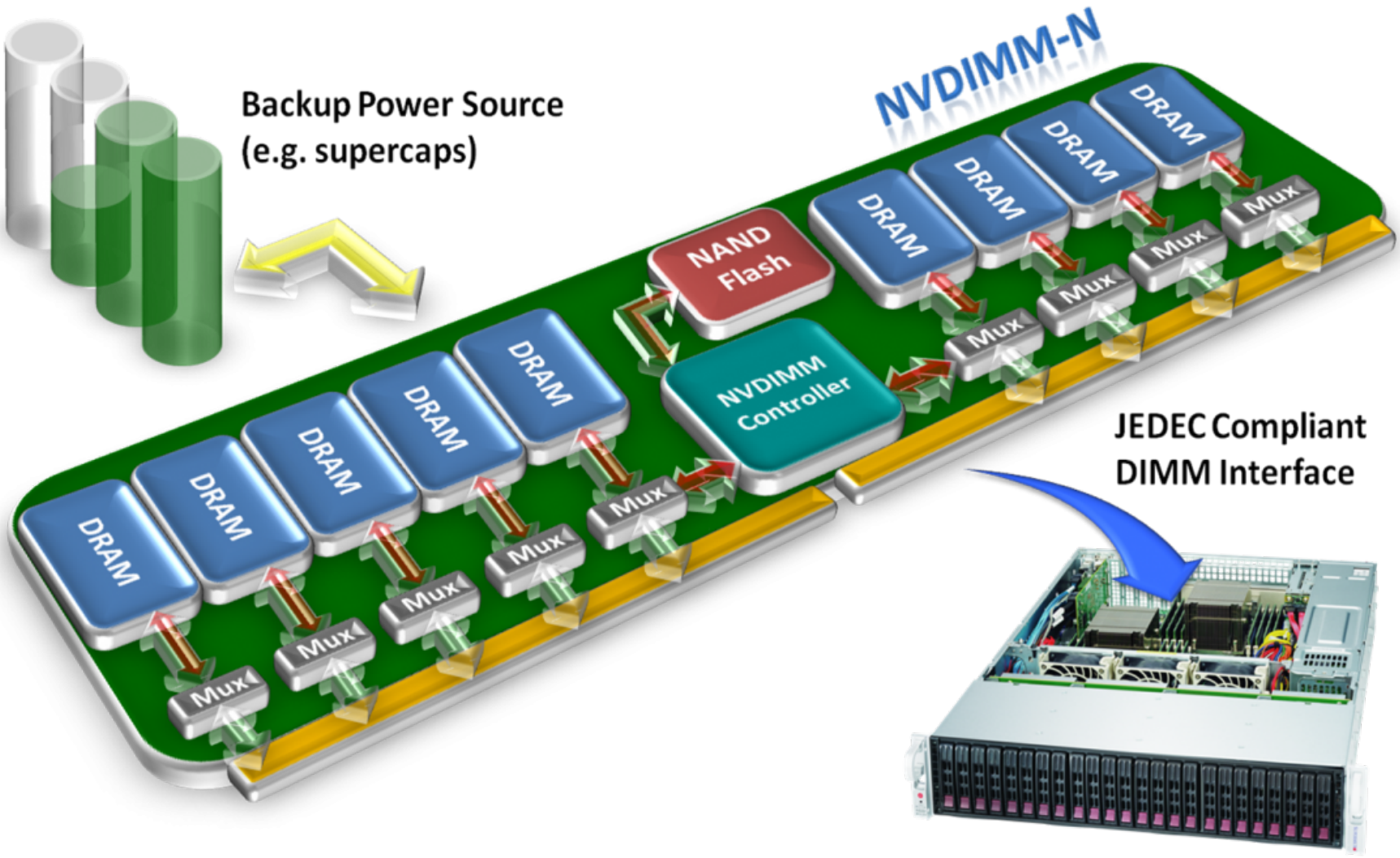
Properties

Reads: 2-4x DRAM

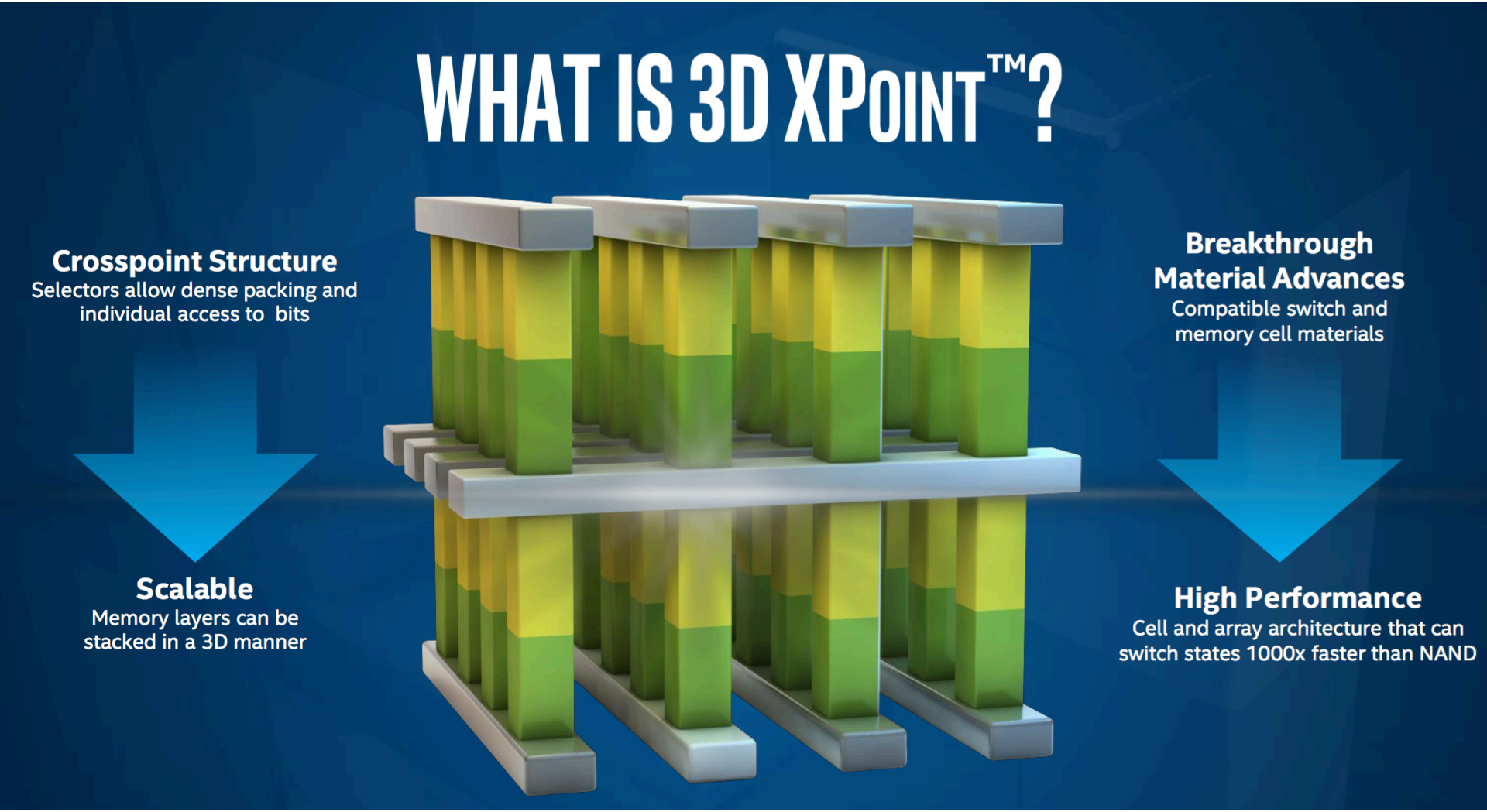
Writes: 5-10x DRAM

Endurance: 10^8+

NVDIMM

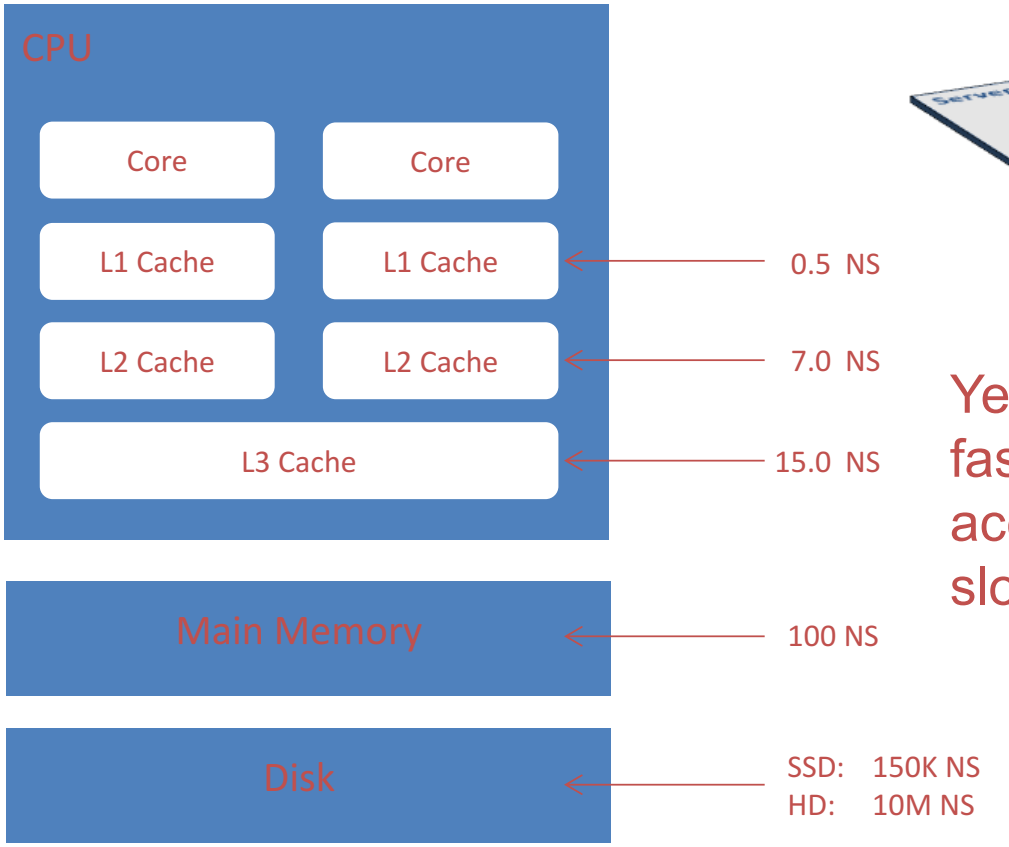

NVDIMM-N
Backup Power Source
(e.g. supercaps)
NAND Flash
NVDIMM Controller
DRAM
Mux
JEDEC Compliant
DIMM Interface

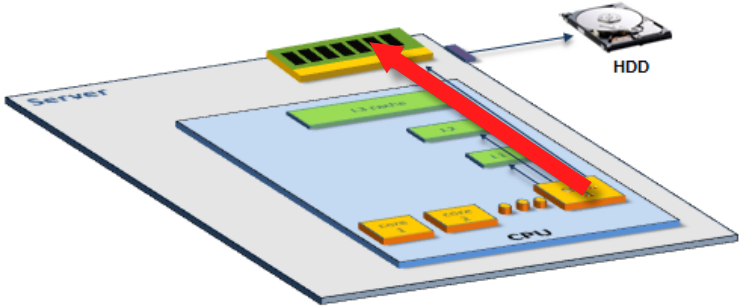
Intel 3D XPoint



Implications on Software

In-Memory Computing





Yes, DRAM is 100,000 times faster than disk, but DRAM access is still 6-200 times slower than on-chip caches

