FOEDUS: OLTP Engine for 1,000 Cores & NVRAM

Hideaki Kimura <hideaki.kimura@hp.com>
HP Labs

At SIGMOD 2015

Talk is Cheap. ***SHOW ME THE CODE.***

– Linus Torvalds

“Sadly, DB people don't seem to have any understanding of good taste.”

“[Database is] the least interesting thing in the computing world.”

Conclusion: *Read* and *Run* the Code!

http://github.com/hkimura/foedus

- ✓From-Scratch, Open-Source, Fully ACID/Serializable Database Kernel in C++
- ✓Orders of Magnitude Faster on *Next-Gen. H/W:*
  - 100s~ of Cores, 10s~ of Sockets
  - TBs~ of NVRAM or Enterprise SSD
- ✓Interested? *Talk to us, Join us, Challenge us!*

Next-Generation Server Hardware?

HP
The Machine

UC Berkeley
Firebox

Intel
Rack Scale
Architecture

...

Differences

- HP Photonics? Infiniband? QPI?
- HP Memristor? PCM? Flash?
- ...

Commonalities

- **1,000s of CPU Cores**
- **Fast Interconnect**
- **Huge Low-Latency NVRAM**
- **Endurance will be an issue**

FOEDUS Key Principles

1. Bring in-memory speed to NVRAM

SILO-like Lightweight Optimistic Concurrency Control

2. Dual-Page: physically independent, logically equivalent

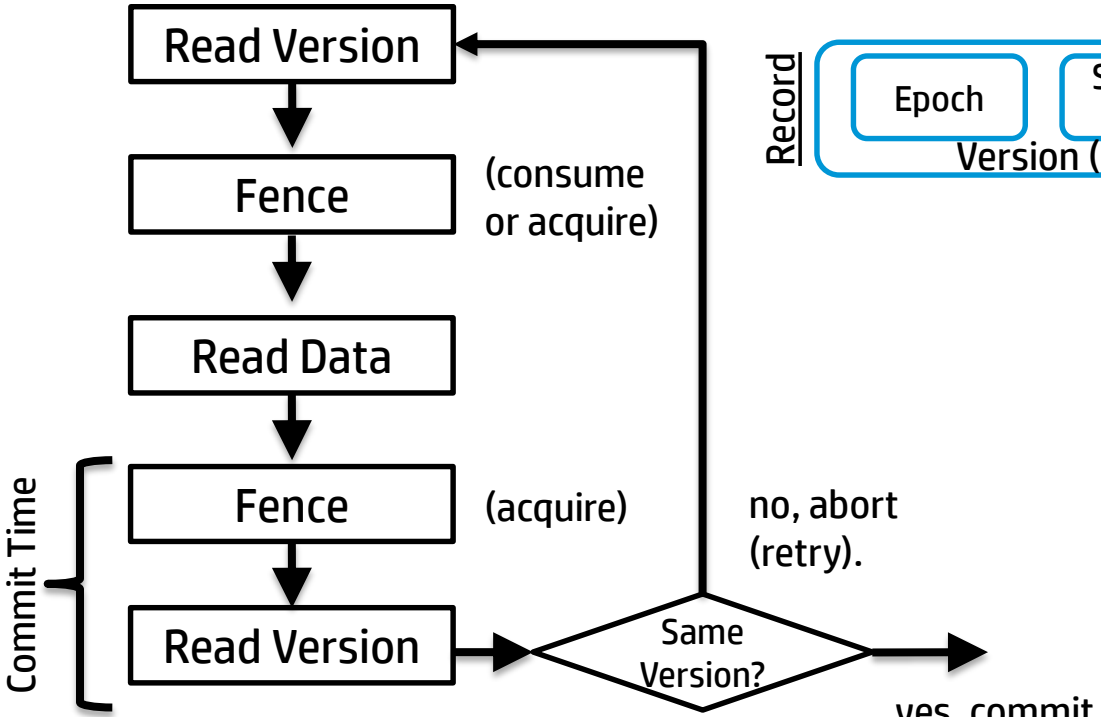
- ✓ Mutable Volatile Pages in DRAM
- ✓ Immutable Snapshot Pages in NVRAM
- ✓ Log-Gleaner t[Skipped Today] sync

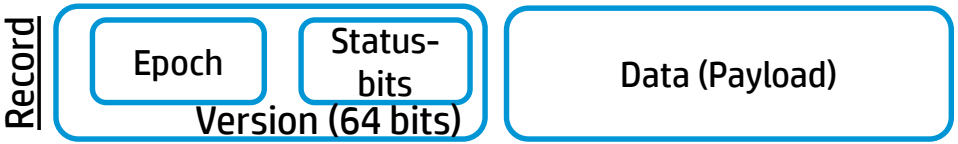
3. Master-Tree: Simple and Scalable OCC for NVRAM

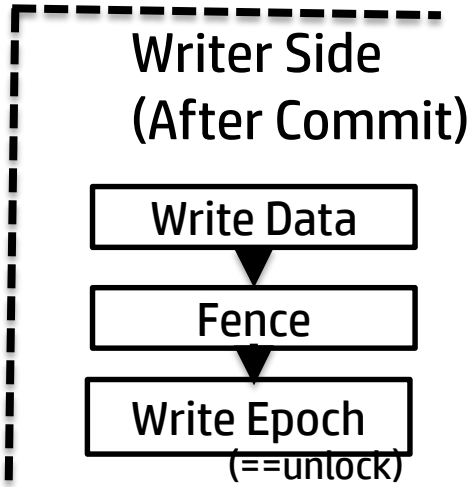
[Skipped Today]

