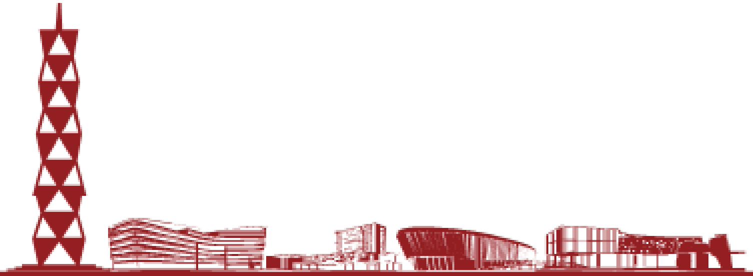
Exploring Architectural Implications to Boost Performance for in-NVM B+-tree

Yanpeng Hu; Qisheng Jiang; Chundong Wang

ShanghaiTech University, Shanghai, China



Outline

- **Introduction & Background**
- **Motivation**
- **Design of Conan**
    - **VIPT cache’s architectural implication**
    - **Conflict-aware node allocation**
    - **Implementation and discussion**
- **Evaluation**
- **Conclusion**

Introduction & Background

- Non-Volatile Memory (NVM) :
  - A recent significant change at the CPU side is the availability of eADR
  - eADR could flush cache lines back to NVM on a power fail with an uninterruptible power supply

- B+ tree in NVM:
  - FAST-FAIR
  - LB+-tree
  - Circ-Tree

Outline

- Introduction & Background
- **Motivation**
- Design of Conan
  - VIPT cache’s architectural implication
  - Conflict-aware node allocation
  - Implementation and discussion
- Evaluation
- Conclusion

Motivation

- eADR frees programmers from explicit cache line flushes

Motivation

- eADR frees programmers from explicit cache line flushes
- The no need of flushing cache lines results in a change in the proportions of time cost

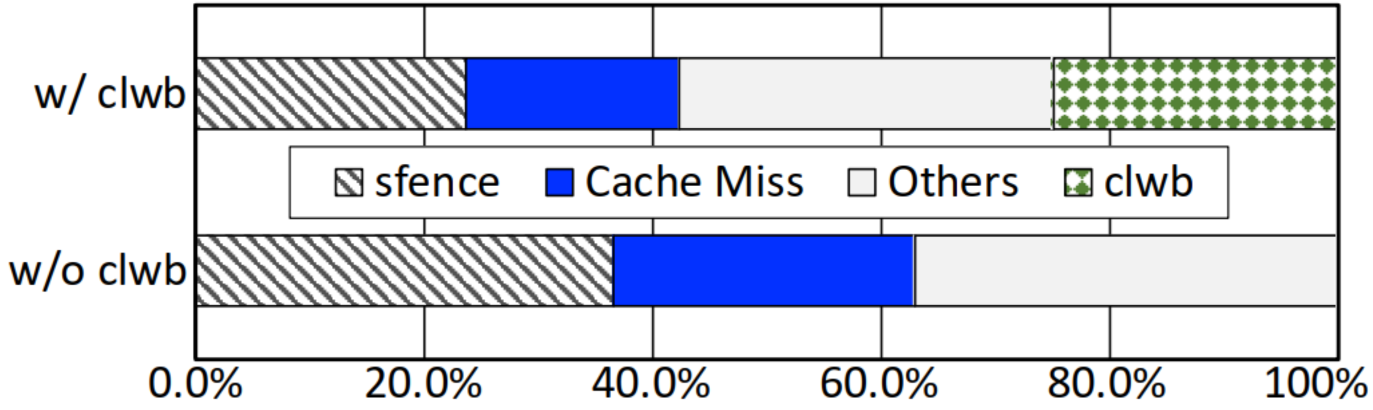


Figure 1: The breakdown of execution time w/ and w/o `clwbs`

Motivation

- eADR frees programmers from explicit cache line flushes
- The no need of flushing cache lines results in a change in the proportions of time cost

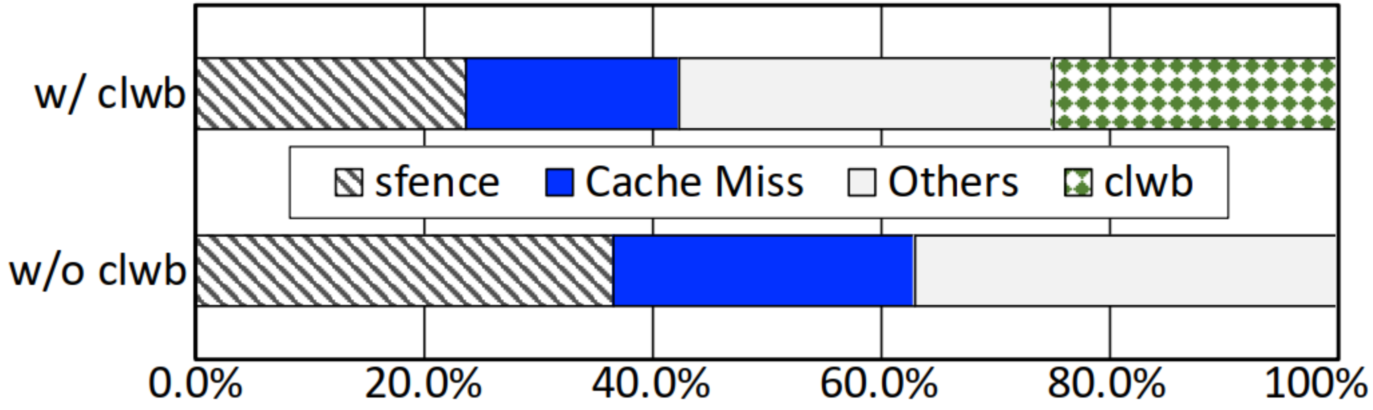


Figure 1: The breakdown of execution time w/ and w/o `clwbs`

- **Consequently, we place our emphasis on minimizing cache misses**

Outline

- Introduction & Background
- Motivation
- **Design of Conan**
  - **VIPT cache’s architectural implication**
  - Conflict-aware node allocation
  - Implementation and discussion
- Evaluation
- Conclusion

Design of CONAN

- Conan : **C**onflict-**A**ware-**N**ode-**A**llocation
- Aim to minimize the conflict cache misses
- Leverage architectural implications from the modern VIPT cache
- Conan is at the application level without any change to OS

VIPT cache’s architectural implication

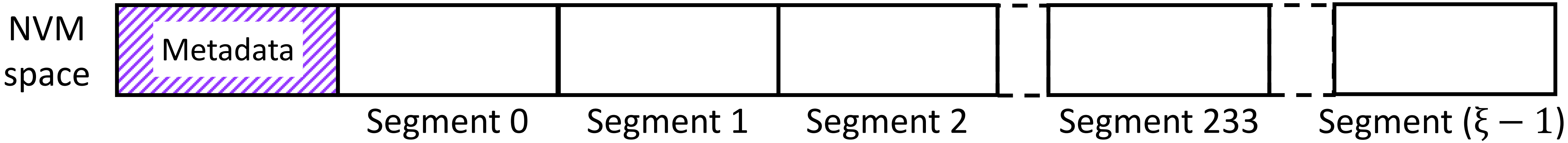
- Researchers generally create a big file for data structures on NVM
- However, past researchers have not particularly considered cache conflicts
  - libpmemobj of Intel PMDK
- We utilize modern VIPT cache
  - We memory-map a big file with a base virtual address **aligned at the cache line boundary**
  - **Virtual address determines** memory location which cache set maps to

