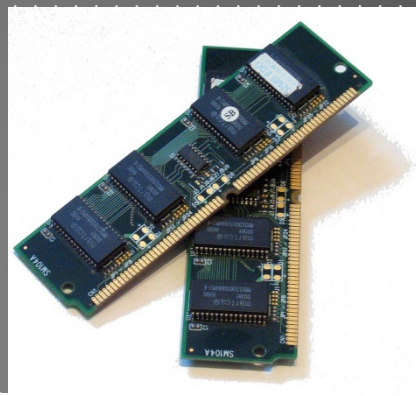
"ANTI-CACHING"-BASED ELASTIC MEMORY MANAGEMENT FOR BIG DATA

Hao Zhang [#], Gang Chen [†‡], Beng Chin Ooi [#],
Weng-Fai Wong [#], Shensen Wu [†‡], Yubin Xia [*]

[‡] yzBigData Co., Ltd.
[#] National University of Singapore
[†] Zhejiang University
[*] Shanghai Jiao Tong University

MOTIVATION

- In-memory databases for Big Data
- Memory never enough
  - Memory is still relatively scarce compared to HDD
  - Energy consumption
    - Memory is a significant contributor to the total system power
  - N-minute rule
    - cheaper to put the data in memory if it is accessed every N-minute
    - Cold data – stay on disk
    - Hot data – resident in memory

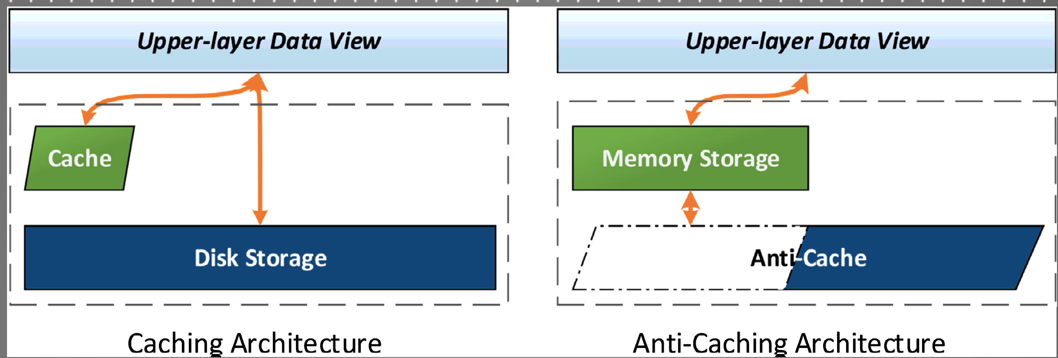


OUTLINE

- Caching vs. “Anti-caching”
- State-of-the-art Approaches
- Understanding the components of anti-caching
- User-space Virtual Memory Management (UVMM)
- Conclusions

CACHING VS. "ANTI-CACHING"

- Common
  - Deal with the same level of storages
- Difference
  - Assumption about the memory size
  - Target for different types of systems
  - Different primary data locations



THE COMPONENTS OF IN-MEMORY DB

- Access tracking
  - Granularity: Tuple vs page
- Eviction Strategy
  - LRU, MRU, CLOCK, WSCLOCK
- Book-keeping
  - Tables (hashed or otherwise), indexes, etc.
- Swapping strategy
  - What, how much, and when

USER VS KERNEL

- At user/application level
  - More semantics information
  - Different granularities (tuple, column, row, tables, page)
  - Platform-independence (possible)
- At kernel level
  - Directly use hardware
  - Only know pages
- Crossing the user-kernel divide is expensive

