Failure-Atomic Slotted Paging for Persistent Memory

Beomseok Nam

UNIST (Ulsan National Institute of Science and Technology)

Byte-Addressable Non-Volatile Memory

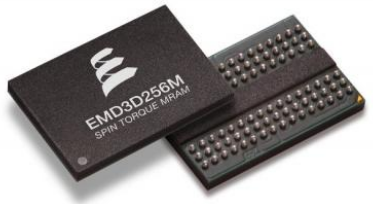
- Non-Volatile Memory (NVM)

	NAND	STT-MRAM	PCM	DRAM
Non-volatility	o	o	o	x
Read (ns)	2.5 X 10^4	5 - 30	20 – 70	10
Write (ns)	2 X 10^5	10 - 100	150 - 220	10
Byte-addressable	x	o	o	o
Density	185.8 Gbit/cm^2	0.36 Gbit/cm^2	13.5 Gbit/cm^2	9.1 Gbit/cm^2

K. Suzuki and S. Swanson. “A Survey of Trends in Non-Volatile Memory Technologies: 2000-2014”, IMW 2015

Non-volatile

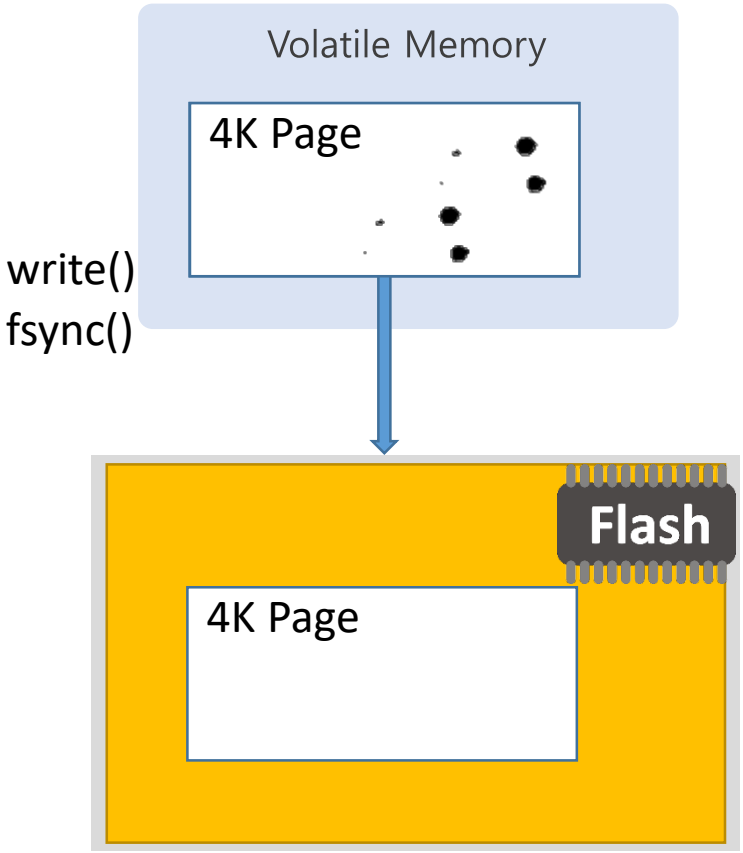
Low Latency



Persistent Memory

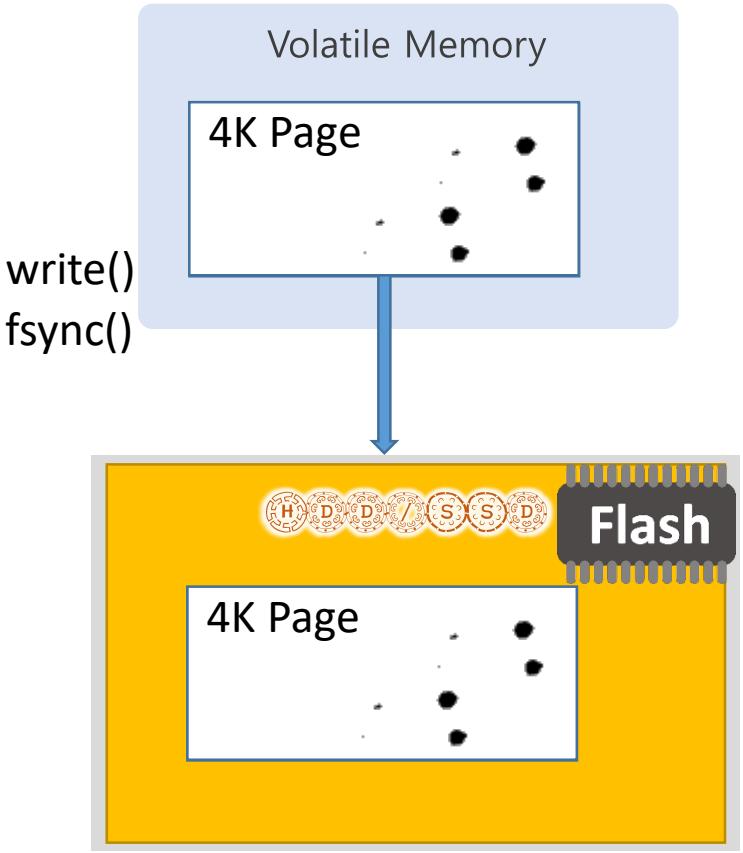
Byte-Addressable Persistency

- When Granularity of Atomicity = Page

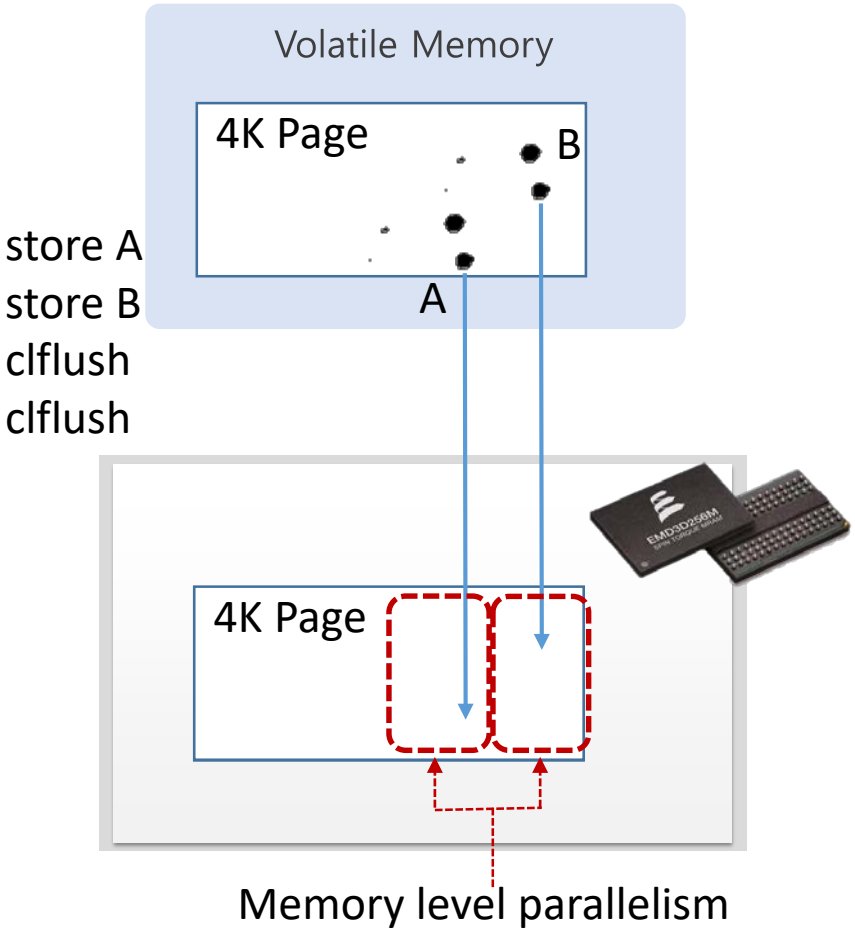


Byte-Addressable Persistency

- When Granularity of Atomicity = Page

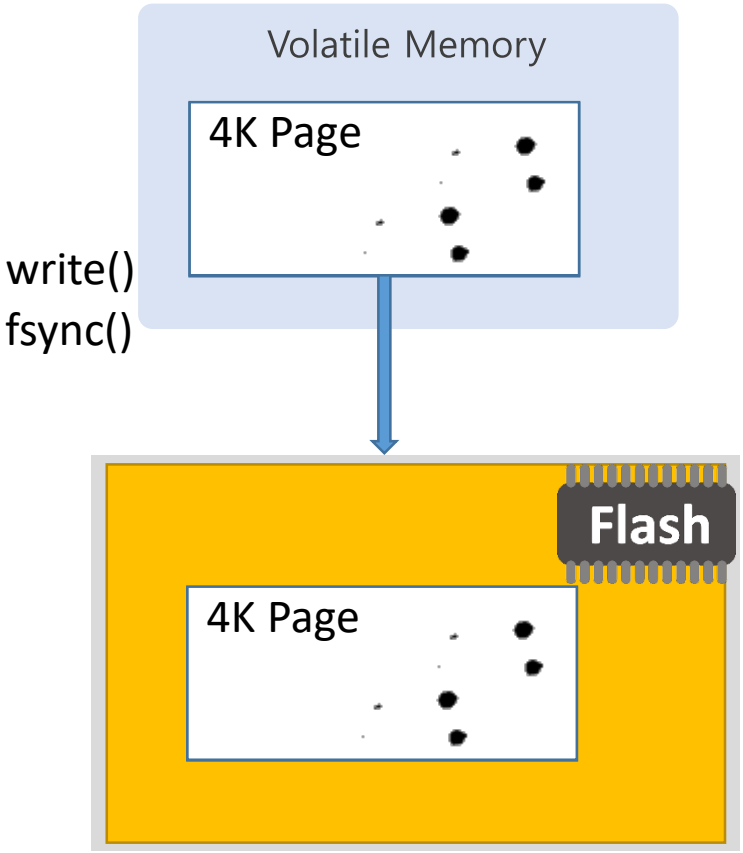


- When Granularity of Atomicity = Cache Line

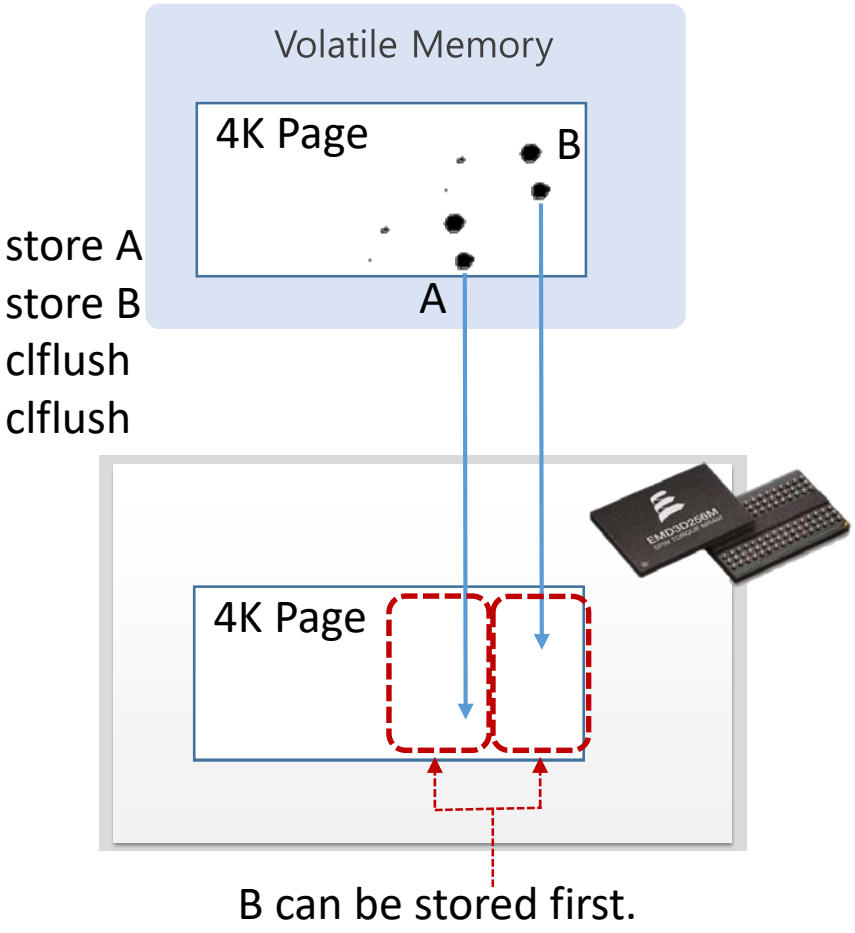


Byte-Addressable Persistency

- When Granularity of Atomicity = Page

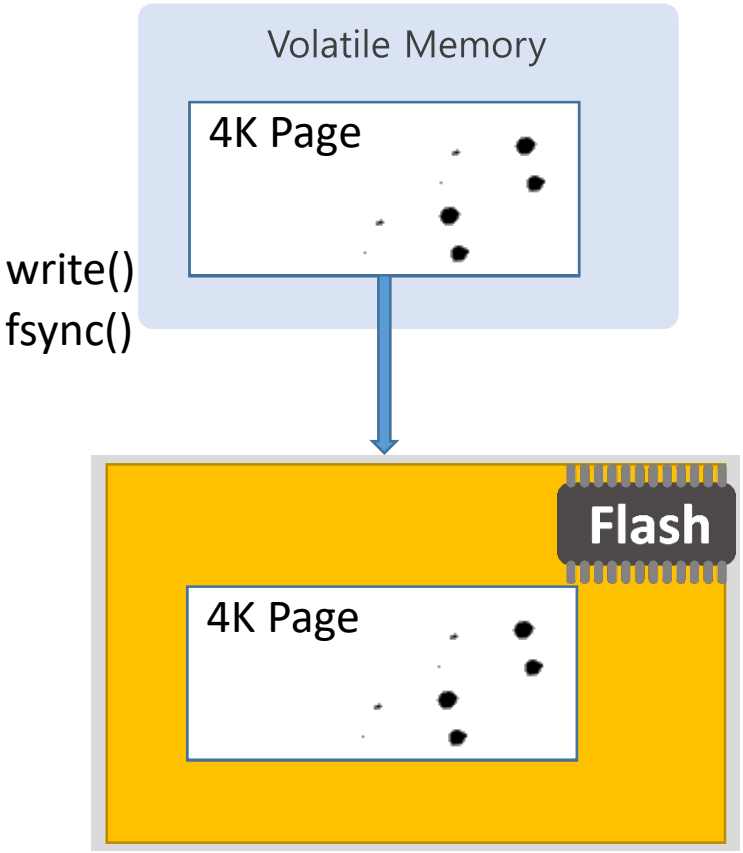


- When Granularity of Atomicity = Cache Line

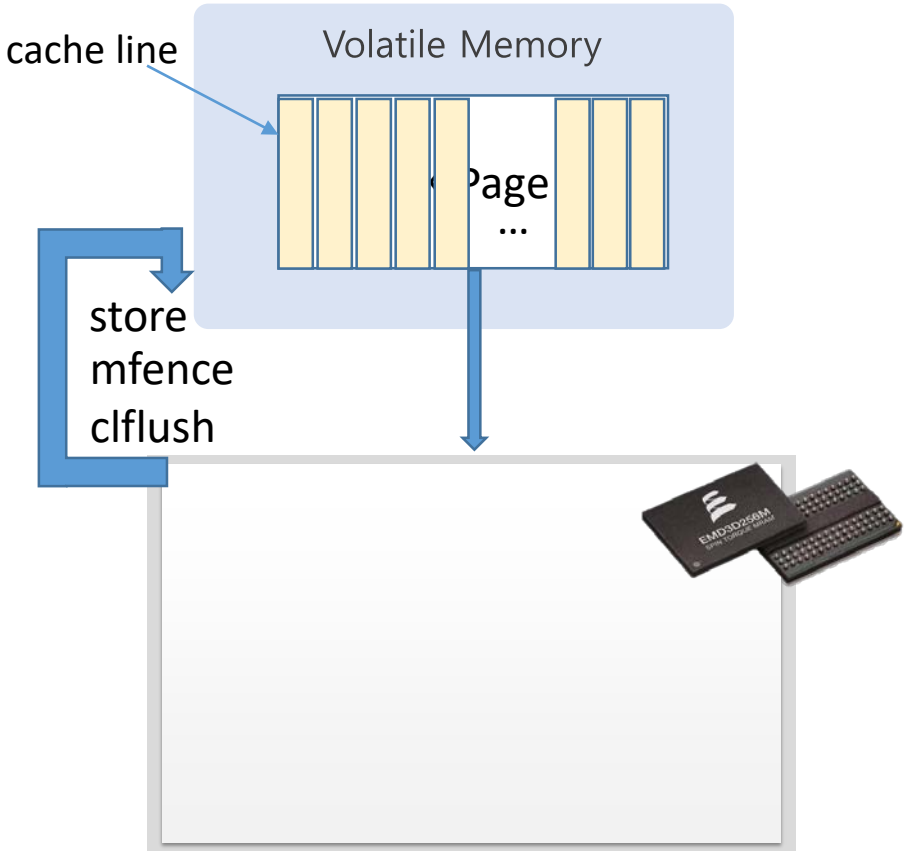


Byte-Addressable Persistency