Source: SAP

Tape is Dead
Disk is Tape
Flash is Disk
RAM Locality is King

Jim Gray
Microsoft
December 2006

In memory of Jim Gray

Tape is Dead

Disk is Tape

Flash is Disk

RAM is Flash?

Cache Locality/Pallelism is King?

Example: 12306

余票查询系统:

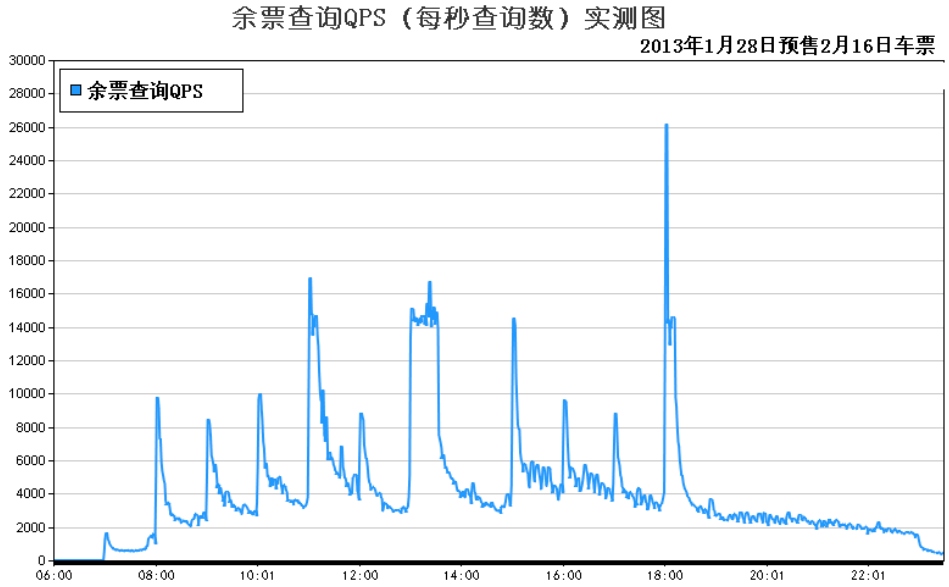
10几台X86服务器v s.以前数十台小型机单次查询的最长时间:15秒->0.2秒以下。

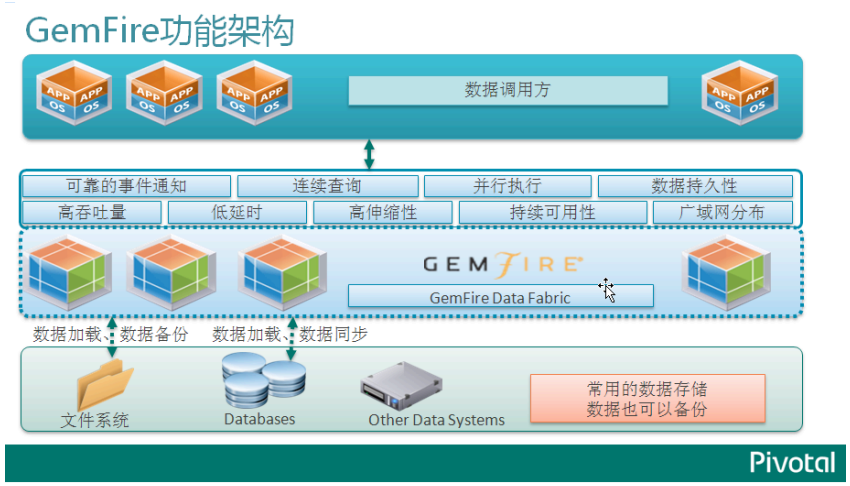
2012年春运: 支持每秒上万次的并发查询，高峰期间达到2.6万QPS吞吐量

订单查询系统改造:

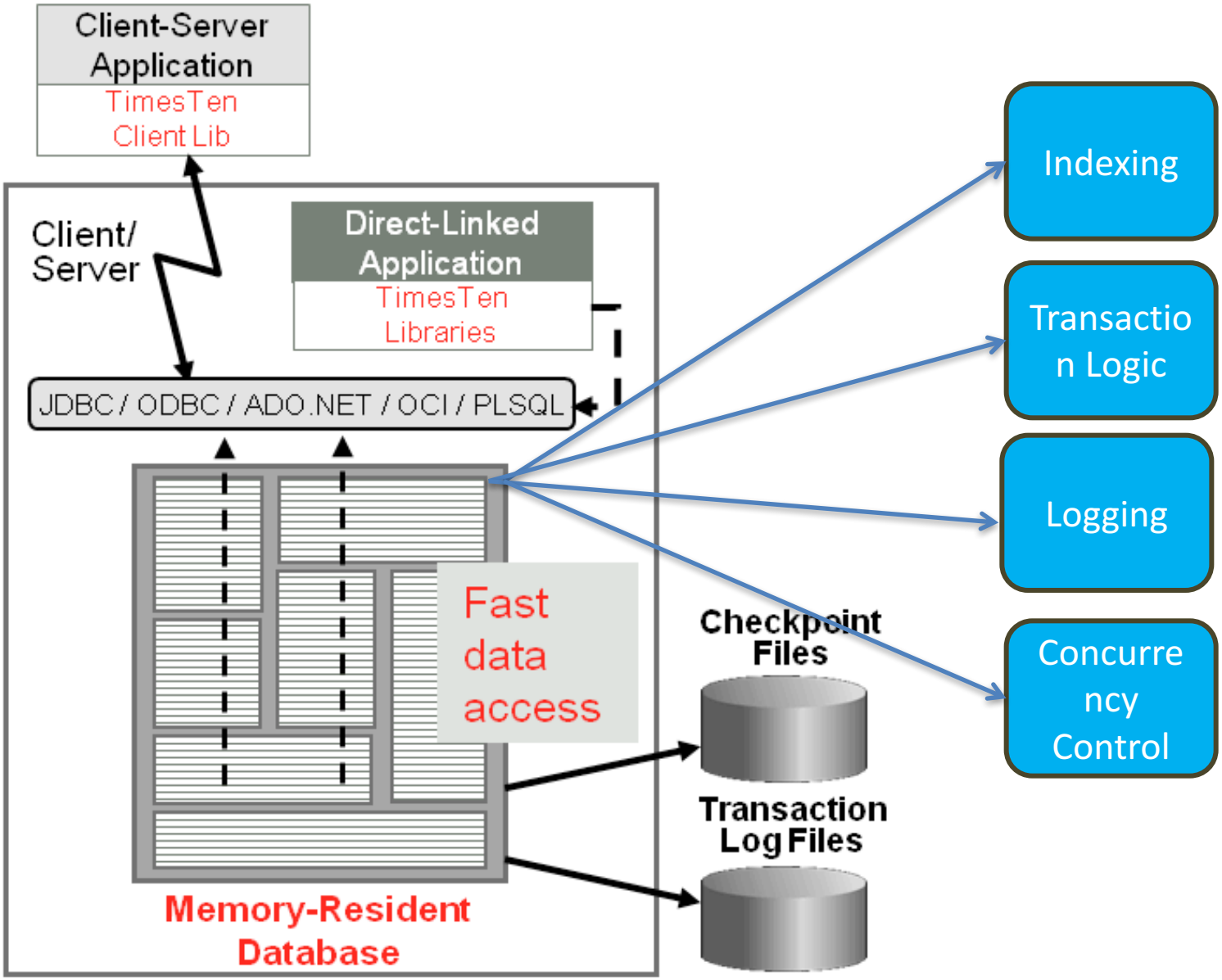
Before:每秒只能支持300-400个QPS的吞吐量

After:上万个QPS的吞吐量，而且查询速度可以保障在20毫秒左右





Example: In-memory Database



Source: Oracle TimesTen In-Memory Database Overview

SYSTEM SOFTWARE FOR PERSISTENT MEMORY

PMFS Overview

Introduction of *pm_wbarrier*

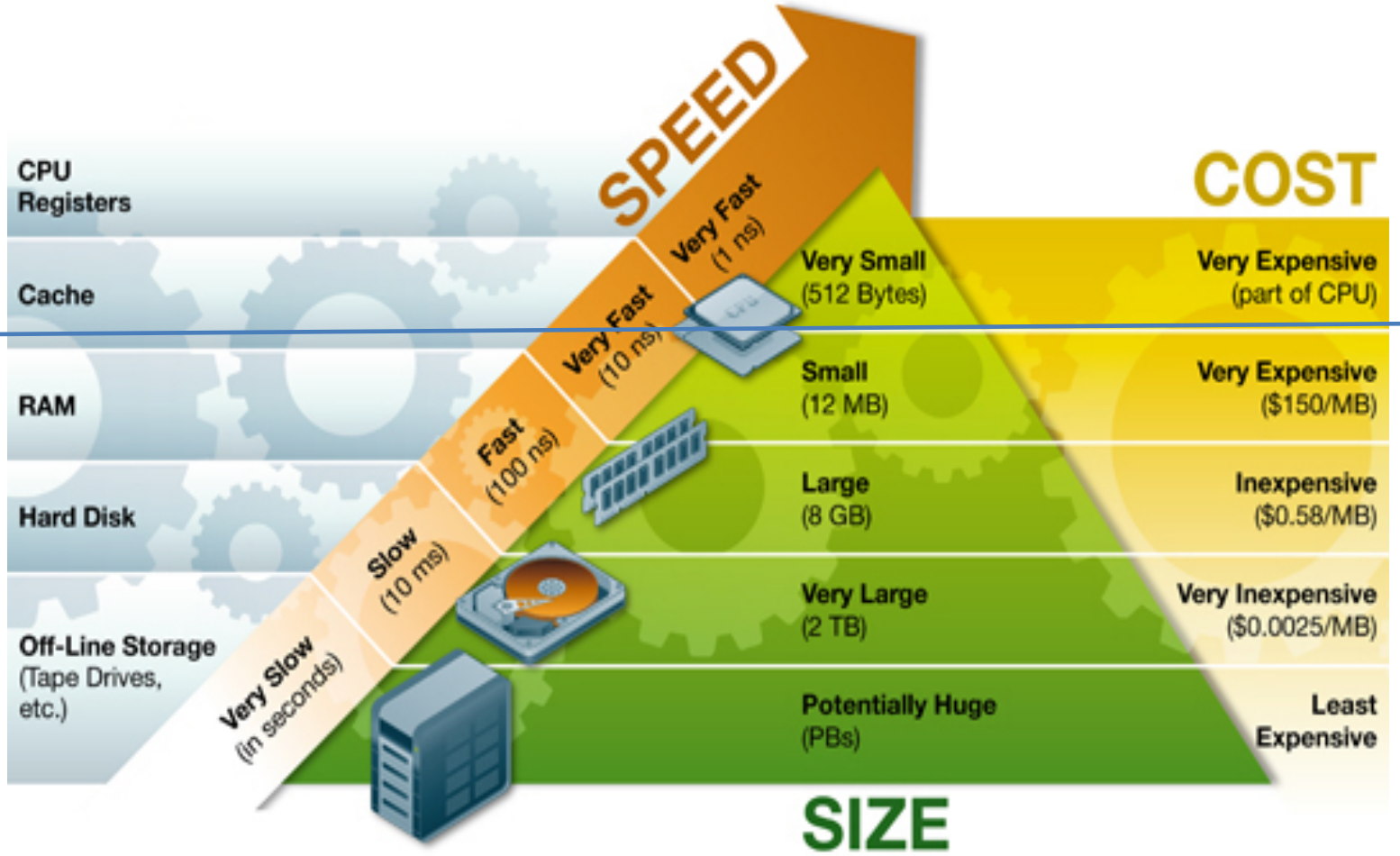
- *Now: Intel's pcommit*

File system architecture optimized for PM

- light-weight and consistent POSIX file system
- memory-mapped I/O
- protecting stray writes

Performance evaluation with PM emulator

Recall: NVM Memory Hierarchy



Caching problem in NVM

- Ordering
- Persistency

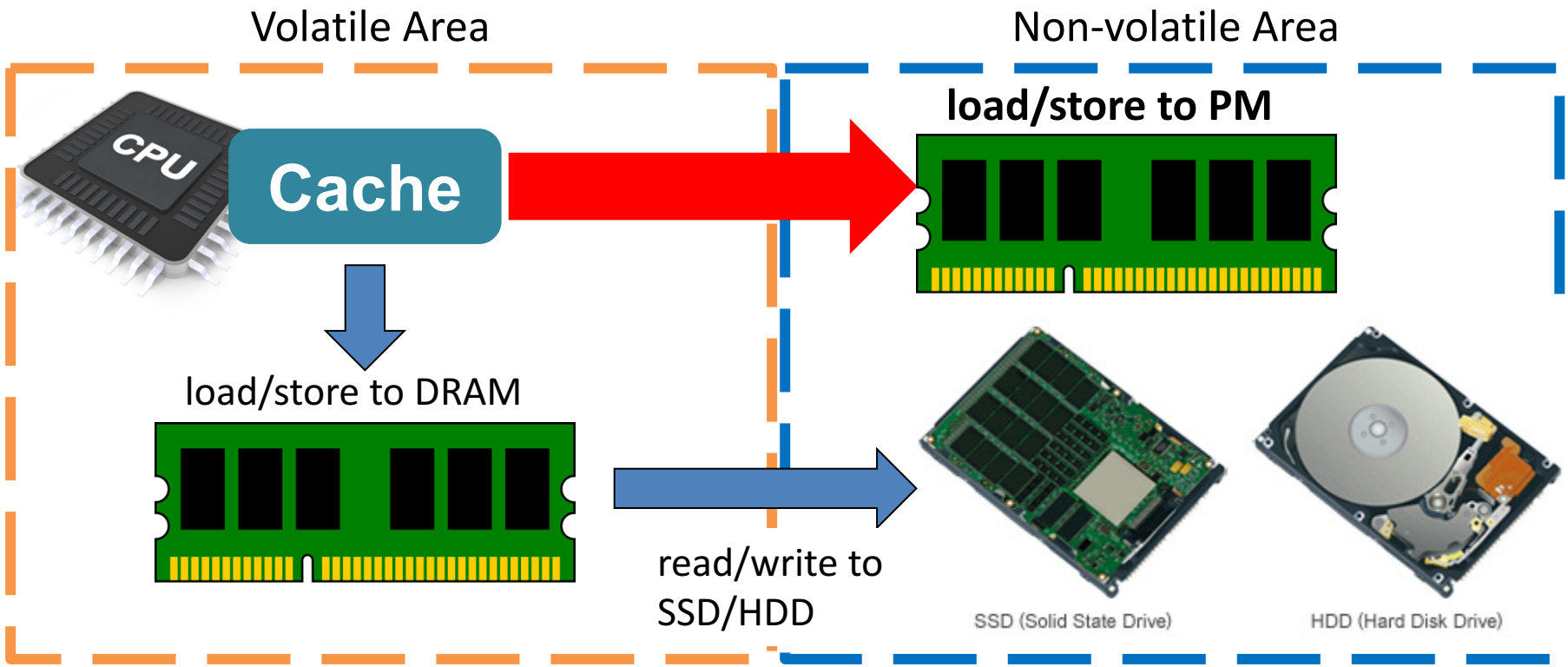


fig of HDD/SSD) http://storage-system.fujitsu.com/jp/lib-f/tech/beginner/ssd/

