Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems

Joy Arulraj, Andrew Pavlo, Subramanya R. Dulloor

Non-Volatile Memory (NVM)

	DRAM	SSD	DISK	NVM
Read Latency	1x	500x	10^5x	2-4x
Write Latency	1x	5000x	10^5x	2-8x
Persistence	✗	✓	✓	✓
Byte-level access	✓	✗	✗	✓
Write endurance	✓	✗	✓	✗

Executive Summary

Design a DBMS storage engine for NVM

- *Re-examine traditional assumptions*
- *Storage and recovery optimizations*

NVM Hardware Emulator

- Configure DRAM load and store latency
- Throttle memory bandwidth
- Two interfaces
  - *Filesystem Interface (fread/fwrite)*
  - *Allocator Interface (malloc/free)*

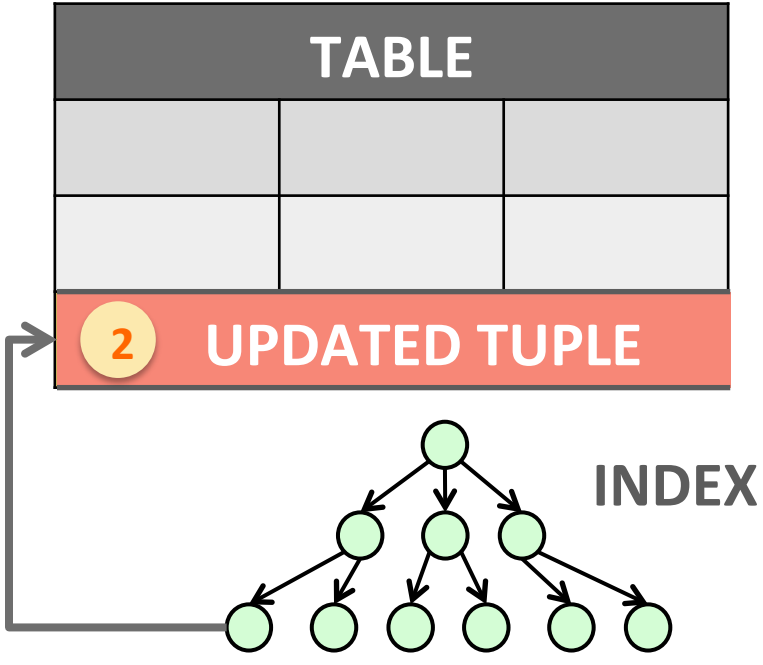
DBMS Platform

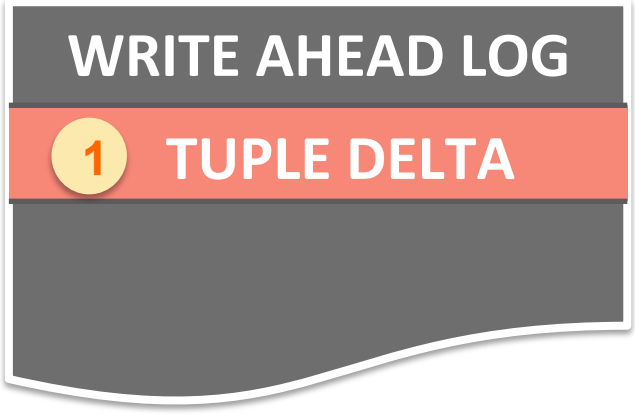
- Lightweight DBMS
  - *NVM-only design*
  - *No volatile DRAM*
  - *Runs on the hardware emulator*
- Pluggable backend storage architecture
- Timestamp-based concurrency control

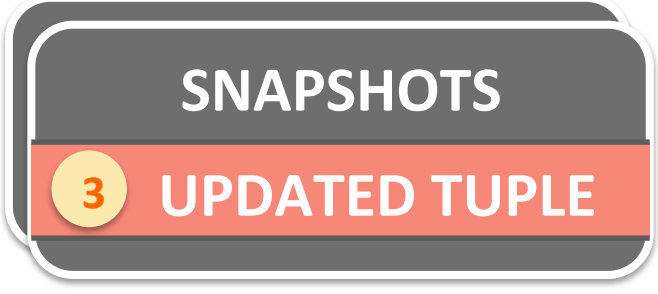
3 Storage Engines

ENGINE TYPE	TABLE STORAGE	LOGGING	EXAMPLE
In-Place Updates	✓	✓	VoltDB
Copy-on-Write Updates	✓	✗	LMDB
Log-Structured Updates	✗	✓	LevelDB

