MichiganEngineering

High Performance Transaction Processing for NVRAM

Steven Pelley, Thomas F. Wenisch

University of Michigan

MichiganEngineering

Nonvolatile memory (NVRAM)

- New NVRAMs provide fast durable storage (phase change, memristor, STT-RAM)

Storage technology	Random read latency	Durable?
Disk	10ms	✓
Flash	90µs	✓
DRAM	100ns	✗
NVRAM	50-1000ns [IBM]	✓

*NVRAM enables **persistent** main memory*

Recoverable systems

- Must protect data against server failure
  - E.g., file systems and databases
  - Typically rely on disk, flash for persistent storage
- NVRAM accelerates persistent storage access
  - Byte-addressable (native memory instructions) further decrease execution overheads
- Create recoverable, high performance data structures (main-memory/DRAM speed)

NVRAM requires new recovery mechanisms

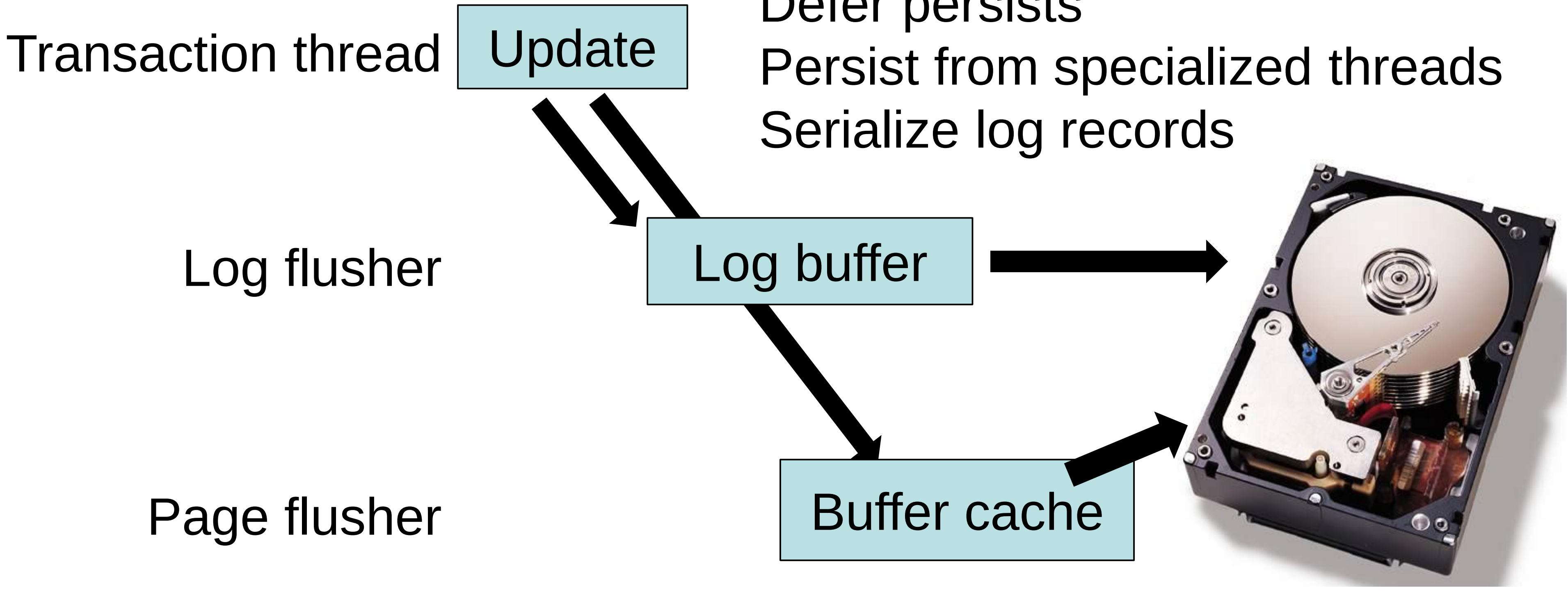
Optimizing NVRAM

- Case study of transaction processing [VLDB]
  - NVRAM as a disk replacement (ARIES/WAL)
  - Fast NVRAM imposes frequent persist ordering
  - New software design to avoid ordering delays
- Future work: explore desirable optimizations and semantics for NVRAM memory systems