Outline

- Introduction & Background
- Motivation
- **Design of Conan**
  - VIPT cache’s architectural implication
  - **Conflict-aware node allocation**
  - Implementation and discussion
- Evaluation
- Conclusion

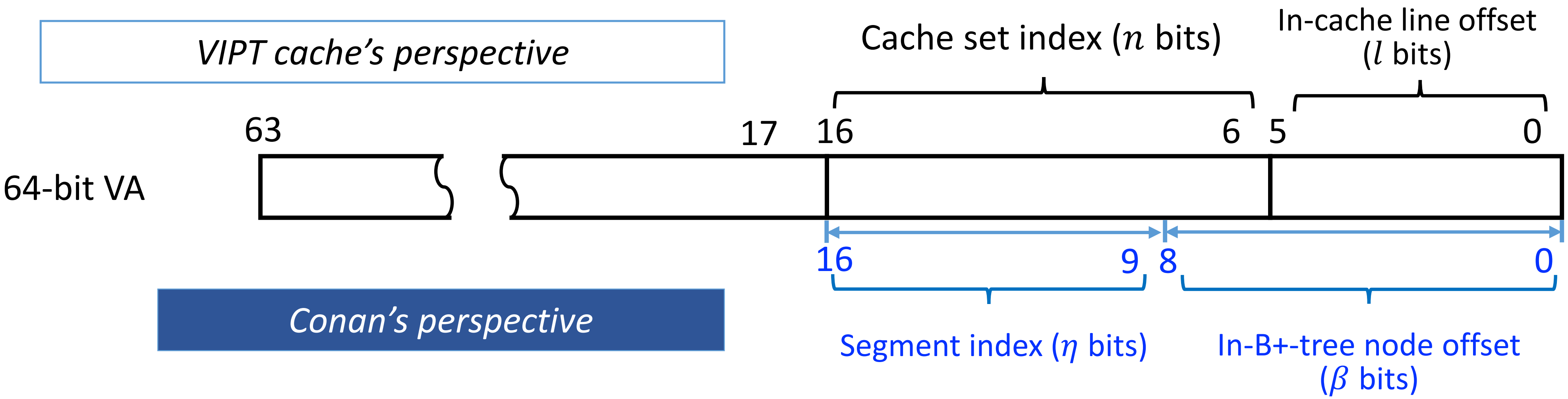
Implementation: Overview

Mmap a big file of NVM and separate it into several logical segments



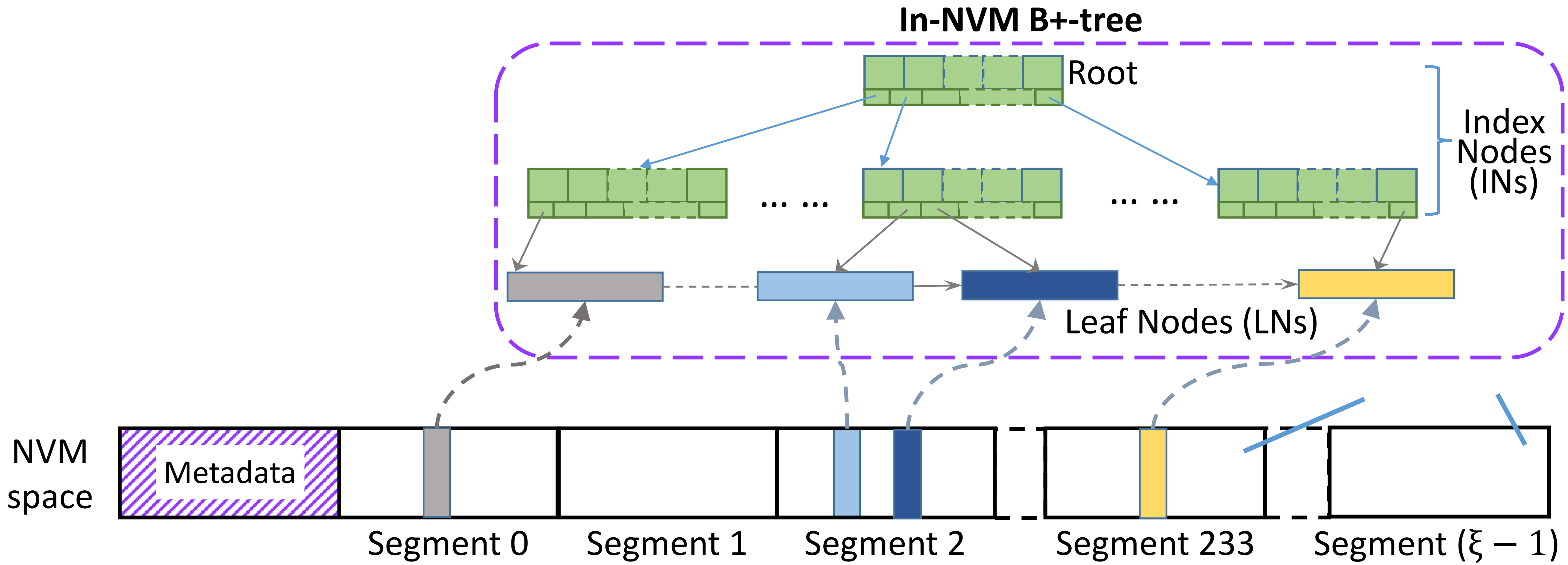
Conflict-Aware Node Allocation

- Assume that the number of cache sets in the LLC is 2^{n}(n>0),
- Assume that the size of a cache line is 2^{l} bytes (l>0)
  - n = 11 and l = 6 for below example


VIPT cache's perspective
Cache set index (n bits)
In-cache line offset (l bits)
63
17
16
6
5
0
64-bit VA
16
9
8
0
Conan's perspective
Segment index (η bits)
In-B+-tree node offset (β bits)

