A Case for Speculative Address Translation with Rapid Validation for GPUs

Junhyeok Park[1] Osang Kwon[1] Yongho Lee[1]

Seongwook Kim[1] Gwangeun Byeon[1] Jihun Yoon[1]

Prashant J. Nair[2] Seokin Hong[1]

1 SUNG KYUN KWAN UNIVERSITY

2 THE UNIVERSITY OF BRITISH COLUMBIA

Outline

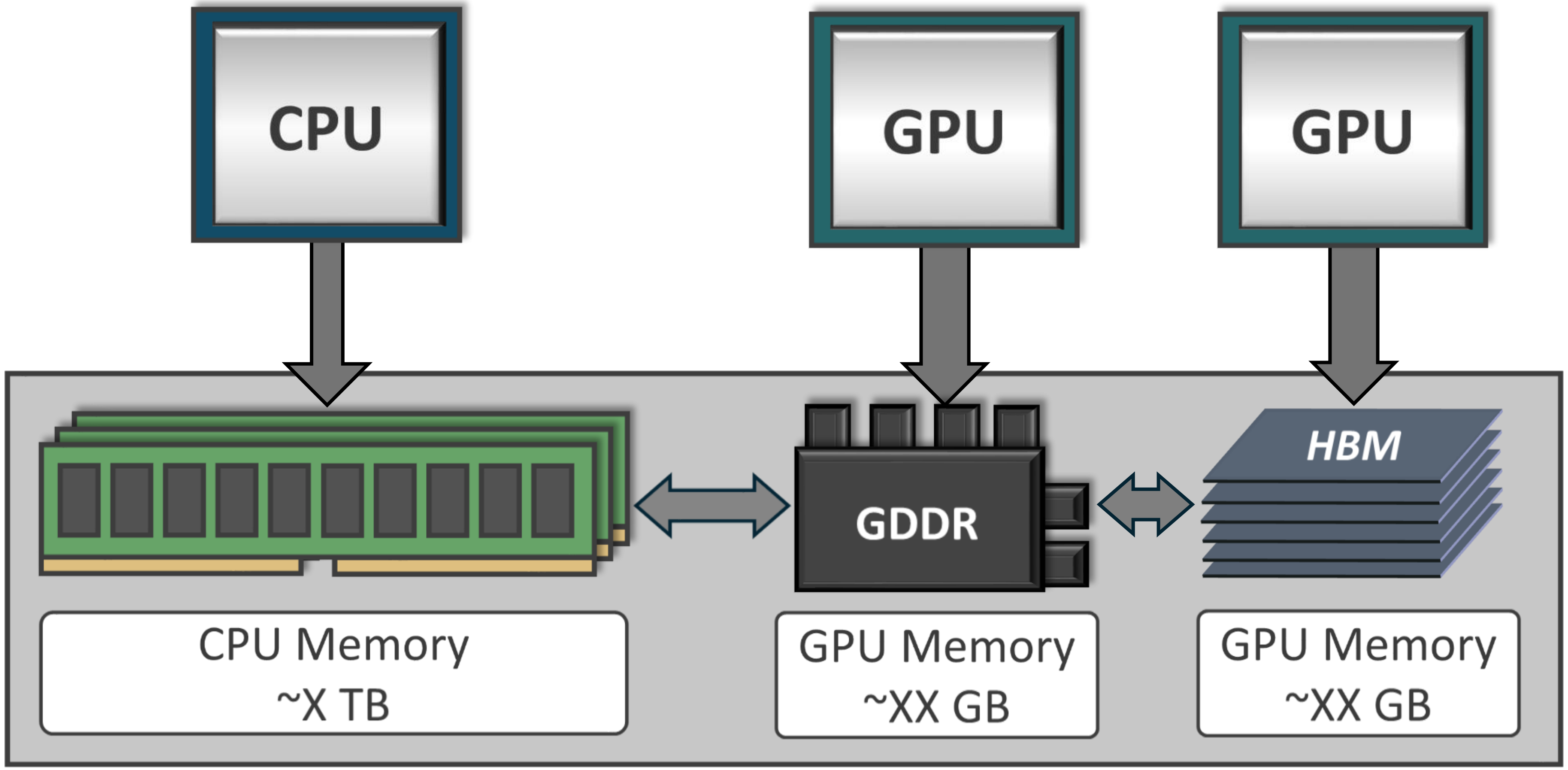
1. Virtual Memory in GPUs – Challenges ☜

2. Avatar
   - Speculative address translation
   - Rapid Validation

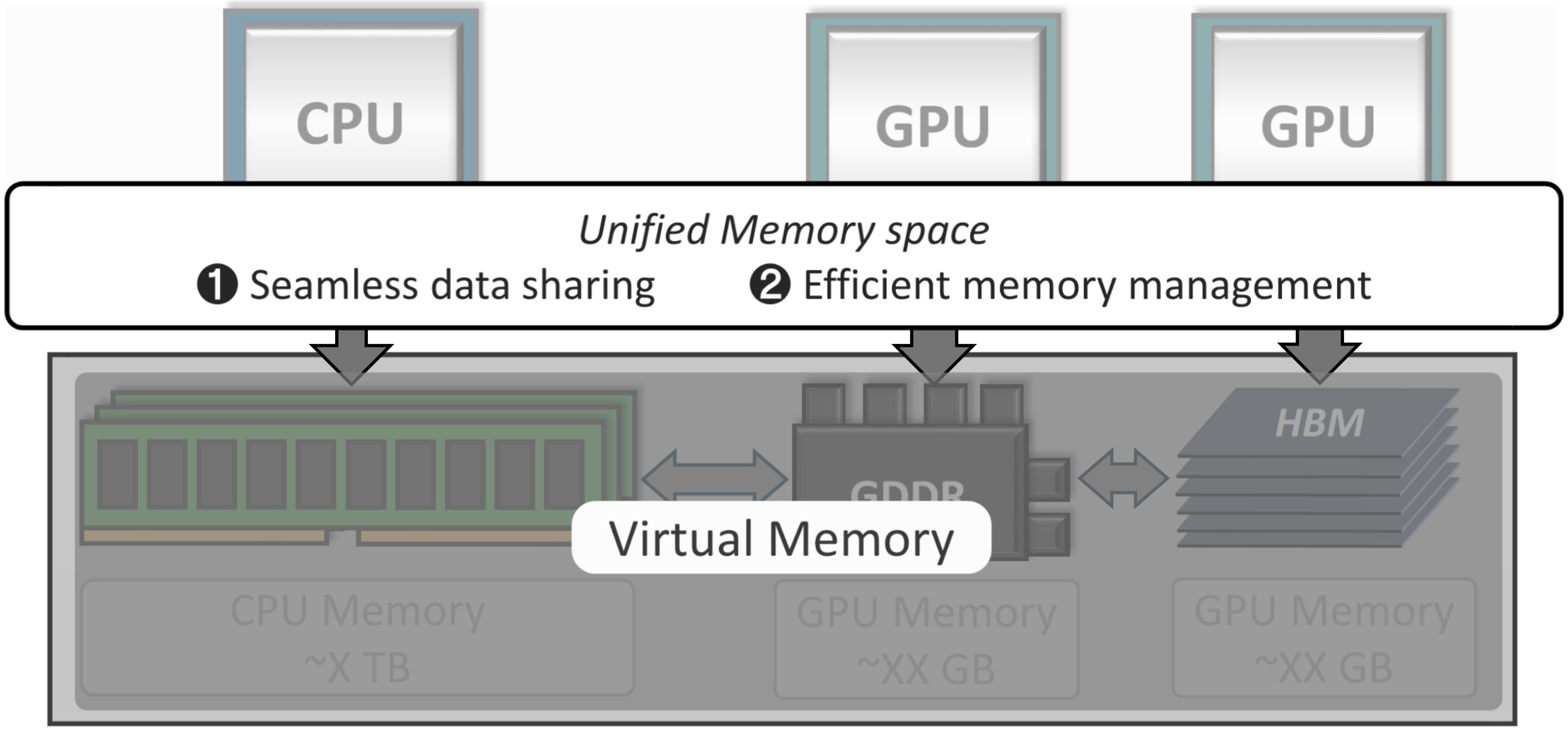
3. Evaluation

4. Summary

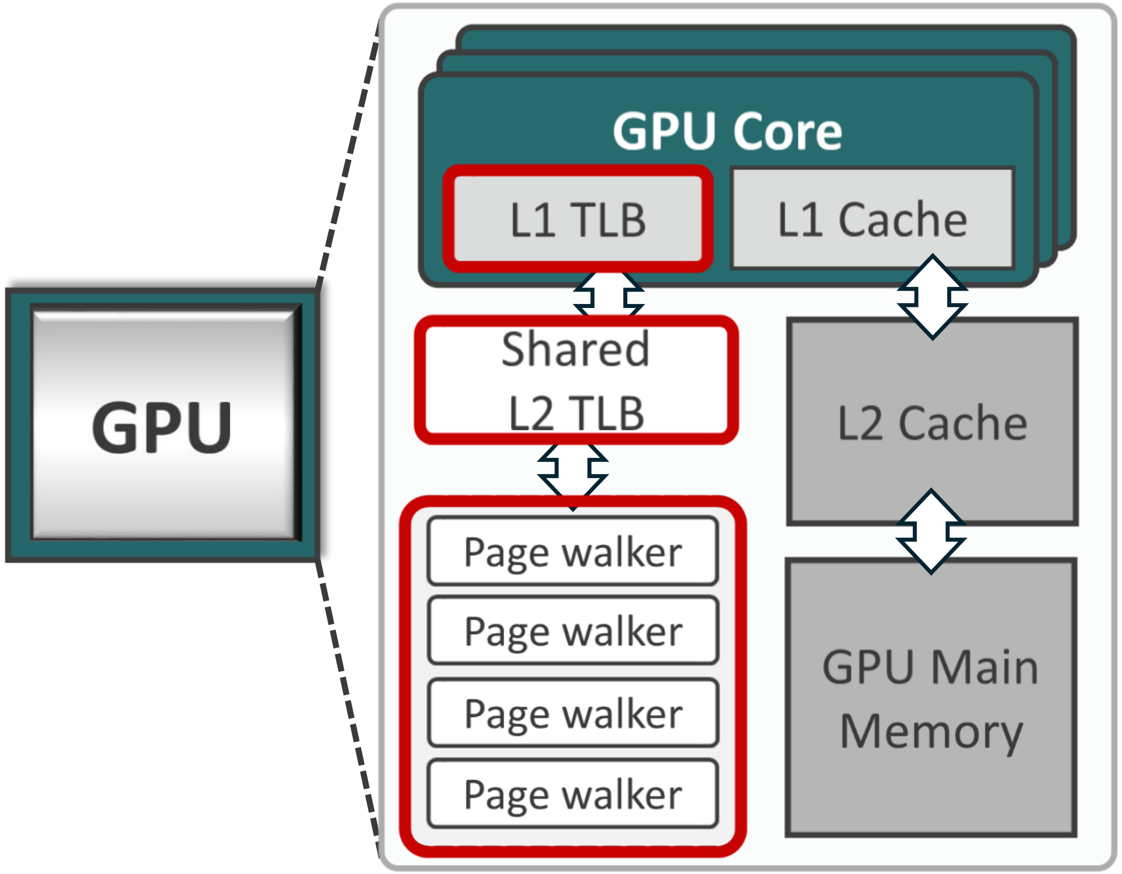
GPUs in Heterogeneous Platform



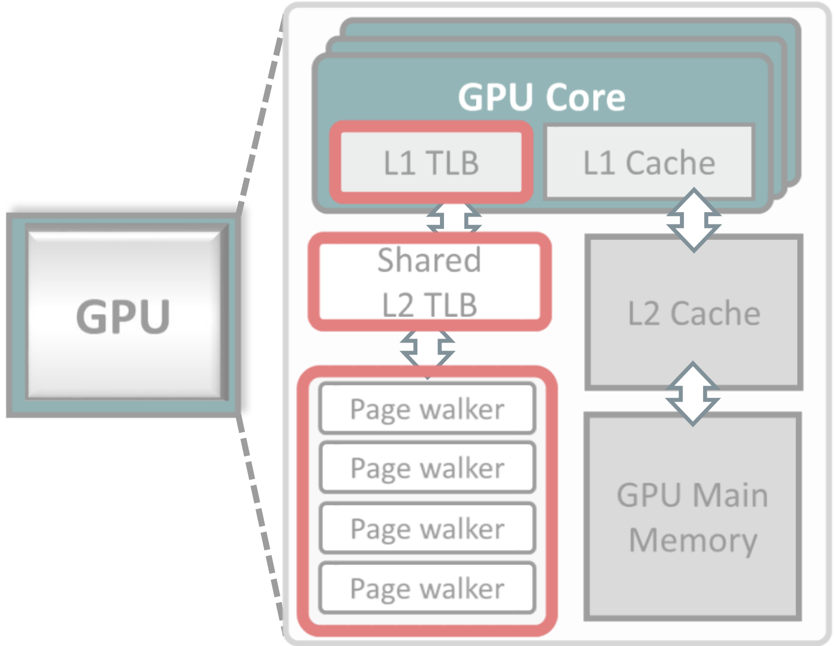
GPUs in Heterogeneous Platform

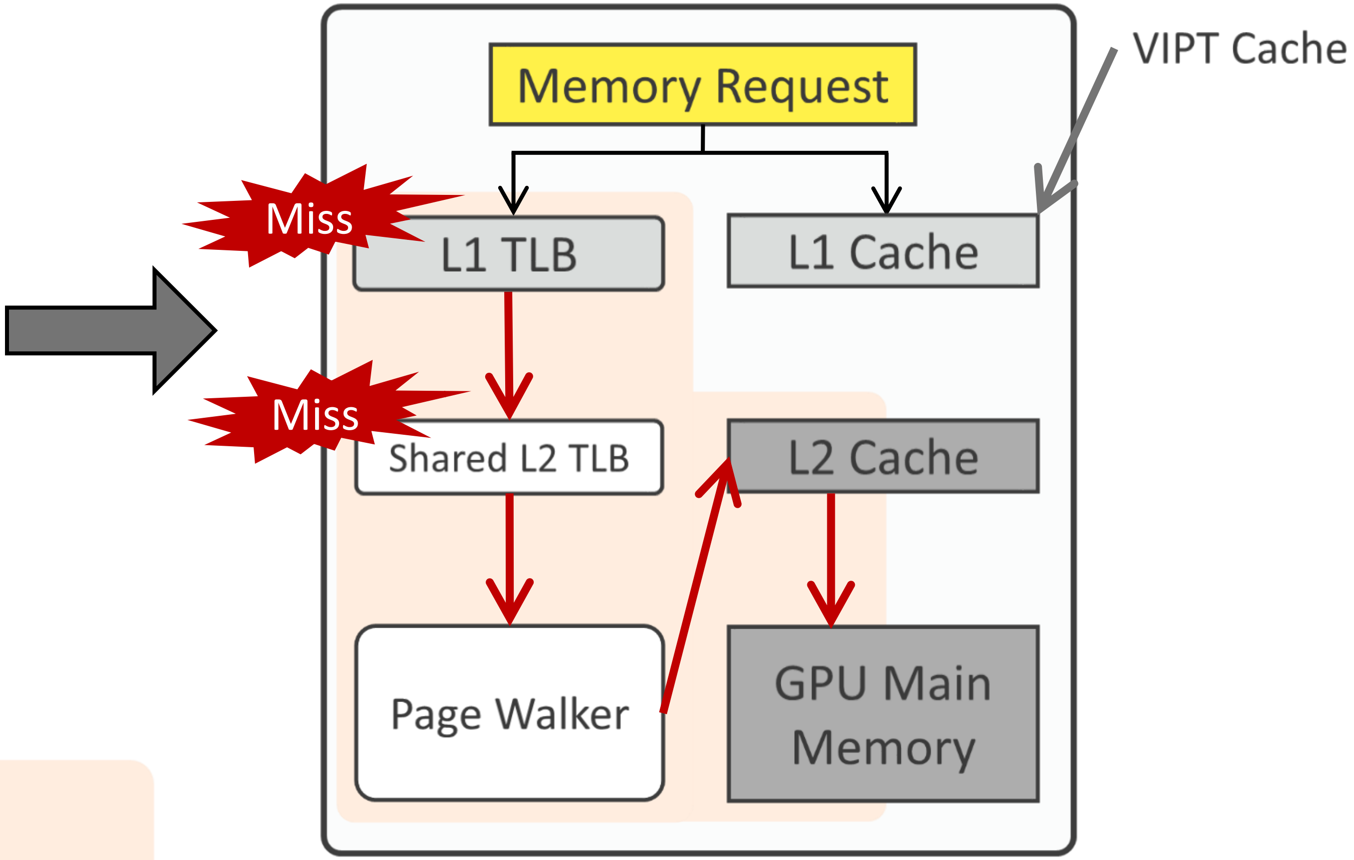


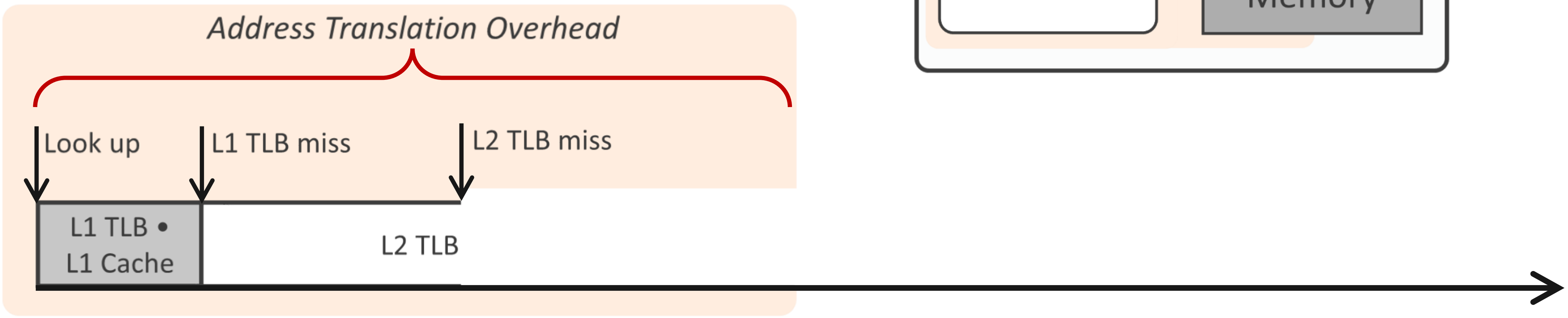
Address Translation in GPUs



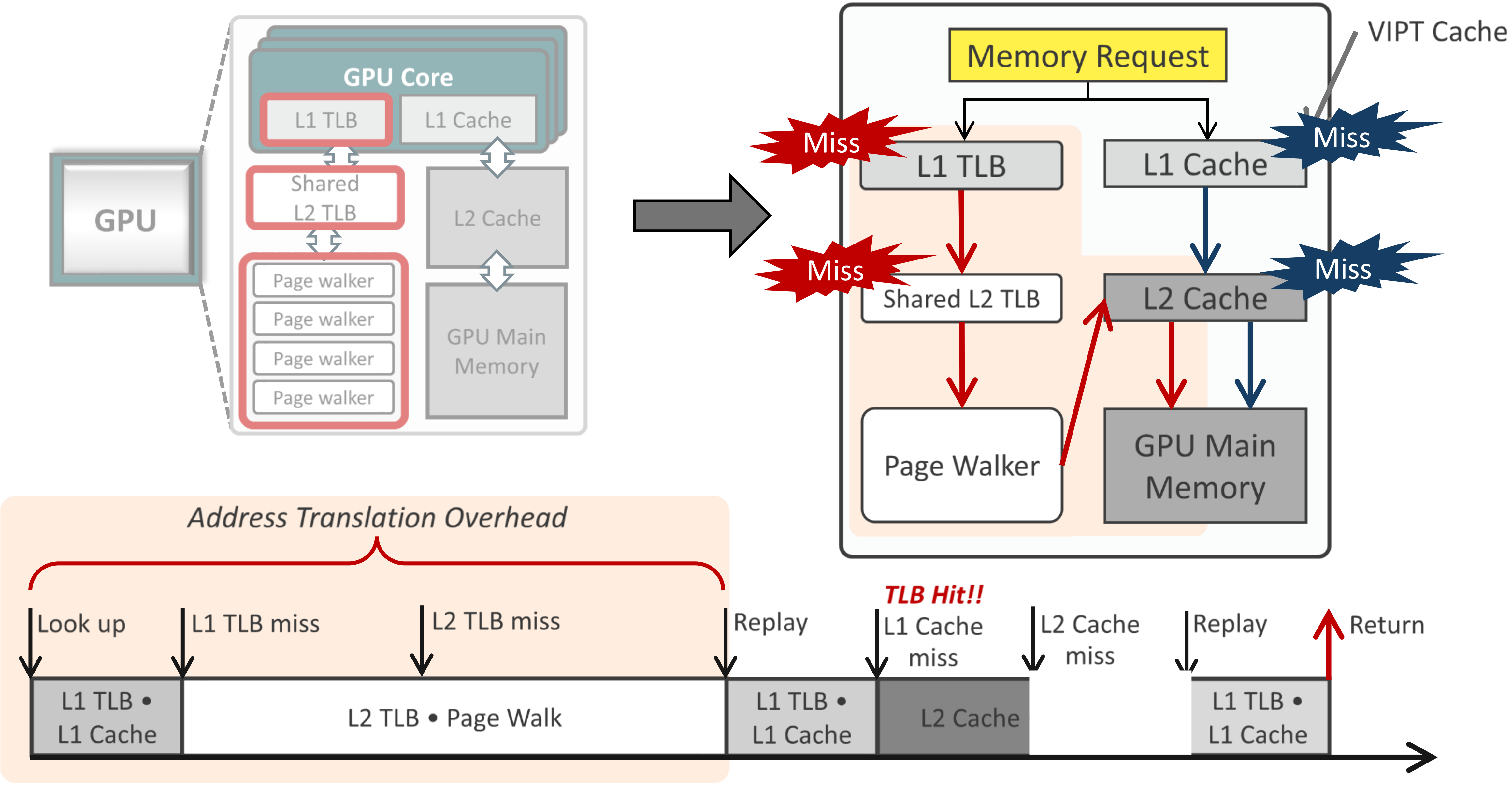
Address Translation in GPUs





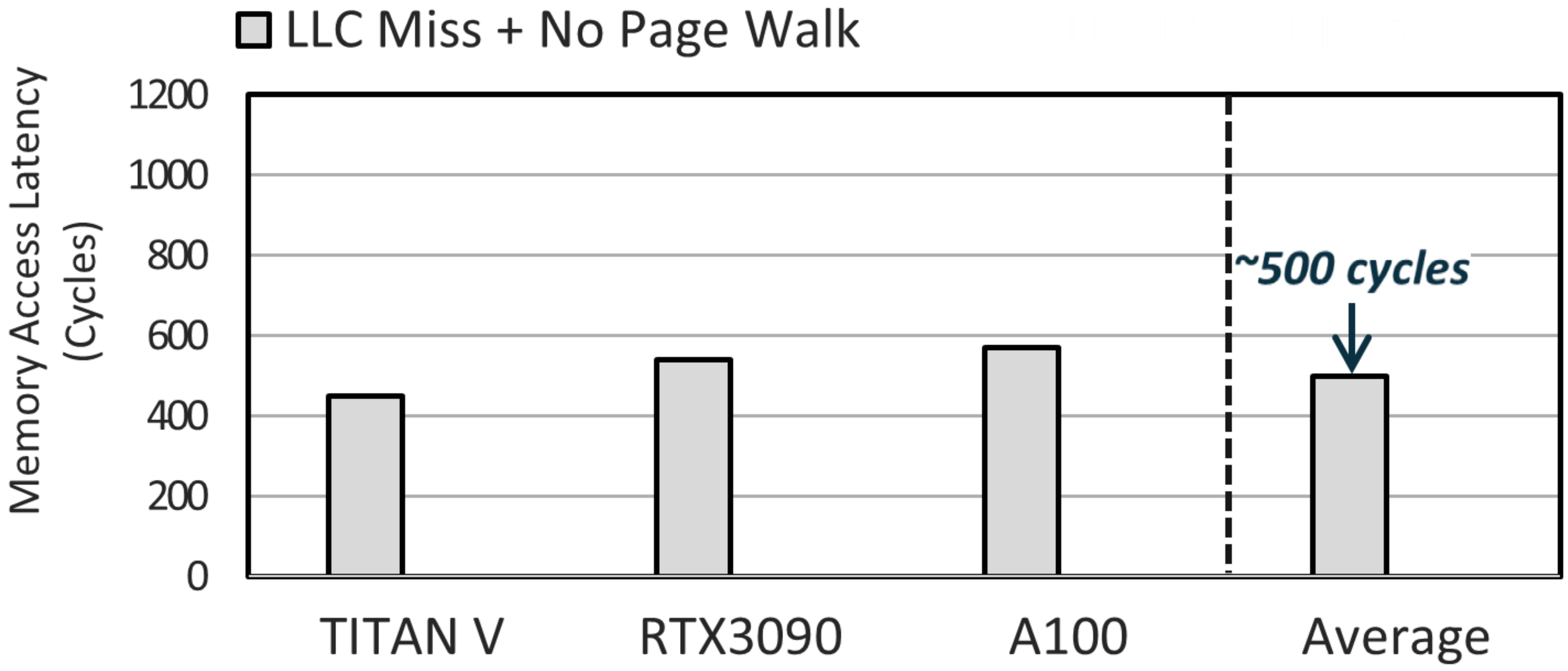


Address Translation in GPUs



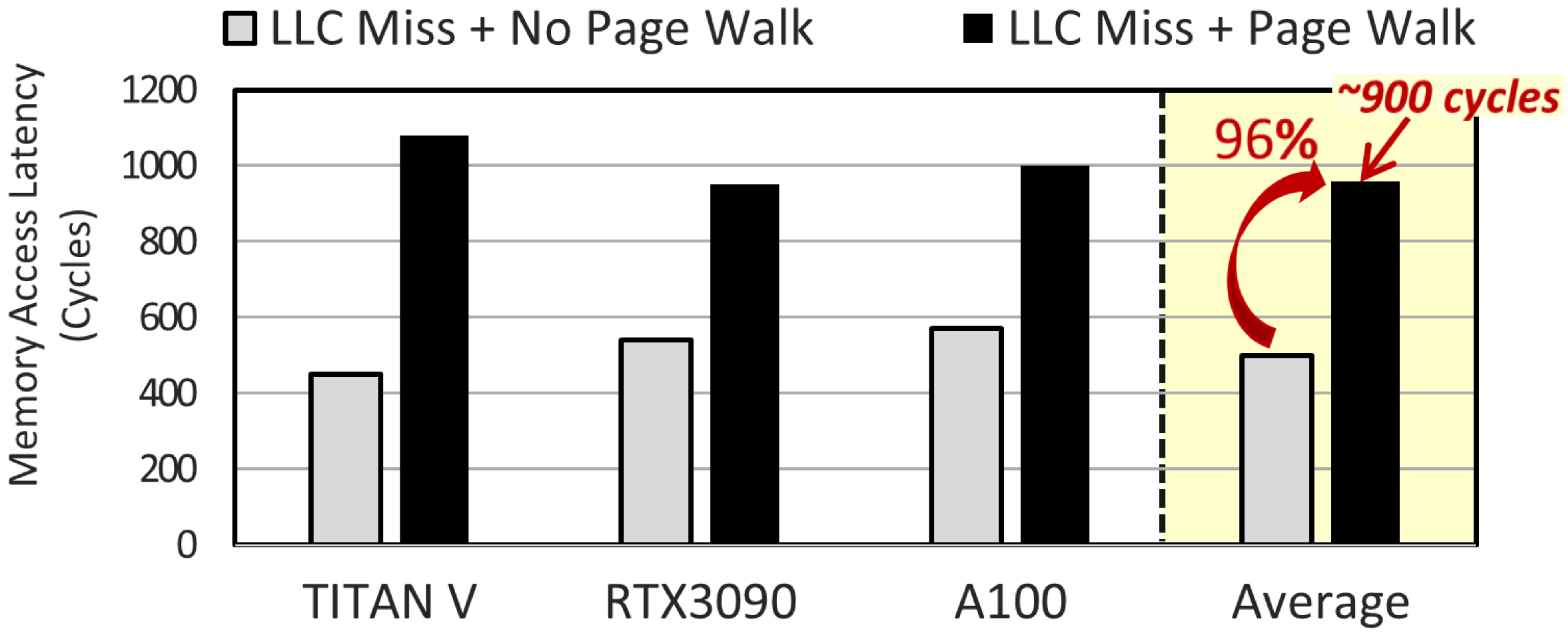
Address Translation in GPUs

Memory access latency considering address translation overhead



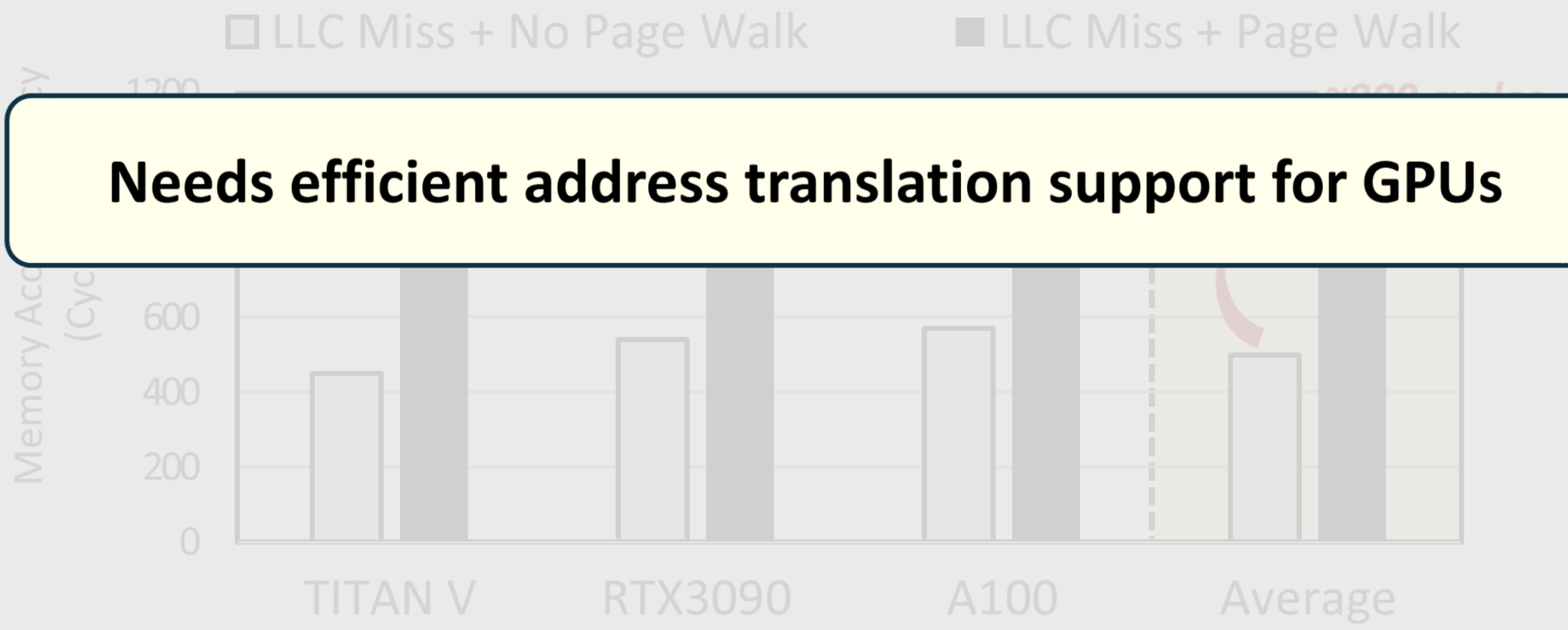
Address Translation in GPUs

Address translation (page walk) increase **memory access latency up to 1.96x**



Address Translation in GPUs

Address translation (page walk) increase memory access latency up to 1.96x



Needs efficient address translation support for GPUs

Outline

1. Virtual Memory in GPUs – Challenges

2. Avatar ☜

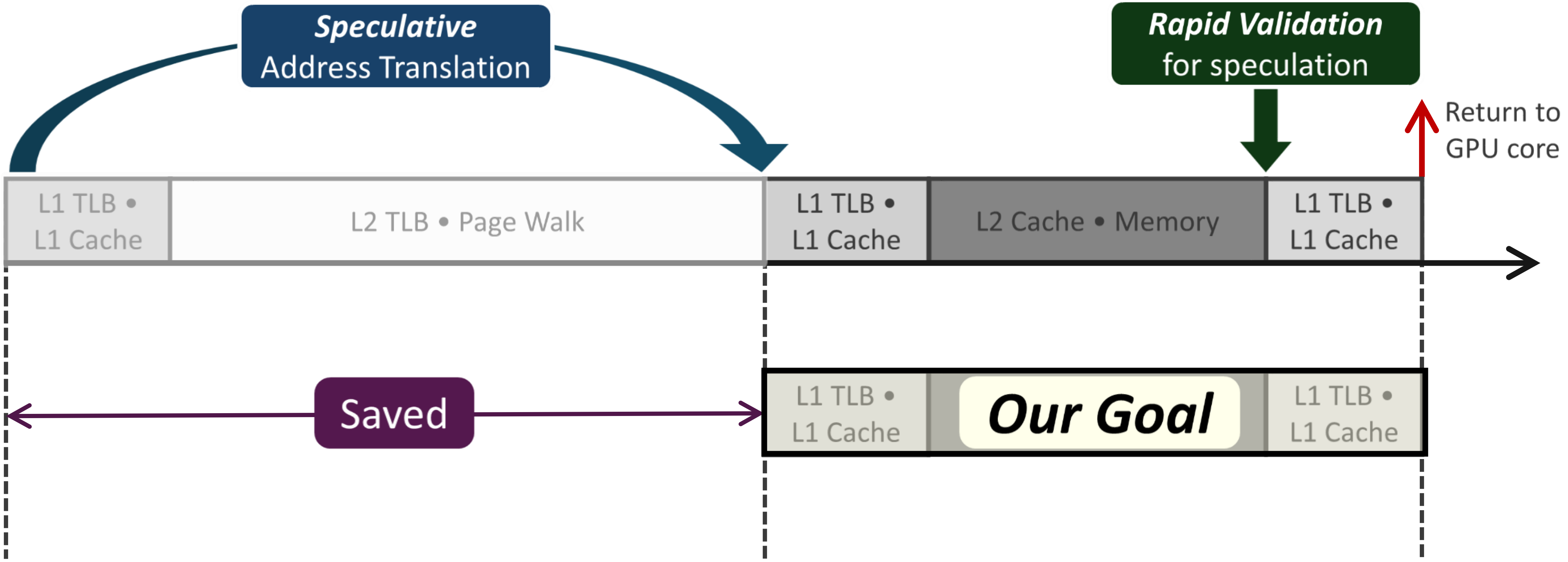
- Speculative address translation
- Rapid Validation

3. Evaluation

4. Summary

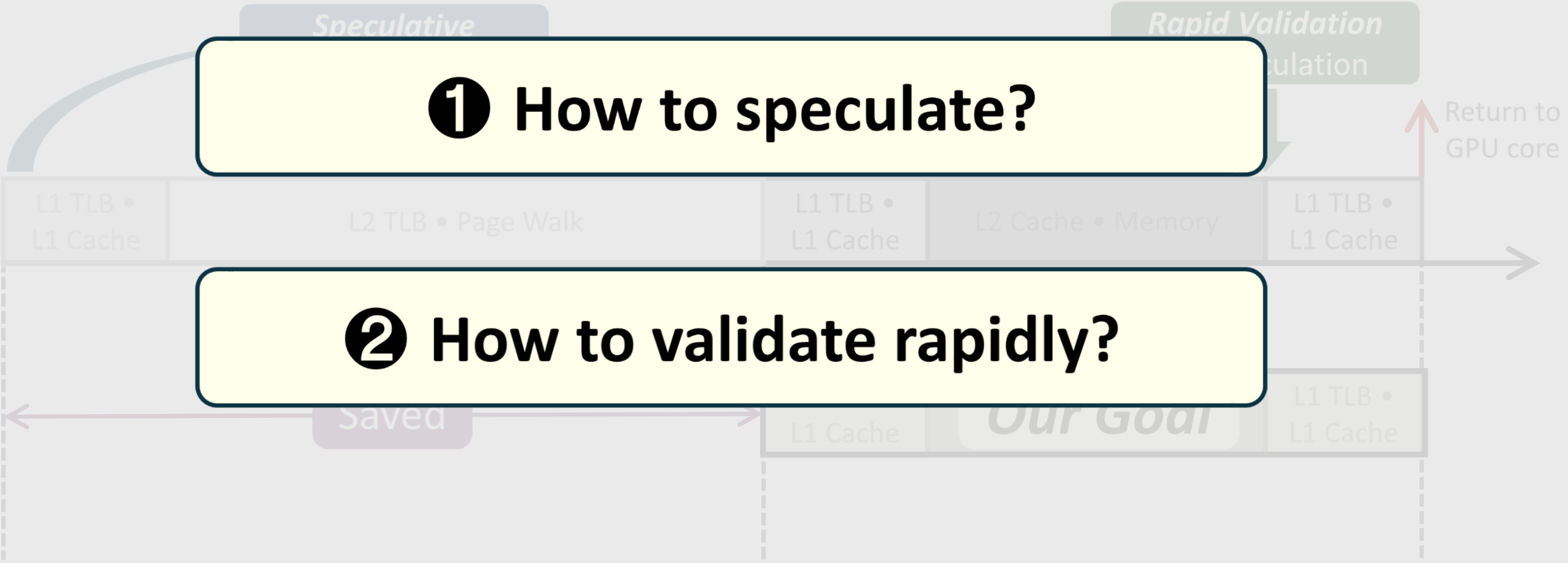
Avatar

Accelerated ***V***irtual ***A***ddress ***T***ranslation with ***A***ddress Speculation and ***R***apid Validation



Avatar

Accelerated Virtual Address Translation with Address Speculation and Rapid Validation



❶ How to speculate?

❷ How to validate rapidly?

Outline

1. Virtual Memory in GPUs – Challenges

2. Avatar
   - **Speculative address translation** ☜
   - Rapid Validation

3. Evaluation

4. Summary

Speculative Address Translation

How to speculate?

❶ Key Observation

❷ Implementation: CAST

Speculative Address Translation

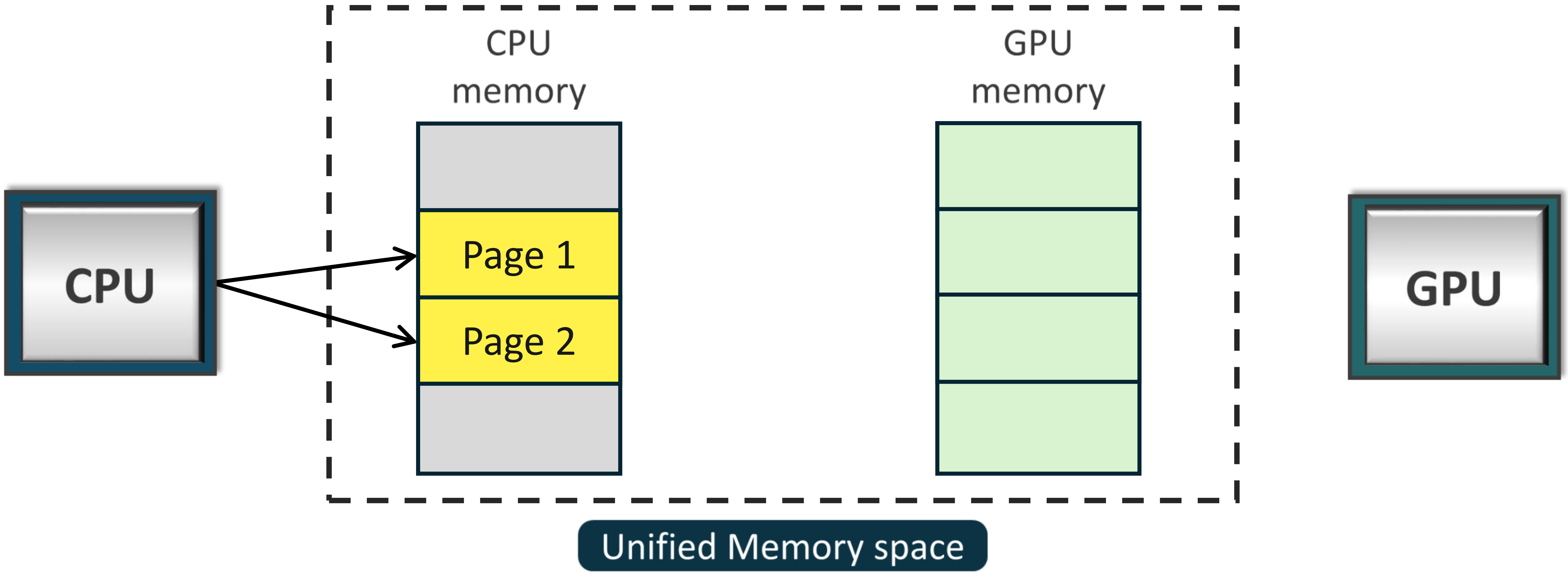
How to speculate?

