A Log-Free and Consistent Chained Hashing for Non-volatile Memory

Renzhi Xiao, Dan Feng*, Yuchong Hu
Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology,
Huazhong University of Science and Technology, {rzxiao, dfeng, yuchonghu}@hust.edu.cn,
Corresponding author: Dan Feng (dfeng@hust.edu.cn).

ABSTRACT

Hashing is an in-memory index that provides fast search performance because it is a flat structure that can quickly locate a key-value item position through a hash function. Non-volatile memory (NVM) technologies have promoted the researches of NVM-based hashing structures since NVM has the durability of hard disk and the performance of DRAM. However, current NVM-based hashing schemes must tackle data inconsistencies and avoid rapid wear due to NVM write reorder and limited endurance. In this paper, we propose a log-free and consistent chained hashing for NVM, called ConHash. The ConHash leverages log-free failure-atomic writes to degrade the overhead of data consistency and exploits the pre-allocation of the bucket in the chain to solve hash conflicts efficiently. ConHash uses the size function to get the number of keys in the hashing instead of the fixed count variable, which may cause the fast wear problem for NVM. Compared with the state-of-the-art schemes, experimental results demonstrate that our ConHash decreases the insertion latency up to 88.2% and degrades the deletion latency up to 82.6%.

1 INTRODUCTION

Hashing data structures are wildly used in in-memory key-value stores, such as RAMCloud, Memcached, Redis due to their constant search performance (O(1)). However, Hashing structure may encounter hash collision which consumes a lot of system sources like CPU and memory. Four main solutions for hash conflicts shown in Figure 1.

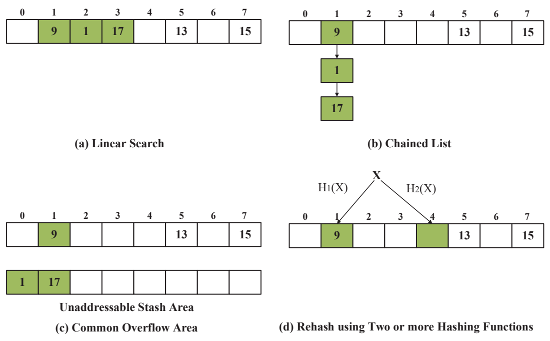


Figure 1: Four main solutions for hash conflicts.

Non-volatile Memory technologies (NVMs) have the performance of DRAM and the persistency of hard disk (HDD) or solid-state disk (SSD) with limited write endurance and higher write latency than reading. In recent years, academic communities have proposed hashing-based structures for NVM, such as PFHT [1], Path hashing [5], Group Hashing [4], and Level Hashing [6]. PFHT and Path hashing mainly solve the write problems but neglect the data inconsistency problem. Therefore, we re-implement PFHT and Path hashing with the logging method to guarantee data consistency. Moreover, these NVM-based hashing schemes suffer from rapid wear due to the local fixed count in NVM.

Therefore, we need to tackle the consistency problem and write problems. In this paper, we propose a novel log-free and consistent chained hashing for NVM called ConHash. The ConHash leverages log-free failure-atomic writes instead of the redo/undo logging method to degrade the overhead of data consistency and exploits the pre-allocation of a bucket in the chain to solve hash conflicts efficiently. ConHash uses the size function to get the number of keys in the hashing instead of the fixed count variable, which may cause the fast wear problem for NVM. We have implemented ConHash and evaluated it using two micro-benchmarks. Experimental results show our proposed ConHash outperforms state-of-the-art schemes up to 88.2% in the insertion latency, 82.6% in the deletion latency, respectively.

The rest of this paper is organized as follows: Section 2 presents the design of ConHash. Section 3 demonstrates performance evaluation.

2 THE DESIGN OF CONHASH

In this section, we describe the design of the ConHash, which is a log-free and consistent NVM-based persistent chained hashing scheme.

2.1 Design Goals

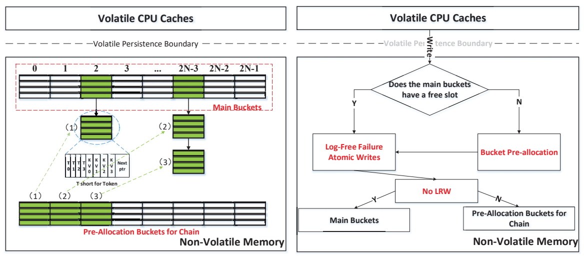
Our design goals of ConHash are as follows:

- **Low data consistency overhead**. ConHash leverages **log-free failure-atomic writes** instead of the logging method to guarantee data consistency of NVM-based hashing.
- **Good bucket allocation in a chain for hash conflicts**. ConHash uses **bucket pre-allocation** technique to separate the bucket allocation operation from the write critical path.
- **No local rapid wear**. ConHash exploits the size function to eliminate fixed count writes in NVM.

2.2 The Overview of ConHash

Figure 2.(a) shows the layout of ConHash, and Figure 2.(b) describes the write flow of ConHash.

2.2.1 Log-Free Failure-Atomic Writes. Redo/undo logging use for write units are larger than 8 bytes to guarantee data consistency. However, they suffer double NVM writes, which damage NVM


Volatile CPU Caches
Volatile Persistence Boundary
0 1 2 3 ... 2N-3 2N-2 2N-1
Main Buckets
(1) (2) (3)
T short for Token
Pre-Allocation Buckets for Chain
Non-Volatile Memory
Write
Does the main buckets have a free slot
Y
N
Log-Free Failure Atomic Writes
Bucket Pre-allocation
No LRW
Main Buckets
Pre-Allocation Buckets for Chain

(a) The layout of ConHash **(b) The write flow of ConHash.**

Figure 2: Overview of ConHash.

life and increase write latency. Copy-On-Write (COW) cause unnecessary reads for unmodified data area (Read amplification) and unnecessary writes for unpolluted data area (Write amplification). Log-Free Failure-Atomic Writes, short for Atomic, are ordered 8-byte atomic writes with memory fence and cache line flush instructions instead of logging. Atomic brings no extra NVM writes for insertion and deletion operations, lazy deletion with an invalid token. Therefore, Atomic can increase the lifetime of NVM and improve the performance of writes.