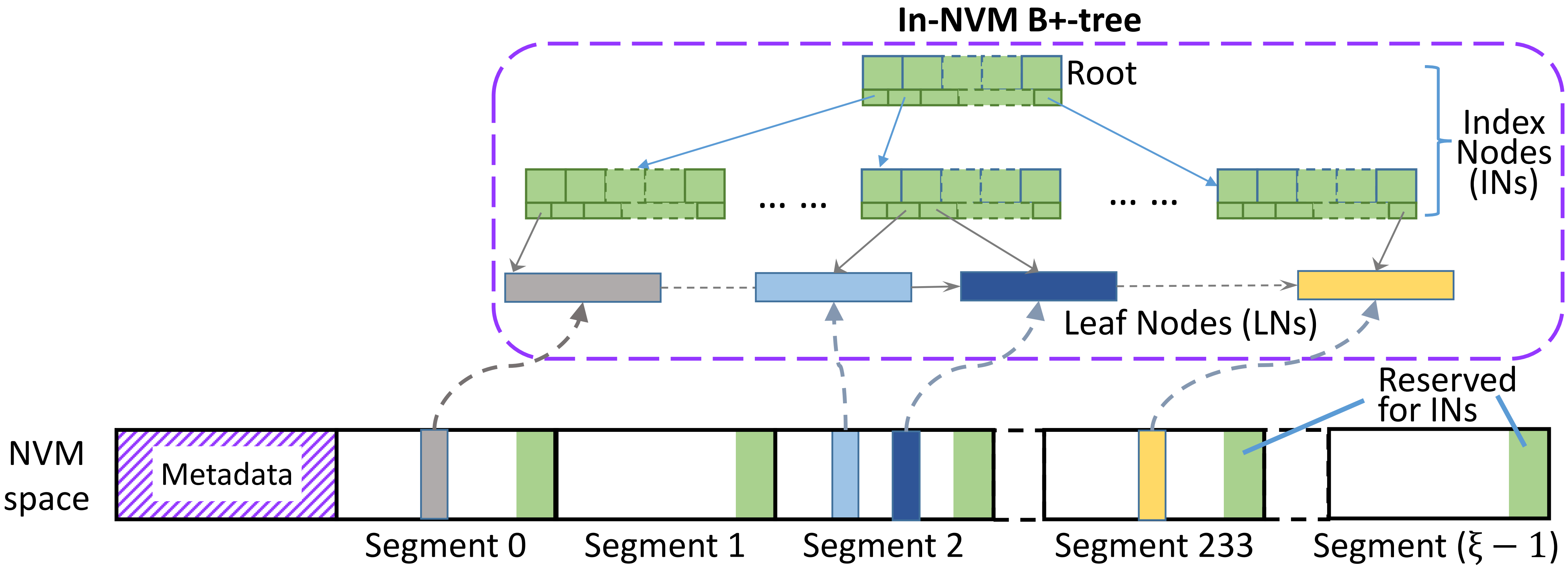
Implementation: Overview

Allocate each new leaf node to the corresponding Segment



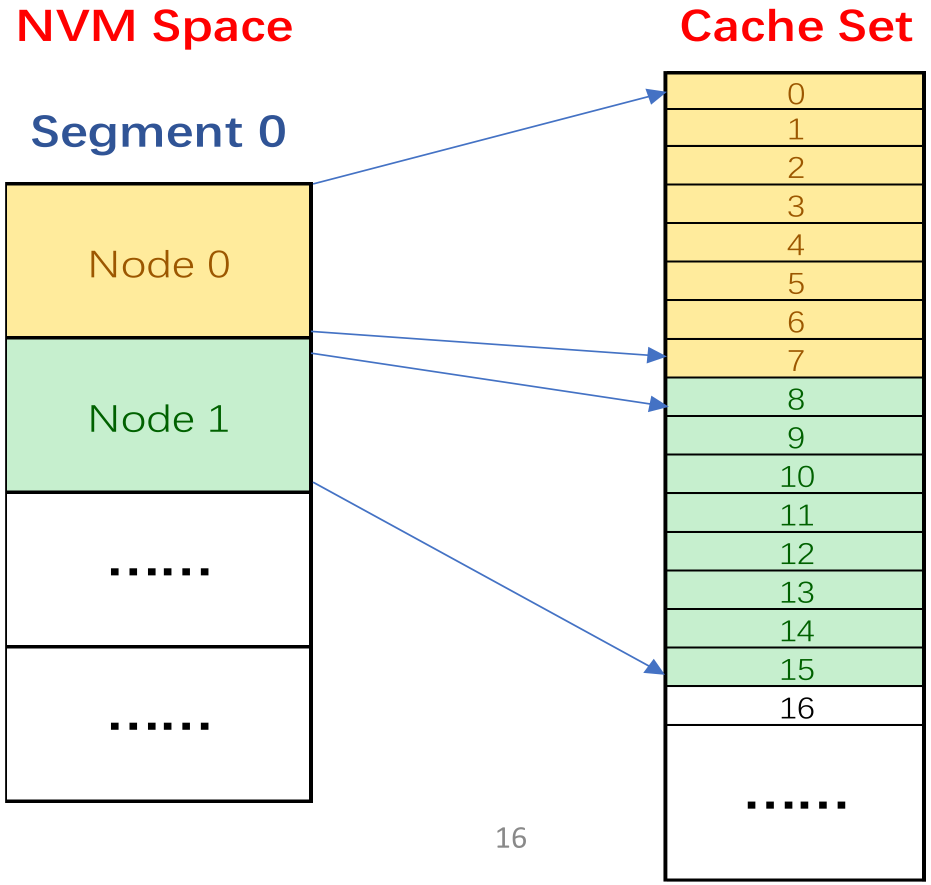
Implementation: Overview

Reserve space in each segment for Index Nodes to accelerate searching and insertions