- When Granularity of Atomicity = Page



- When Granularity of Atomicity = Cache Line

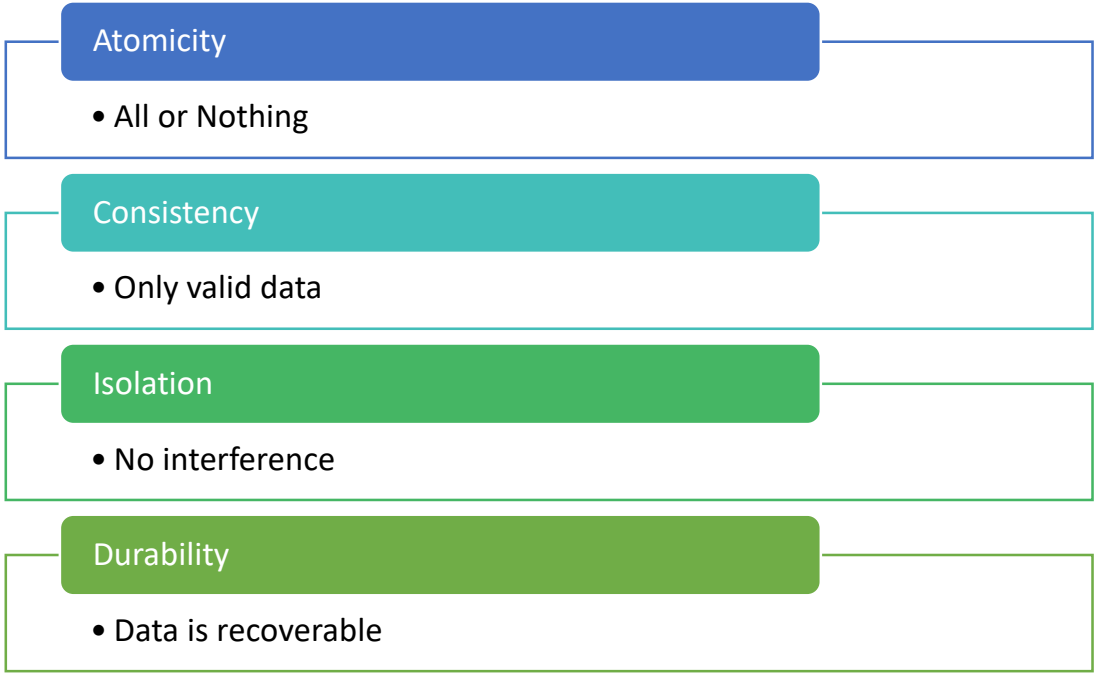


Legacy Block IO Interface requires too many barriers and clflushes unnecessarily

Byte-Addressable ACID

- fsync() vs. a group of mfence and clflush instructions
  - Faster than flash memory, but there's room for improvement.

- Need to make transactions be aware of byte-addressability of NVM

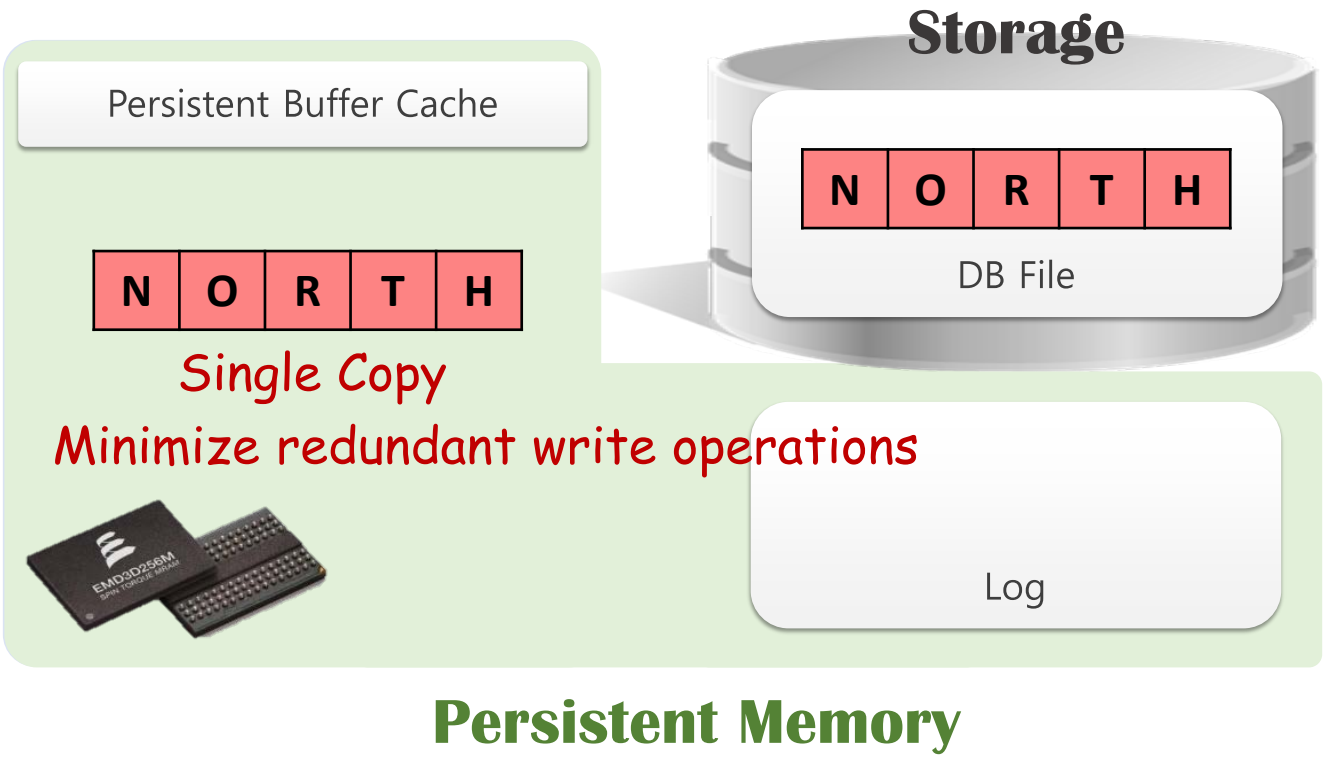


Failure-Atomic Slotted Paging for Persistent Memory

ASPLOS 2017

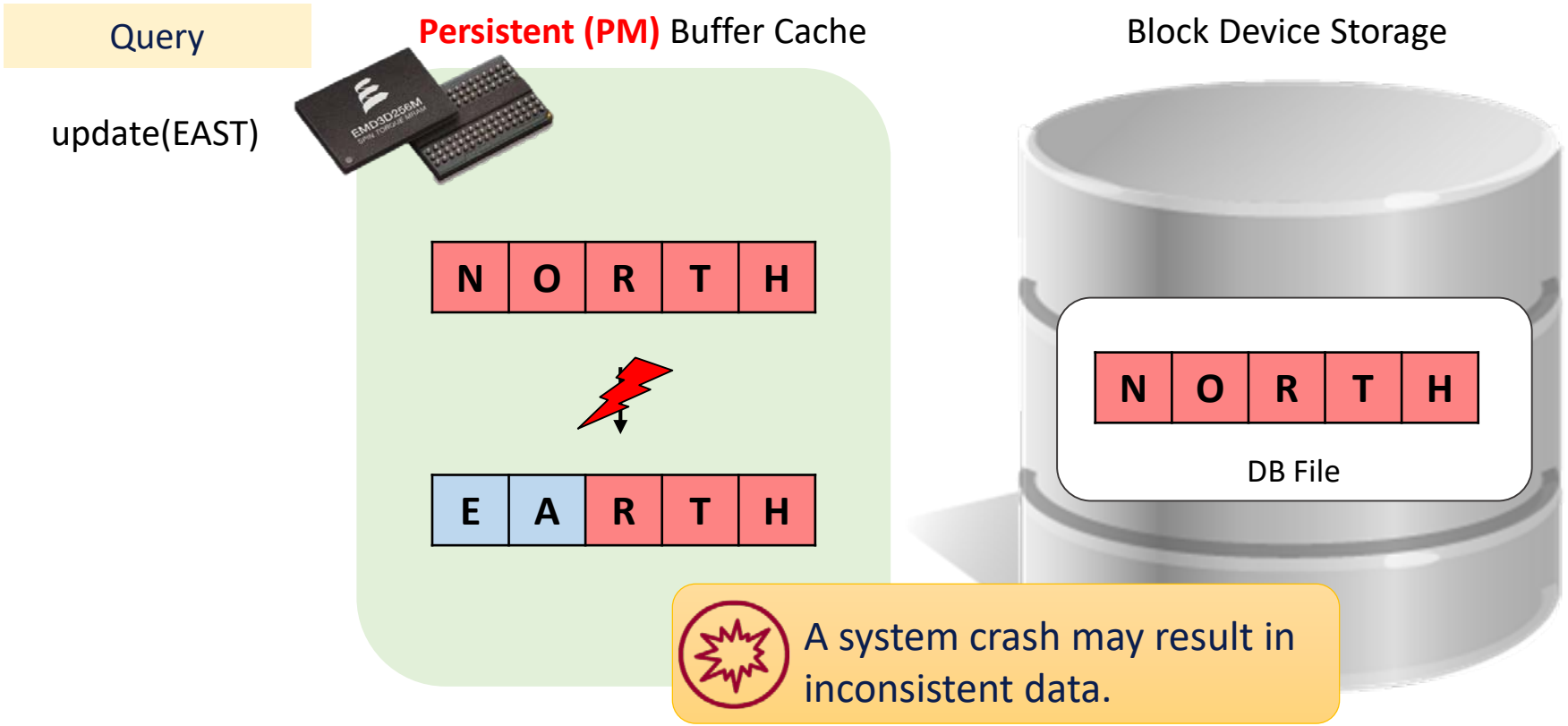
How can DBMS benefit from non-volatile buffer cache?

