PACTree: A High Performance Persistent Range Index Using *PAC Guidelines*

Wook-Hee Kim, R. Madhava Krishnan, Xinwei Fu,
Sanidhya Kashyap, and Changwoo Min

Talk outline

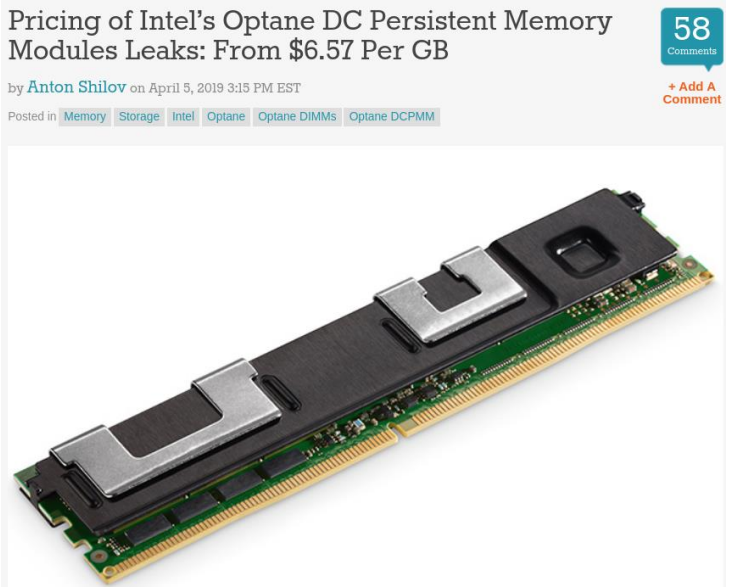
- Background
- Packed Asynchronous Concurrency (PAC) Guidelines
- PACTree : A High Performance Persistent Range Index Using PAC Guidelines
- Evaluation
- Conclusion

Talk outline

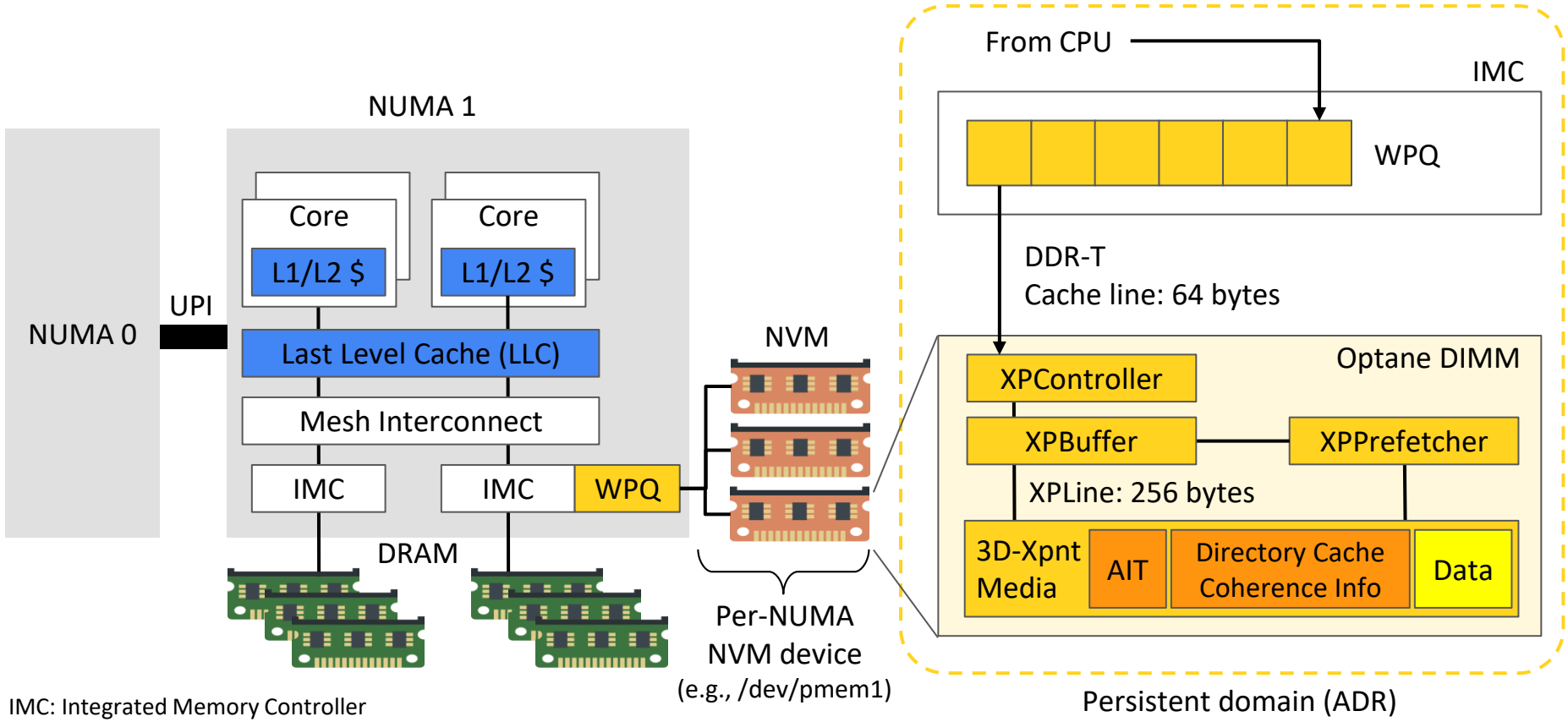
- **Background**
- Packed Asynchronous Concurrency (PAC) Guidelines
- PACTree : A High Performance Persistent Range Index Using PAC Guidelines
- Evaluation
- Conclusion

Non-Volatile Memory (NVM) is now available!

- **Intel Optane DCPMM is the first commercially available NVM in the market.**
  - Byte-addressable like DRAM and durable storage like SSD
  - Huge capacity
    - 3TB per socket (e.g., 4-socket server = 12TB NVM + 6TB DRAM)
  - Direct access to NVM using load/store bypassing the storage stack



NVM Hardware: Intel DC Persistent Memory



IMC: Integrated Memory Controller
WPQ: Write Pending Queue
AIT: Address Indirection Table
UPI: Ultra Path Interconnect

NVM is not a slow DRAM

- Previous studies focus on hardware aspects of NVM
  - Yang et al.[FAST'20], Wang et al.[Micro'20], Gugnani et al.[VLDB'21]

- The limited bandwidth and slow latency of NVM is the fundamental limiting factor in designing storage systems.

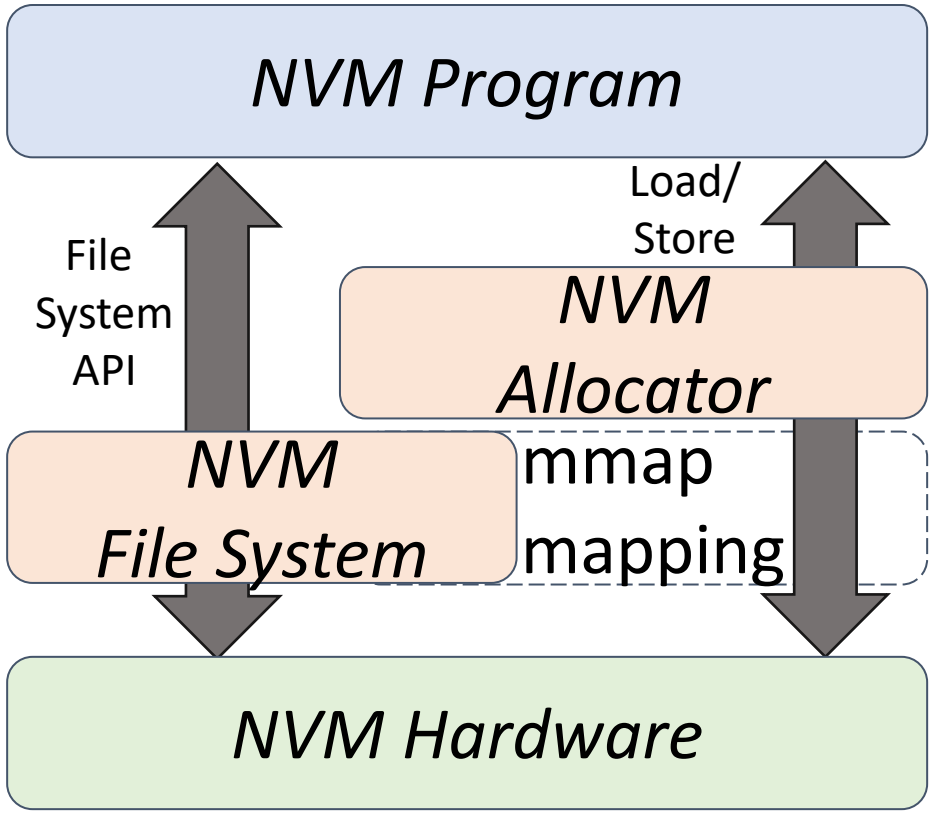
Let's investigate performance properties of NVM (Optane) and explore their implications for core storage system design -- index.

