Big and Fast

Anti-Caching in OLTP Systems

Justin DeBrabant

BROWN
Database Group

Online Transaction Processing

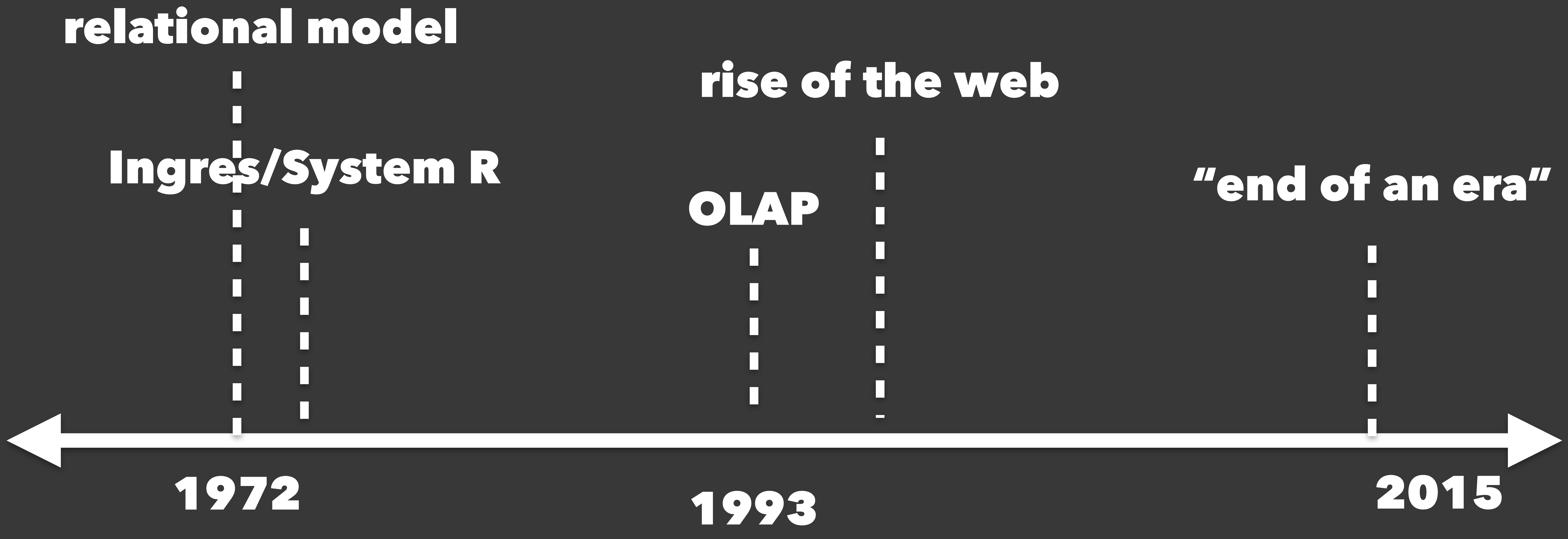
transaction-oriented

small footprint

write-intensive

A bit of history...

OLTP Through the Years



Modern OLTP Requirements

1. web-scale (big)
2. high-throughput (fast)

Thesis Motivation

- traditional disk-based architectures aren't fast enough
- newer main memory architectures aren't big enough

Can we have main-memory performance for larger-than-memory datasets?

Thesis Overview: Contributions

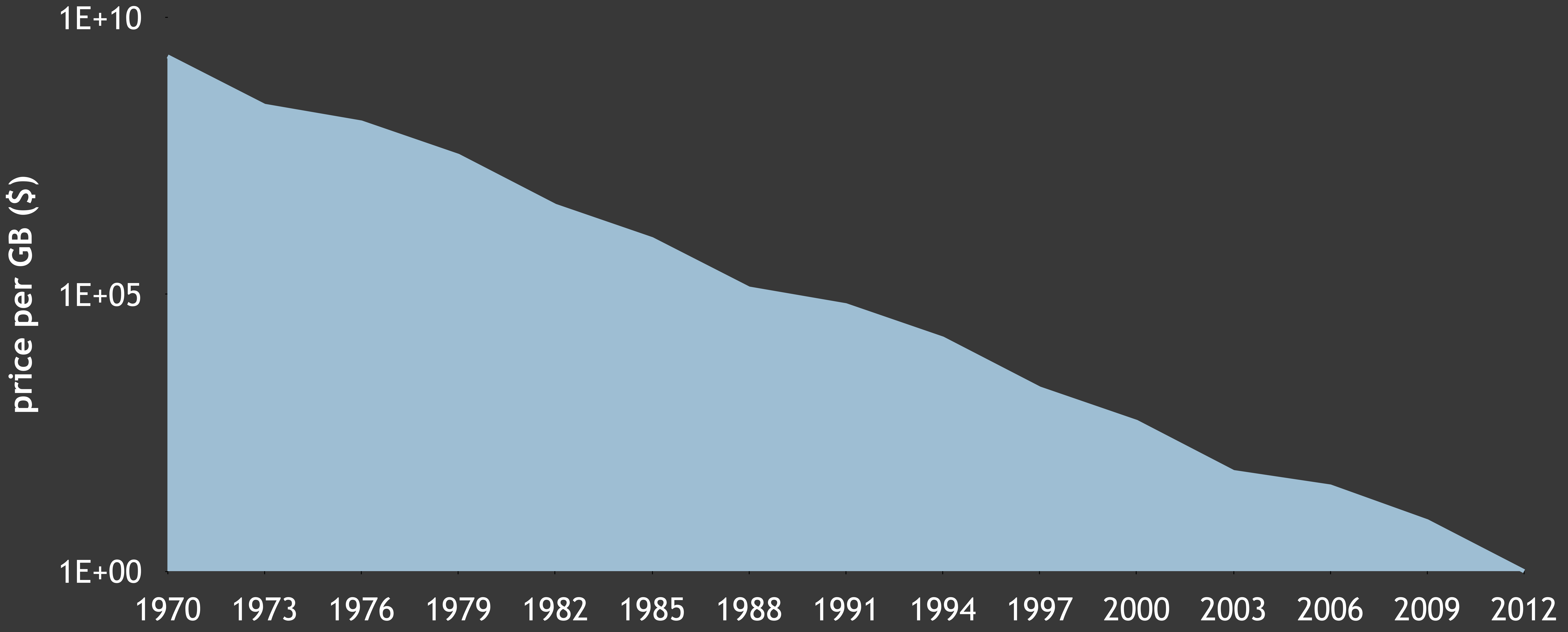
1. **anti-caching architecture**
   - larger than memory datasets in main memory DBMS
2. **anti-caching + persistent memory**
   - exploring next-generation hardware and OLTP systems

Outline

Disk-Oriented Architectures

- assumption: data won't fit in memory
- disk-resident data, main memory buffer pool for execution
- concurrency is a must
  - transaction serialization and locks

Memory Costs



BROWN
Database Group

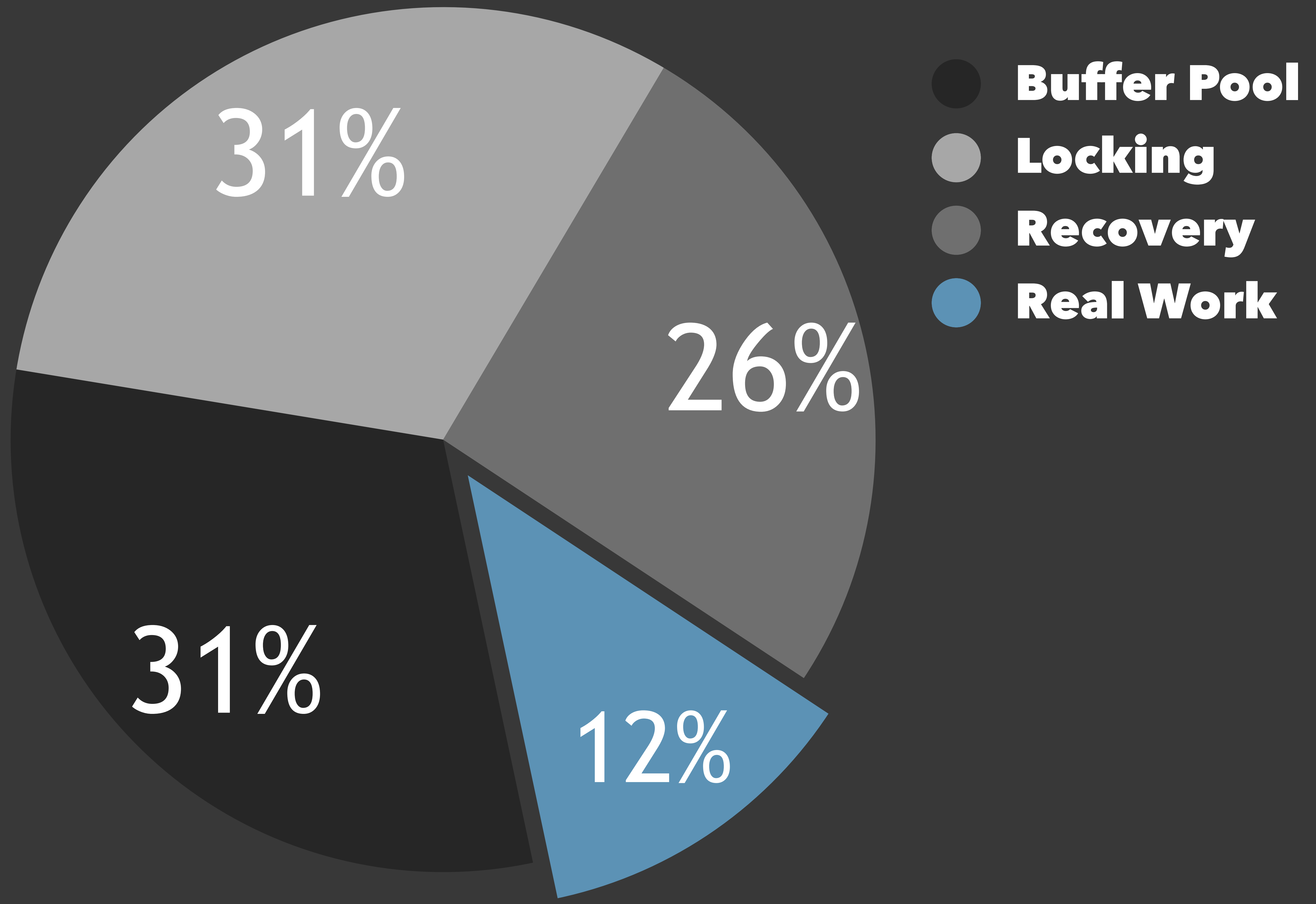
Now What?

1. DBMS buffer pool
2. distributed cache
3. in-memory DBMS

Buffer Pool

- must still…
  - maintain buffer pool
  - lock/latch data
  - maintain ARIES-style recovery logs
- question: What is the overhead of all these things?

BROWN
Database Group



OLTP Through the Looking Glass, and What We Found There
SIGMOD '08

Now What?

1. DBMS buffer pool
2. distributed cache
3. in-memory DBMS

Cache Layer

Persistence Layer

Main Memory Cache

- fast *and* scalable, but…
- key-value interface
- not ACID (AI, not CD)

Consistency and Durability

- reads are easy, writes are not
  - multiple copies of data
  - application's responsibility
- for OLTP, writes are common and consistency is essential