- **Considering NVRAM as main memory**
  - Do we need a Log file when DB buffer cache is non-volatile?

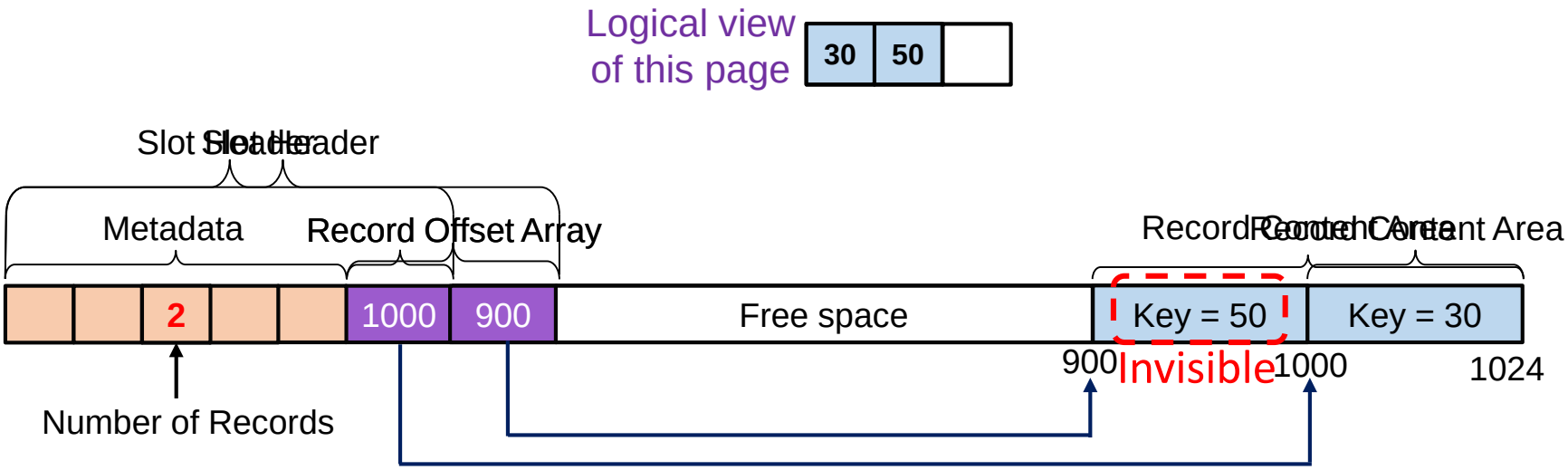


How can DBMS benefit from persistent memory?

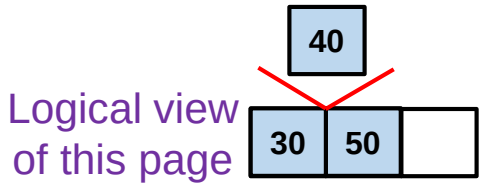
- **Challenge**
  - How to guarantee ACID with persistent buffer caching?
    - E.g.) Updates must be invisible until the transaction commits

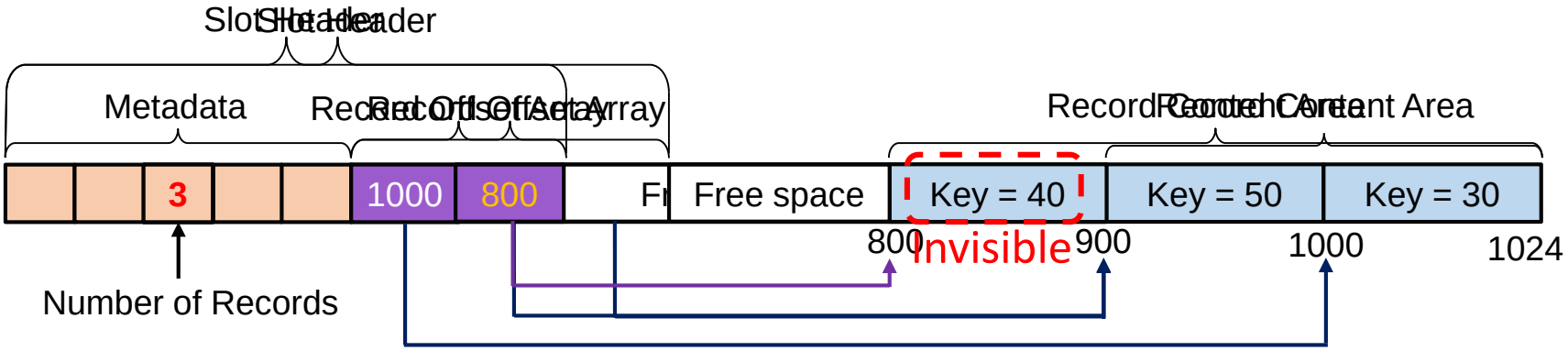


- **Slotted Page Structure**
  - The most widely used database page format for variable-length records
  - Structure
    - Slot Header
      - # of records
      - Record offset array for the ordering of keys
    - Record Content Area
      - Record contents (Append Only)
    - Free Space

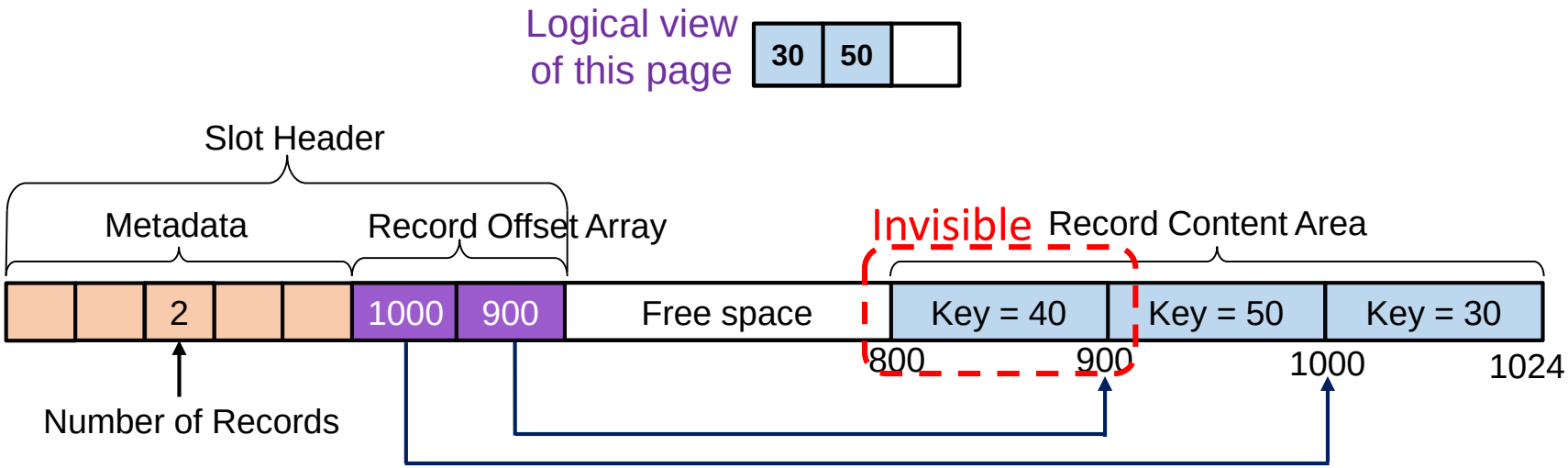


- **Slotted Page Structure**
  - The most widely used database page format for variable-length records
  - Structure
    - Slot Header
      - # of records
      - Record offset array for the ordering of keys
    - Record Content Area
      - Record contents (Append Only)
    - Free Space





