上海科技大学
ShanghaiTech University

CS211
Advanced Computer Architecture

L09 Advanced Topics on Cache and Memory

Chundong Wang

October 22nd, 2025

立志成才 报国裕民

Promising Resistive Memory Technologies

- PCM
  - Inject current to change material phase
  - Resistance determined by phase
- STT-MRAM
  - Inject current to change magnet polarity
  - Resistance determined by polarity
- Memristors/RRAM/ReRAM
  - Inject current to change atomic structure
  - Resistance determined by atom distance

Opportunity: PCM Advantages

- Scales better than DRAM, Flash
  - Requires current pulses, which scale linearly with feature size
  - Expected to scale to 9nm (2022 [ITRS])
  - Prototyped at 20nm (Raoux+, IBM JRD 2008)
- Can be denser than DRAM
  - Can store multiple bits per cell due to large resistance range
  - Prototypes with 2 bits/cell in ISSCC'08, 4 bits/cell by 2012
- Non-volatile
  - Retain data for >10 years at 85°C
- No refresh needed, low idle power

Phase Change Memory Properties

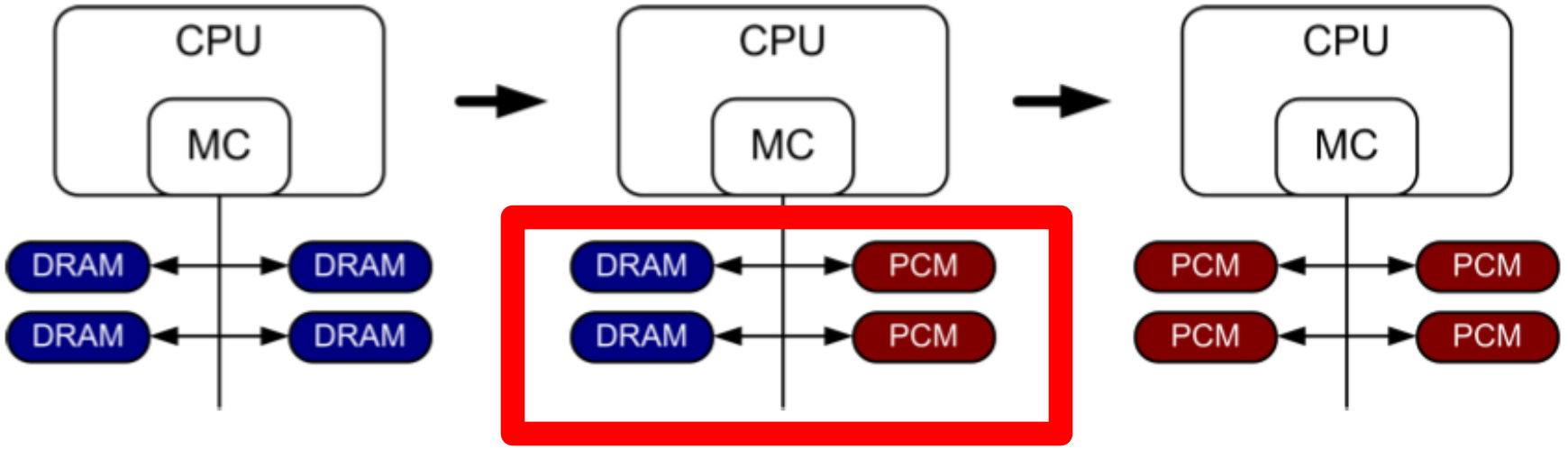
- Dynamic Energy
  - 40 uA Rd, 150 uA Wr
  - 2-43x DRAM, 1x NAND Flash
- Endurance
  - Writes induce phase change at 650C
  - Contacts degrade from thermal expansion/contraction
  - 10^8 writes per cell
  - 10^{-8}x DRAM, 10^3x NAND Flash
- Cell Size
  - 9-12F^2 using BJT, single-level cells
  - 1.5x DRAM, 2-3x NAND (will scale with feature size, MLC)

Phase Change Memory: Pros and Cons

- Pros over DRAM
  - Better technology scaling (capacity and cost)
  - Non volatile → Persistent
  - Low idle power (no refresh)
- Cons
  - Higher latencies: ~4-15x DRAM (especially write)
  - Higher active energy: ~2-50x DRAM (especially write)
  - Lower endurance (a cell dies after ~10^8 writes)
  - Reliability issues (resistance drift)
- Challenges in enabling PCM as DRAM replacement/helper:
  - Mitigate PCM shortcomings
  - Find the right way to place PCM in the system

PCM-based Main Memory (I)

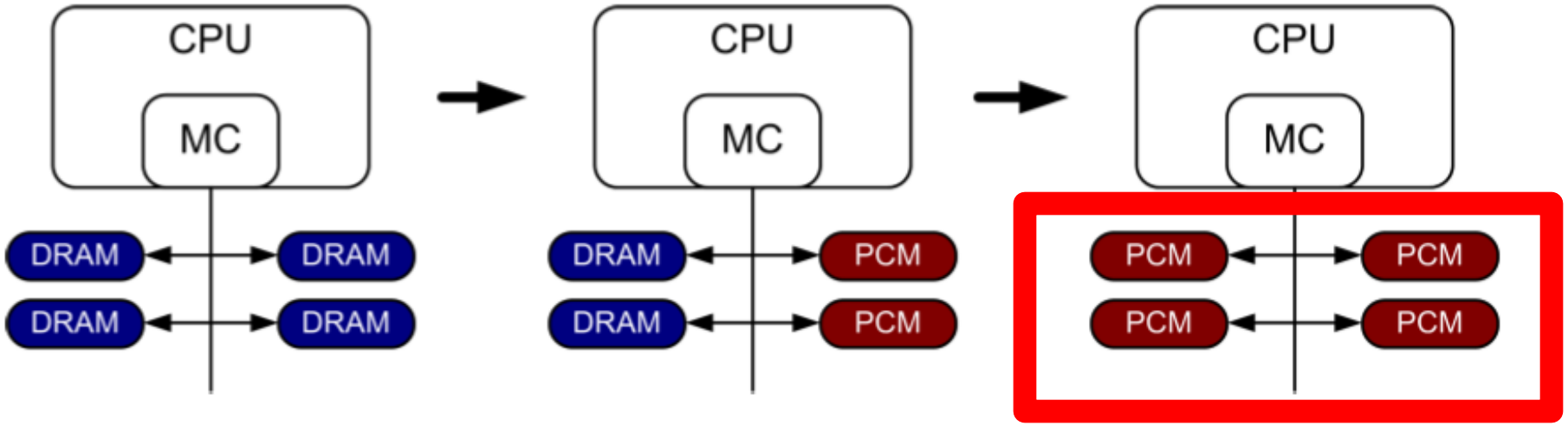
- How should PCM-based (main) memory be organized?



- Hybrid PCM+DRAM [Qureshi+ ISCA'09, Dhiman+ DAC'09]:
  - How to partition/migrate data between PCM and DRAM

PCM-based Main Memory (II)

- How should PCM-based (main) memory be organized?



- Pure PCM main memory [Lee et al., ISCA'09, Top Picks'10]:
  - How to redesign entire hierarchy (and cores) to overcome PCM shortcomings

Wear Leveling of PCM

- Wear leveling
  - Limited write endurance
  - A mechanism that tries to make the writes uniform by remapping heavily written lines to less frequently written lines