Masstree + Foster B-Tree + *Foster-Twin*

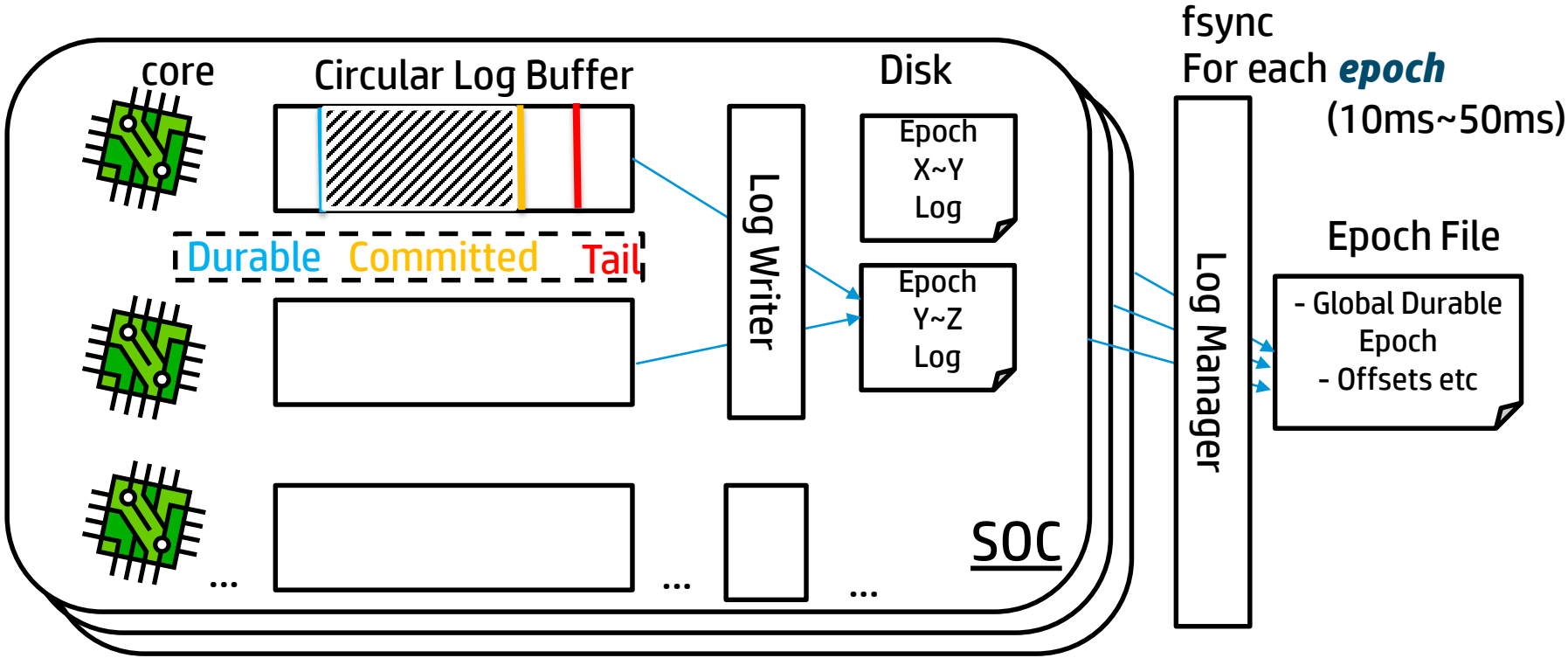
SILO: Lightweight/Decentralized OCC [Tu et al]







SILO: Decentralized Logger with Epoch [Tu et al]



SILO's OCC Benefits

✓ Extremely Lightweight and Scalable

Block-free (lock-free with retries)

No Write for Reads (cf., read-lock)

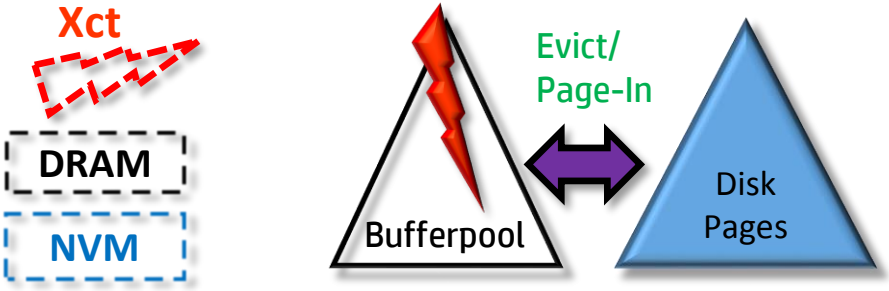
✓ In-page Lock Mechanism

No centralized lock manager

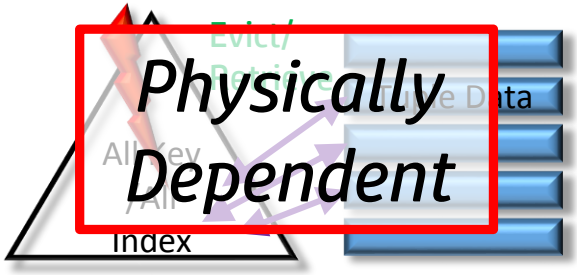
Cache Friendly

But, how can we go beyond DRAM?