#1: In-Place Updates



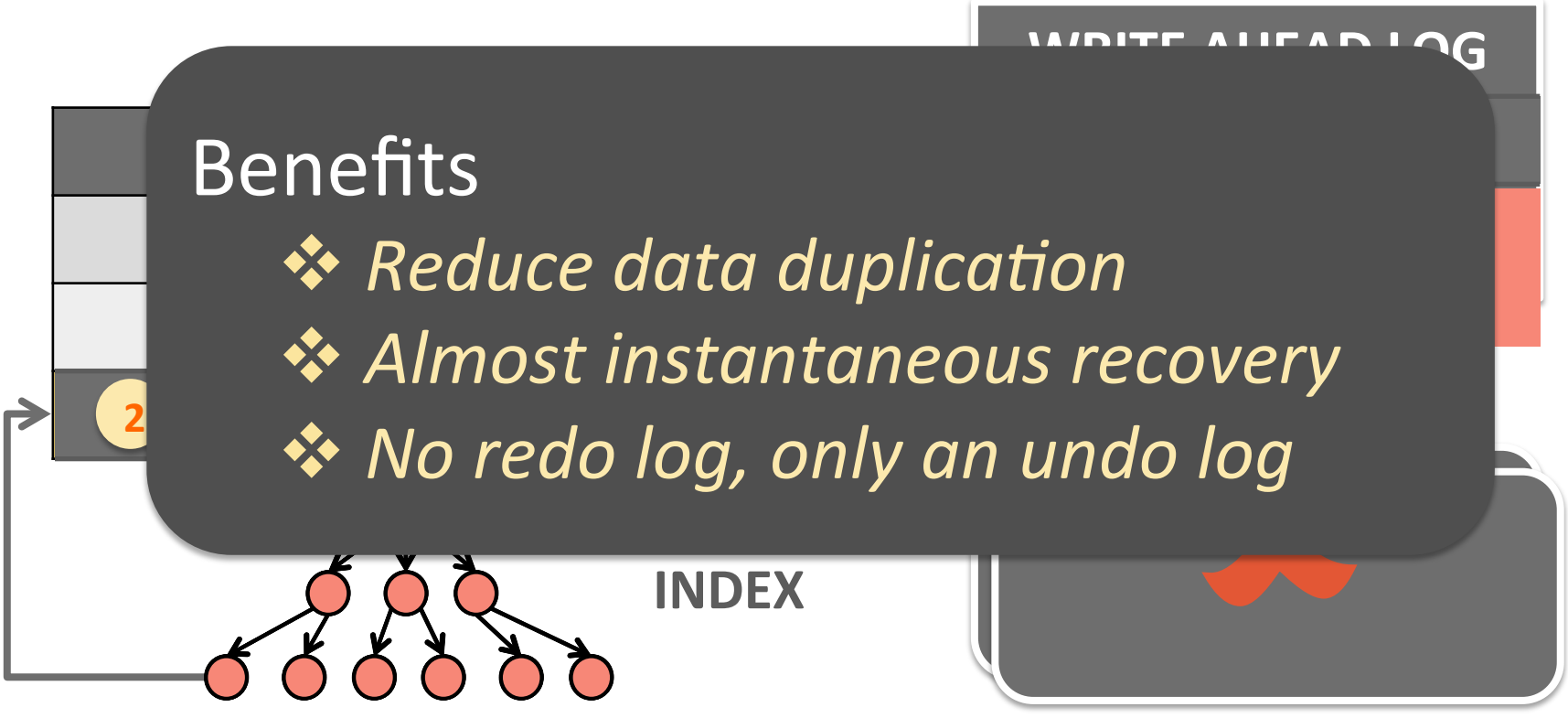




Optimizing for NVM

- Non-volatile pointer
  - *Non-volatile data structures*
  - *Valid even after system restarts*
- Exclusively use allocator interface
  - *Byte-addressable NVM*
  - *Not filesystem interface*