Start-gap algorithm

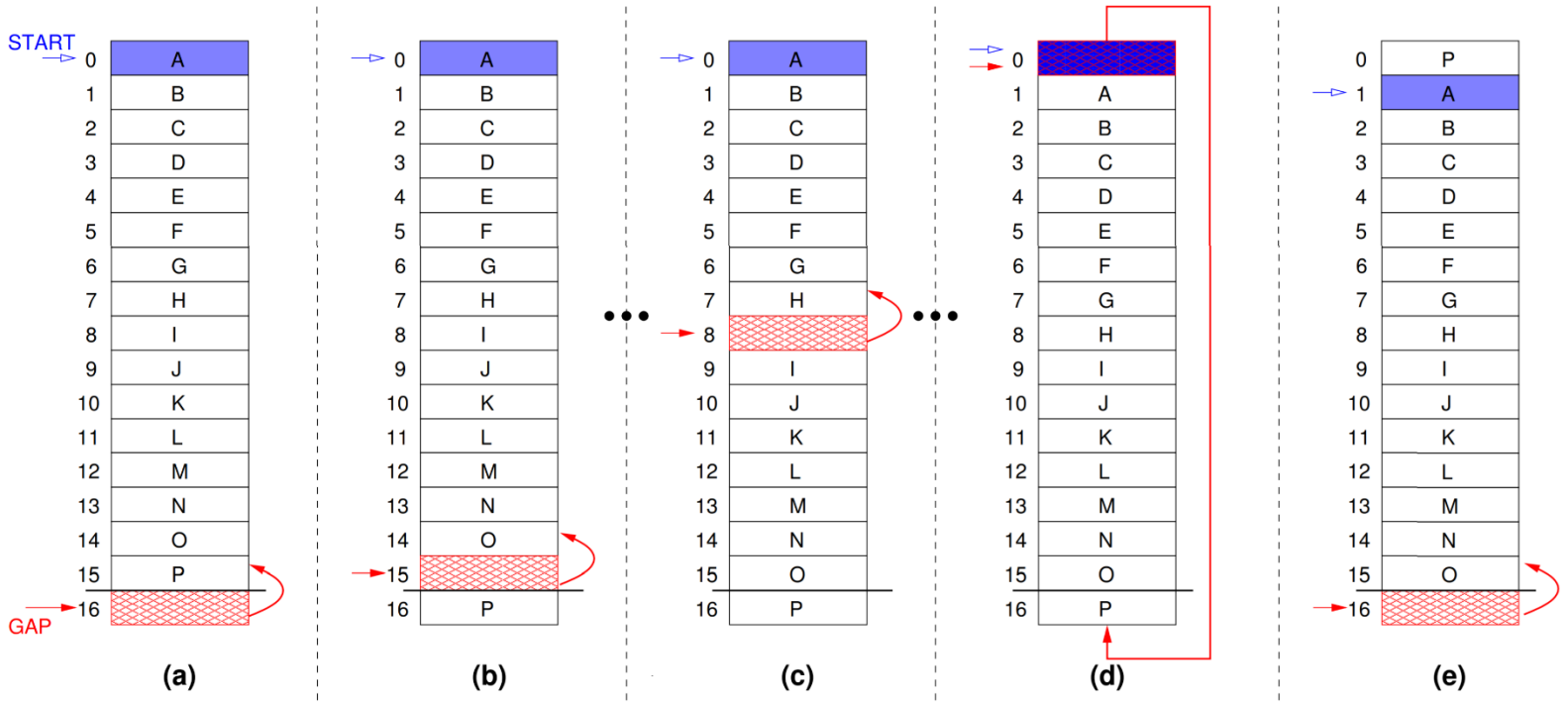


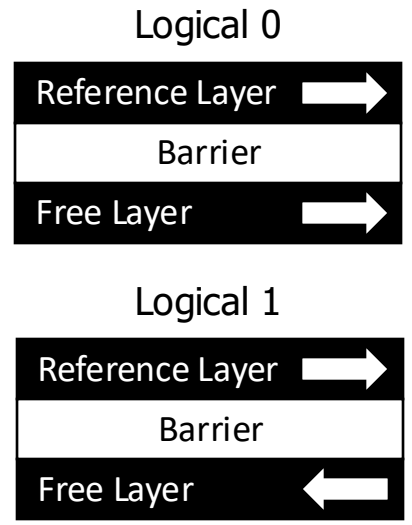
Figure 5: Start-Gap wear leveling on a memory containing 16 lines.

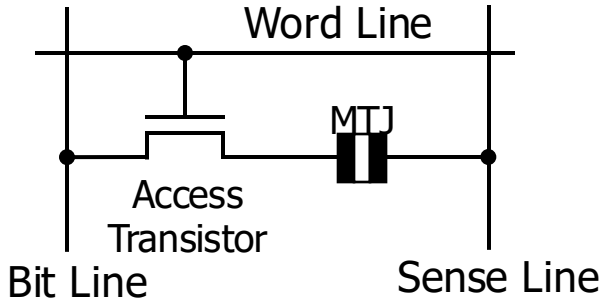
- A start line and a gap line
- Gap line: a memory location without useful data
- Start line: initially pointing to location 0
- Every *n* writes, gap to move up by one (gap--);
- Once start meets gap, start to move down by one (start++).

https://ieeexplore.ieee.org/document/5375309

STT-MRAM

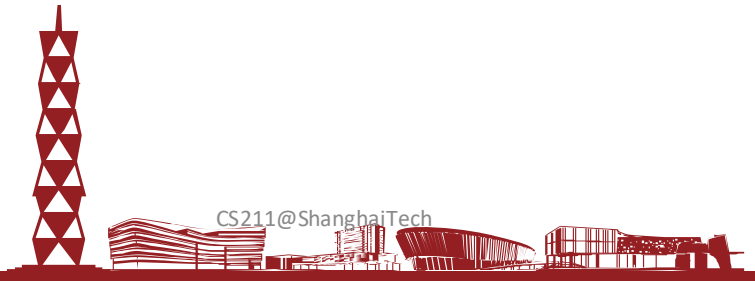
- A candidate as “Universal Memory”
  - Cache, main memory, …
- Magnetic Tunnel Junction (MTJ) device
  - Reference layer: Fixed magnetic orientation
  - Free layer: Parallel or anti-parallel
- Magnetic orientation of the free layer determines logical state of device
  - High vs. low resistance
- Write: Push large current through MTJ to change orientation of free layer
- Read: Sense current flow





STT-MRAM: Pros and Cons

- Pros over DRAM
  - Better technology scaling (capacity and cost)
  - Non volatile → Persistent
  - Low idle power (no refresh)
- Cons
  - Higher write latency
  - Higher write energy
  - Poor density (currently)
  - Reliability?
- Another level of freedom
  - Can trade off non-volatility for lower write latency/energy (by reducing the size of the MTJ)



STT-MRAM as cache

- Multi-Retention Level STT-RAM Cache Designs with a Dynamic Refresh Scheme [Sun et al., MICRO '11]
- Data retention time
  - How long data can be retained in a memory cell after being written
  - Non-volatility of STT-MRAM is supposed to be 4-10 years
- Observations
  - Non-volatility can be relaxed, say, from years to be micro-seconds, by reducing switching current ➔ Saving energy and improving write performance
  - Any use cases for such relaxed non-volatility?
    - L1, L2, L3 caches, with different data retention time
    - Data not living for years in caches
  - Different retention levels by using different currents
    - Refresh scheme needed to avoid data loss

https://www.comp.nus.edu.sg/~wongwf/papers/MICRO44.pdf

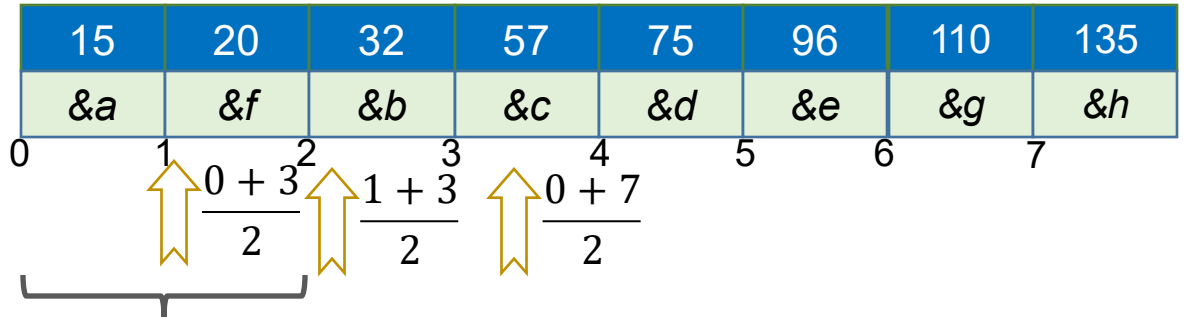
The Impact of Cache

- Binary search vs. linear search

How about he impact of branch predictor and cache prefetch? 😐

Binary search key 32

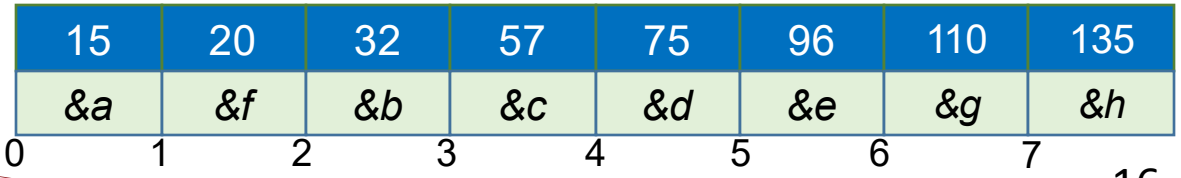
O(logn)



Two cache misses, three comparisons

Linear search key 32

O(n)

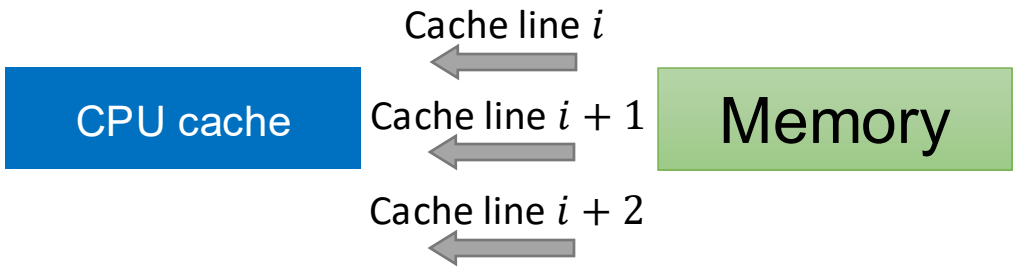


Two cache misses, three comparisons

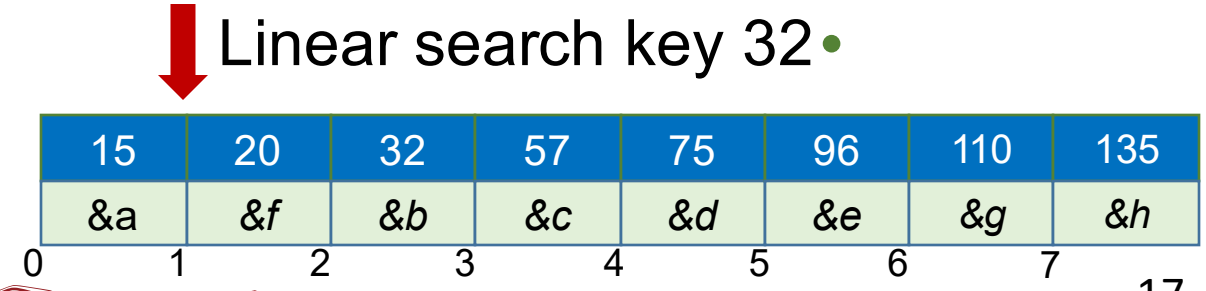
C. Ye and C. Wang, "Boosting the Search Performance of B+-tree with Sentinels for Non-volatile Memory," 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), 2022, pp. 244-249, doi: 10.1109/ASP-DAC52403.2022.9712580.

Linear search

- Read amplification
  - Many comparisons before the target key is found
  - Worst case: all keys compared



Is linear search optimal?



C. Ye and C. Wang, "Boosting the Search Performance of B+-tree with Sentinels for Non-volatile Memory," 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), 2022, pp. 244-249, doi: 10.1109/ASP-DAC52403.2022.9712580.