NVRAM as a disk replacement

Write Ahead Logging (WAL) via ARIES

Defer persists
Persist from specialized threads
Serialize log records

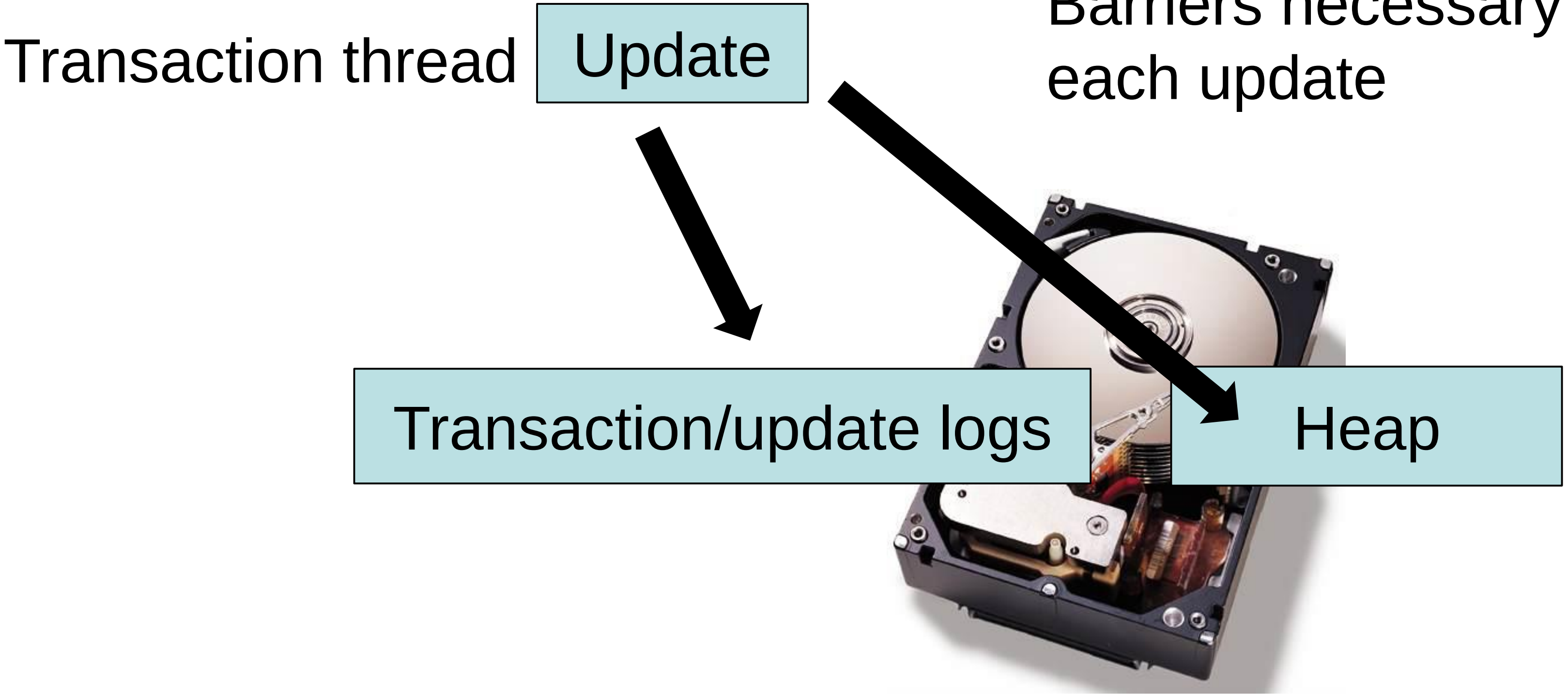


Complexity necessary for slow disk now excessive overhead

Alternative: order persists immediately

- Persist barriers [BPFS: SOSP ‘09]
  - Persists following barrier instruction may not complete before persists prior to barrier
  - Persists that are not separated by barrier instruction are *concurrent*
- Barriers delay execution
  - Frequency of barriers and *exposed barrier latency* determine resulting performance