[1] Yang et al., An Empirical Guide to the Behavior and Use of Scalable Persistent Memory, FAST'20

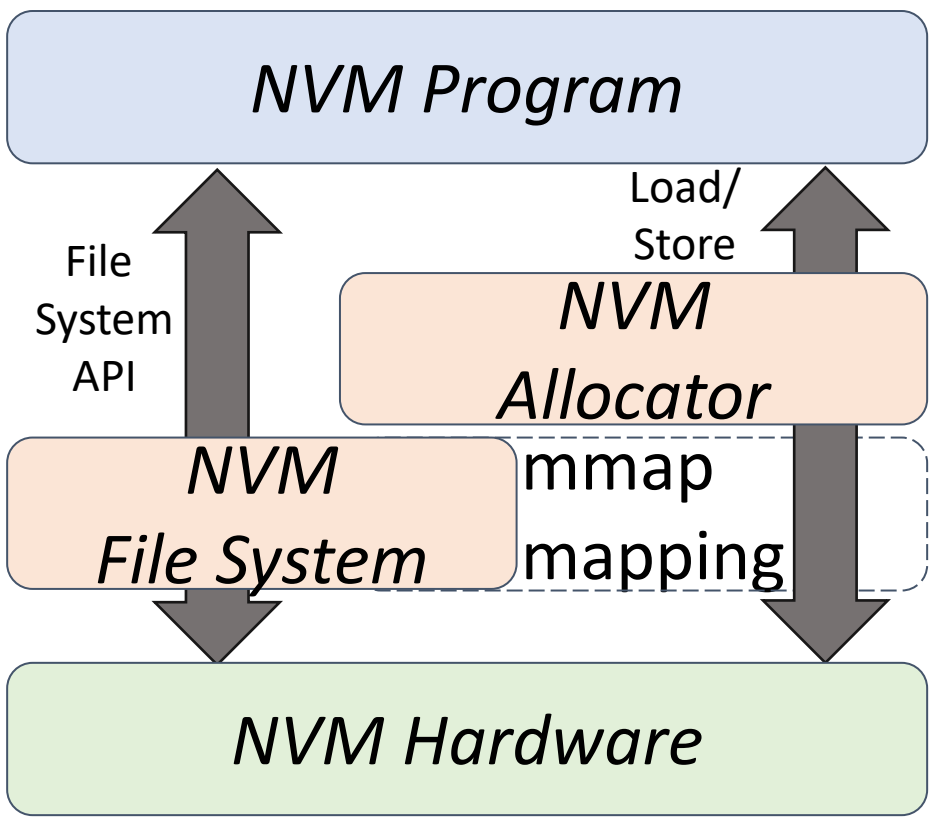
Talk outline

- Background
- **Packed Asynchronous Concurrency (PAC) Guidelines**
- PACTree : A High Performance Persistent Range Index Using PAC Guidelines
- Evaluation
- Conclusion

NVM Software stack



Packed Asynchronous Concurrency (PAC) guidelines



Guidelines on Concurrency Control

Guidelines on Index Structures

Guidelines on NVM System Software

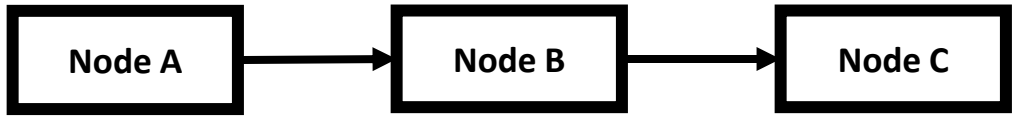
Findings on NVM Hardware

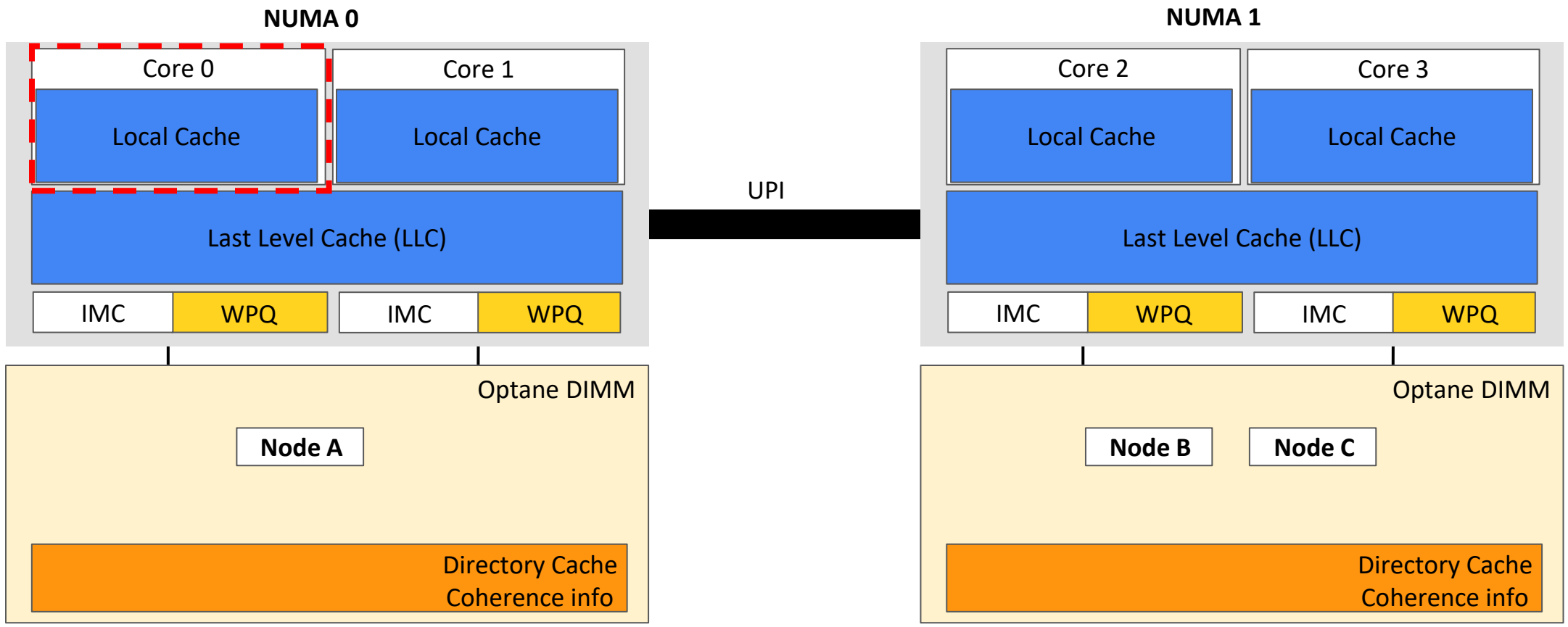
Packed Asynchronous Concurrency (PAC) guidelines

- Finding on NVM hardware
  - FH5. Cache coherence protocol impedes NUMA scalability.
- Guidelines on NVM system software
  - GS1. Persistent memory allocation is very expensive.
- Guidelines on persistent index algorithm
  - GA1. Lookup operation should consume minimal NVM bandwidth.
- Guidelines on concurrency control of a persistent index
  - GC2. Minimize the blocking time of structural modification operations.