The less, the better

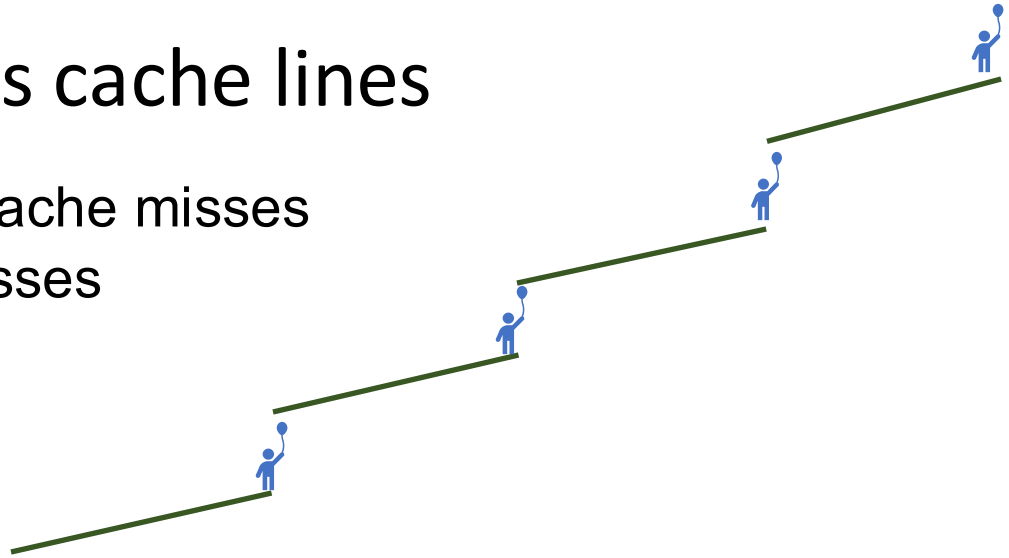
- To further reduce cache misses
  - A sorted array spans contiguous cache lines

Normal linear search, three cache misses
With sentinels, two cache misses

Find key 75

20	57	96	135

A small sentinel array with fewer cache lines



Keys monotonically increasing in cache lines

One cache line

15	20	32	57	75	96	110	135
&a	*&f*	*&b*	*&c*	*&d*	*&e*	*&g*	*&h*
0	1	2	3	4	5	6	7

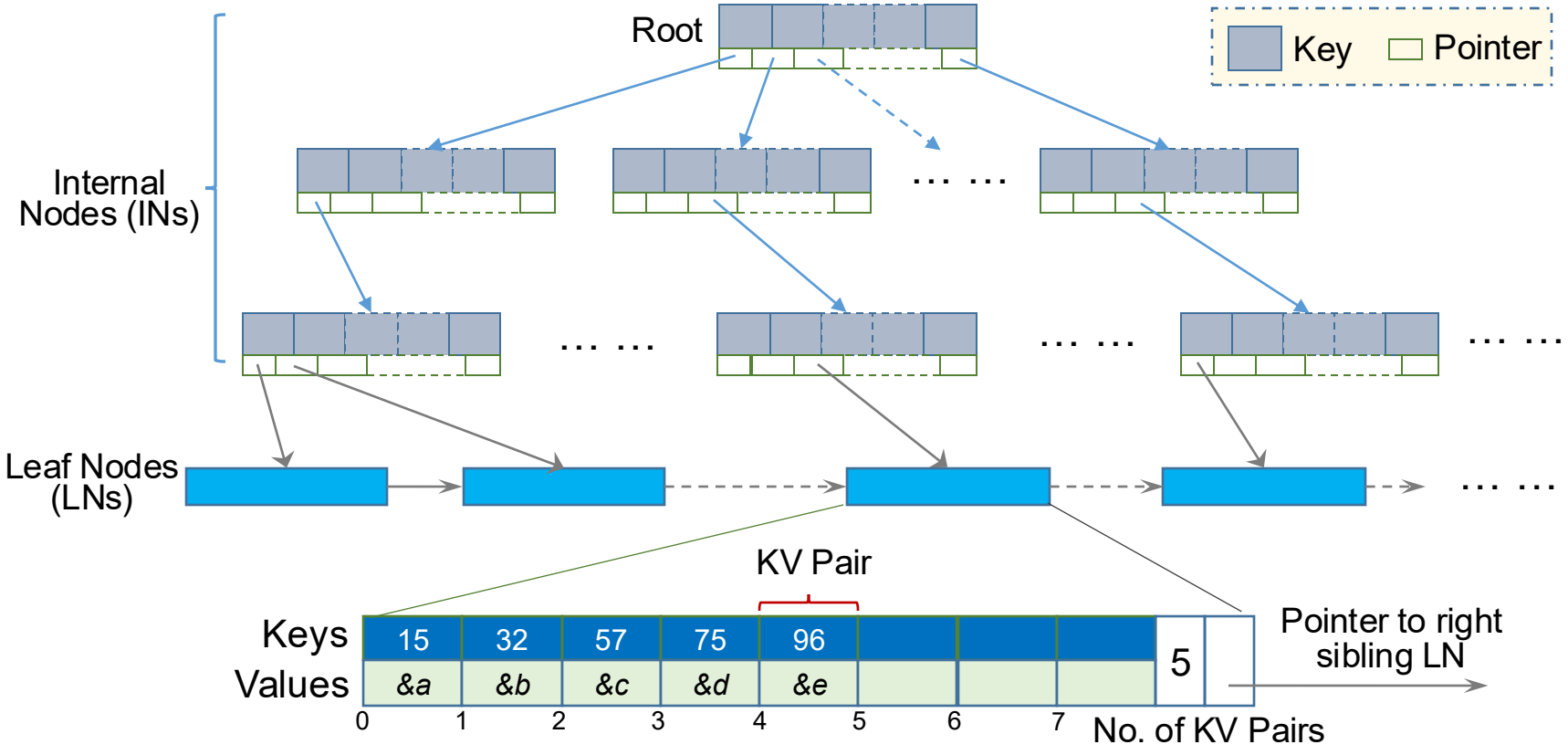
Less read amplification for higher performance

C. Ye and C. Wang, "Boosting the Search Performance of B+-tree with Sentinels for Non-volatile Memory," 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), 2022, pp. 244-249, doi: 10.1109/ASP-DAC52403.2022.9712580.

In-NVM B+-tree

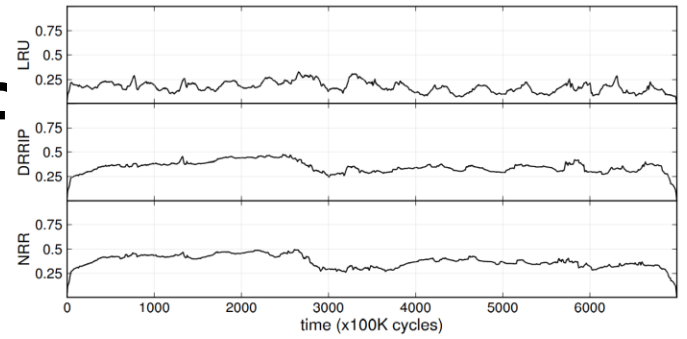
- From root to leaves, nodes well structured

A standard B+-tree node is a sorted array



C. Ye and C. Wang, "Boosting the Search Performance of B+-tree with Sentinels for Non-volatile Memory," 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), 2022, pp. 244-249, doi: 10.1109/ASP-DAC52403.2022.9712580.

Smaller LLC, higher perf



(a) Changes in the fraction of live lines over time

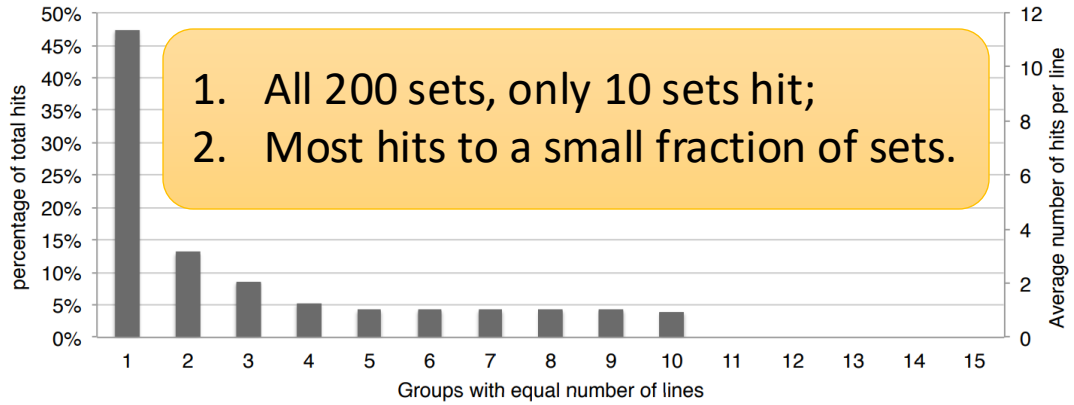
Half to 83.8% LLC lines not useful

- What is the problem?
  - A considerable portion of the shared SLLC is dead
- Why?
  - LLC accesses are ones missed in L1 and L2 ←Quantity & frequency
  - Locality in LLC not realistic ←Access pattern
  - LLC uniformly handles accesses ←Cache management
- How to resolve the problem?
  - Leveraging reuse locality to **selectively** allocate LLC space

Jorge Albericio, Pablo Ibáñez, Víctor Viñals, and José M. Llabería. 2013. The reuse cache: downsizing the shared last-level cache. In Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-46). Association for Computing Machinery, New York, NY, USA, 310–321.

Reuse locality for allocat

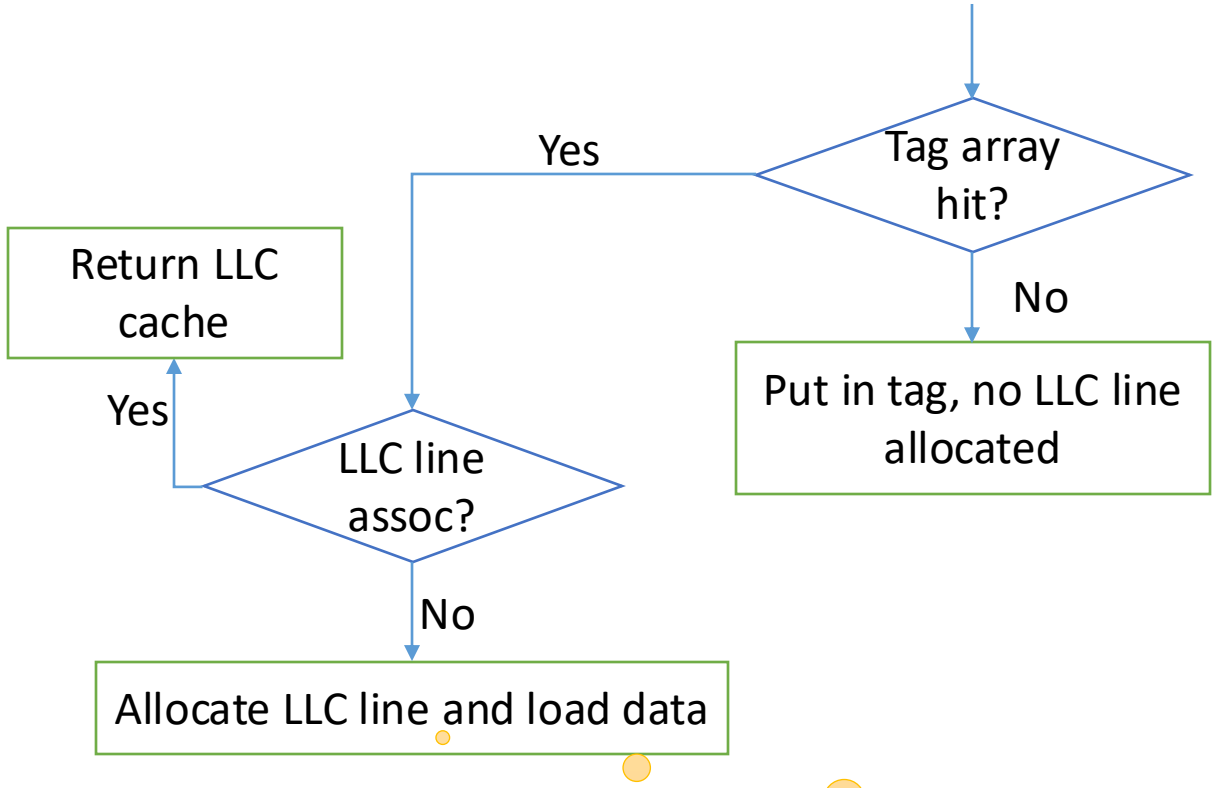
- Reuse locality

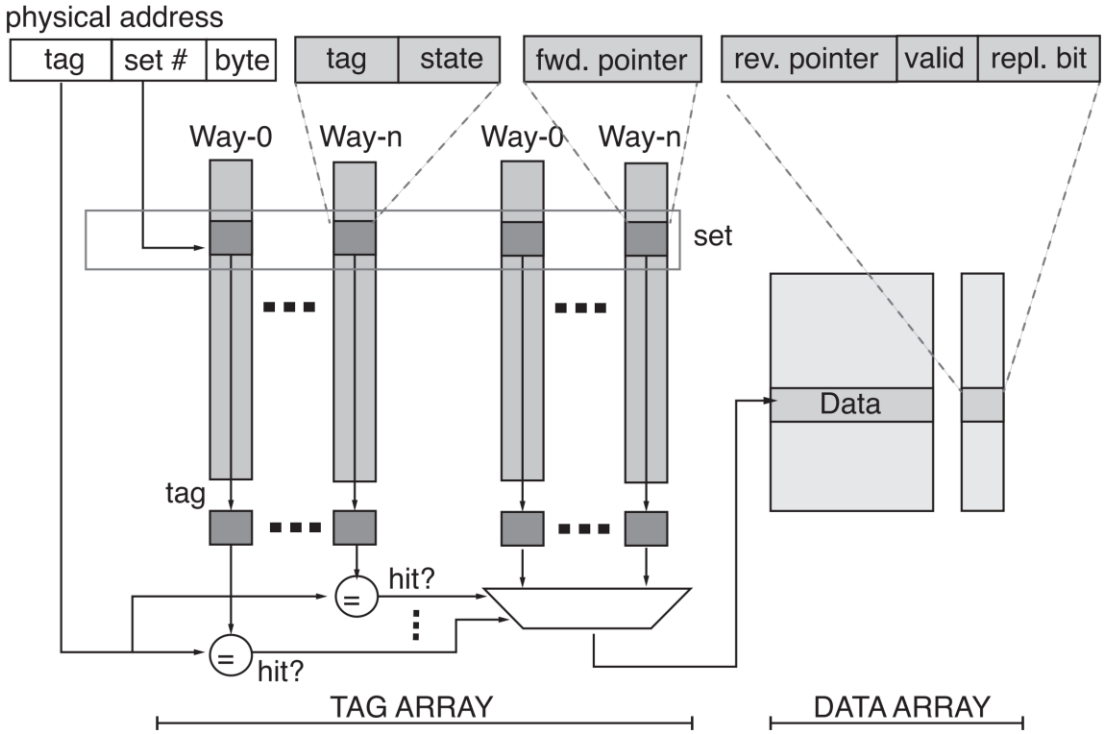


(b) Distribution of hits among all lines loaded (or reloaded) into the LRU SLLC during their stay. Each group represents 0.5% of the loaded lines

- Selective allocation
  - Tag and cache line decoupled
    - Classic cache, one tag to one cache line
    - Now: mores tag as place holders
  - Only reused cache line kept

Selective allocation

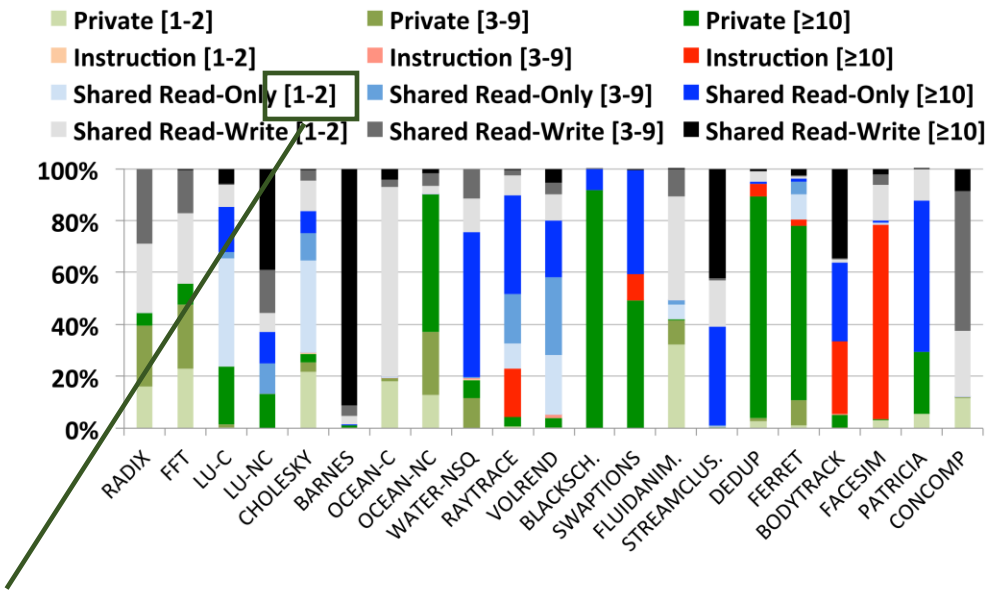




On replacement, tag would be retained for the cache line. Once re-hit (reuse) ➔reload data.

Replicate data for high performance

- What is the problem?
  - Multi-cores sharing LLC, different distances to LLC location
  - "Put data closer to my core"
- Why?
  - Non-uniform Access Cache (NUCA)
- How to resolve the problem?
  - Replicate cache lines for the core

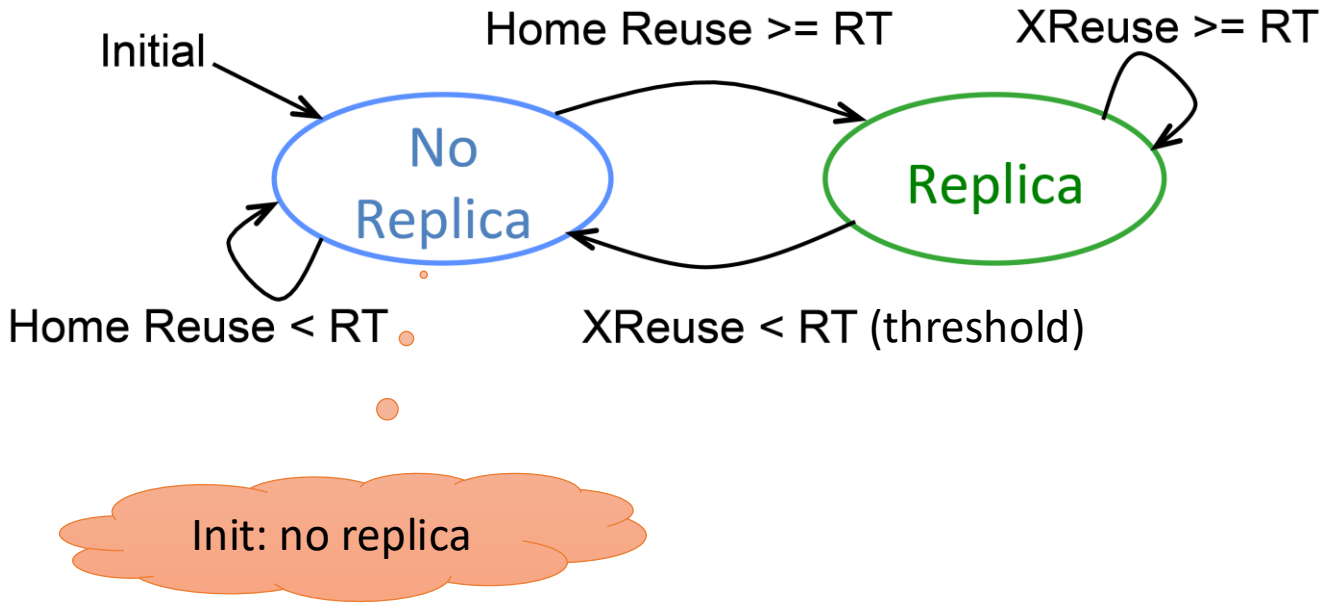


Run-length is defined as the number of accesses to a cache line (at the LLC) from a particular core before a conflicting access by another core or before it is evicted.