In-Memory DBMS Beyond DRAM



a) Traditional Databases



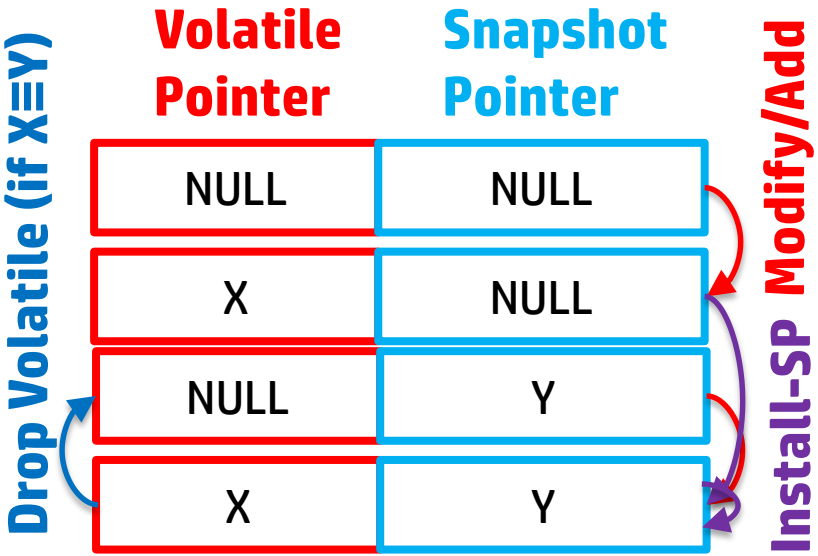
b) H-Store/Anti-Cache [Pavlo et al.]

c) Hekaton/Siberia [Larson et al.]

d) FOEDUS

Logically Equivalent, Physically Independent Dual

- **Volatile Page: Mutable, on DRAM**
- **Snapshot Page: Immutable, on NVRAM**



: nullptr

: Volatile-Page is just made. No Snapshot yet.

: Snapshot-Page is latest. No modification.

: X is the latest truth, but Y *may* be equivalent.

Dual Pages: Benefits

✓ **Snapshot Pages are Immutable**

Drastically simplifies Caching/Replication/Traversal/Commit/etc

✓ **Physically Independent**

Transactions never interfere w/ construction/retrieval/eviction of Snapshot Pages

✓ **Logically Equivalent**

- No Bloom Filter or global data needed for Serializability ↔ LSM-Trees
- Volatile Pages guaranteed to contain latest data
- Snapshot Pages guaranteed to be complete *as of snapshot-epoch*

Experiments

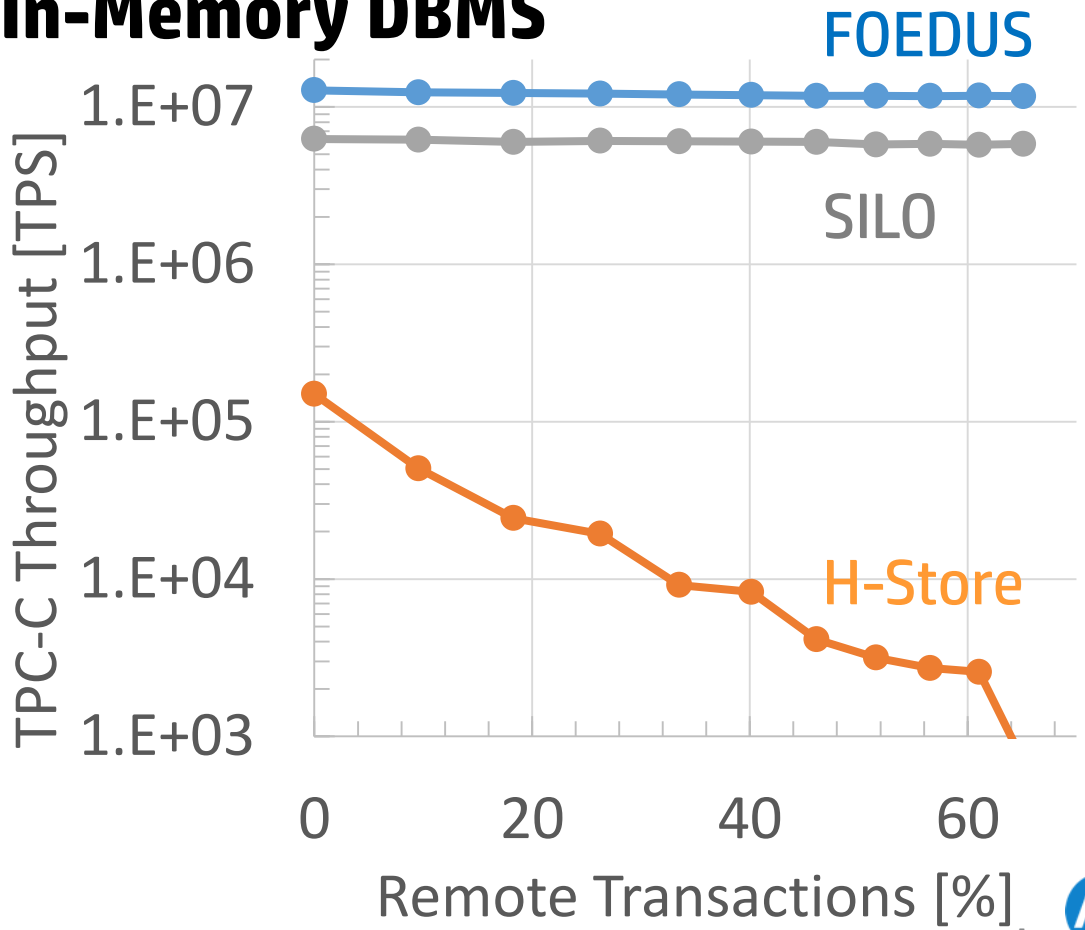
- **TPC-C, Serialized**
- **vs. H-Store, SILO, Shore-MT**
- (Traditional DBs (e.g., MySQL) are too slow to compare with.)
- **HP Superdome X (DragonHawk):**
  16 sockets, 240 cores, 12TB DRAM.
- **Emulated NVRAM**