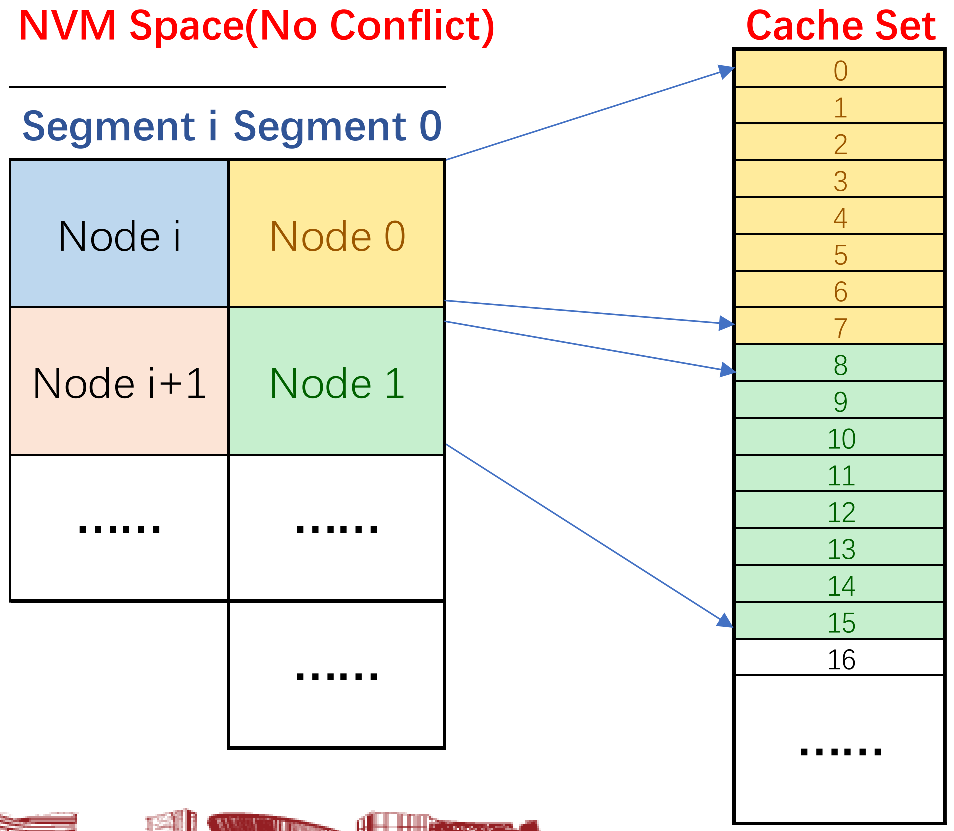
Conflict-Aware Node Allocation

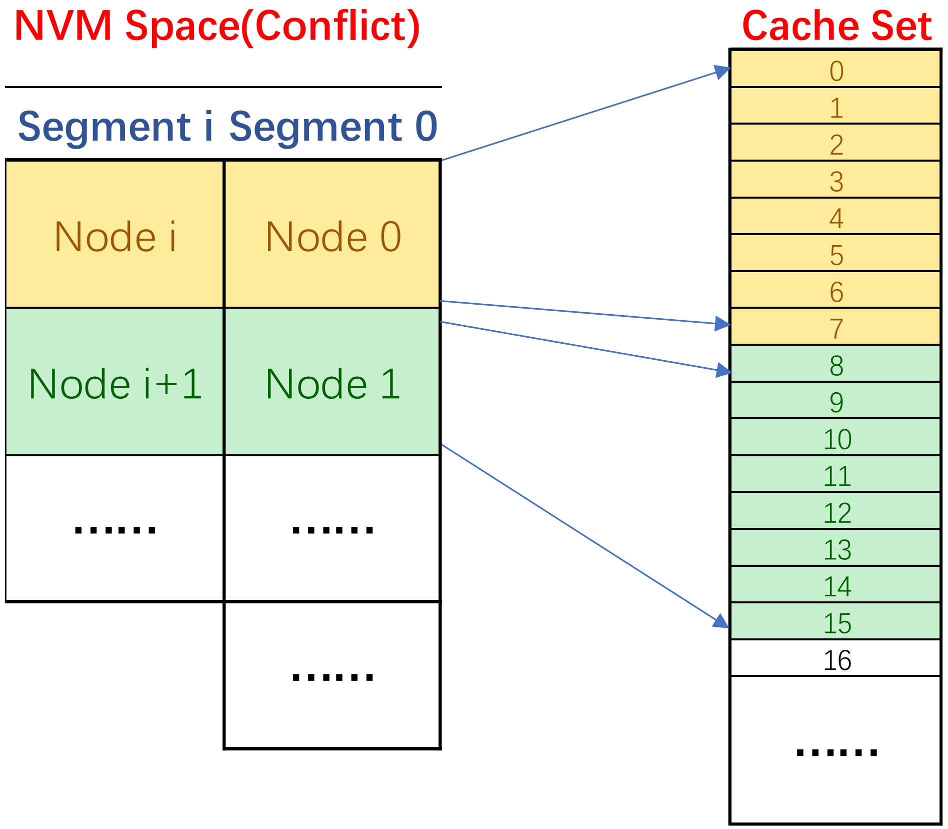
- Theorem 1 (Intra-Segment Non-Conflict)
  - One single B+-tree node or any two B+-tree nodes allocated from the same segment
    - =>incur **no conflict** into any cache set



Conflict-Aware Node Allocation

- Theorem 2 (Inter-Segment Conflict)
  - Two B+-tree nodes allocated from different segments
    - =>may conflict in the same cache set





Conflict-Aware Node Allocation

- Cache line is 2^{l} bytes (l>0)
- B^{+} tree node size is 2^{β} bytes (β>0)

- Theorem 3
  - The expected probability of conflict for Inter-Segment is small.
  - It could be calculated from below formula

$$2^{(\beta-s)}$$

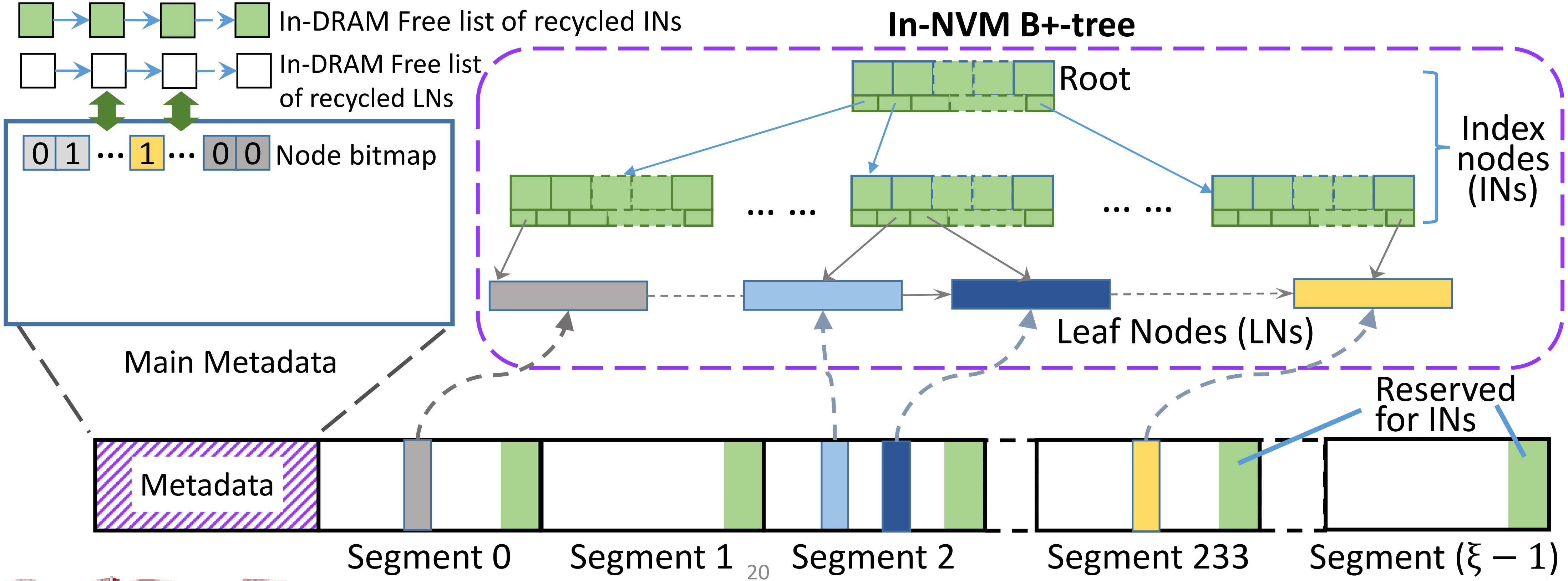
  - When B+ tree node size is 512B and a segment is 128KB
  - => probability is < 0.4%

Outline

- Introduction & Background
- Motivation
- **Design of Conan**
  - VIPT cache's architectural implication
  - Conflict-aware node allocation
  - **Implementation and discussion**
- Evaluation
- Conclusion

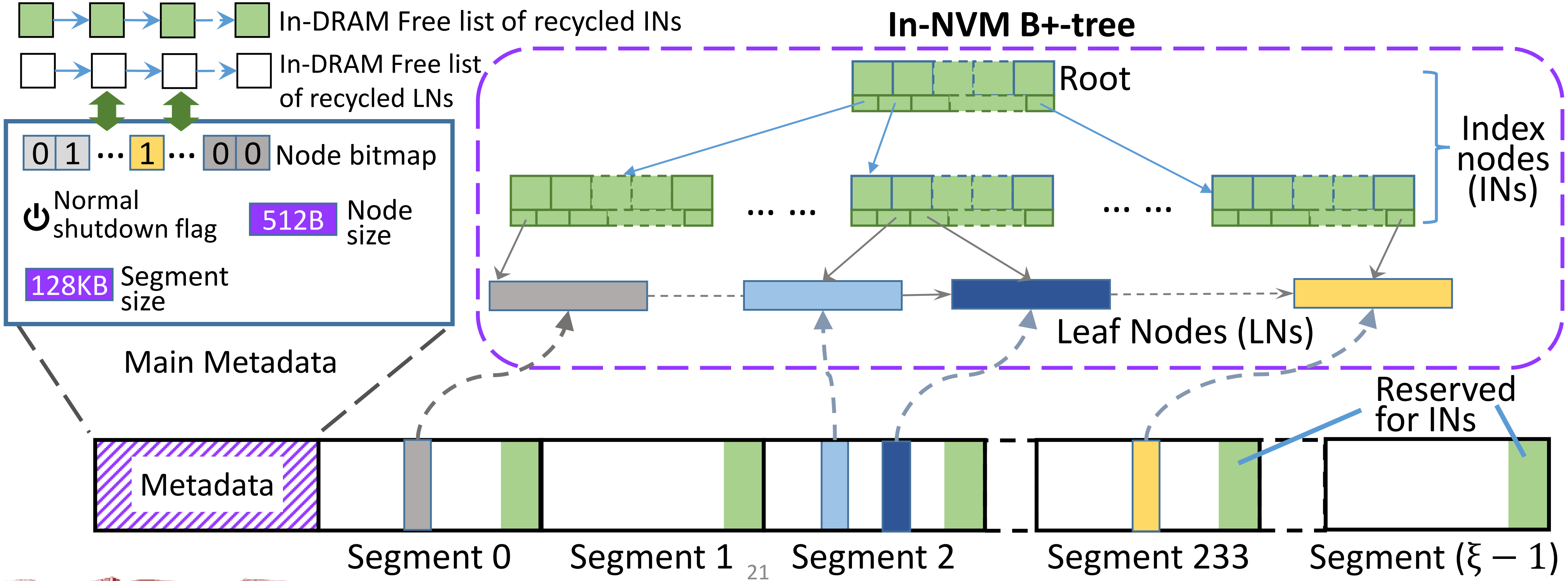
Implementation and Discussion

- **Allocation and deallocation occasions**
  - Use in-DRAM skiplists to accelerate allocations and deallocations
  - A bitmap in NVM to make skiplists durable



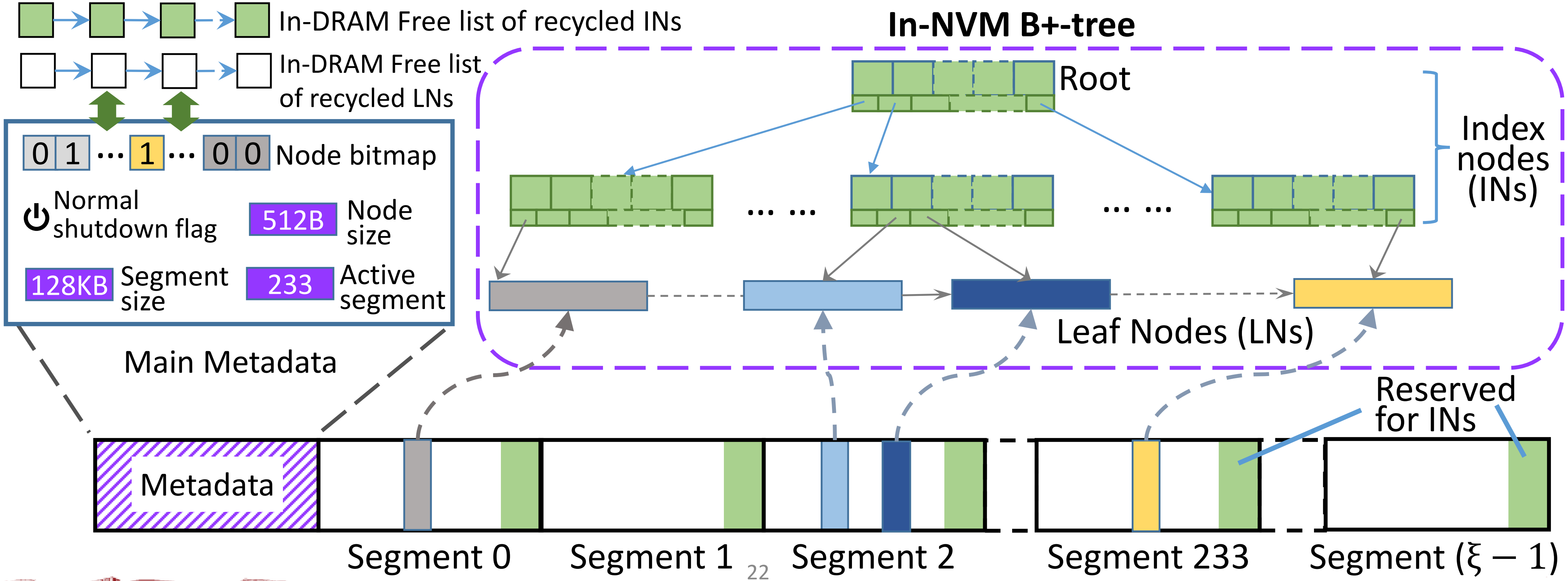
Implementation and Discussion

- **Crash consistency of Conan**
  - Utilize the bitmap to track
  - Modern 64-bit CPUs allow an atomic write of 64 bits
  - => Rule out inconsistency issue for node allocation/deallocation



Implementation and Discussion

- **Concurrency of Conan**
  - A fine-grained lock per segment in DRAM to support concurrency



Outline

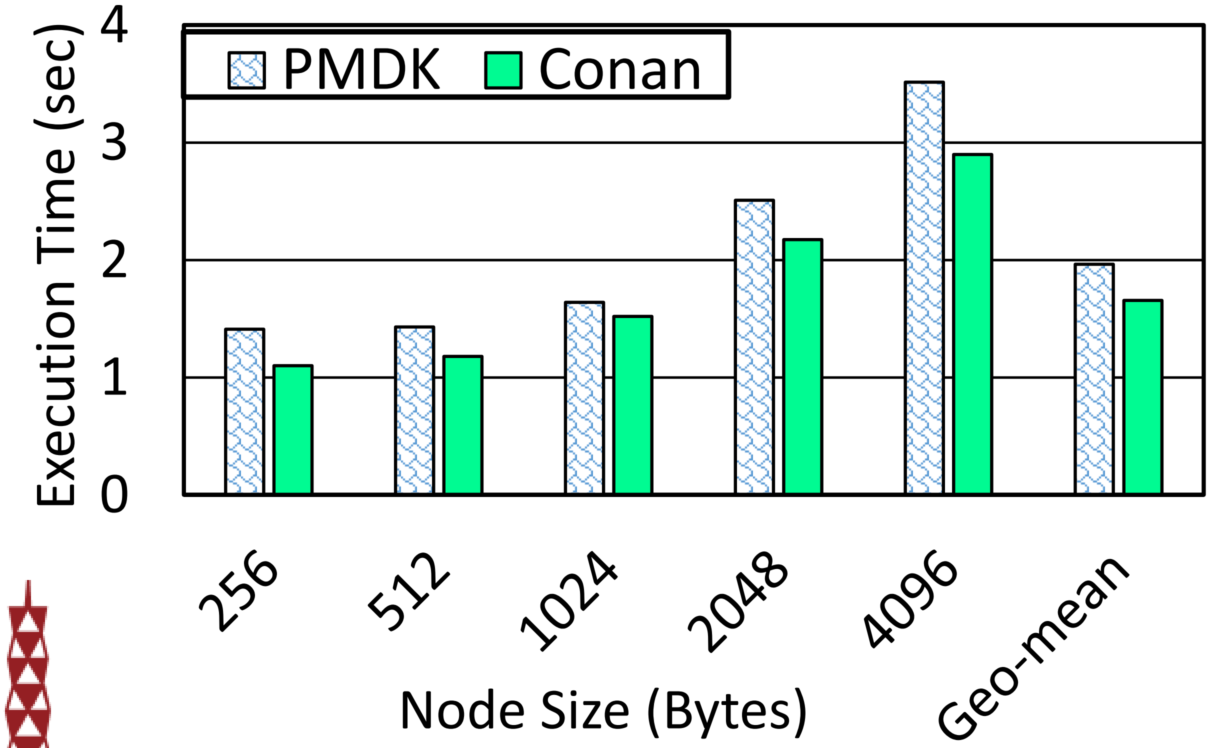
- Introduction & Background
- Motivation
- Design of CONAN
  - VIPT cache’s architectural implication
  - Conflict-aware node allocation
  - Implementation and discussion
- **Evaluation**
- Conclusion

Evaluation: Setup

- CPU:
  - 48-core Intel Xeon Gold 6342 with eADR feature
  - To eliminate the impact of NUMA, only use NVM space in node 0 and CPU in socket 0
- CACHE:
  - L1D, L2 and L3 caches are 2.3MB, 60MB and 72MB, respectively
  - 2048 cache sets in the shared last-level L3 cache with 12-way associativity
  - Size per cache line is 64B
- DRAM & NVM:
  - The size of DRAM: 256GB
  - Intel Optane persistent memory: 1024GB
- PLATFORM:
  - Ubuntu 21.04 with kernel version 5.13.2 ; GCC/G++ 10.3.0
  - Ext4-DAX

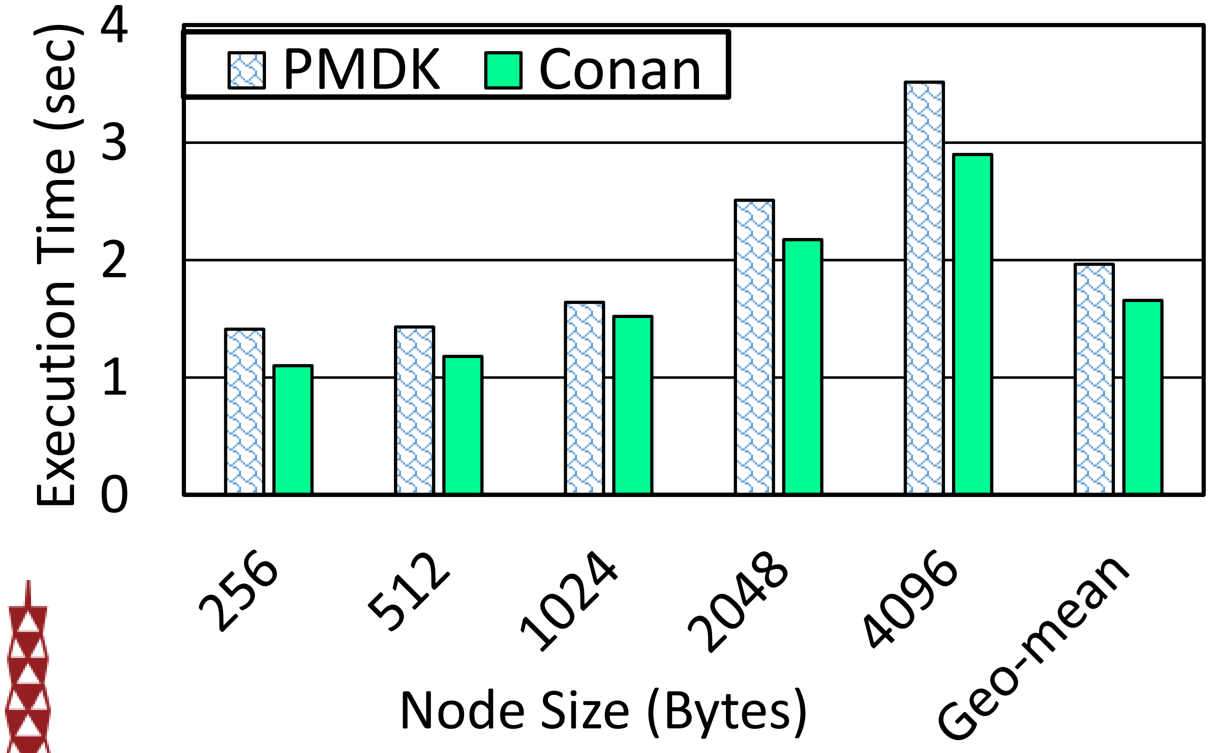
Evaluation: Micro-Benchmark

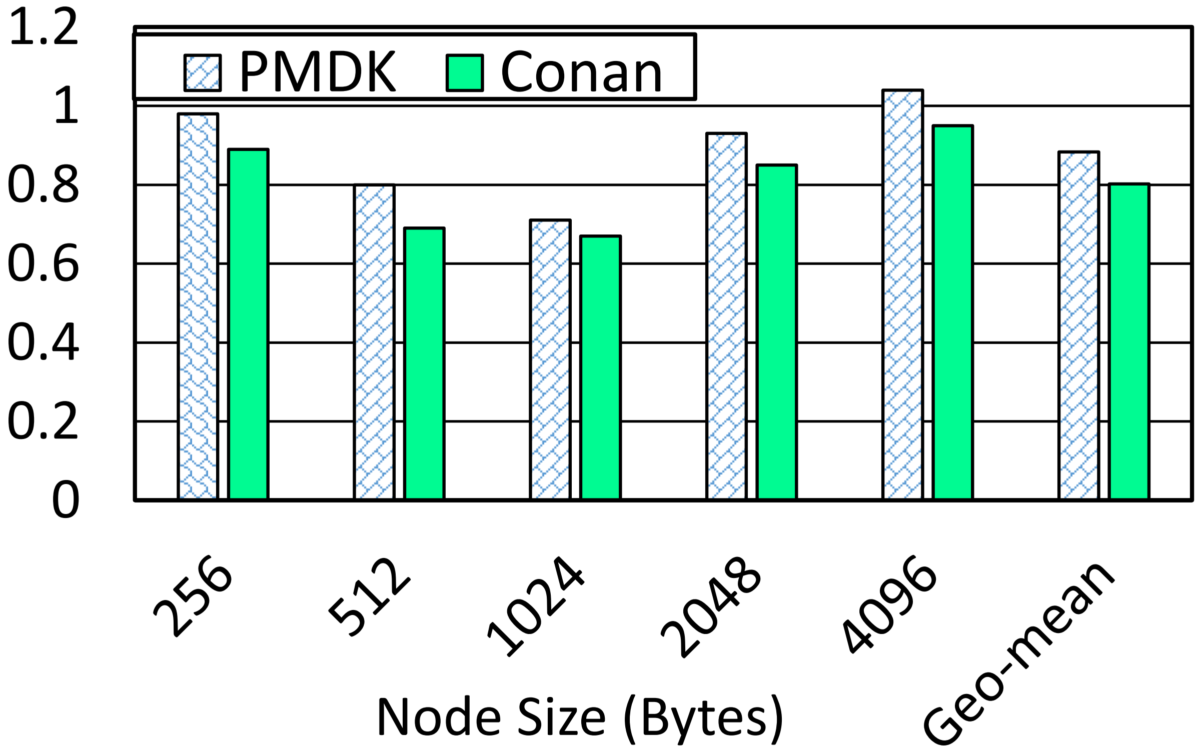
- **Insertion**: Saves the execution time by 15.7% on average and achieves highest boost by 22.0%