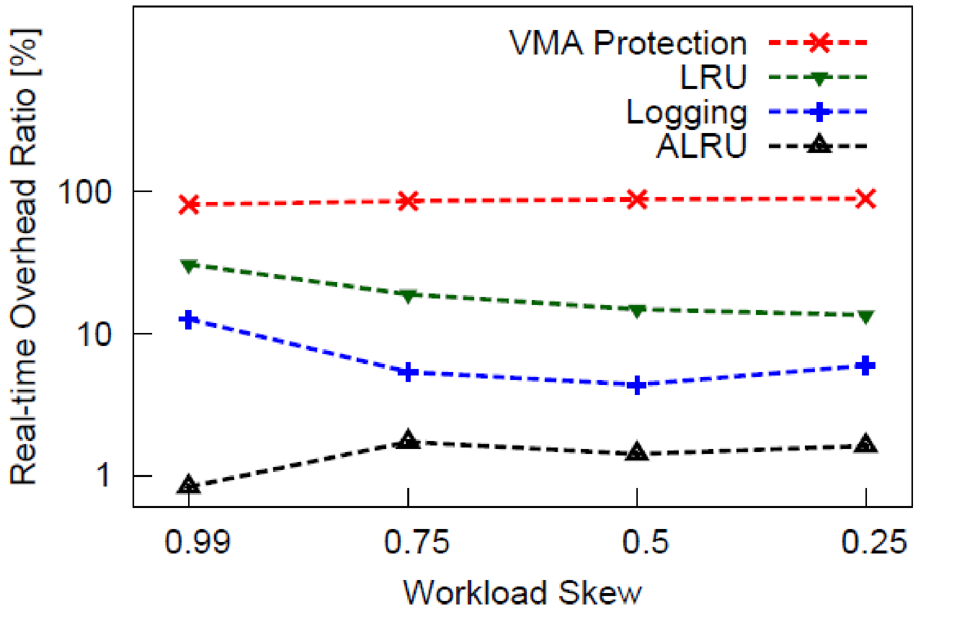
STATE-OF-THE-ART APPROACHES

Approaches	Access Tracking	Eviction Strategy	Book-keeping	Data Swapping
H-Store anti-caching	Tuple-level tracking	LRU	Evicted table and index	Block-level swapping
Hekaton Siberia	Tuple-level access logging	Offline classification	Bloom and range filter	Tuple-level migration
Spark	N/A	LRU based on insertion time	Hash table	Block-level swapping
Cache Systems	Tuple-level tracking	LRU, approximate LRU, etc	N/A	N/A
Buffer Management	Page-level tracking	LRU, MRU, CLOCK, etc	Hash table	Page-level swapping
OS Paging	h/w-assisted page-level tracking	LRU, NRU, WSCLOCK, PPRA, etc	Page table	Page-level swapping
Efficient OS Paging	Tuple-level access logging	Offline classification and OS Paging	OS-dependent	OS-dependent
Access Observer in Hyper	h/w-assisted page-level tracking	N/A	N/A	N/A

A DETAILED STUDY OF THE COMPONENTS

- Platform
  - Implement different approaches inside one system – Memcached
    - *To avoid interference introduced by other components*
    - *More fair to various approaches*
    - *Simple to monitor and perf*
- Benchmark
  - YCSB (synthetic)
    - Varying skewness
    - Varying ratio of available memory to data

ACCESS TRACKING OVERHEAD



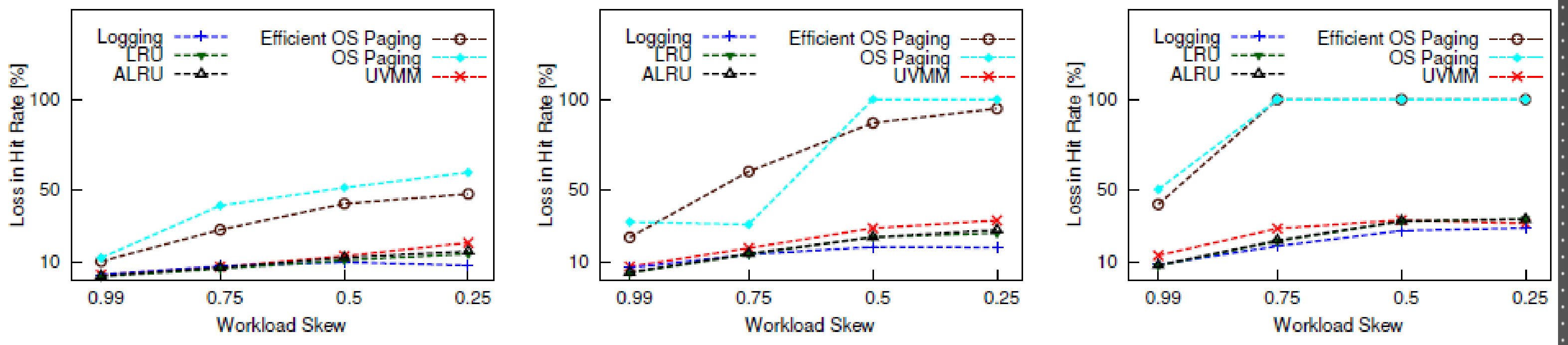
(d) Real Time Overhead

ACCESS TRACKING - INSIGHTS

- Virtual memory access (VMA) is very expensive

- If the average tuple size is less than 4-KB for doubly-linked LRU list, or 1-KB for ALRU, their memory overheads are much higher than that of page-table based tracking.