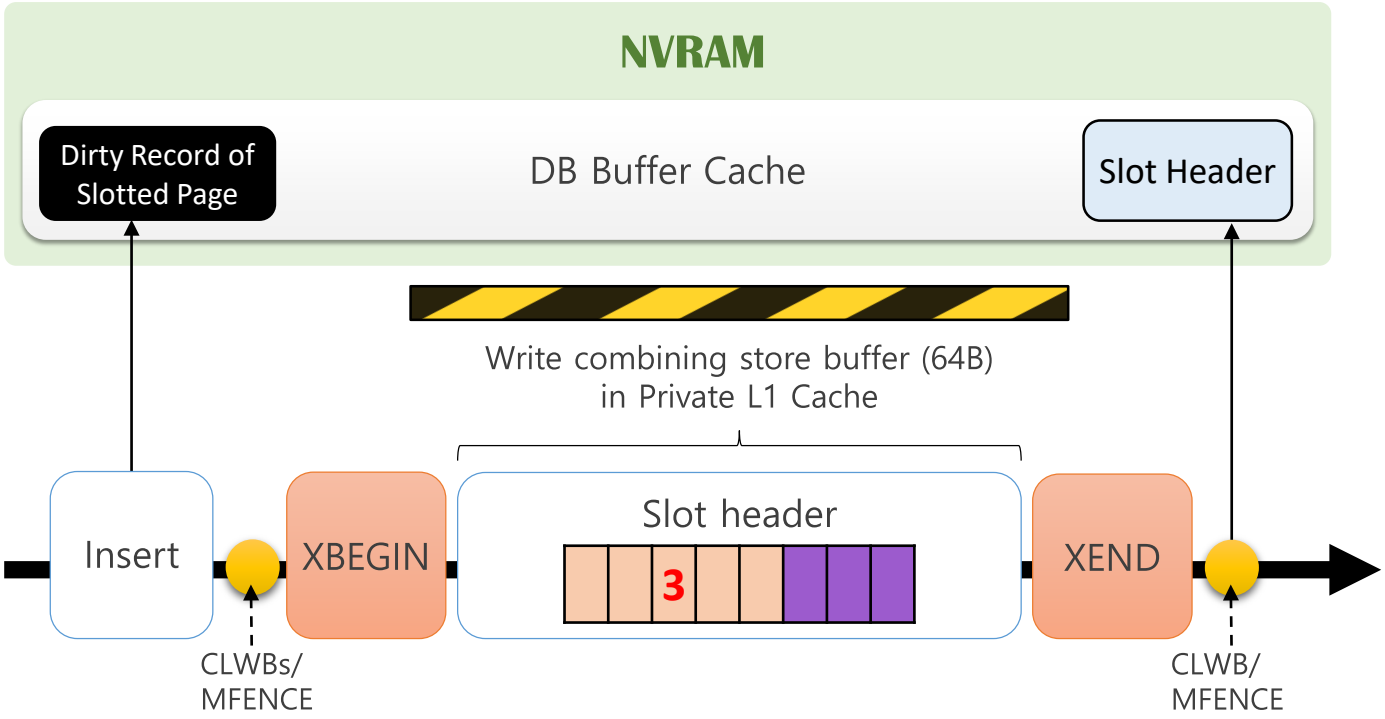
- **Failure-Atomic In-Place Commit Scheme**



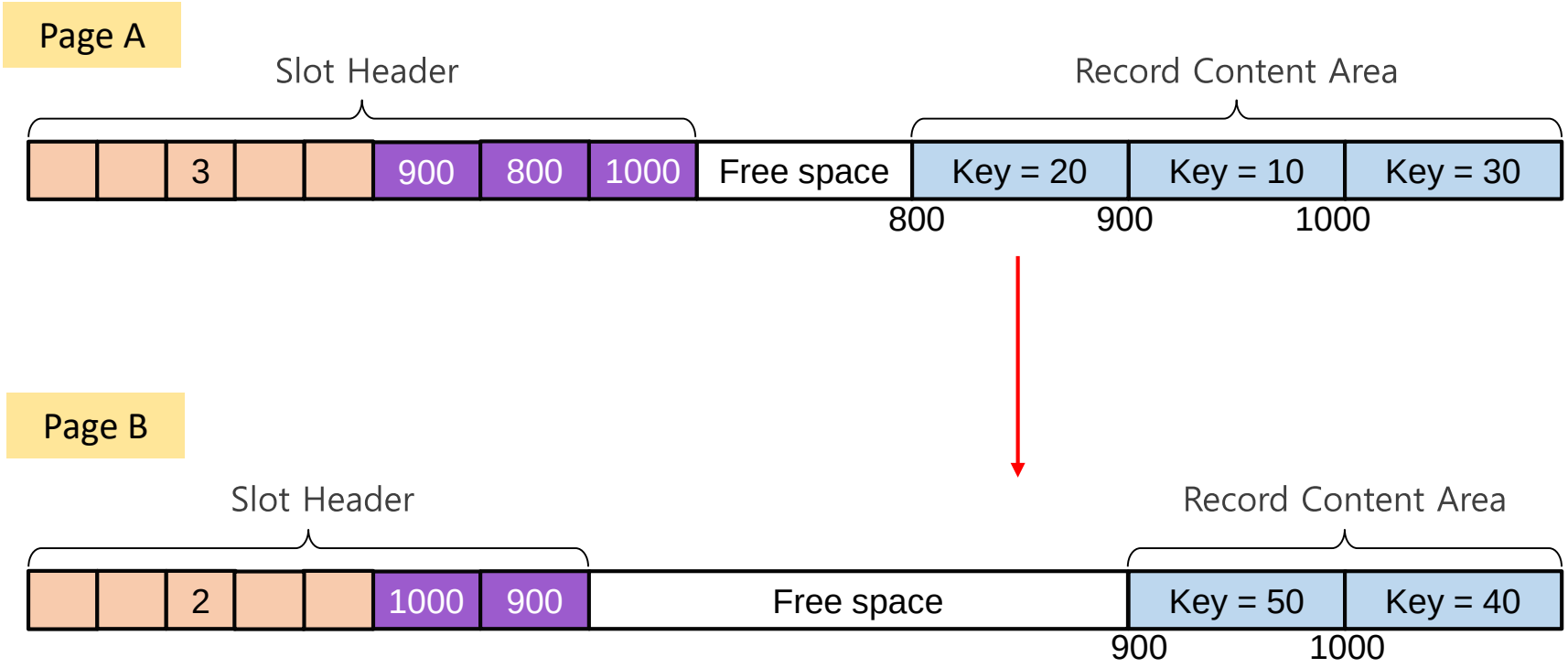
Unless record offset array is updated, modifications are not visible.
- Let's use the slot header as a commit marker!

NVRAM is expected to guarantee failure-atomic writes of 8-bytes.
How do we atomically update the slot header (>8 bytes)?

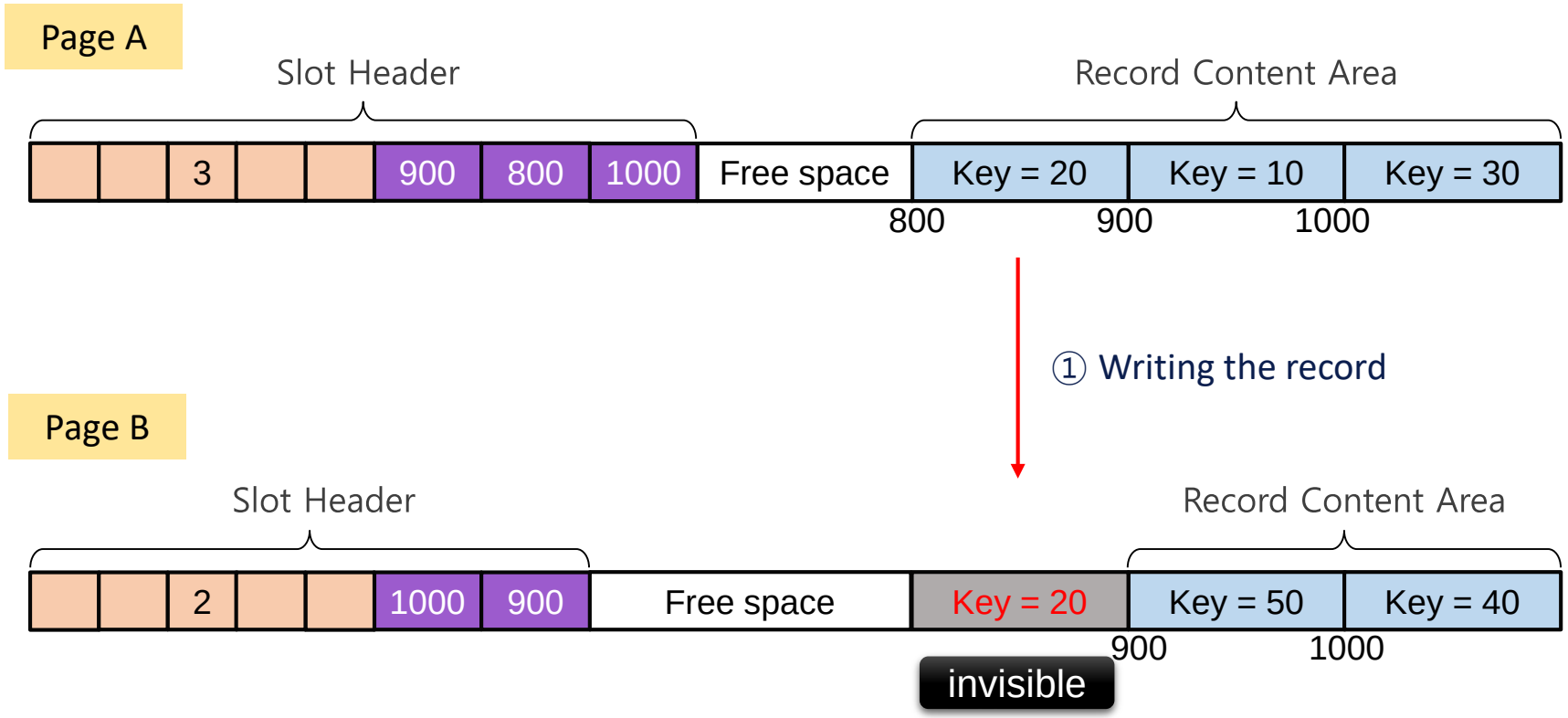
- **Failure-Atomic In-Place Commit Scheme**
  - Hardware Transactional Memory (HTM)
    - We can guarantee failure-atomic cache line write operation using HTM
    - Intel's Restricted Transactional Memory (RTM) is used to prevent a partially updated cache line from being written to NVRAM.