Ordering

- Recovery depends on write ordering

```
STORE data[0] = 0xFOOD
STORE data[1] = 0xBEEF
STORE valid = 1
```

Reordering breaks recovery

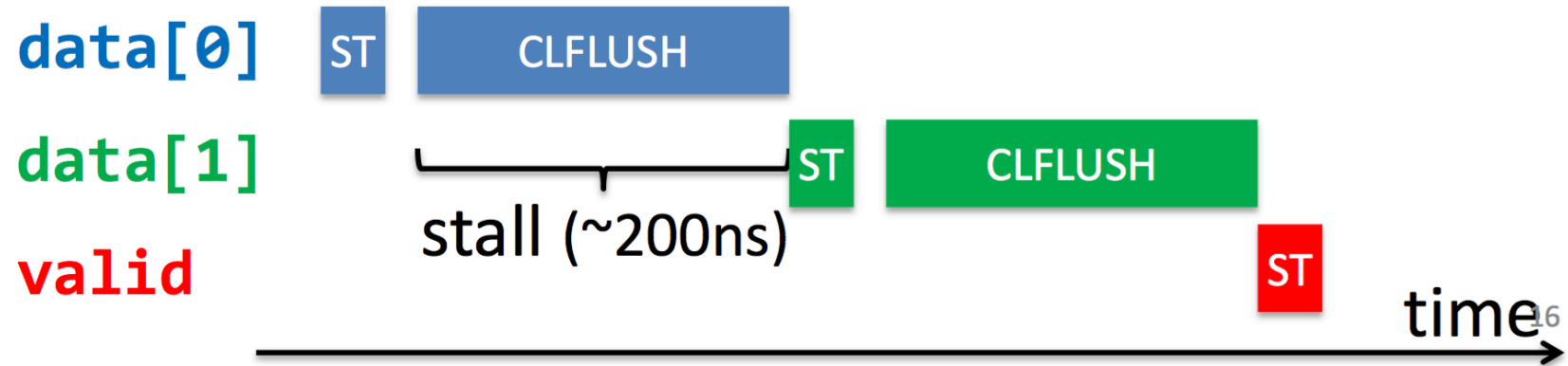
Recovery incorrectly considers garbage as valid data

Ordering with Existing Hardware

- Order writes by flushing cachelines via CLFLUSH

```
STORE data[0] = 0xFOOD
STORE data[1] = 0xBEEF
CLFLUSH data[0]
CLFLUSH data[1]
STORE valid = 1
```

- But CLFLUSH:
  - Stalls the CPU pipeline and serializes execution



Clflush

CLFLUSH—Flush Cache Line

Opcode	Instruction	Op/ En	64-bit Mode	Compat/ Leg Mode	Description
0F AE /7	CLFLUSH *m8*	A	Valid	Valid	Flushes cache line containing *m8*.