G. Kurian, S. Devadas and O. Khan, "Locality-aware data replication in the Last-Level Cache," 2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), 2014, pp. 1-12, doi: 10.1109/HPCA.2014.6835921

How to maintain duplicate cache lines?

- Based on reuse



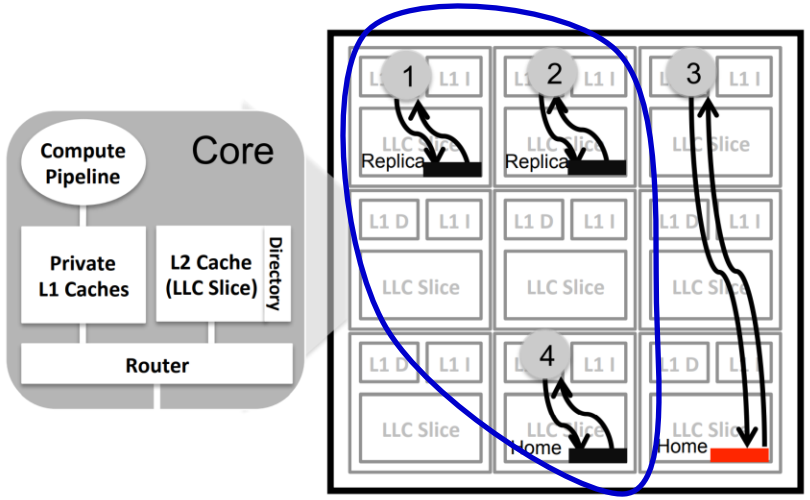
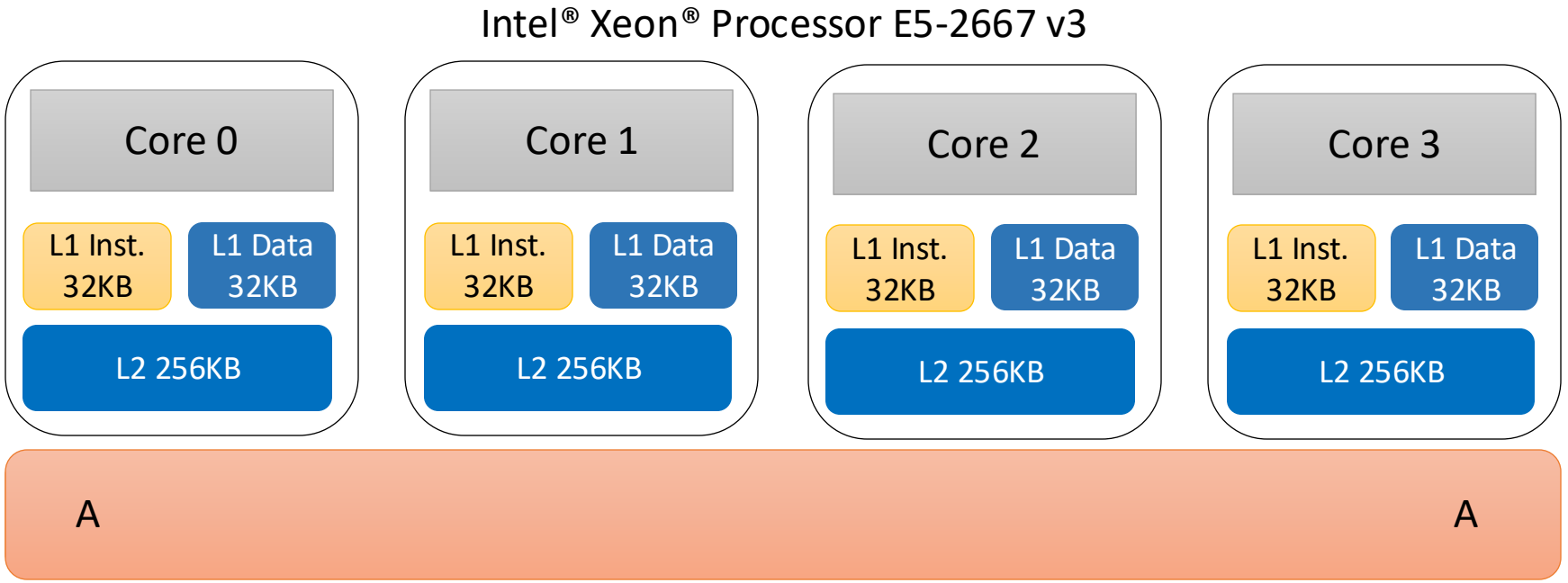
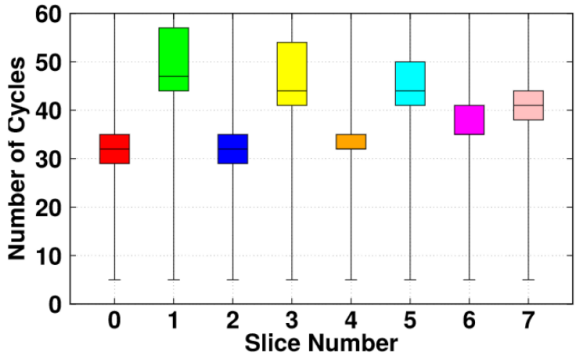


Figure 2. ① – ④ are mockup requests showing the *locality-aware LLC replication* protocol. The *black* data block has high reuse and a local LLC replica is allowed that services requests from ① and ②. The low-reuse *red* data block is not allowed to be replicated at the LLC, and the request from ③ that misses in the L1, must access the LLC slice at its home core. The home core for each data block can also service local private cache misses (e.g., ④).

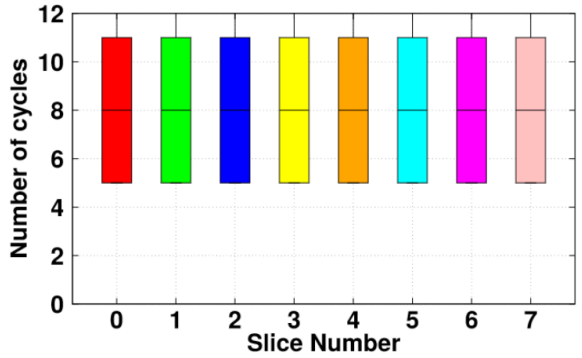
In real CPU's LLC, is it possible to put data closer to a core?

Real LLC slices, time truly differs





(a) Read.



(b) Write.

Alireza Farshin, Amir Roozbeh, Gerald Q. Maguire, and Dejan Kostić. 2019. Make the Most out of Last Level Cache in Intel Processors. In Proceedings of the Fourteenth EuroSys Conference 2019 (EuroSys '19). Association for Computing Machinery, New York, NY, USA, Article 8, 1–17.

To put data in slices closer to a core

- Method
  - Firstly, revers-engineering the algorithm of cache management
  - Secondly, place data according to the management algorithm
- Effect
  - With key-value store, GET performance 12.2%↑

Alireza Farshin, Amir Roozbeh, Gerald Q. Maguire, and Dejan Kostić. 2019. Make the Most out of Last Level Cache in Intel Processors. In Proceedings of the Fourteenth EuroSys Conference 2019 (EuroSys '19). Association for Computing Machinery, New York, NY, USA, Article 8, 1–17.

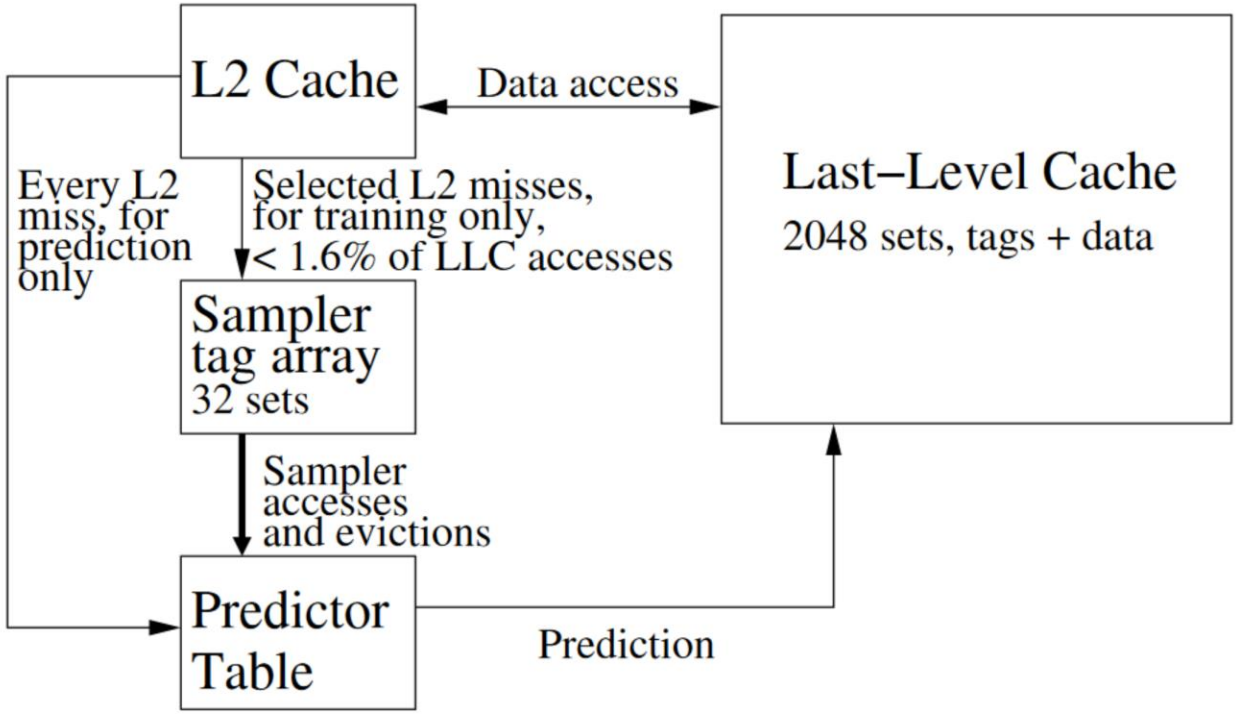
Predicting dead blocks

- What is the problem?
  - How to efficiently predict what blocks are dead
- Why?
  - Many cache lines are dead line (never/hardly used)
- How?
  - Sampling

Samira Manabi Khan, Yingying Tian, and Daniel A. Jimenez. 2010. Sampling Dead Block Prediction for Last-Level Caches. In Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '43). IEEE Computer Society, USA, 175–186. DOI:https://doi.org/10.1109/MICRO.2010.24

Sampling → predicting dead blocks

- Sampling
  - Set-associ
    - 2048 set
  - No need t
    - Lower 1!
    - Partial P



SHiP: Signature-based Hit Predictor

Samira Manabi Khan, Yingying Tian, and Daniel A. Jimenez. 2010. Sampling Dead Block Prediction for Last-Level Caches. In Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '43). IEEE Computer Society, USA, 175–186. DOI:https://doi.org/10.1109/MICRO.2010.24

Carole-Jean Wu, Aamer Jaleel, Will Hasenplaugh, Margaret Martonosi, Simon C. Steely, and Joel Emer. 2011. SHiP: signature-based hit predictor for high performance caching. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture* (*MICRO-44*). Association for Computing Machinery, New York, NY, USA, 430–441.

