Exploring Replacement Policy for Memcached

Gi Lee, Byoung Jin Kim, and Eui-Young Chung
Department of Electrical and Electronic Engineering
Yonsei University
Seoul, Republic of Korea
leegi01713@yonsei.ac.kr, bryankim@yonsei.ac.kr, eychung@yonsei.ac.kr

Abstract—**Caching efficiency is critical to performance when an in-memory database system acts as a cache for a disk-oriented database system. *Memcached*, one of the widely used in-memory database that stores key-values, roles as a cache for high-speed data storage layer. It replaces items using LRU replacement policy when memory capacity is used up. Other than LRU policy for maximizing temporal locality, in this paper, we confirm the impact of replacement policies for workloads with random pattern operations. We analyze hit counts of workload by applying BIP and SRRIP replacement policies to *Memcached*. Experimental results show that hit counts of *Memcached* to which SRRIP is applied is improved by up to 12%.**

Keywords; In-memory database system; Replacement policy

I. INTRODUCTION

In-memory database systems have been widely used for high-speed data storage layer. Disk-oriented database systems have limitations of long access latency caused by slow storage devices. On the other hand, in-memory database systems leverage fast memory devices for higher performance. However, the database size of in-memory database systems can be limited by the capacity of memory devices. When an in-memory database system acts as a cache for a disk-oriented database system, caching efficiency is critical to maximizing performance with limited capacity.

Memcached is one of the well-known open-source in-memory database systems, which serves as a cache[1]. When *Memcached* has no more memory space to store data, some of items are flushed according to the replacement policy. The replacement policy implemented in *Memcached* is Least Recently Used(LRU) which could be inefficient with real workloads.

LRU policy assumes that data that have not been referenced for the longest time is less likely to be used in the future. In a real database system, however, workloads that have random access patterns dominate. This means that LRU policy cannot be effective on real system workloads because the temporal locality is less affected. In this paper, we apply various replacement policies to *Memcached* and analyze hit counts and performance.

II. BACKGROUND

This section introduces Bimodal Insertion Policy(BIP) and Static Re-Reference Interval Prediction(SRRIP) policy among the various replacement policies. In our experiment, BIP and SRRIP are selected as the replacement policy of *Memcached*.

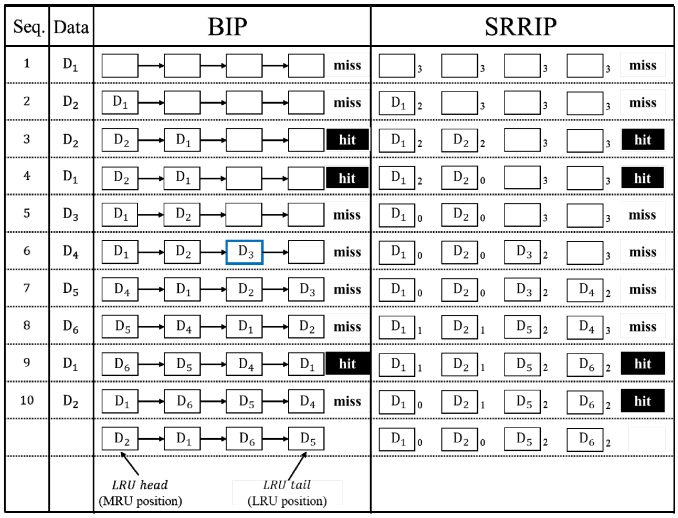

Figure 1. Operation example of BIP(left) and SRRIP(right)

A. Bimodal Insertion Policy(BIP)

BIP has an efficiency of cache for not only temporal locality but also random characteristics[2]. BIP inserts newly cached data at LRU position with a certain probability, *bimodal throttle parameter*(ε), unlike LRU policy that always inserts data in Most Recently Used(MRU) position. When the cache is full, data located at LRU position are evicted same as the LRU policy. When accessing to the data already in the list, BIP updates the list with deleting data from the list and reinserting them according to the *bimodal throttle parameters*. Fig. 1-left shows an example of detailed BIP operation.

B. Static Re-Reference Interval Prediction(SRRIP)

SRRIP replaces data considering re-reference interval[3]. SRRIP keeps data that have a high possibility of reuse based on the *Re-Reference Prediction Value*(RRPV) and replaces the oldest data with the lowest possibility of re-reference. If cache hits, SRRIP modifies the RRPV of data to '0'. The RRPV of '0' implies that the cache block is predicted to be re-referenced in the near-immediate future while the highest RRPV means that the cache block is predicted to be re-referenced in the distant future. An example of SRRIP which has the RRPV as 3 is shown in Fig. 1-right.

2020 International SoC Design Conference (ISOCC) | 978-1-7281-8331-2/20/$31.00 ©2020 IEEE | DOI: 10.1109/ISOCC50952.2020.9332942

III. EVALUATION

We analyze the characteristics of LRU/BIP and SRRIP policy for workloads with random data access patterns. To evaluate, we implemented BIP and SRRIP on *Memcached*.