- Invalidates the cache line from all levels of the processor cache hierarchy (data and instruction)
- The invalidation is broadcast throughout the cache coherence domain
- If, at any level of the cache hierarchy, the line is inconsistent with memory (dirty) it is written to memory before invalidation.

Ordering with Existing Hardware

- Order writes by flushing cachelines via CLFLUSH

```
STORE data[0] = 0xFOOD
STORE data[1] = 0xBEEF
CLFLUSH data[0]
CLFLUSH data[1]
STORE valid = 1
```

- But CLFLUSH:
  - Stalls the CPU pipeline and serializes execution
  - Invalidates the cacheline
  - Only sends data to the memory subsystem – **does not commit** data to NVM

Caching problem in PM

- clflush cannot flush from memory controller
  - Cannot guarantee real persistency

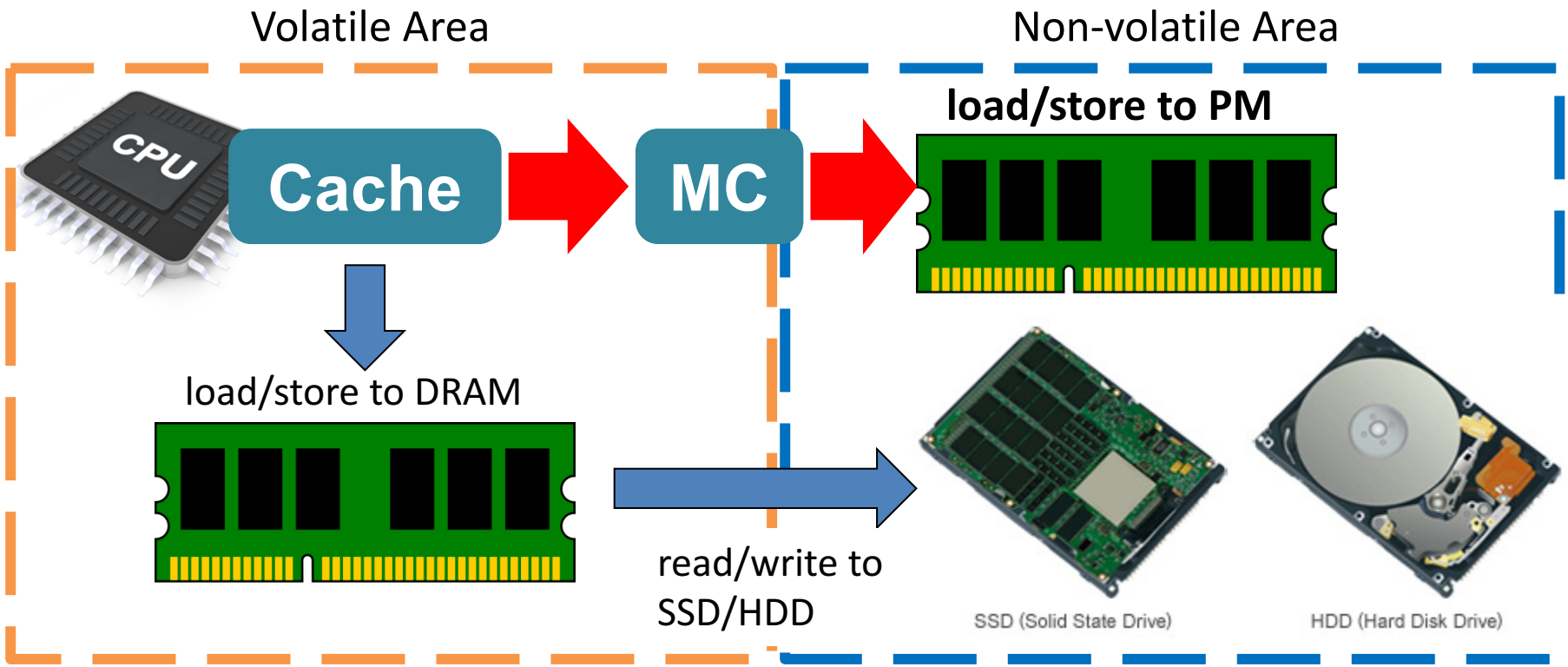


fig of HDD/SSD) http://storage-system.fujitsu.com/jp/lib-f/tech/beginner/ssd/

Proposal: pm_wbarrier

Feature

- Enforce the durability of a cacheline

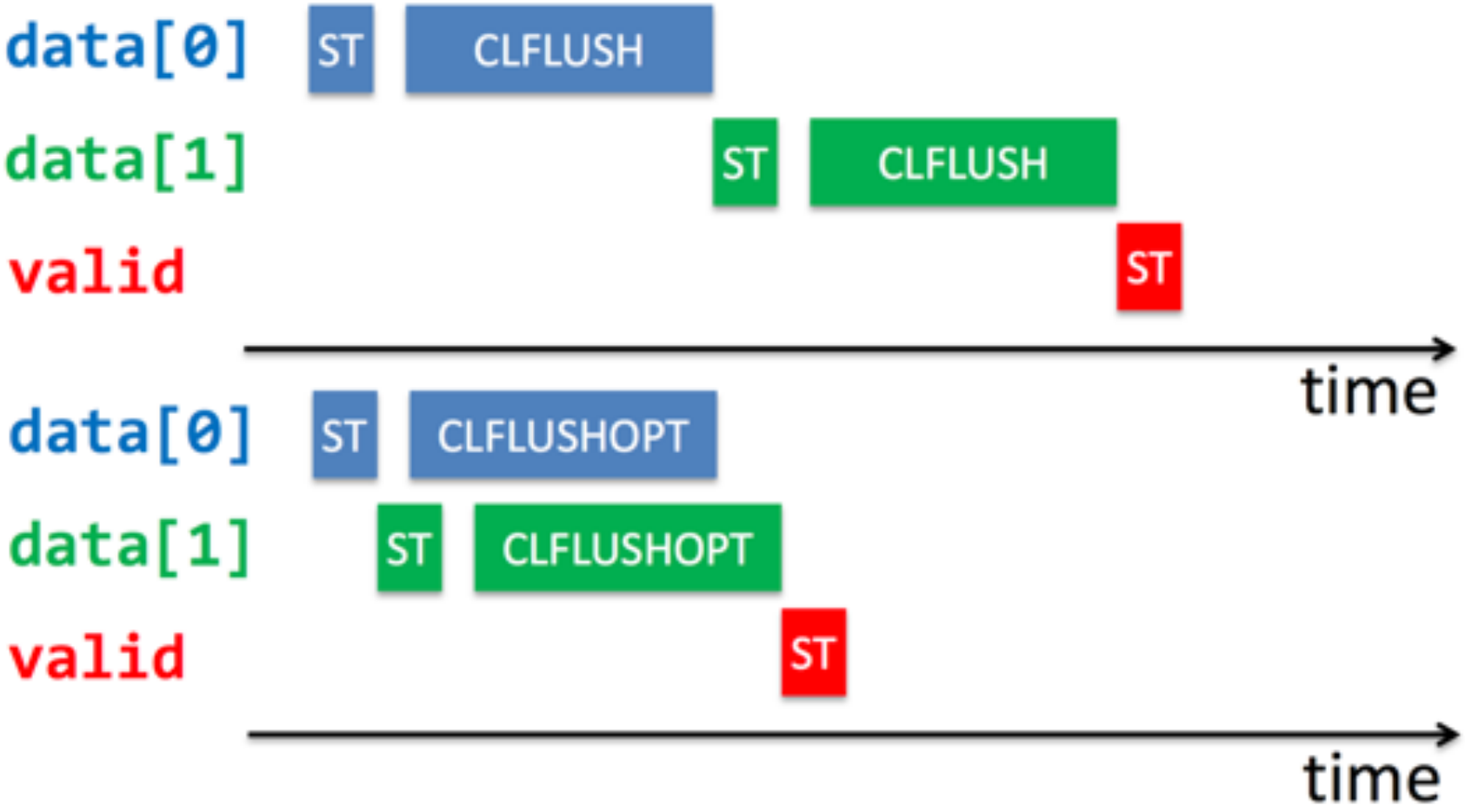
Steps of usage

1. clflush A
   - flush the cacheline contains A
2. sfence
   - ensure the completion of store
3. pm_wbarrier
   - ensure the durability of every store to PM