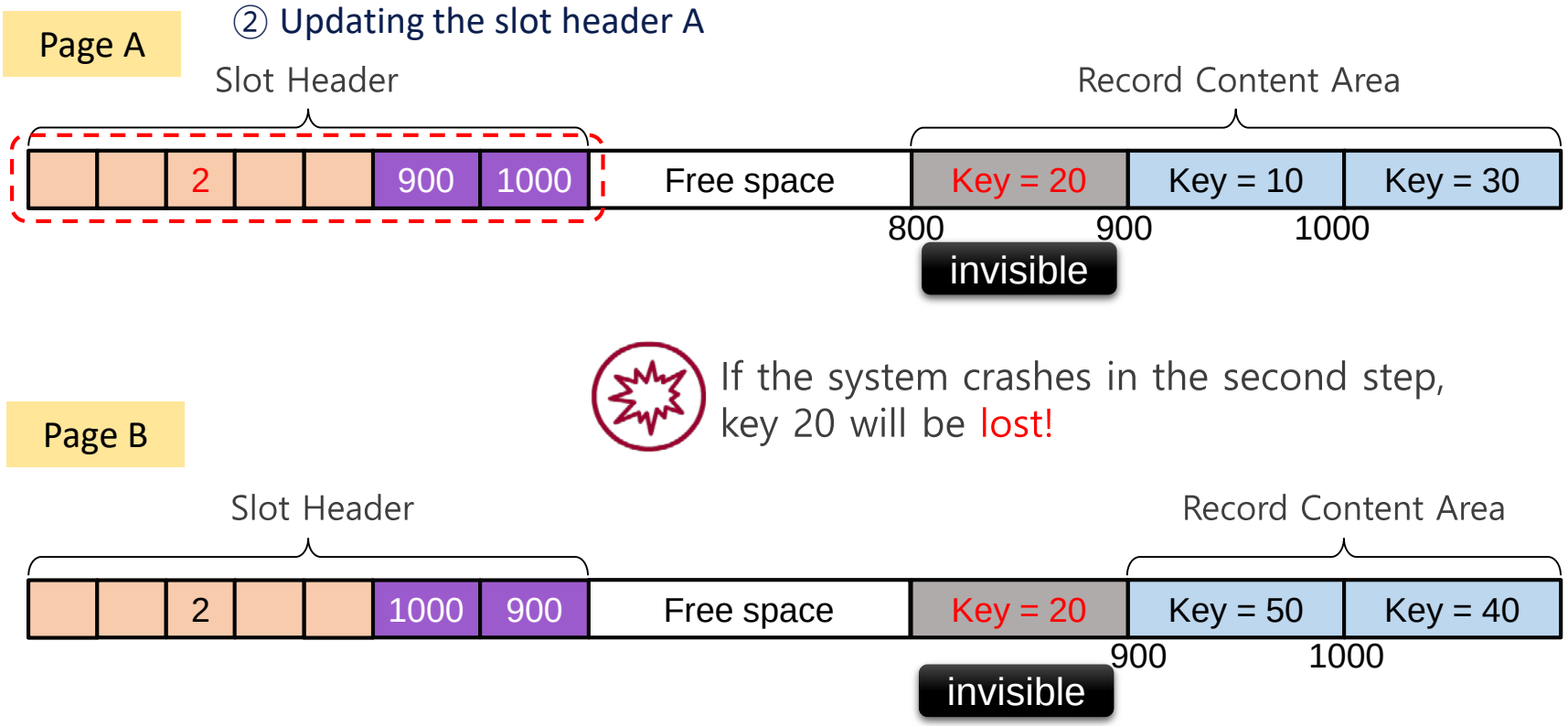
- **Failure-Atomic Slot-Header Logging Scheme**
  - What if a DB transaction modifies multiple pages?
  - The Failure-Atomic In-place Commit scheme works only for a single page modification.
  - ➢ Example: Move data with key 20 from page A to page B



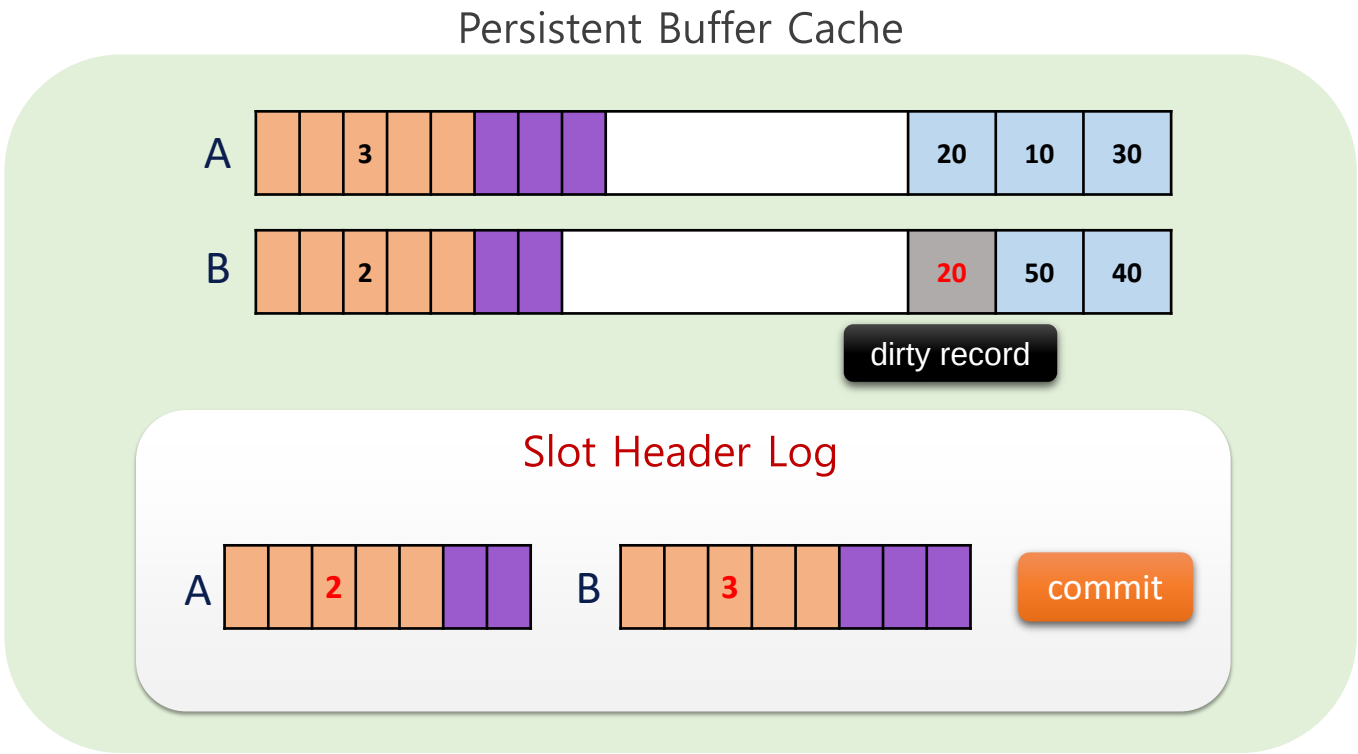
- **Failure-Atomic Slot-Header Logging Scheme**
  - What if a DB transaction modifies multiple pages?
  - The Failure-Atomic In-place Commit scheme works only for a single page modification.
  - ➢ Example: Move data with key 20 from page A to page B