2.2.2 Bucket Pre-Allocation. The Bucket Pre-Allocation (BPA) technique first pre-allocates a certain number of buckets for hash conflicts and separates bucket allocation operations from critical paths for write operations. Therefore, BPA decreases the write latency. However, original chained hashing temporarily allocates a bucket from the heap, which increases insertion latency due to bucket allocation operation in the write critical path.

2.2.3 No Local Rapid Wear (LRW). Previous NVM-based hashing schemes use the fixed count to calculate the number of items in NVM which Shorten the lifetime of NVM with unbalance writes. No LRW leverages size function to eliminate fixed count writes in NVM, which can avoid local fast wear through more reading and increase the lifetime of NVM.

3 PERFORMANCE EVALUATION

In this section, we evaluate request latencies for different NVM-based hashing schemes.

3.1 Experimental Setup

We emulate NVM using Hewlett Packard's Quartz [2], which has been widely used [3, 6]. Our server configurations are the same as this work [3]. We use two micro-benchmarks in our experiments are RandomNum [3–5] and SequenceNum [3]. The NVM read and write latency is set by default to 200 and 600 nanoseconds.

3.2 Insertion Latency

As shown in Figure 3, ConHash shows lower insertion latency than Level (up to 43.2%) and PFHT-Log (up to 88.2%) with a low data consistency guarantee. ConHash demonstrates the lowest insertion latency both in RandomNum and SequenceNum.

3.3 Deletion Latency

As shown in Figure 4, ConHash shows lower deletion latency than Level (up to 50.8%) and PFHT-Log (up to 82.6%). ConHash demonstrates the lowest deletion latency.

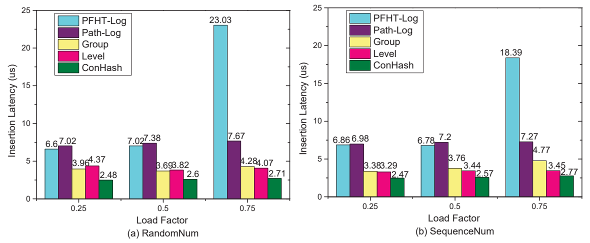


Figure 3: Average latency when inserting a key-value item.

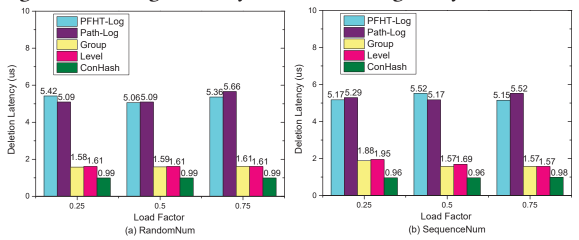


Figure 4: Average latency when deleting a key-value item.

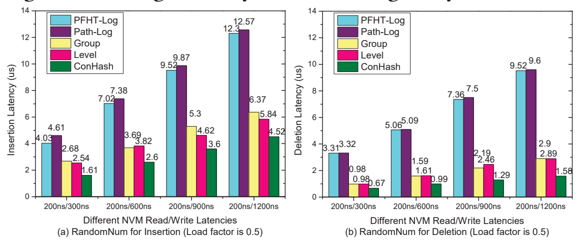


Figure 5: The average insertion and deletion latency of a key-value pair request with different NVM write latencies.

3.4 Effect of NVM Write Latency

As NVM write latency increases, various hashing insertion and deletion latencies increase accordingly shown in Figure 5. ConHash shows the lowest insertion and deletion latencies with different NVM write latencies (e.g., 300/600/900/1200 nanosecond).

REFERENCES

[1] Biplob Debnath, Alireza Haghdoost, Asim Kadav, Mohammed G Khatib, and Cristian Ungureanu. 2016. Revisiting hash table design for phase change memory. *ACM SIGOPS Operating Systems Review* 49, 2 (2016), 18–26.

[2] Haris Volos, Guilherme Magalhaes, Ludmila Cherkasova, and Jun Li. 2015. Quartz: A lightweight performance emulator for persistent memory software. In *Middleware*. ACM, New York, NY, USA, 37–49.

[3] Renzhi Xiao, Dan Feng, Yuchong Hu, Fang Wang, Xueliang Wei, Xiaomin Zou, and Mengya Lei. 2021. Write-Optimized and Consistent Skiplists for Non-Volatile Memory. *IEEE Access* 9 (2021), 69850–69859.

[4] Xiaoyi Zhang, Dan Feng, Yu Hua, Jianxi Chen, and Mandi Fu. 2018. A Write-efficient and Consistent Hashing Scheme for Non-Volatile Memory. In *ICPP*. ACM, New York, NY, USA, 87.

[5] Pengfei Zuo and Yu Hua. 2017. A write-friendly hashing scheme for non-volatile memory systems. In *MSST*. IEEE, Piscataway, NJ, USA, 1–10.

[6] Pengfei Zuo, Yu Hua, and Jie Wu. 2018. Write-optimized and high-performance hashing index scheme for persistent memory. In *OSDI*. USENIX, Berkeley, CA, USA, 461–476.