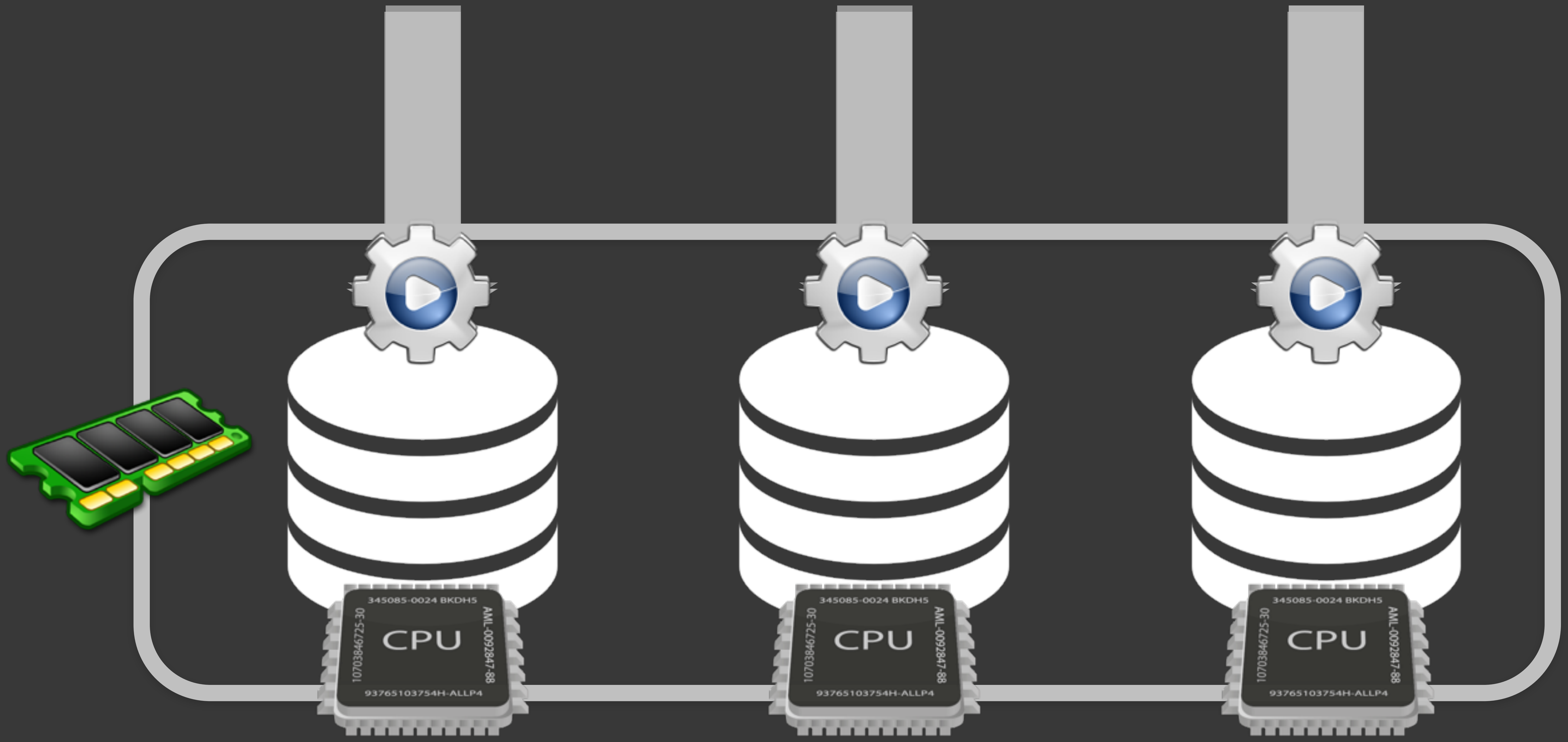
Now What?

1. DBMS buffer pool
2. distributed cache
3. in-memory DBMS

H-Store

H-Store Architecture

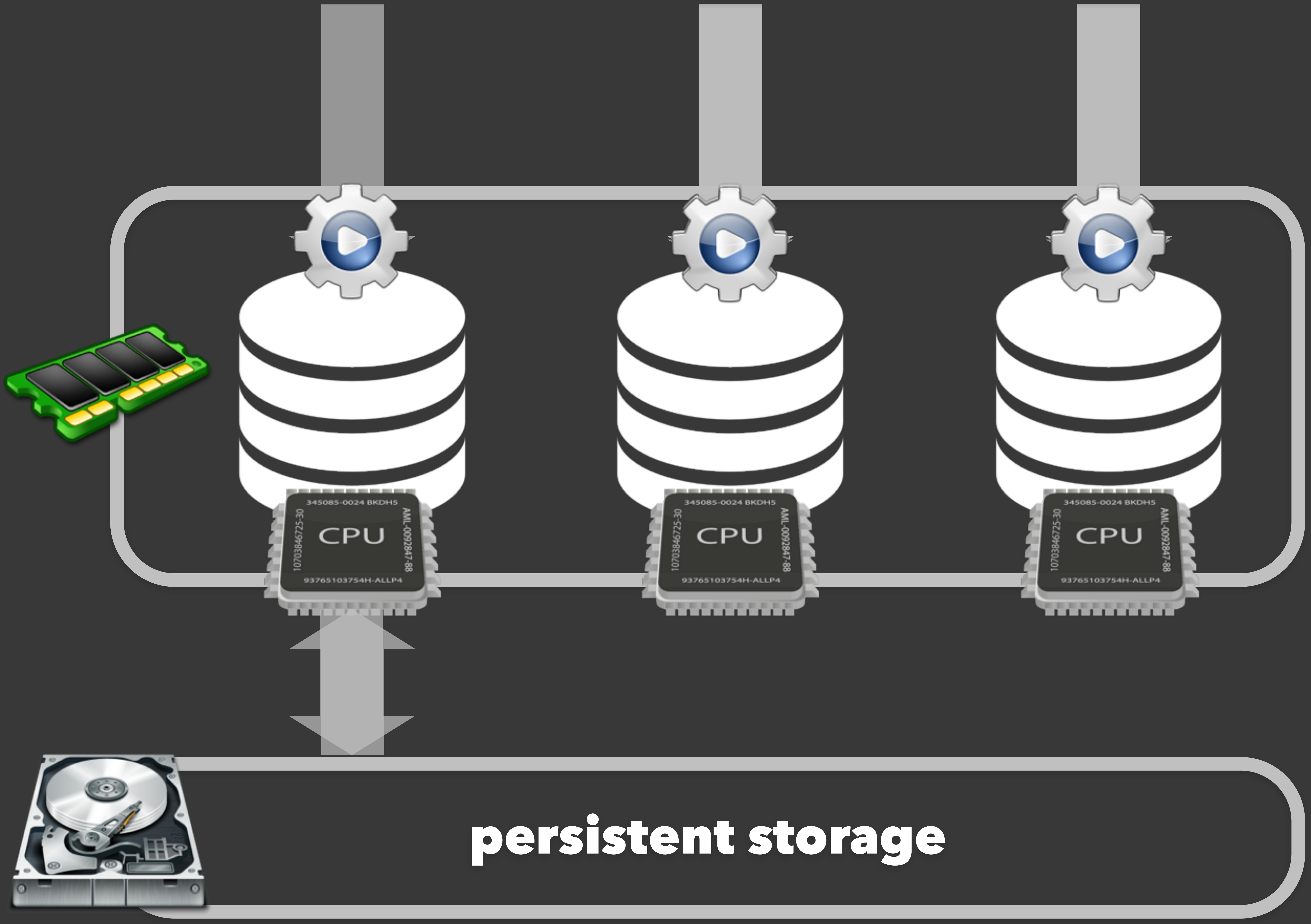
- partitioned, shared-nothing
- single-threaded main memory execution
  - no need for locks and latches
- lightweight recovery
  - snapshots + command log



CPU
CPU
CPU

virtual memory?

data -> memory?