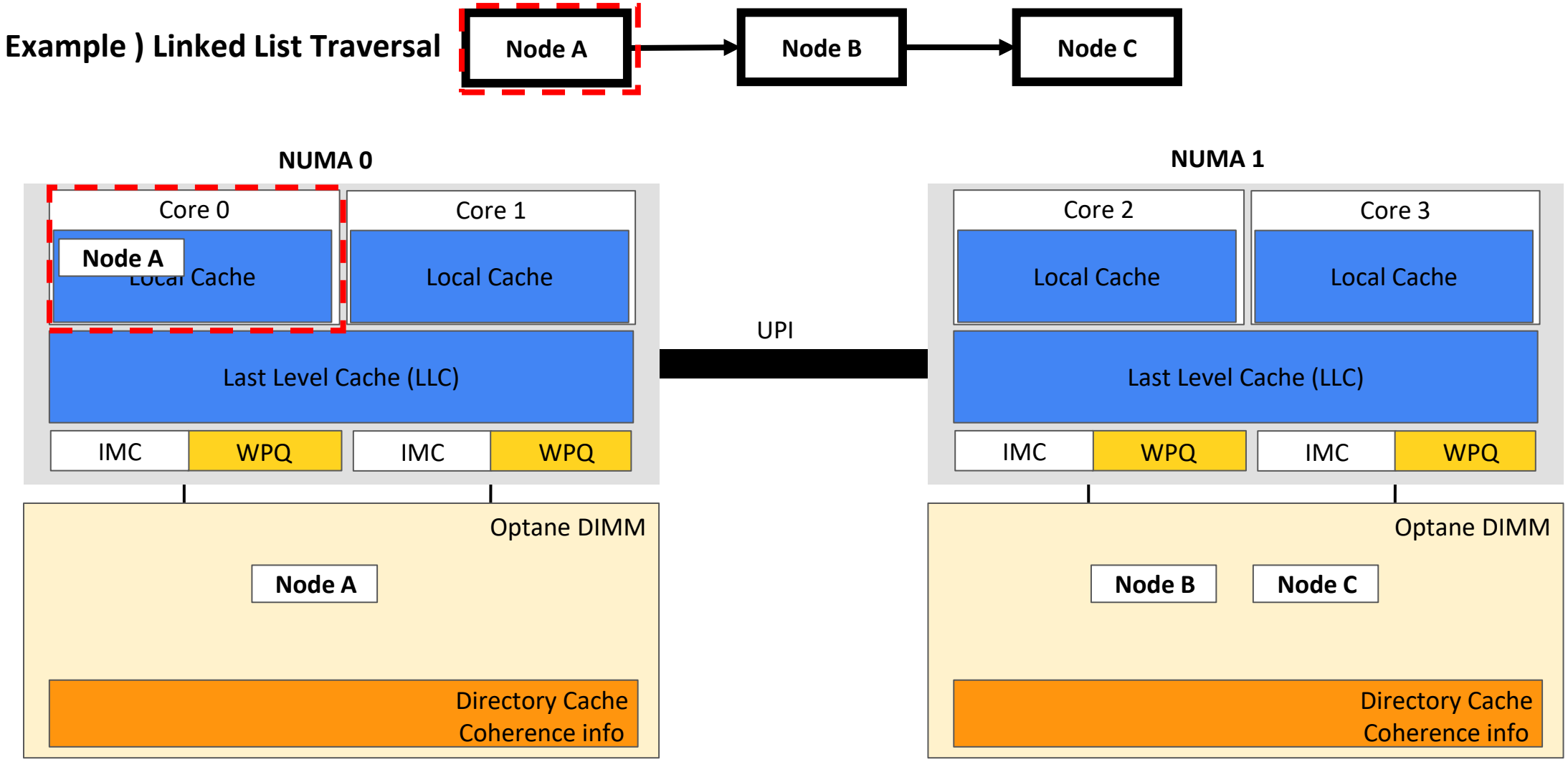
Cache coherence protocol impedes NUMA scalability

Example) Linked List Traversal



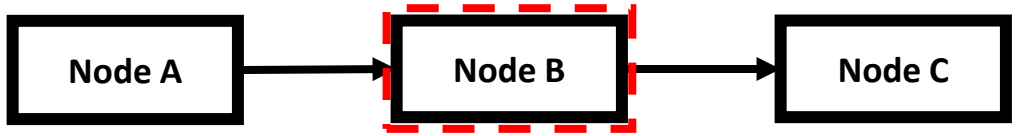


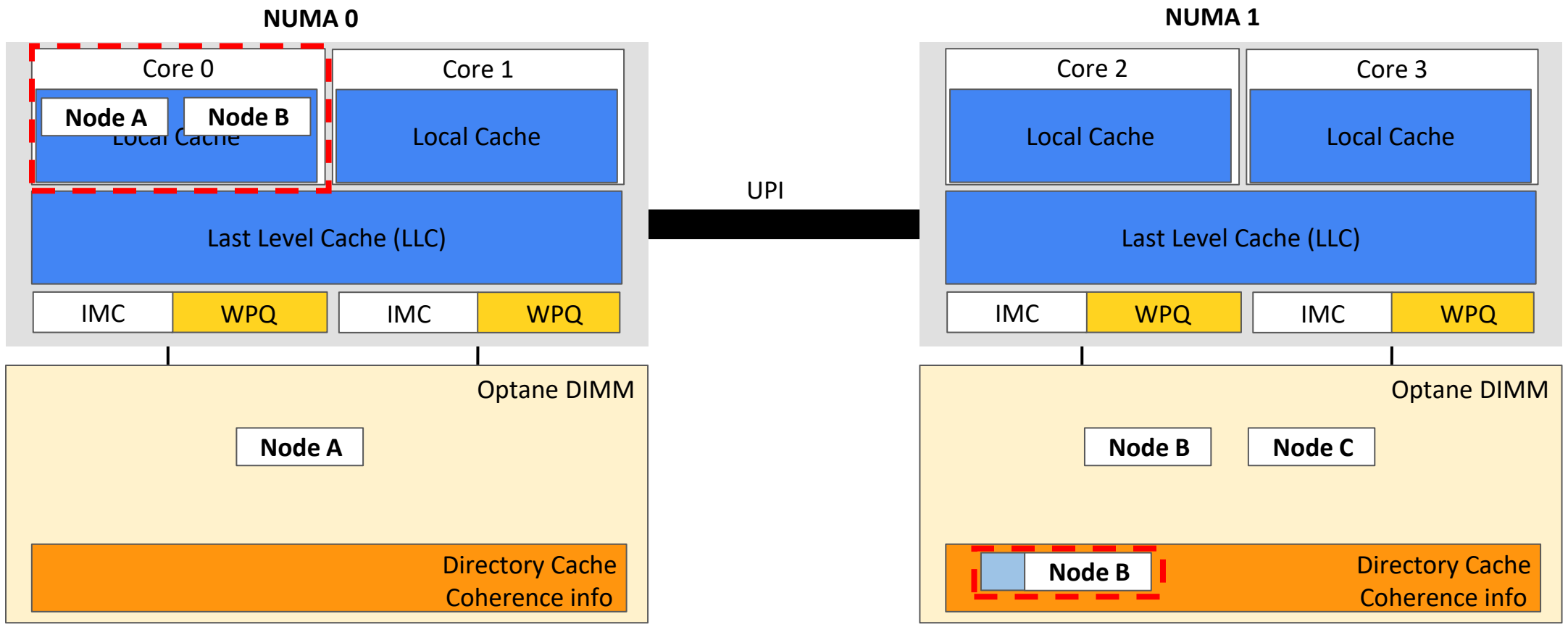
Cache coherence protocol impedes NUMA scalability



