To Keep or Not to Keep – The Volatility of Replacement Policy Metadata in Hybrid Caches

2nd Workshop on Disruptive Memory Systems (DIMES ’24), Austin, TX, USA

Nils Wilbert, Stefan Wildermann, Jürgen Teich

Friedrich-Alexander-Universität Erlangen-Nürnberg, Hardware-Software-Co-Design

November 03, 2024

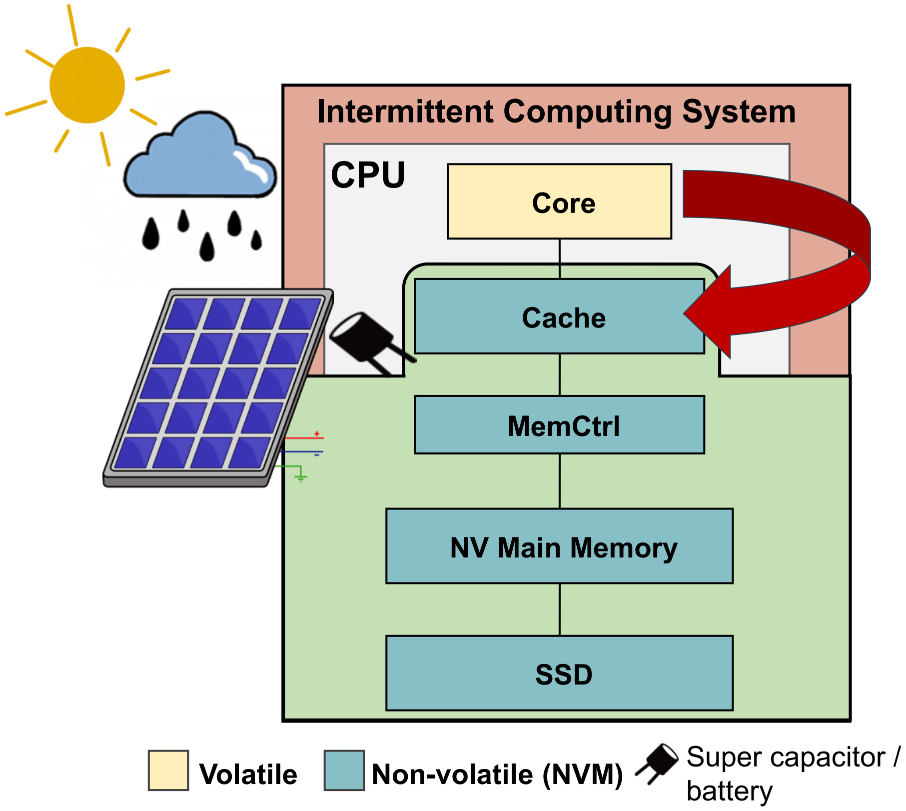
Introduction

Intermittent Computing

- IoT devices, wearables etc. powered by energy harvesting modules, e.g., solar panels

→ Unstable power supply

- System must not lose state and data due to a power shortage
- NVM technologies promise great potential for intermittently powered embedded systems



Introduction

Intermittent Computing

- IoT devices, wearables etc. powered by energy harvesting modules, e.g., solar panels

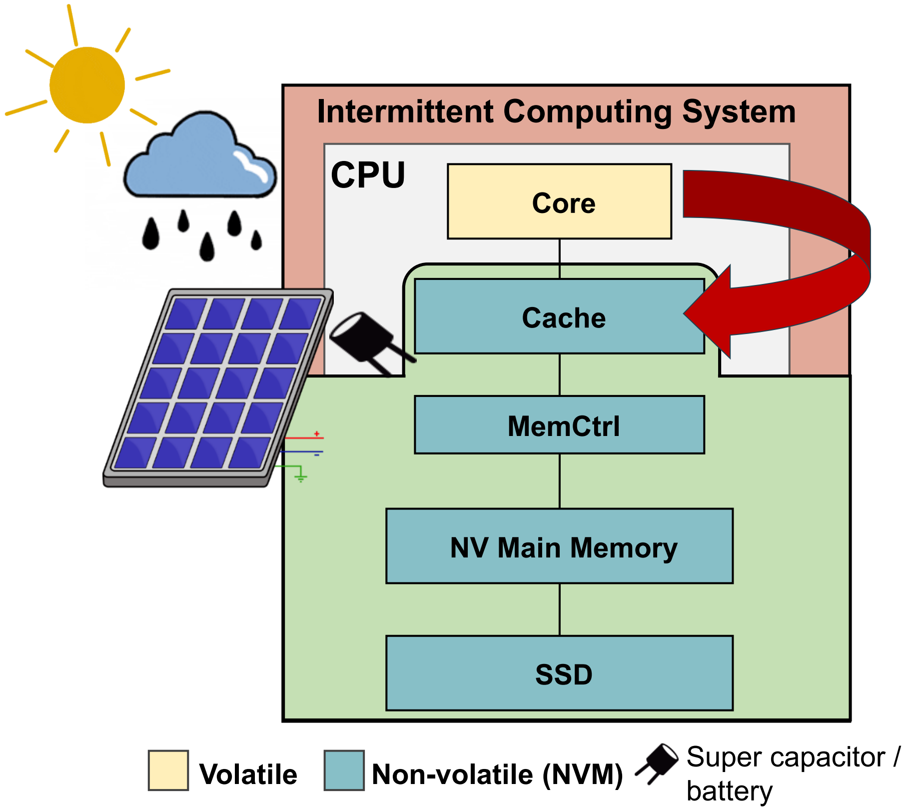
→ Unstable power supply

- System must not lose state and data due to a power shortage
- NVM technologies promise great potential for intermittently powered embedded systems

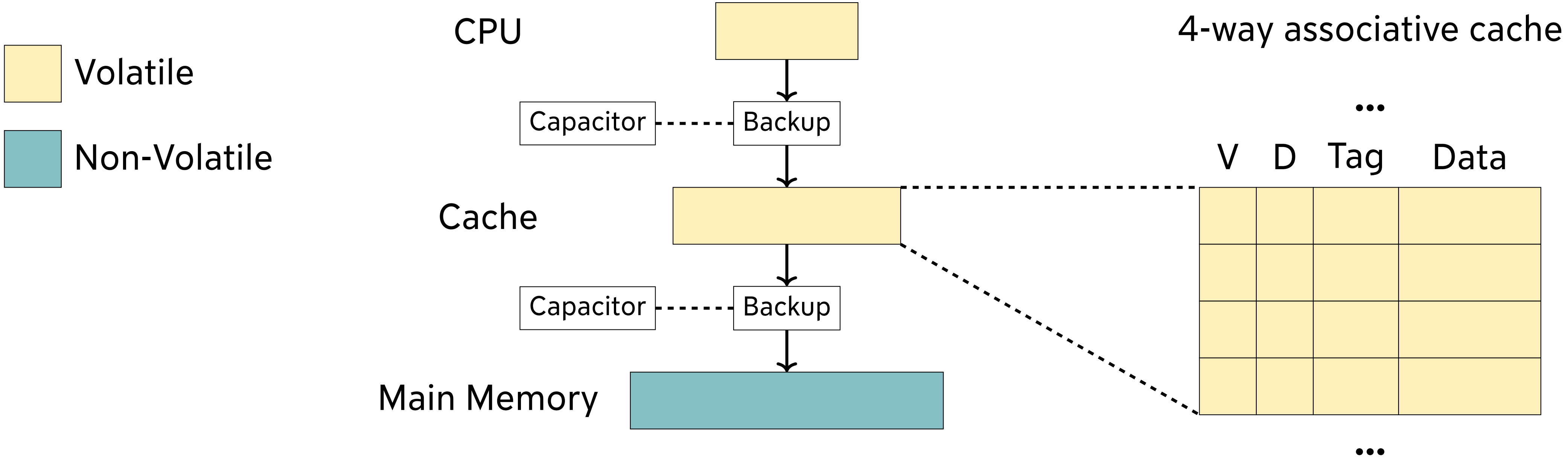
Instruction Level Persistence:

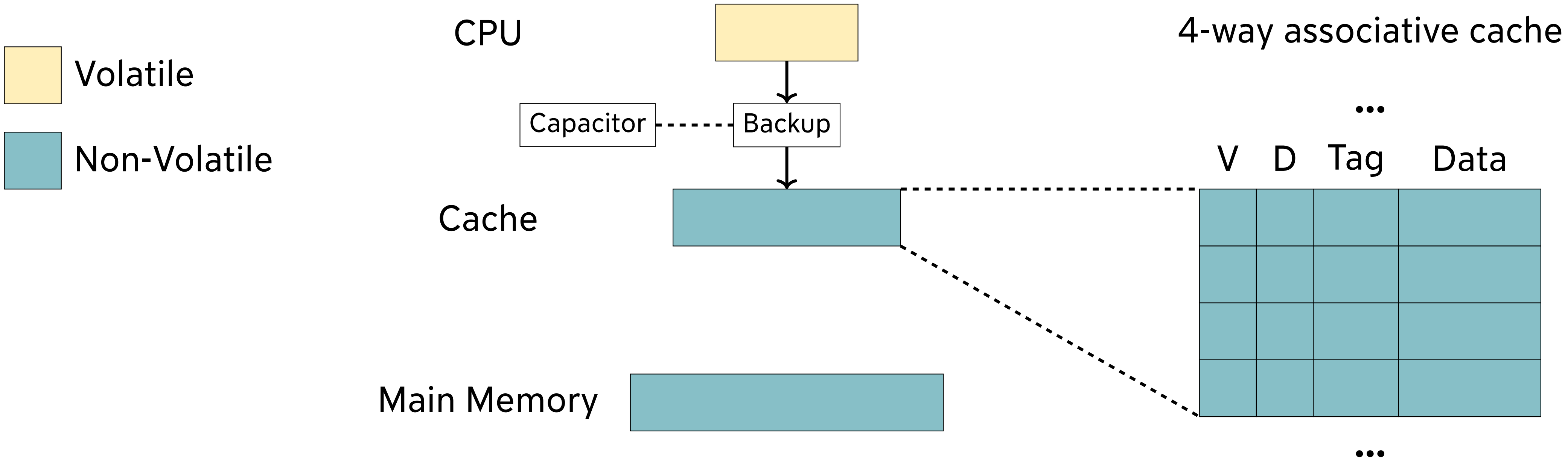
- Supply voltage below threshold → power outage detected
- Run currently issued instructions to completion
- Write back all volatile modified data to a persistent memory

→ Continue at instruction where execution was halted, once power is restored

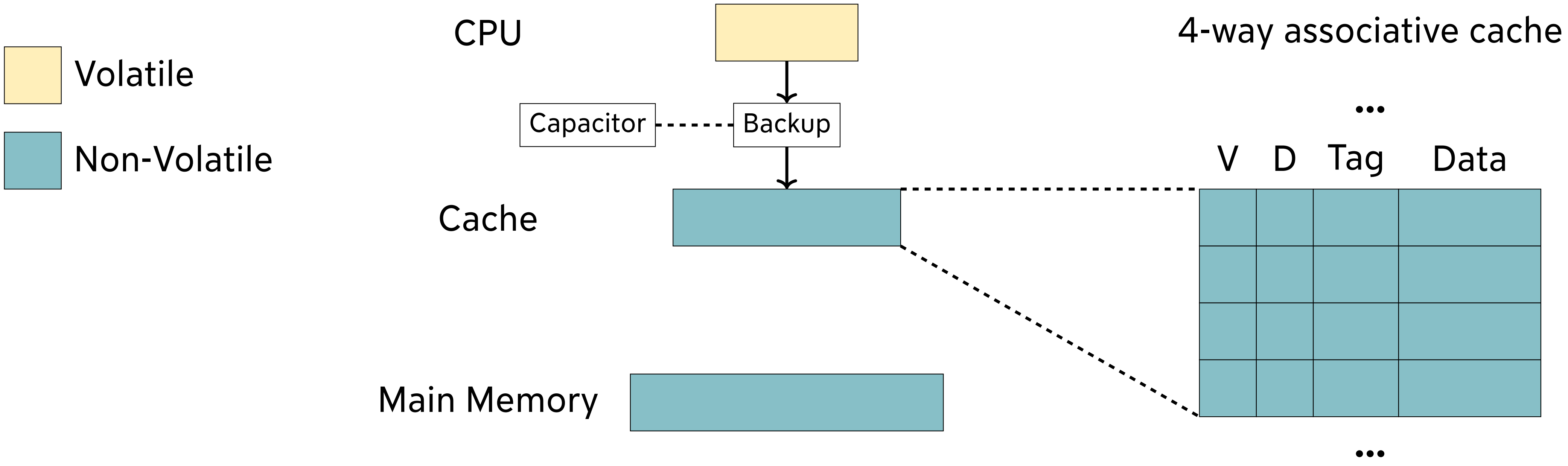


Introduction
Hybrid Caches





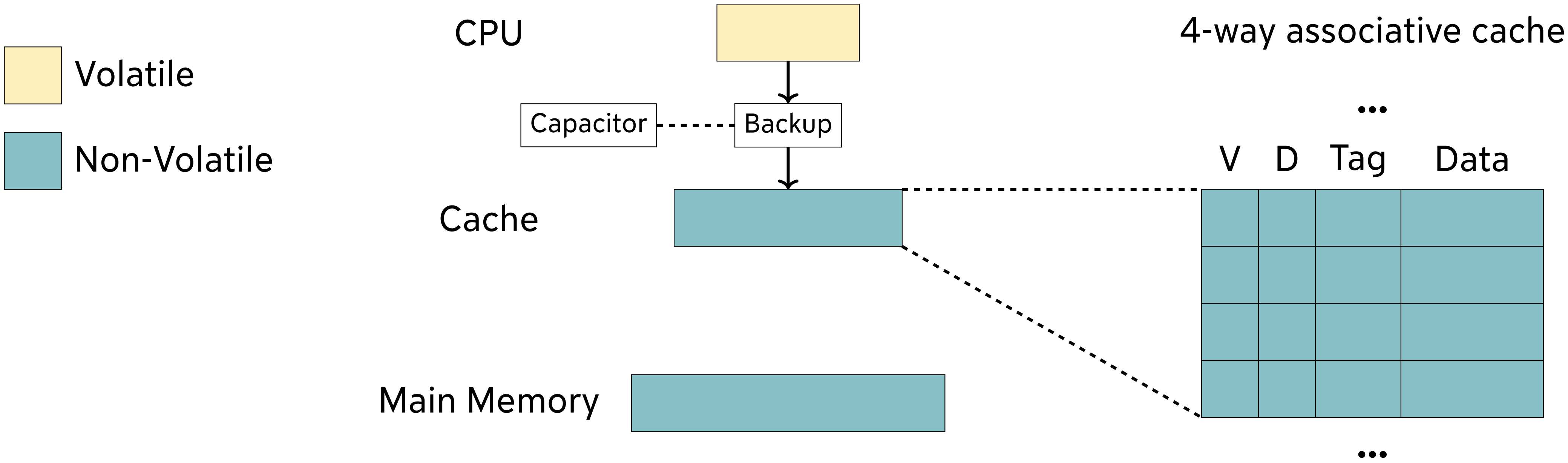
Volatile
Non-Volatile
CPU
Capacitor
Backup
Cache
Main Memory
4-way associative cache
...
V
D
Tag
Data
...



STT-RAM caches compared to conventional SRAM caches:

Introduction

Hybrid Caches

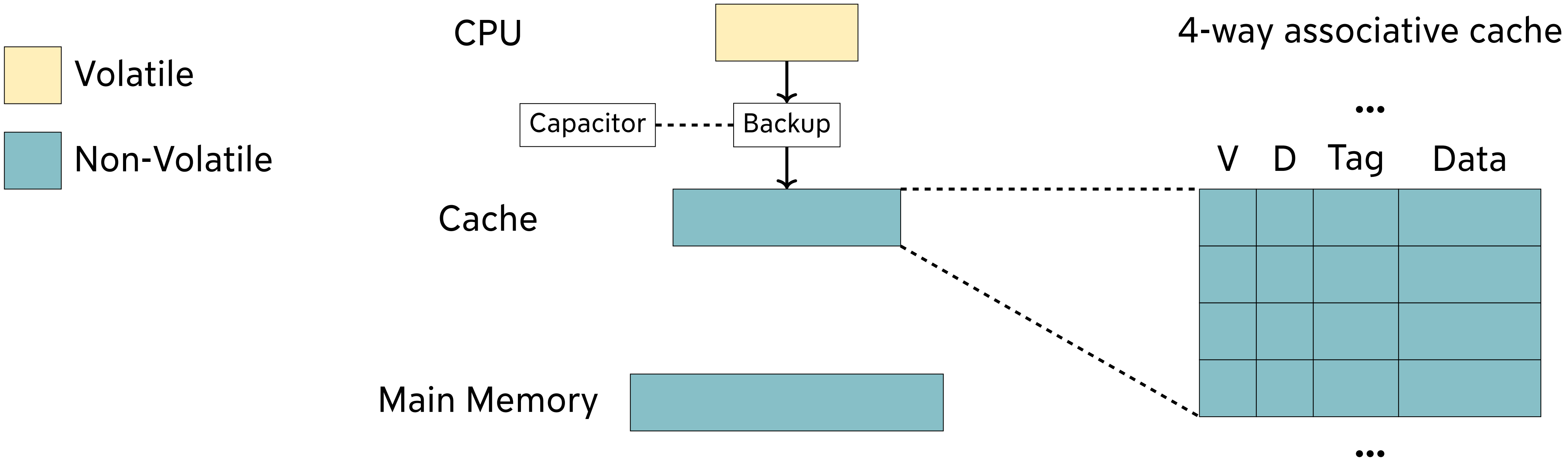


STT-RAM caches compared to conventional SRAM caches:

- High write latency
- High write energy
- Limited endurance

Introduction

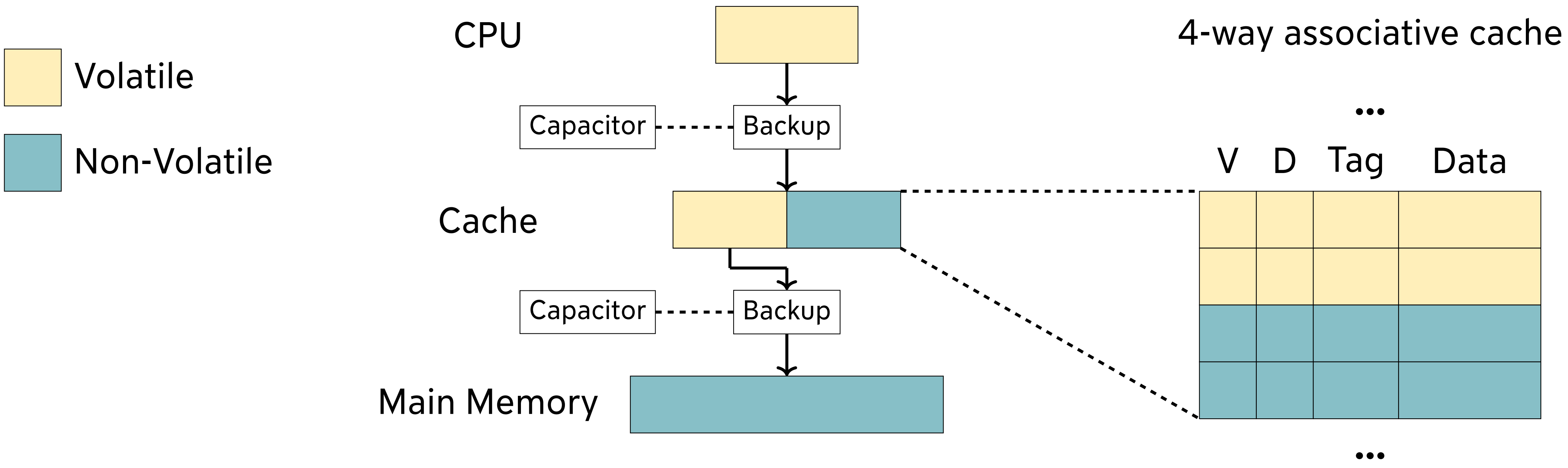
Hybrid Caches



STT-RAM caches compared to conventional SRAM caches:

- – High write latency
- – High write energy
- – Limited endurance

- + Non-volatility
- + Low read energy
- + High density
- + Low static power



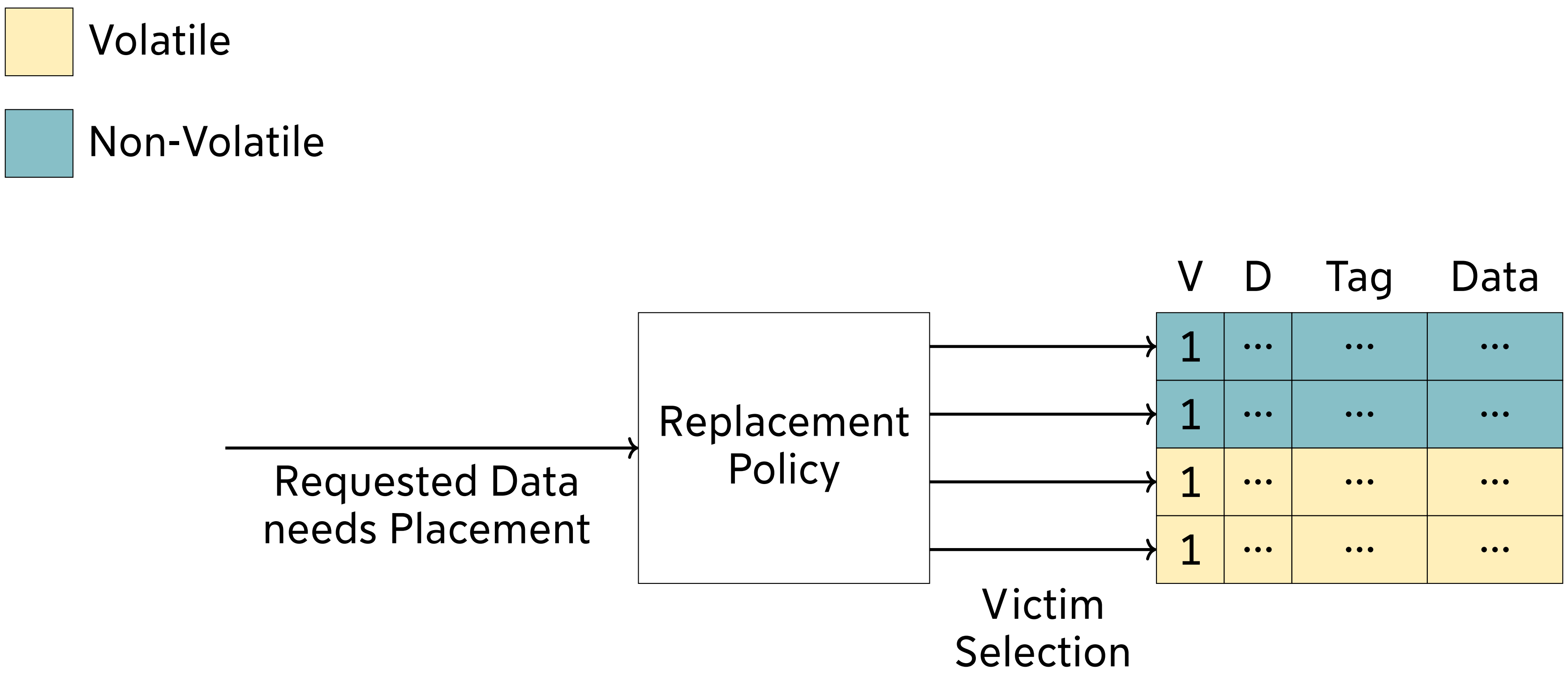
STT-RAM caches compared to conventional SRAM caches:

- – High write latency
- – High write energy
- – Limited endurance

- + Non-volatility
- + Low read energy
- + High density
- + Low static power

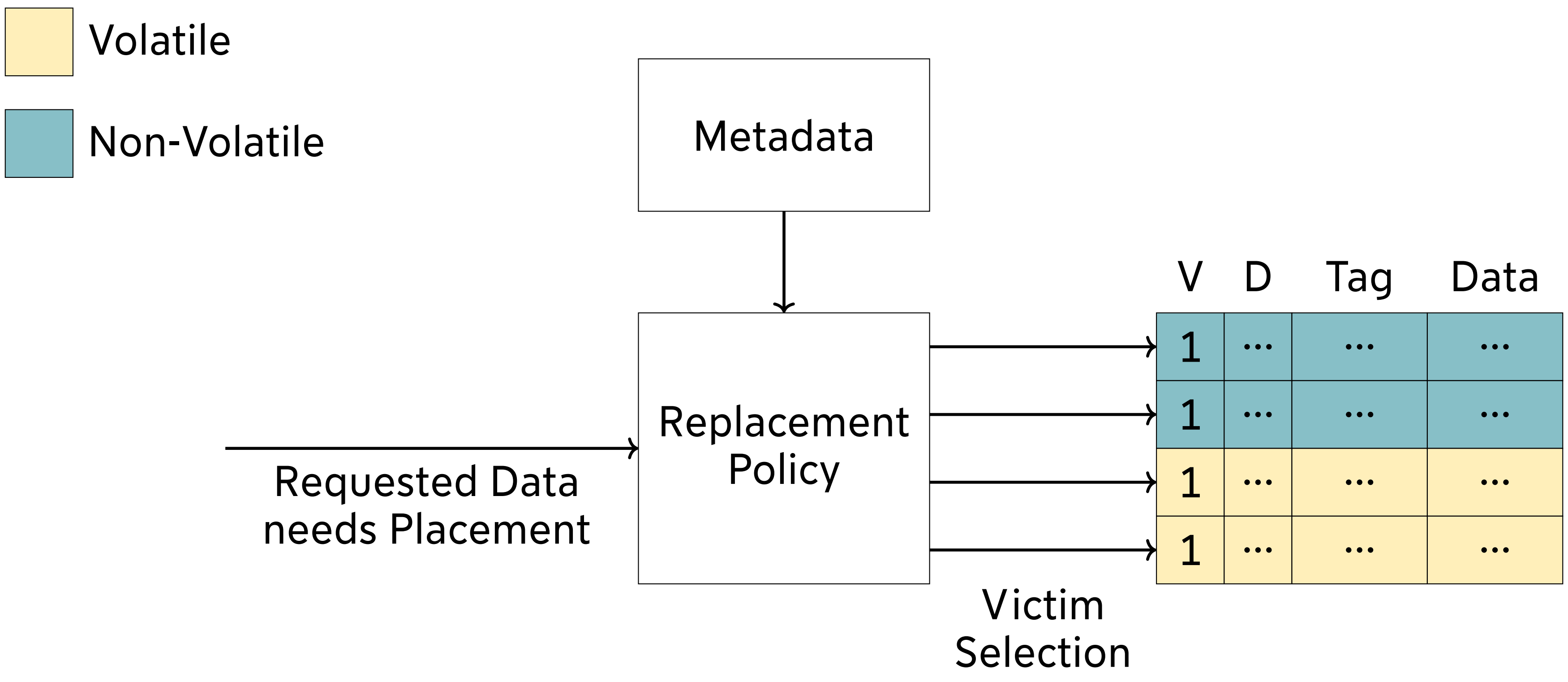
Introduction

Problem Definition



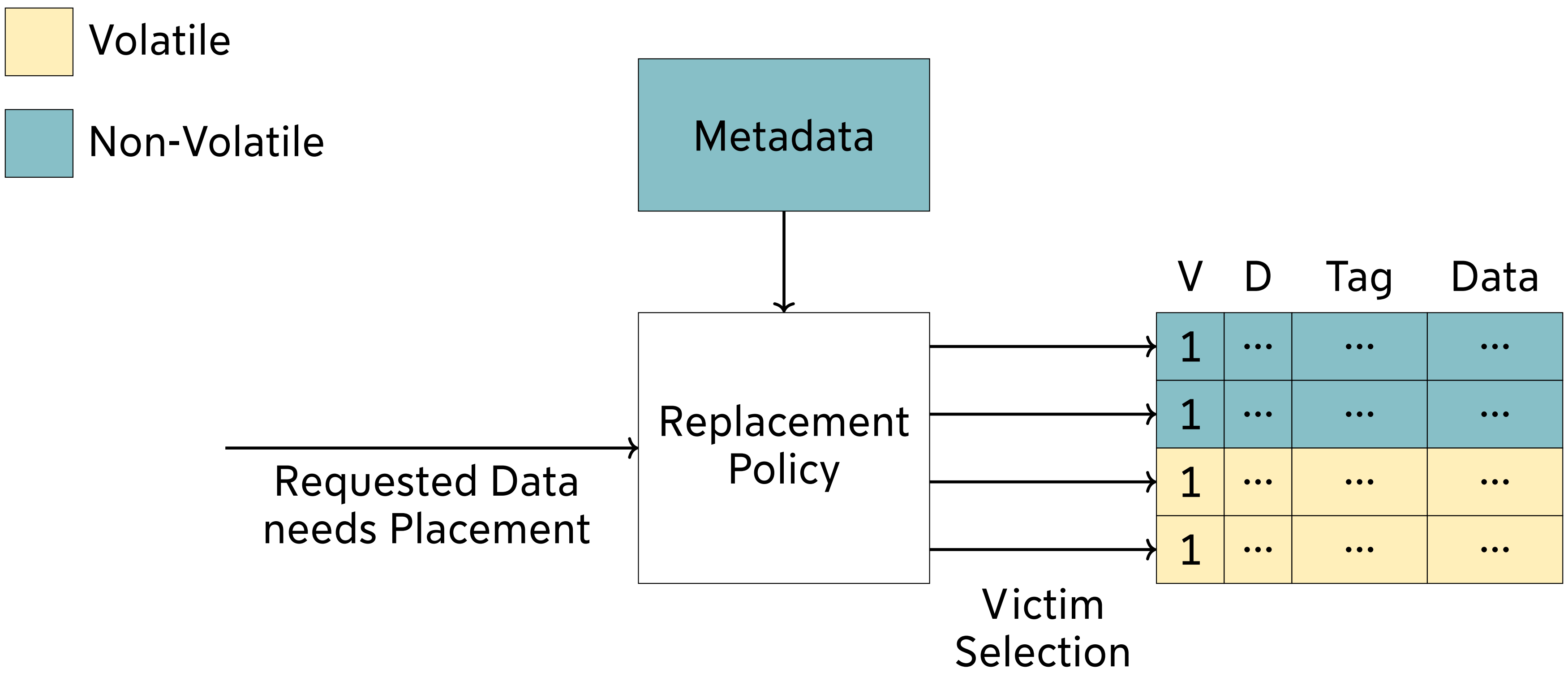
Introduction

Problem Definition



Introduction

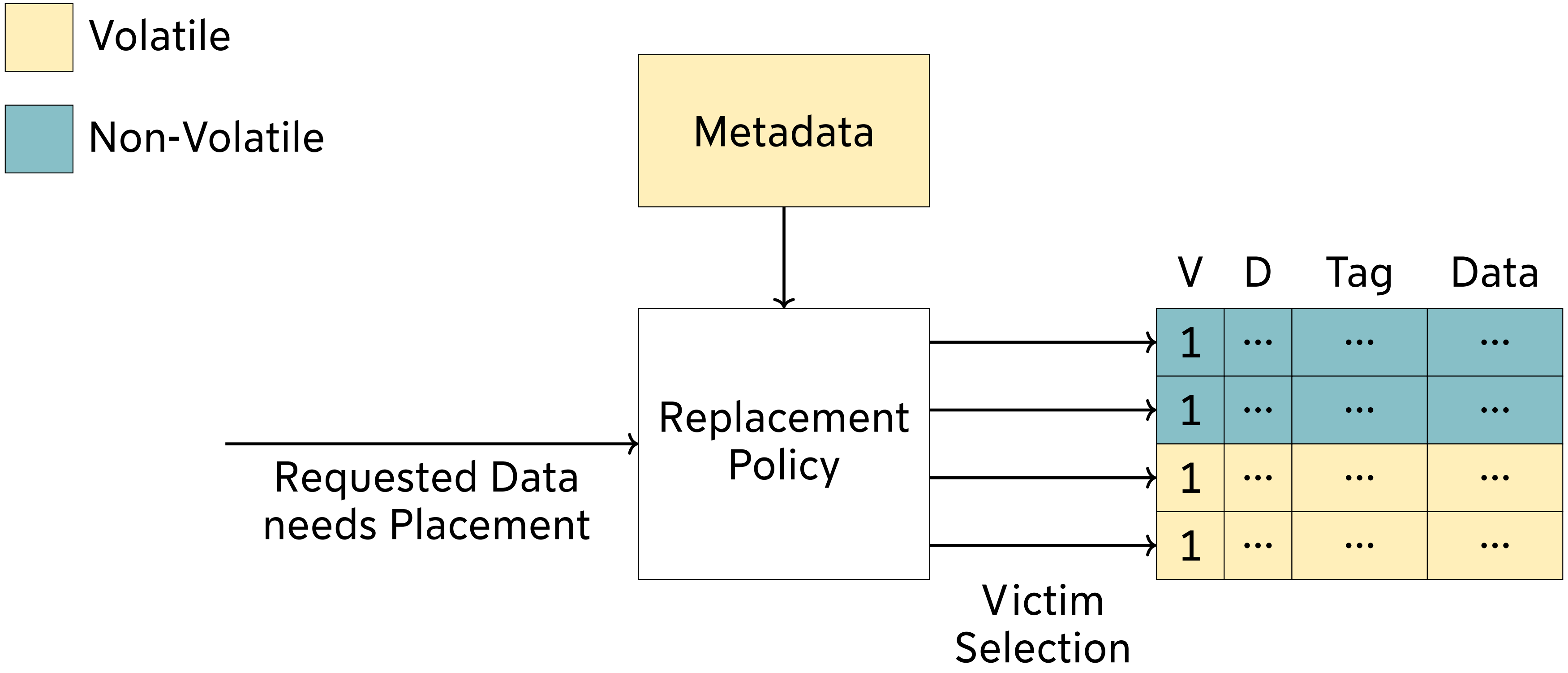
Problem Definition



+ Keep previously acquired knowledge on, e.g., access patterns after a power outage
– Potential NVM endurance issues

Introduction

Problem Definition



– Lose previously acquired knowledge on, e.g., access patterns after a power outage
+ Metadata reset following power outages as an opportunity to balance out accumulated mispredictions

Research Questions

- Is it worth considering this niche in the design space of hybrid caches?
- If so, is there a general rule on how replacement policy metadata should behave following a power outage?
- What does this imply for future and related research on hybrid caches for intermittent computing?

Round-Robin Replacement

	V	D	Tag	Data
	1	1	…	…
Next Victim →	0	0	…	…
	1	0	…	…
	0	0	…	…

- Pointer at next cache line (i.e., the next way in the cache set) to be replaced

Round-Robin Replacement

	V	D	Tag	Data
Next Victim ⟶	1	1	…	…
	1	0	…	…
	1	0	…	…
	0	0	…	…

- Pointer at next cache line (i.e., the next way in the cache set) to be replaced
- Increment pointer after a cache line has been placed

Round-Robin Replacement

	V	D	Tag	Data
	1	0	...	...
	1	0	...	...
	1	0	...	...
Next Victim →	0	0	...	...

- Pointer at next cache line (i.e., the next way in the cache set) to be replaced
- Increment pointer after a cache line has been placed
- Wrap around pointer after reaching highest way index

Round-Robin Replacement

	V	D	Tag	Data
	1	0	...	...
	1	0	...	...
	1	0	...	...
Next Victim ⟶	0	0	...	...

- Pointer at next cache line (i.e., the next way in the cache set) to be replaced
- Increment pointer after a cache line has been placed
- Wrap around pointer after reaching highest way index

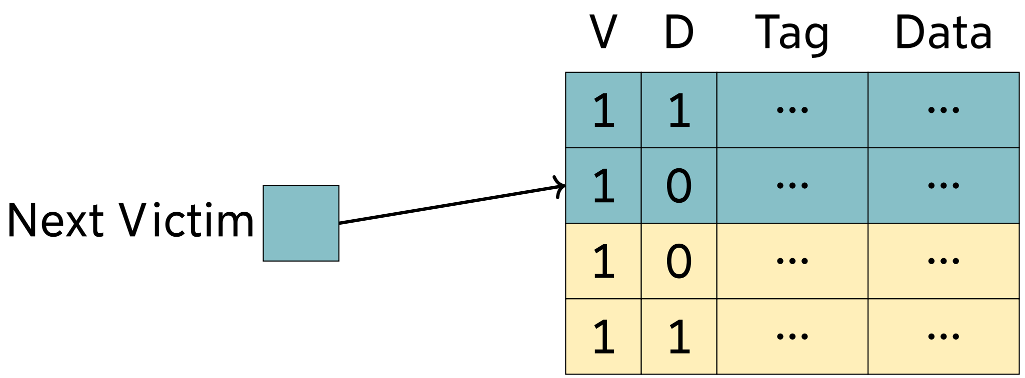
→ Supported by most ARM processors

Round-Robin Pointer Volatility

Non-Volatile Pointer

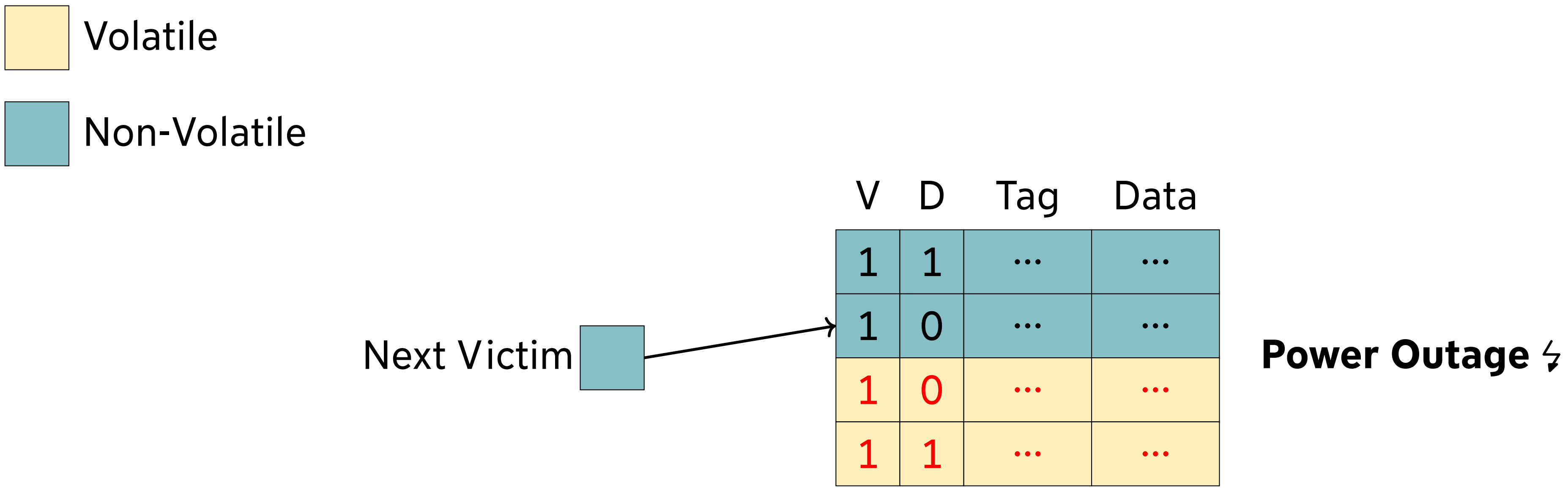
Volatile

Non-Volatile



Round-Robin Pointer Volatility

Non-Volatile Pointer

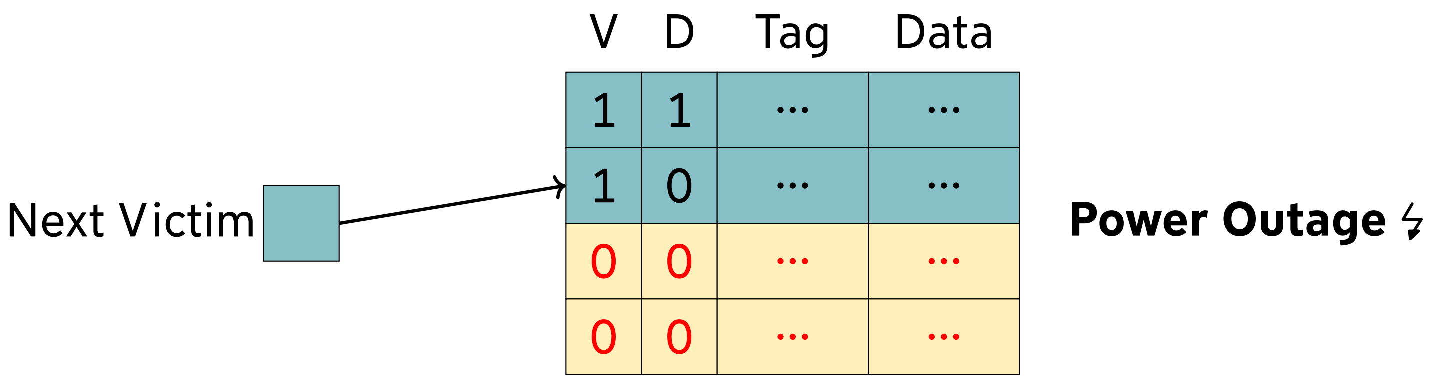


Round-Robin Pointer Volatility

Non-Volatile Pointer

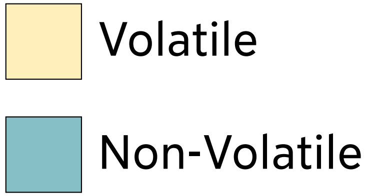
Volatile

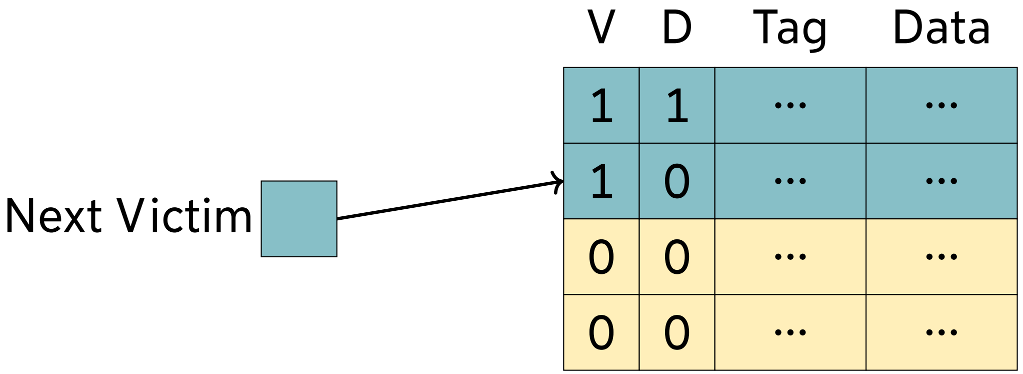
Non-Volatile



Round-Robin Pointer Volatility

Non-Volatile Pointer





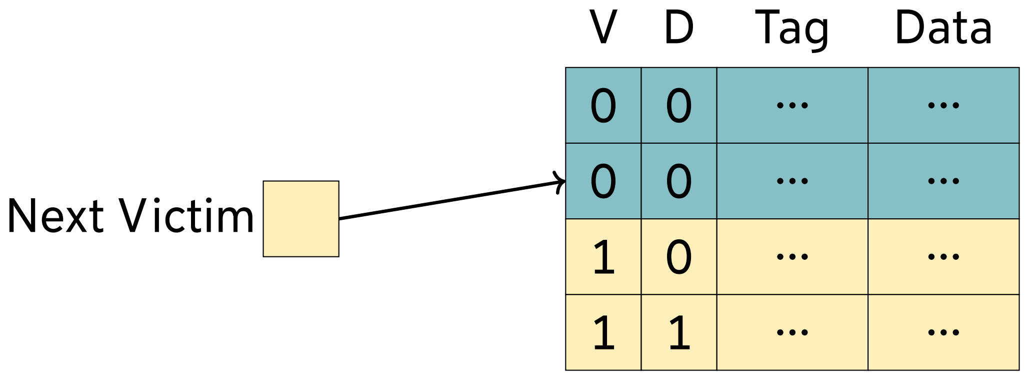
→ Invalid cache lines available, yet valid cache lines are chosen as the next victim

Round-Robin Pointer Volatility

Volatile Pointer

Volatile

Non-Volatile

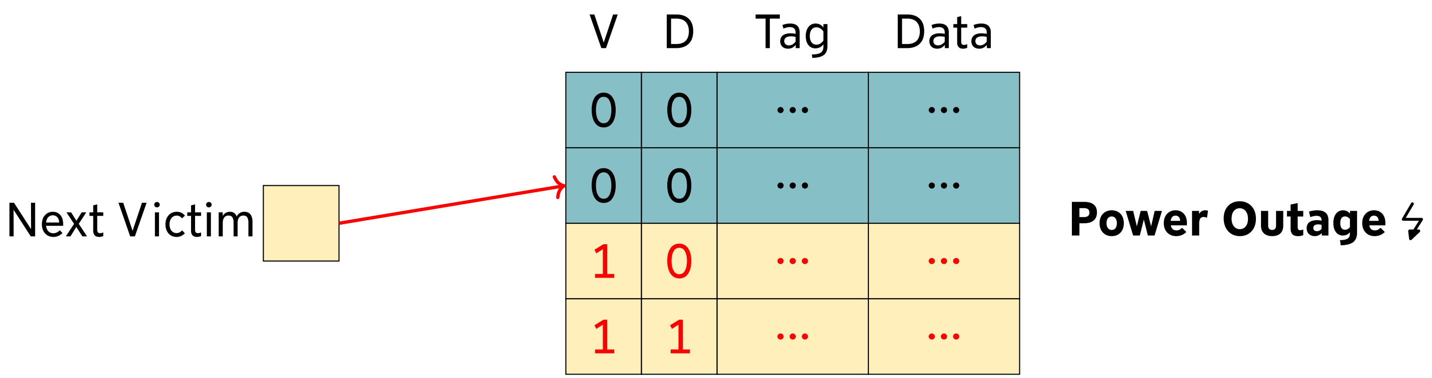


Round-Robin Pointer Volatility

Volatile Pointer

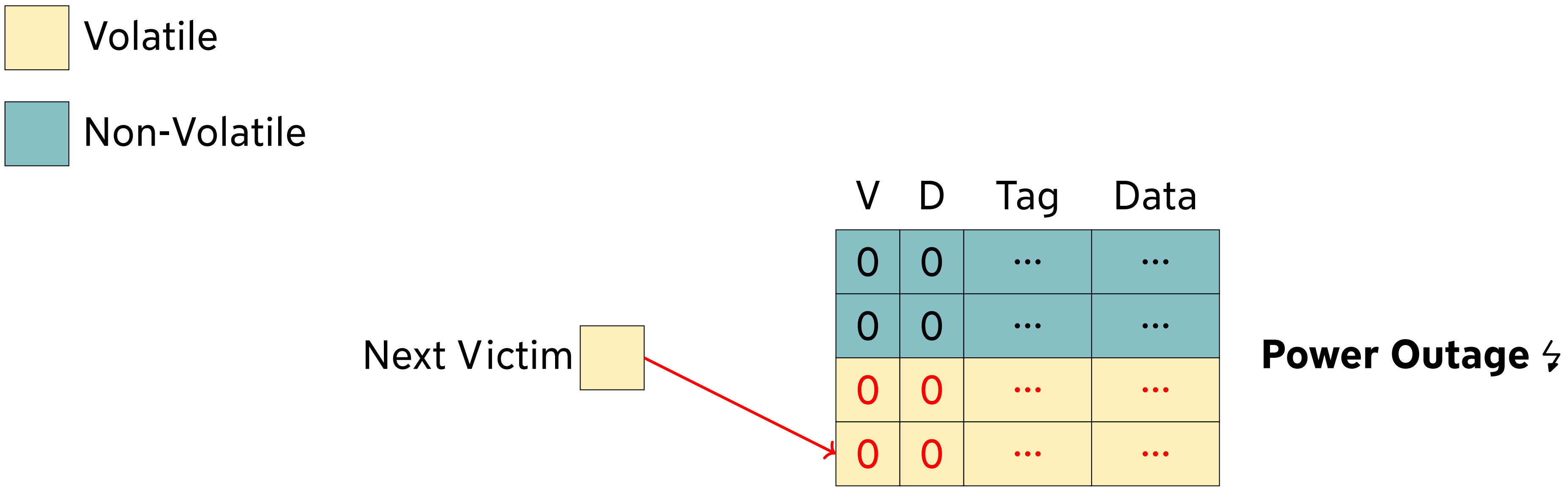
Volatile

Non-Volatile



Round-Robin Pointer Volatility

Volatile Pointer

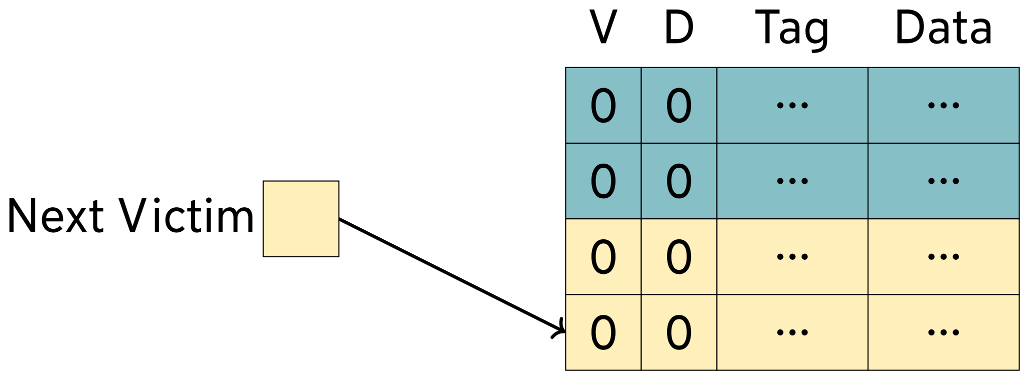


Round-Robin Pointer Volatility

Volatile Pointer

Volatile

Non-Volatile



→ For small working sets and/or frequent power outages, non-volatile cache lines are not exploited

Experimental Setup

Architecture:

- Generic pipelined single-core out-of-order ARM CPU
- CPU clock of 240 MHz, system clock of 480 MHz
- 4-way associative 32 KB large SRAM/STT-RAM **hybrid data cache** (single-level)
- Cache parameters obtained using NVSim [Don+12]
- PCRAM main memory modeled after [Cho+12]

Experimental Setup

Architecture:

- Generic pipelined single-core out-of-order ARM CPU
- CPU clock of 240 MHz, system clock of 480 MHz
- 4-way associative 32 KB large SRAM/STT-RAM **hybrid data cache** (single-level)
- Cache parameters obtained using NVSim [Don+12]
- PCRAM main memory modeled after [Cho+12]

Memory characteristics:

	Read Latency	Write Latency	Read Energy (per access)	Write Energy (per access)
SRAM Cache	2 Cycles @240 MHz	2 Cycles @240 MHz	0.009 nJ	0.009 nJ
STT-RAM Cache	2 Cycles @240 MHz	8 Cycles @240 MHz	0.007 nJ	0.056 nJ
PCRAM Main Memory	48 Cycles @400 MHz (tRCD)		0.081 nJ	1.685 nJ

Experimental Setup

Applications:

- Merge Sort *(write-intensive)*: Sort an input array containing 65,536 integers
- Image Processing *(read-intensive)*: 2D convolution on a 640×640 large image using a 3×3 large kernel

Experimental Setup

Applications:

- Merge Sort *(write-intensive)*: Sort an input array containing 65,536 integers
- Image Processing *(read-intensive)*: 2D convolution on a 640×640 large image using a 3×3 large kernel

Simulation Parameters:

- gem5 simulator [Bin+11] coupled with NVMain 2.0 [Por+15] to simulate non-volatile main memories
- A power outage is triggered every 2,500,000 CPU cycles
- Baseline architecture featuring a random replacement policy

Experimental Setup

Applications:

- Merge Sort *(write-intensive)*: Sort an input array containing 65,536 integers
- Image Processing *(read-intensive)*: 2D convolution on a 640×640 large image using a 3×3 large kernel

Simulation Parameters:

- gem5 simulator [Bin+11] coupled with NVMain 2.0 [Por+15] to simulate non-volatile main memories
- A power outage is triggered every 2,500,000 CPU cycles
- Baseline architecture featuring a random replacement policy

Objectives:

- Latency in clock cycles
- Dynamic energy consumption normalized to baseline architecture

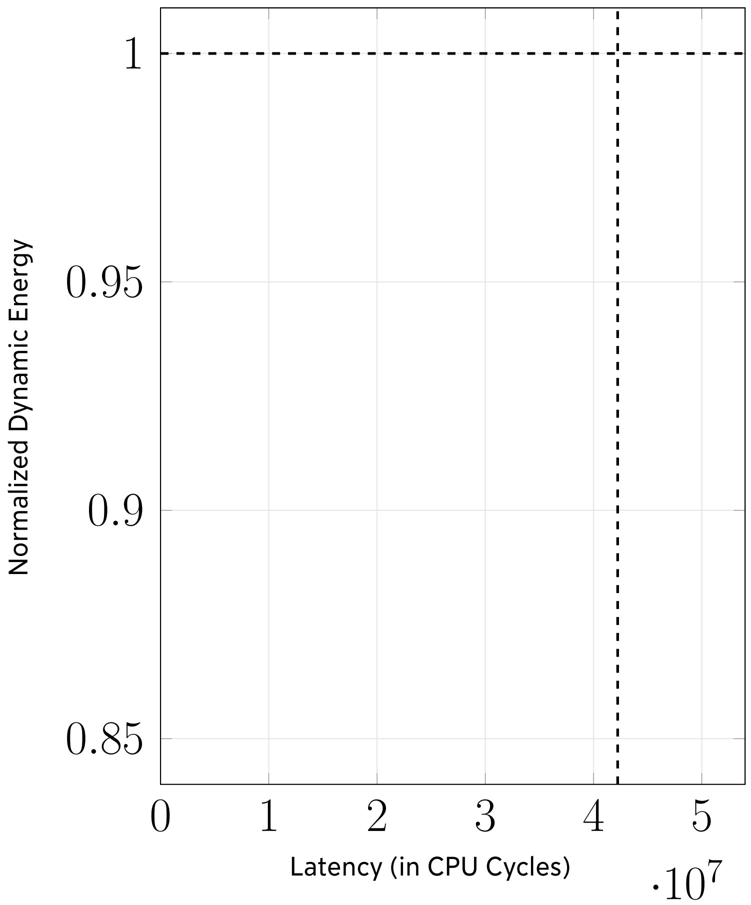
Goals Revisited

Analyze latency and energy trade-offs by comparing:

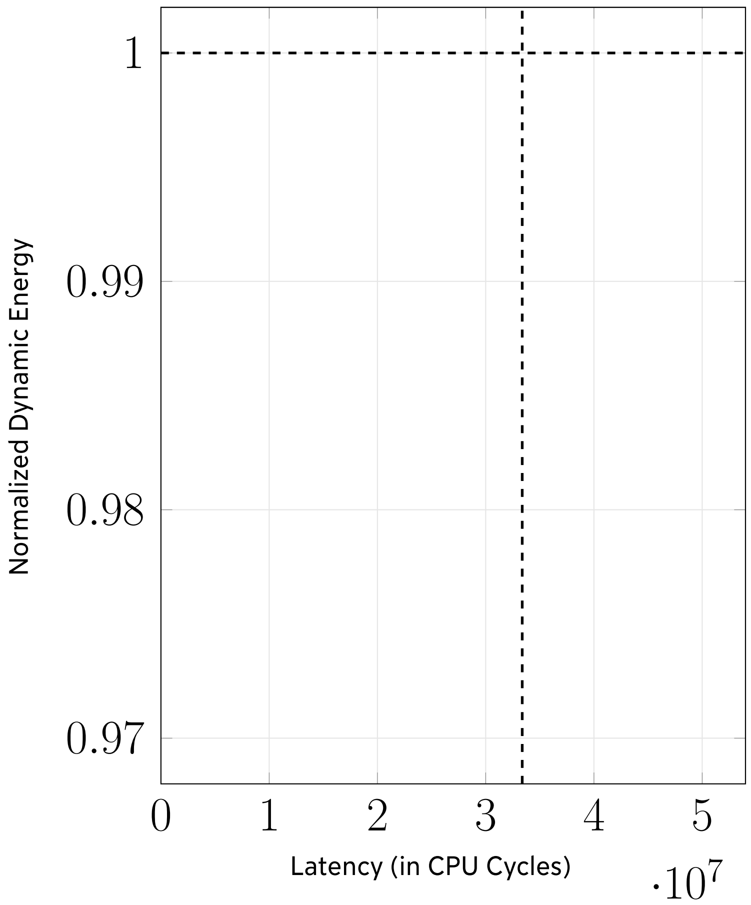
- A round-robin policy with a non-volatile pointer towards the next victim
- A round-robin policy with a volatile pointer that, following power outages, is reset to volatile cache lines

Experimental Results

Round-Robin (RR) Policy



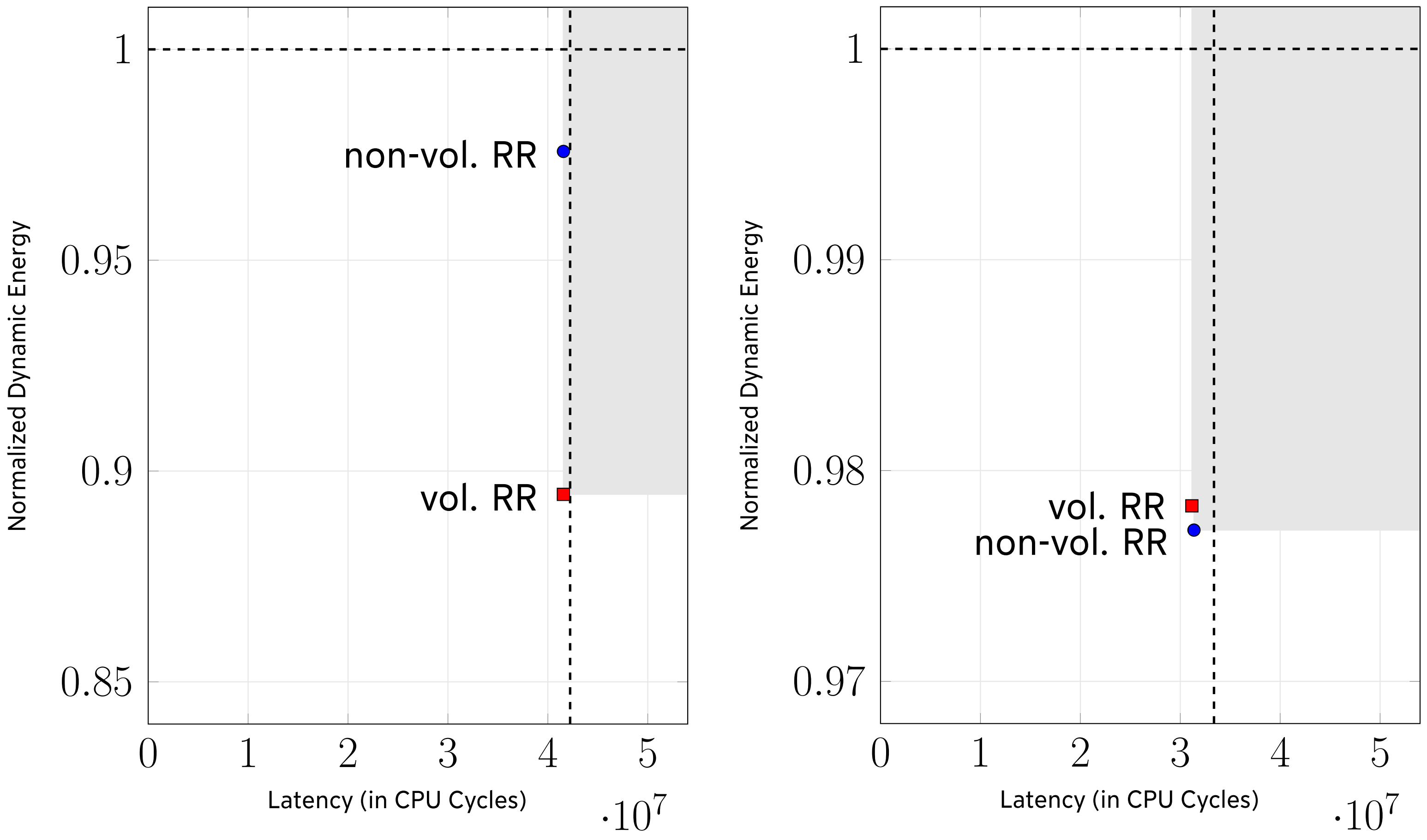
(a) **Merge Sort**



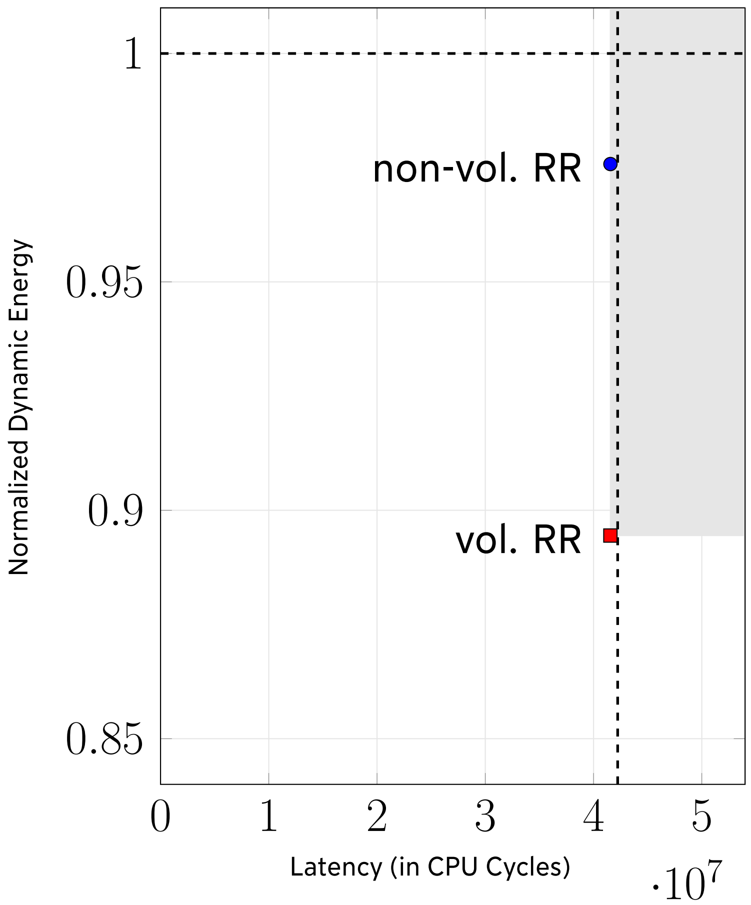
(b) **Image Processing**

Experimental Results

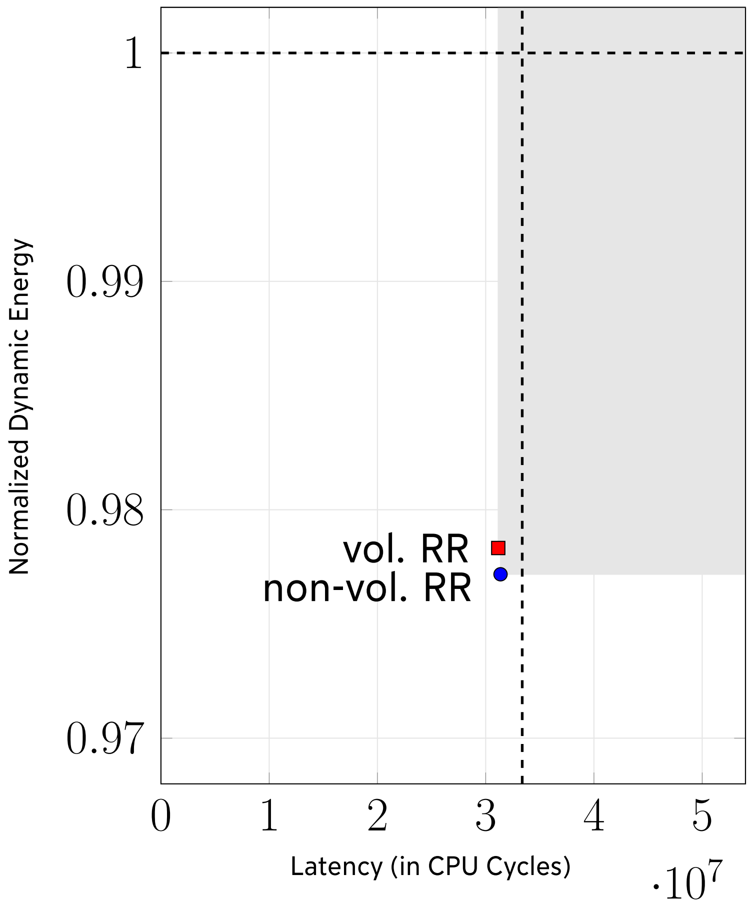
Round-Robin (RR) Policy



(a) **Merge Sort** (b) **Image Processing**



(a) **Merge Sort**

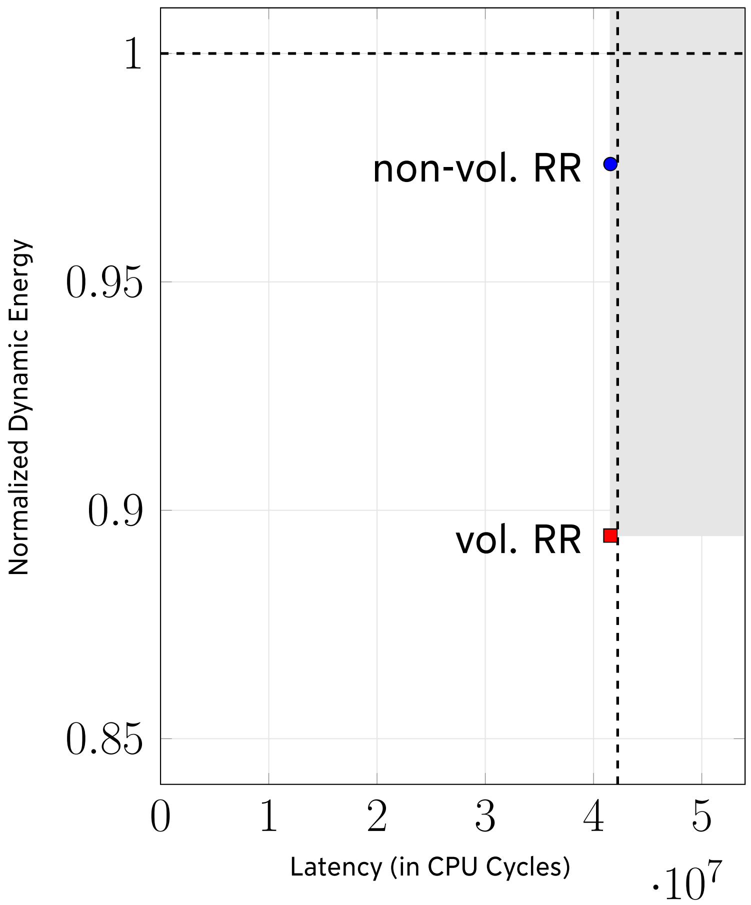


(b) **Image Processing**

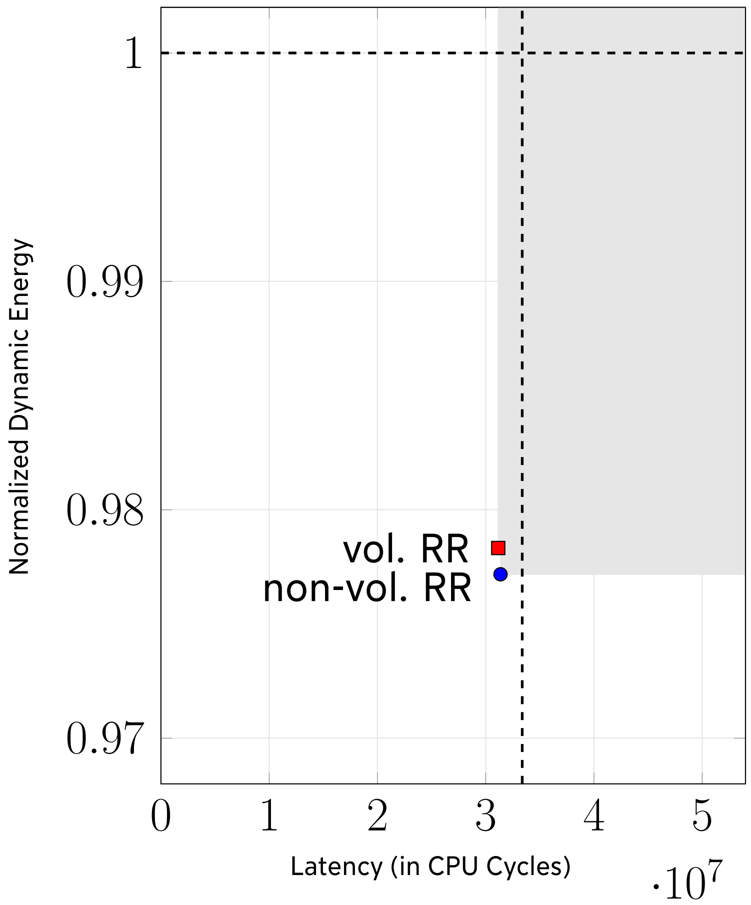
Key takeaways:

- Both volatility options outperform a randomized approach
- The two RR approaches do not dominate each other

Experimental Results
Round-Robin (RR) Policy



(a) **Merge Sort**



(b) **Image Processing**

Key takeaways:

- Both volatility options outperform a randomized approach
- The two RR approaches do not dominate each other
- Volatile RR pointer leads to accesses mainly revolving around the volatile section
- Up to 8.4% difference in dynamic energy consumption depending on the volatility of the RR pointer

Write Intensity Prediction [AYC16]

At each Program Counter (PC), the invoked data accesses can lead to a cache miss, with future cache hitting accesses to this cache line being either...

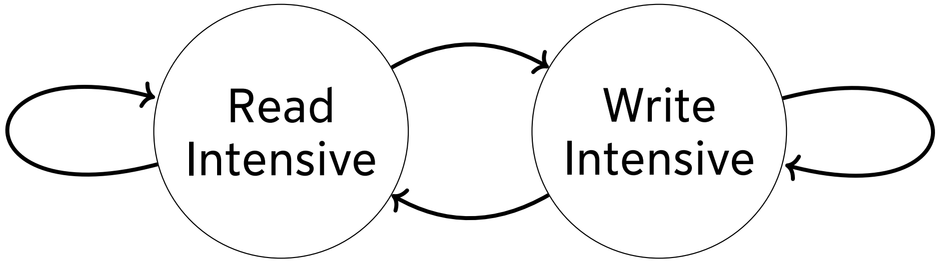
Write Intensity Prediction [AYC16]

At each Program Counter (PC), the invoked data accesses can lead to a cache miss, with future cache hitting accesses to this cache line being either...

- Read Intensive
- Write Intensive

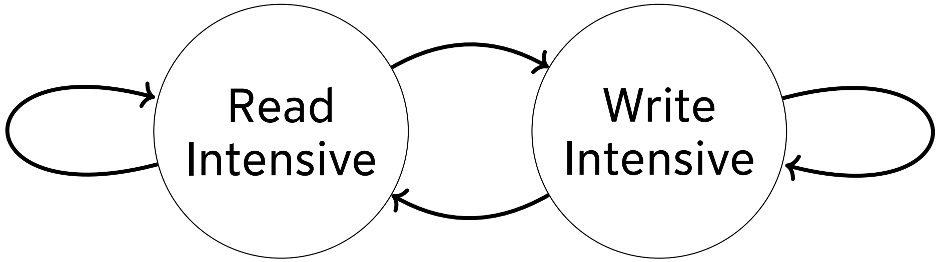
Write Intensity Prediction [AYC16]

At each Program Counter (PC), the invoked data accesses can lead to a cache miss, with future cache hitting accesses to this cache line being either...



Write Intensity Prediction [AYC16]

At each Program Counter (PC), the invoked data accesses can lead to a cache miss, with future cache hitting accesses to this cache line being either...



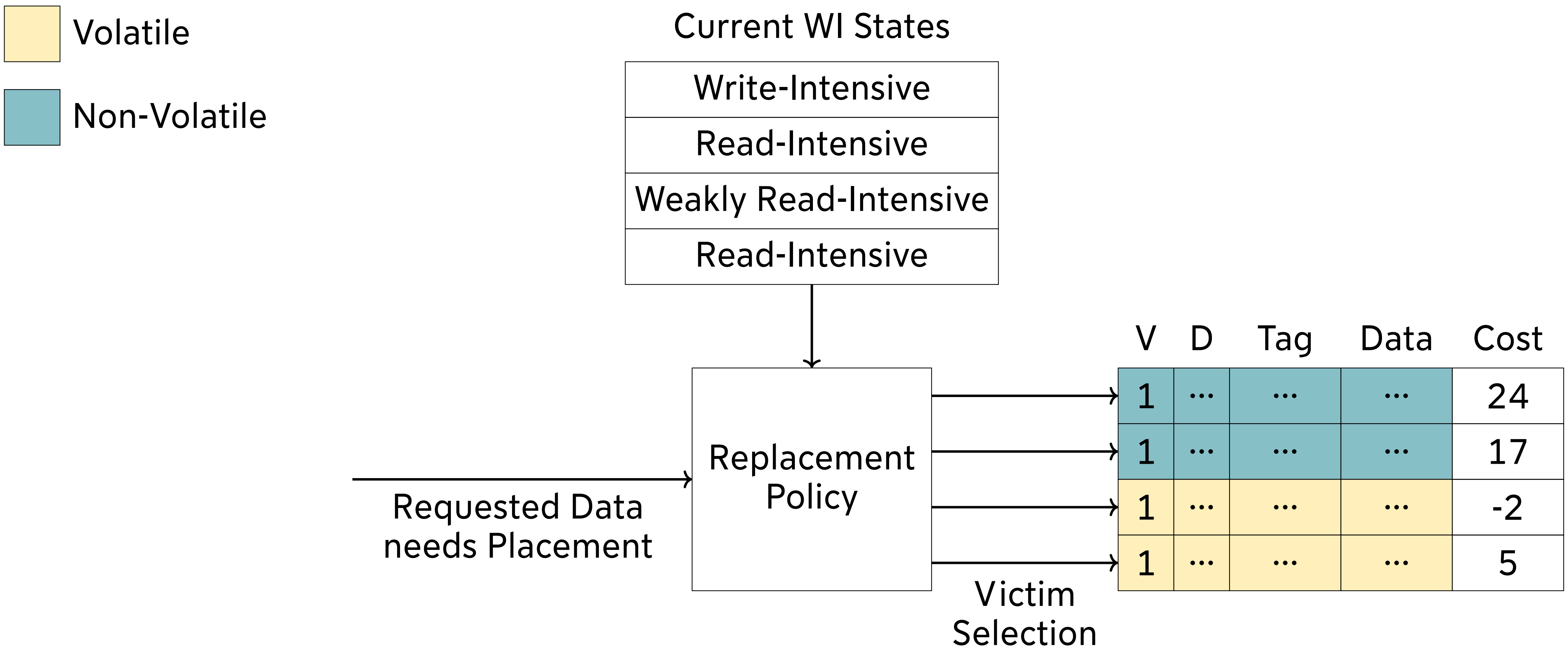
Write Intensity (WI) Policy fundamentals:

- Predict write intensity to suitably place data in either the volatile or non-volatile cache section
- **State table:** Contains current state for all state machines

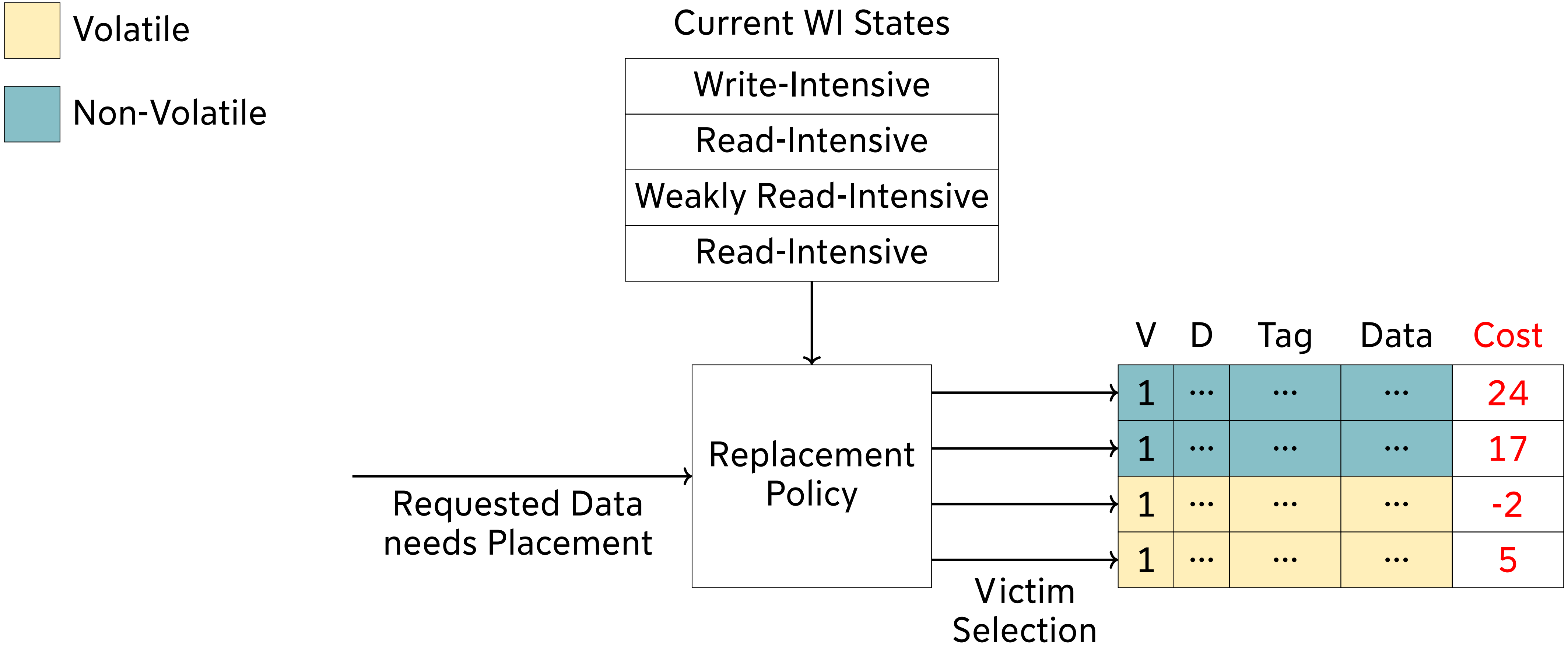
→ Add "Weakly Write-Intensive" and "Weakly Read-Intensive" states

- **Costs:** Track accesses to cache line. Used to update state machine on eviction