Experiment 1: vs In-Memory DBMS

Observations

1. SILO-style OCC much more resilient to contention
2. **100x~** faster than H-Store.
3. More Cores = More Speedup

Cores	Speed-up [/H-Store]
16	33x
60	66x
240	394x



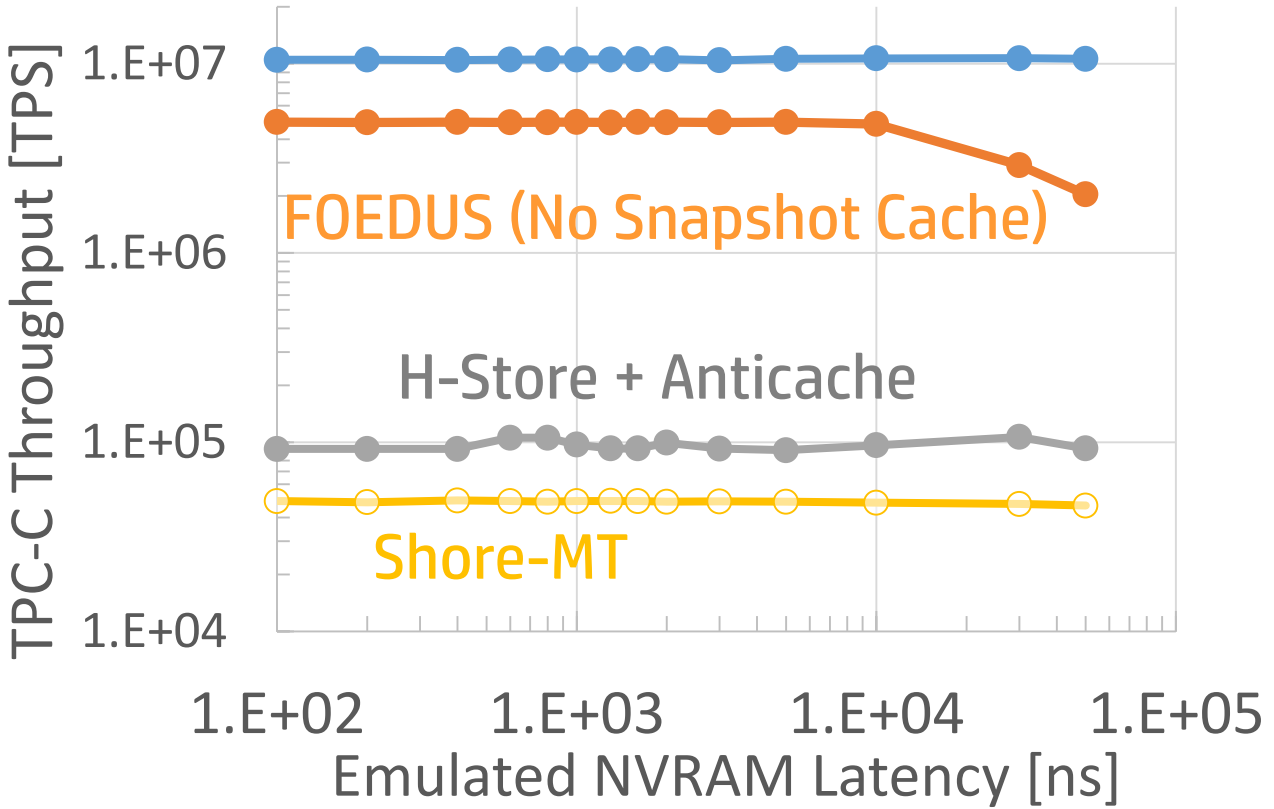
Experiment 2: DBMS on NVRAM (emulated)

<u>Environments</u>

- tmpfs-based NVRAM emulation (varied latency)
- Data/xlog in NVRAM
- H-Store w/ anti-cache