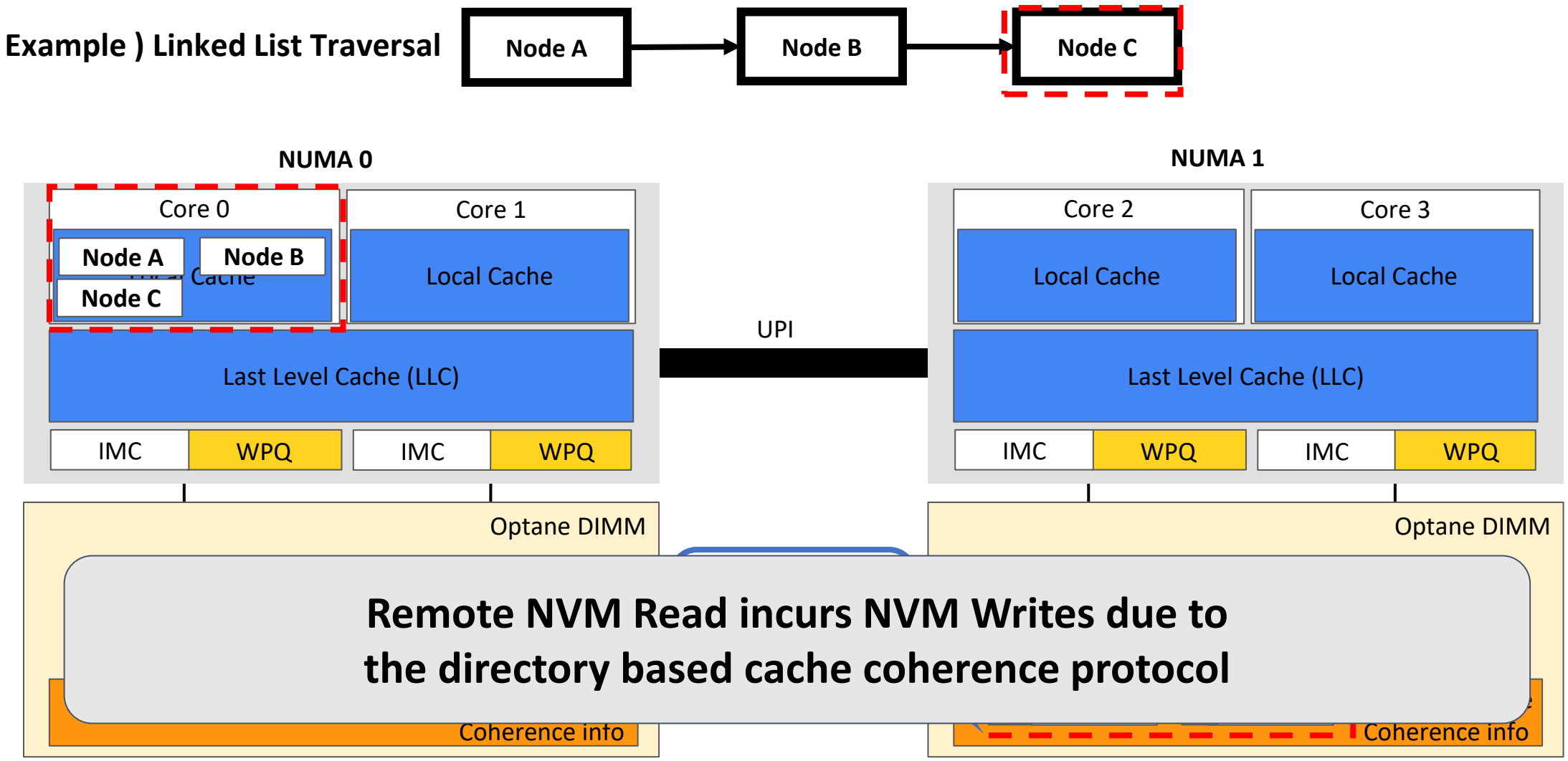
Cache coherence protocol impedes NUMA scalability

Example) Linked List Traversal



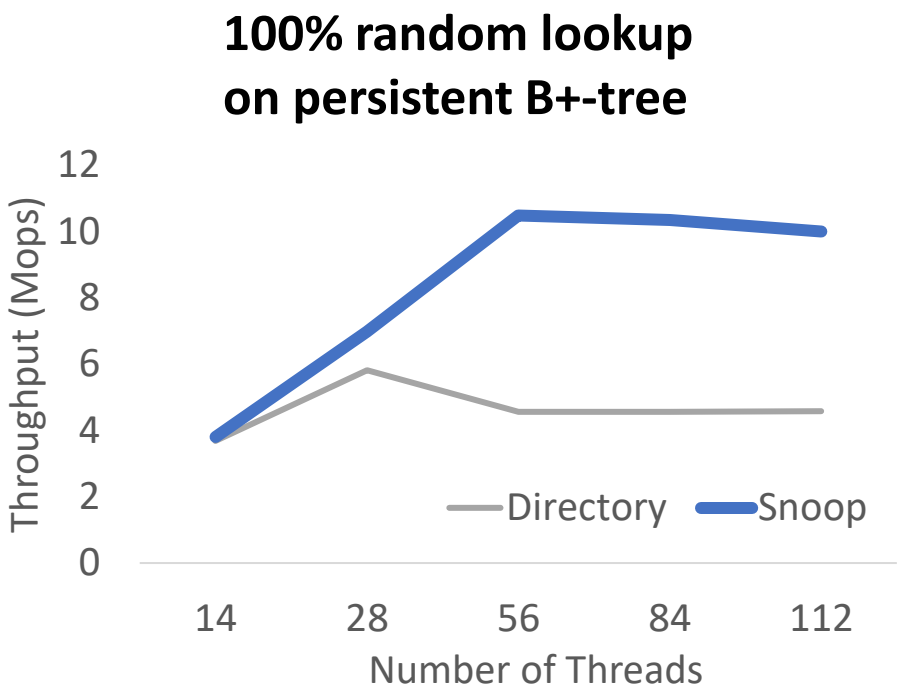


Cache coherence protocol impedes NUMA scalability



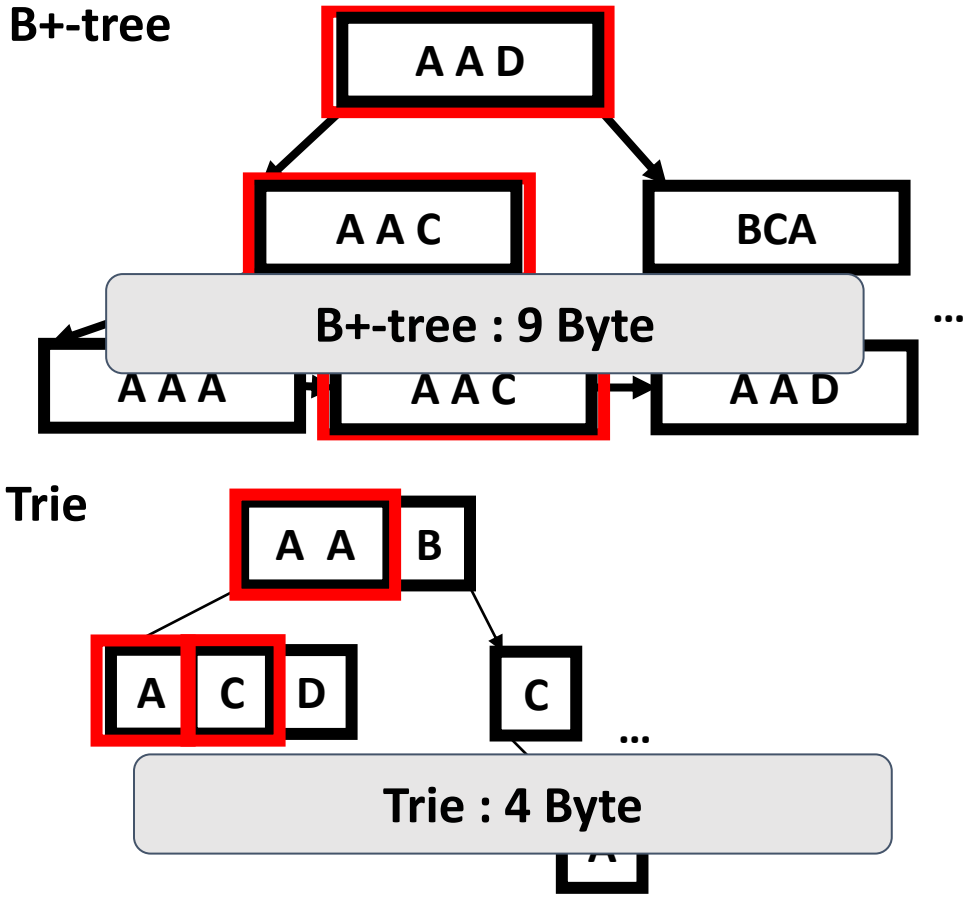
Cache coherence protocol impedes NUMA scalability

