Anti-Caching: A New Approach to Database Management
System Architecture

Guide: Dr. Sunnie Chung
Helly Patel (2655077)
Kush Patel (2641883)
Abstract

- Earlier DBMS blocks stored on disk, with a main memory block cache.
- Uses locking to allow multiple transactions to access the database at the same time.
- Introduce anti-caching, to overcome limitations which stores cold data in the disk and the hot data in the Main Memory.
- Main memory is the primary storage device in the Anti Caching approach.
Introduction

- DBMSs invariably maintain a buffer pool of blocks in main memory for faster access.
  - When an executing query attempts to read a disk block, the DBMS first checks to see whether the block already exists in this buffer pool. If not, a block is evicted to make room for the needed one.

Problem:
- Cost of maintaining a buffer pool is nearly one-third of all the CPU cycles used by the DBMS.
- Improved performance is only achievable when the database is smaller than the amount of physical memory available in the system.
- Page fault occurs due to Main Memory access.
  Ex: H-Store doesn’t use locking
DBMS Architectures

- Disk Oriented DBMS
  The disk is primary storage for database and data is brought into main memory as needed.

- Disk Oriented DBMS with Distributed Cache
  Disk is primary storage and application first looks in the cache for the tuple of interest. If this tuple is not in the cache, then the application executes a query in the DBMS to fetch the desired data.
Main Memory DBMS with Anti-Caching

- Gathers the “coldest” tuples and writes them to disk with minimal translation
- Freeing up space for more recently accessed tuples.
- The “hotter” data resides in main memory
- The colder data resides on disk in the anti-cache portion of the system
- Similar to Virtual Memory Swapping
- LRU order is used for deciding cold and hot data.
- Ex: H-Store System
Anti-Caching over Virtual Memory

- Fine-grained Eviction
  - Tuple-level in Anti-Caching
  - Page-level in virtual memory.
  - In virtual memory, a single hot tuple cause the entire page to be hot and will be in memory.
  - Same level of granularity in Anti-Caching.

- Non-Blocking Eviction
  - In virtual memory, page fault occur. Every page fault triggers a disk read and multiple access to reads results in several page fault.
  - In anti-caching, Transaction is aborted and restarted later. Uses a pre-pass execution phase that attempts to identify all evicted block
H-Store System

- To overcome waiting for disk, H-Store on Main Memory-only node.
- Manages one or more partitions.
- Partition is assigned a single-threaded execution engine at its node that is responsible for executing transactions and queries for that partition.
- A client application initiates a transaction by sending a request to any node in the cluster.
- Transaction request contains the name of a stored procedure and the input parameters for that procedure’s control code.
  - Single-Partition Transactions
  - Multi-Partition Transactions
Anti-Caching Models

Storage Architecture

- The anti-cache storage manager within each partition contains three components:
  1. A disk-resident hash table that stores evicted blocks of tuples called the Block Table
  2. An in-memory Evicted Table that maps evicted tuples to block ids, and
  3. An in-memory LRU Chain of tuples for each table.
Anti-Caching Models

- Block Eviction & Transaction Execution
  - Single global ordering of tuples in the system, thus globally tracking hot and cold data.
  - If the transaction accesses evicted data, then the transaction enters pre-pass execution.
  - The goal of the pre-pass phase is to determine all of the evicted data that the transaction needs to access so that it can be retrieved together.
  - To evict data the DBMS must determine
    - (1) what tables to evict data from
    - (2) the amount of data that should be evicted from a given table
Anti-Caching Models

- Block Retrieval
  - Attempts to access evicted tuples, transaction aborted and the DBMS schedules the retrieval of the blocks.
  - The system first issues a non-blocking read to retrieve the blocks from disk.
  - When requested blocks are retrieved, the aborted transaction are rescheduled.

Key Issue:
- How much data to merge from a retrieved block back into the in-memory storage.

Solution:
Block Merging & Tuple Merging
Anti-Caching Models

- Distributed Transactions
  - H-Store will switch a distributed transaction into the “pre-pass” mode.
  - The aborted transaction are not requeued until blocks that it needs have been retrieved from the nodes in the cluster.

- Snapshots & Recovery
  - DBMS serializes the contents of the Evicted Table, and writes it to disk.
  - DBMS also makes a copy of the Block Table on disk.
  - To recover after a crash, the DBMS loads in the last snapshot from disk.
  - Due to snapshot it sets up the tables, indexes, Block Table, and Evicted Table as it existed before the crash.
Experiments & Results

- YCSB - Yahoo! Cloud Serving Benchmark
- Performed the experiment with workload mixtures:
  - Read-Heavy: 90% reads / 10% updates
  - Write-Heavy: 50% reads / 50% updates
  - Read-Only: 100% reads
- Performed with skew workloads to control how often a tuple is accessed by transactions.
- Results:
  Anti-caching provides a 7 times improvement in throughput over the other architectures.
Conclusion

- New architecture for managing datasets that are larger than the available memory while executing OLTP workloads.

- With anti-caching, memory is the primary storage and cold data is evicted to disk.

- Cold data is fetched from disk as needed and merged with in-memory data while maintaining transactional consistency.

- For skewed workloads anti-caching has an 8*17 performance advantage over a disk-based DBMS and a 2*9 a disk-based system fronted with a distributed main memory cache.
References: