GPHash: An Efficient Hash Index for GPU with Byte-Granularity Persistent Memory

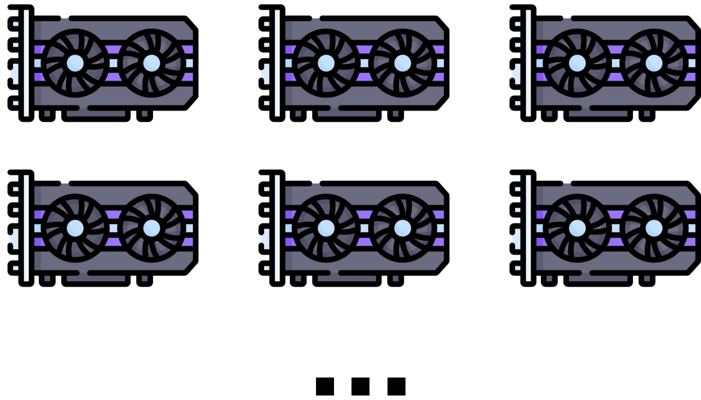
Menglei Chen, Yu Hua, Zhangyu Chen, Ming Zhang, Gen Dong

Huazhong University of Science and Technology, China

23rd USENIX Conference on File and Storage Technologies (**FAST**), 2025

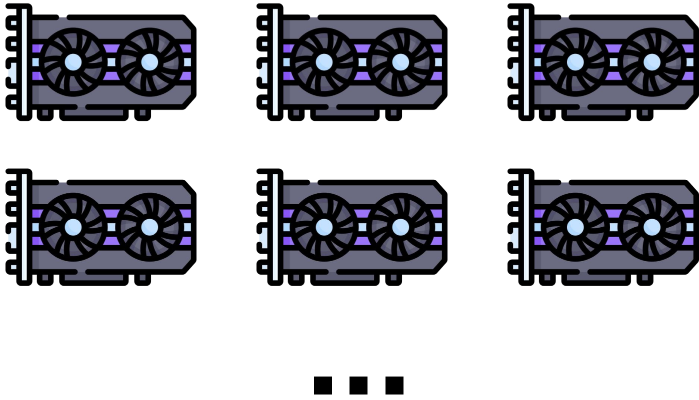
Various Applications Are Powered by GPUs

Various Applications Are Powered by GPUs



Various Applications Are Powered by GPUs

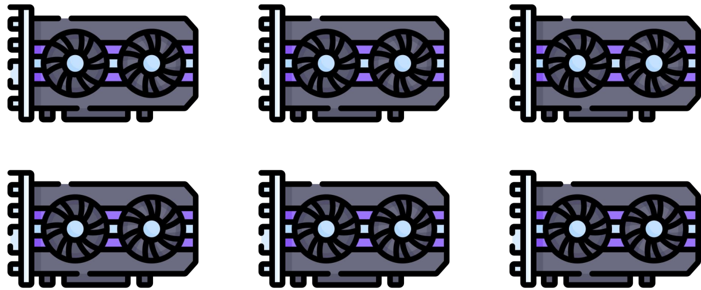
GPUs



Enhance

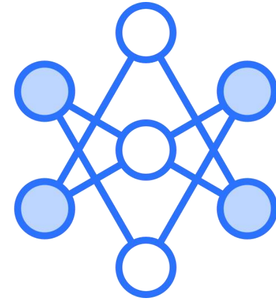
Various Applications Are Powered by GPUs

GPUs

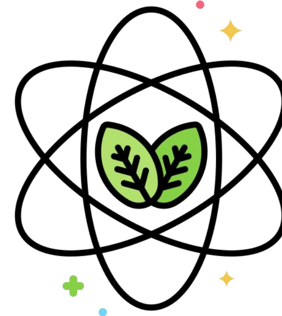


...