345085-0024 BKDH5
CPU
AML-0092847-88
10703846725-30
93765103754H-ALLP4
persistent storage

Big and Fast

big: disk-oriented

fast: memory-oriented

big *and* fast: anti-caching

OLTP workloads are *skewed*

Design Principles

- asynchronous disk fetches
  - don't block
- maintain ordering of evicted data accesses
  - ensures transactional consistency
- single copy of data
  - consistency is free
- efficient memory use, no swizzling

Outline

- Introduction
- Overview and Motivation
- Anti-Caching Architecture
- Memory Optimizations
- Anti-Caching on NVM
- Future Work and Conclusions

Architectural Overview

- memory is primary storage, cold data is evicted to disk-based anti-cache
- reading data from the anti-cache is done in 3 phases
  - avoids blocking, ensures consistency

Anti-Caching Phases

- evict
- pre-pass
- fetch
- merge

Evict

1. data > anti-cache threshold
2. dynamically construct anti-cache blocks of coldest tuples
3. asynchronously write to disk

Pre-Pass

1. a transaction enters pre-pass when evicted data is accessed
2. continues execution, creating list of evicted blocks
3. abort, queue blocks to be fetched

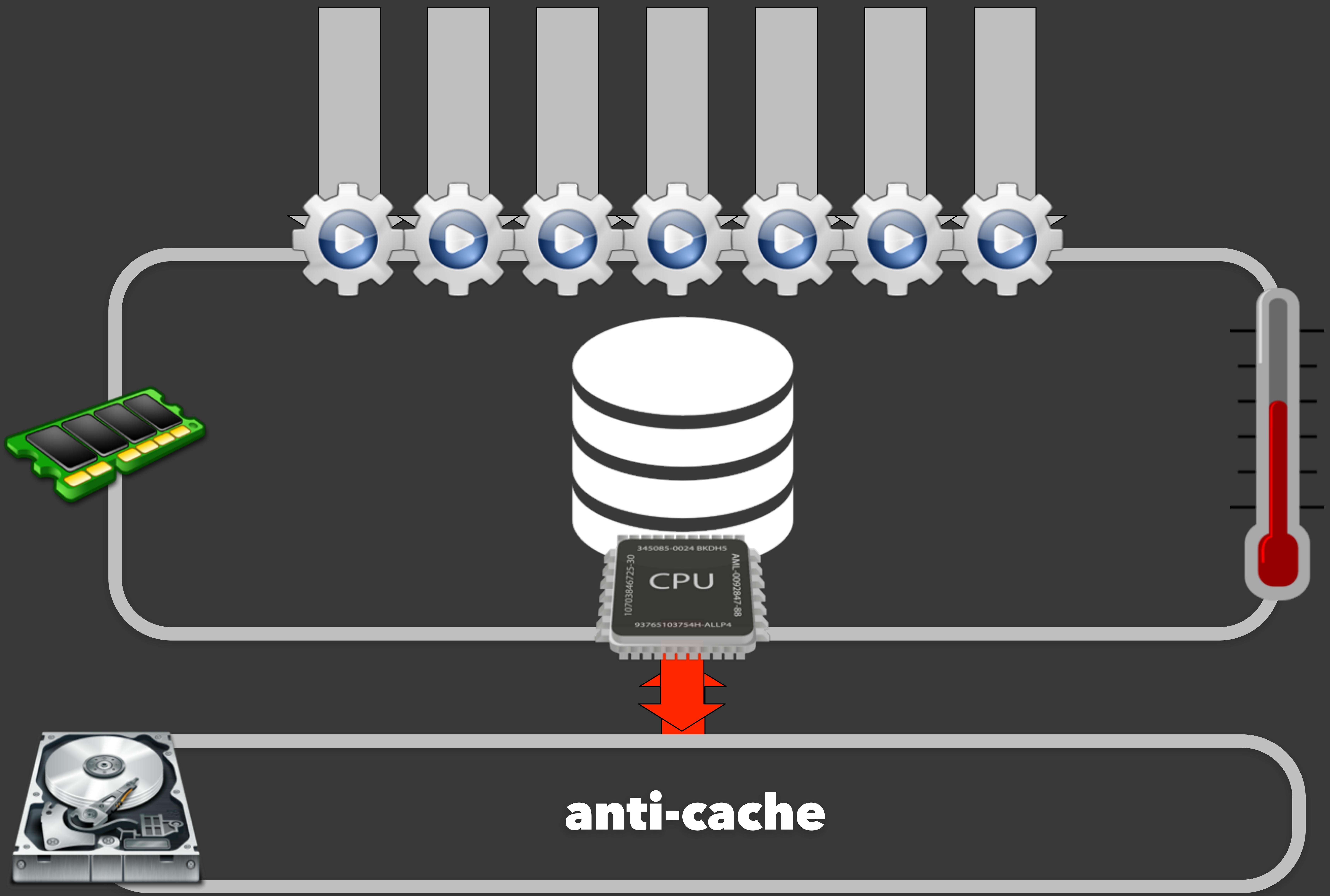
Fetch

1. data is fetched asynchronously from disk
   - avoids blocking
2. moved into merge buffer

Merge

1. data is moved from in-memory merge buffer to in-memory table
2. previously aborted transaction is restarted
3. transaction executes normally

BROWN
Database Group


CPU
anti-cache

Tracking Access Patterns

- done online, more responsive to changes in workload
- goal is low CPU and memory overhead
- approximate ordering is OK