Dead blocks, from dead pages

- By studying TLB, many pages are dead-on-arrival (DOA)
  - 82% LLTLB entries not used for many apps ← Dead pages
    - 86% of them DOA
- Dead blocks and dead pages
  - > 70% DOA blocks from DOA pages
- Predict dead page (TLB entries)
  - Using PC
  - Using dead pages to locate dead blocks

TABLE III: Percentage of LLC DOA blocks that map on to a DOA page in LLT

Workload	LLC blocks (%)	Workload	LLC blocks (%)
cactusADM	72.22	canneal	64.15
cc	67.76	pr	33.33
cg.B	92.14	graph500	81.40
sssp	93.25	bfs	81.00
lbm	99.98	bc	62.38
Triangle	73.33	mis	62.23
KCore	68.18	mcf	66.18

C. Mazumdar, P. Mitra and A. Basu, "Dead Page and Dead Block Predictors: Cleaning TLBs and Caches Together," *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 507-519, doi: 10.1109/HPCA51647.2021.00050.

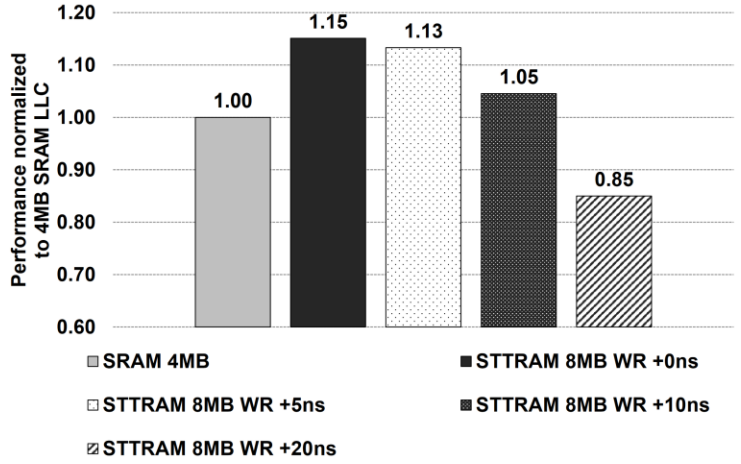
Reduce hit rate of non-volatile LLC

- NVM
  - Phase-change memory, STT-MRAM, Intel Optane memory
  - NVM being used to reshape CPU cache
    - e.g., STT-MRAM
- STT-MRAM vs. SRAM
  - Higher density ➔ Capacity
  - Lower power consumption
  - Write/read asymmetry
    - Write costs longer time than read

Kunal Korgaonkar, Ishwar Bhati, Huichu Liu, Jayesh Gaur, Sasikanth Manipatruni, Sreenivas Subramoney, Tanay Karnik, Steven Swanson, Ian Young, and Hong Wang. 2018. Density tradeoffs of non-volatile memory as a replacement for SRAM based last level cache. In Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA '18). IEEE Press, 315–327.

Lower NV LLC hits ➔ less congestion

- Increased capacity
  - Slower store
  - Writes congested at LLC
- Bypassing LLC but writing to memory
  - LLC hit rate ↓
  - Performance ↑
- What to be bypassed?
  - Dead blocks or ones hardly reused



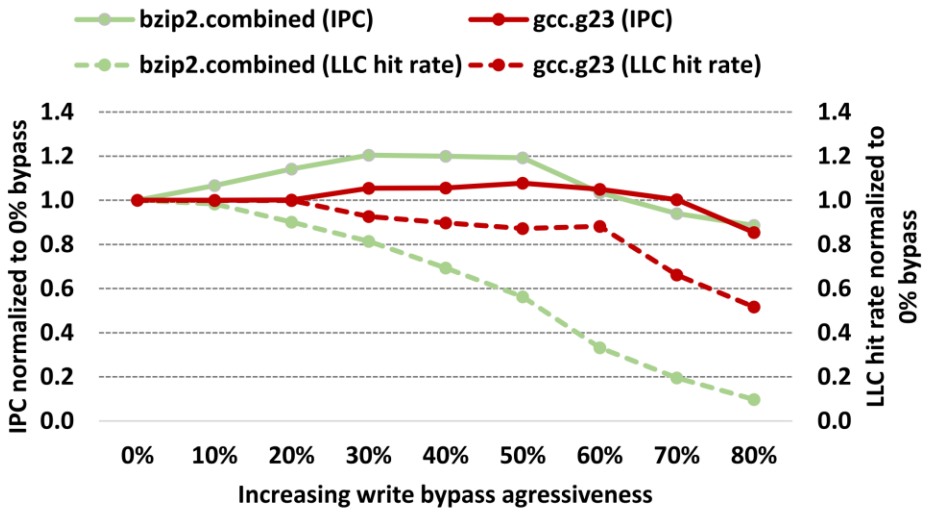
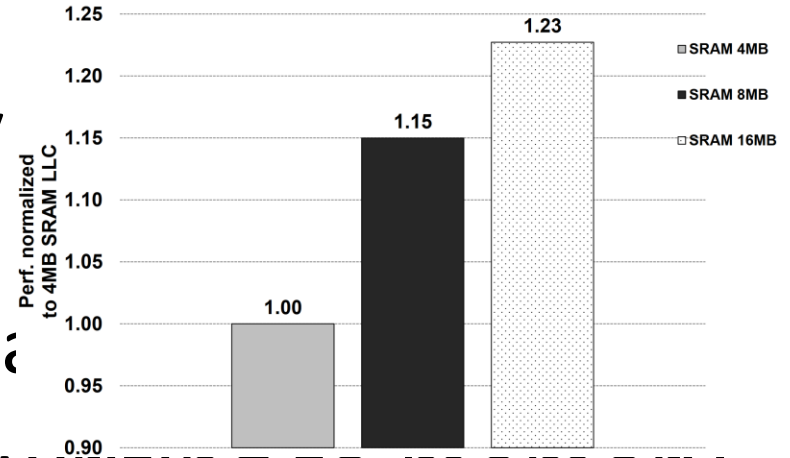
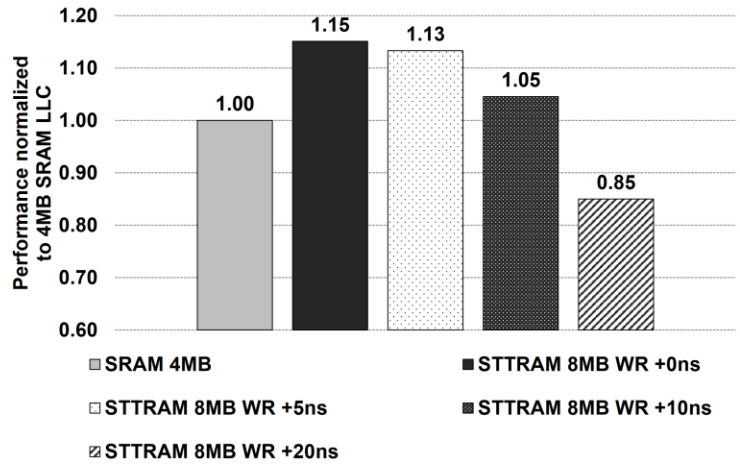


Figure 5: Impact of write bypass on STTRAM LLC

Lower NV LLC hits ➔ less congestion

- Increased capacity
  - Slower store
  - Writes congested a
- Bypassing LLC but writing to memory
  - LLC hit rate ↓
  - Performance ↑
- What to be bypassed?
  - Dead blocks or ones hardly reused