Evaluation: Micro-Benchmark

- **Insertion**: Saves the execution time by 15.7% on average and achieves highest boost by 22.0%
- **Search:** Conan spends 13.8% less time compared to PMDK



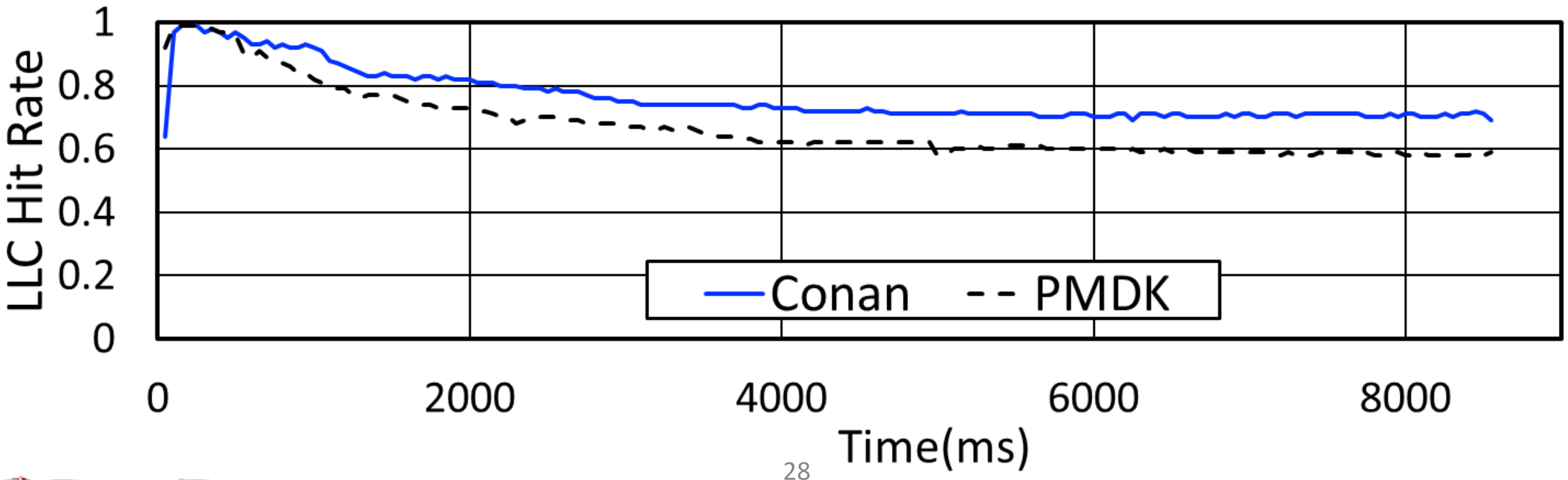


Evaluation: Reason

- Nodes allocated by PMDK might not be cache line-aligned
  - Recorded virtual addresses in-NVM roots for three times:
    - 0x7f25403c0558
    - 0x7fcdc03c0558
    - 0x7fe5803c0558
  - **Virtual addresses are misaligned at a cache line boundary**

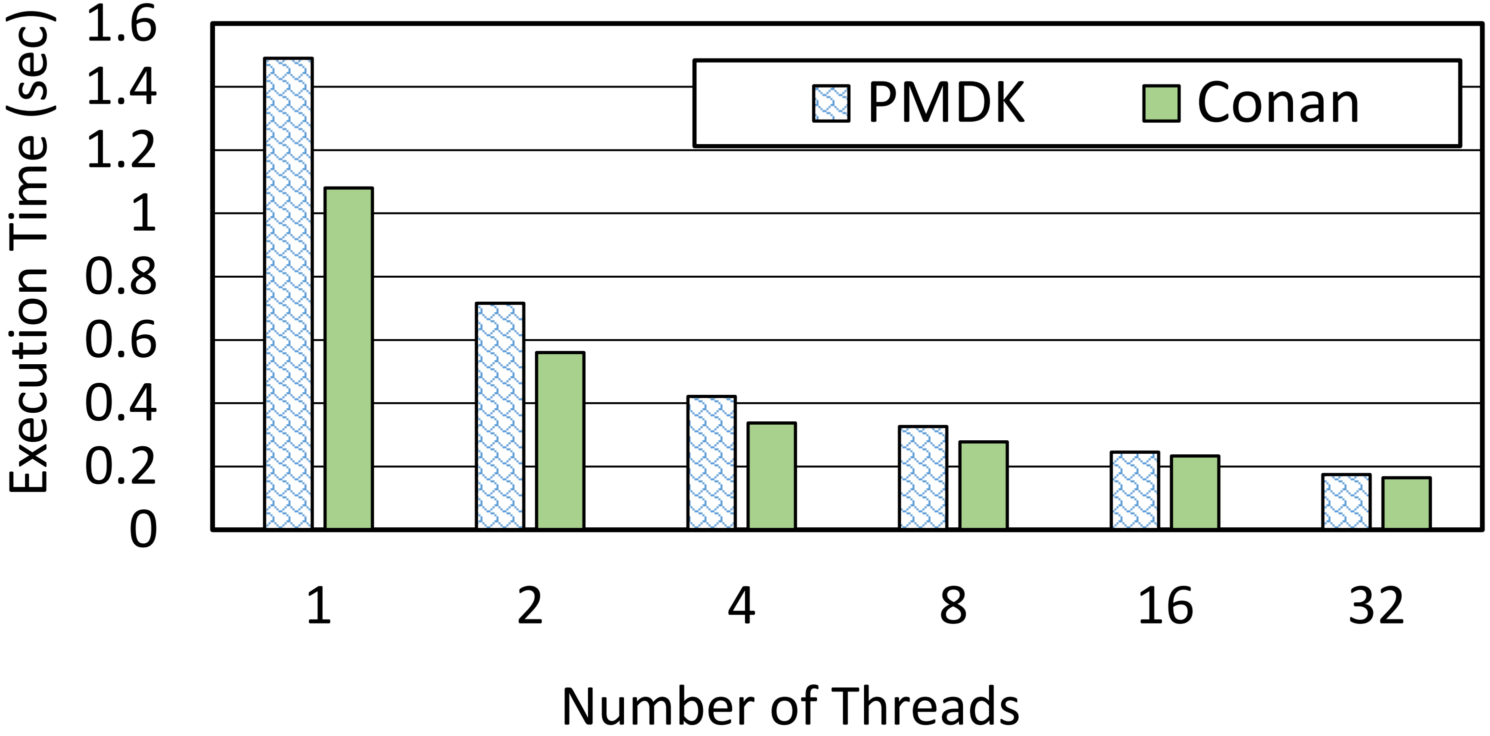
Evaluation: Reason

- Nodes allocated by PMDK might not be cache line-aligned
  - Recorded virtual addresses in-NVM roots for three times:
    - 0x7f25403c0558
    - 0x7fcdc03c0558
    - 0x7fe5803c0558
  - **Virtual addresses are misaligned at a cache line boundary**
- Higher LLC hit rate of Conan