Approximate LRU (aLRU)

- maintain LRU chain embedded in tuple headers
- per-partition
- transactions that update LRU chain are sampled randomly
  - configurable sample rate

Anti-Caching vs. Swapping

- fine-grained eviction
  - blocks constructed dynamically
- asynchronous batched fetches
- possible because of transactions

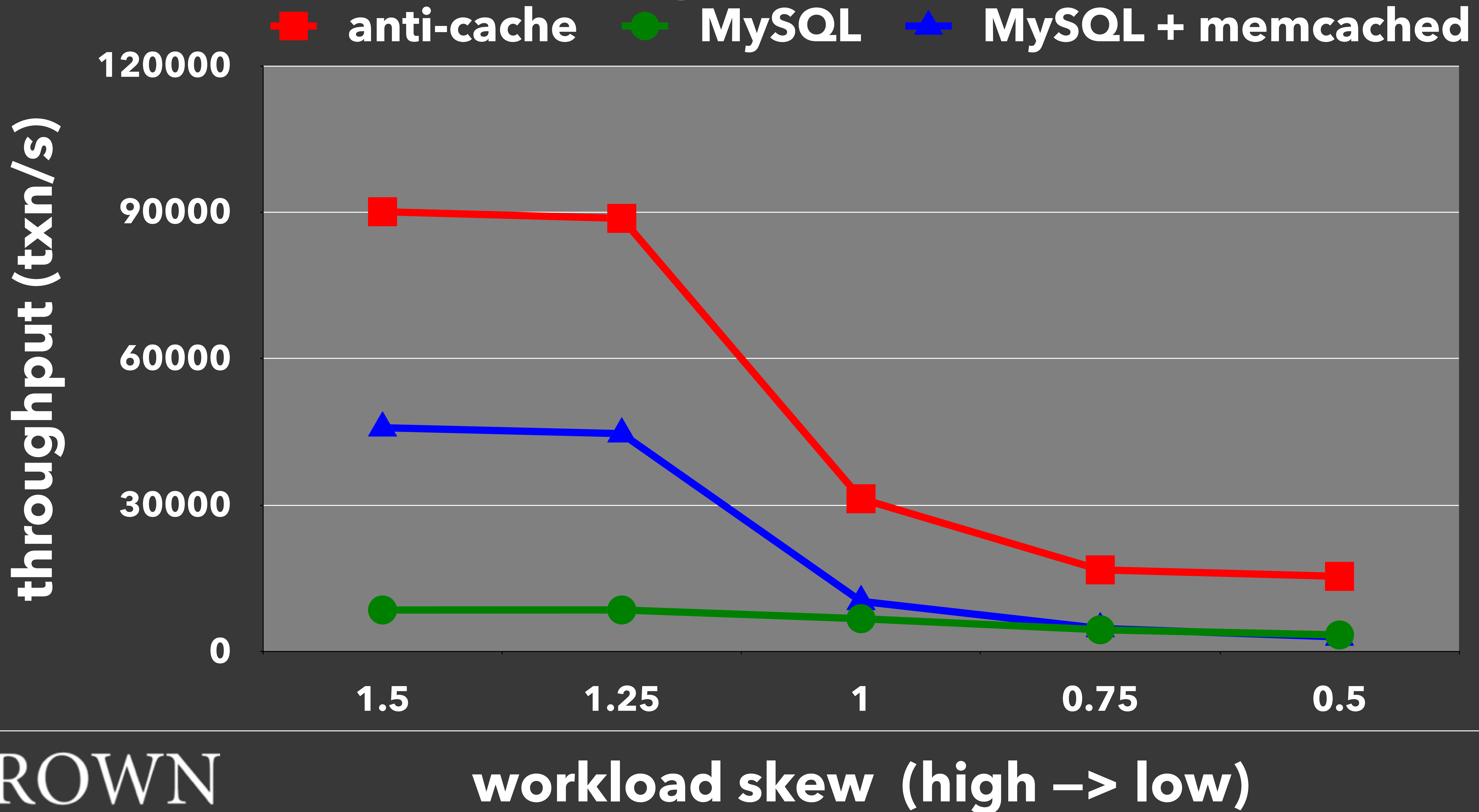
Anti-Caching vs. Caching

- data exists in exactly one location
  - caching architectures have multiple copies, must maintain consistency
  - data is moved, not copied
- goal is increased data size, not throughput

