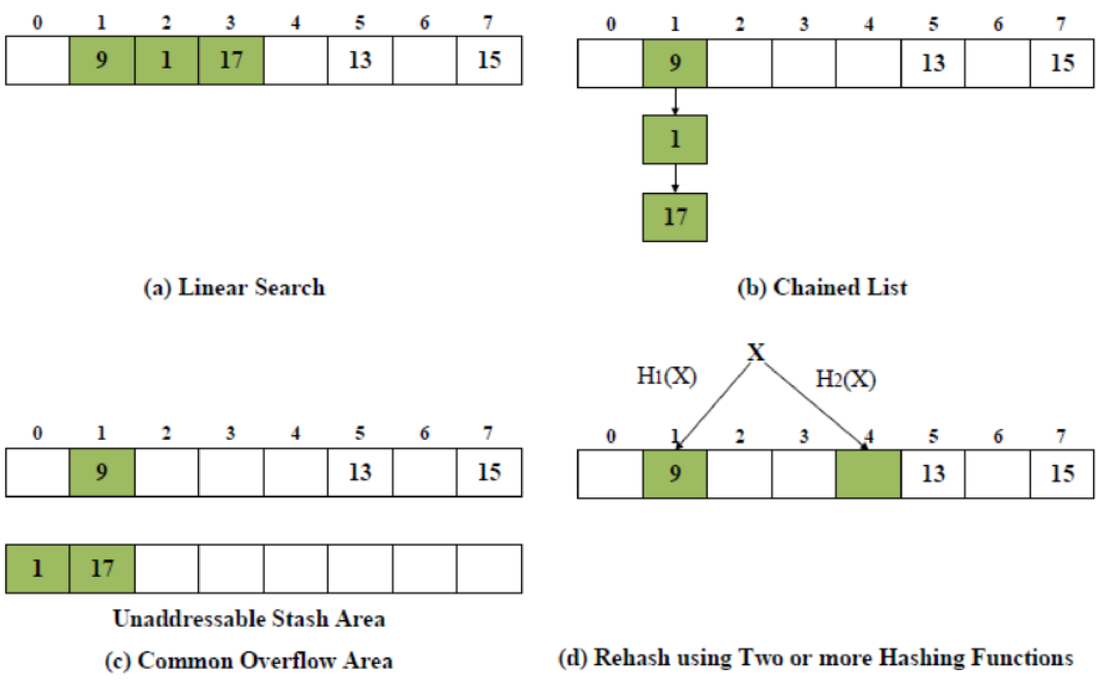
A Log-Free and Consistent Chained Hashing for Non-volatile Memory

Renzhi Xiao, Dan Feng, Yuchong Hu
Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, China

Background and Challenges

- **NVM-based Hashing**
  - 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))
  - Solutions for Hash Conflicts

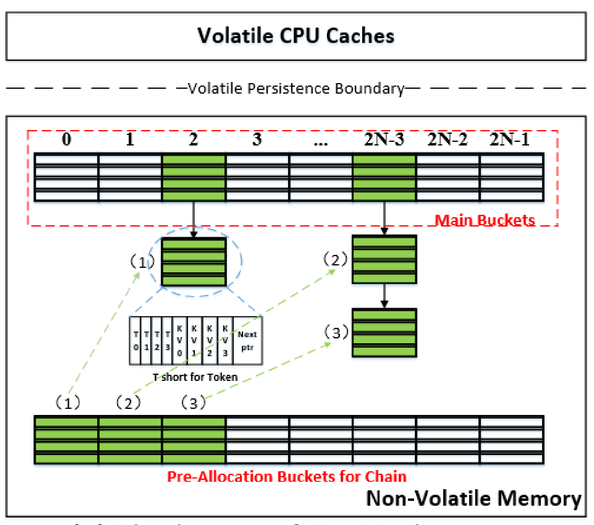


  - NVMs have the performance of DRAM and the persistency of hard disk (HDD) or solid state disk (SSD) with limited write endurance and higher write latency than reading
  - Current NVM-based hashing studies include PFHT [1], Path [5], Group [4], Level [6], etc
- **Challenges of NVM-based Hashing**
  - Data inconsistencies due to NVM write reorder or partial write in the face of system failure
  - Rapid wear due to the local fixed count in NVM

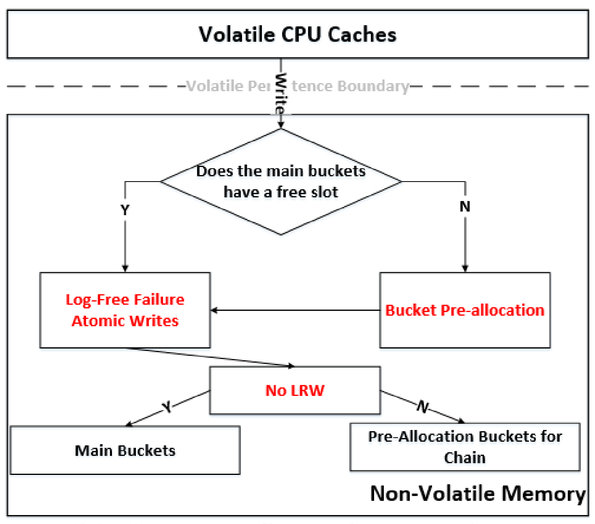
1. Expensive data consistency (redo/undo logging requires double NVM writes)
2. Low lifetime of NVM (limited to local rapid wear, short for LRW)