❶ Key Observation

❷ Implementation: CAST

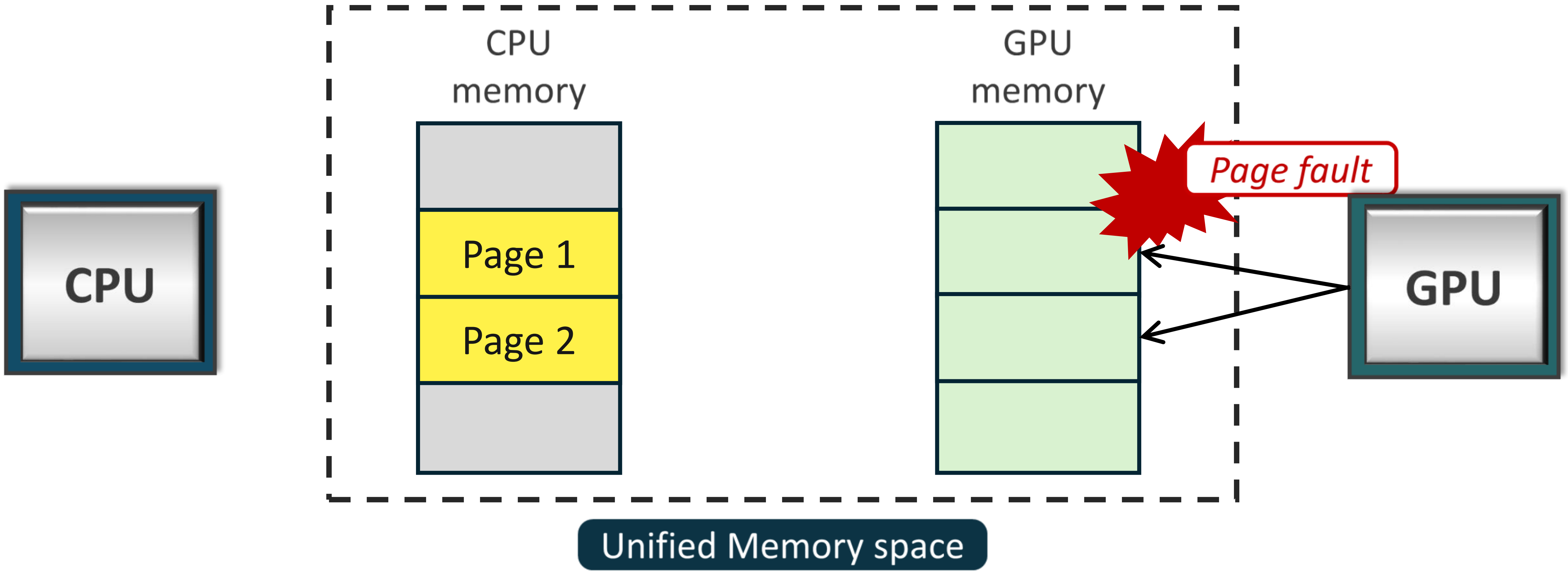
Key Observation – Page Contiguity

Memory Management in Unified Memory



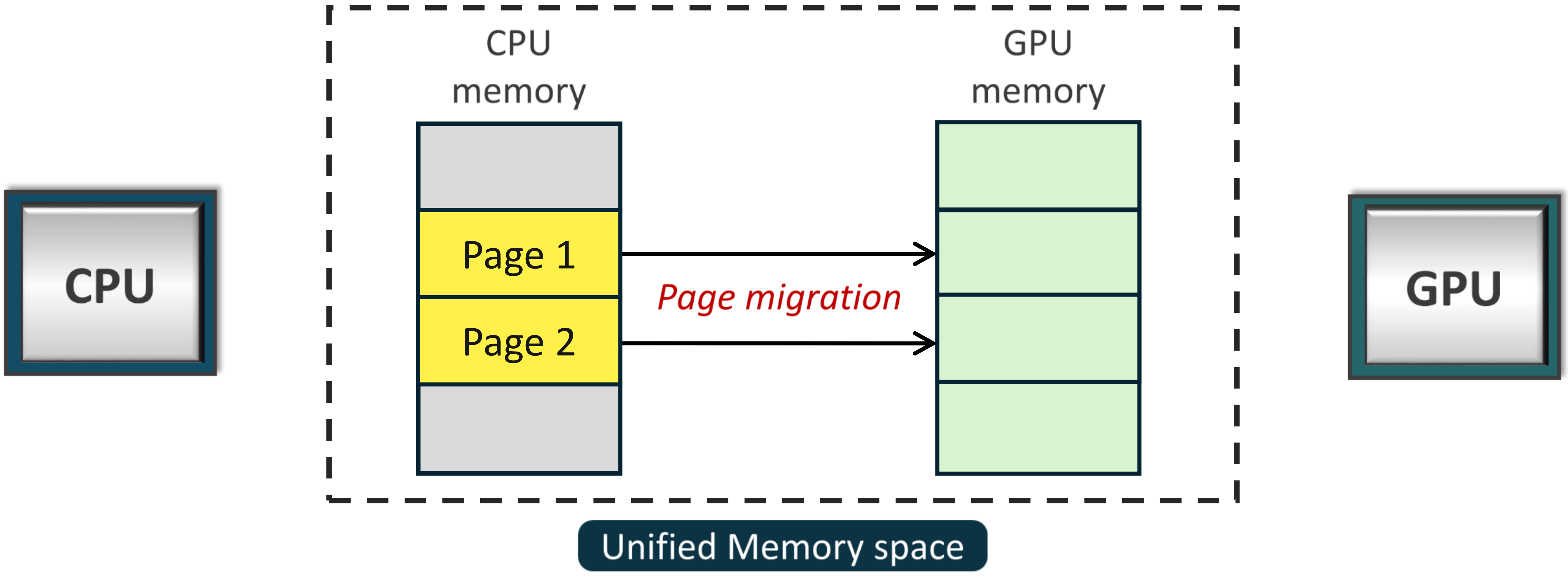
Key Observation – Page Contiguity

Memory Management in Unified Memory



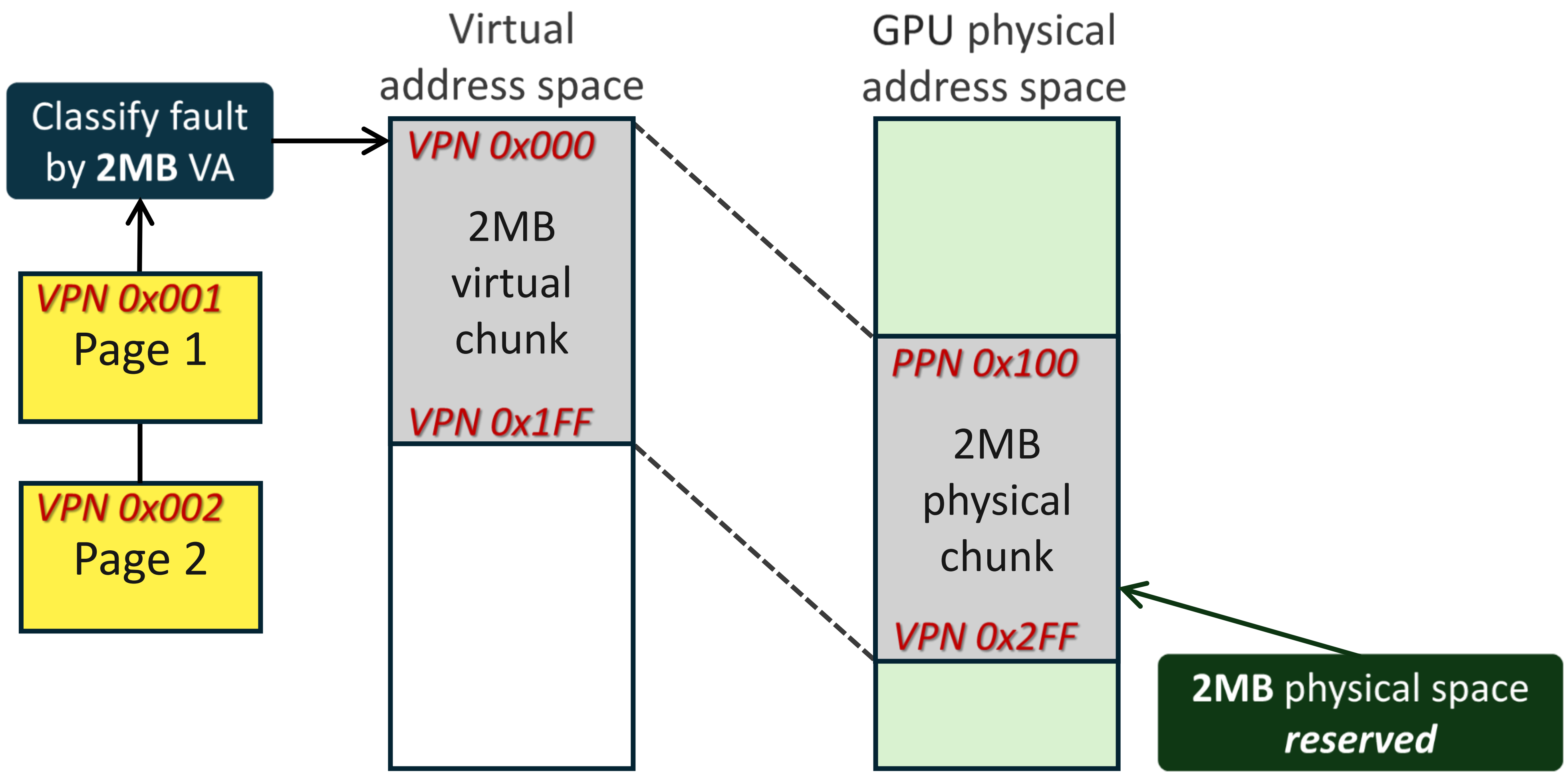
Key Observation – Page Contiguity

Memory Management in Unified Memory



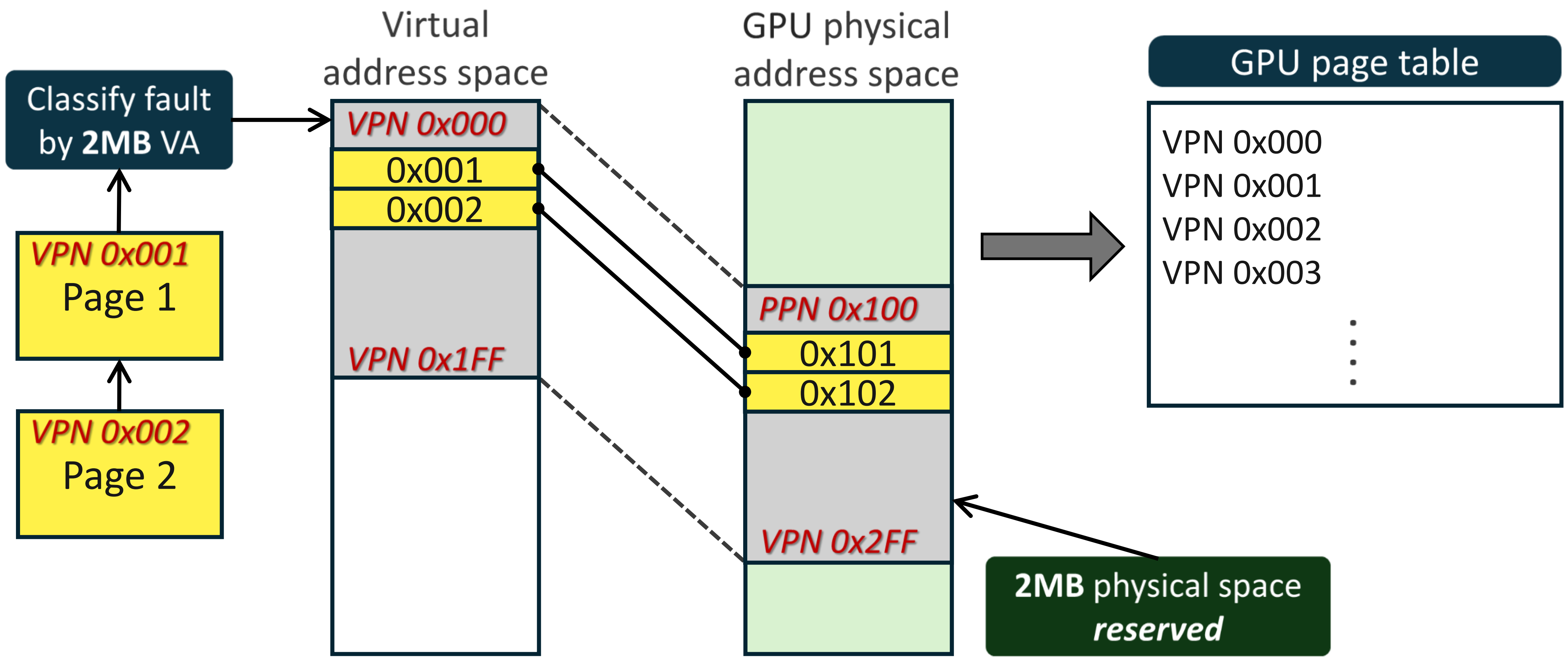
Key Observation – Page Contiguity

Leveraging 2MB page contiguity

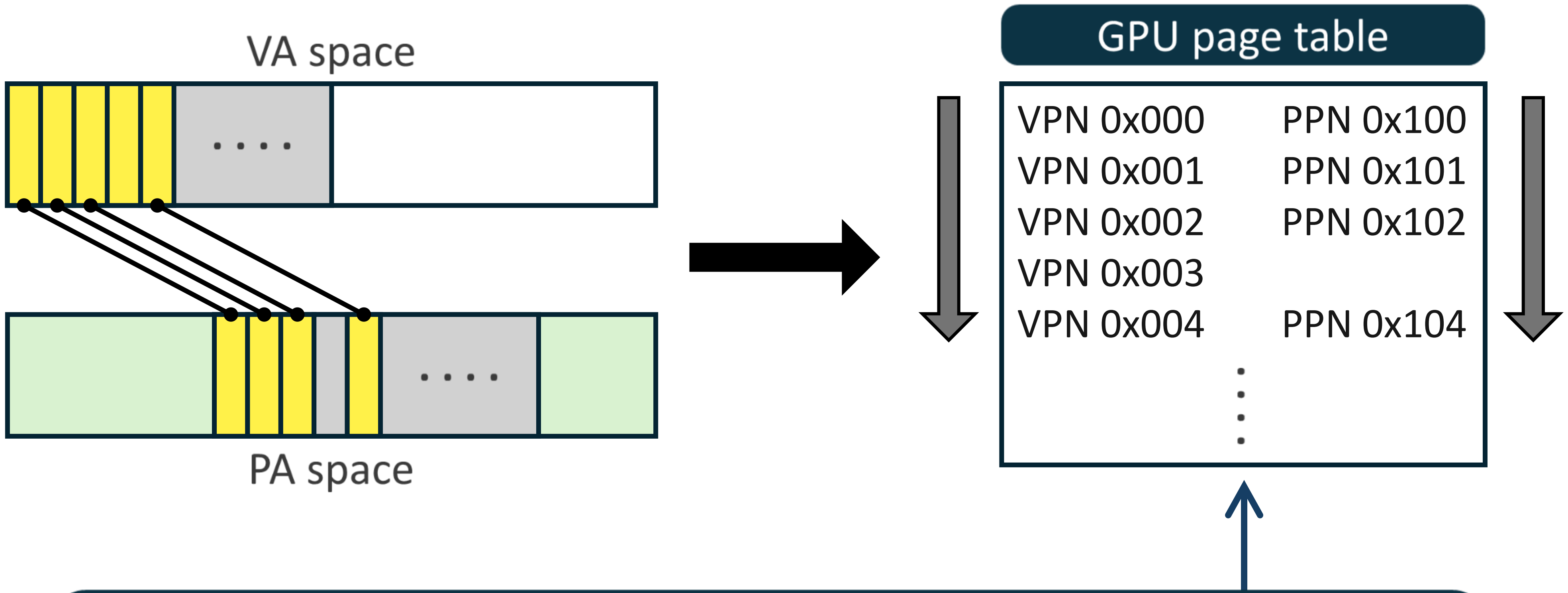


Key Observation – Page Contiguity

Leveraging 2MB page contiguity

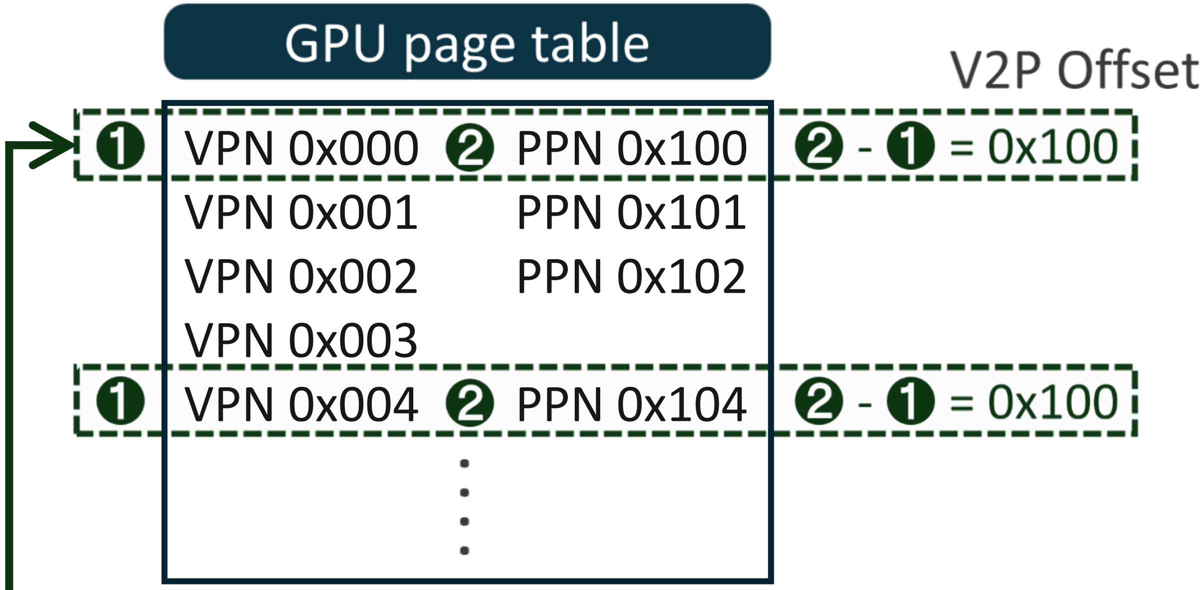


Key Observation – Page Contiguity



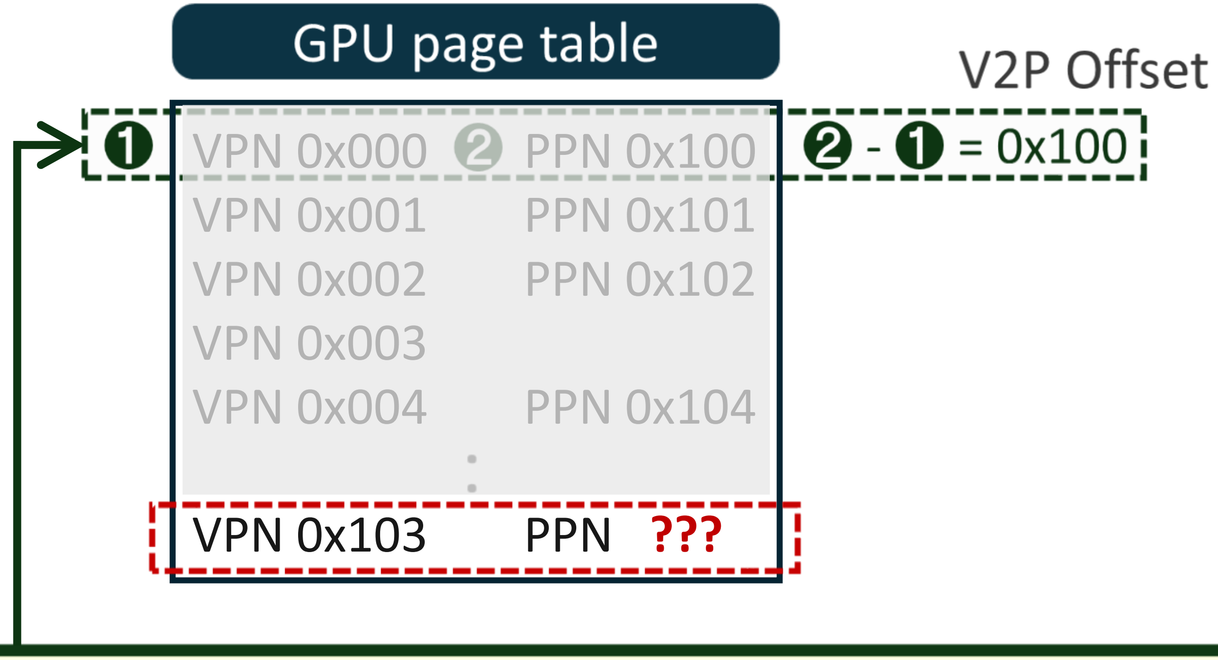
Page contiguity *:*
contiguous virtual addresses are mapped to contiguous physical addresses, **particularly within each 2MB region**