Benchmarking

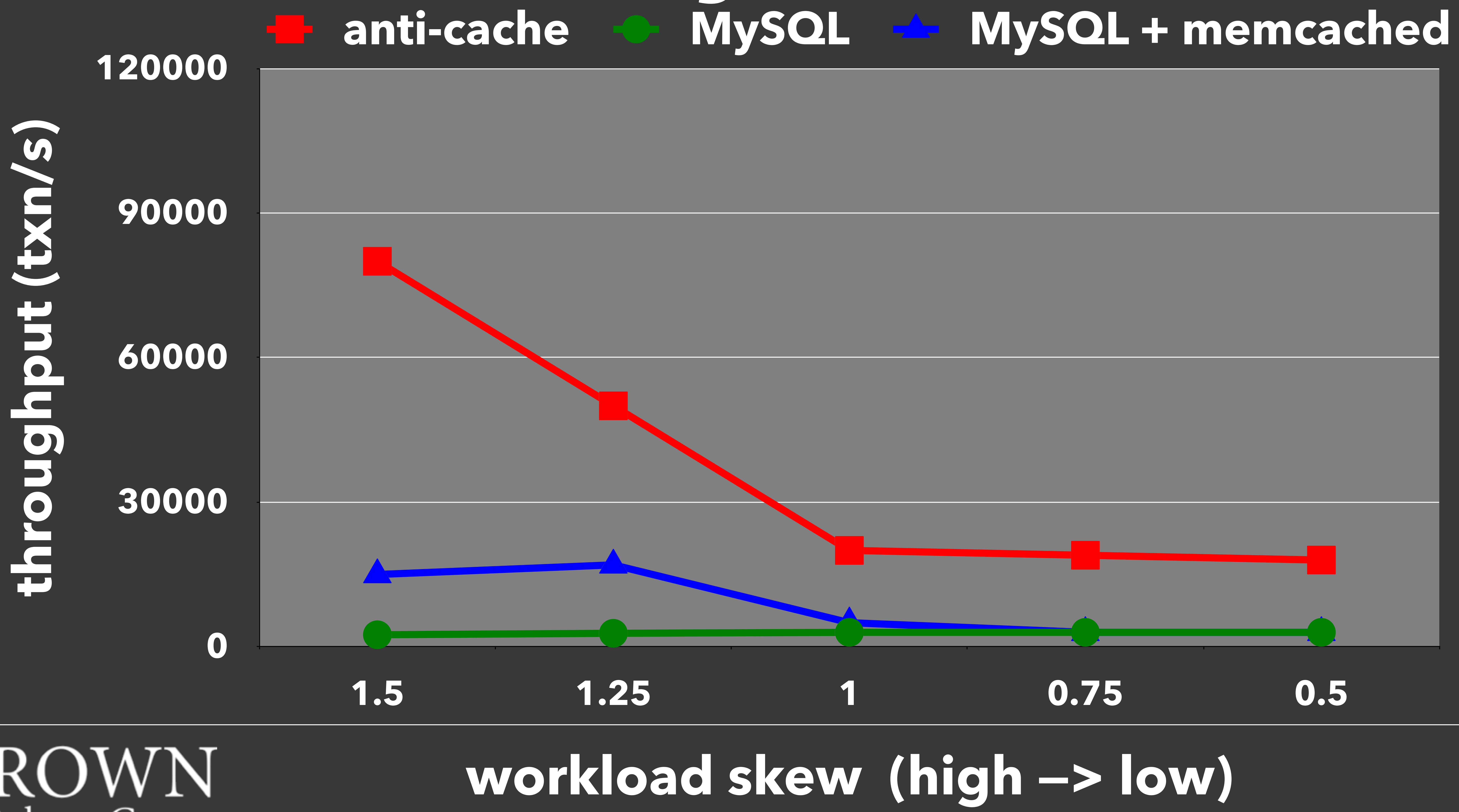
- YCSB
- Zipfian skew
- data > memory
- read/write mix
- MySQL, MySQL + memcached

YCSB, read-only, data 8X memory



BROWN Database Group

YCSB, read-heavy, data 8X memory



BROWN Database Group

Tracking Accesses Revisited

- approximate ordering is OK
- original implementation
  - aLRU (linked list)
  - compute vs. memory

Can we reduce the memory overhead?

Timestamp-Based Eviction

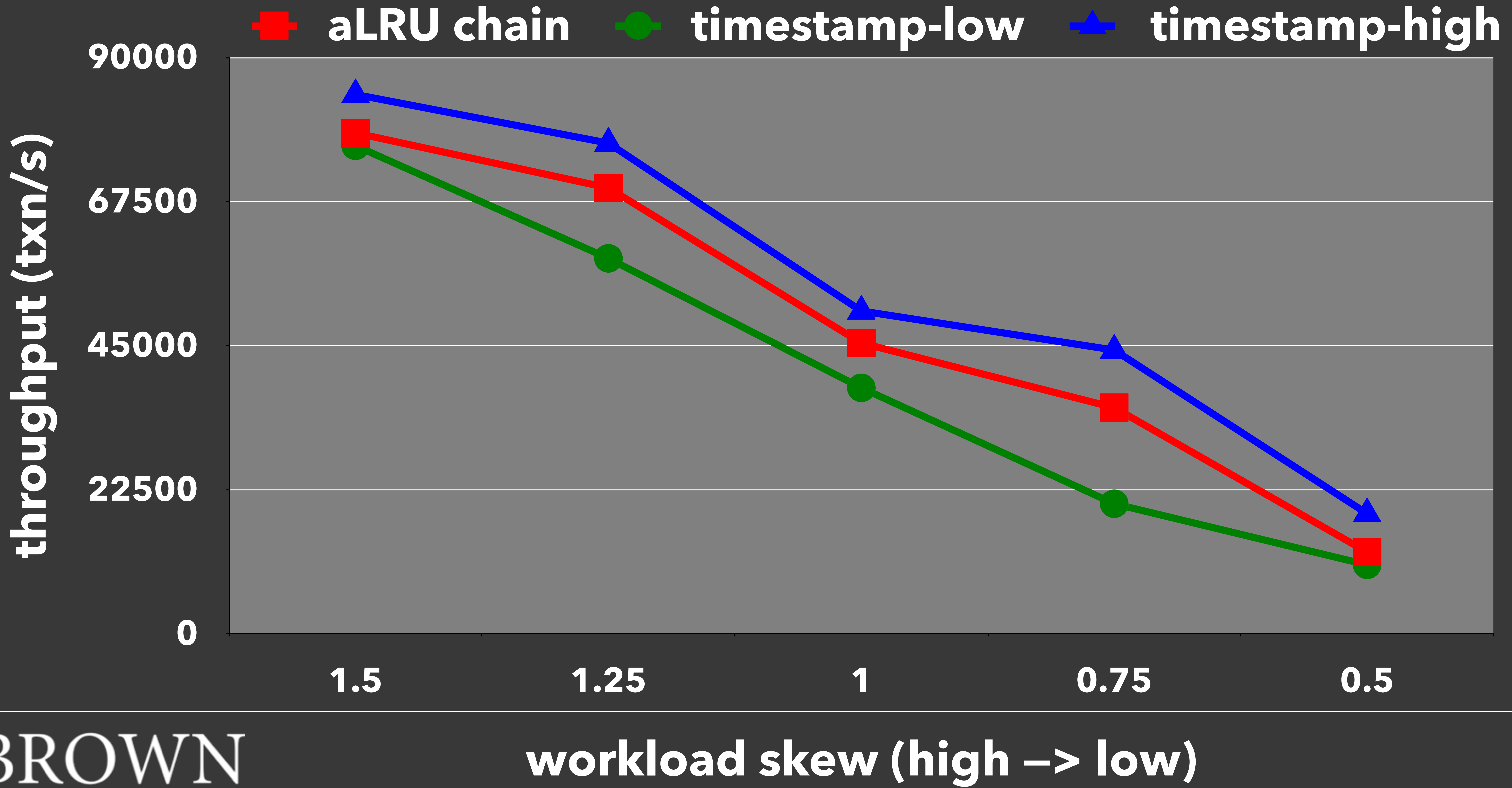
- use relative timestamps to track accesses
- to evict, take subset of tuples and evict based on timestamp age
- questions:
  - timestamp granularity
  - sample size (power of two)

BROWN
Database Group

Timestamp Granularity

- 4 byte timestamps
  - use instruction counter
- 2 byte timestamps
  - use epochs, set the timestamp to the current epoch

YCSB, read-heavy, data 8X



BROWN
Database Group

Key Take-Aways

- 8-17X improvement for skewed workloads at larger-than-memory data sizes
- disk becomes the bottleneck for lower skew

Hardware Assumptions are Key

- heavily influence system architectures
- many factors
  - capacity
  - latency
  - volatility

What’s next for OLTP?