Proposed: Atomic, Pre-allocation and No LRW

- **We propose a novel log-free and consistent chained hashing for non-volatile memory, called ConHash :**
  - Log-Free Failure-Atomic Writes, short for Atomic
  - Bucket Pre-Allocation
  - No Local Rapid Wear (LRW)



(a) The layout of ConHash



(b) The write flow of ConHash

Log-Free Failure-Atomic Writes

- **Redo/undo logging**
  - Useful for write unit larger than 8 bytes for data consistency
  - Double NVM writes damage NVM life and increase write latency
- **Copy-On-Write (COW)**
  - Unnecessary reads for unmodified data area (Read amplification)
  - Unnecessary writes for unmodified data area (Write amplification)
- **Our approach (Log-Free Failure-Atomic Writes, Atomic)**
  - Ordered 8-byte atomic writes with memory fence and cache line flush instructions instead of logging
  - No extra NVM writes for insertion and deletion operations
  - Lazy deletion with invalid token
  - Increase lifetime of NVM and improve the performance of writes

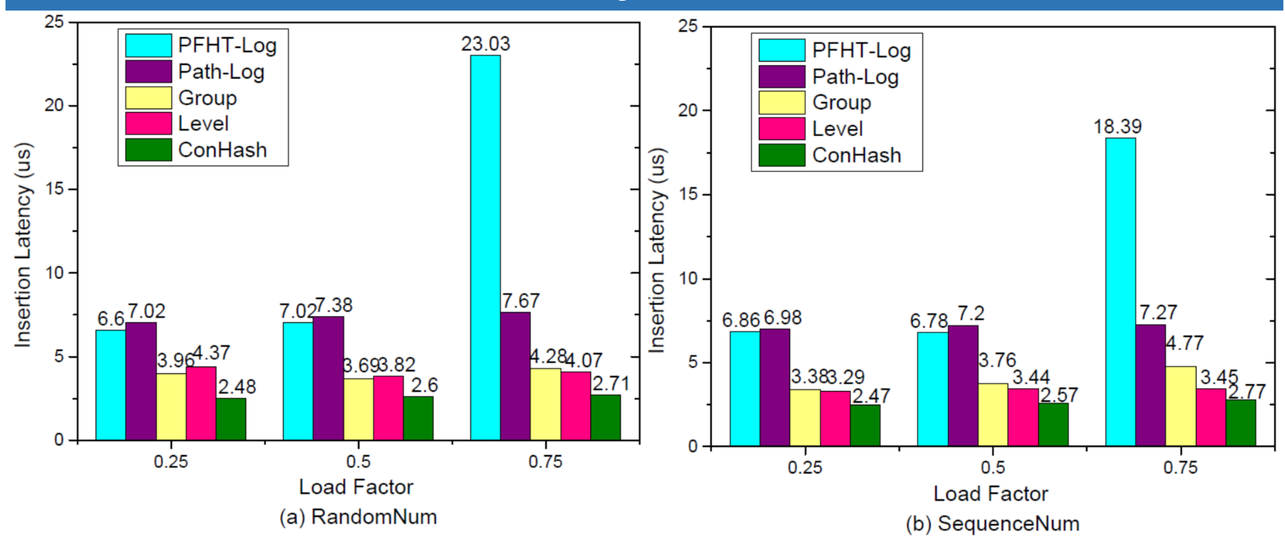
Bucket Pre-Allocation

- **Original Chain-based hashing**
  - Temporary allocation of a bucket from heap to the chain during a hash conflict
  - Bucket allocation operation in the critical path increase the insertion latency
- **Our method (Bucket Pre-Allocation)**
  - Pre-allocate a certain number of buckets for hash conflicts
  - bucket allocation operations are separated from critical paths for write operations, thus reducing the insertion latency

