Lightning Performance in a Non-Main-Memory Transactional Key-Value Store

David Lomet

With

Justin Levandoski, Sudipta Sengupta, Ryan Stutsman, and Rui Wang

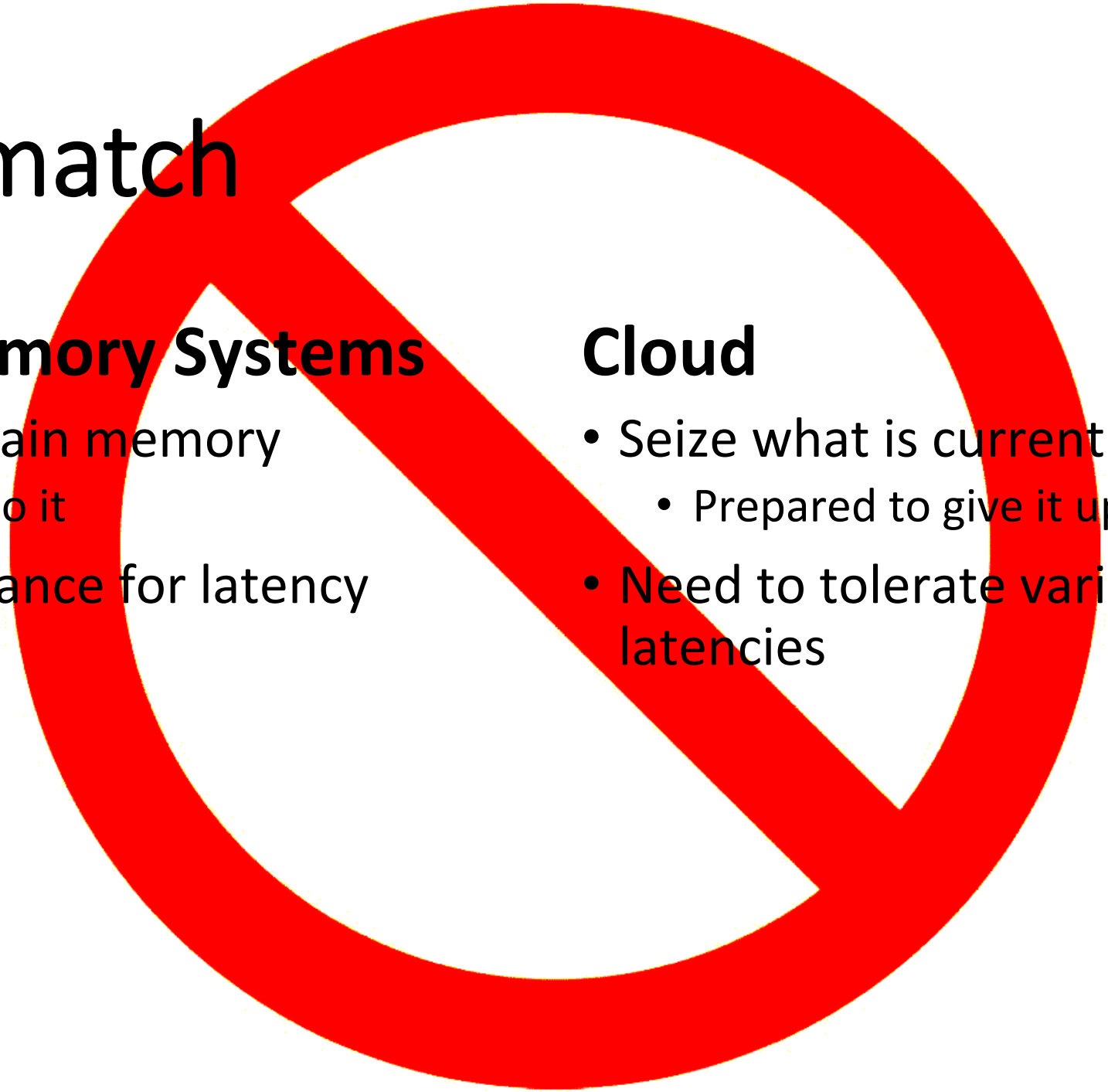
High Performance Main Memory (MM) Systems

- **Most high performance database systems and key value stores are now main memory systems**
  - Main memories are large and growing- so most OLTP fits
  - Restricting to main memory avoids I/O latency
- **Latency ruins performance on most of these systems**
  - Concurrency control struggles- too many aborts when using the now commonplace optimistic methods
  - Number of versions can explode with MVCC methods

The Cloud as Economic Destiny

- **Economic case for the cloud is compelling**
  - Order of magnitude cheaper than in-house when factoring in
    - Hardware costs
    - Operations costs
- **Economic case based mostly on Multi-tenancy**
  - Large number of services, etc. share machine resources
  - Service level agreements and user requirements result in variable size resource partitioning
  - AND varying load enables over committing by scaling resources to the present load
    - VMs or database instances grow and shrink their cpu and memory footprints

A Mismatch



Main Memory Systems

- Seize all main memory
  - Hold onto it
- Little tolerance for latency

Cloud

- Seize what is currently needed
  - Prepared to give it up as needed
- Need to tolerate variable cloud latencies

Deuteronomy to the Rescue

Deuteronomy shares parts of MM Approaches

- **Latch free**
  - **Without partitioning: like Hekaton**
  - And not like H-store with partitioning
    - Which trades a latching problem for a partition management problem
- **Multi-version concurrency control for transactions**
  - **MVCC dramatically reduces RW conflicts**
    - Enabling abort on conflict instead of blocking
  - **Protects accesses when made: called pessimistic**
    - Not like most MM systems which validate later: called optimistic
  - **Crucially- pessimistic increases commit likelihood in face of I/O latency**