Fixing CLFLUSH: Recent Intel x86 Extensions

- CLFLUSHOPT
  - Provides unordered version of CLFLUSH
  - Supports efficient cache flushing
- CLWB
  - Write backs modified data of a cacheline
  - Does not invalidate the line from the cache
  - Marks the line as non-modified
- PCOMMIT
  - Commits data writes queued in the memory subsystem to NVM

CLFLUSHOPT


data[0]
ST
CLFLUSH
data[1]
ST
CLFLUSH
valid
ST
time
data[0]
ST
CLFLUSHOPT
data[1]
ST
CLFLUSHOPT
valid
ST
time

PCOMMIT

```
STORE data[0] = 0xFOOD
STORE data[1] = 0xBEEF
CLWB data[0]
CLWB data[1]
SFENCE // orders subsequent PCOMMIT
PCOMMIT // commits data[0], data[1]
SFENCE // orders subsequent stores
STORE valid = 1
```

Outline

- Volatile cache problem
- **Architecture**
  - Consistency
  - Write protection from stray writes
- Implementation
- Evaluation
- Related Work
- Conclusion

Layout of PMFS

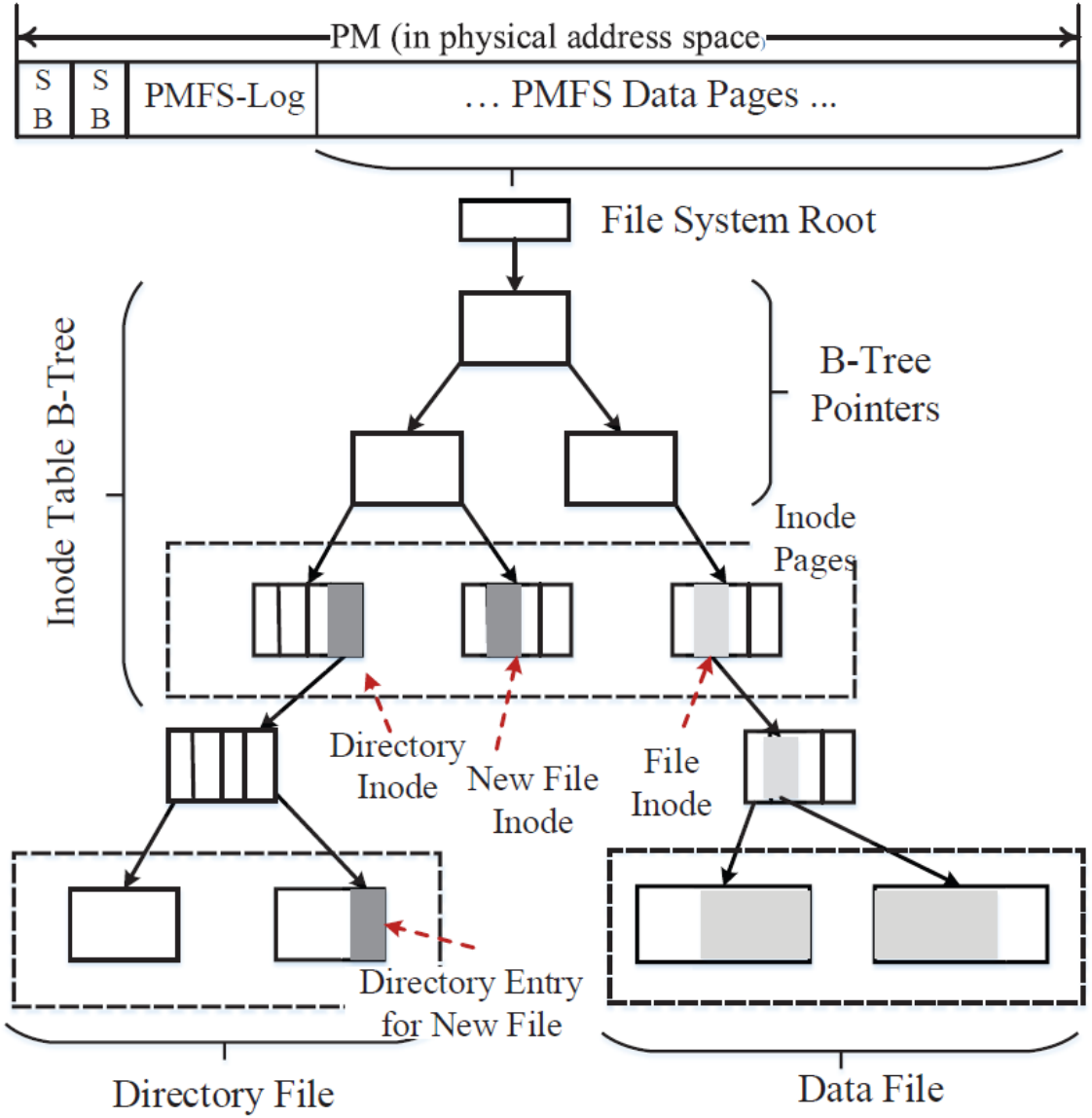


Figure 3: PMFS data layout

Consistency

Three existing techniques:

- Copy on Write (CoW)
- Journaling
- Log-structured updates

Used for updates on Data Area

Used for updates on Meta Data (inode)

One more PM specific technique:

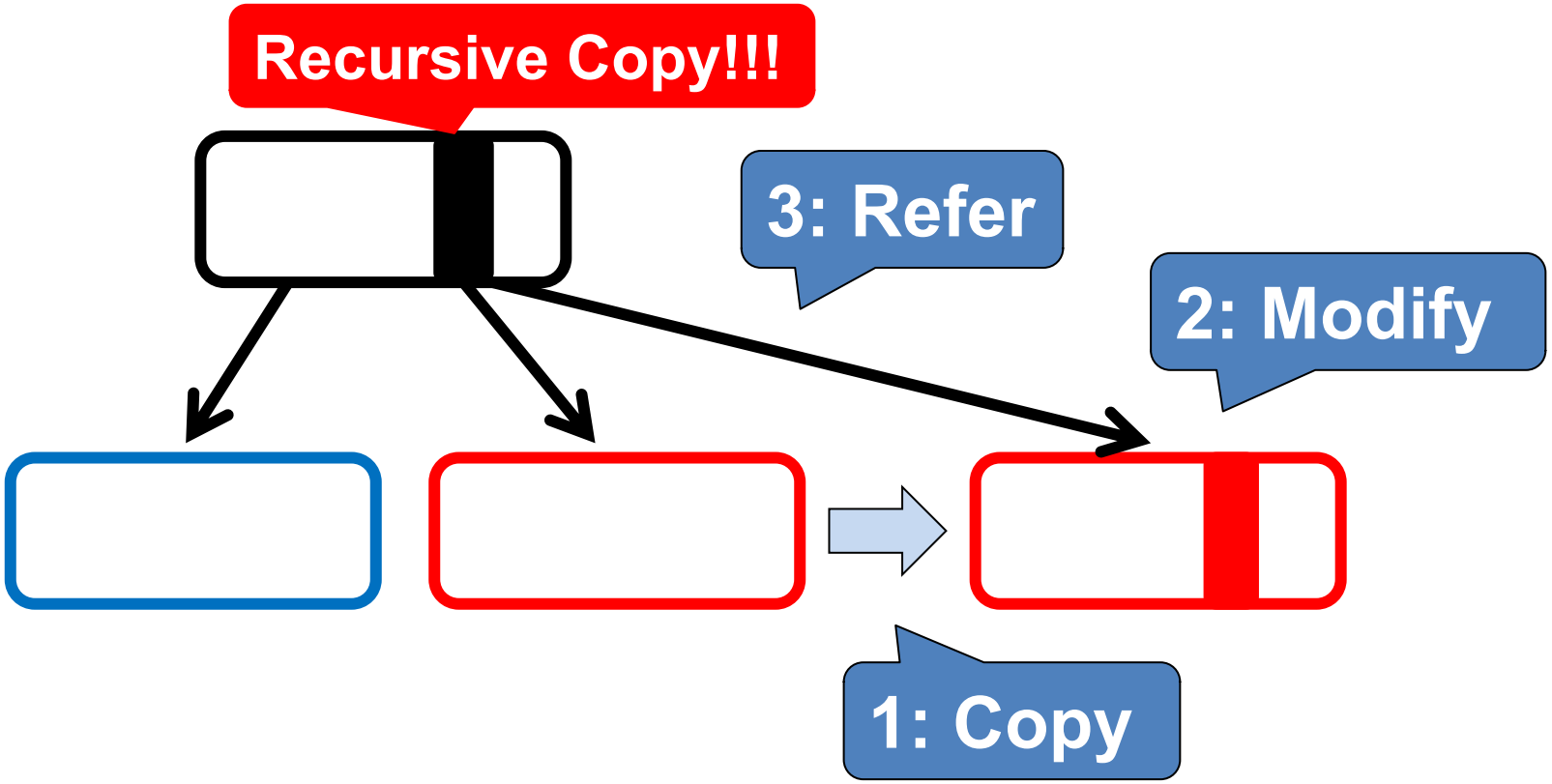
- Atomic in-place writes

Used for updates of small portion of data

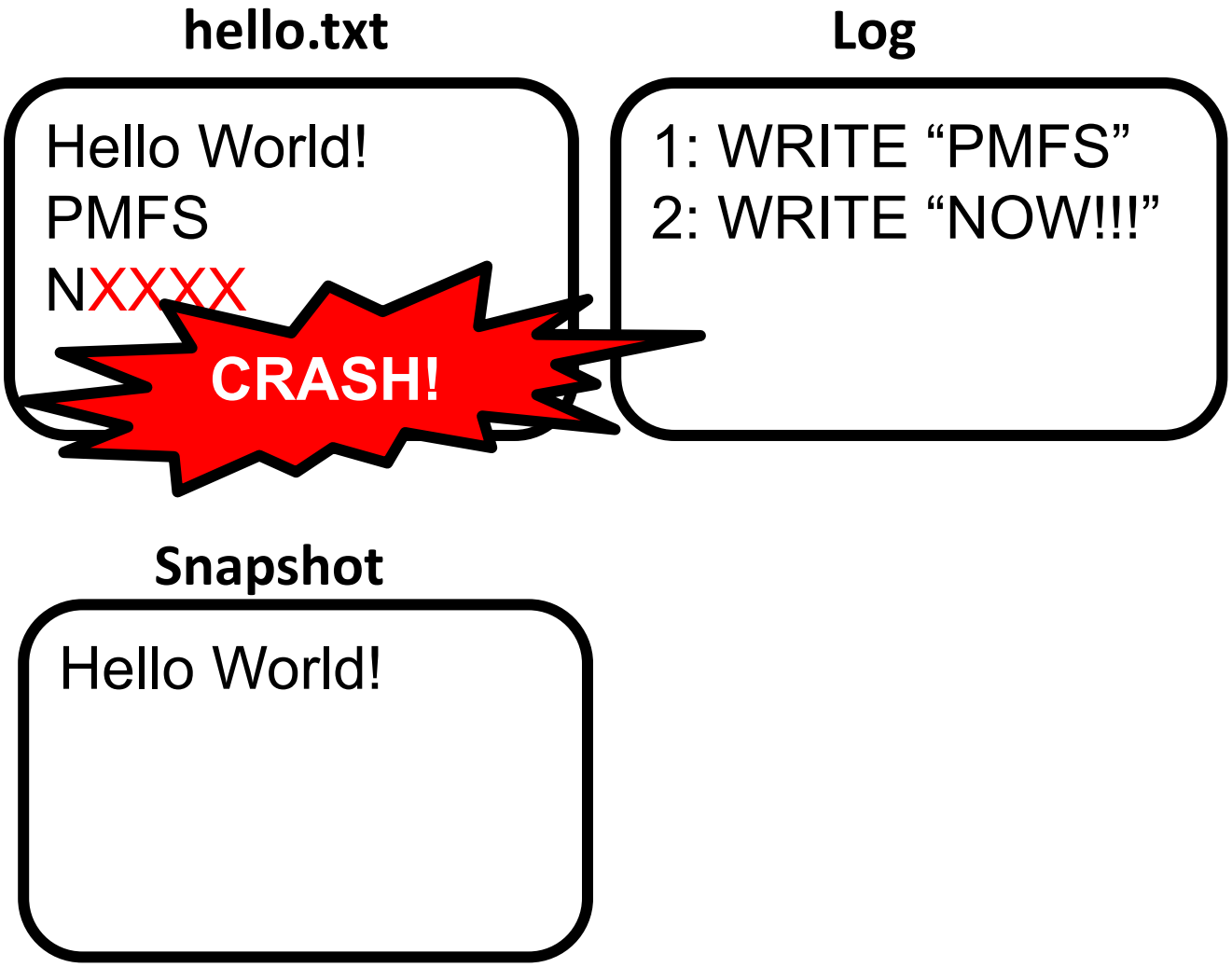
Copy on Write (Shadow Paging)

Safe and consistent method to modify data

Three steps: Copy, Modify, Refer



Journaling in PMFS



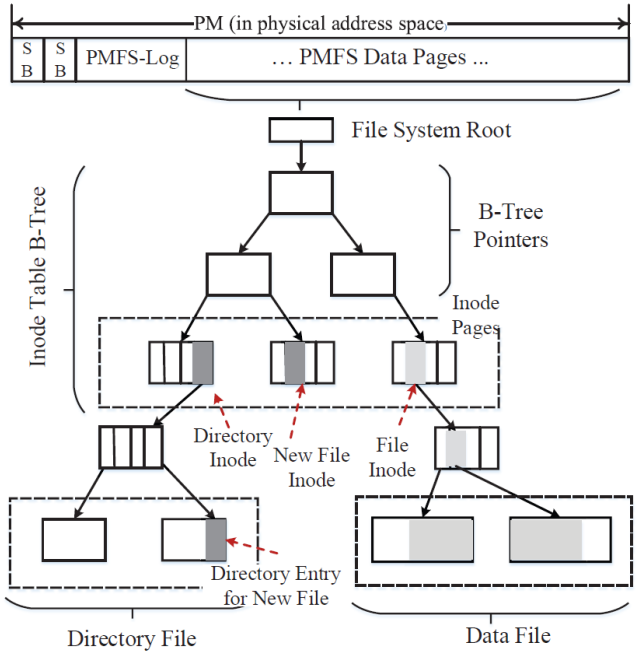
Hybrid Approach in PMFS

Metadata

- Updated by *fine-grained logging*

Data

- Use Copy on Write method



	Distributed small modification	Centralized large modification
Copy on Write	✕ (Write Amplification)	◯ (Freely after copy)
Journaling	◯ (Just append logs)	✕ (Double writes)