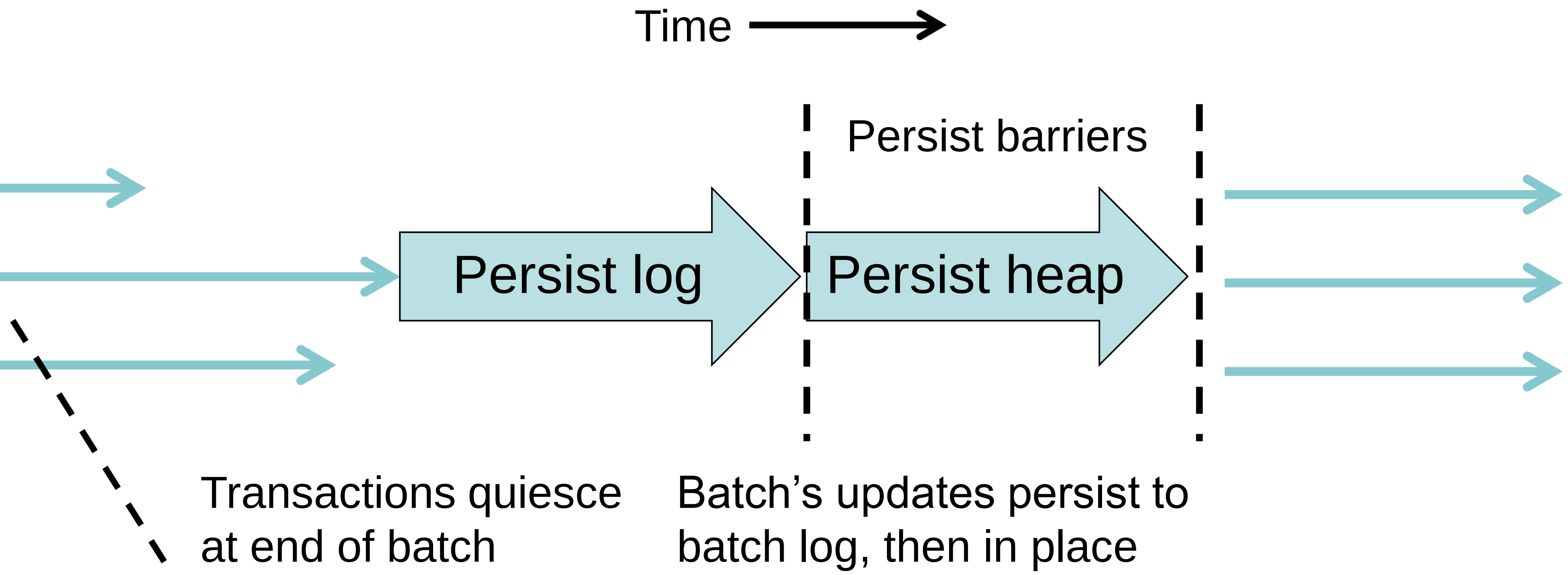
NVRAM in-place updates



Persist immediately
Remove centralized log
Barriers necessary at each update

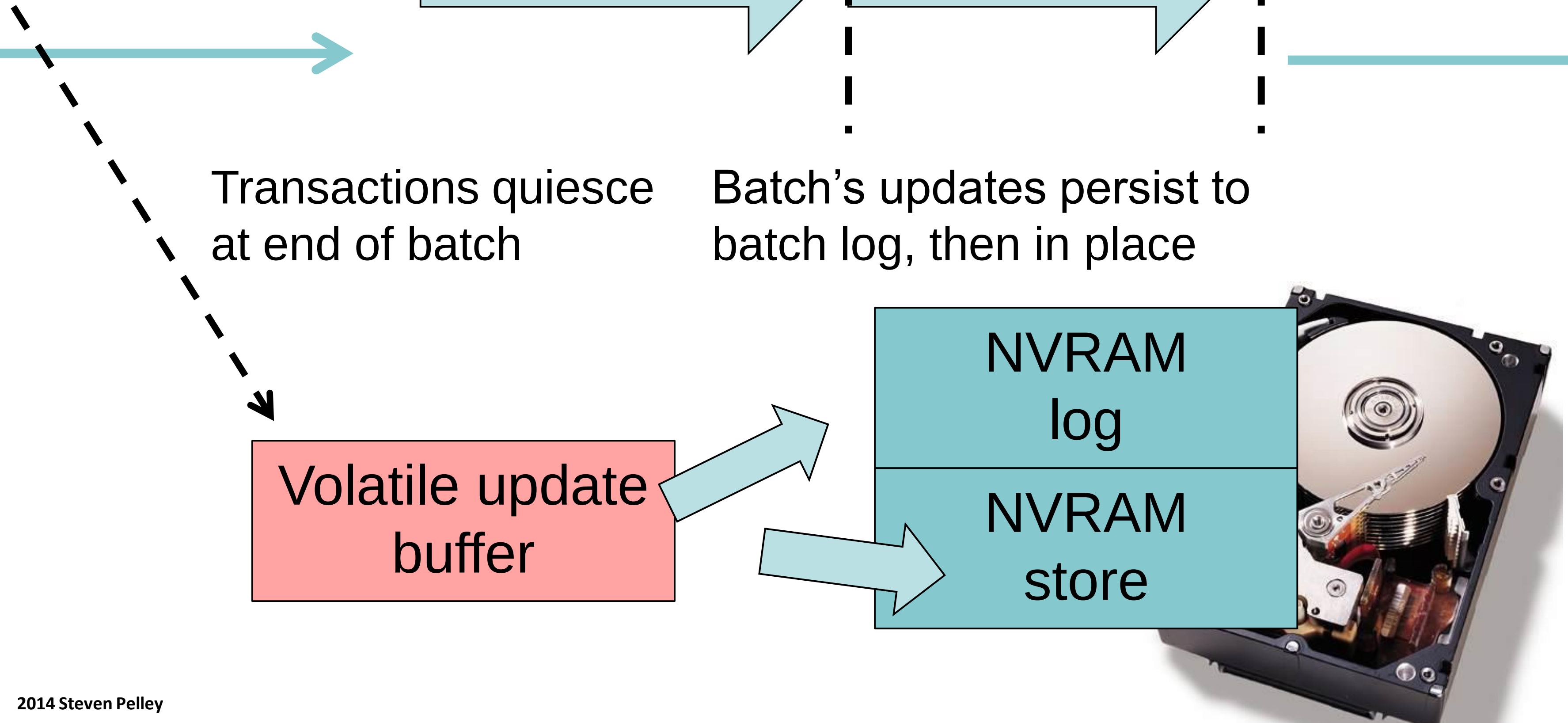
Simple, removes software overhead, but frequent persist ordering