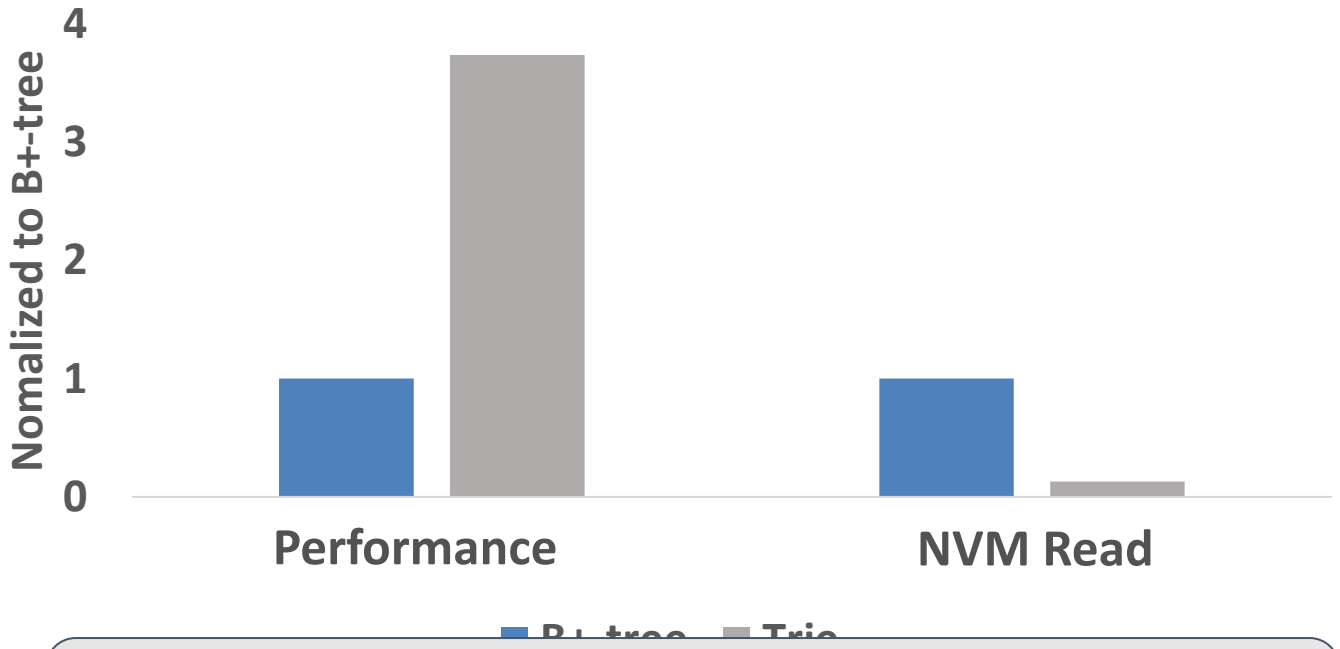
- Such coherence write traffic could be significant!
  - Example 1) 100% 64-byte random read of 870MB
  - NVM read: 870MB, NVM write: 481MB (55%!)
- A tentative solution is to change the cache coherence protocol to snoop coherence at BIOS.
  - Ultimately, the directory information should not be stored in NVM.
- Snoop coherence shows much higher performance.
  - Example 2) 100% random lookup on a persistent B+tree



Lookup operation should consume minimal NVM bandwidth

- Let's compare the NVM bandwidth consumption in two representative index design, B+-tree and Trie, using an example of a lookup of a key “AAC”





Access in a packed fashion to save the limited NVM bandwidth

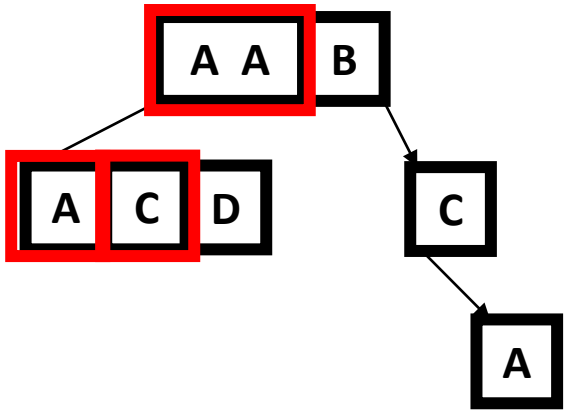
Talk outline

- Background
- Packed Asynchronous Concurrency (PAC) Guidelines
- **PACTree : A High Performance Persistent Range Index Using PAC Guidelines**
- Evaluation
- Conclusion