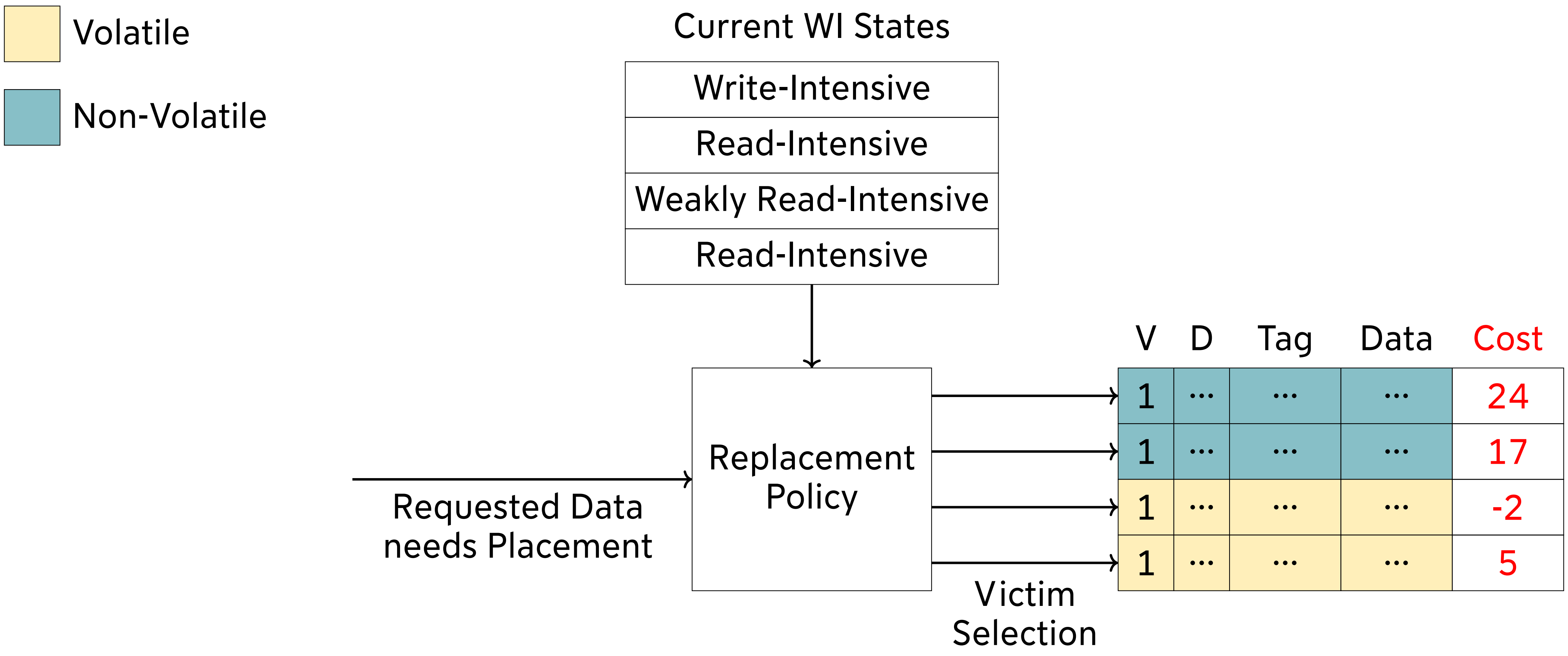
Candidates for Different Technological Implementations



Candidates for Different Technological Implementations

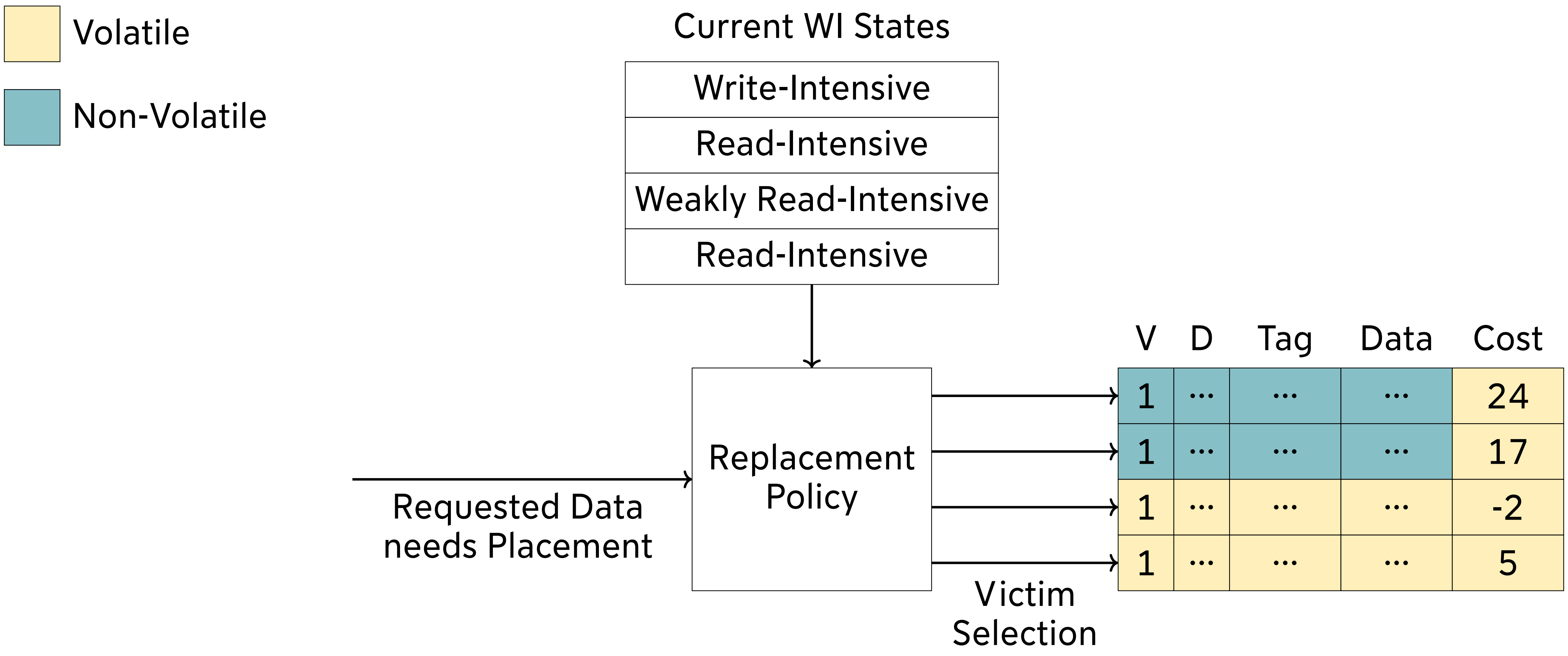


Candidates for Different Technological Implementations



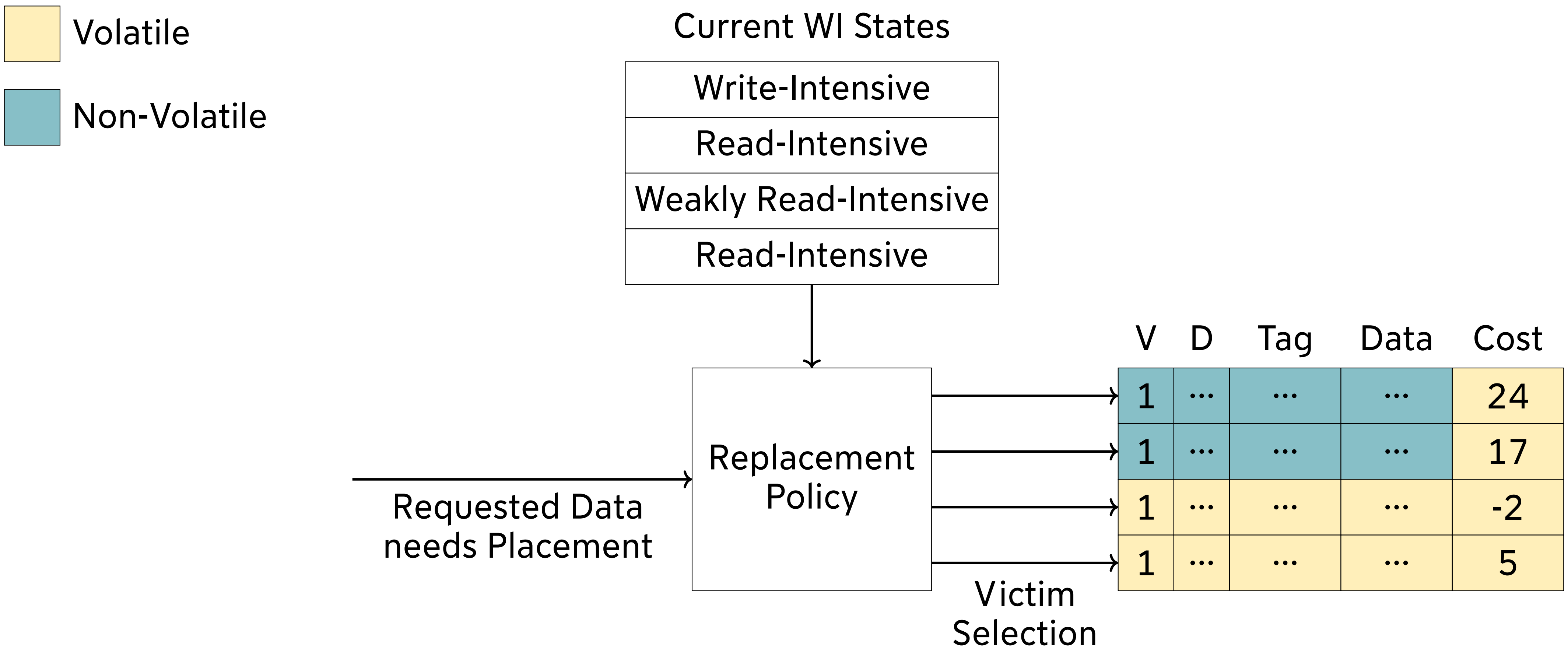
Cost field updated on every access to respective cache line
→ Many writes, thus unsuitable for NVM implementation (endurance issues)

Candidates for Different Technological Implementations

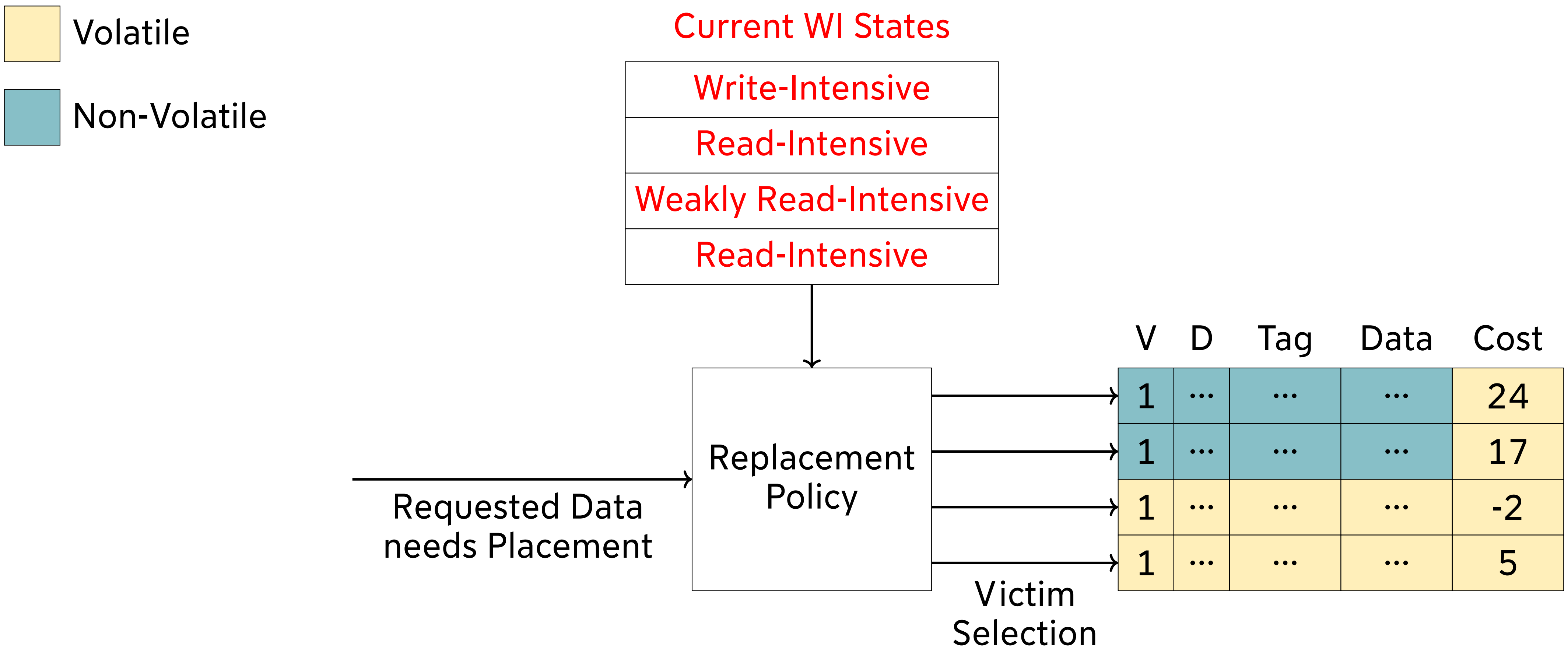


Cost field updated on every access to respective cache line
→ Many writes, thus unsuitable for NVM implementation (endurance issues)

Candidates for Different Technological Implementations



Candidates for Different Technological Implementations

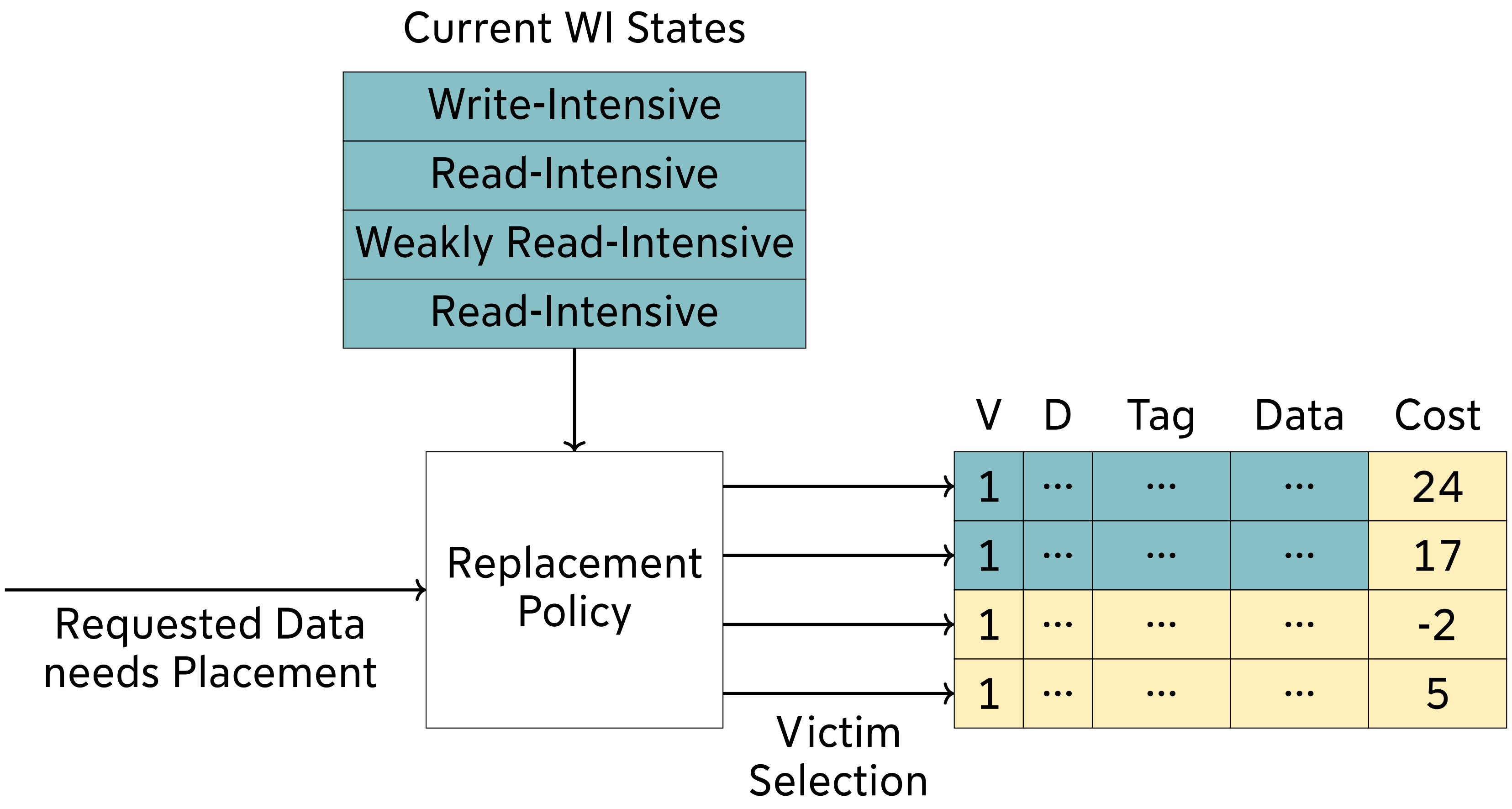


WI states updated on eviction (transition determined by cost function)