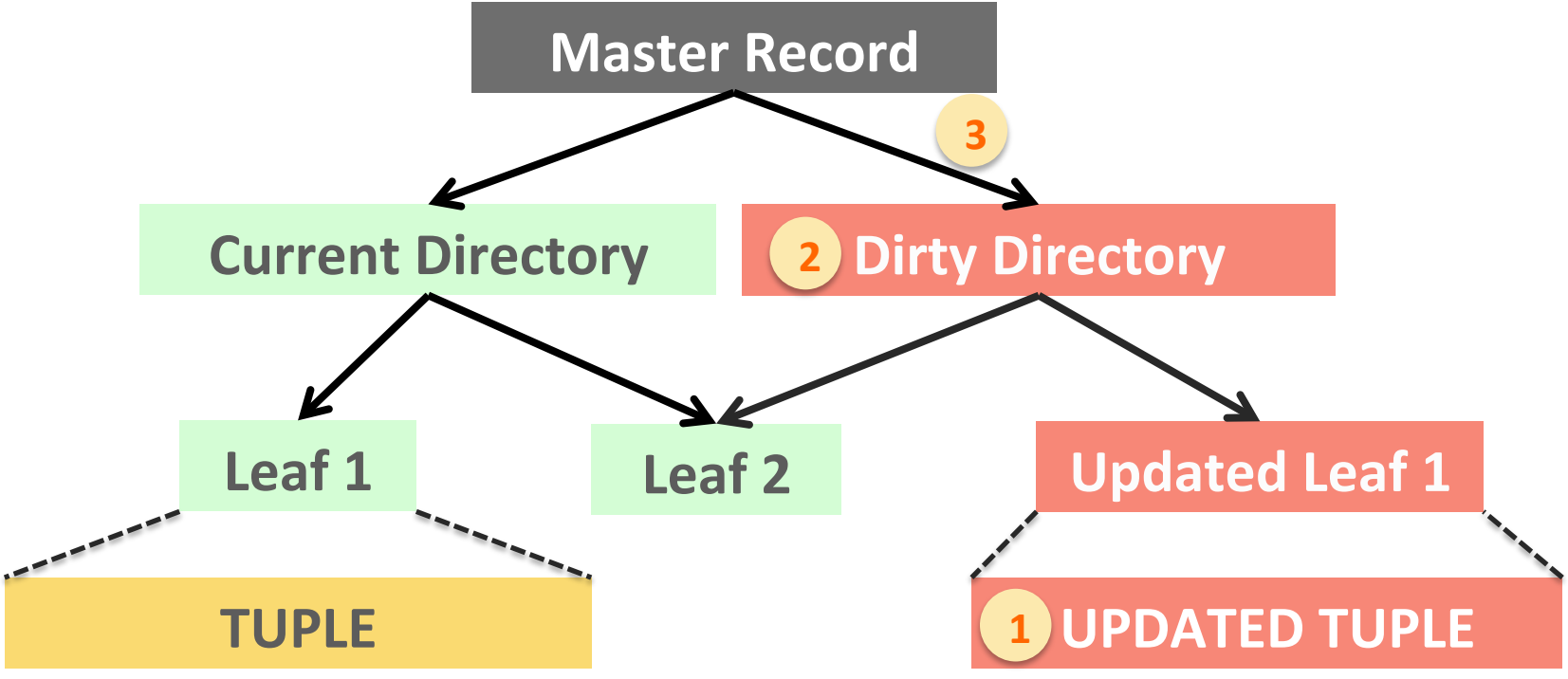
#1: NVM In-Place Updates



Benefits

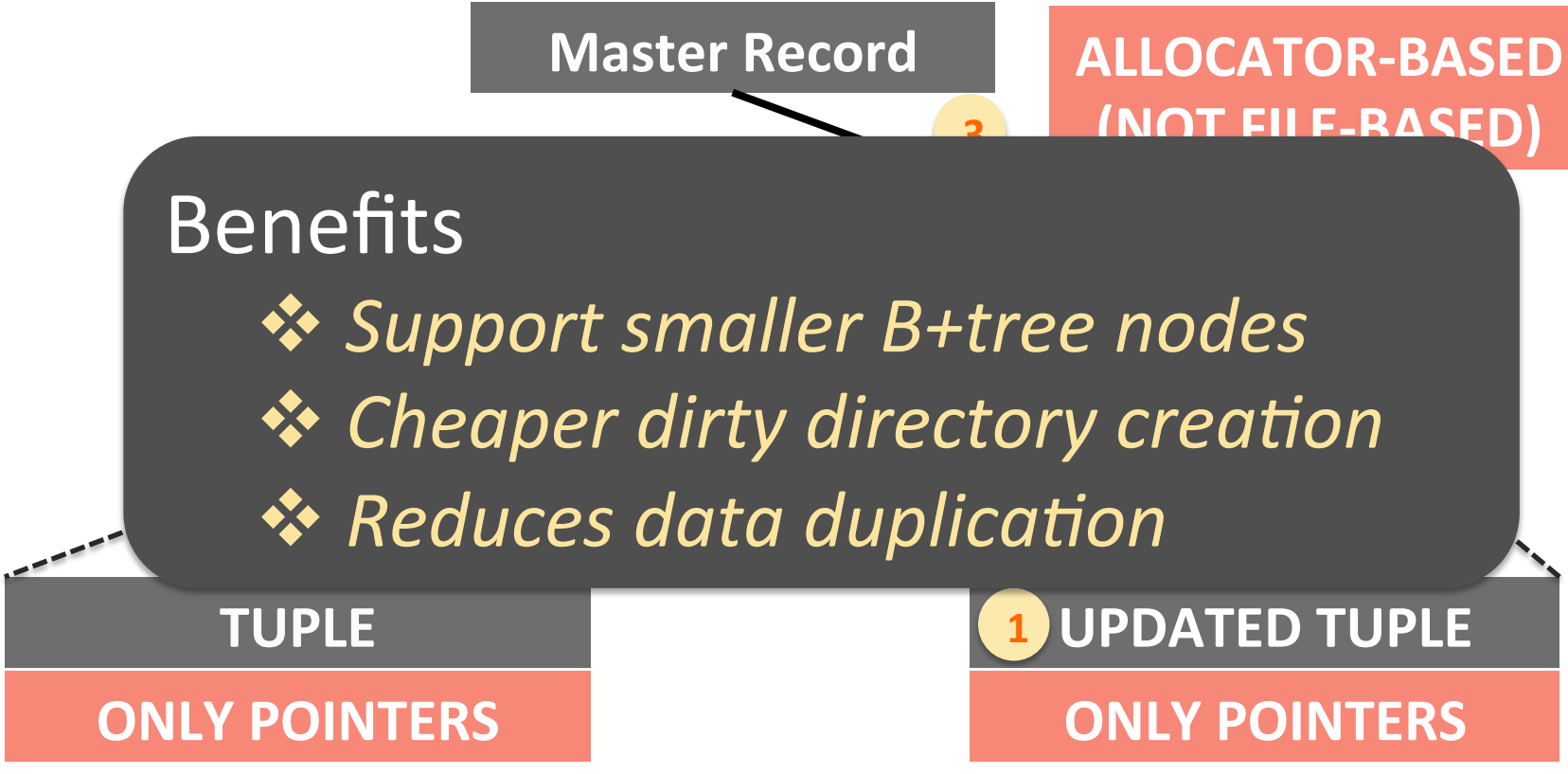
- *Reduce data duplication*
- *Almost instantaneous recovery*
- *No redo log, only an undo log*