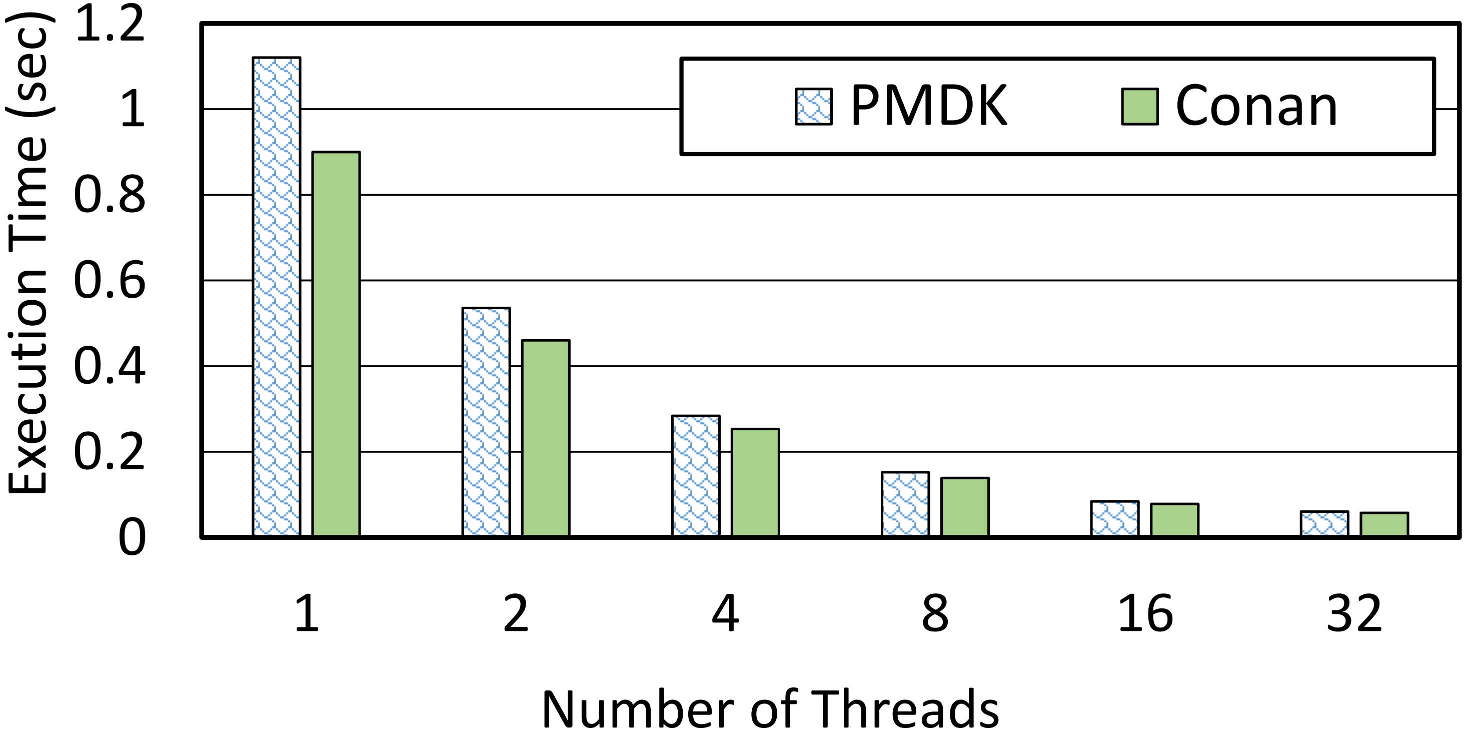
Evaluation: Multi-thread

- **Insertion:** ten million KV pairs with the node size of 1024B



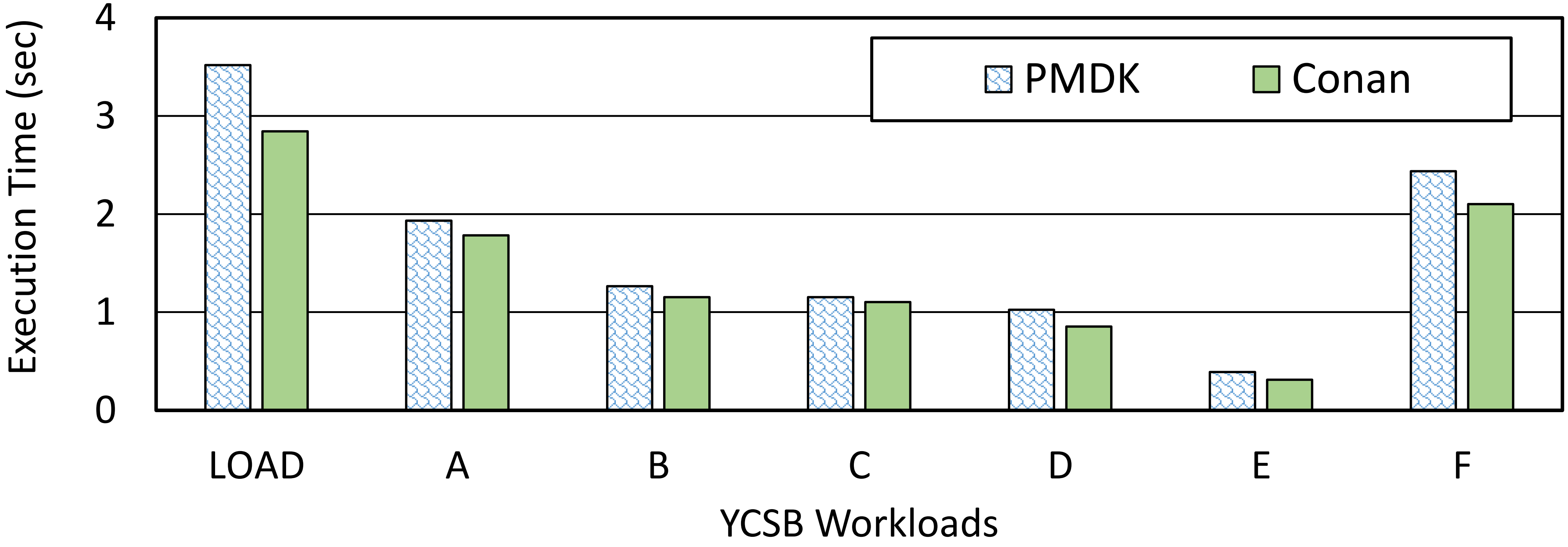
Evaluation: Multi-thread

- **Search:** ten million KV pairs with the node size of 1024B



Evaluation: YCSB

- **YCSB:** Improvement up to 19.8% with YCSB workloads



Outline

- Introduction & Background
- Motivation
- Design of CONAN
  - VIPT cache’s architectural implication
  - Conflict-aware node allocation
  - Implementation and discussion
- Evaluation
- **Conclusion**

Conclusion

- Conan takes into account the minimization of conflict cache
- Conan mainly explores the mapping between VIPT cache and NVM space exposed with virtual addresses
- Conan saves the execution time in writing and reading by up to 22.0% and 13.8%, respectively
- Conan is simplistic and realistically effectual

Thanks :-)

Presenter: Yanpeng Hu

Email: huyp@Shanghaitech.edu.cn

https://toast-lab.tech