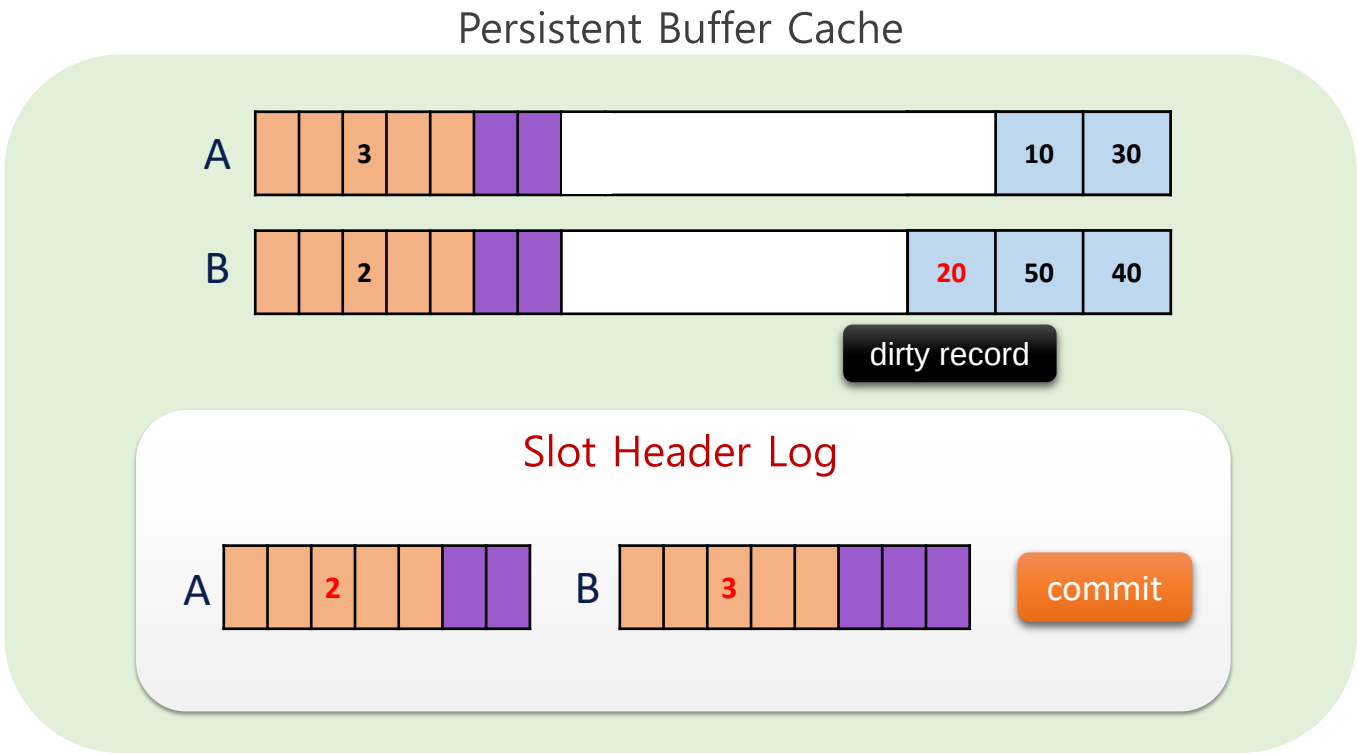
- **Failure-Atomic Slot-Header Logging Scheme**
  - What if a DB transaction modifies multiple pages?
  - The Failure-Atomic In-place Commit scheme works only for a single page modification.
  - ➢ Example: Move data with key 20 from page A to page B



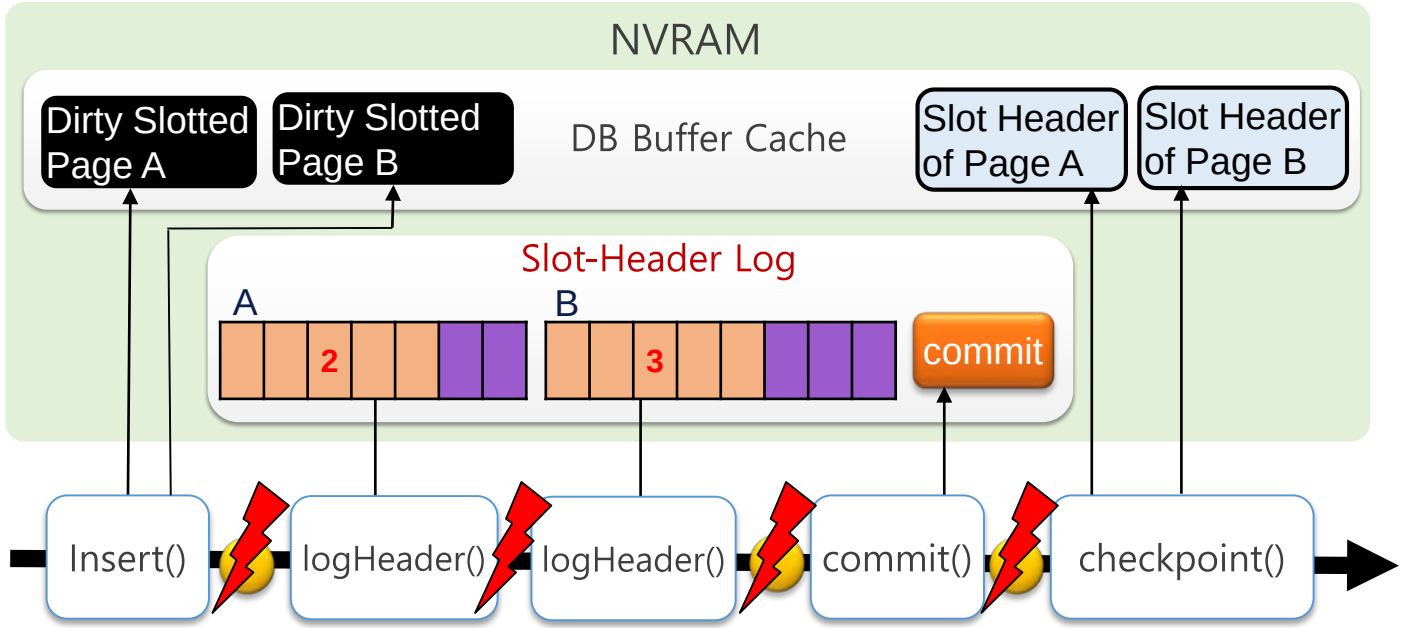
- **Failure-Atomic Slot-Header Logging Scheme**
  - Even with RTM, multiple slot headers cannot be atomically updated.
  - RTM is available only in high-end Xeon CPUs.
  - Logging seems inevitable in database transactions. Therefore, we propose "Slot-Header Logging" that logs only slot-headers but not records.



- **Failure-Atomic Slot-Header Logging Scheme**
  - Even with RTM, multiple slot headers cannot be atomically updated.
  - RTM is available only in high-end Xeon CPUs.
  - Logging seems inevitable in database transactions. Therefore, we propose "Slot-Header Logging" that logs only slot-headers but not records.