#2: Copy-on-Write Updates

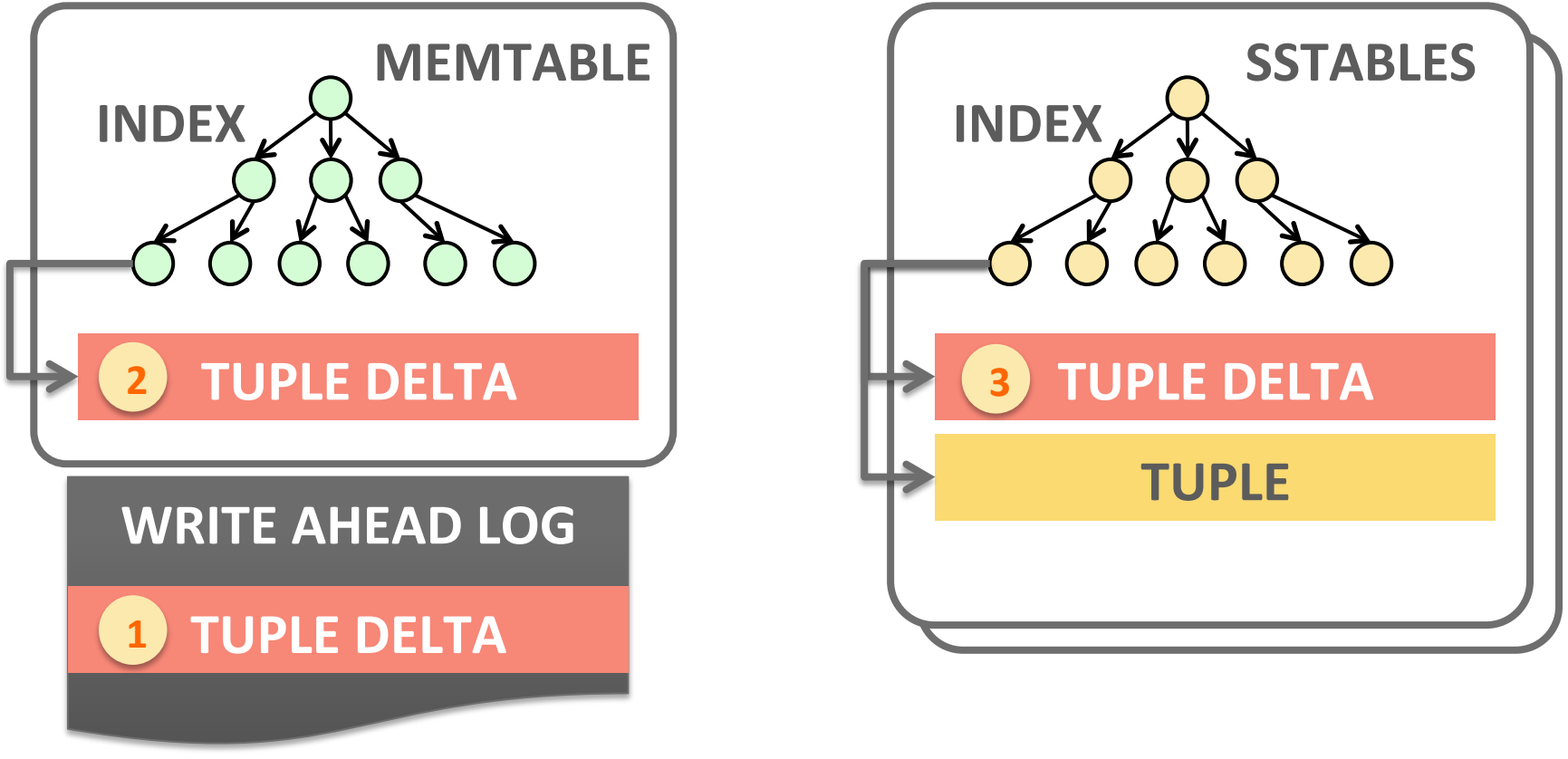


Copy-on-Write B+Tree

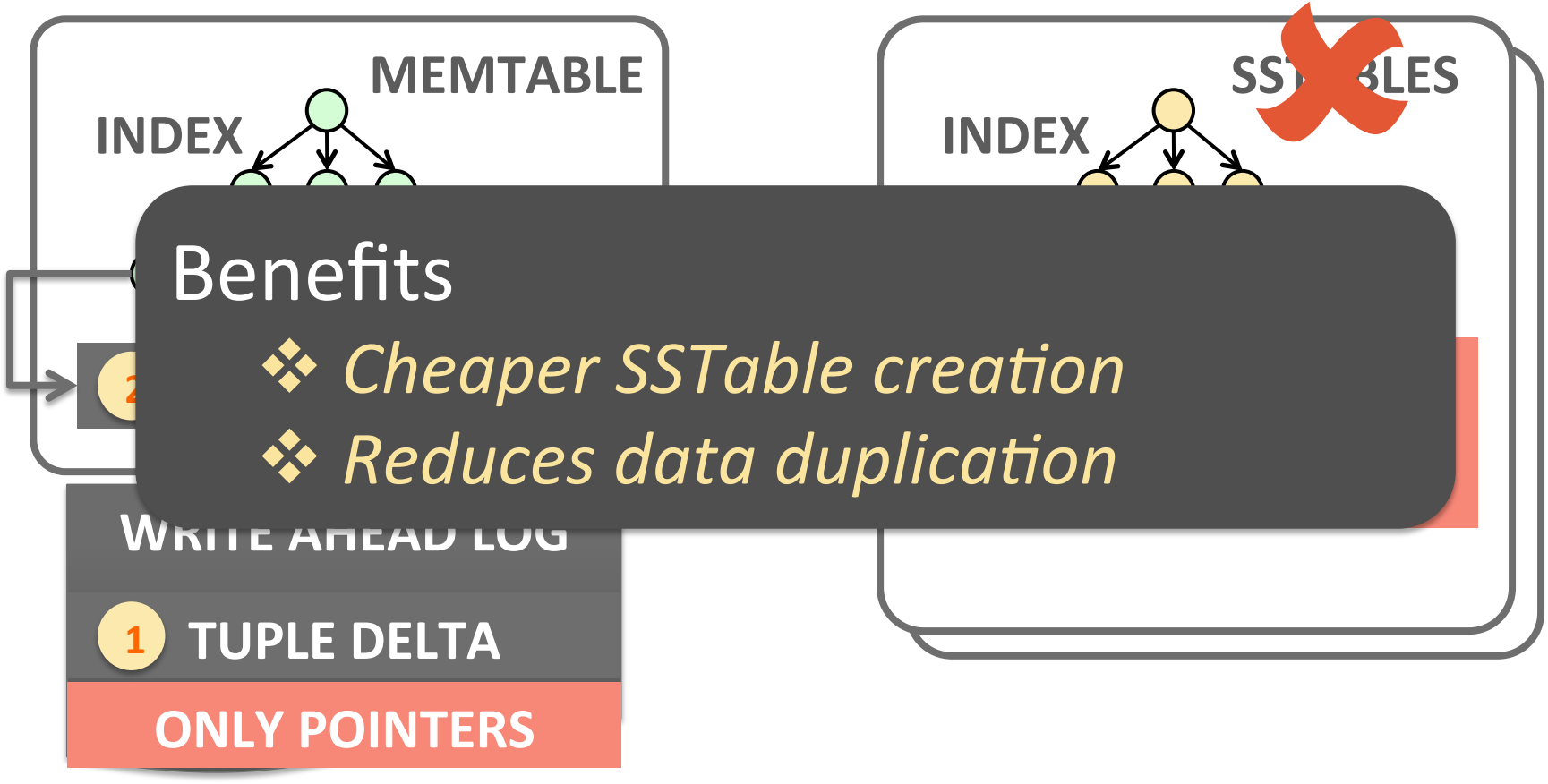
#2: NVM Copy-on-Write Updates



#3: Log-Structured Updates



#3: NVM Log-Structured Updates