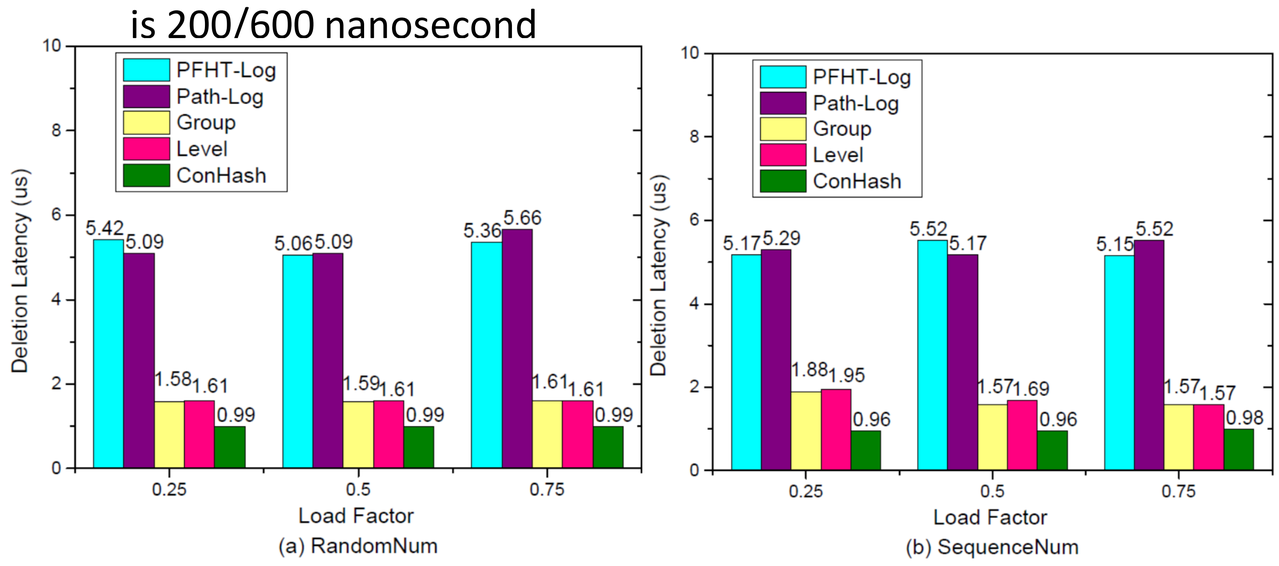
No LRW

- **Original NVM-based hashing**
  - Use the fixed count to calculate the number of items in NVM
  - Shorten the lifetime of NVM with unbalance writes
- **Our method (No LRW)**
  - Eliminate fixed count writes in NVM with size function
  - Avoid local fast wear with more reading
  - Increase the lifetime of NVM for ConHash

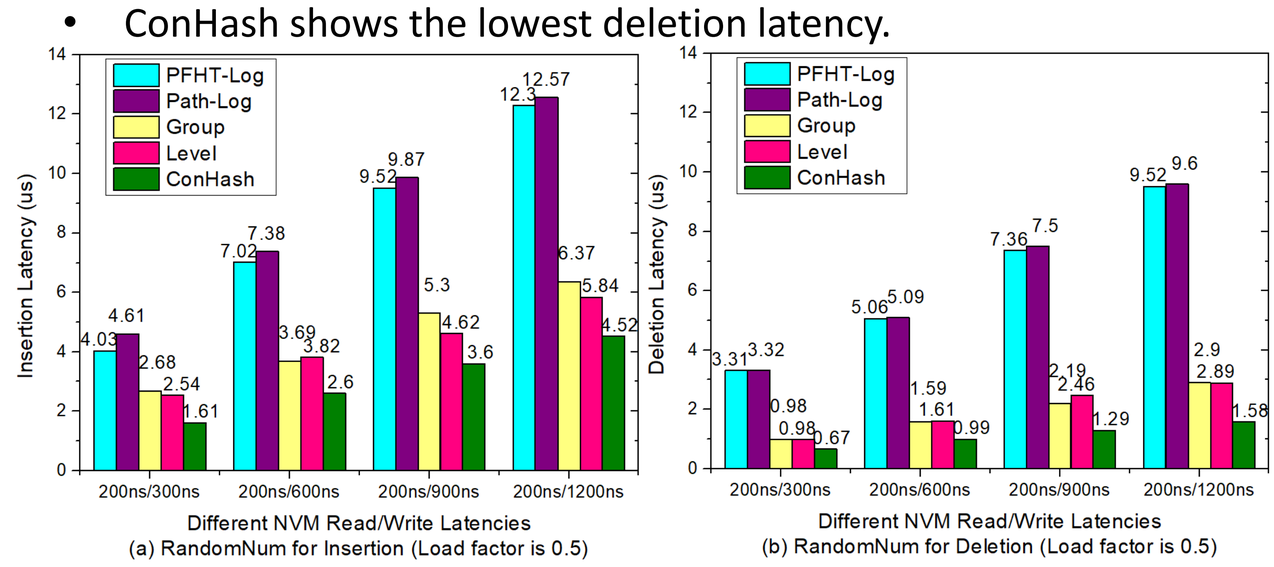
Preliminary Evaluations



- ConHash shows lower insertion latency than Level (up to 43.2%) and PFHT-Log (up to 88.2%) with low data consistency guarantee
- ConHash shows the lowest insertion latency both in RandomNum and SequenceNum when NVM read/write latency is 200/600 nanosecond



- ConHash shows lower deletion latency than Level (up to 50.8%) and PFHT-Log (up to 82.6%)
- ConHash shows the lowest deletion latency.



- As NVM write latency increases, various hashing insertion and deletion latencies increase accordingly.
- ConHash shows the lowest insertion and deletion latencies with different NVM read/write latencies.

References: https://xiaorz.github.io/refersOfConHash.html