<u>Observations</u>

1. FOEDUS **100x**~ faster
2. Performs best when NVRAM read **<10us**



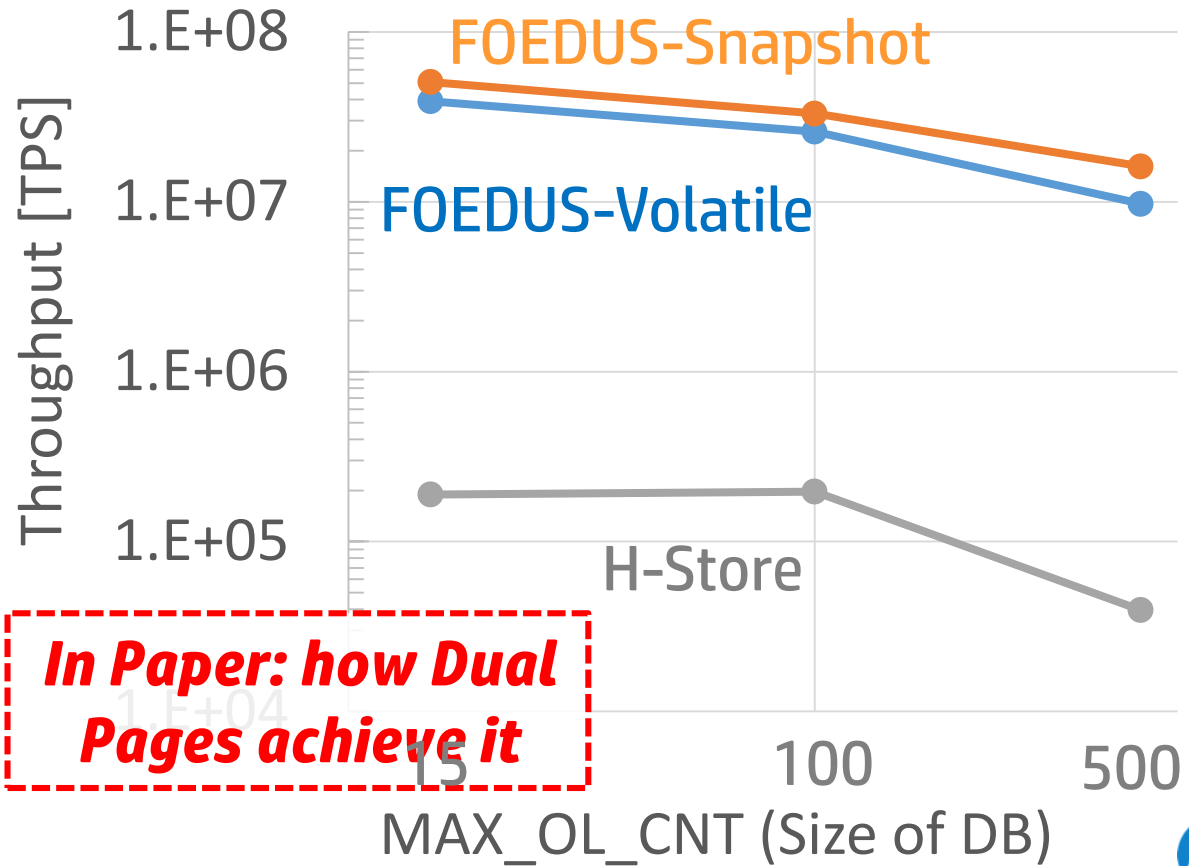
Experiment 3: OLAP Workload

Workload

- 30x Larger Data
- Cursor-Heavy
- Read-Only
- Volatile Pages Only vs Snapshot Pages Only

Observations

1. 100x~ faster than H-Store.
2. FOEDUS runs **faster** when database is cold (!!)