Non-Volatile Memory

Properties of NVM

- non-volatile
- random-access
- high write endurance
  - except flash
- byte-addressable
  - except flash

The NVM Arms Race

- FeRAM
  - high write endurance
- MRAM
  - DRAM-like latency
- PCM (PRAM)
  - DRAM-like capacity

BROWN
Database Group

Looking Forward...

- OLTP architectures and NVM
  - anti-cache architecture
  - disk-based architecture
- open questions
  - Which architecture is best suited for NVM?
  - What adaptations are needed?

NVM Emulation

- goal: provide product-independent analysis
- test wide range of latency profiles
- automatically add specified latency
- built by collaborators at Intel

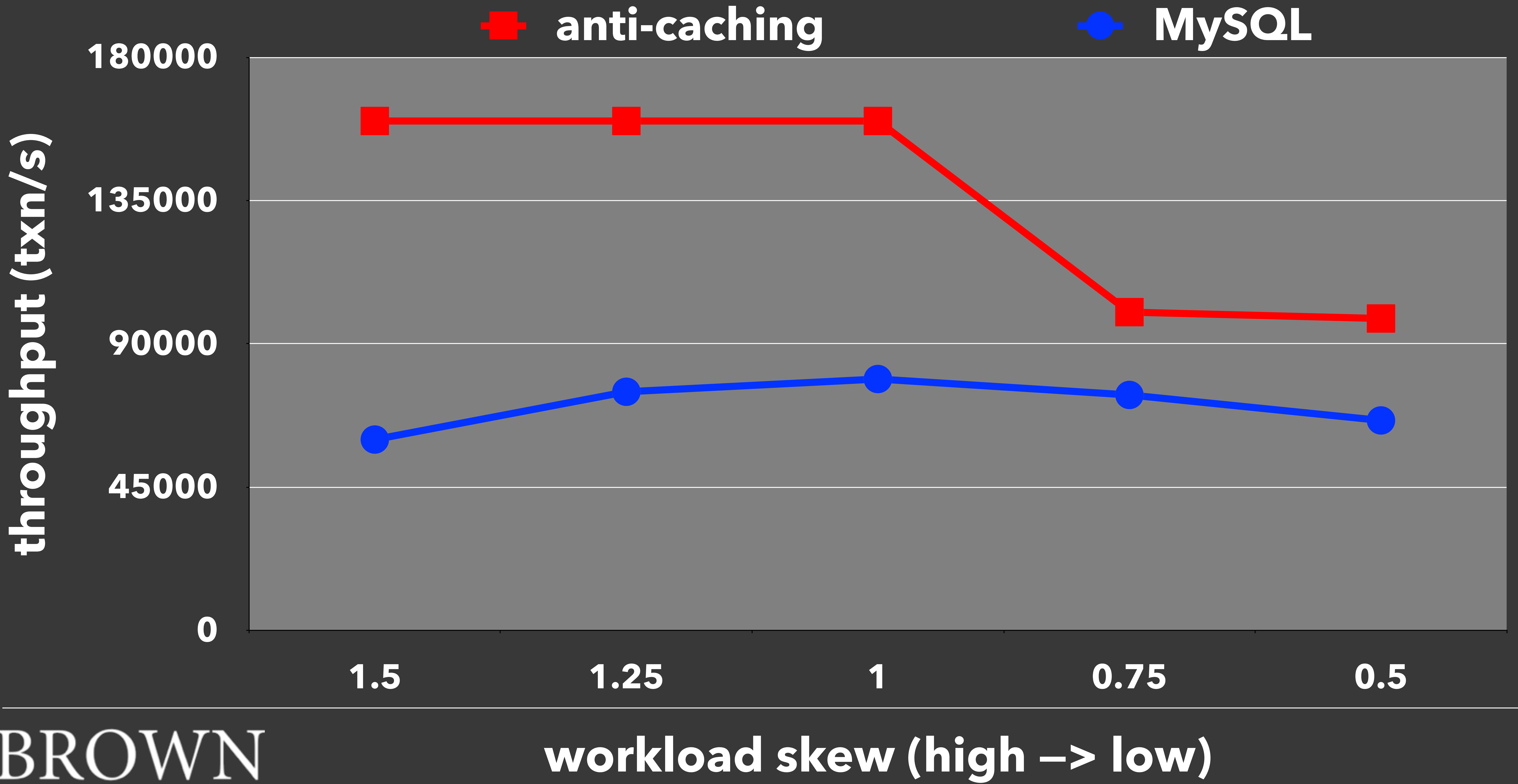
Anti-Caching on NVM

- replace disk with NVM
- several adaptations necessary
  - lightweight array-based anti-cache
    - utilizes mmap interface
  - fine-grained block and tuple eviction interface