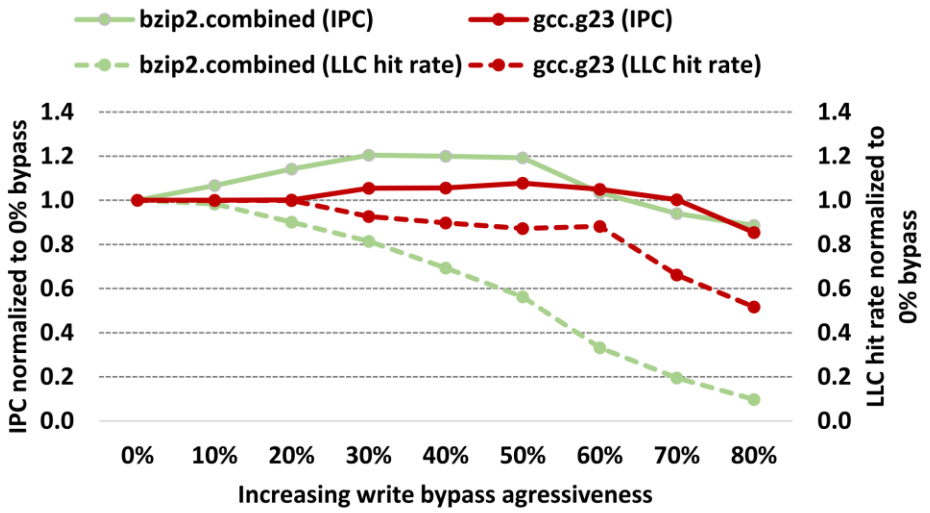


Figure 5: Impact of write bypass on STTRAM LLC

Adjusting Cache for DRAM+NVM main memory

- Intel Optane
  - Read slower than write

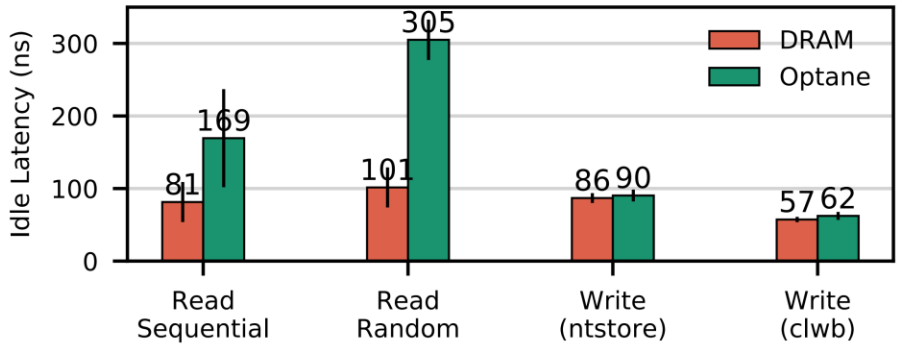
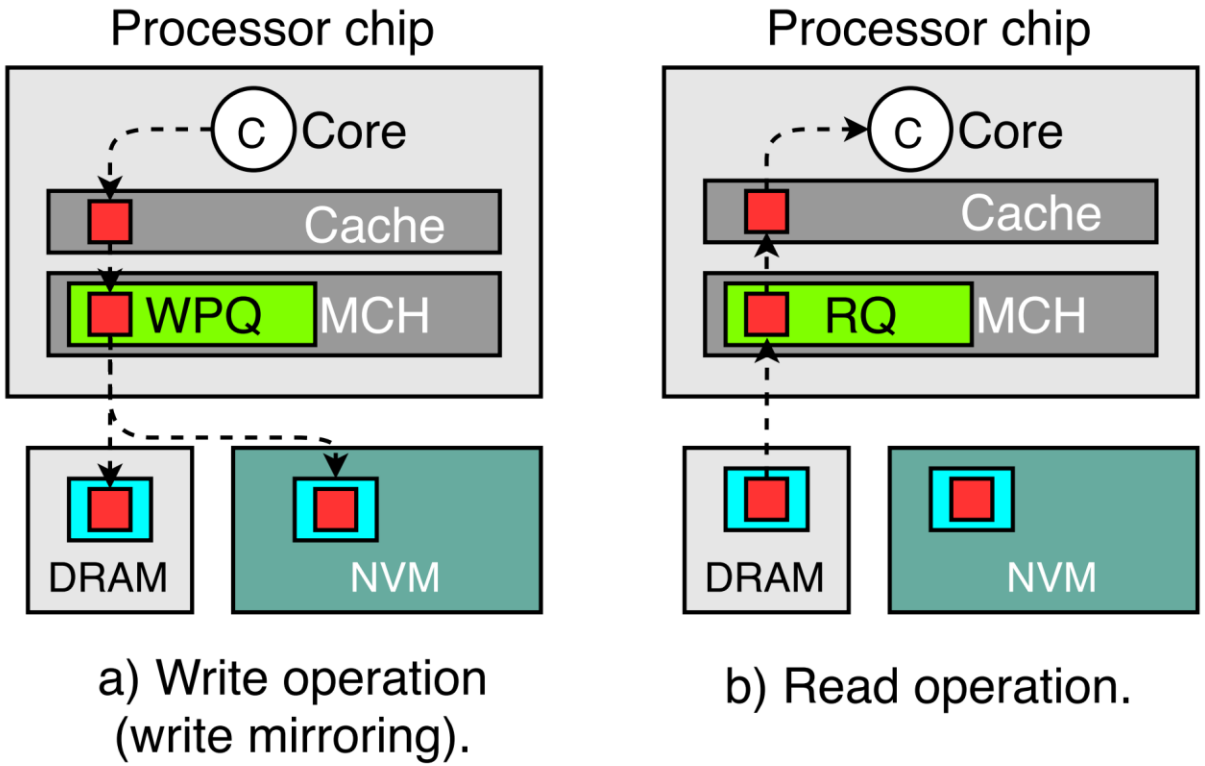


Figure 2: **Best-case latency** An experiment showing random and sequential read latency, as well as write latency using cached write with `clwb` and `ntstore` instructions. Error bars show one standard deviation.



a) Write operation (write mirroring).

b) Read operation.

Fig. 4: Read/Write operations in Stealth-Persist.

Stealth-Persist@HPCA 21

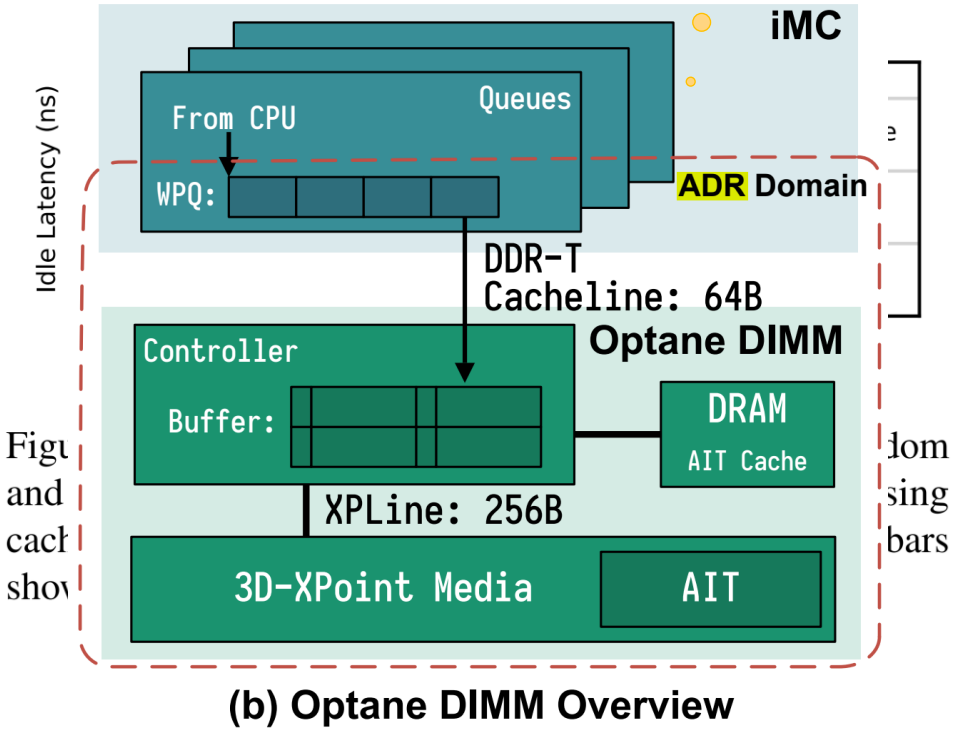
Yang, Jian, et al. “An Empirical Guide to the Behavior and Use of Scalable Persistent Memory.” 18th USENIX Conference on File and Storage Technologies (FAST 20), 2019, pp. 169–182.

M. Alwadi, V. R. Kommareddy, C. Hughes, S. D. Hammond and A. Awad, "Stealth-Persist: Architectural Support for Persistent Applications in Hybrid Memory Systems," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 139-152, doi: 10.1109/HPCA51647.2021.00022.

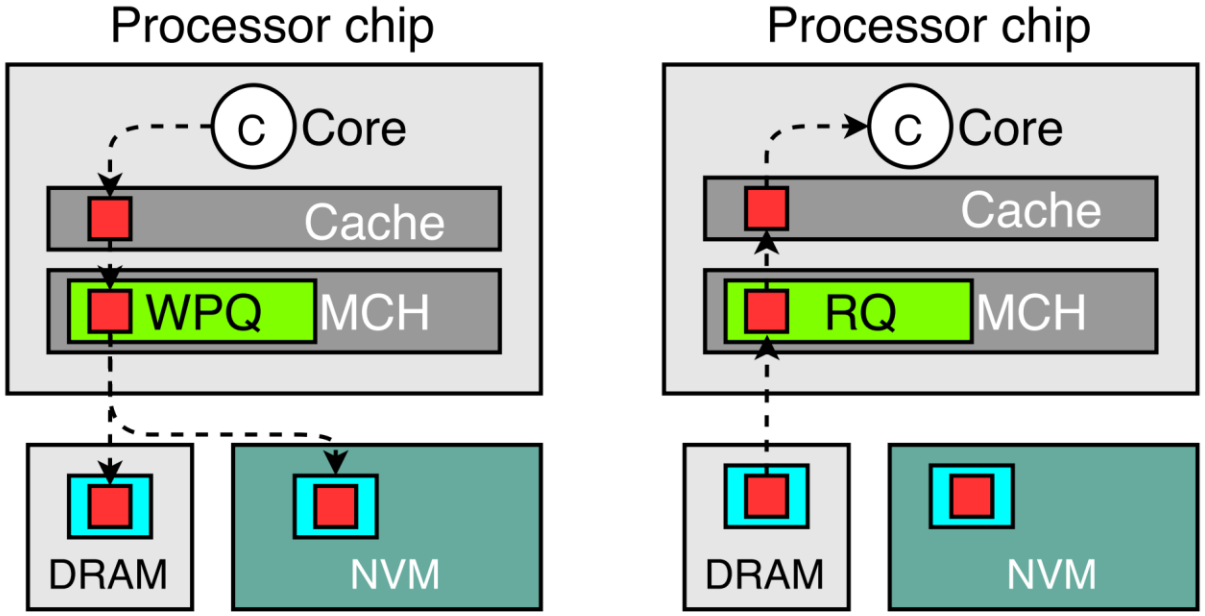
Adjusting Cache for DRAM+NVM main memory

- Intel Optane
  - Read slower than write

ADR+in-pmem buffer



(b) Optane DIMM Overview



a) Write operation (write mirroring).

b) Read operation.