A. Experimental settings

We run open-source *Memcached* on the local system for all experiments in the paper. Implemented replacement policies on *Memcached* are BIP and SRRIP. More details of the experiment settings are shown in Table 1.

TABLE I. DETAILED EXPERIMENTAL SETTINGS

Parameters	Properties
bimodal throttle parameter(ε)	1/32
RRPV	3
DRAM size	64MB
Total operation counts	100K
Value size	From 1K to 100K

The workload used in the experiment is the dataset of Yahoo Cloud Serving Benchmark(YCSB)[4]. Workload B of YCSB, which is a read-heavy workload, uses Zipfian distribution to generate loads. Zipfian distribution guarantees that the generated items have randomness. We experimented while adjusting the portion of 'get' command and 'update' command for workload B as shown in Table 2.

TABLE II. WORKING SET OF YAHOO CLOUD SERVING BENCHMARK

Set	'get' ratio(%)	'update' ratio(%)
B1	90	10
B2	95	5
B3	99	1

B. Experimental Results

Fig. 2 shows total hit counts of each workload when LRU/BIP and SRRIP are applied. Compared to LRU policy, hit counts of BIP have no noticeable increment. Otherwise, hit counts of SRRIP increased obviously for all workloads. For workload B2, specifically, you can see that hit counts in SRRIP are about 12% higher than the hits in LRU policy.

Fig. 3 shows the throughput of each workload. Throughput refers to the total number of operations over the overall time. If data are hit about 'get' command, *Memcached* processes data in memory and returns them to the client. In contrast, if data are miss, there is no return value processing in *Memcached*. Therefore, low throughput means there are more successful operations to be processed in memory. This indicates that the caching efficiency of *Memcached* has been improved and that *Memcached* is acting properly as a cache. As shown in Fig. 2 and Fig. 3, SRRIP with more hit counts has higher performance than other replacement policies.

Experimental results show the performance of SRRIP is effective for all workloads with random data access patterns. BIP, which considers both temporal locality and random characteristic, has no significant difference compared to LRU

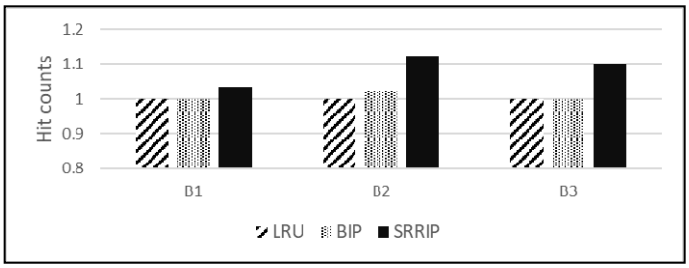

Figure 2. Memcached total hit count of YCSB workloads

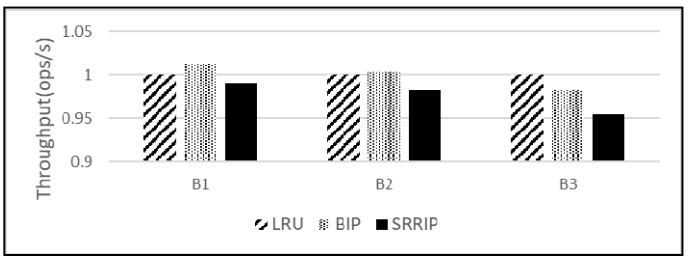

Figure 3. Throughput(ops/s) of YCSB workloads

policy. This is because the consideration of temporal locality is dominant in BIP. SRRIP adds one metric for data access to predict data access patterns, so it has considerable caching efficiency for random pattern workloads.

IV. CONCLUSION

This paper explores replacement policies on *Memcached*. In a real database system, data are accessed randomly because many clients load and store different data. Conventional *Memcached* has LRU replacement policy that can be only suitable with temporal locality. If SRRIP is applied to *Memcached* that is used as a real database system, it can serve as a cache more effectively than a conventional *Memcached* for real workloads with random characteristics.

ACKNOWLEDGMENT

This research was supported by the MOTIE(Ministry of Trade, Industry & Energy) (10080722) and KSRC(Korea Semiconductor Research Consortium) support program for the development of the future semiconductor device. And the chip fabrication and EDA Tool were supported by the IC Design Education Center.

REFERENCES

[1] Fitzpatrick, B. "Distributed caching with memcached." *Linux journal* 124 (2004).

[2] Qureshi, M. K., Jaleel, A., Patt, Y. N., Steely, S. C., & Emer, J. "Adaptive insertion policies for high performance caching." *ACM SIGARCH Computer Architecture News* 35.2 (2007): 381-391.

[3] Jaleel, A., Theobald, K. B., Steely Jr, S. C., & Emer, J. "High performance cache replacement using re-reference interval prediction (RRIP)." *ACM SIGARCH Computer Architecture News* 38.3 (2010): 60-71.

[4] Cooper, B. F., Silberstein, A., Tam, E., Ramakrishnan, R., & Sears, R. "Benchmarking cloud serving systems with YCSB." *Proceedings of the 1st ACM symposium on Cloud computing*. 2010