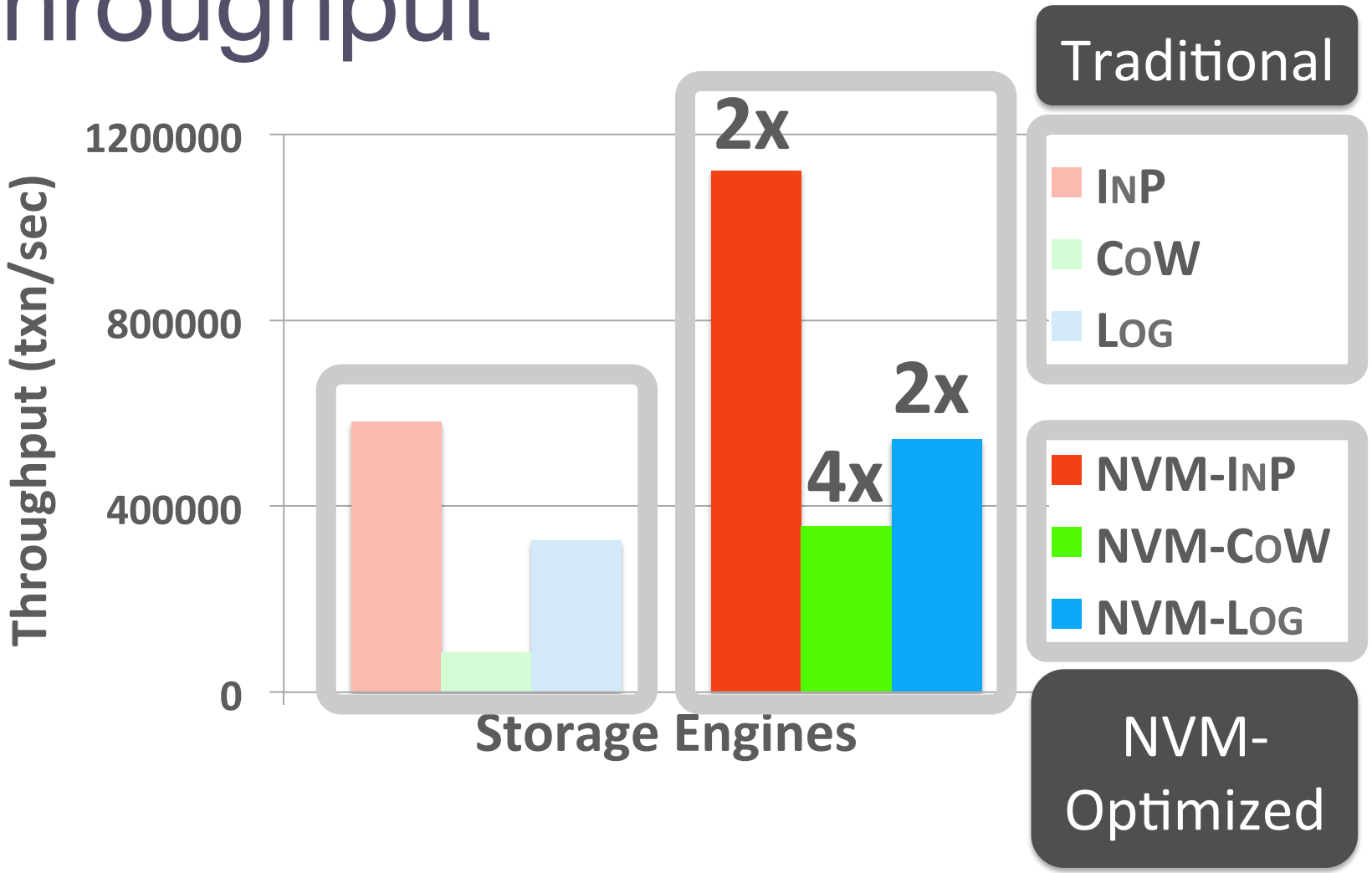
Summary

- Storage optimizations
  - *Avoid unnecessary data duplication*
  - *Leverage byte-addressability*
- Recovery optimizations
  - *NVM-optimized recovery protocols*
  - *Non-volatile data structures*