Fig. 4: Read/Write operations in Stealth-Persist.

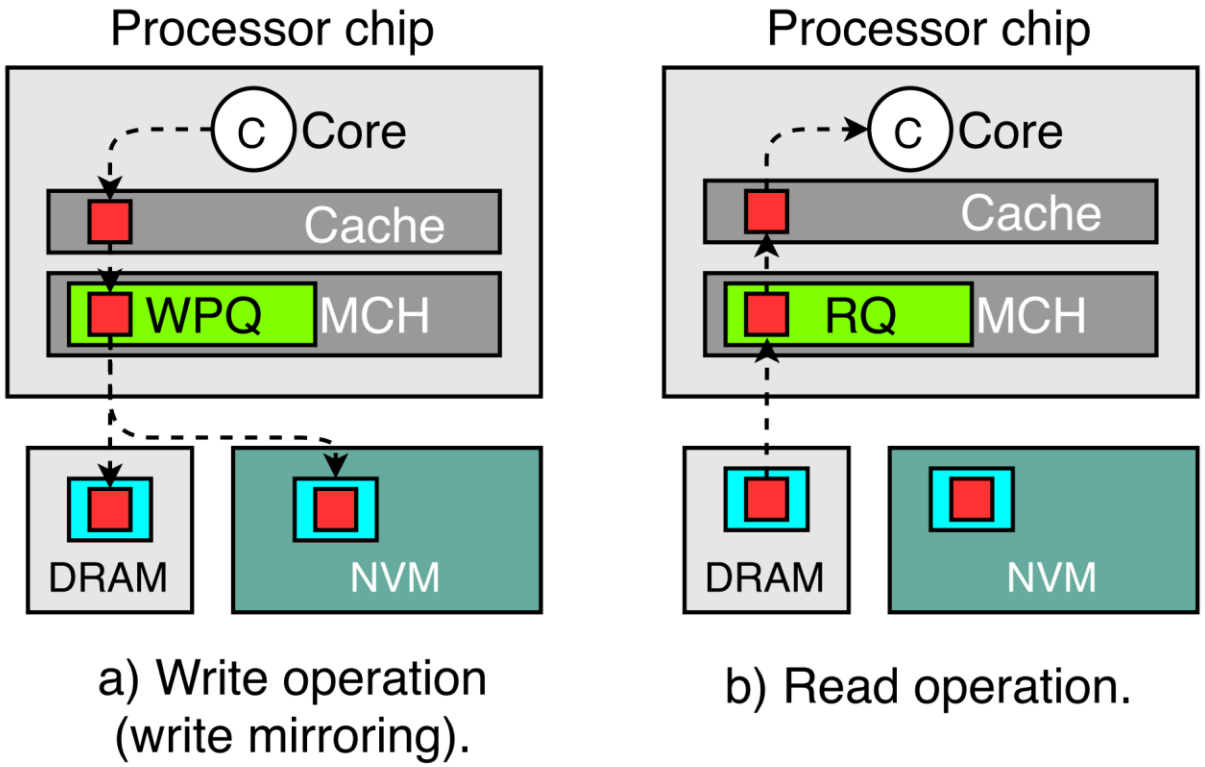
Stealth-Persist@HPCA 21

Yang, Jian, et al. "An Empirical Guide to the Behavior and Use of Scalable Persistent Memory." 18th USENIX Conference on File and Storage Technologies (FAST 20), 2019, pp. 169–182.

M. Alwadi, V. R. Kommareddy, C. Hughes, S. D. Hammond and A. Awad, "Stealth-Persist: Architectural Support for Persistent Applications in Hybrid Memory Systems," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 139-152, doi: 10.1109/HPCA51647.2021.00022.

Adjusting Cache for DRAM+NVM main memory

- Intel Optane
  - Read slower than write
- Read operation
  - Blocking operation
- Blocking write?
  - clflush
  - fsync



a) Write operation (write mirroring).

b) Read operation.

Fig. 4: Read/Write operations in Stealth-Persist.

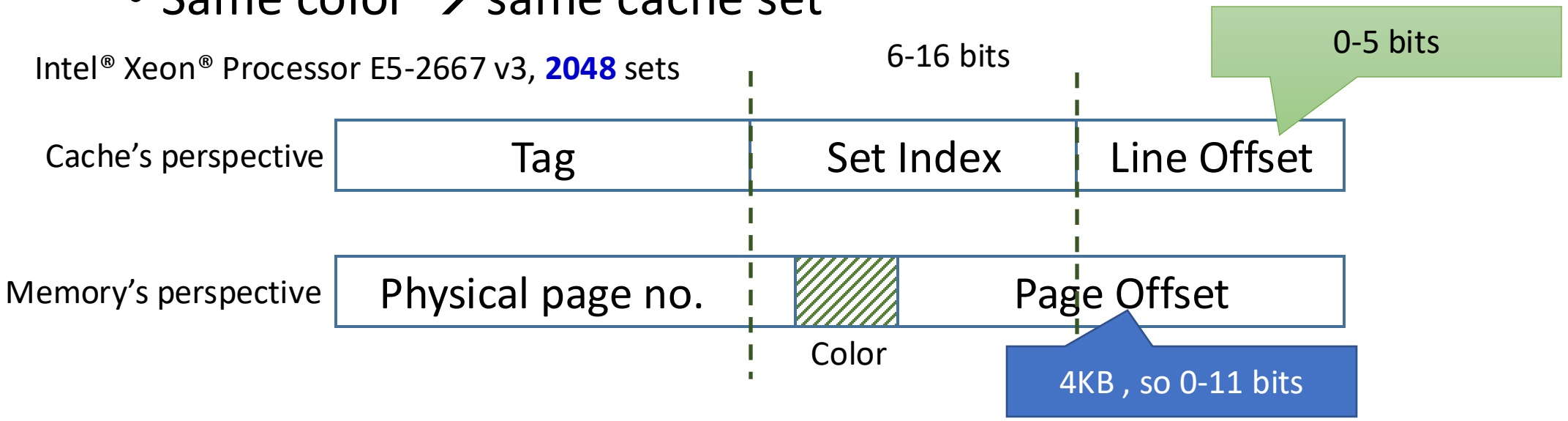
Write: both DRAM/NVM; read: load from DRAM

Ren, Yujie, et al. “CrossFS: A Cross-Layered Direct-Access File System.” 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), 2020, pp. 137–154.

M. Alwadi, V. R. Kommareddy, C. Hughes, S. D. Hammond and A. Awad, "Stealth-Persist: Architectural Support for Persistent Applications in Hybrid Memory Systems," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 139-152, doi: 10.1109/HPCA51647.2021.00022.

Cache coloring for high performance

- Page coloring (cache coloring)
  - Different colors to physical pages
    - Same color → same cache set

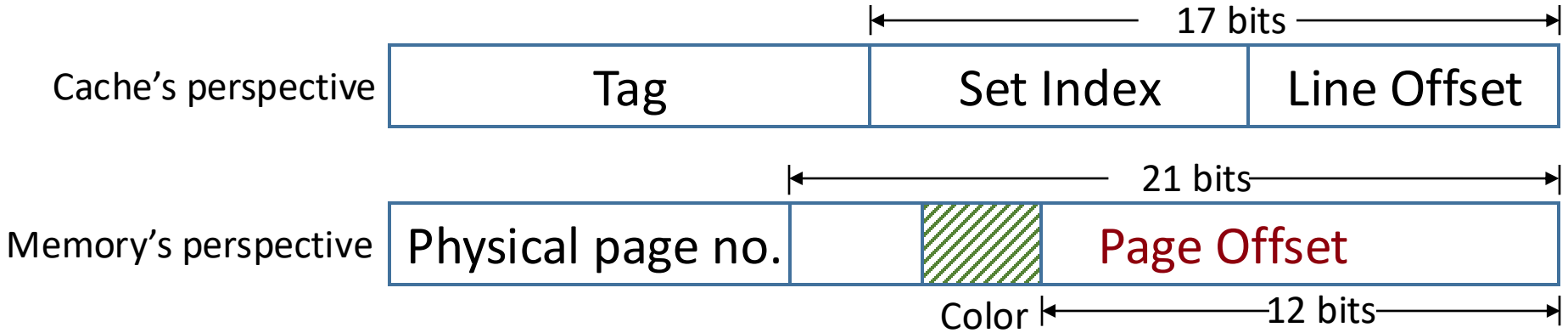


- How to use page coloring to put data into different sets

Kefei Wang, Jian Liu, and Feng Chen. 2020. Put an elephant into a fridge: optimizing cache efficiency for in-memory key-value stores. Proc. VLDB Endow. 13, 9 (May 2020), 1540–1554. DOI:https://doi.org/10.14778/3397230.3397247

Cache coloring for high performance

- Virtual page no → physical page no
  - In the charge of OS
- In-page offset



- Huge Page
  - 2MB, 1GB, …

With huge page, programmers customize in-page layout
➔data in a page not contending one set, less conflict

Kefei Wang, Jian Liu, and Feng Chen. 2020. Put an elephant into a fridge: optimizing cache efficiency for in-memory key-value stores. Proc. VLDB Endow. 13, 9 (May 2020), 1540–1554. DOI:https://doi.org/10.14778/3397230.3397247

Cache is changing

- ADR → eADR
  - A super-capacitor installed to flush all cache lines to memory (if pesistent) in case of power fail

Taiyu Zhou, Yajuan Du, Fan Yang, Xiaojian Liao, and Youyou Lu. 2023. Efficient Atomic Durability on eADR-Enabled Persistent Memory. In Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT '22). Association for Computing Machinery, New York, NY, USA, 124–134. https://doi.org/10.1145/3559009.3569676

Chongnan Ye, Meng Chen, Qisheng Jiang, and Chundong Wang. 2023. Hercules: Enabling Atomic Durability for Persistent Memory with Transient Persistence Domain. ACM Trans. Embed. Comput. Syst. Just Accepted (July 2023). https://doi.org/10.1145/3607473

Summary

- Memory hierarchy is important and interesting
- Developing/optimizing cache management for high performance
- Utilizing cache management for high performance