Exploiting Page Contiguity for Speculation



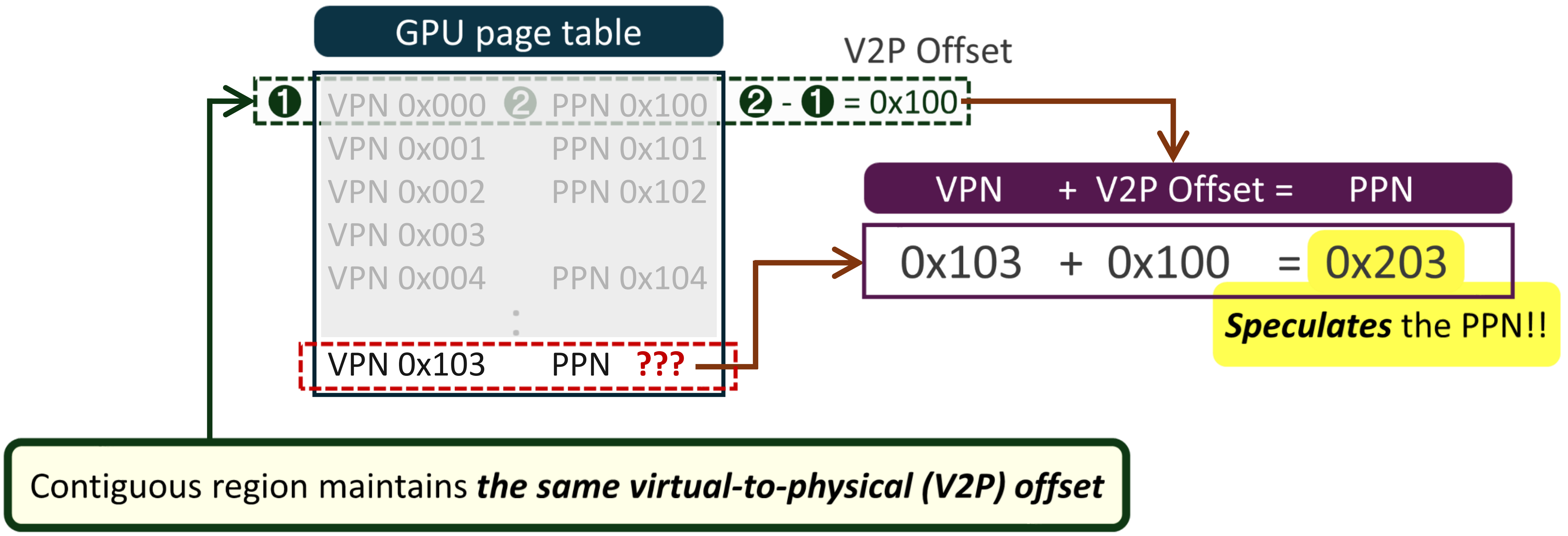
Contiguous region maintains ***the same virtual-to-physical (V2P) offset***

Exploiting Page Contiguity for Speculation

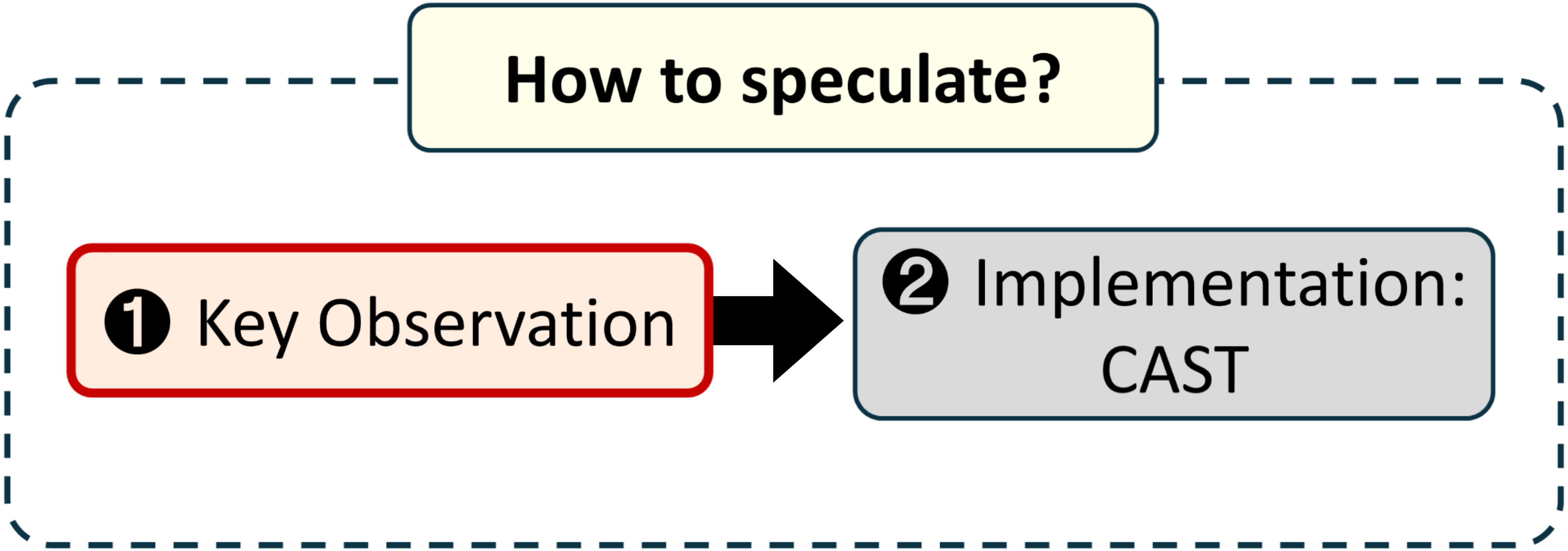


Contiguous region maintains ***the same virtual-to-physical (V2P) offset***

Exploiting Page Contiguity for Speculation



Speculative Address Translation

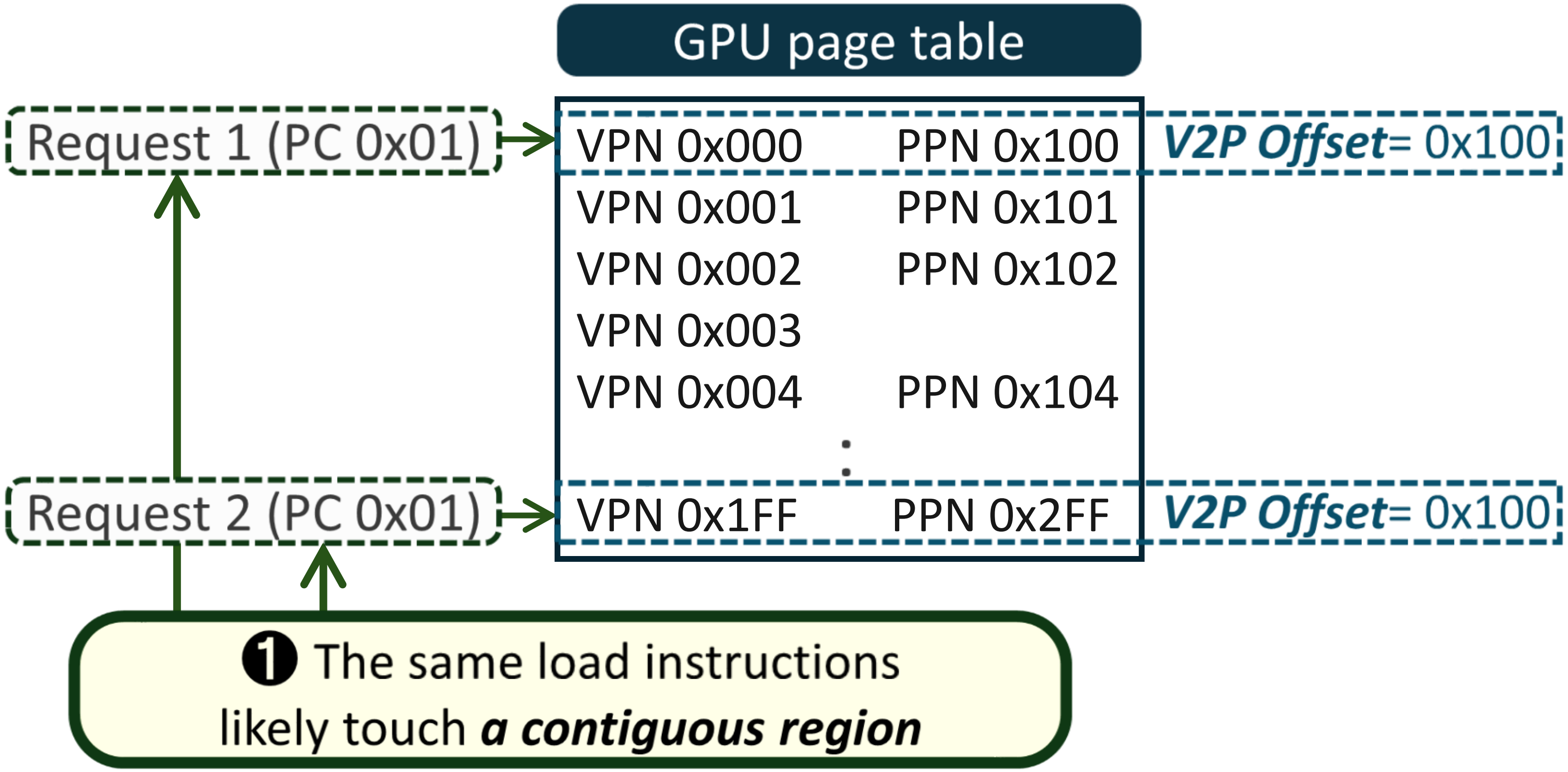


Speculative Address Translation

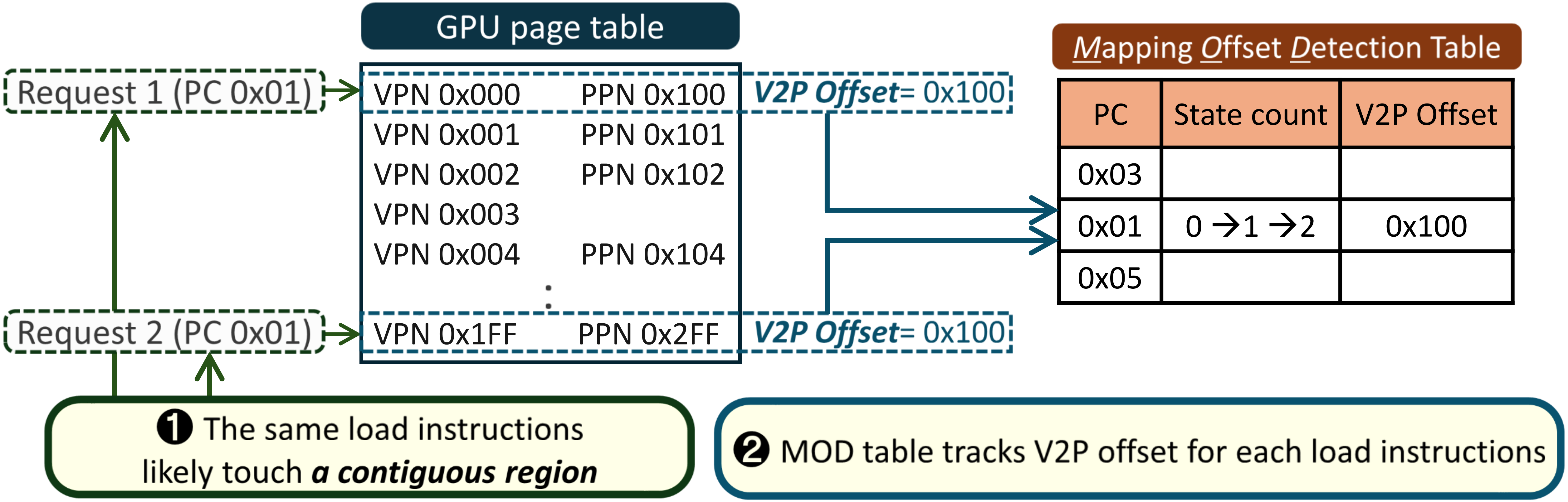
How to speculate?

❶ Key Observation → ❷ Implementation: CAST

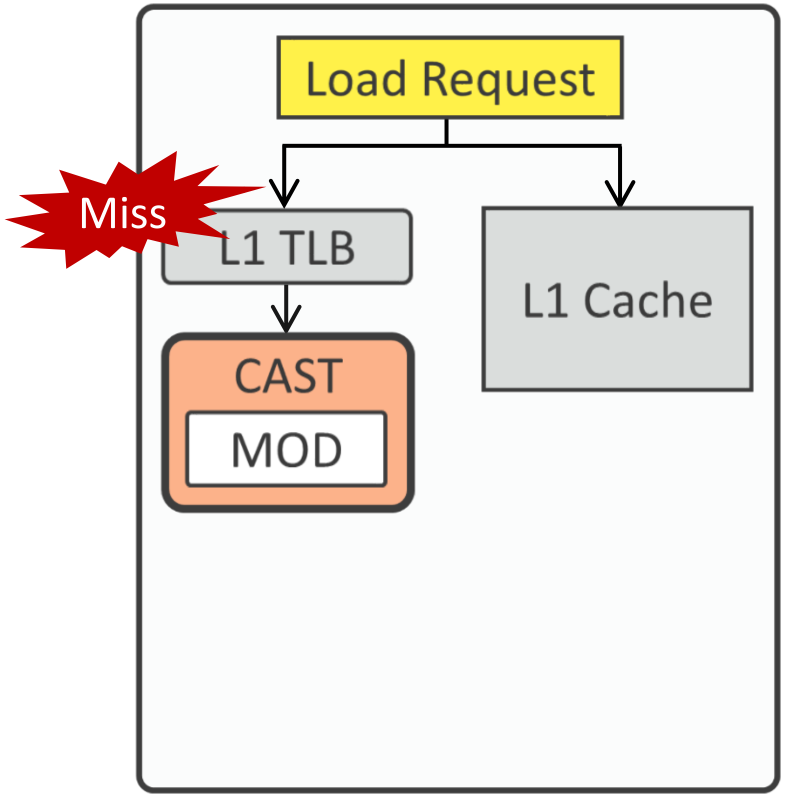
CAST: Contiguity-Aware Speculative Translation



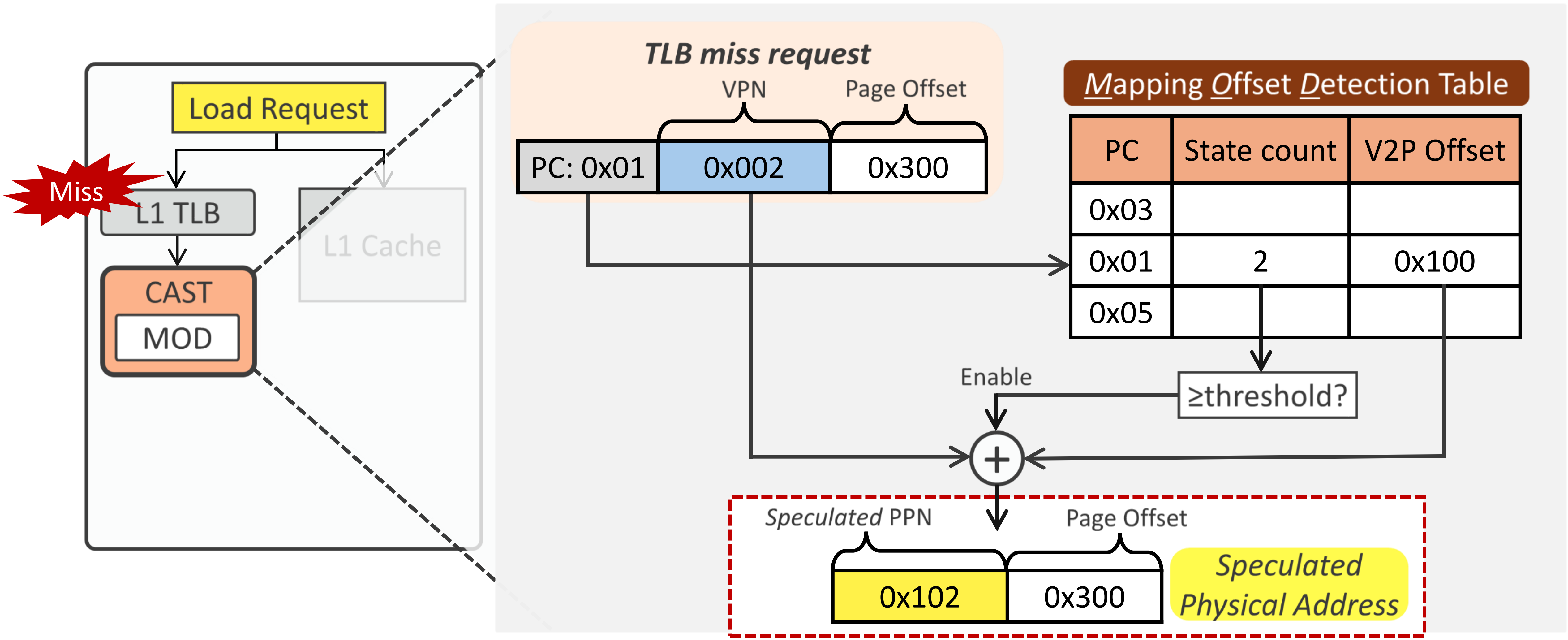
CAST: Contiguity-Aware Speculative Translation



CAST: Contiguity-Aware Speculative Translation



CAST: Contiguity-Aware Speculative Translation



Outline

1. Virtual Memory in GPUs – Challenges

2. Avatar
   - Speculative address translation
   - **Rapid Validation** ☜

3. Evaluation

4. Summary

Rapid Validation

How to validate rapidly?

❶ Prior Approach
- Lazy validation

❷ Our Approach
- Key observation
- Implementation: CAVA

Rapid Validation

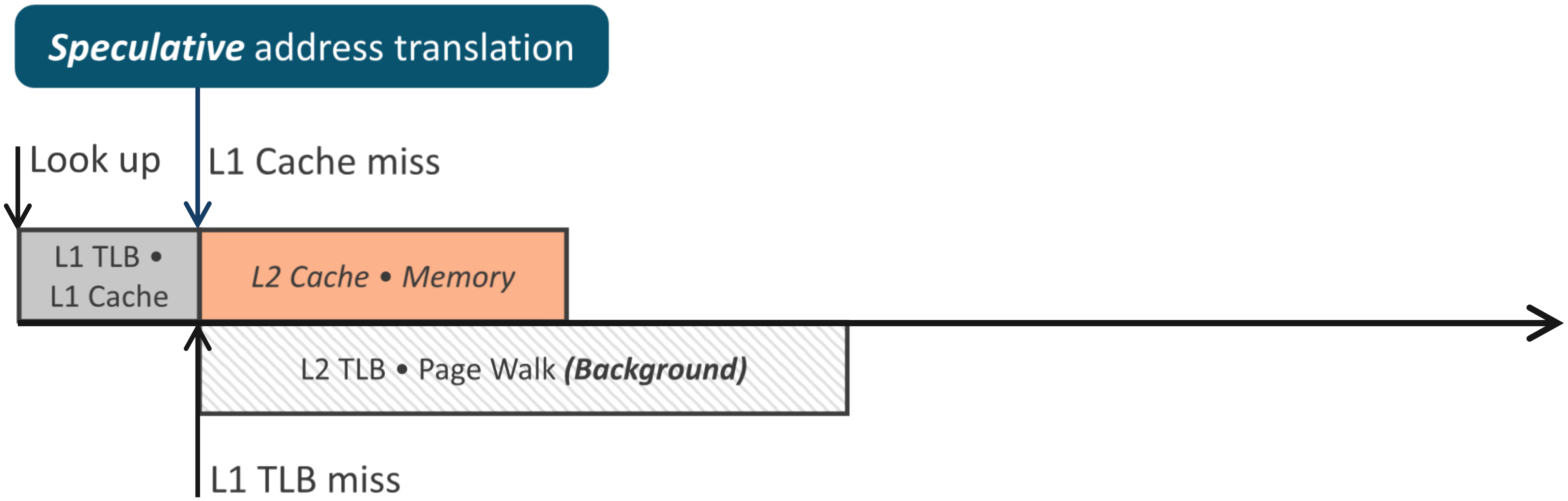
How to validate rapidly?

❶ Prior Approach
- Lazy validation

❷ Our Approach
- Key observation
- Implementation: CAVA

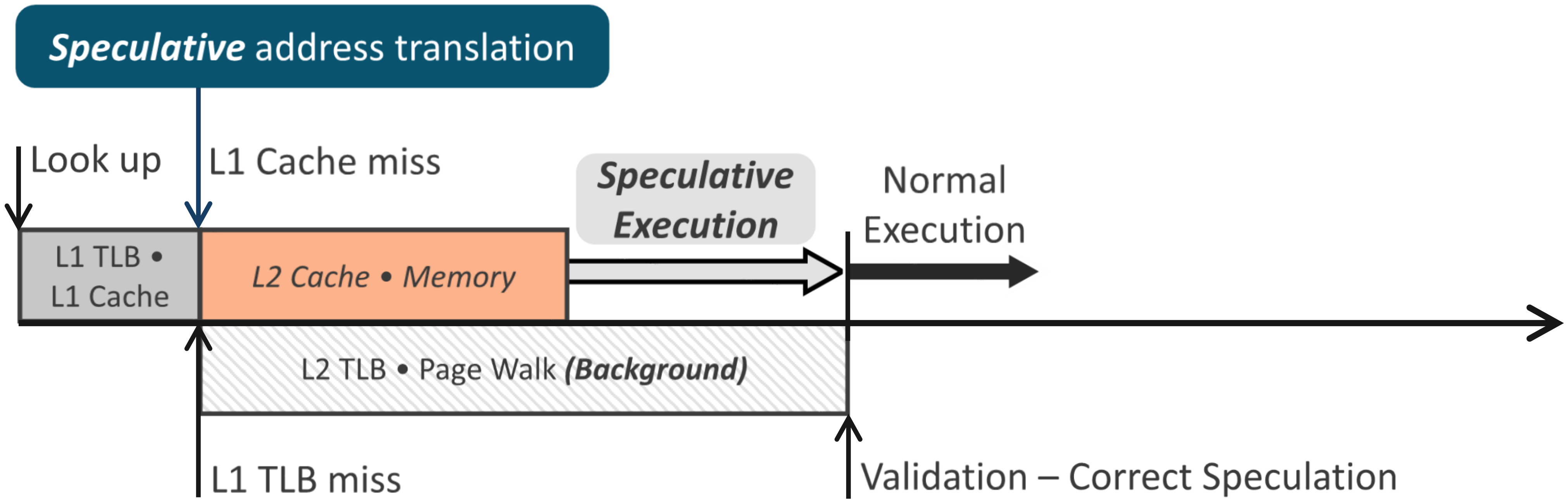
Limitation in Prior Approach

Lazy validation for speculative translation in CPUs

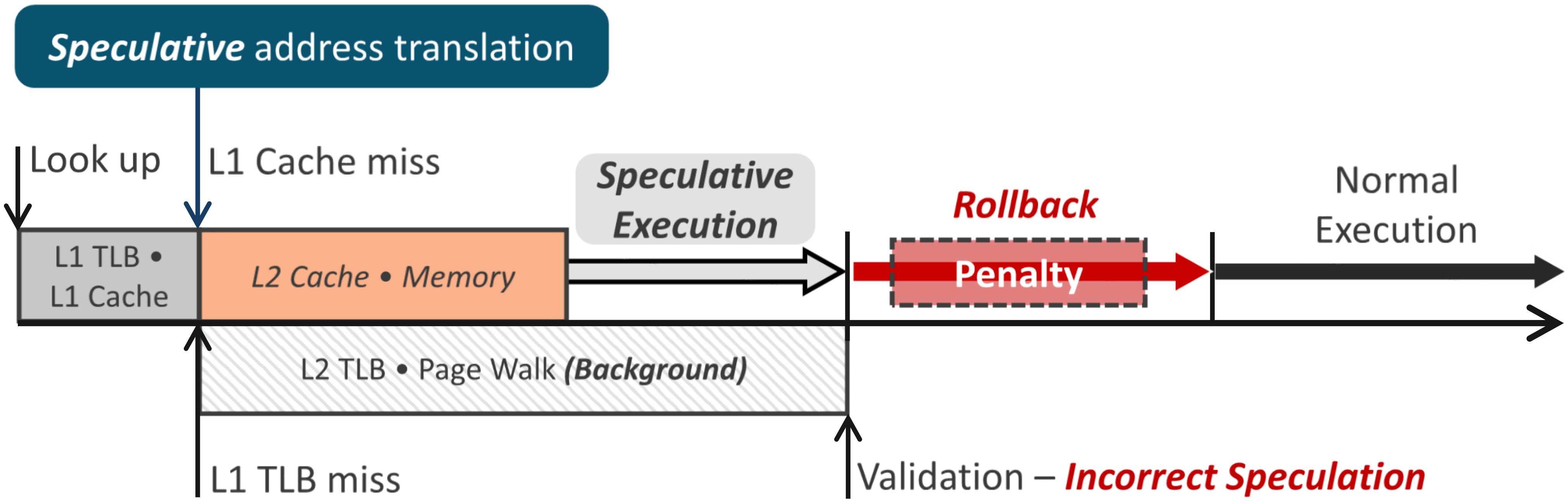


Limitation in Prior Approach

Lazy validation for speculative translation in CPUs

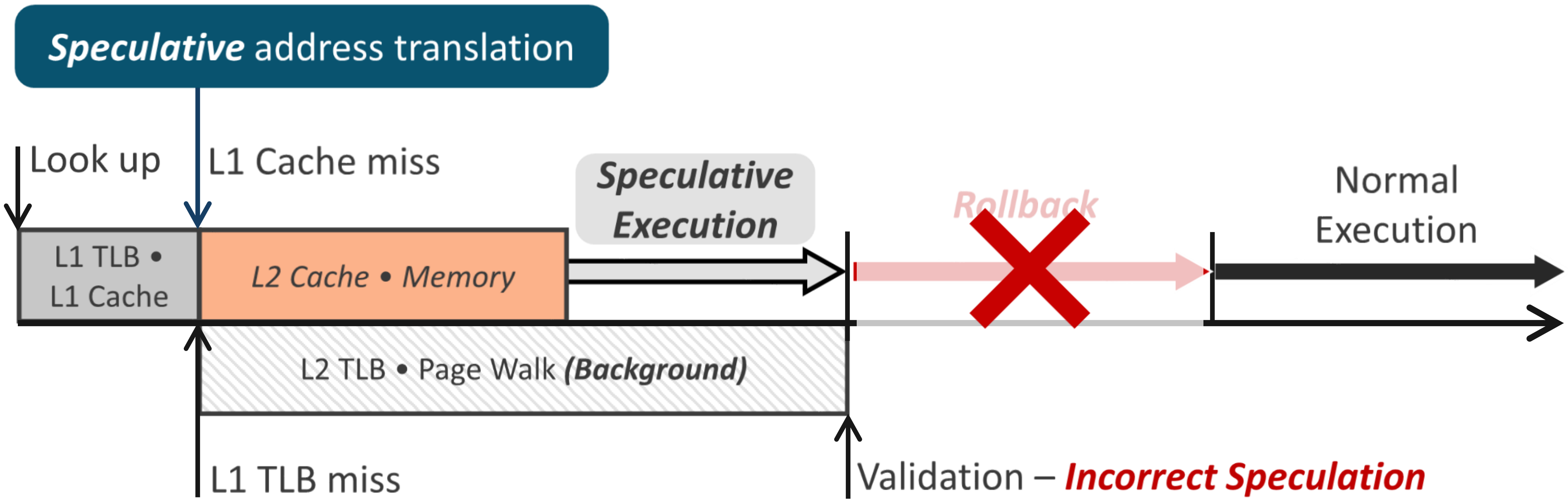


Limitation in Prior Approach



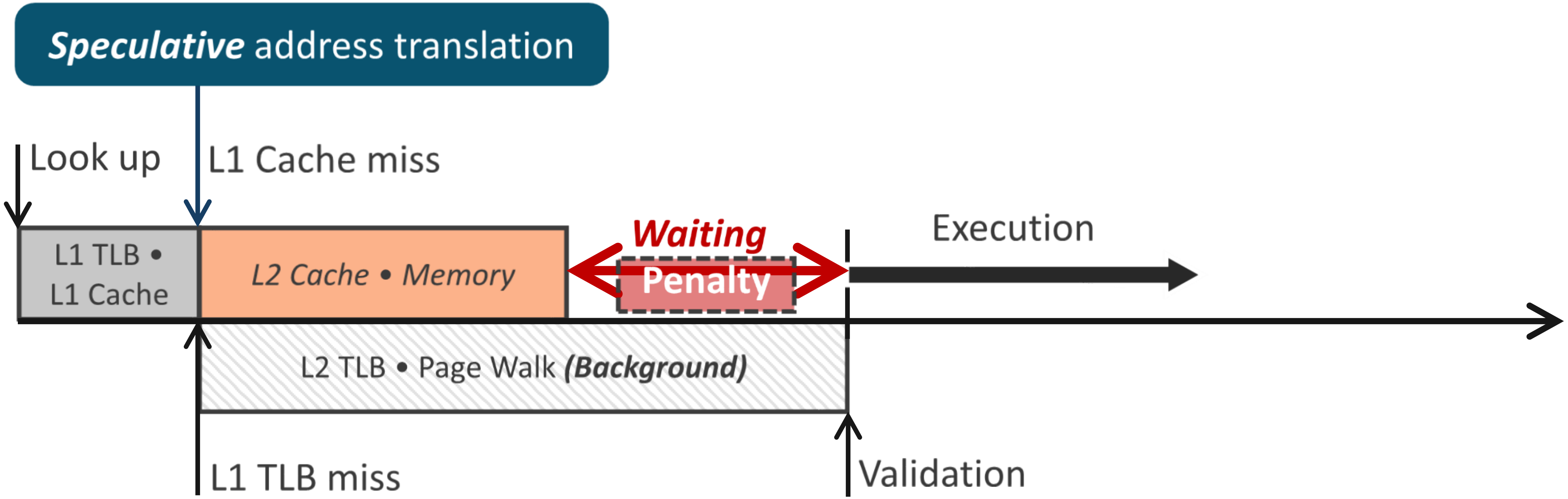
Limitation ❶: Expensive roll-back process

Limitation in Prior Approach



Limitation ❷: GPUs do not ***even*** support rollback mechanisms

Limitation in Prior Approach



Simple Solution: To wait for validation to complete

Rapid Validation

How to validate rapidly?

❶ Prior Approach
- Lazy validation

❷ Our Approach
- Key observation
- Implementation: CAVA

Rapid Validation

How to validate rapidly?

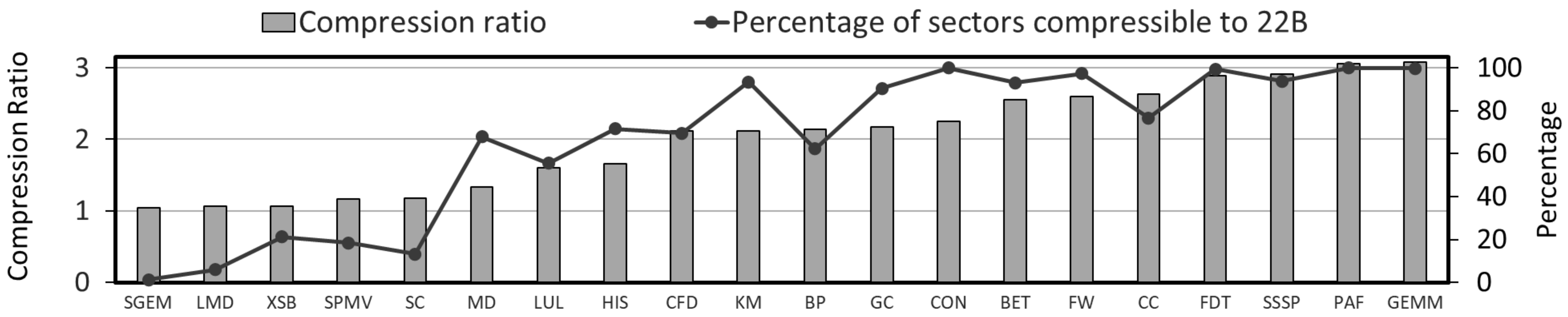
❶ Prior Approach
- Lazy validation

❷ Our Approach
- Key observation
- Implementation: CAVA

Key Observation – Memory Compression

Compresses **each 32-byte sector to 22 bytes** (BPC Compression [Kim+, ISCA'16])

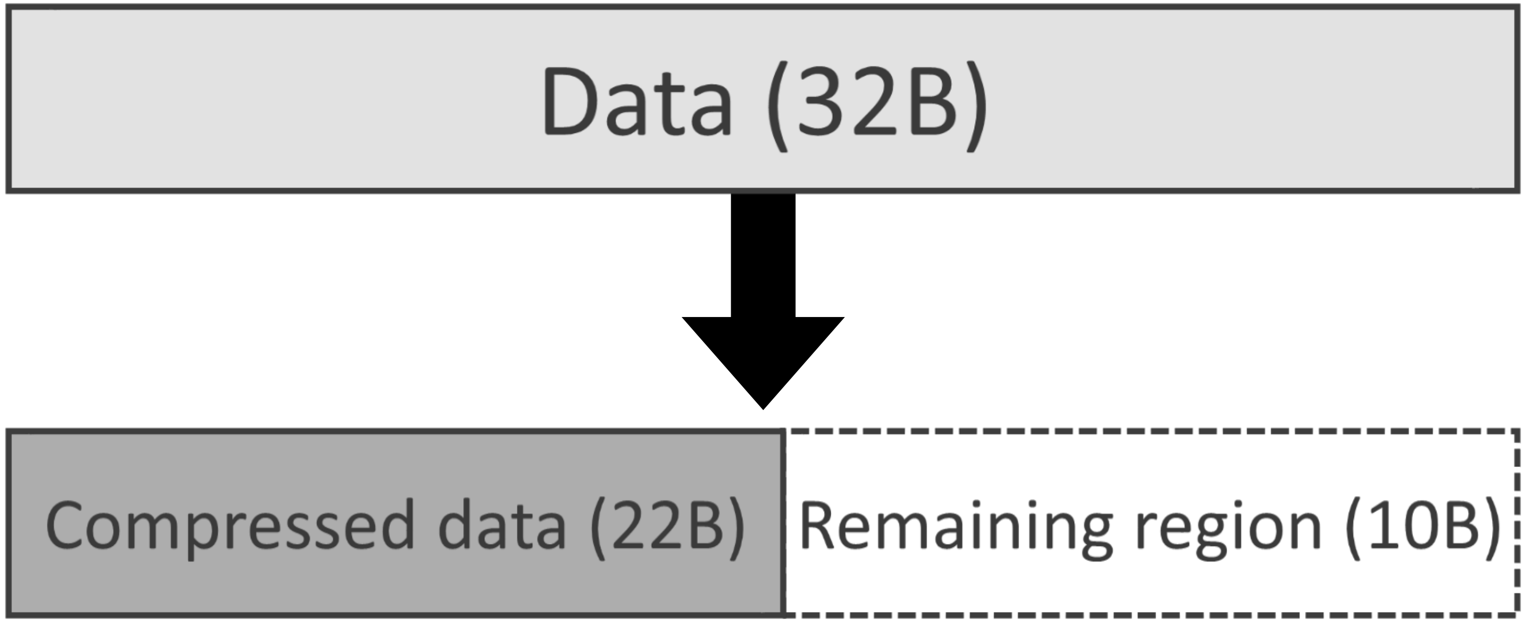
*sector: the basic data fetch unit in modern GPUs



On average, 67.5% of sectors can be compressed to 22 bytes

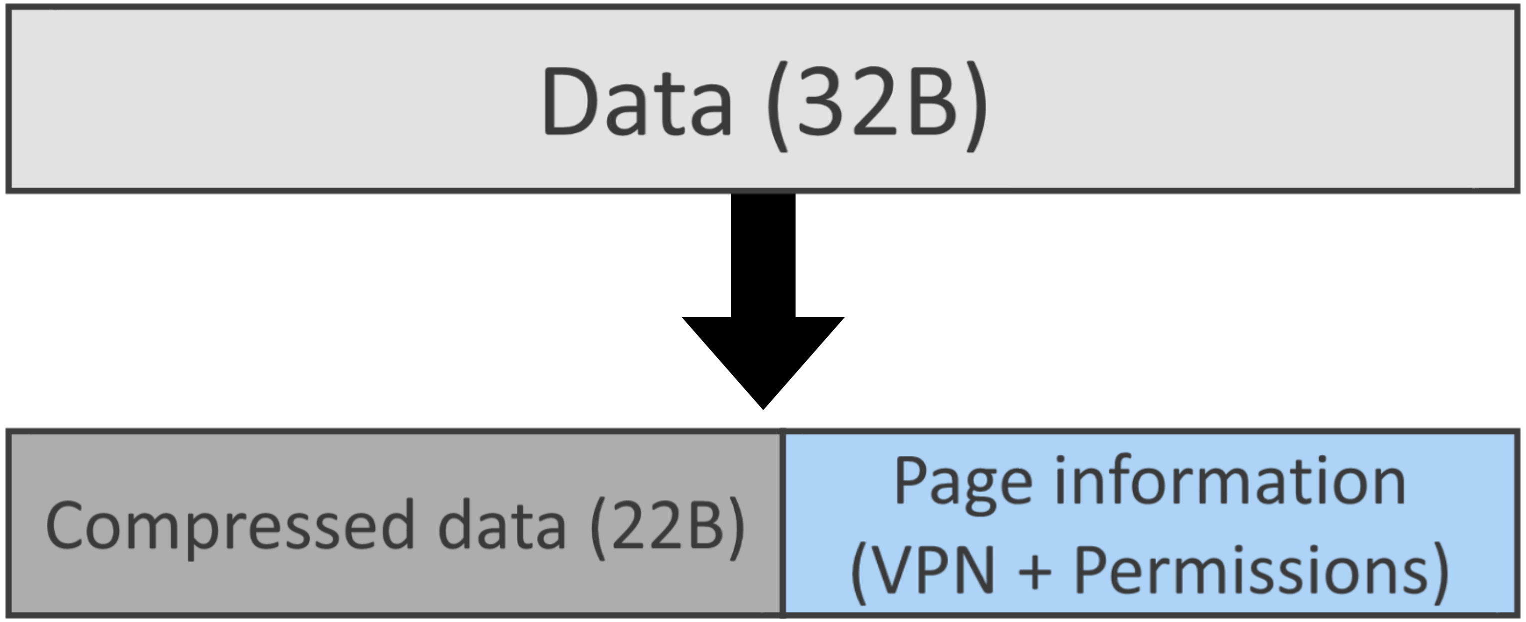
CAVA: In-Cache Validation

Embedding page information within data for validation

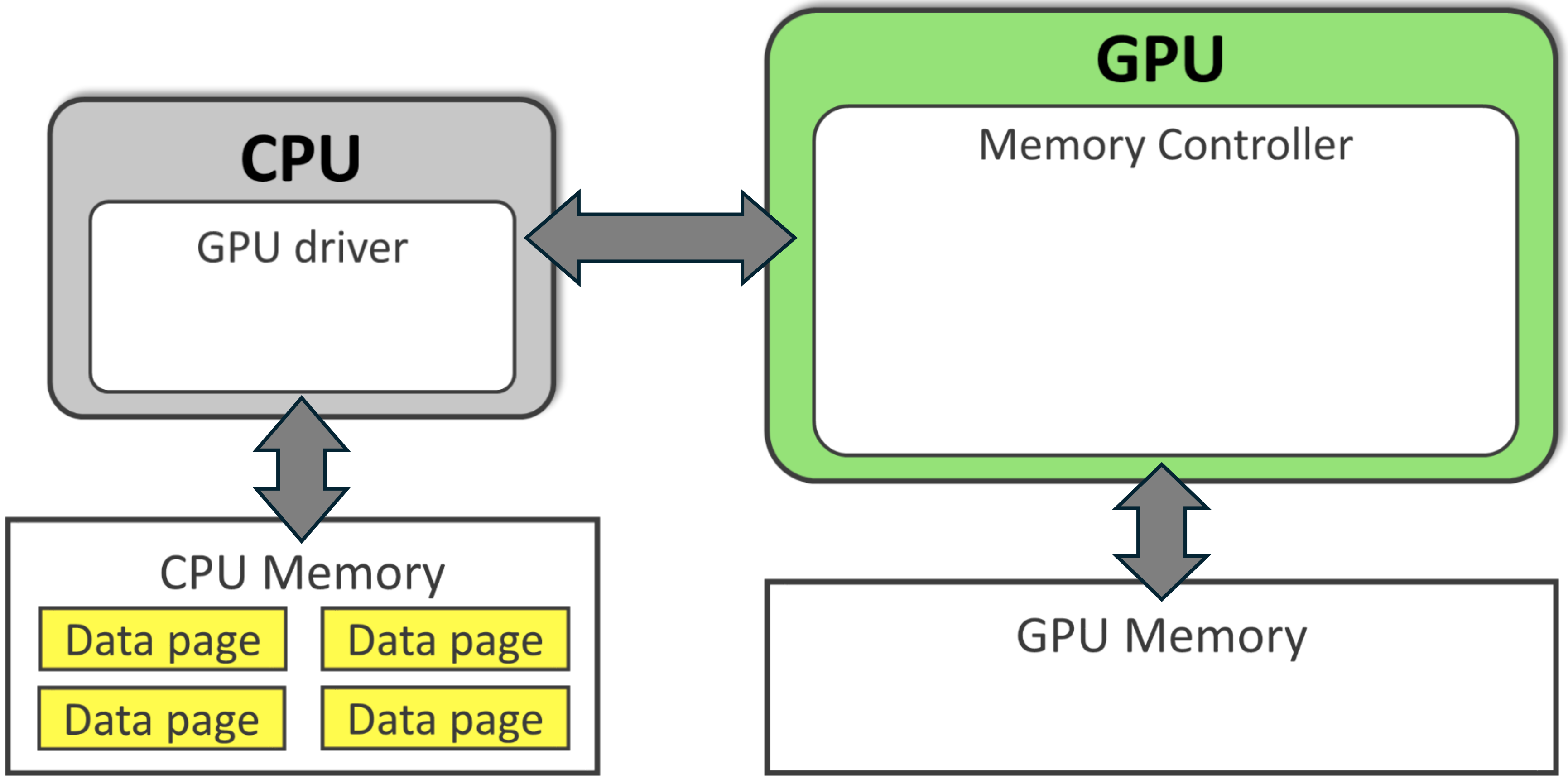


CAVA: In-Cache Validation

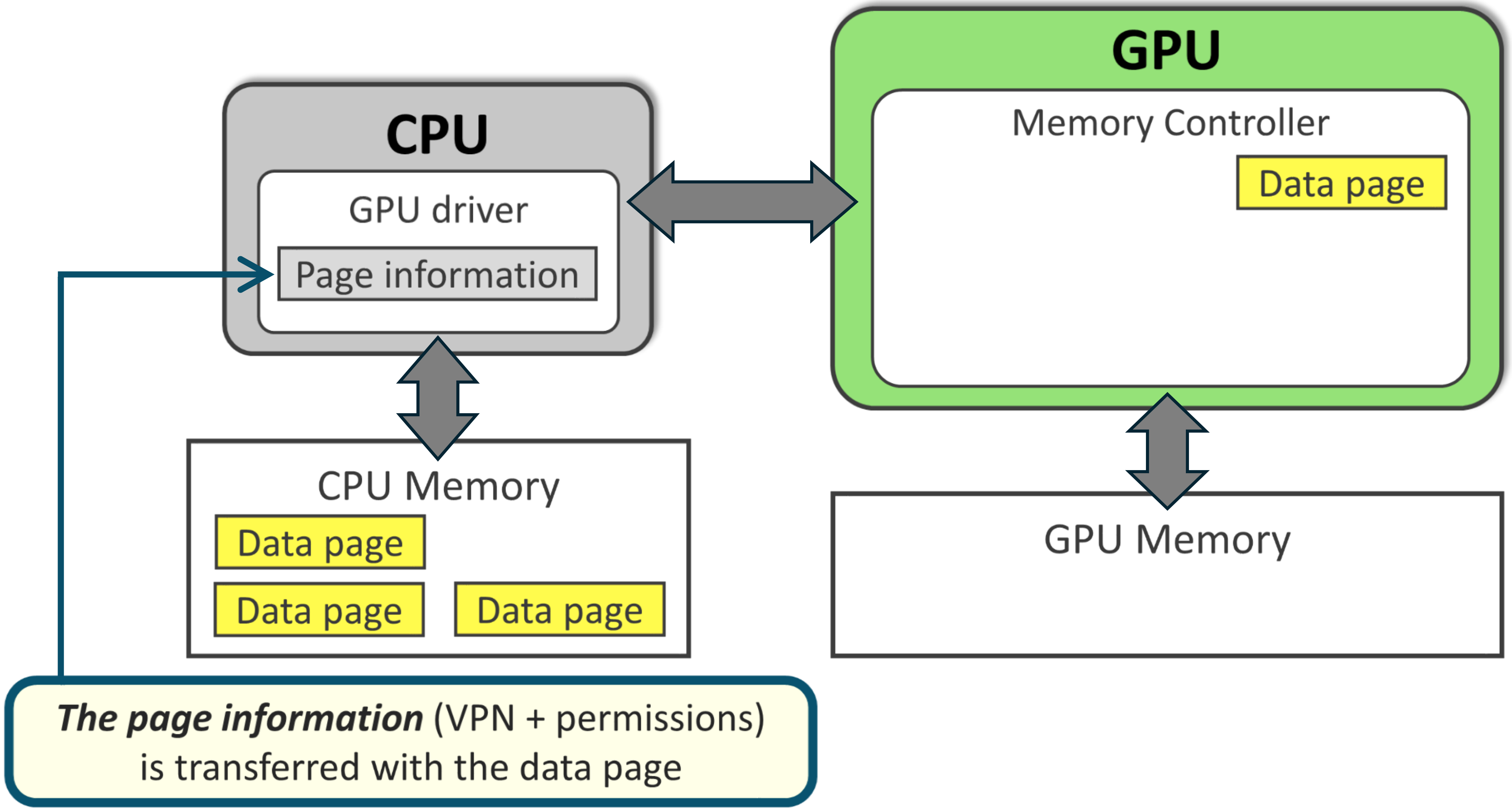
Embedding page information within data for validation



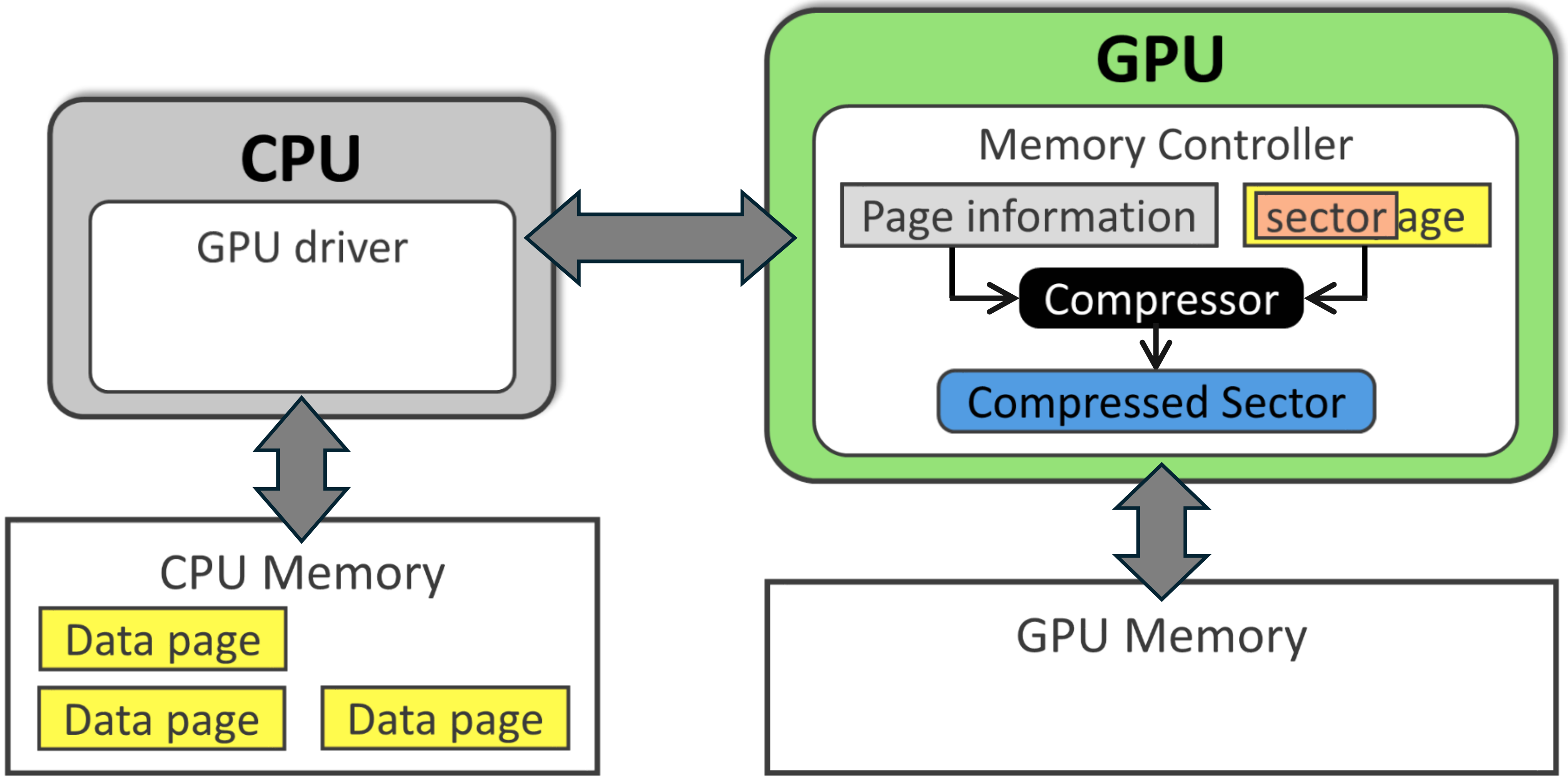
CAVA: In-Cache Validation



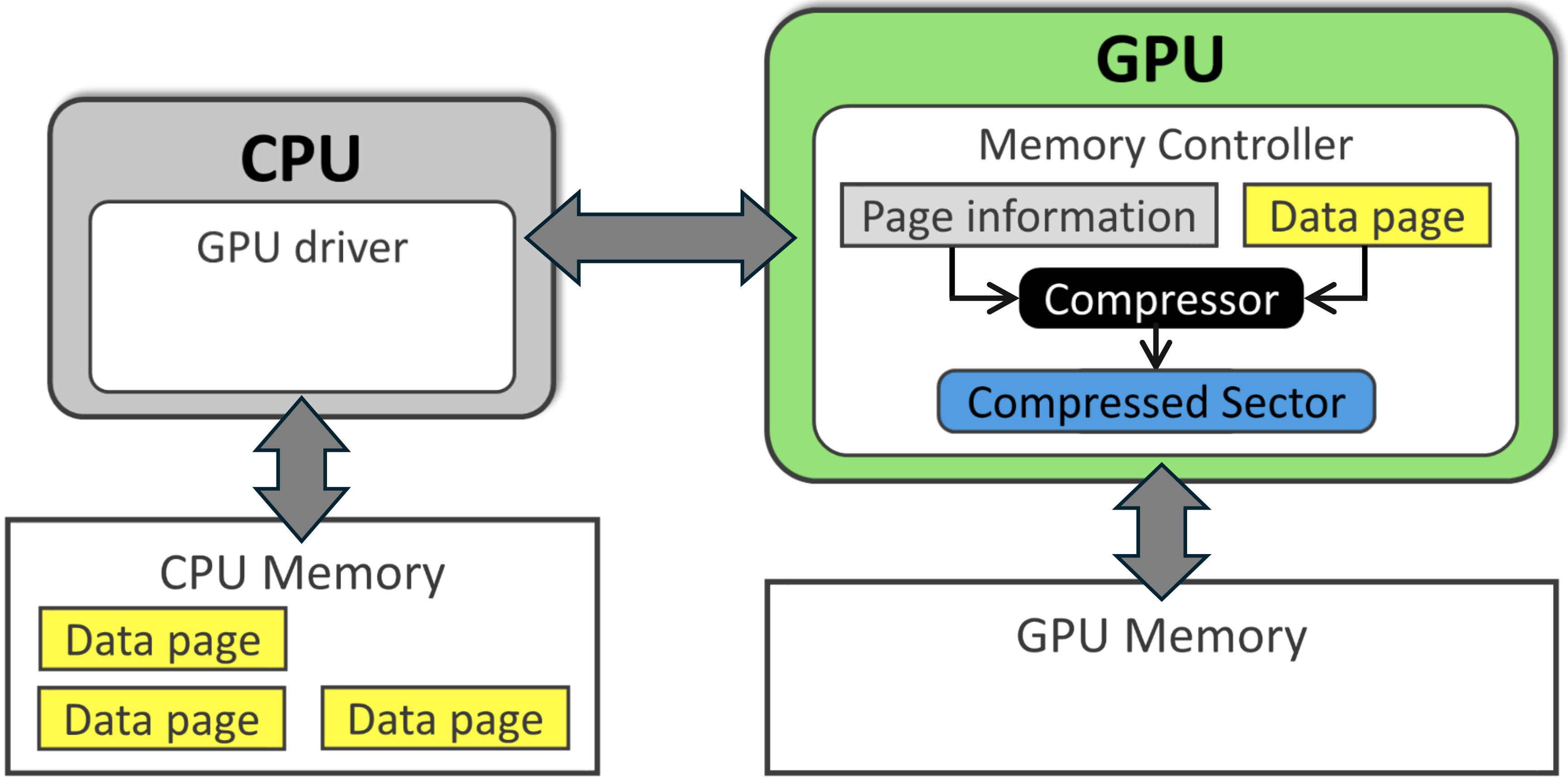
CAVA: In-Cache Validation



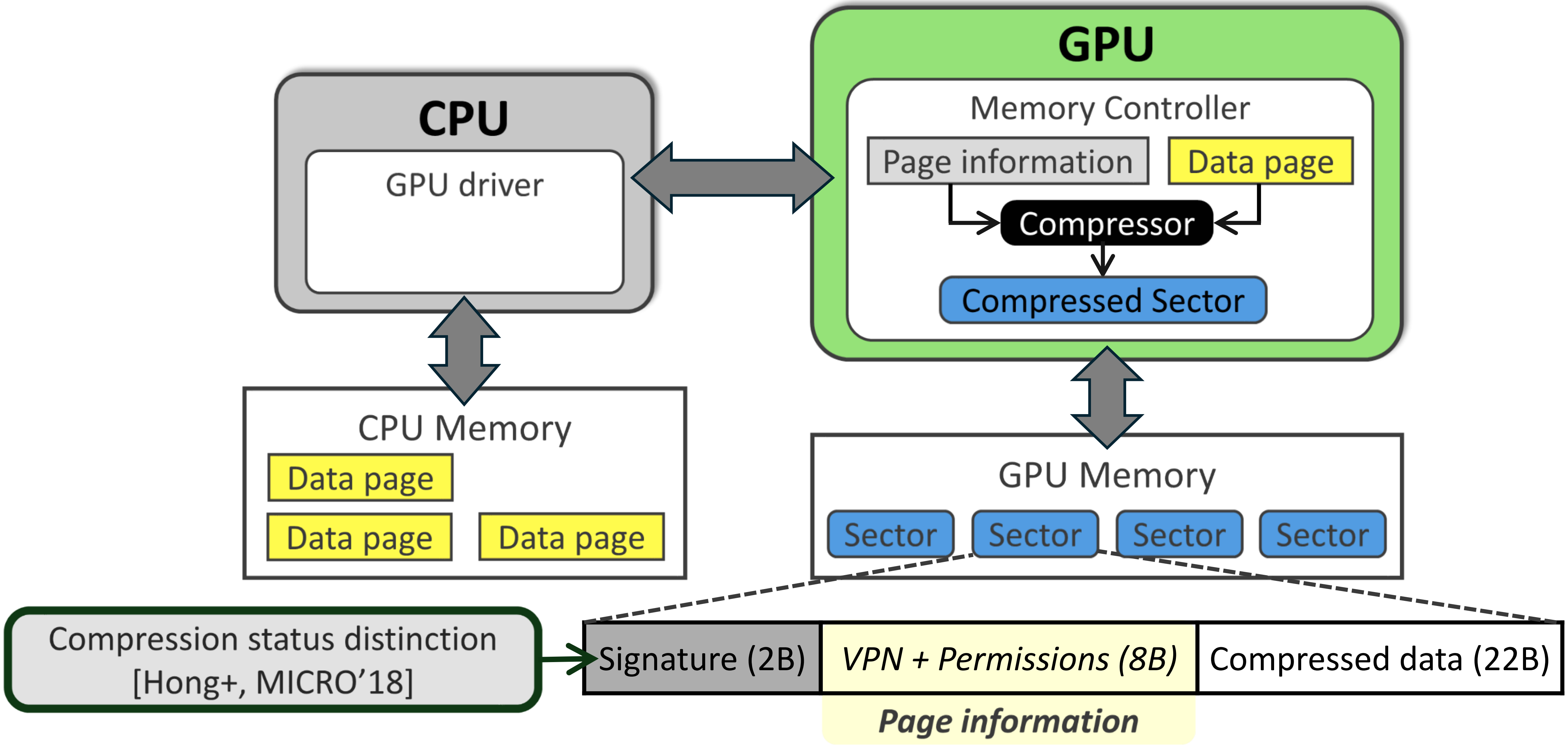
CAVA: In-Cache Validation



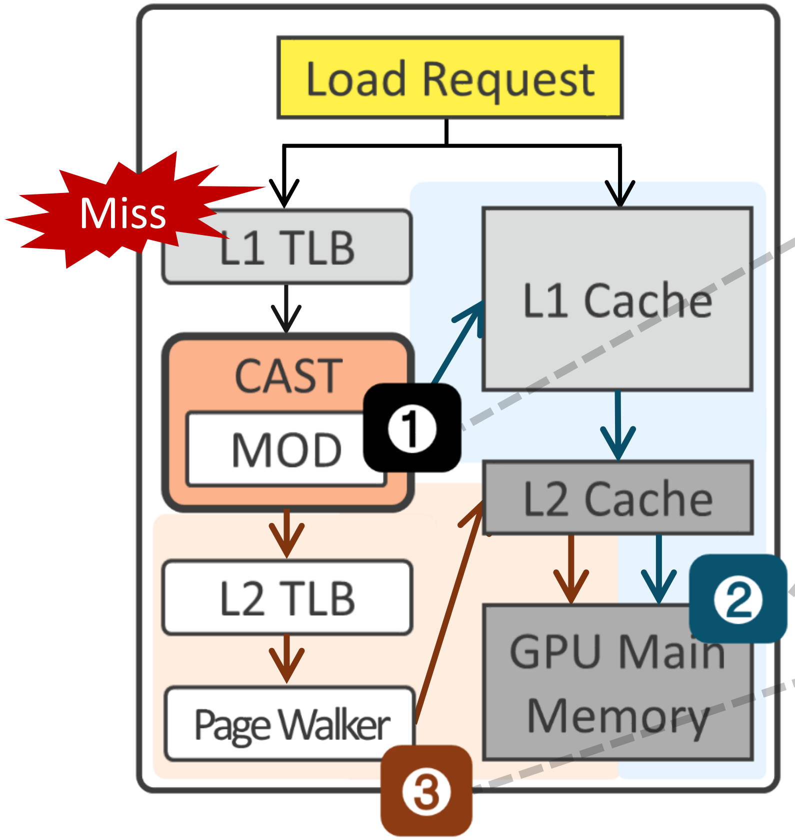
CAVA: In-Cache Validation



CAVA: In-Cache Validation



CAVA: In-Cache Validation

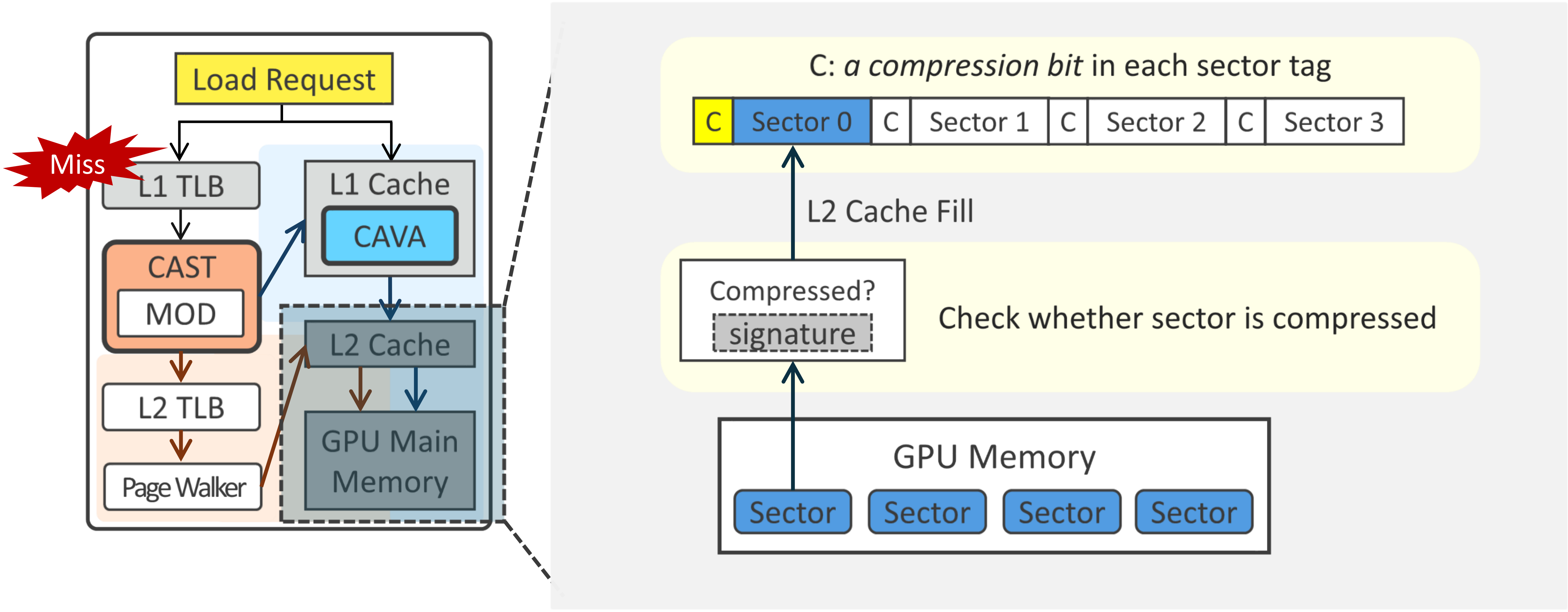


❶ ***Speculation with CAST***

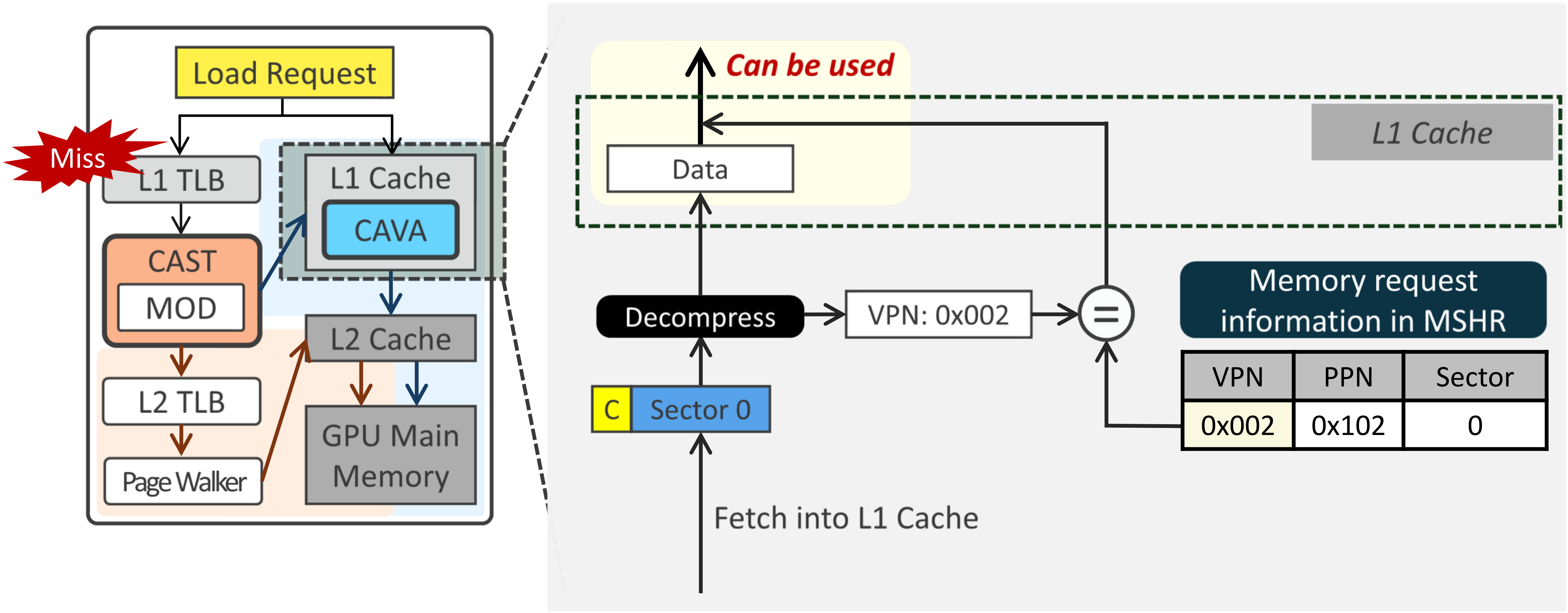
❷ Data fetched from the speculated address

❸ Address translation in the background

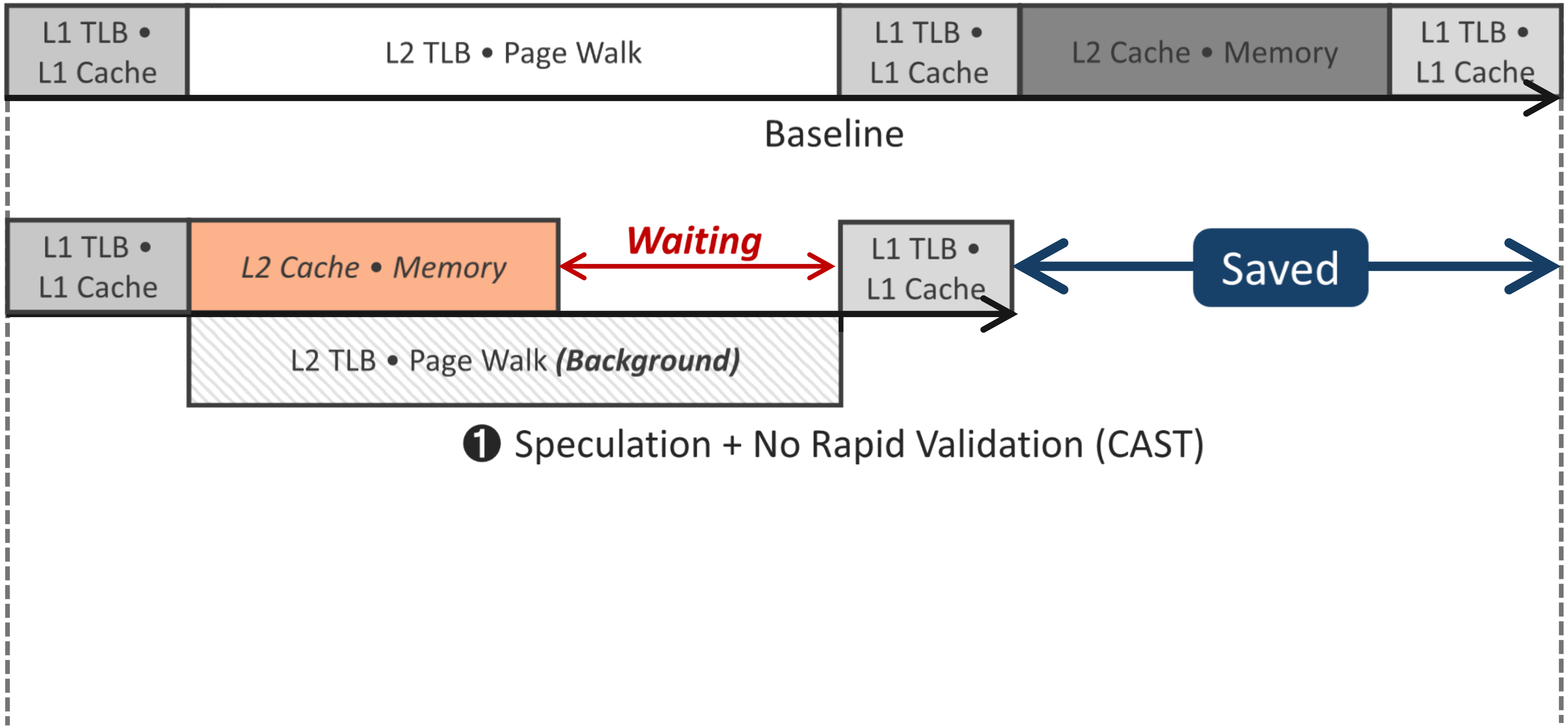
CAVA: In-Cache Validation



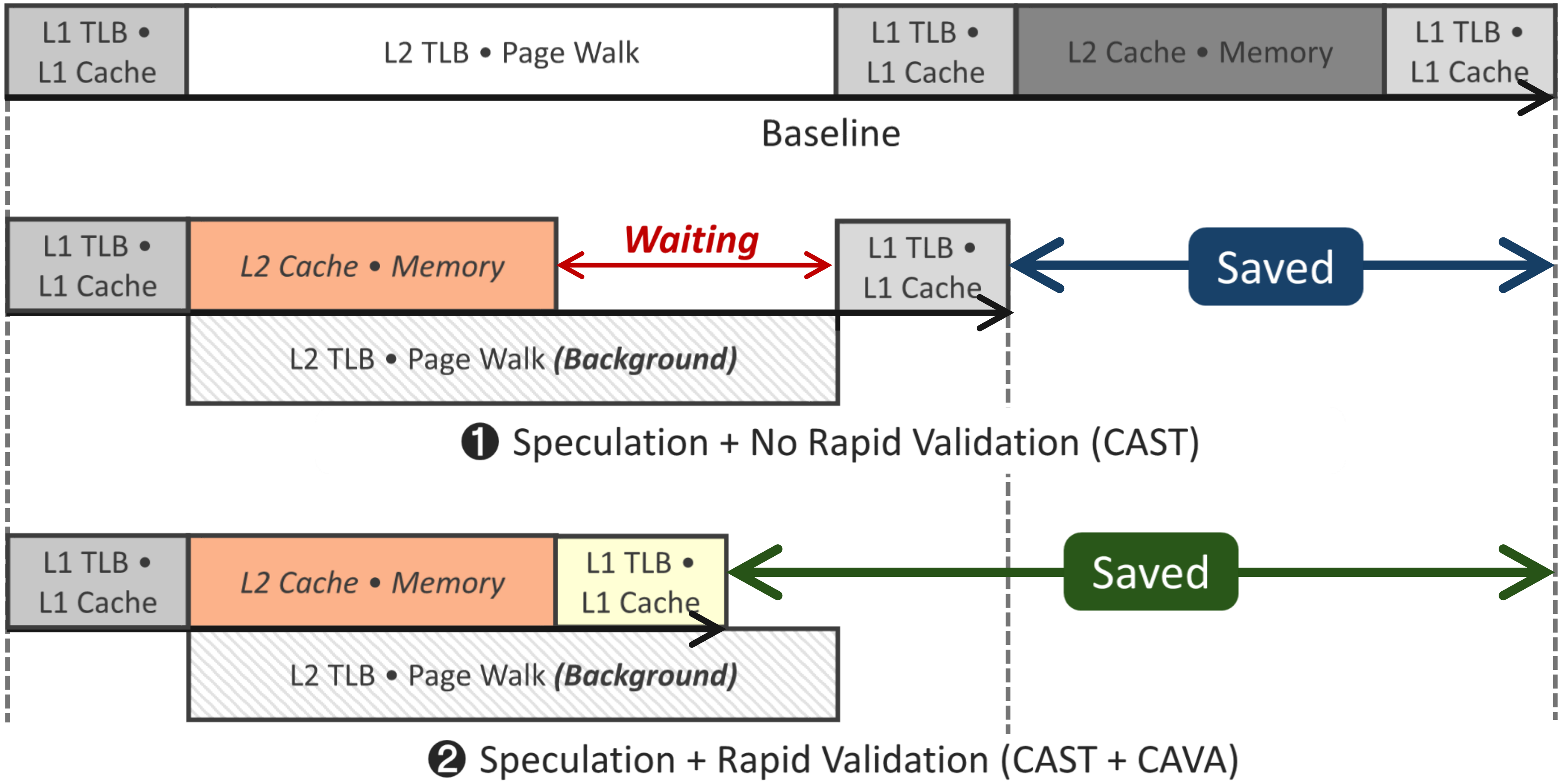
CAVA: In-Cache Validation



Timeline Comparison



Timeline Comparison



Outline

1. Virtual Memory in GPUs – Challenges

2. Avatar
    - Speculative address translation
    - Rapid Validation

3. **Evaluation** ☜

4. Summary

Methodology

Simulator: GPGPU-Sim 4.0 [Khairy+, ISCA’20]

- Virtual memory support (Unified Memory / 4KB page for baseline)

Component	Parameter
L1 TLB	32 entries (4KB), 16 entries* (2MB)
L2 TLB	1024 entries (4KB), 128 entries* (2MB)
Page walk	16 page walkers

* 2MB TLBs for page size promotion

Avatar Framework:

- 32-entry MOD table
- Data compression using the Bit-Plane Compression method [Kim+, ISCA’16]

Methodology

Workloads: 20 applications from various benchmark suites

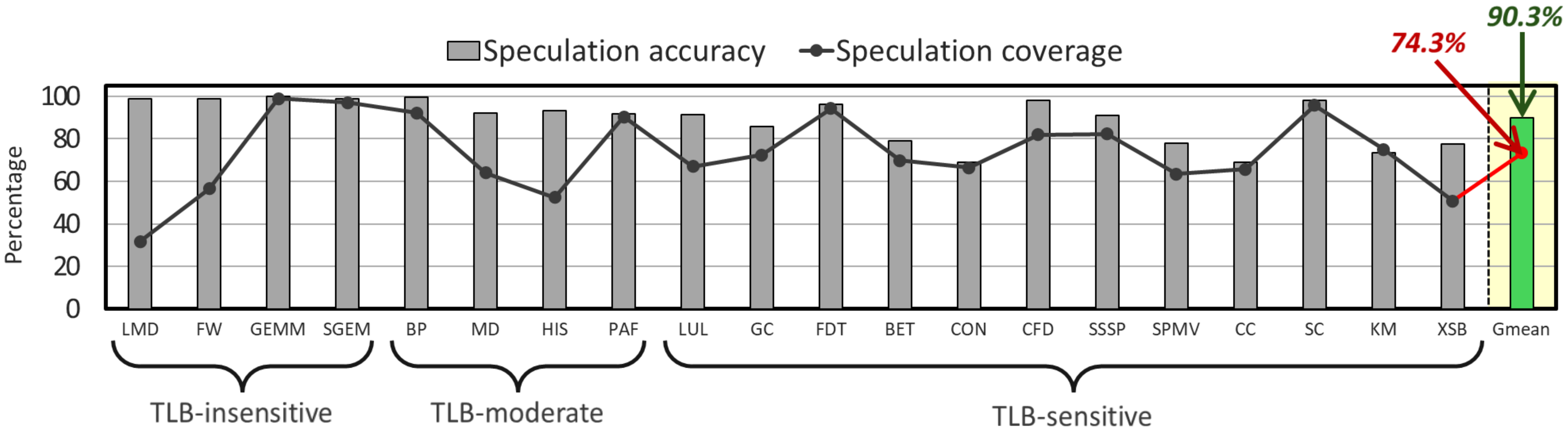
- Diverse data types
- Classified on TLB sensitivity

Comparison to Prior Works:

- *Promotion* [Ausavarungnirun+, MICRO’17]: Page size promotion
- *CoLT* [Pham+, MICRO’12]: Fixed-size TLB coalescing
- *SnakeByte* [Lee+, HPCA’23]: Multi-size TLB coalescing