In Paper: how Dual Pages achieve it

Ongoing Work

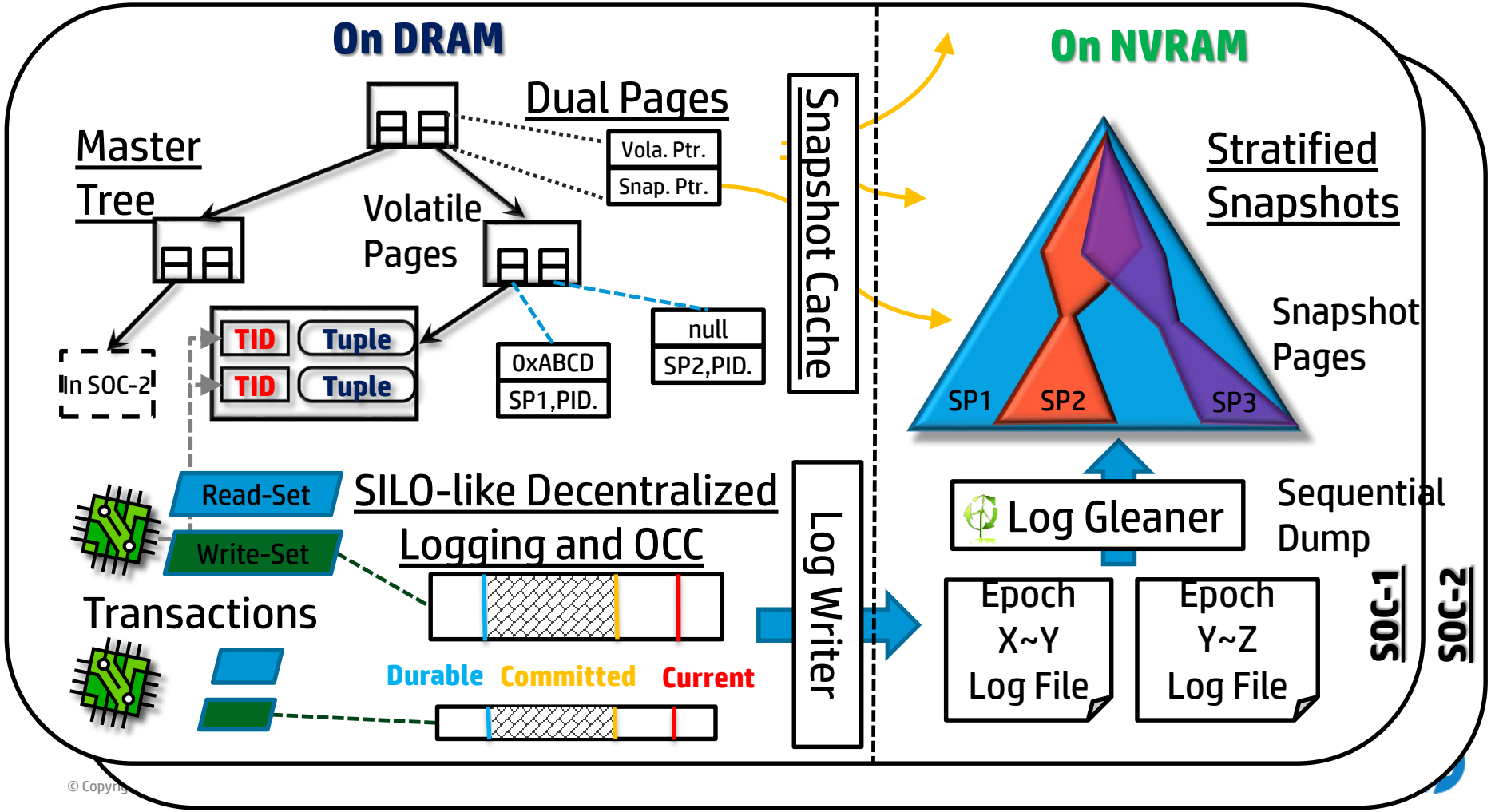
- ✓Try on 1,000 Cores~
- ✓Further Resilience to Contentions

  Combining Pessimistic Approach When Beneficial
- ✓SIMD, Xeon-Phi

  Log-Gleaner's Sorting, Page Construction, etc
- ✓Non-C++ Interface

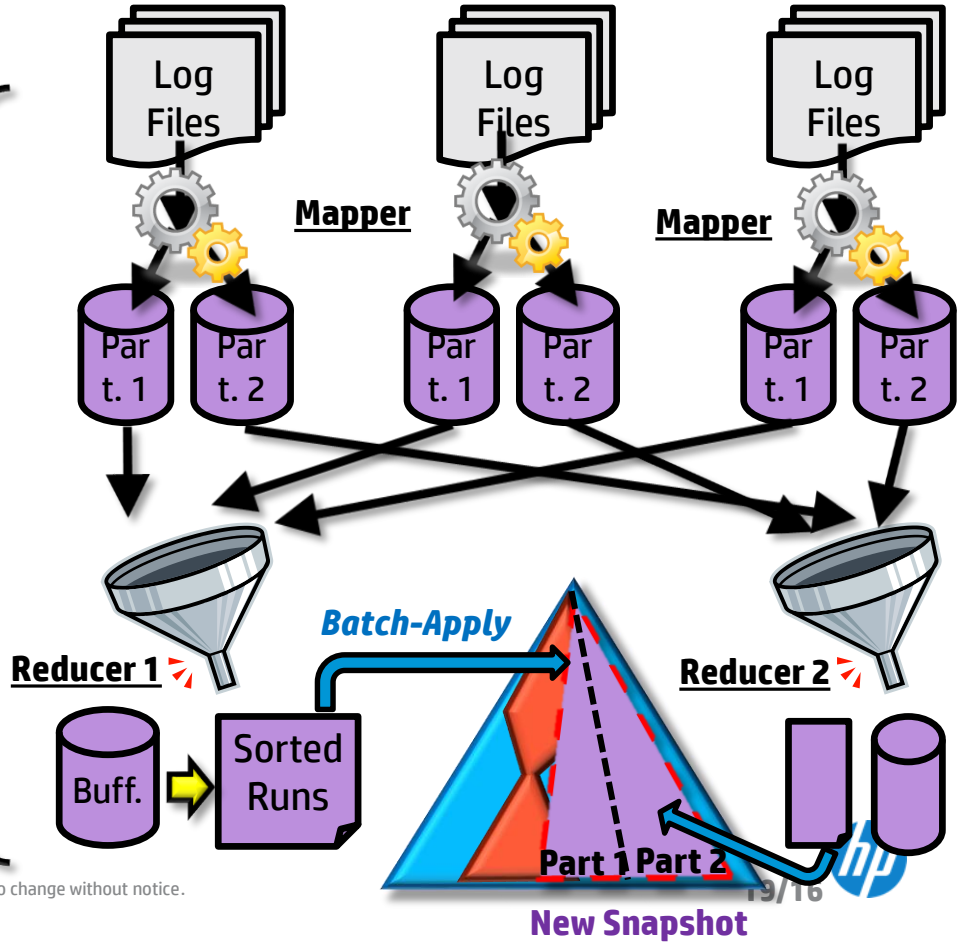
  SQL/JDBC/ODBC for OLAP. But, for OLTP???