EVICTION STRATEGY



(a) Loss in Hit Rate ($\frac{\mathbf{M}}{\mathbf{D}} = \frac{3}{4}$) (b) Loss in Hit Rate ($\frac{\mathbf{M}}{\mathbf{D}} = \frac{1}{2}$) (c) Loss in Hit Rate ($\frac{\mathbf{M}}{\mathbf{D}} = \frac{1}{4}$)

EVICTION STRATEGY - INSIGHTS

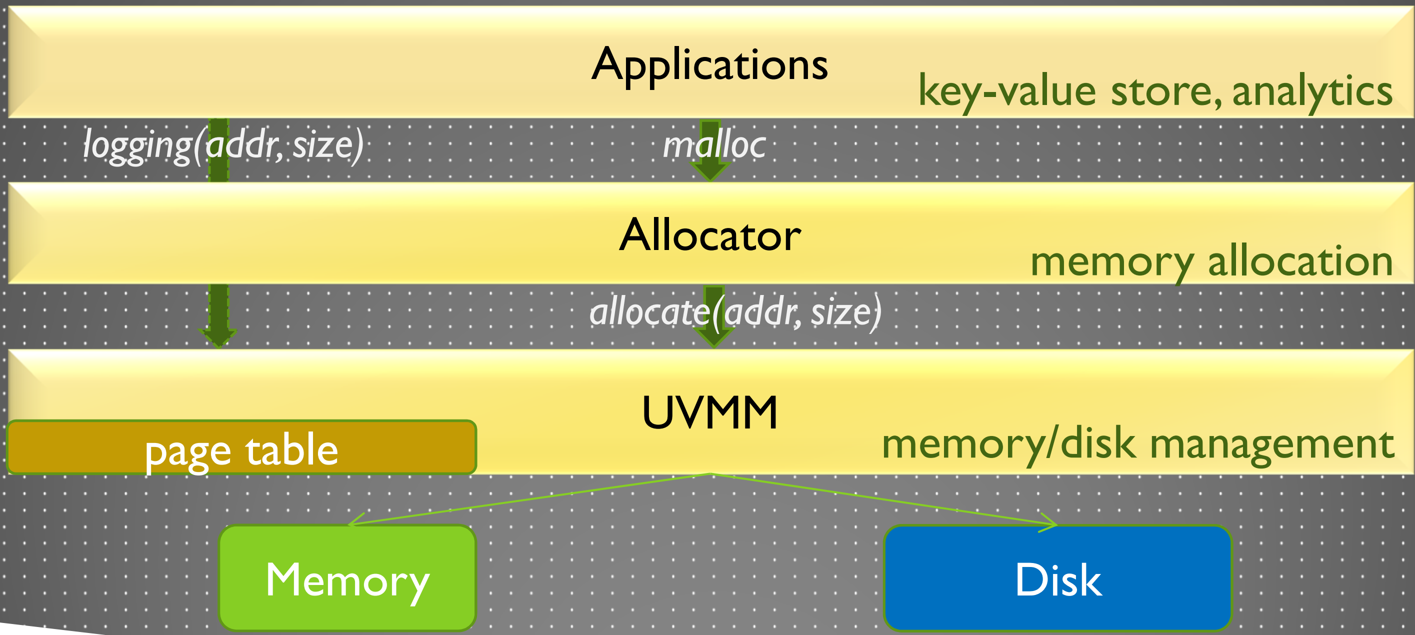
- Kernel-based eviction approaches suffer from poor accuracy
  - Lack of semantics information
- Access-logging based offline classification do well
- LRU/ALRU do reasonably well

BOOK-KEEPING - INSIGHTS

- Book-keeping using indexed eviction table has higher space overhead
- Bloom and other filters are quite space efficient
- Page tables and VMA use tables that are there anyway. So overhead is lowest.