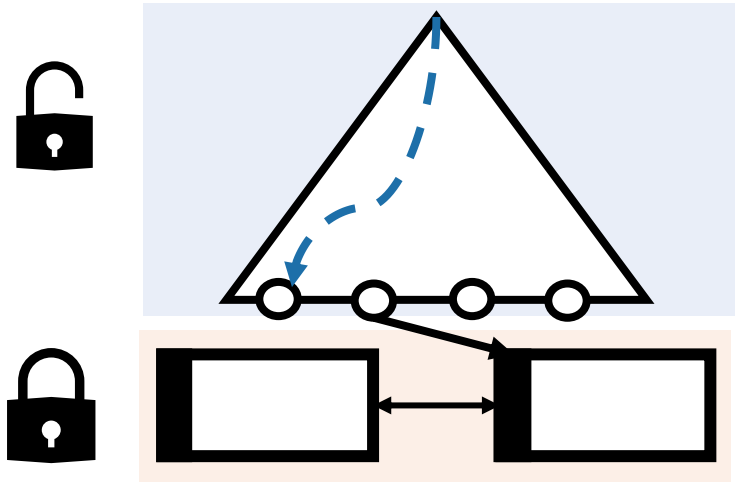
Key take away from *PAC Guidelines*

- A high-performance persistent index should provide

Packed Access to NVM



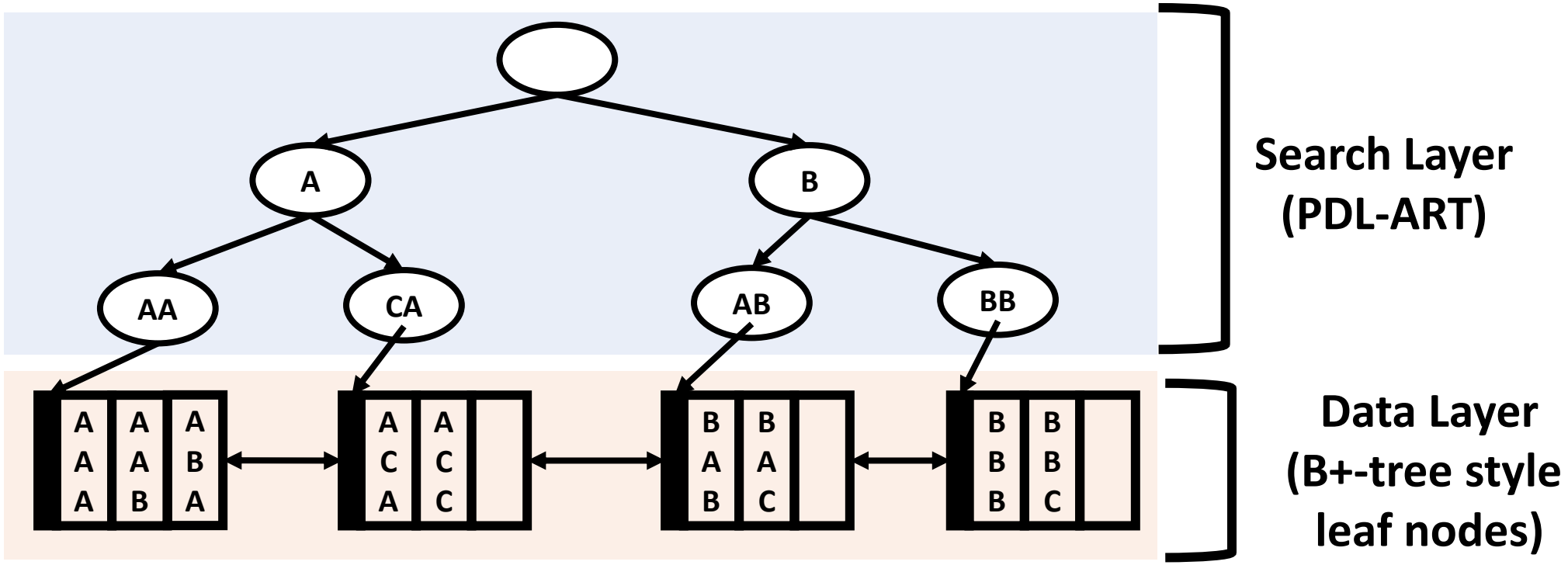
Asynchronous Concurrency



Saving bandwidth

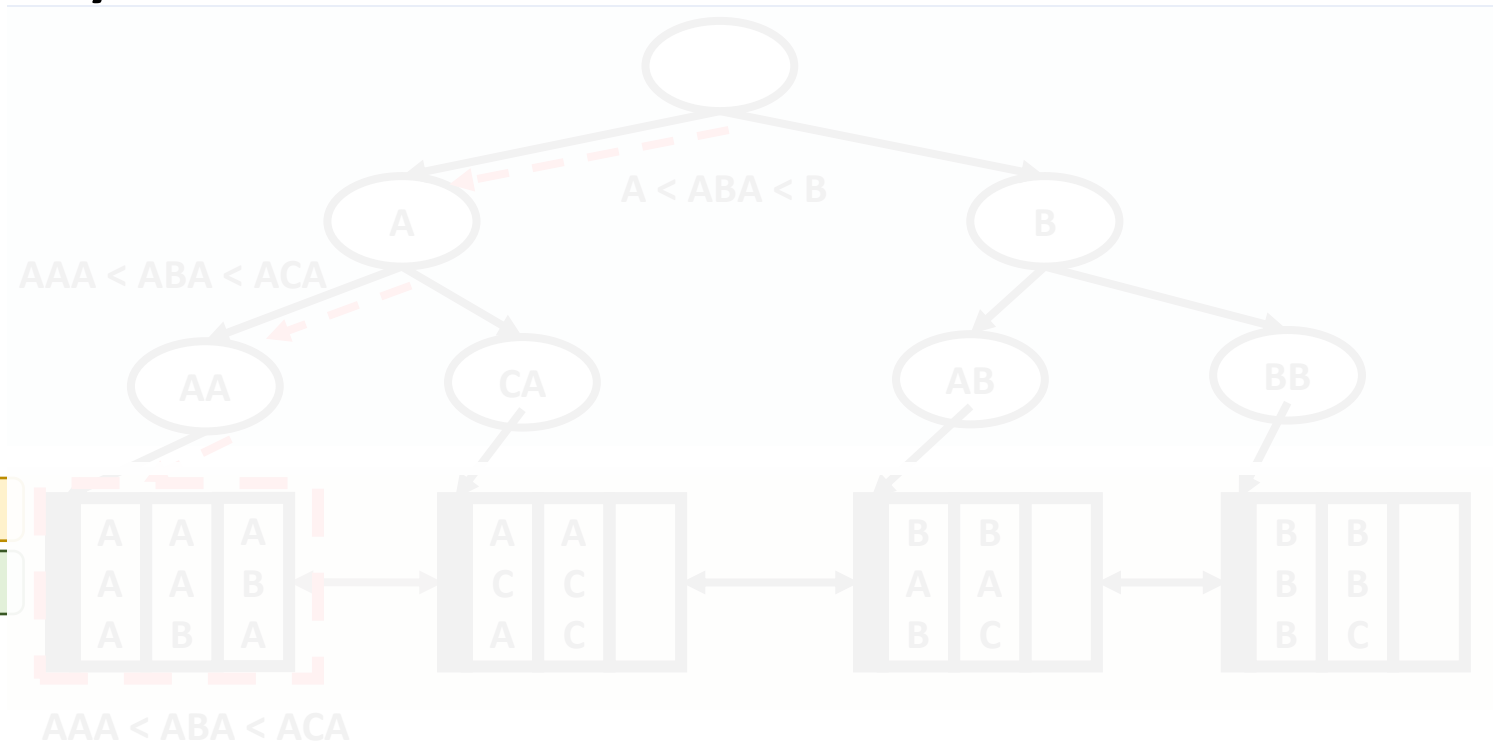
Decoupling slow NVM latency

PACTree: a persistent index based on the PAC guidelines



Lookup operation

Find the key ‘ABA’