Reserved Slides


On DRAM
On NVRAM
Dual Pages
Vola. Ptr.
Snap. Ptr.
Master Tree
Volatile Pages
Snapshot Cache
Stratified Snapshots
Snapshot Pages
SP1
SP2
SP3
In SOC-2
TID
Tuple
TID
Tuple
0xABCD
SP1,PID.
null
SP2,PID.
Read-Set
Write-Set
Transactions
SILO-like Decentralized Logging and OCC
Durable
Committed
Current
Log Writer
Log Gleaner
Sequential Dump
Epoch X~Y Log File
Epoch Y~Z Log File
SOC-1
SOC-2

Constructing Stratified Snapshot from Logical Logs

1. Assign Partitions
2. Run *Mappers/Reducers*
3. Combine Root/Metadata
4. Install/Drop Pointers



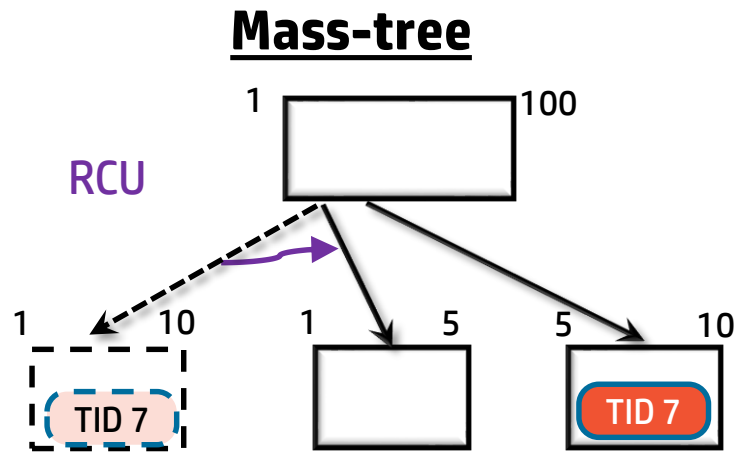
Benefits of Stratified Snapshots

- **Serializable transactions check only a single version of data ↔ LSM-Trees**
- **Drastically more scalable and efficient construction of NVRAM-resident data pages**
  - Separate from transactions/logging
  - Everything Batched, Processed in a tight loop
  - Large Sequential Write only. No frequent flush.

Master-Tree in a Nutshell

- **Mass-tree: *Cache Crafty B-tree* and *OCC* with *in-page* Lock Mechanism.**
- **Foster B-tree: *Single incoming pointer* via *foster-child* to ease page in/out.**
- ***Foster-Twins*: *Drastically Simplifies OCC and Reduces Aborts/Retries.***

OCC Problem 1: Unnecessary Retries/Aborts




```
findborder(node root, key k):
retry:    n ← root; v ← stableversion(n)
          if v.isroot is false:
              root ← root.parent; goto retry
descend:  if n is a border node:
              return ⟨n, v⟩
          n′ ← child of n containing k
          v′ ← stableversion(n′)
          if n.version ⊕ v ≤ "locked": // hand-over-hand validation
              n ← n′; v ← v′; goto descend
          v″ ← stableversion(n)
          if v″.vsplit ≠ v.vsplit:
              goto retry                    // if split, retry from root
          v ← v″; goto descend              // otherwise, retry from n
```

Local Retry!

Global Retry!

Figure 6. Find the border node containing a key.

Further, SILO must take Page-Version set and abort if changed by pre-commit.

Lots of Aborts!

OCC Problem 2: Difficult to make it right!

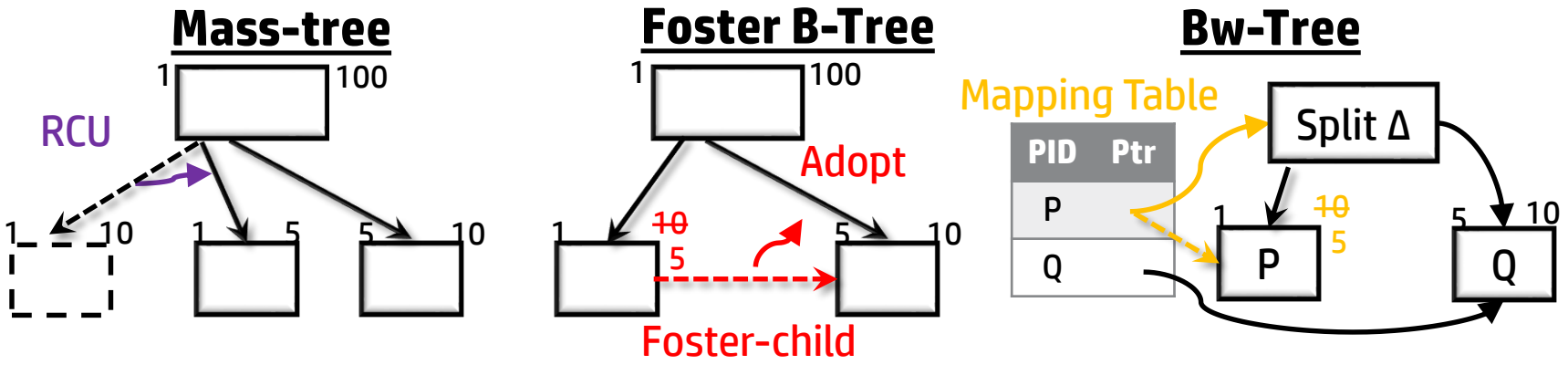
- Hand-over-hand Verification
- Complex Read/Write Procedure
- Memory Fences here'n there