Candidates for Different Technological Implementations

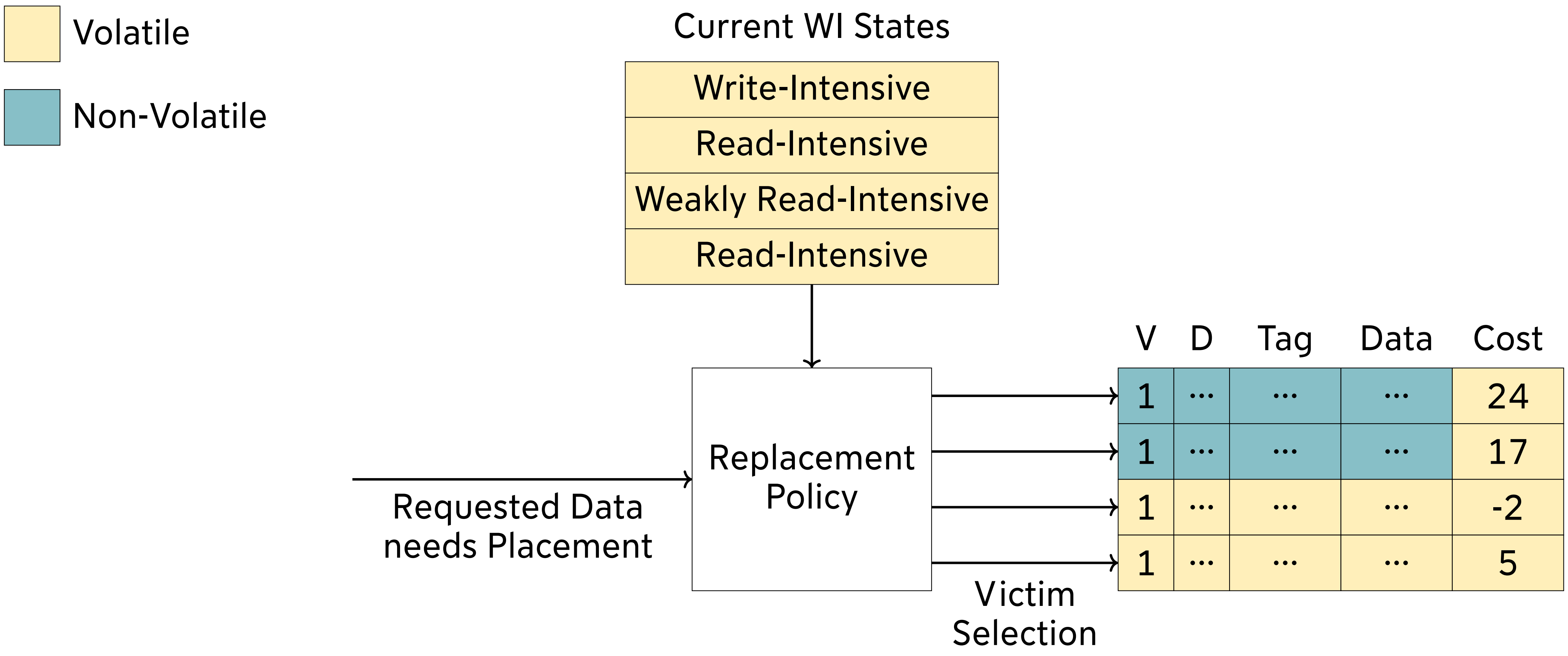
Volatile

Non-Volatile



WI states updated on eviction (transition determined by cost function)
→ NVM implementation possible

Candidates for Different Technological Implementations

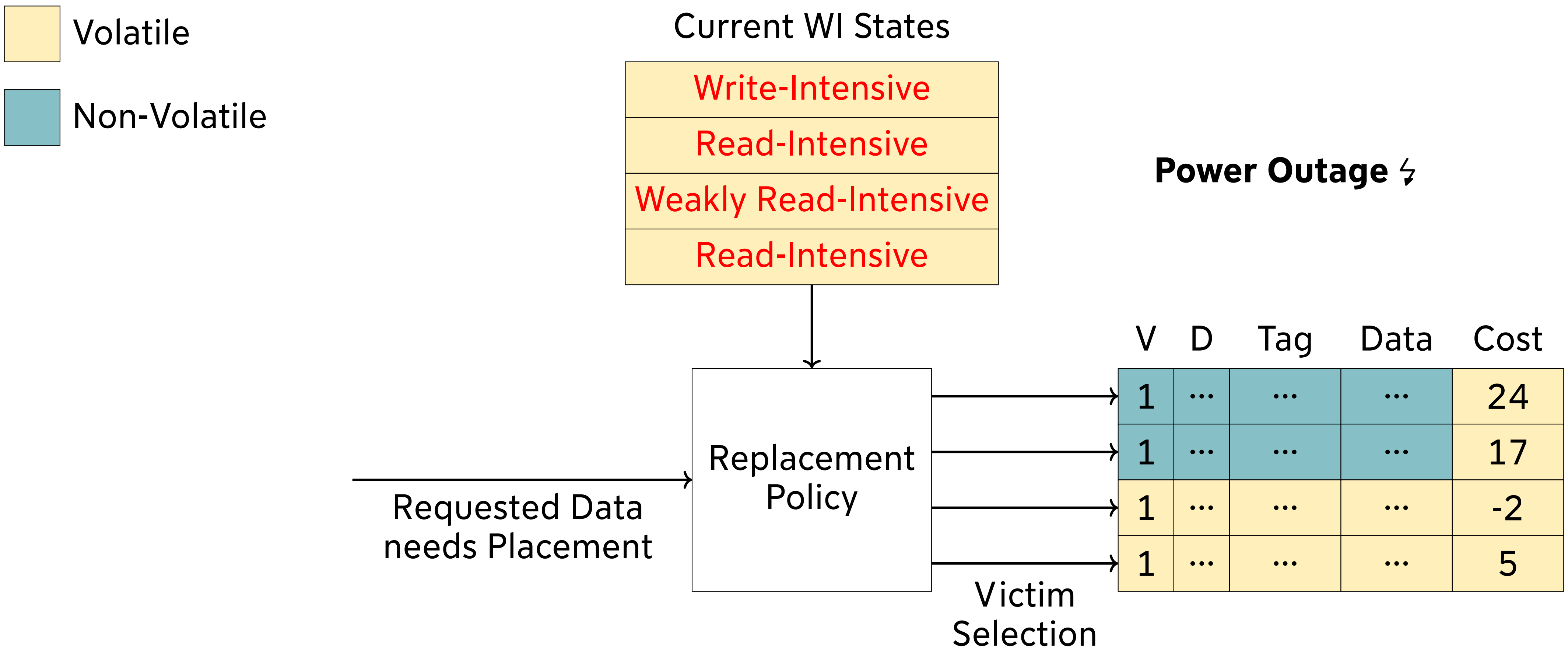


WI states updated on eviction (transition determined by cost function)

→ NVM implementation possible

→ Volatile implementation to reset all state machines to one of the 4 WI states (depending on state encoding)

Candidates for Different Technological Implementations



WI states updated on eviction (transition determined by cost function)

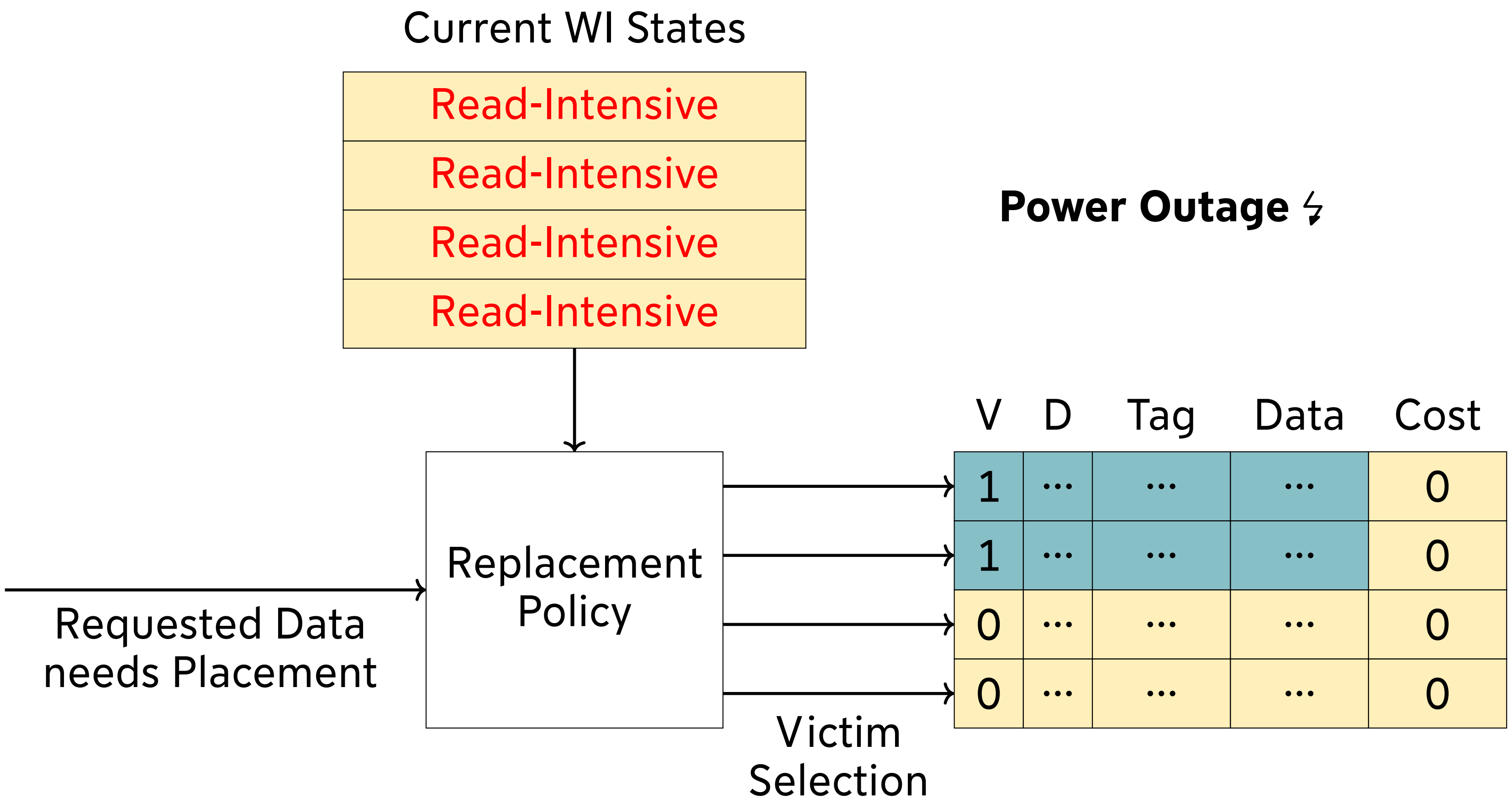
→ NVM implementation possible

→ Volatile implementation to reset all state machines to one of the 4 WI states (depending on state encoding)

Candidates for Different Technological Implementations

Volatile

Non-Volatile



WI states updated on eviction (transition determined by cost function)

→ NVM implementation possible

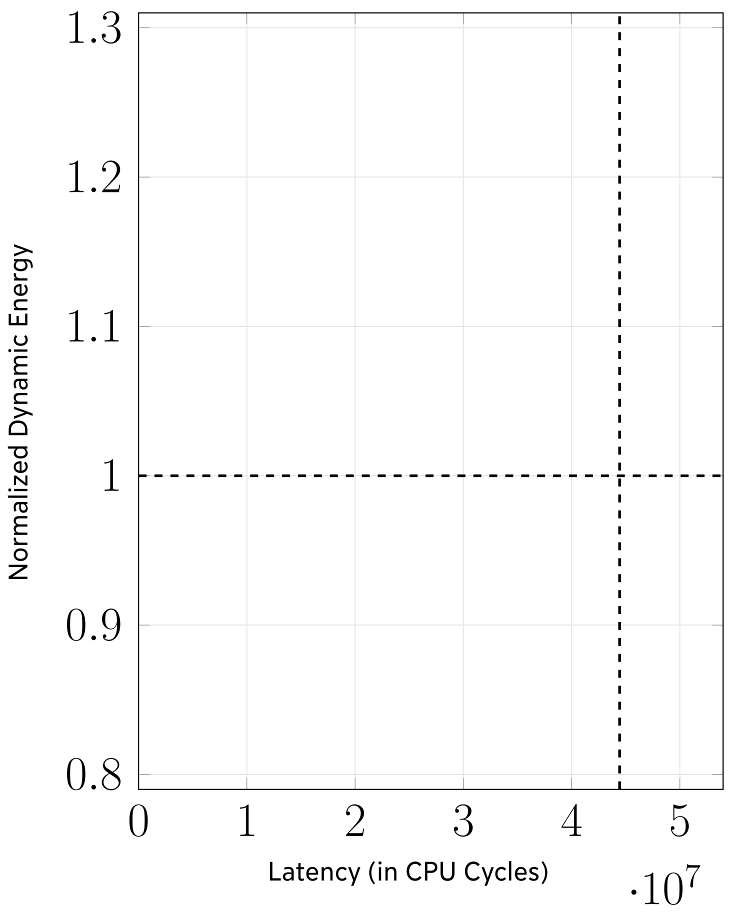
→ Volatile implementation to reset all state machines to one of the 4 WI states (depending on state encoding)

Goals Revisited

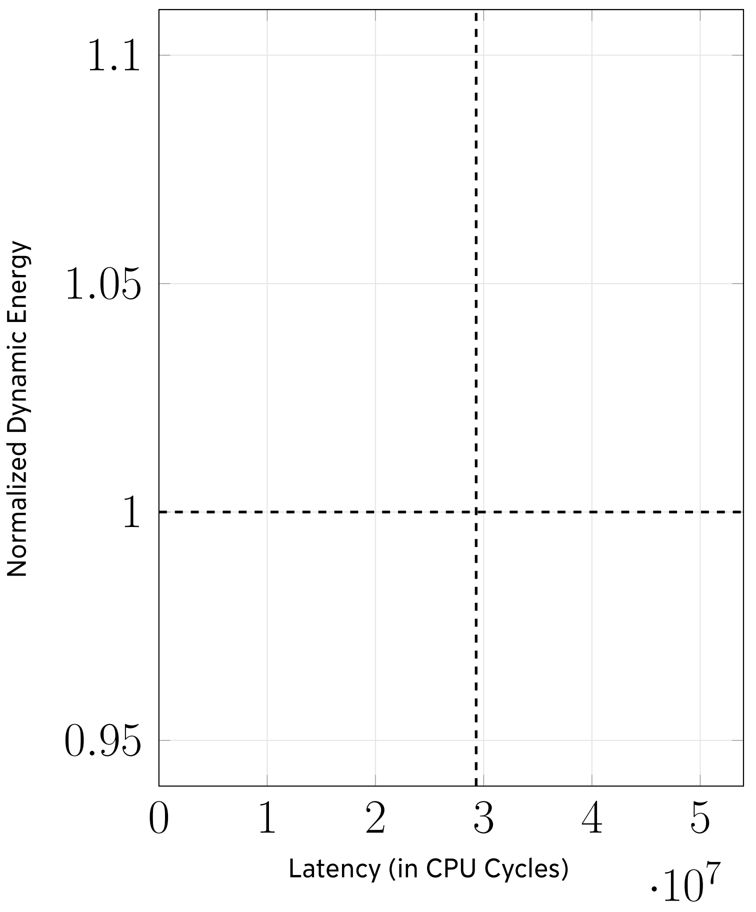
Analyze latency and energy trade-offs by comparing:

- A WI policy with a non-volatile state table (here, serving as the baseline)
- A WI policy with a volatile state table and different state encodings resetting all state machines to either
  - the Read-Intensive (RdI) state
  - the Weakly RdI state
  - the Write-Intensive (WrI) state
  - the Weakly WrI state

→ 5 different comparison points

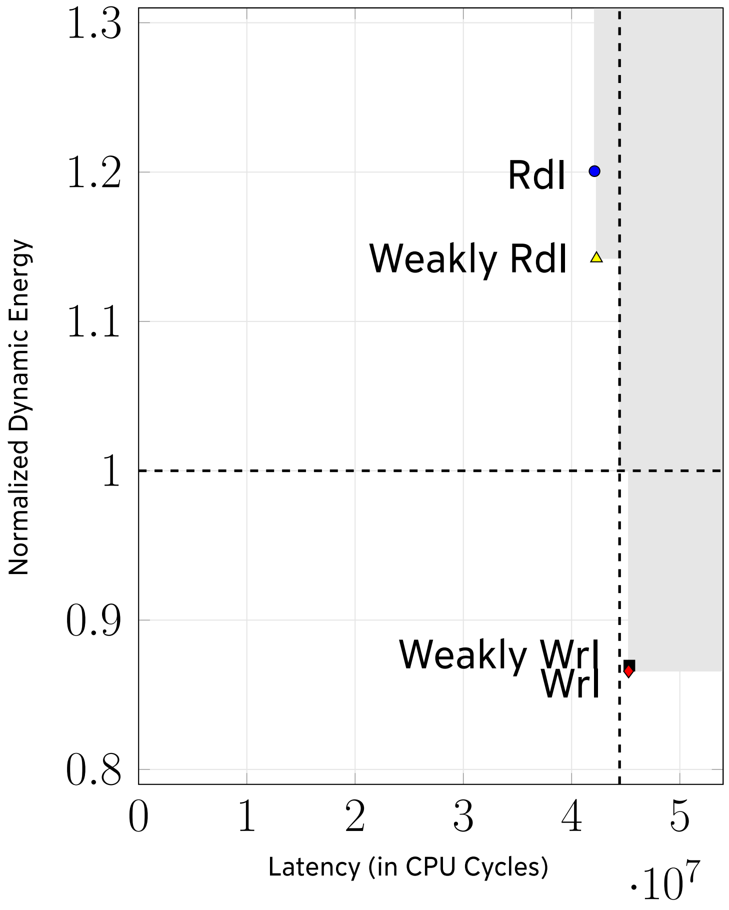


(a) **Merge Sort**

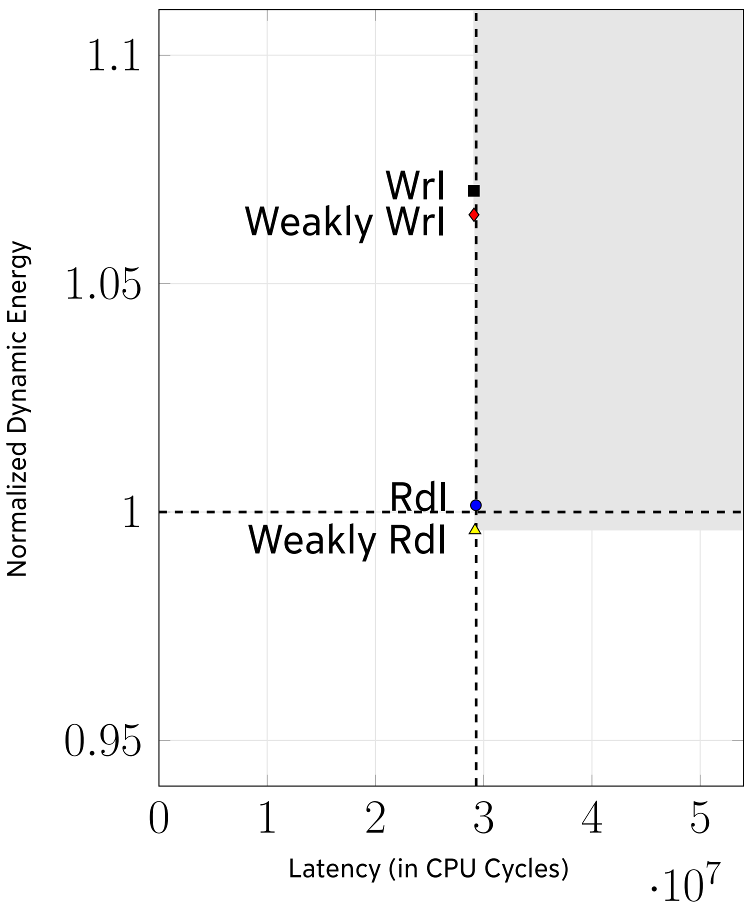


(b) **Image Processing**

Experimental Results
WI Policy



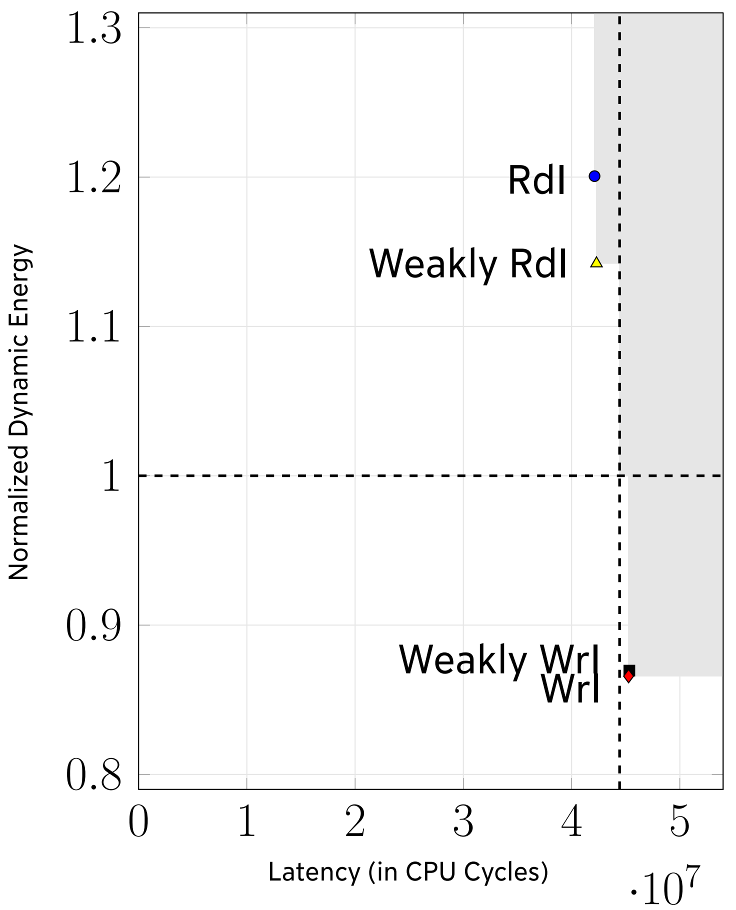
(a) **Merge Sort**



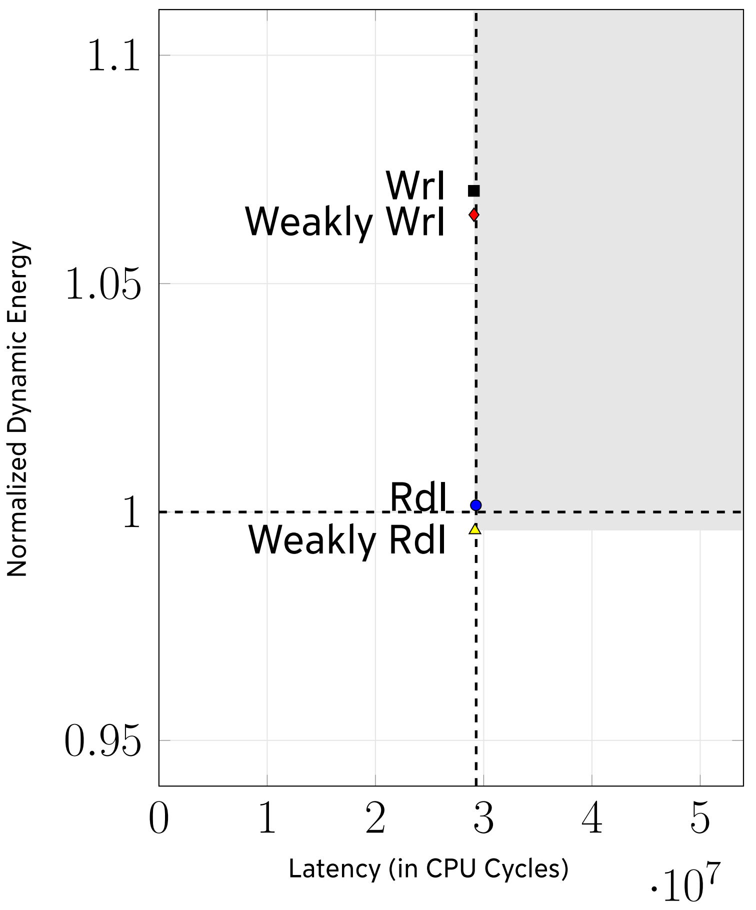
(b) **Image Processing**

Experimental Results

WI Policy



(a) **Merge Sort**



(b) **Image Processing**

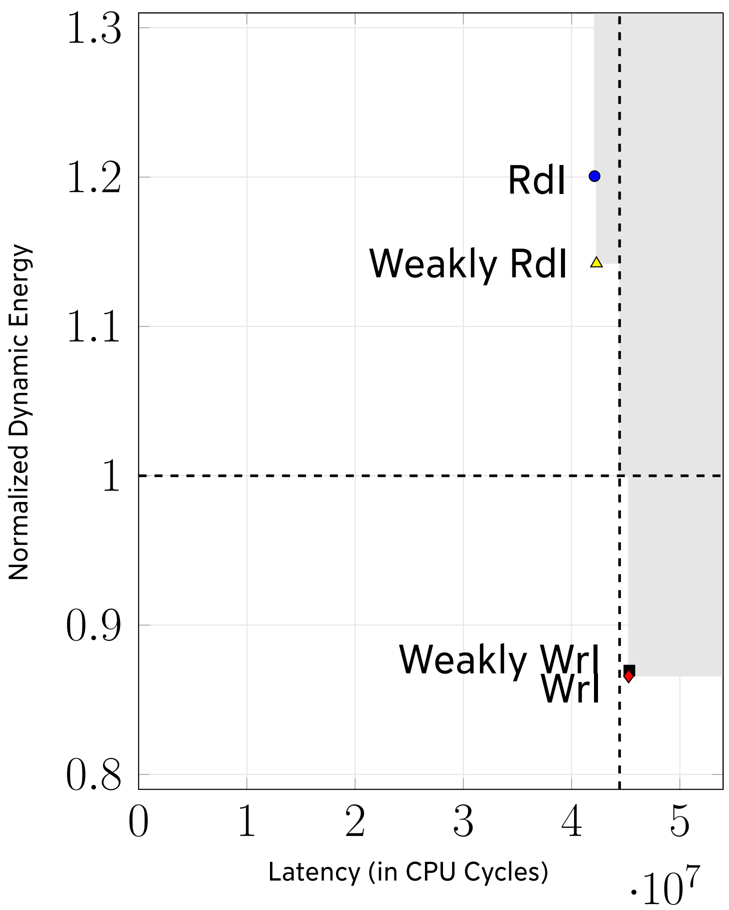
Key takeaways:

- State-machines tend to drift towards the read-intensive side following power outages

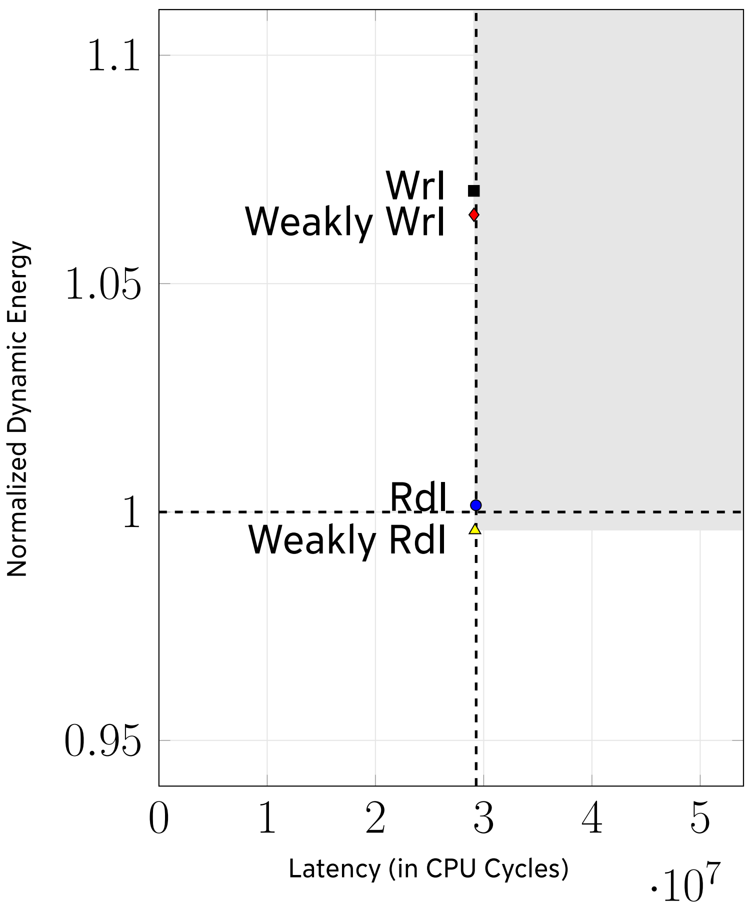
→ Harmful for write-intensive application

Experimental Results

WI Policy



(a) **Merge Sort**



(b) **Image Processing**

Key takeaways:

- State-machines tend to drift towards the read-intensive side following power outages

→ Harmful for write-intensive application

- Resetting state machines after power outages can counter mispredictions
- Most suitable reset state depends on application characteristics

→ State encoding can make up to 28% difference in dynamic energy consumption

Conclusion and Outlook

- To keep or not to keep? No general consensus regarding the ”best” option for implementing policy metadata
- However, it’s an important design decision: We have seen up to 28% difference in energy consumption by switching to a different approach of implementing replacement policy metadata

Conclusion and Outlook

- To keep or not to keep? No general consensus regarding the "best" option for implementing policy metadata
- However, it's an important design decision: We have seen up to 28% difference in energy consumption by switching to a different approach of implementing replacement policy metadata

→ When developing new policies: Evaluate different options for technologically implementing their metadata to unlock additional potential in energy/latency savings

→ Knowing about the characteristics (write intensity) of your applications, helps to realize better design decisions

→ Do not undermine the role of niches in the design space

Thank you for your interest and attention!

Any questions?

References

Sources

[AYC16] J. Ahn, S. Yoo, and K. Choi. "Prediction Hybrid Cache: An Energy-Efficient STT-RAM Cache Architecture". In: *IEEE Transactions on Computers* 65.3 (2016), pp. 940–951. DOI: 10.1109/TC.2015.2435772.

[Bin+11] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. "The gem5 simulator". In: *SIGARCH Comput. Archit. News* 39.2 (Aug. 2011), pp. 1–7. DOI: 10.1145/2024716.2024718.

[Cho+12] Y. Choi et al. "A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth". In: *2012 IEEE International Solid-State Circuits Conference*. 2012, pp. 46–48. DOI: 10.1109/ISSCC.2012.6176872.

[Don+12] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31.7 (2012), pp. 994–1007. DOI: 10.1109/TCAD.2012.2185930.

[Por+15] M. Poremba et al. "NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems". In: *IEEE Computer Architecture Letters* 14 (2015), pp. 140–143.