Mnemo:
Boosting Memory Cost Efficiency in Hybrid Memory Systems

Thaleia Dimitra Doudali
Ada Gavrilovska
Problem Space
Cost of memory dominates in the cloud.

Big Data Analytics

In-Memory Key-Value Stores

Memory Optimized Virtual Machines with up to few TBs of memory.

Volatile Memory (DRAM)
e.g. DDR-4

60% - 85% of the hourly VM cost!

Non Volatile Memory (NVM)
e.g. Intel Optane DC Persistent Memory
(Will be available in Google Cloud instances with capacity up to 7 TBs)

Memory Technology Options

cheaper but slower to access than DRAM
Problem Statement
Capacity Sizing of the Hybrid Memory Components.

Facts:
- Future clouds will feature hybrid memory components.
- These components have different cost and access latencies.

Problem:
What is the ideal capacity ratio between the hybrid memory components?

Goals:
- Maximize system’s cost efficiency.
- Keep performance guarantees.
Existing Solutions
Data Tiering over *fixed* capacities.

Existing solutions for data tiering over fixed capacities:

- **In-Memory Key-Value Stores**
- **Data Tiering over fixed capacities.**

**Profiling Tool** decides:

1. **Which keys are hot/cold?**
   - Track every single memory access per key.

2. **Which keys to place/move to DRAM?**
   - Estimate performance benefit from DRAM placement.
   - Use analytical model with performance baselines.

Output: Data Tiering across DRAM and NVM.

If different DRAM:NVM capacity ratio:

1. Run profiling again to get a new tiering.
2. Run application and observe change in performance.

**Problem:** Currently no way to know how much DRAM vs NVM to use.
Solution Preview

Mnemo

Offline Profiling Tool for in-memory Key-Value Stores

- **Data Tiering**
  Which keys should be allocated in DRAM vs NVM?

- **Capacity Sizing**
  How many keys should be allocated in DRAM, so that application performance remains high, but memory cost remains low?

Mnemo quickly generates an accurate trendline of application performance for incremental DRAM to NVM capacity ratio, thus memory cost.
Motivation

Which parameters affect key-value store performance over hybrid memory systems?

- More keys in DRAM
- Increasing DRAM capacity
- Increasing memory cost
Motivation

Which parameters affect key-value store performance over hybrid memory systems?

- More keys in DRAM
- Increasing DRAM capacity
- Increasing memory cost

Varying key access pattern
Motivation

Which parameters affect key-value store performance over hybrid memory systems?

![Graph showing throughput increase vs. % of DRAM-only Cost]

- More keys in DRAM
- Increasing DRAM capacity
- Increasing memory cost

Varying read:write request ratio

- More reads

Application
Performance
Increase
Which parameters affect key-value store performance over hybrid memory systems?

Motivation

- More keys in DRAM
- Increasing DRAM capacity
- Increasing memory cost

Application

- Performance
  - Increase

Varying key-value size

Trending Preview

- thumbnail
- text post
- photo caption
- estimate

Throughput increase (%)

% of DRAM-only Cost

20 28 36 44 52 60 68 76 84 92 100

bigger values

More keys in DRAM
Increasing DRAM capacity
Increasing memory cost
Motivation

Which parameters affect key-value store performance over hybrid memory systems?
- Key access pattern
- Read:Write requests
- Key-Value sizes

These parameters determine the shape of the curve.
The height of the curve also depends on the latency difference in accessing DRAM vs NVM.

Takeaways:
In order to estimate performance we’ll need to capture:
- The workload parameters.
- Performance baselines for DRAM vs NVM.
Mnemo Usage

User Input

**Mnemo**
Generates the performance-to-cost trendline

- Fast
- Lightweight
- Accurate

Provides just a workload description

- Minimal User Effort
- No Application Modifications

Output to User

User chooses the sweet spot that brings the desired performance under his cost budget.

- Maximum Cost Efficiency
- Desired performance levels

User Input

Minimal User Effort

No Application Modifications
**Detailed Design**

- **Sensitivity Engine**: Workload Execution over DRAM-only and NVM-only.
  - Key Request Pattern
  - Key-Value Sizes
  - DRAM-NVM price difference

- **Pattern Engine**: Key prioritization for DRAM allocation.
  - weight = # accesses / key-value size

- **Estimate Engine**: Generates performance estimate across the key space.
  - Keys ordered by weight
  - Throughput
    - No. of keys in DRAM
    - Increasing DRAM capacity
    - Increasing memory cost

- **Placement Engine**: Populates the servers with the selected data tiering.
  - Key-Value pairs
  - All-data-in-DRAM
  - All-data-in-NVM
  - Read Time Diff (NVM-DRAM)
  - Write Time Diff (NVM-DRAM)
  - User choice
  - Throughput Estimate
    - Reads x + Writes x
    - Reads + Writes

*Sweet spot*
Evaluation Methodology

Metrics:
Estimate Accuracy❓ Cost Efficiency❓ Profiling Overhead❓

Implementation:

- synthetic workloads
- Facebook-like actions varying key access pattern, read:write ratio, value sizes
- modified client
  - request to server with DRAM
- unmodified server
  - Redis
  - Memcached
  - DynamoDB
  - DRAM
- actual hardware
  - NVM (emulated)
Evaluation Results
Mnemo successfully captures the trade-off between performance and cost
Evaluation Results
Mnemo allows for significant cost reductions

The higher the better

For chosen performance sweet spot to be 10% slowdown from all-data-in-DRAM.
Evaluation Results
Mnemo is extremely accurate

- Estimate Accuracy
  - memcached: 0.66%
  - redis: -0.59%
  - dynamodb: 0.06%
  - overall: 0.07%

Error (%)
In-Memory Key-Value Stores

key access pattern

Mnemo

Hot keys

Cold keys

Performance

Hybrid Memory Cost

Sweet spot

1. Which keys are hot/cold?
Extract access information from workload description.

2. Which keys to place/move to DRAM?
Get performance baselines through workload execution.
Use analytical model with performance baselines.

How much capacity of each memory type?

Lightweight

Accuracy

Profiling Speed

Fast

No code modifications. Workload description.

✅ Lightweight

✅ Accurate

✅ Fast
Summary

Mnemo

- hot keys
- cold keys

- DRAM
- NVM

How much capacity of each memory type?

- Fast
- Lightweight
- Accurate
- Open source

https://github.com/Thaleia-DimitraDoudali/mnemo