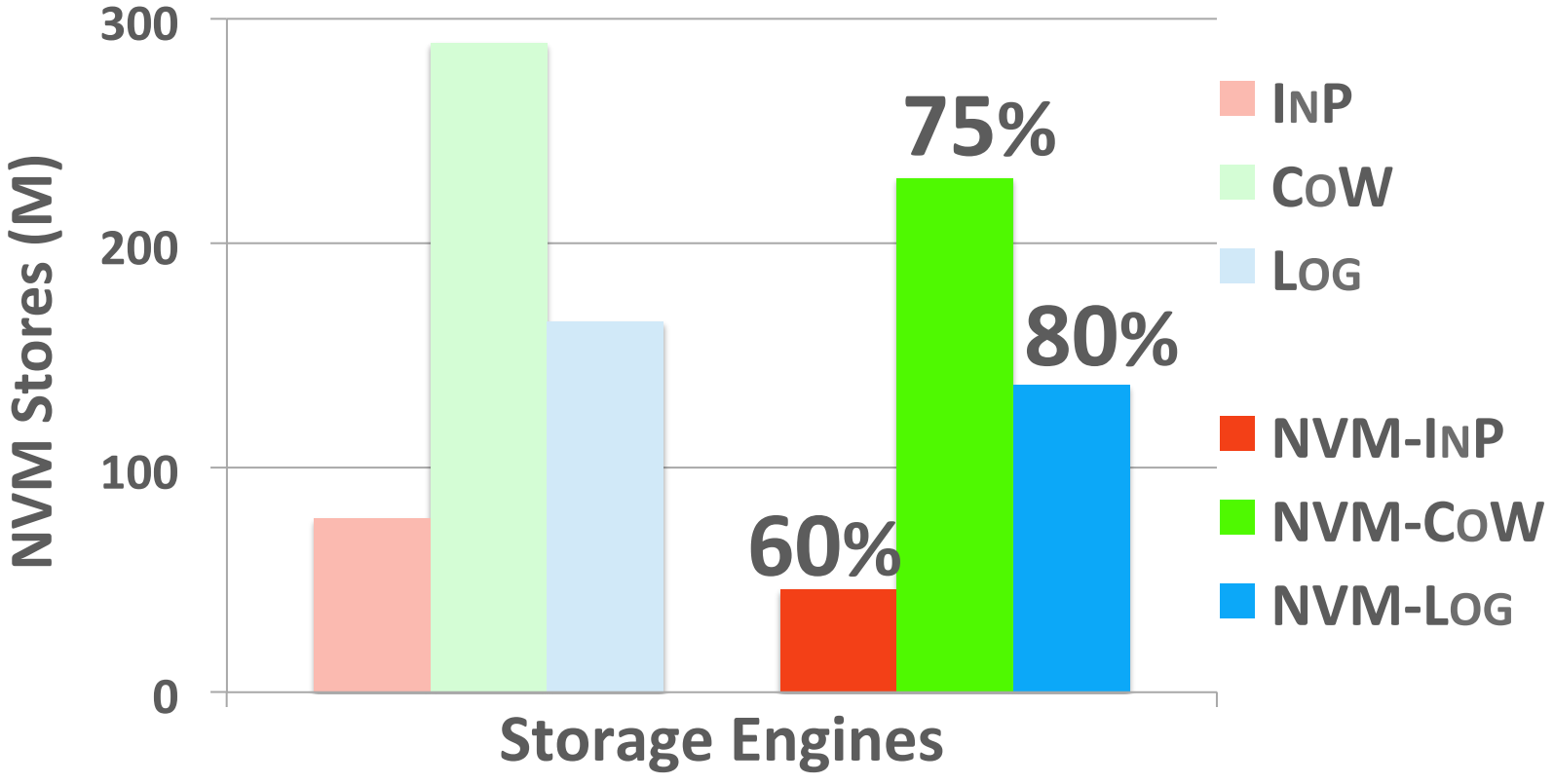
Experimental Evaluation

- NVM Hardware Emulator
  - *NVM latency = 2x DRAM latency*
- Yahoo! Cloud Serving Benchmark
  - *6 storage engines*
  - *2 million records + 1 million transactions*
  - *Write-heavy workload*
  - *High-skew setting*

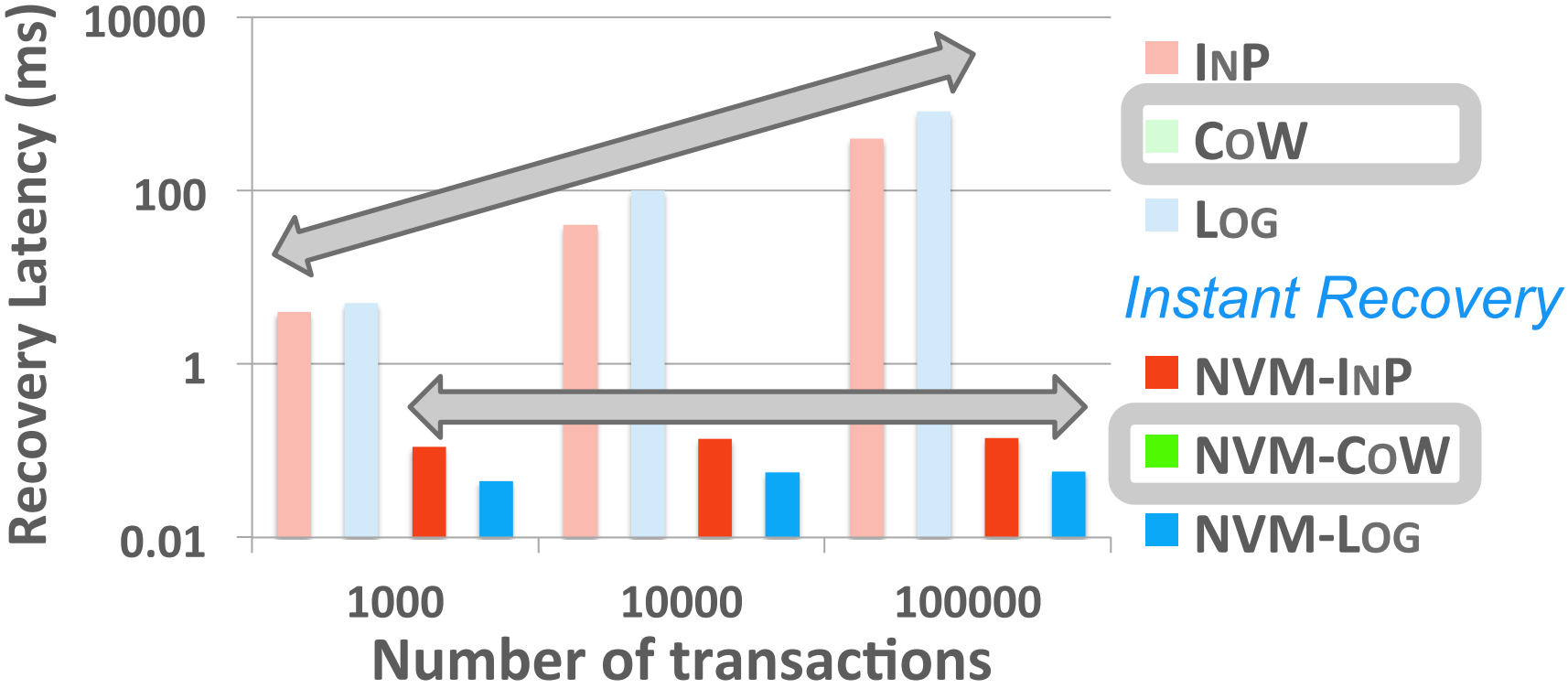
Throughput


Traditional
2x
1200000
800000
400000
0
Throughput (txn/sec)
INP
COW
LOG
2x
4x
NVM-INP
NVM-COW
NVM-LOG
Storage Engines
NVM-Optimized

Write Endurance



Recovery Latency



Takeaways

- Designing for NVM is important
  - *Higher throughput*
  - *Longer device lifetime*
  - *Faster recovery*
- System design principles
  - *Non-volatile data structures*
  - *Need a system-level rethink*

Peloton @ CMU

- Hybrid storage hierarchy
  - *NVM + DRAM oriented design*
- HTAP workloads
  - *Real-time analytics and fast transactions*

END

Thanks !

jarulraj@tinder

Filesystem Interface

- Optimized for byte-addressable NVM
- Bypass page-cache & block device layer
- File I/O requires only one copy
  - *7x better performance than EXT4*

Allocator Interface

- NVM-aware memory allocator
  - *No system calls*
  - *Bypass kernel's VFS layer*
- Flush CPU cache for durable writes
  - *10x better performance than FS interface*