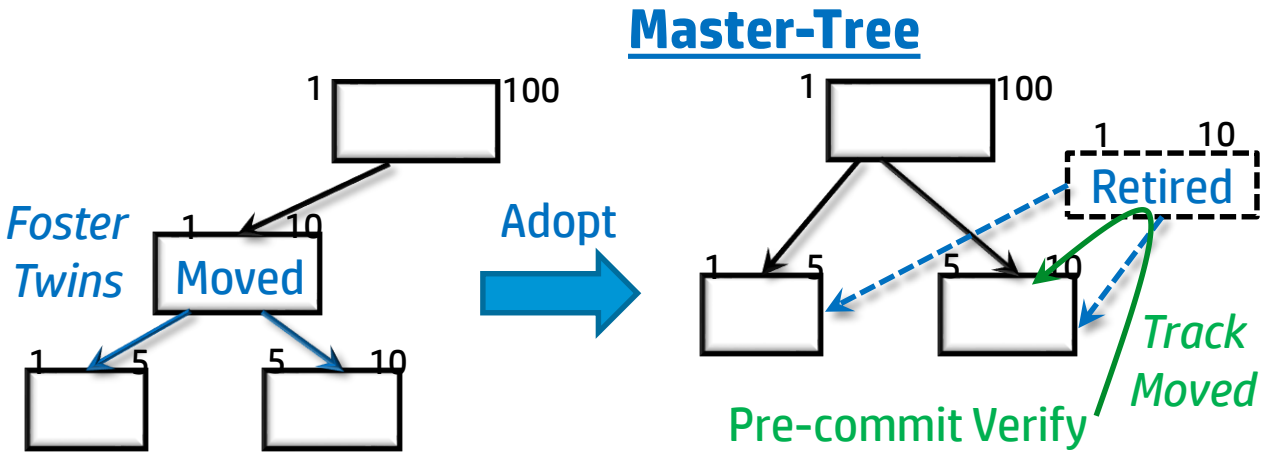
} ALL critical to correctness

You must be REALLY smart!

Foster Twins here to Save You!

- Strong Invariants to drastically simplify OCC and eliminate unnecessary aborts/retries
- Still everything in-page
  - No per-tuple GC or compaction/migration
  - No centralized data structure





- All data/metadata placed **in-page**
- **Immutable range**
- All information **always track-able**
- All invariants **recursively** hold

Foster-Twins Commit Protocol

Algorithm 1: SILO precommit protocol [13]

Input: R: Read set, W: Write set, N: Node set

`/* Precommit-lock-phase */`
Sort W by unique order;
foreach $w \in W$ **do** Lock w;

Fences, get commit epoch;
`/* Precommit-verify-phase */`
foreach $r, observed \in R$ **do** **if** $r.tid \neq observed$ *and* $r \notin W$ **then** abort;

foreach $n, observed \in N$ **do** **if** $n.version \neq observed$ **then** abort;

Generate TID, apply W, and publish log;

Algorithm 2: FOEDUS precommit protocol

Input: R: Read set, W: Write set, P: Pointer set

`/* Precommit-lock-phase */`
while *until all locks are acquired* **do**
  foreach $w \in W$ **do** **if** *w.tid.is-moved()* **then** w.tid ← track-moved(w.page, w.record)

  Sort W by unique order;
  foreach $w \in W$ **do** Try lock w. If we fail and find that w.tid.is-moved(), release all locks and retry
end

Fences, get commit epoch;
`/* Precommit-verify-phase */`
foreach $r, observed \in R$ **do**
  if *r.tid.is-moved()* **then** r.tid ← track-moved(r.page, r.record)
  if $r.tid \neq observed$ *and* $r \notin W$ **then** abort;
end
foreach $p \in P$ **do** **if** $p.volatile\text{-}ptr \neq null$ **then** abort;

Generate TID, apply W, and publish log;

Easy OCC with Foster Twins

❑Simple, Robust, and Efficient Search:

Find a pointer that *probably* contains the key;
Follow the page pointer;
If the page's *key-range* contain the key: go on;
Else: *locally* retry;

No hand-over-hand verification/split counters.
No unnecessary local retry.
Absolutely no global retry.

- No Page-Version-set nor unnecessary aborts: All Records/TIDs always track-able via Foster-Twin!