Fine-grained Logging

64 Bytes granularity is good
for logging of file system metadata

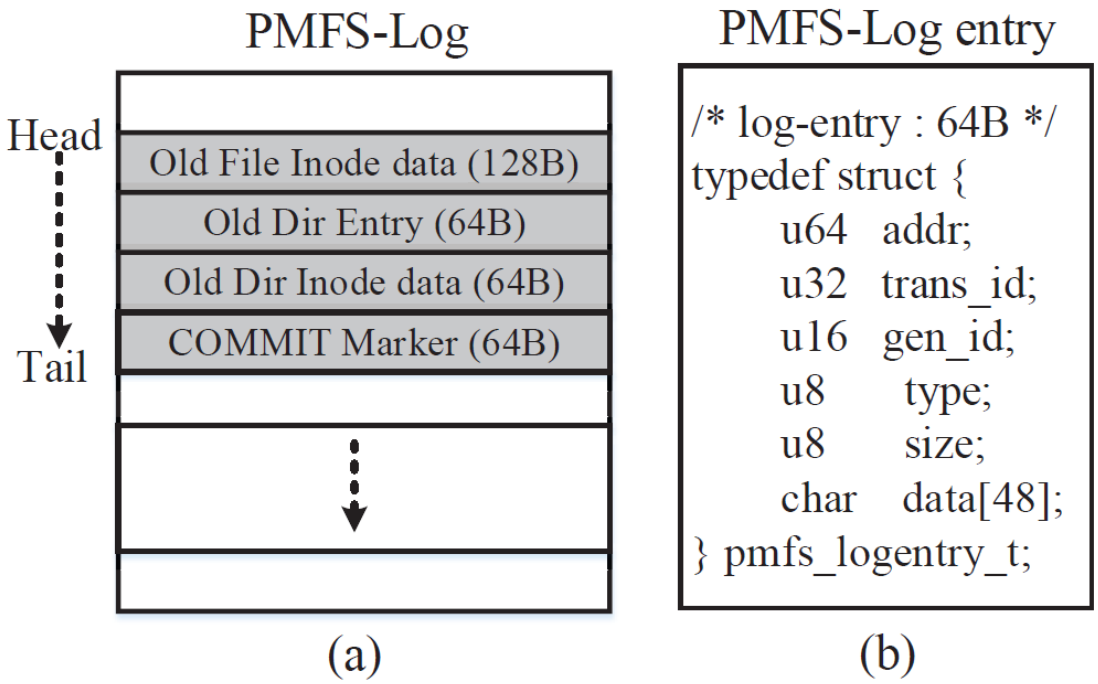


Figure 4: PMFS journaling data structures

Redo vs. Undo

- Redo: the new data is logged and made durable before writing the data to the file system
  - new data is written to the file system only when the transaction commits successfully
- Undo: the old data (in the file system) is first logged and made durable
  - The new data is written directly (in-place) to the file system
- What does PMFS use? Why?

Extended Atomic In-place Writes

8 bytes

- CPU natively supports 8-byte atomic writes
- Update inode's access time

16 bytes

- Using *cmpxchg16b* instruction
- Update inode's size and modification time

64 bytes

- Using RTM (introduced in Haswell)
- Update a number of inode fields like delete

Write Protection Stray Writes

Supervisor Mode Access Protection (SMAP)

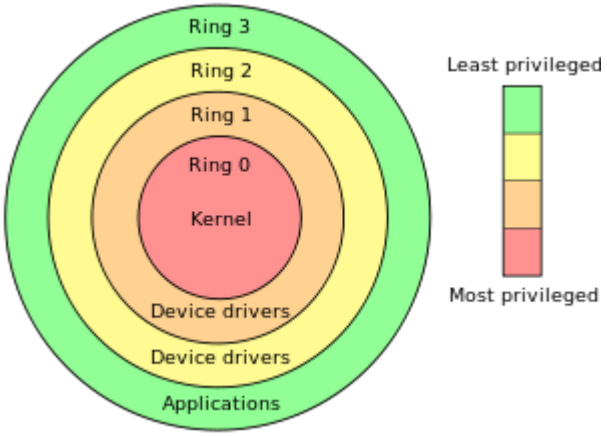
- Prohibit writes into user area

Write windows (introduced in this paper)

- Mount as read-only
- When writing, CR0.WP is set to zero

	User	Kernel
User	Process Isolation	SMAP
Kernel	Privilege Levels	Write windows

Table 2: Overview of PM Write Protection



Right) http://en.wikipedia.org/wiki/Protection_ring

Implementation on Linux

Execution In Place (XIP)

- Interface of loading data from Flash directly in limited RAM environment
- Used to avoid the block device/page cache layer

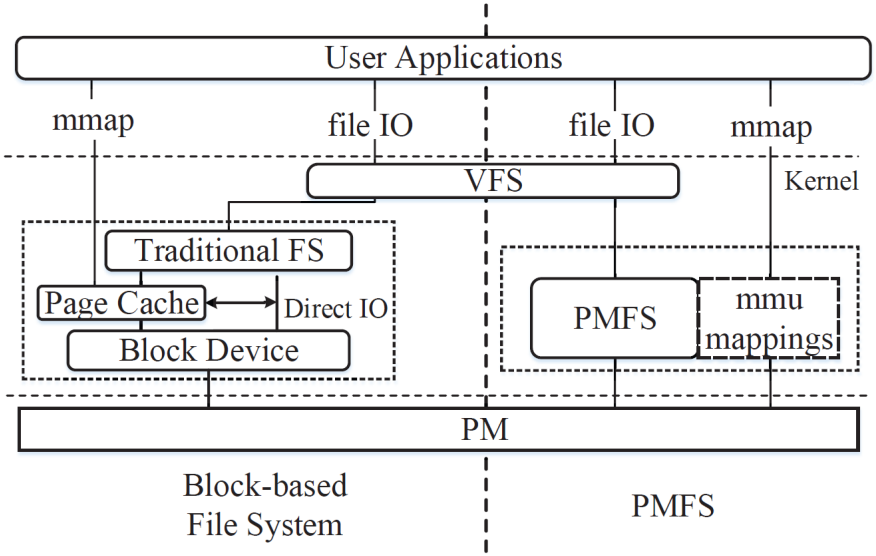


Figure 2: PMFS vs Traditional File Systems

Testing and Validation

- Yat: Hypervisor-based validation framework
  - Ensure **cache flushing** and **pm_wbarrier** are executed in correct order
  - Paper is published in USENIX ATC'14

Evaluation

Environment

- PM Emulation Platform (PMEP)
- PM Block Driver (PMBD)

Results

- File-based Access
- Memory-Mapped I/O
- Write Protection

Summary

- Non-volatile memory to the market soon
- Changes the landscape of systems software
- PMFS
  - PM_barrier interface
  - Well-considered consistency protocol
  - Real evaluation with PM emulator