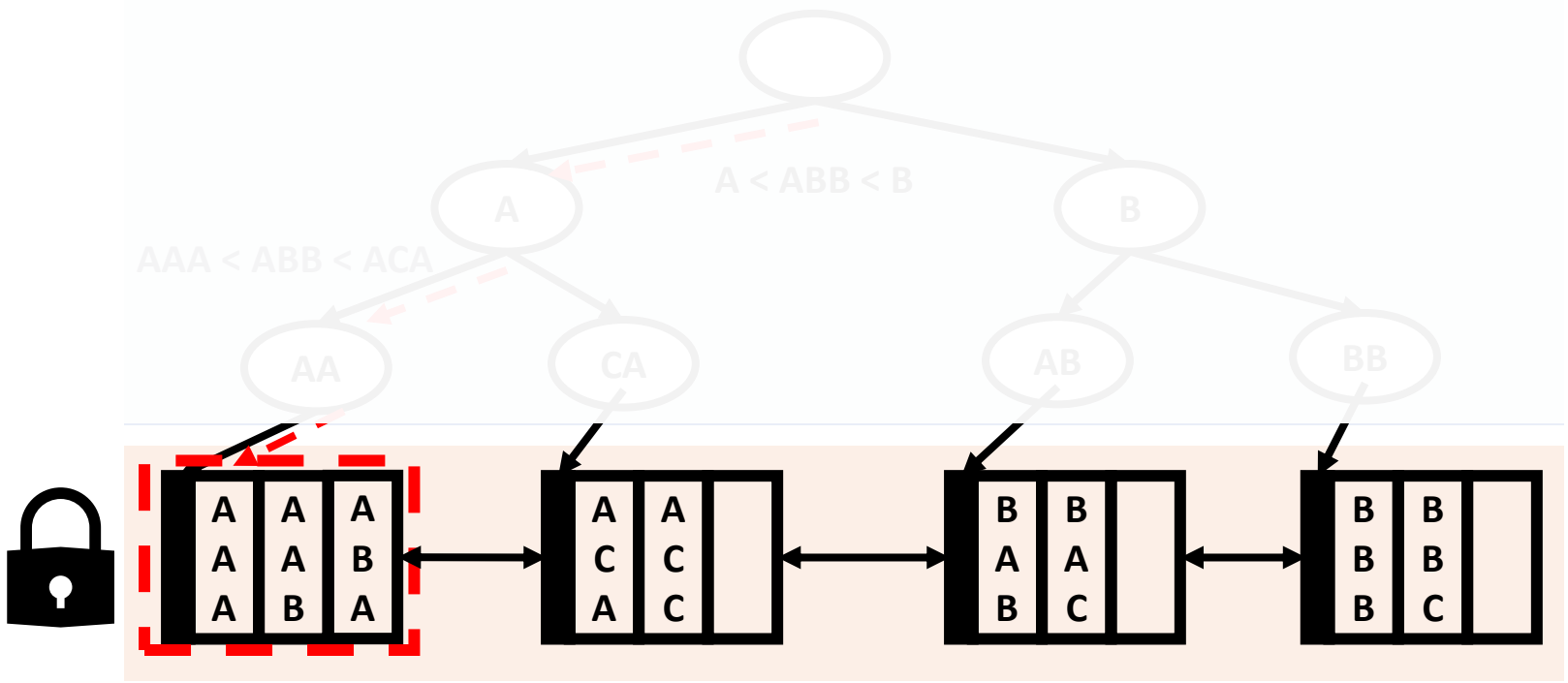


Jump node

Target node

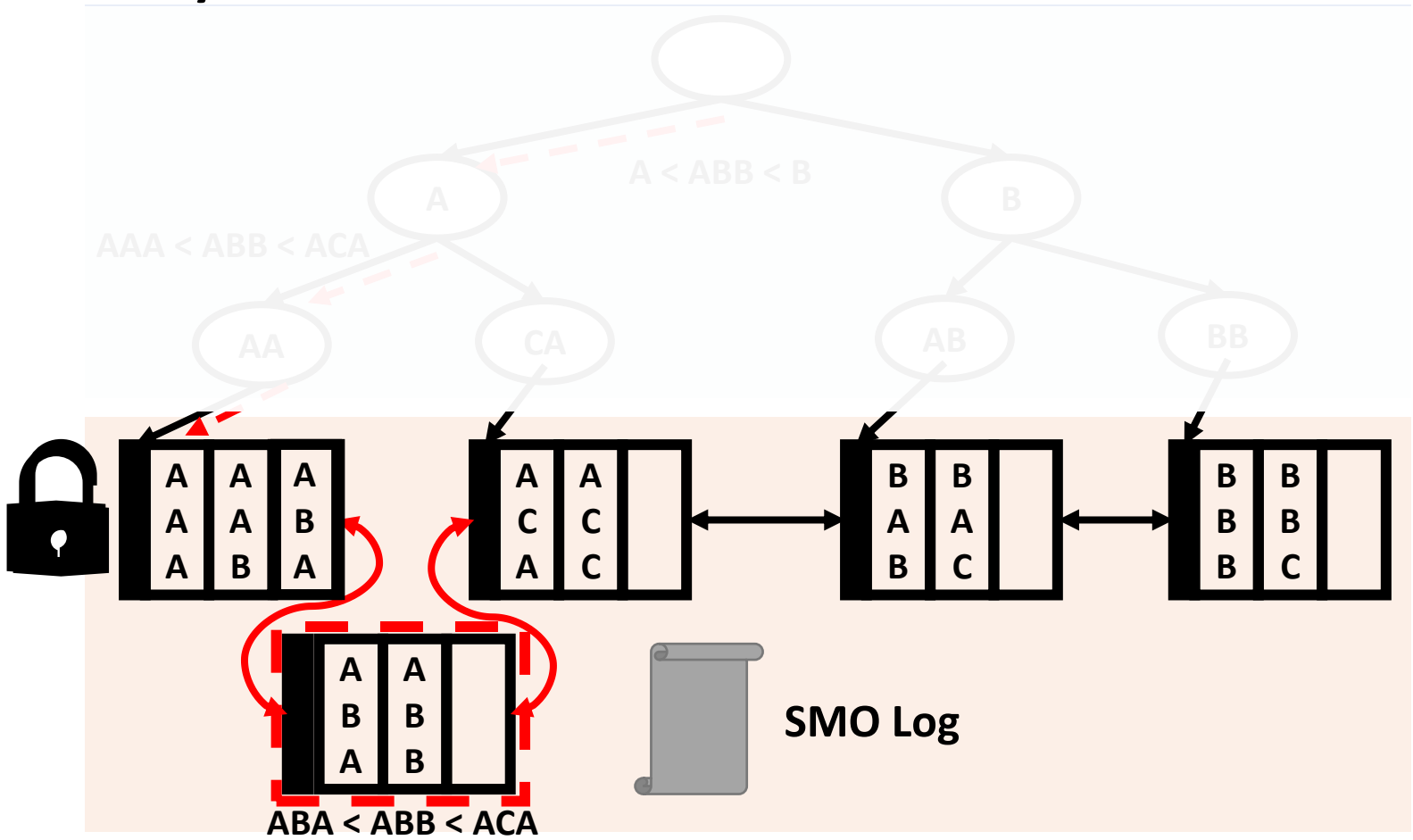
Asynchronous Update: Data layer update

Insert the key ‘ABB’



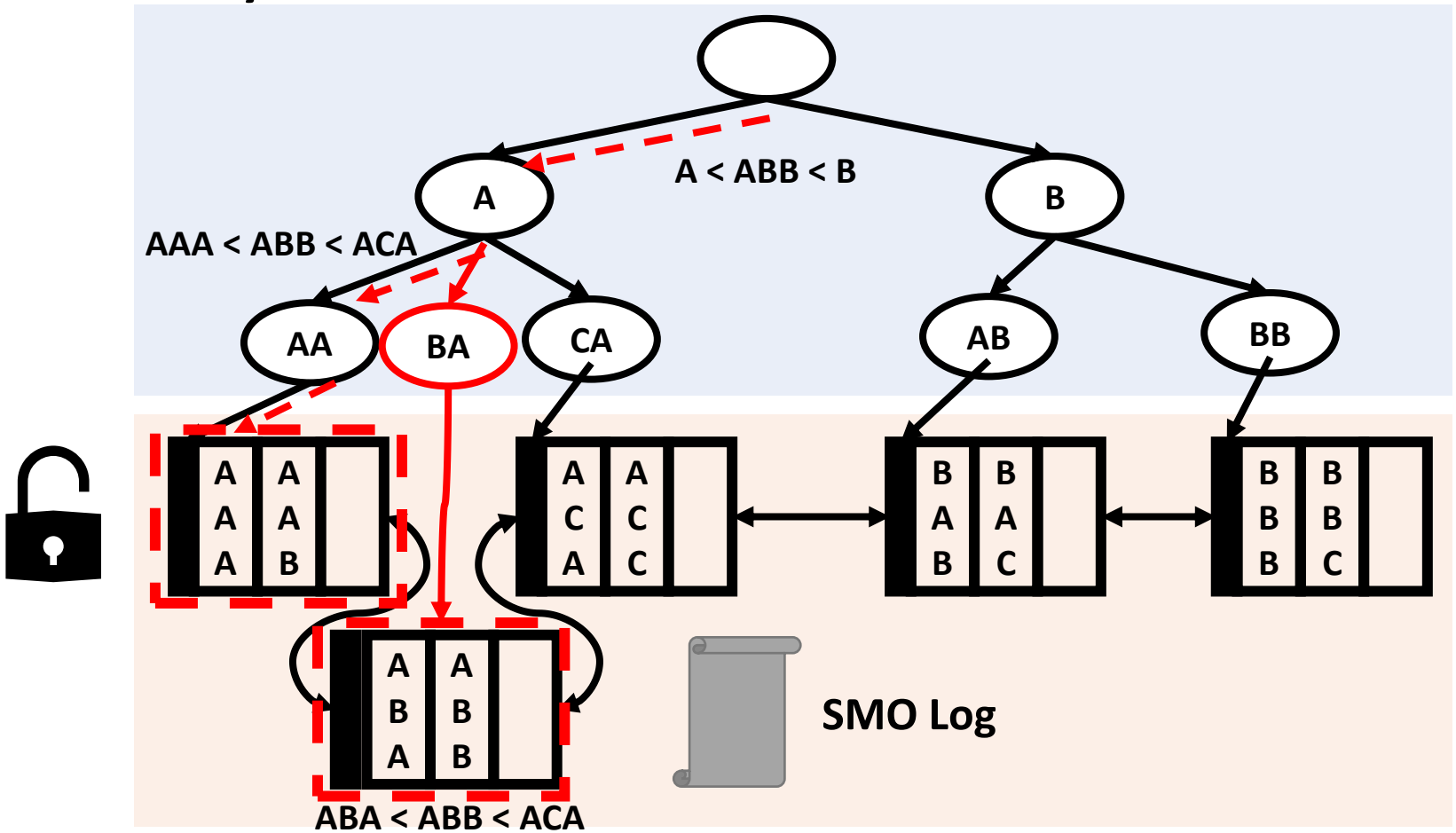
Asynchronous Update: Data layer update done

Insert the key ‘ABB’



Asynchronous Update: Search layer update

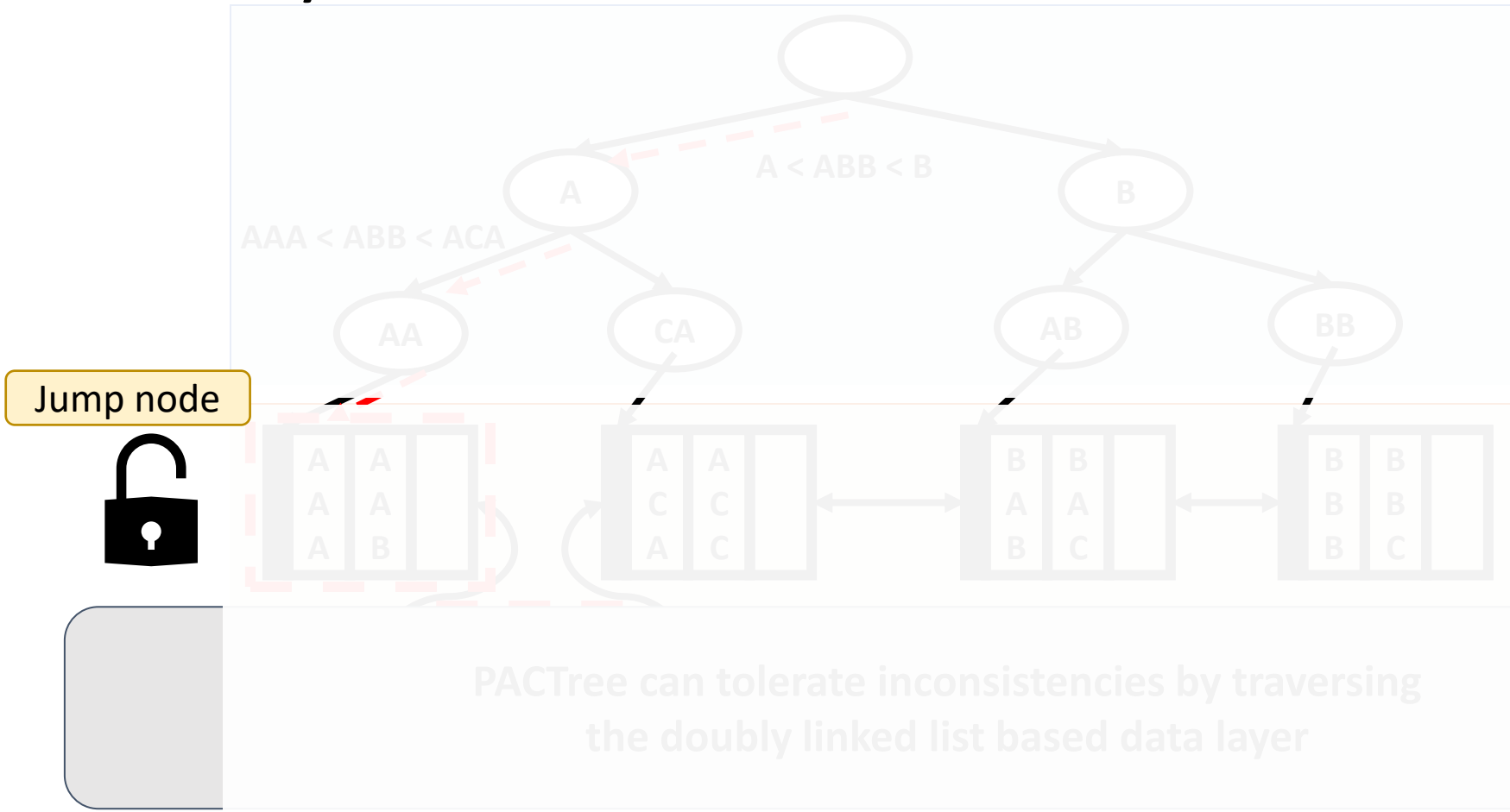
Insert the key ‘ABB’