- **Failure-Atomic Slot-Header Logging Scheme**

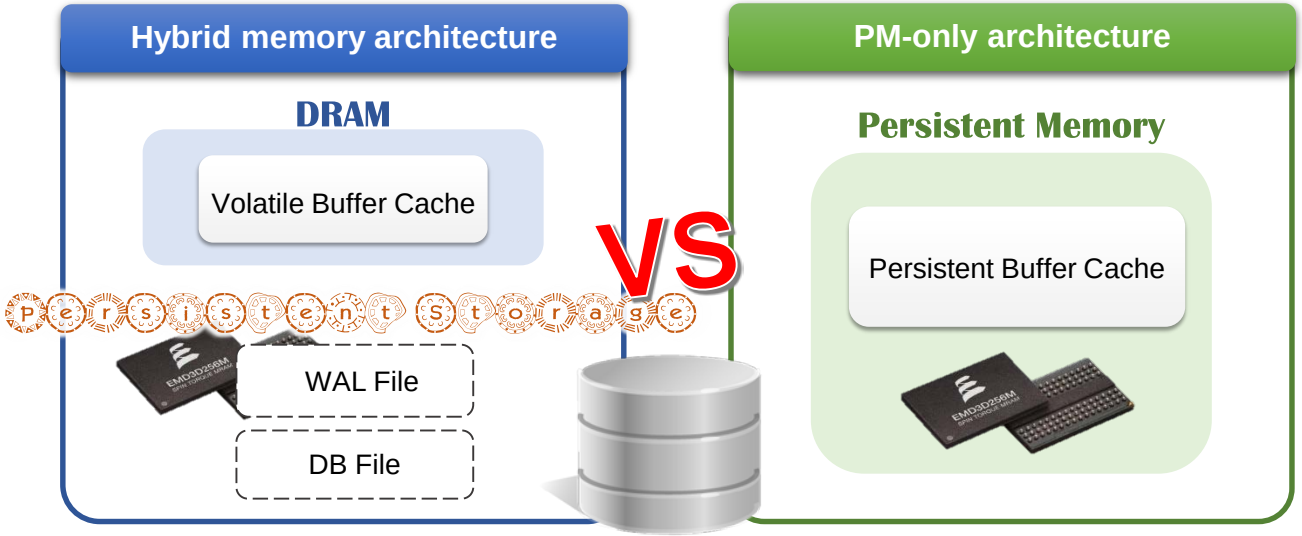


Experimental Environment

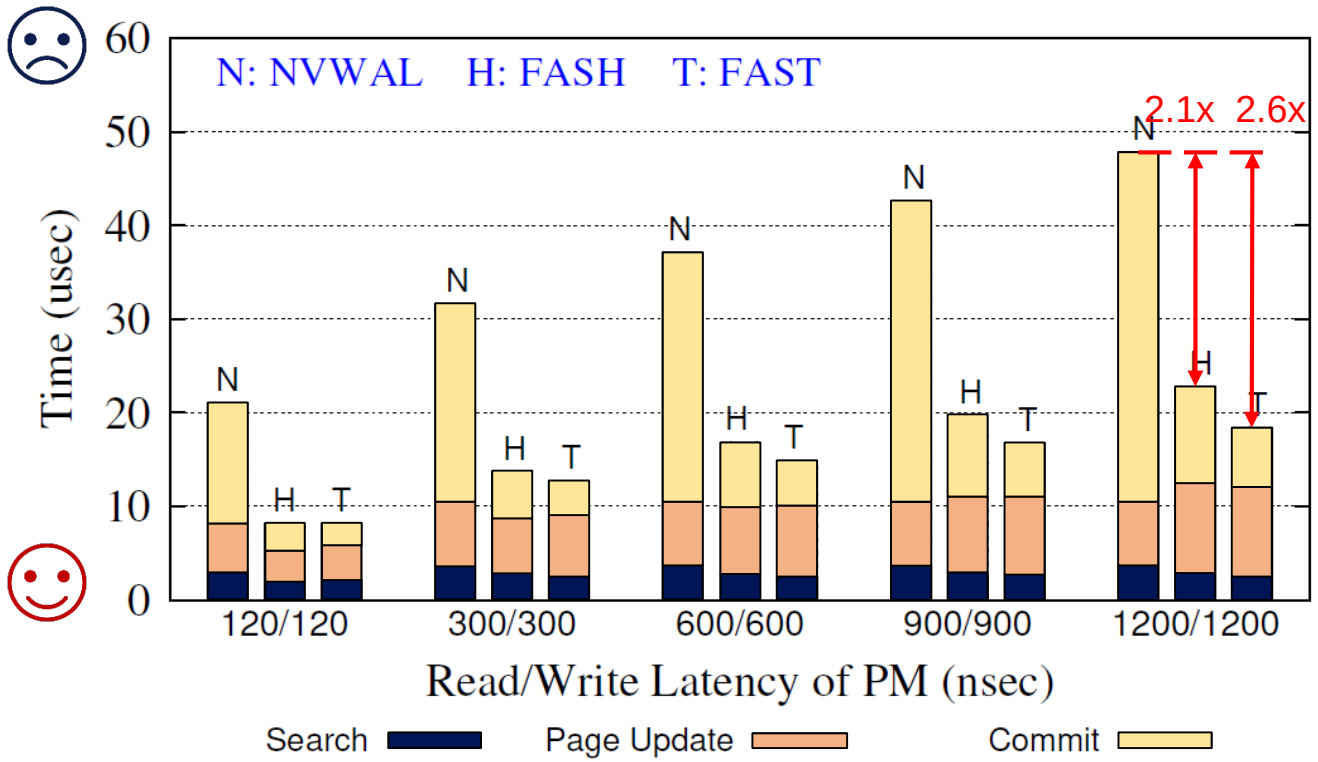
- **Experimental Setup**
  - Testbed
    - Intel Xeon Haswell-EX E7-8860 v3 (2.2GHz)
    - 256 GB DDR3 Memory
  - We implemented Failure-Atomic Slotted Paging in SQLite 3.8
    - FASH : **F**ailure-**A**tomic **S**lot-**H**eader logging
      - Slot-Header Logging always
    - FAST : **F**ailure-**A**tomic **S**lot-header with in-place commi**T**
      - Single page updates use HTM(RTM)
      - Multiple page updates use Slot-Header Logging
  - FASH, FAST, NVWAL
    - FAST and FAST proposed for use in PM-only architecture
    - NVWAL proposed for use in hybrid memory (DRAM+PM)
  - We used software persistent memory emulator - Quartz for read latency
    - For write latency, we injected additional delay after *clflush* instruction

- **FAST, FASH and NVWAL**

	NVWAL	FASH	FAST
Single page update	Differential logging	Slot-header logging	In-place commit
Multiple page update			Slot-header logging
Buffer cache	In DRAM	In PM	In PM
Log	In PM	In PM	In PM

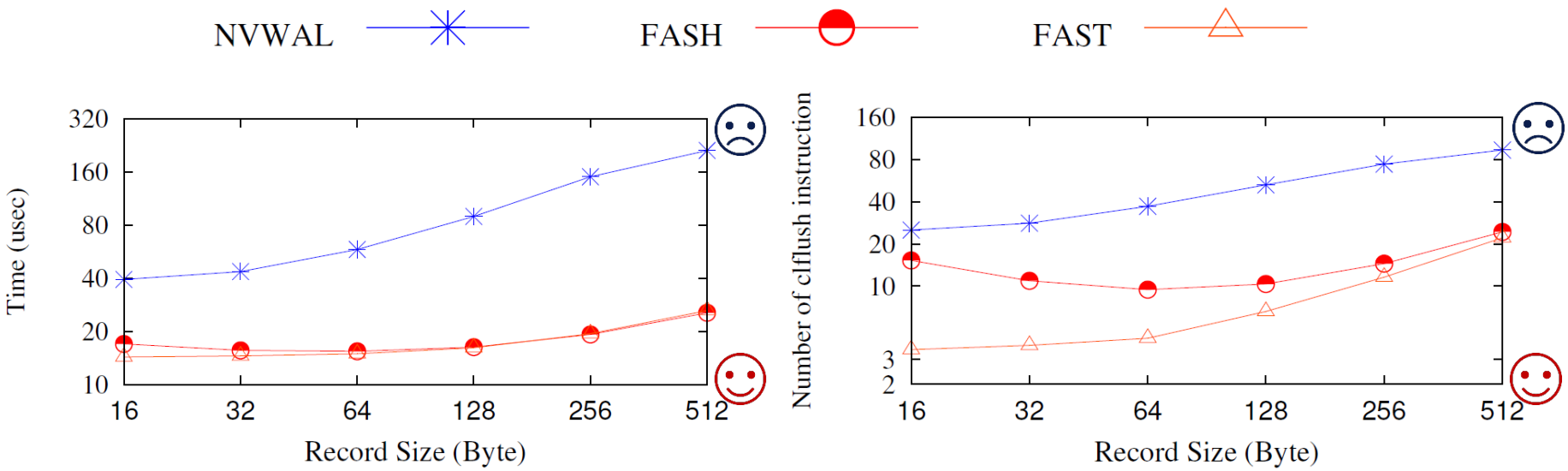


- **Breakdown of Time Spent for B-tree Insertion**



- Both failure-atomic slotted paging schemes outperform NVWAL
  - NVWAL incurs considerable commit overhead due to multiple copies
- FAST is faster than FASH
  - FAST doesn't have logging overhead if overflow or underflow does not happen

Insertion Performance



- FAST and FASH consistently outperform NVWAL
  - FAST and FASH do not duplicate write operations for records
  - NVWAL generates large log frames for large records
- FASH calls more clflush instructions for small record sizes
  - The reason is that with smaller records, the slotted-page can hold more records
- FAST calls about 3 clflush instructions when the record is smaller than 64 bytes
  - The slot-header size of FAST must be less than 64bytes.

Conclusions

- "Failure-atomic slotted paging scheme" eliminates the necessity of redundant copies by integrating logging into database buffer caching.
- PM-only memory systems can perform faster than hybrid memory systems that consist of both PM and DRAM
- Even with a small PM, we can significantly reduce IO traffic via Slot-Header Journaling.


Q
&
A