NVRAM Group Commit

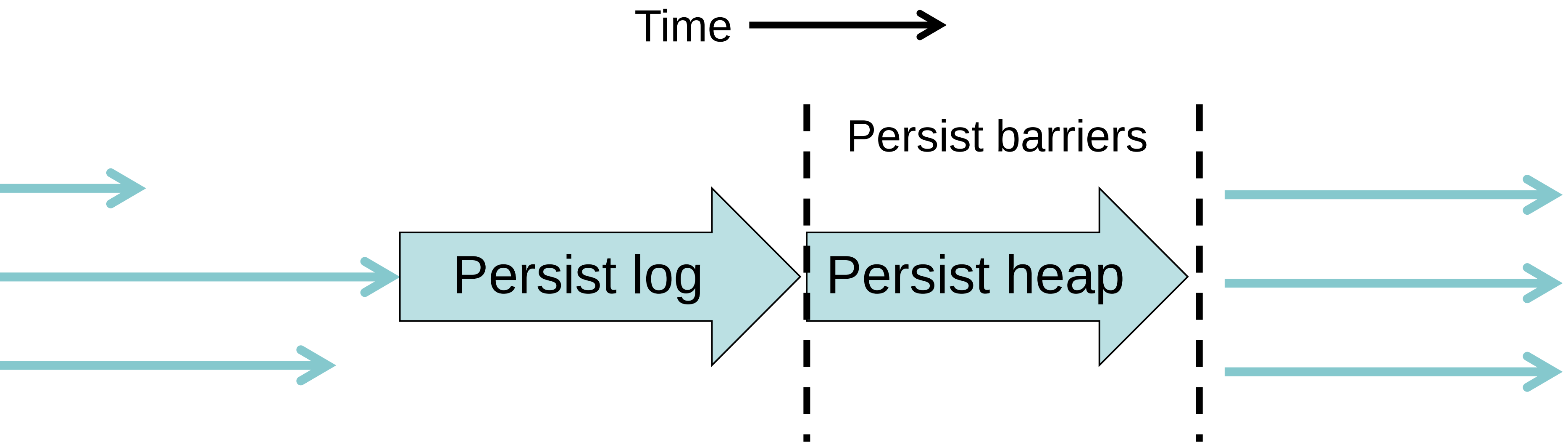


Transactions quiesce at end of batch

Batch's updates persist to batch log, then in place



NVRAM Group Commit



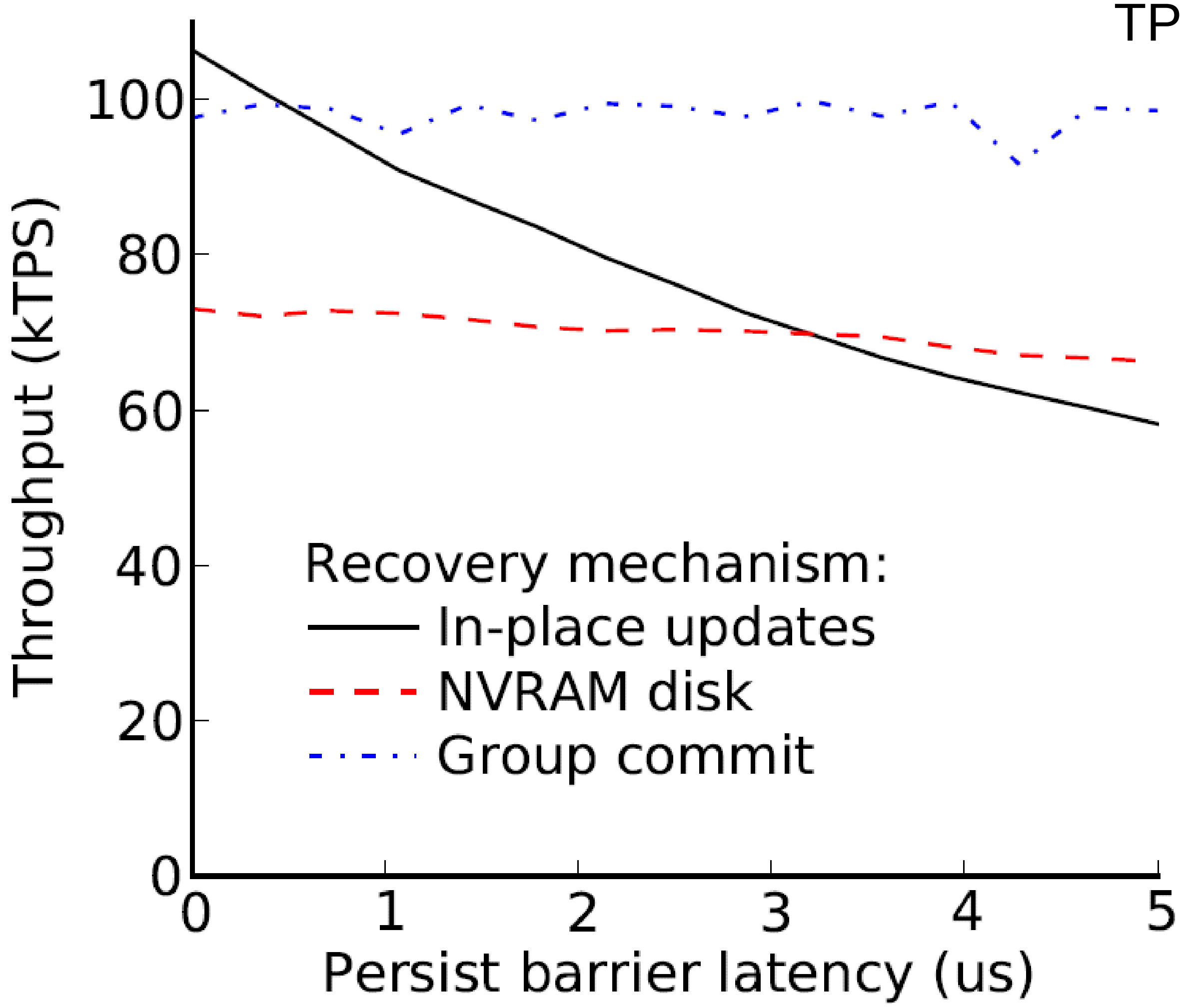
Constant number of persist barriers per batch
No centralized logging
Throughput high so long as batch quiesce/persist short
Several other tradeoffs (e.g., increased transaction latency)

Modeling unavailable devices

- Run database on *real hardware*
  - Log and db heap on RAMDisk (or just in DRAM)
  - Introduce precise delays (20ns precision using x86 RDTSCP) to model persist barrier latency
- Build recovery mechanisms in software
  - Shore-MT: research platform for high performance transaction processing
  - Rely on dirty bit fields to track buffer pool writes during transaction, page latch, or batch