Background Thread

Asynchronous Update: Tolerating Inconsistency

Find the key ‘ABB’



Talk outline

- Background
- Packed Asynchronous Concurrency (PAC) Guidelines
- PACTree : A High Performance Persistent Range Index Using PAC Guidelines
- **Evaluation**
- Conclusion

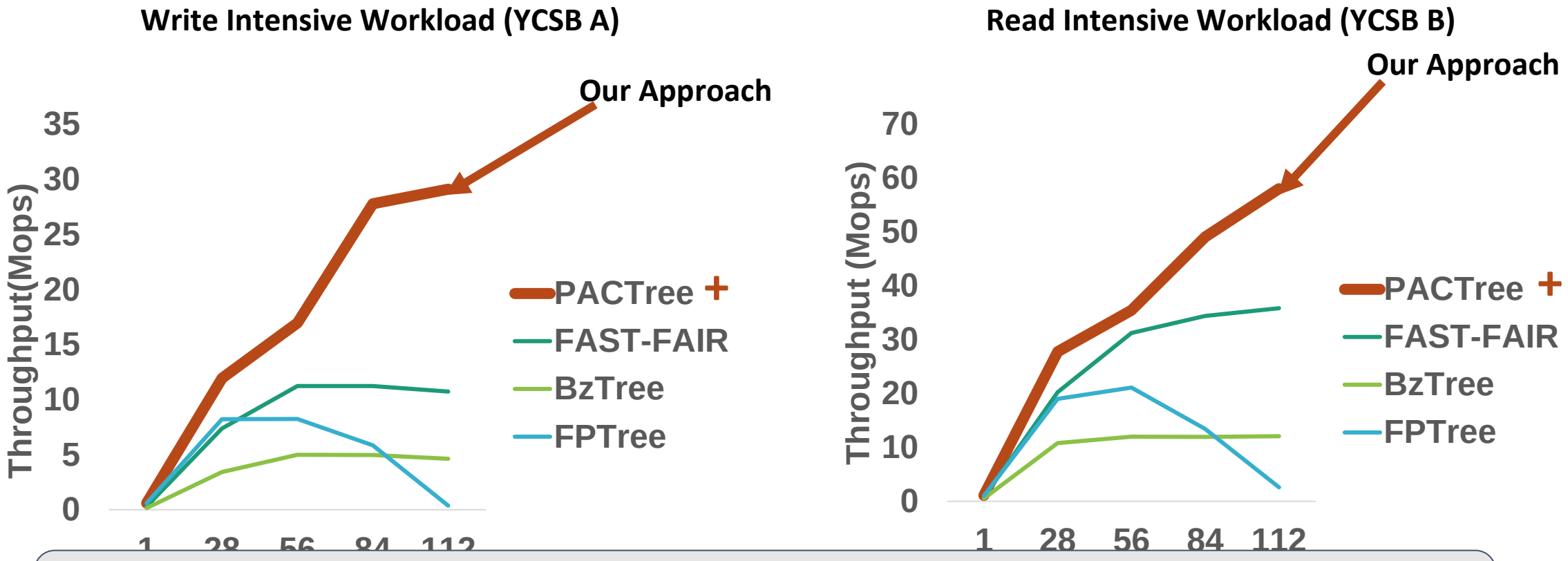
Evaluation Environment

- **Two Socket Real NVM machine**
  - 2 x Intel Xeon Platinum 8280 processors (28 physical/56 logical cores) per socket,
  - 3.0 TB of DCPMM (256GB x 6 (1.5TB) per-socket), and 768 GB of DRAM

- **Workloads**
  - YCSB Workloads
    - Write intensive, Read intensive, and Scan workload

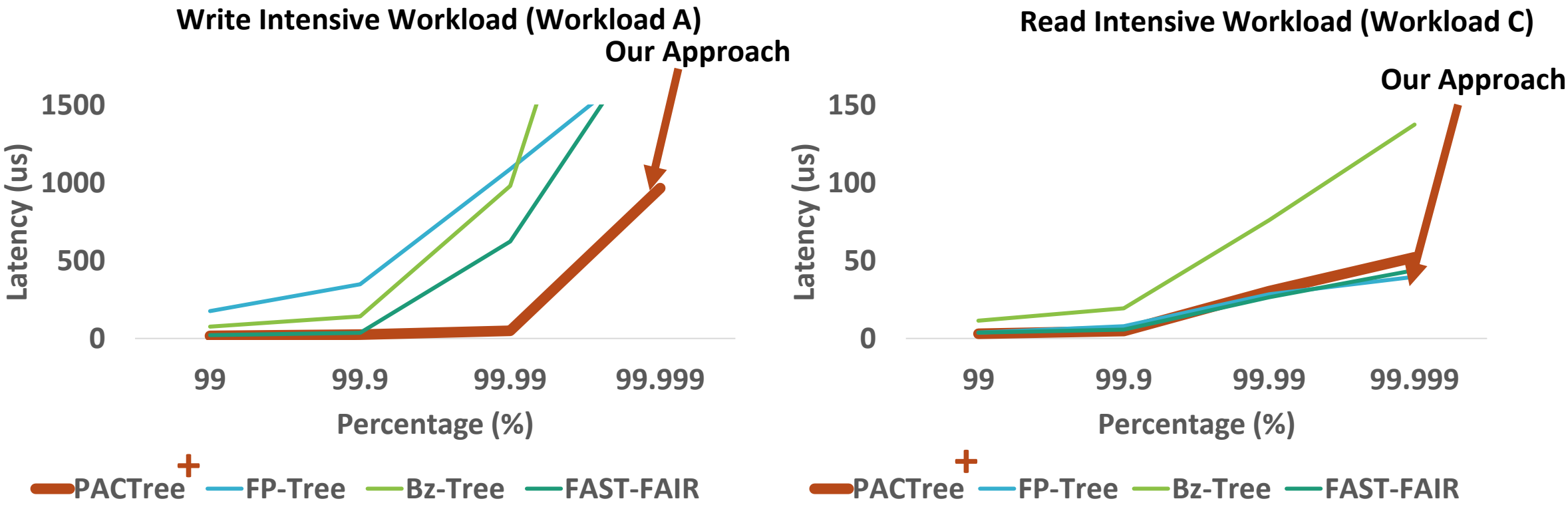
- **Competitors**
  - **FPTree** [SIGMOD'17] : DRAM/NVM hybrid B+-tree
  - **FAST-FAIR** [FAST'18] : Persistent B+-tree supporting non-blocking read
  - **BzTree** [VLDB'18] : Lock-free persistent B+-tree

Evaluation result: Throughput of YCSB



PACTree shows better scalability and performance than other persistent indexes

Evaluation Result: Tail Latency of YCSB



PACTree shows low tail latency
in both write intensive workload and read intensive workload

Talk outline

- Background
- Packed Asynchronous Concurrency (PAC) Guidelines
- PACTree : A High Performance Persistent Range Index Using PAC Guidelines
- Evaluation
- **Conclusion**

Conclusion

- **PAC guideline: design guidelines for persistent index**
  - Access in a **packed fashion** to save the limited NVM bandwidth
  - Exploit **asynchronous concurrency** control to decouple the long NVM latency from the critical path.

- **PACTree: high-performance persistent index designed under the PAC guideline**
  - Packed Access to NVM using Trie-based Search Layer
  - Asynchronous update for Search Layer and Data Layer

- **PAC guidelines can be applied to other NVM software design**
  - Performance critical system software (e.g., File systems, Database systems)
  - In-memory storage systems that use NVM as large volatile memory