Evaluation

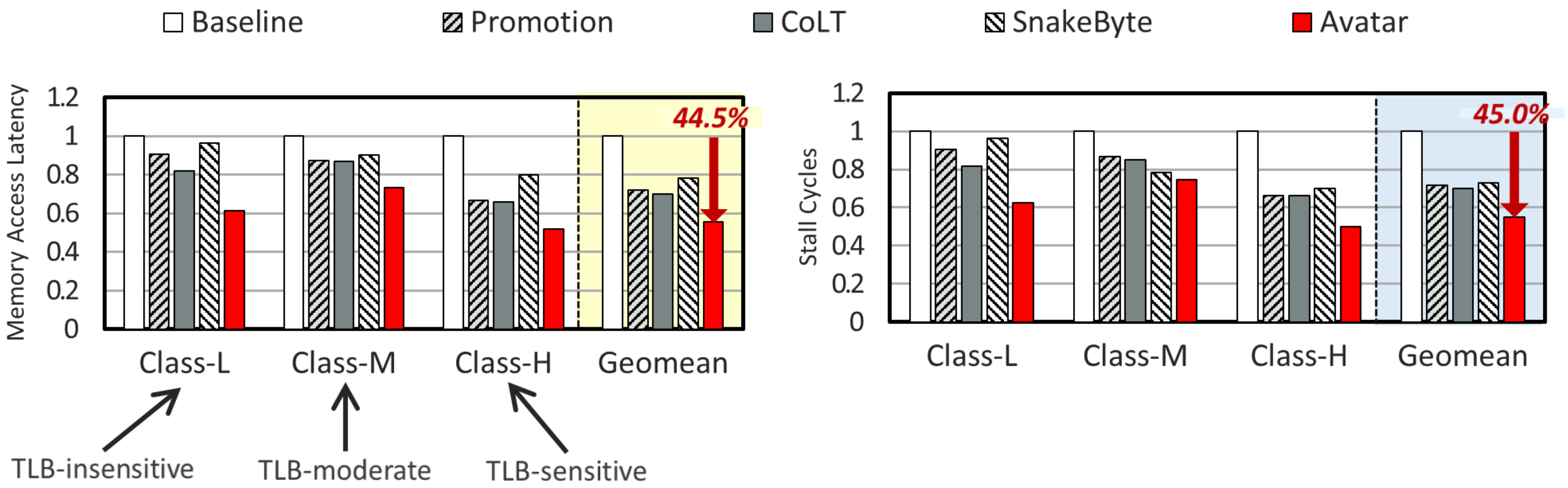
Speculation Accuracy - How many speculative translations are correct?
→ 90.3%
Speculation Coverage - How many TLB misses are handled by CAST?
→ 74.3%



Evaluation

Memory Access Latency Reduction
Avatar - ***44.5%↓***

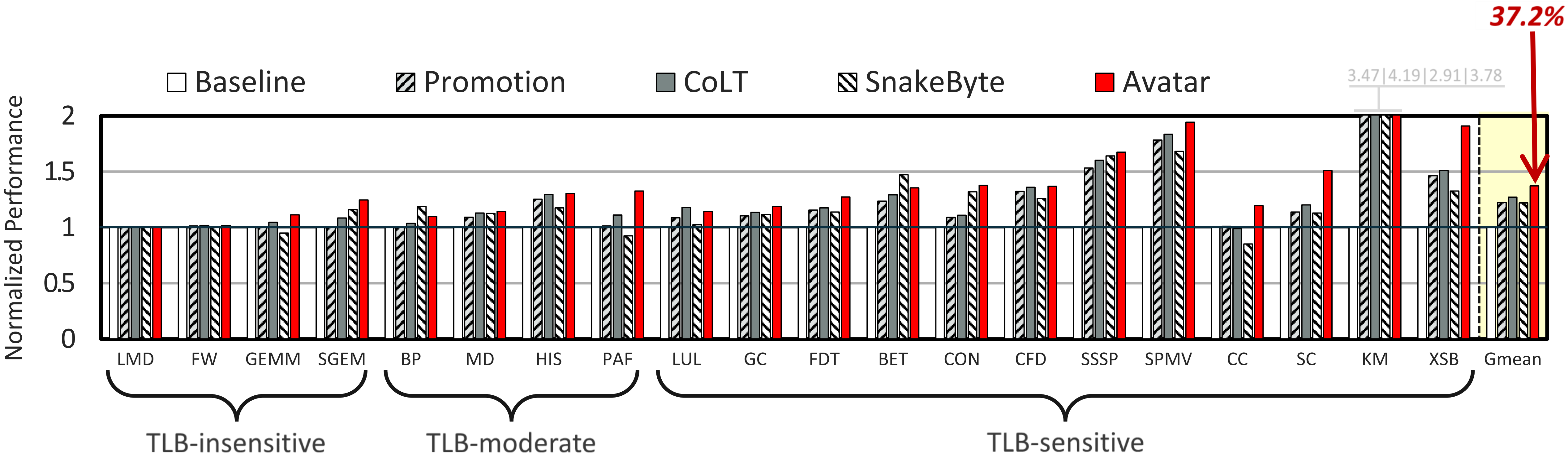
GPU Core Stall Cycles Reduction
Avatar - ***45.0%↓***



Evaluation

Performance Improvement

Avatar - ***37.2%↑*** > Best of prior works - 27.1%↑



Outline

1. Virtual Memory in GPUs – Challenges

2. Avatar
   - Speculative address translation
   - Rapid Validation

3. Evaluation

4. Summary ☜

Summary

Problem: Address translation overhead in GPUs (34.5% slower compared to Ideal)

Summary

Problem: Address translation overhead in GPUs (34.5% slower compared to Ideal)

Avatar: Speculative address translation ***overcoming GPUs' architectural limit***

- ***Speculative translation*** by exploiting page contiguity
- ***Rapid validation*** using page information embedded within data blocks

Summary

Problem: Address translation overhead in GPUs (34.5% slower compared to Ideal)

Avatar: Speculative address translation ***overcoming GPUs' architectural limit***

- ***Speculative translation*** by exploiting page contiguity
- ***Rapid validation*** using page information embedded within data blocks

Avatar achieves an average speed up of 37.2% across various workloads

A Case for Speculative Address Translation with Rapid Validation for GPUs

Thanks!

Junhyeok Park[1] Osang Kwon[1] Yongho Lee[1]

Seongwook Kim[1] Gwangeun Byeon[1] Jihun Yoon[1]

Prashant J. Nair[2] Seokin Hong[1]

1 SUNG KYUN KWAN UNIVERSITY

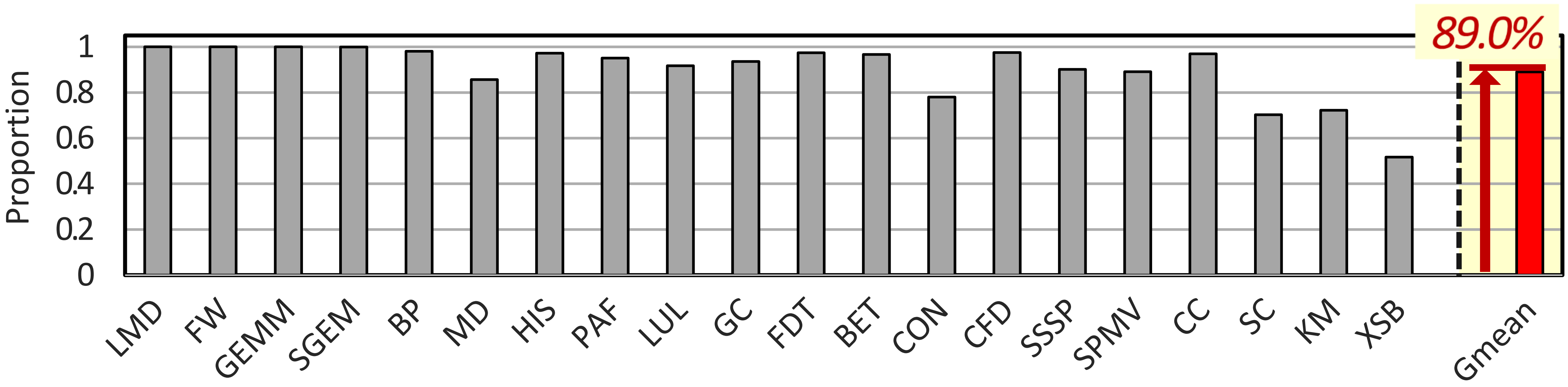
2 THE UNIVERSITY OF BRITISH COLUMBIA

Backup Slides

- Page Access Pattern of Load
- Without Rapid Validation
- ML Workloads
- Security Implications
- TLB Shootdown
- Workload Categorization

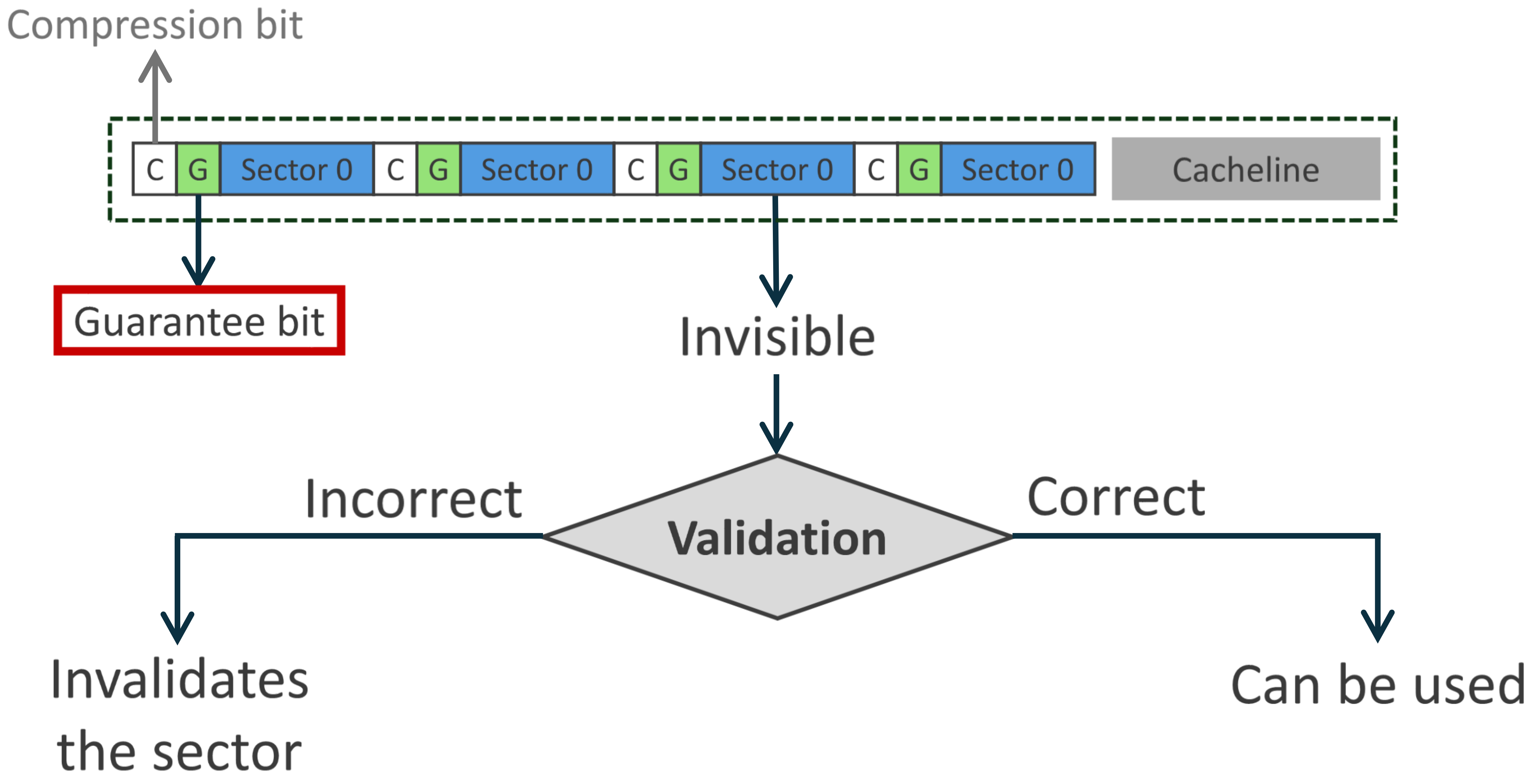
Page Access Pattern of Load

On average, 89.0% of memory accesses from the same instruction fall within a 2MB memory chunk.



- Backup Slides

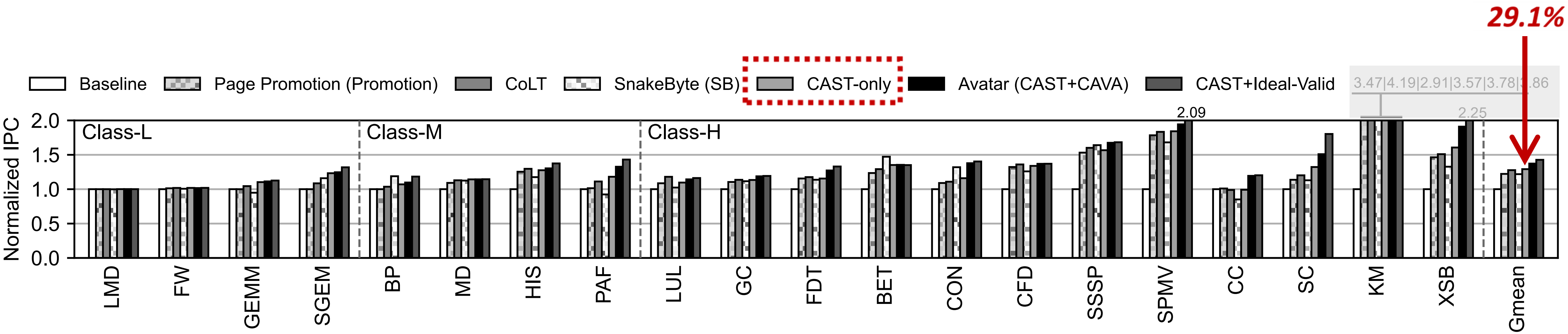
Security Implications



- Backup Slides

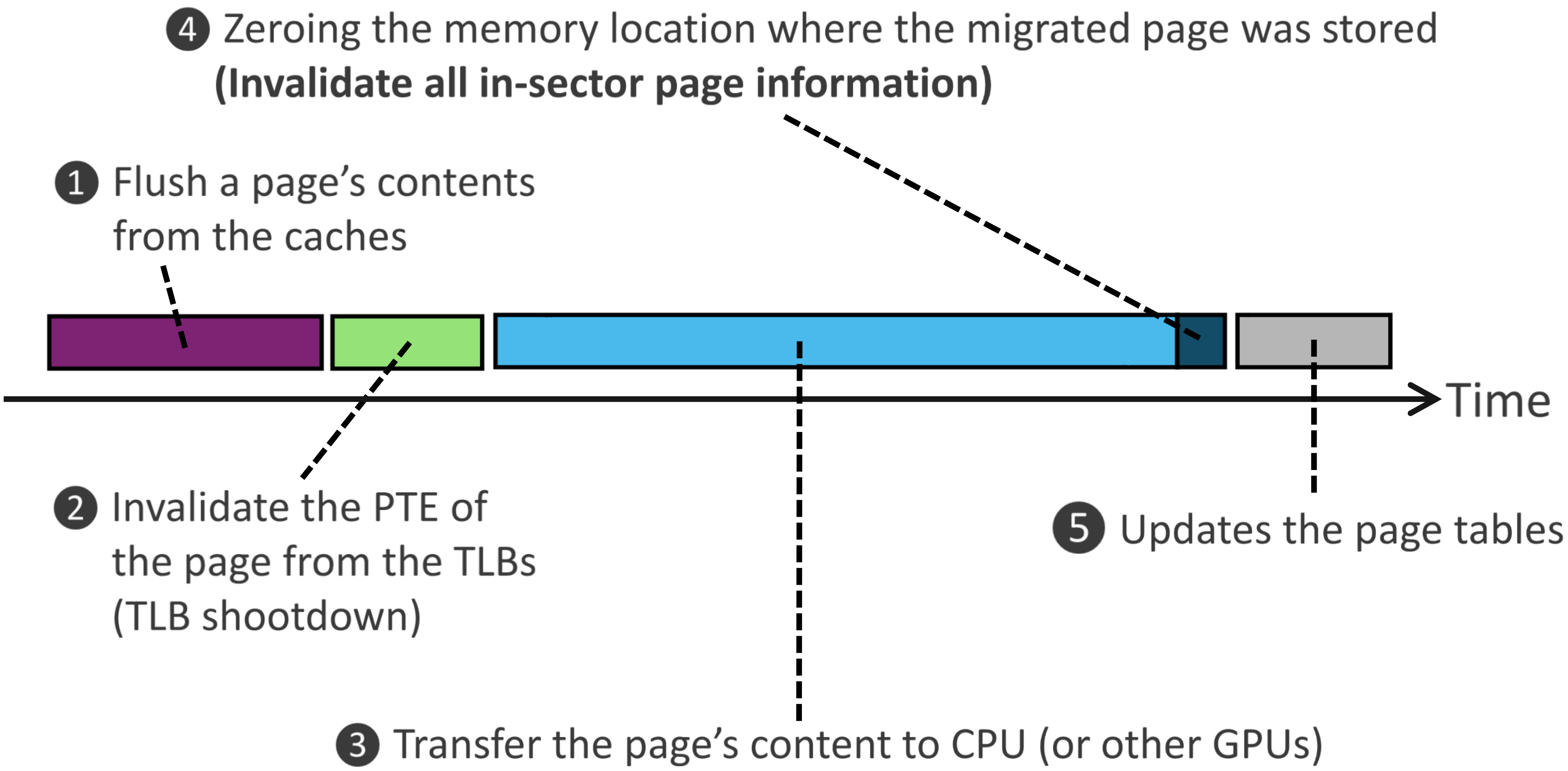
Without Rapid Validation

CAST-only (without rapid validation) achieves **a 29.1% speedup.**



• Backup Slides

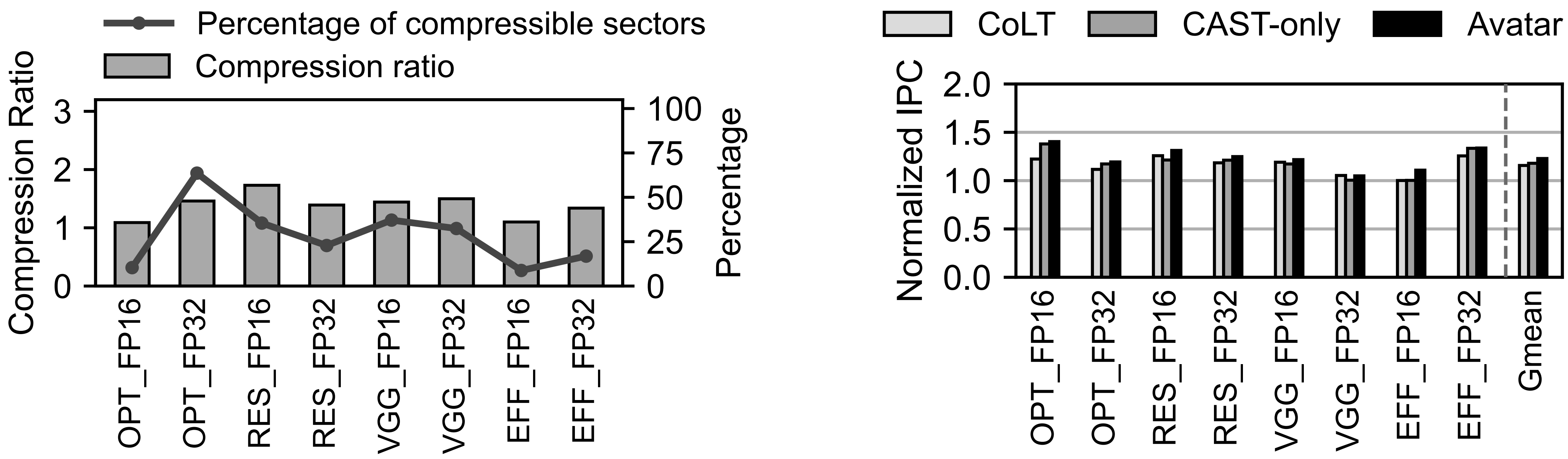
TLB Shootdown



• Backup Slides

ML Workloads

Avatar outperforms **the best prior work** by **7.1% on average.**



(a) Compressibility

(b) Performance

• Backup Slides

Workload Categorization

Class	Benchmark	Abbr.	Data Type	Benchmark	Abbr.	Data Type
L	fw	FW	**I**	lavaMD	LMD	**D**
	gemm	GEMM	**F**	sgemm	SGEM	**F**
M	backprop	BP	**F**	shoc-MD	MD	**I**
	histogram	HIS	**UI**	pathfinder	PAF	**I**
H	lulesh	LUL	**F**	color-max	GC	**I**
	fdtd2d	FDT	**F**	betweenness	BET	**UI**
	convolution	CON	**F**	cfd	CFD	**F**
	sssp	SSSP	**I**	spmv	SPMV	**I/F**
	connected	CC	**UI**	s.cluster	SC	**F**
	kmeans	KM	**F**	XSBench	XSB	**I/D**

* Denoted as I/UI/F/D for int/unsigned int/float/double

• Backup Slides