Deuteronomy Database State resides on 2nd-ary storage

- **Log structured store**
  - A bit like LSM– and hence RocksDB
    - But without 3+ trees and a merge process
  - Removes write I/O as a performance factor
- **Blind writes to pages**
  - Page need not be present in main memory for this write
    - **No read needed for writes**
- **Reads may be fragmented if not in main memory**
  - We use SSDs to mitigate this

Performance (fully cached)

YCSB: Four ops/transaction

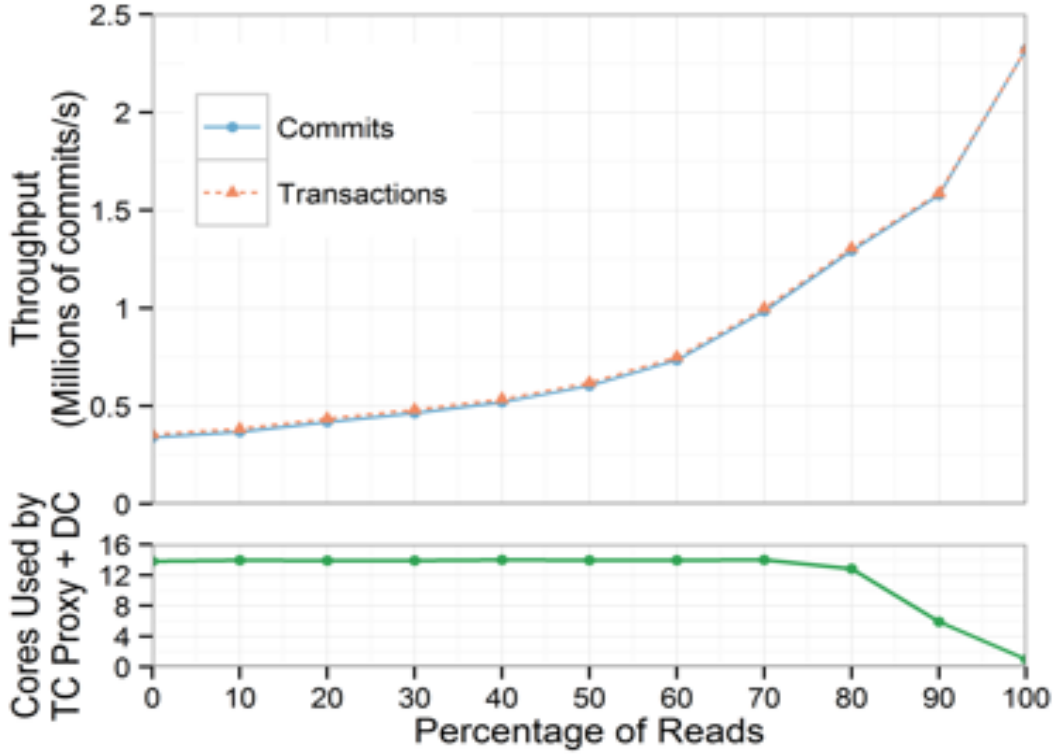


Figure 6: The impact of workload read/write mix on performance and DC core utilization.

YCSBE: With Ranges

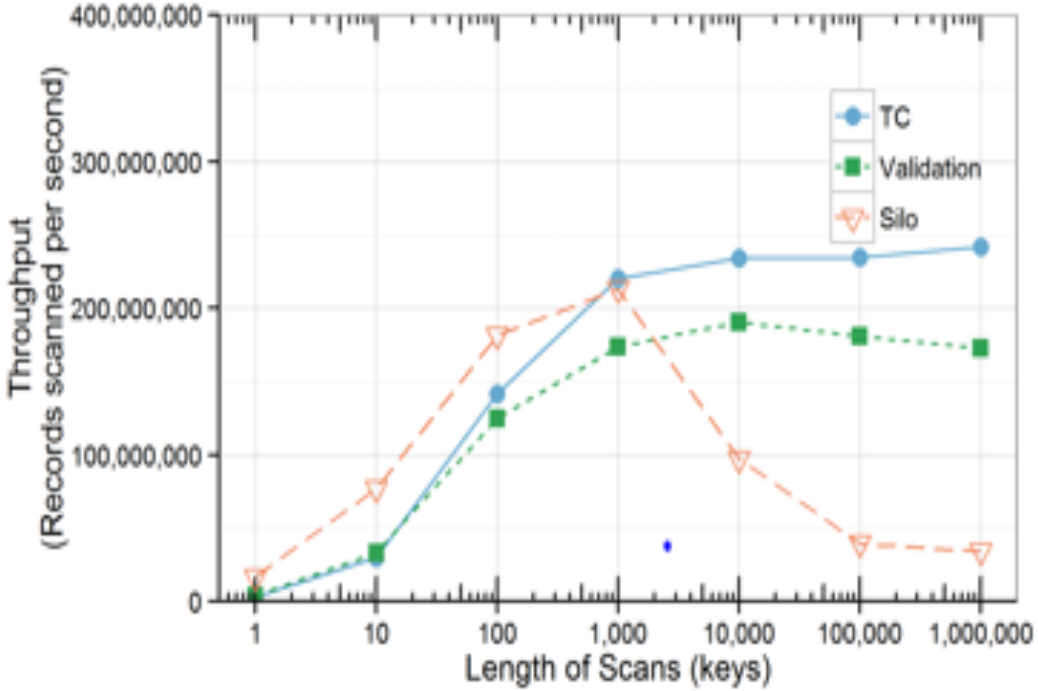


Figure 8: MVCC range scan throughput for various approaches.

Deuteronomy-
you can’t go wrong!!