TOWARDS AN EFFICIENT GENERAL APPROACH
- USER-SPACE VIRTUAL MEMORY MANAGEMENT (UVMM)

- Three-layer Hierarchy



TOWARDS AN EFFICIENT GENERAL APPROACH

- USER-SPACE VIRTUAL MEMORY MANAGEMENT (UVMM)

- Design Principles
  - No indirection
    - *Real pointer*
  - Non-intrusiveness
    - *Backward compatibility and transparent upgrading*
    - *API-based (e.g., malloc)*
  - Flexibility
    - *List of options for different levels of intrusiveness*
    - *Optional user-provided access logging*

TOWARDS AN EFFICIENT GENERAL APPROACH

- USER-SPACE VIRTUAL MEMORY MANAGEMENT (UVMM)

- Design Principles
  - Reduced CPU overhead for normal operations
    - *Page table*
    - *User-supplied access logging*
  - Reduced Memory overhead
    - *Page level tracking*
    - *Access distribution provided by logging (within one page)*

TOWARDS AN EFFICIENT GENERAL APPROACH
- USER-SPACE VIRTUAL MEMORY MANAGEMENT (UVMM)

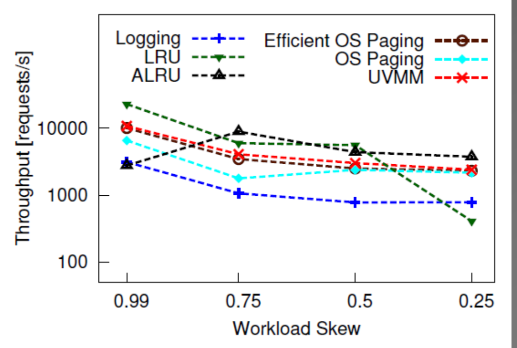
- Implementation
  - Access Tracking
    - *A combination of access tracking methods*
    - *Page table, malloc-inject, access logging, etc.*
  - Eviction Strategy
    - *Optimized LRU/WSCLOCK with consideration of user-provided access logging*
    - *Standard eviction strategies: aging-based LRU, WSCLOCK, FIFO, RANDOM*

TOWARDS AN EFFICIENT GENERAL APPROACH
- USER-SPACE VIRTUAL MEMORY MANAGEMENT (UVMM)

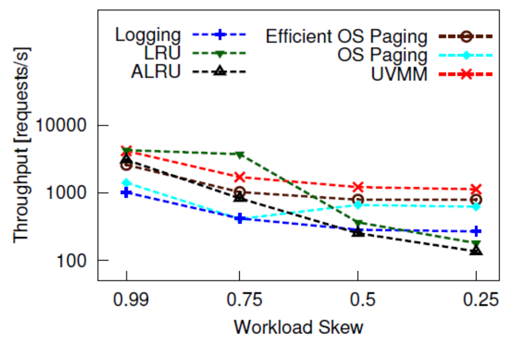
- Implementation
  - Book-keeping
    - *VMA protection*
  - Data swapping
    - *Compression – lz4*
    - *Kernel Asynchronous I/O*

PUTTING THEM TOGETHER

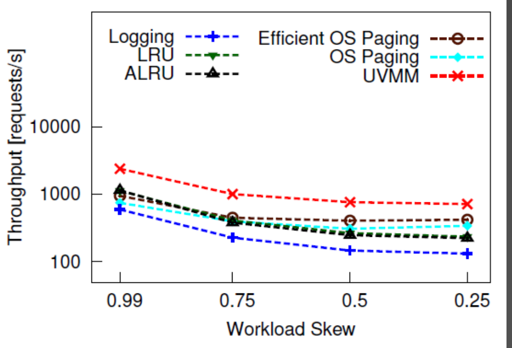
- Throughput



(a) Throughput ($\frac{\mathbf{M}}{\mathbf{D}} = \frac{3}{4}$)



(e) Throughput ($\frac{\mathbf{M}}{\mathbf{D}} = \frac{1}{2}$)



(i) Throughput ($\frac{\mathbf{M}}{\mathbf{D}} = \frac{1}{4}$)

LRU – H-store
OS Paging – standard
UVMM – our proposal

ALRU – Redis
Efficient OS Paging – Stoica & Ailamaki

Logging - Hekaton Siberia

CONCLUSIONS

- User- and kernel-space approaches exhibit different strengths
  - User-space: more application semantics, finer operation granularity, more accurate eviction strategy
  - Kernel-space: hardware (CPU, I/O) assistance, good resource utilization
- Combination of user- and kernel-space approaches needed for the best anti-caching performance
  - CPU, I/O performance, memory utilization
  - General and efficient
  - User-space virtual memory management (UVMM)

THANK YOU

Q&A