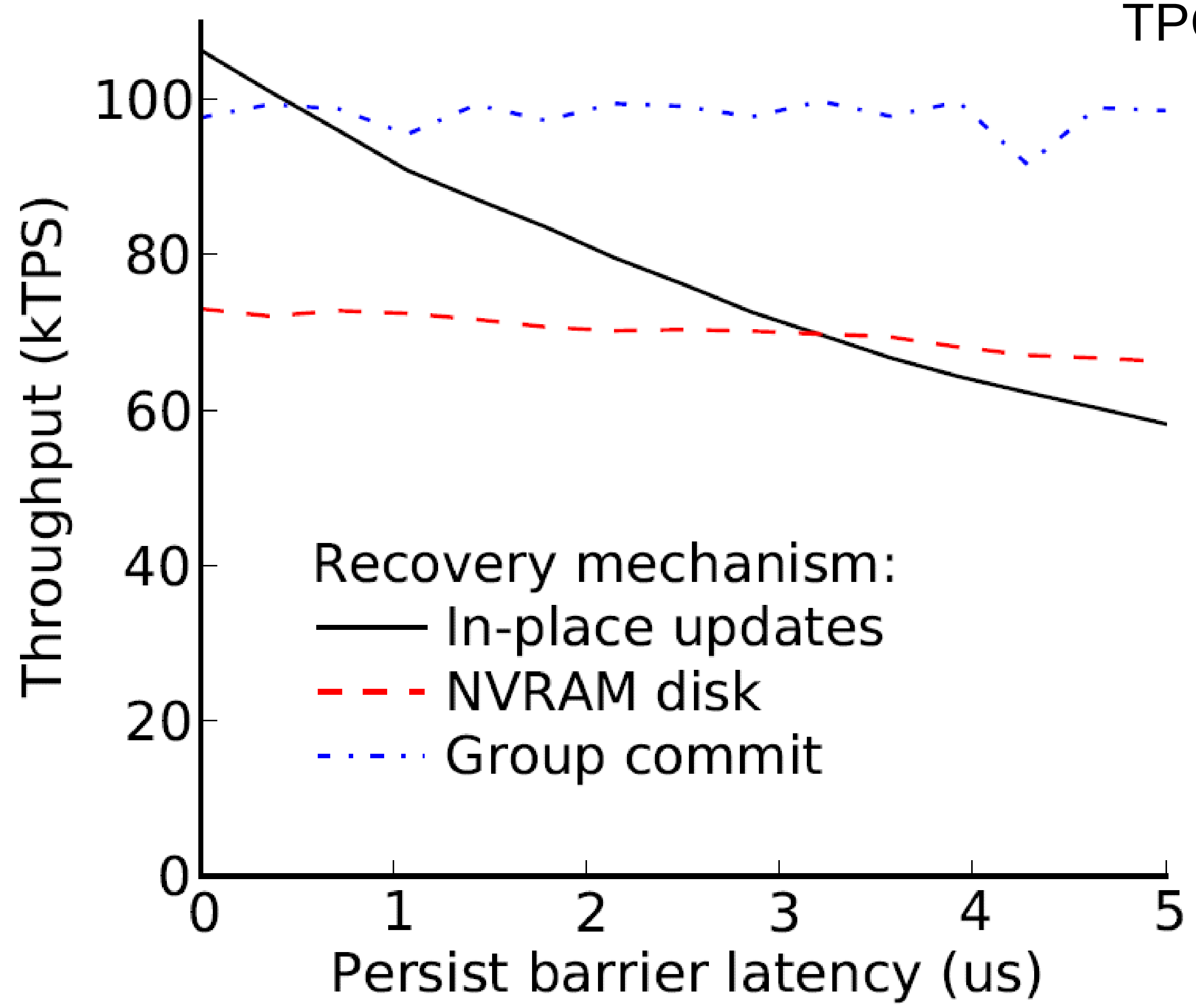
Recovery management performance

TPCB benchmark



Recovery management performance

TPCB benchmark

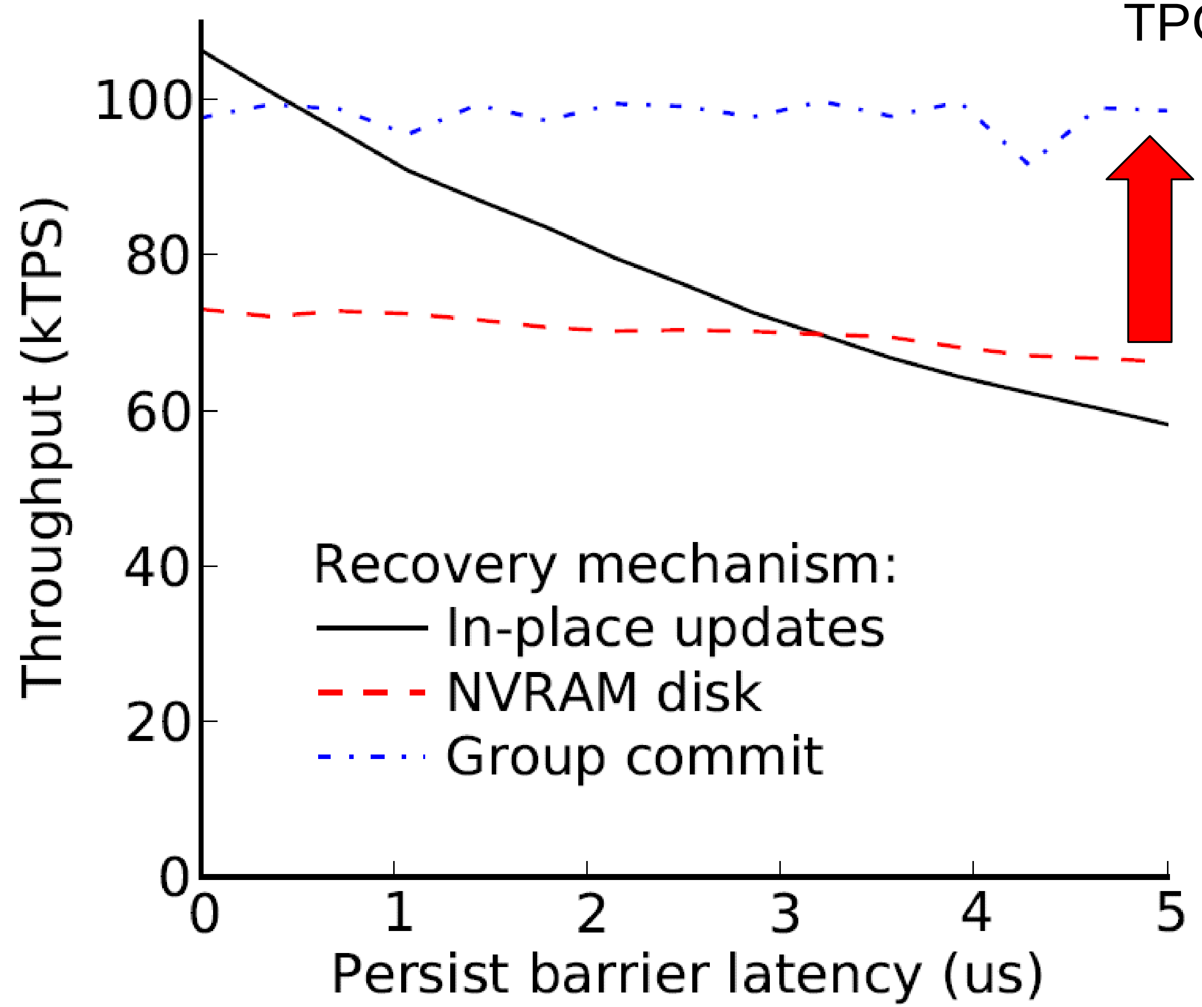


Disk replacement insensitive to latency, but slow

MichiganEngineering

Recovery management performance

TPCB benchmark

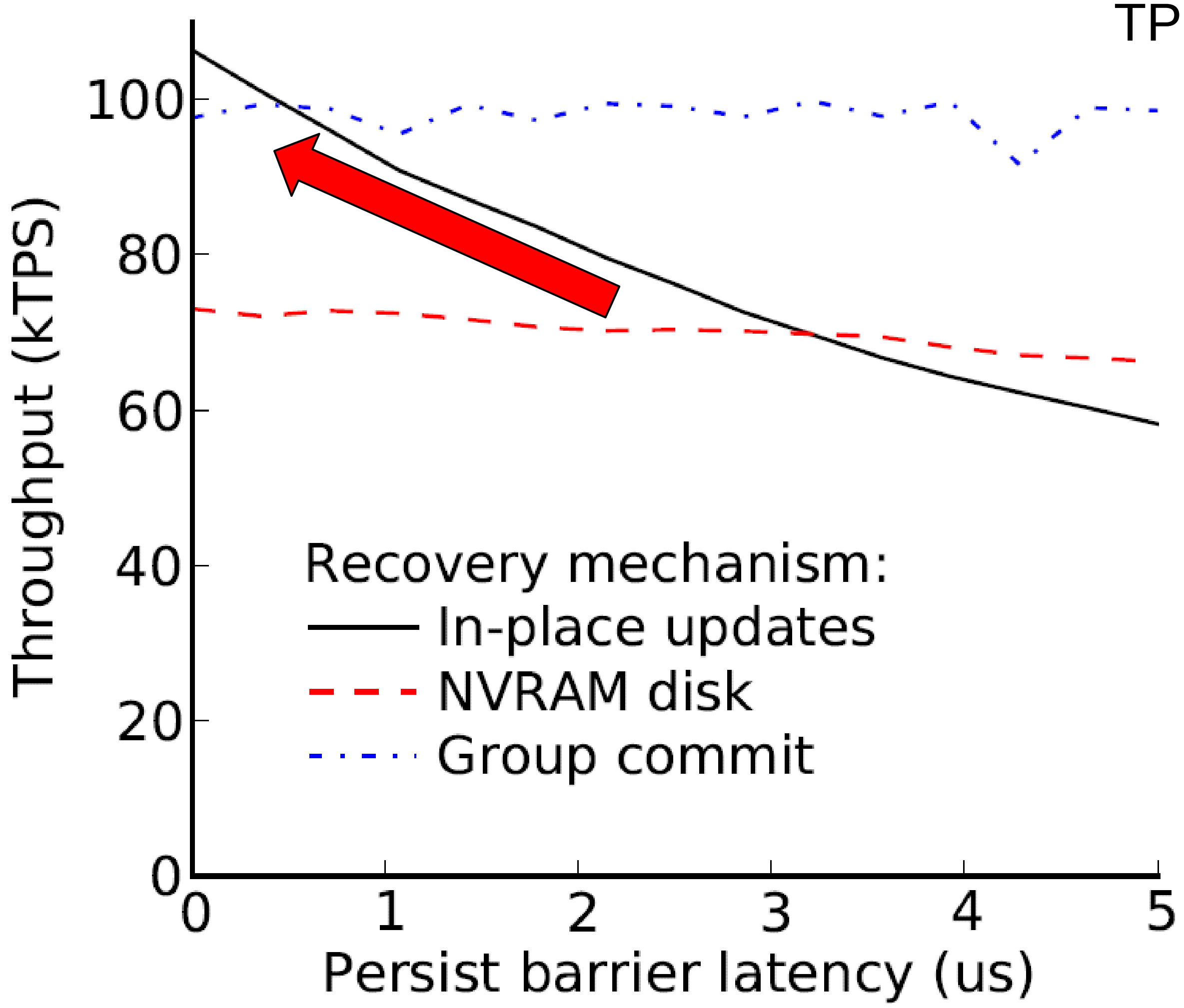


Group commit recovers throughput when barrier latency exposed

MichiganEngineering

Recovery management performance

TPCB benchmark



High performance barriers and memory system ideal

In the VLDB paper…

- Implementation and methodology details
- Persist bandwidth modeling
- Group commit trade-offs and transaction latency analysis
- Read-cache performance based on memory trace analysis

Future work: NVRAM optimizations

- Asynchronous persists
  - Allow execution to proceed ahead of persists
  - Hides persist and persist barrier latency
  - *Persist ordering* critical path limits persist rate
- Persist coalescing
  - Omit persists if ordering constraints not violated
  - Acts as bandwidth filter (like write-back cache)
  - Reduces persist ordering critical path

*Need mechanisms to precisely label persist ordering constraints [upcoming ISCA: **Memory Persistency**]*

Conclusion

- Disk-based software carries baggage
- Frequent persist synchronization also slow
- New software and memory systems improve performance and simplify software design
- References
  - Website: http://web.eecs.umich.edu/~spelley
  - Steven Pelley, Thomas F. Wenisch, Brian T. Gold, Bill Bridge: Storage Management in the NVRAM Era. PVLDB 7(2): 121-132 (2013)
  - [BPFS] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. 2009. Better I/O through byte-addressable, persistent memory. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles* (SOSP '09)