Disk-Oriented Architectures on NVM

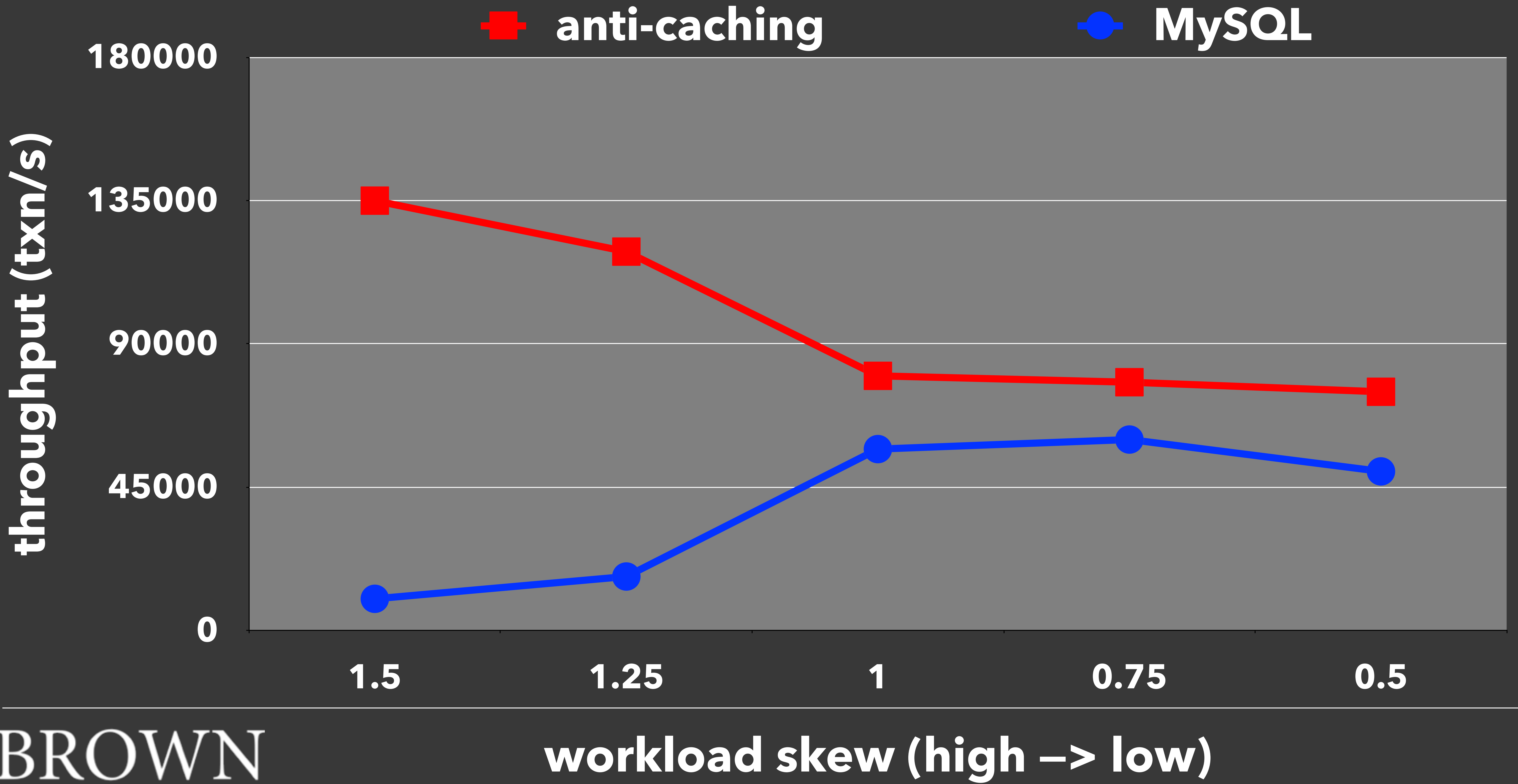
- must adapt both storage and log files to be use NVM mmap interface
- configure to use fine-grained buffer pool pages

YCSB, read-only, data 8X



BROWN
Database Group

YCSB, read-heavy, data 8X



BROWN
Database Group

Future Work

Multi-Tier Architectures

- DRAM -> NVM -> Disk/SSD
- open questions
  - indexing structures
  - synchronous/asynchronous fetches

Anti-Caching Indexes

- index size can be significant
- can cold index ranges be evicted to an anti-cache?
- open questions
  - how/what to evict
  - execution changes

Semantic Anti-Caching

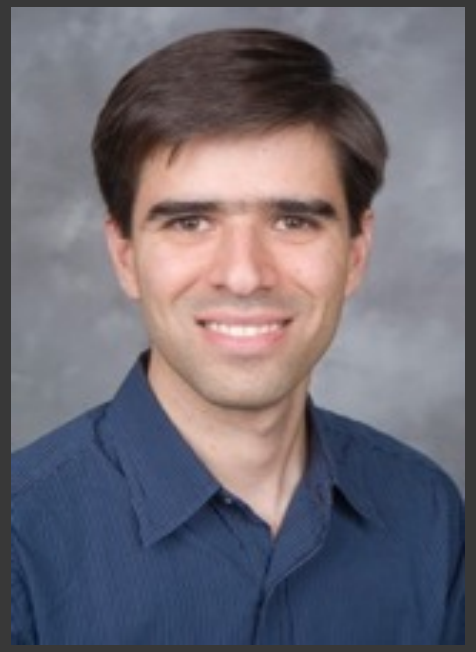
- current implementation makes no assumption about types of skew
- skew typically as semantic meaning
  - e.g., temporal, spatial
- can we leverage these domain semantics?

Conclusions

- anti-caching architecture outperforms and outscales previous OLTP architectures
- well-suited for next-generation NVM-based architectures

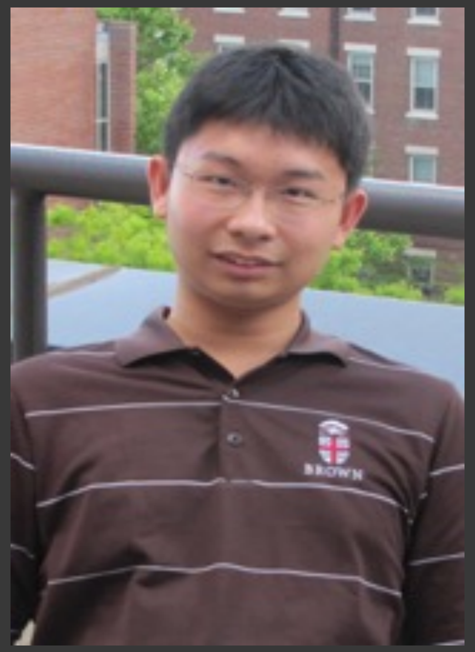












Questions?

debrabant@cs.brown.edu