Enhance



Deep Neural Network

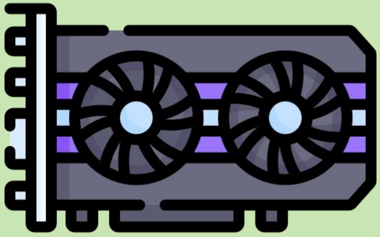


Scientific Computing

Autonomous driving

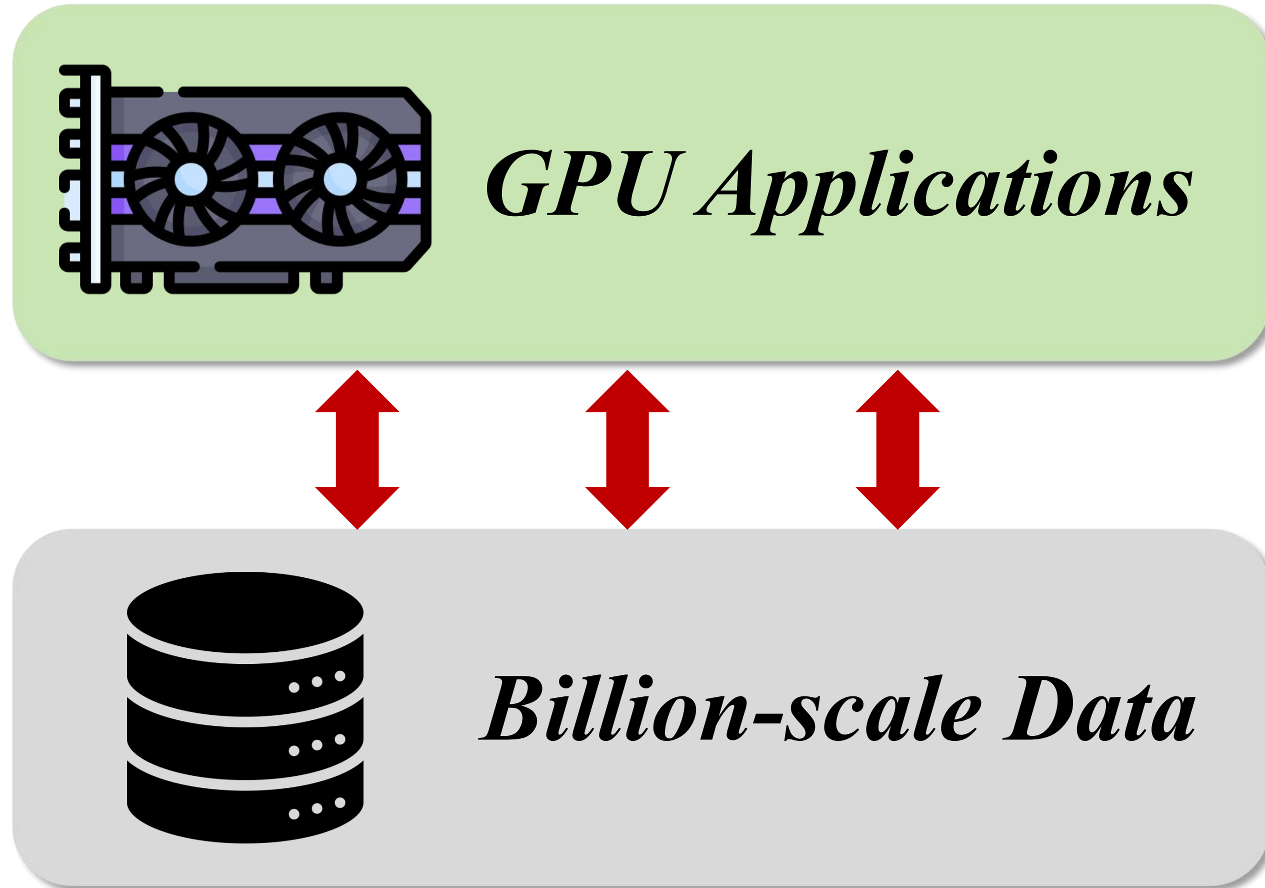
Unprecedented Data Scale of GPU App.

Unprecedented Data Scale of GPU App.

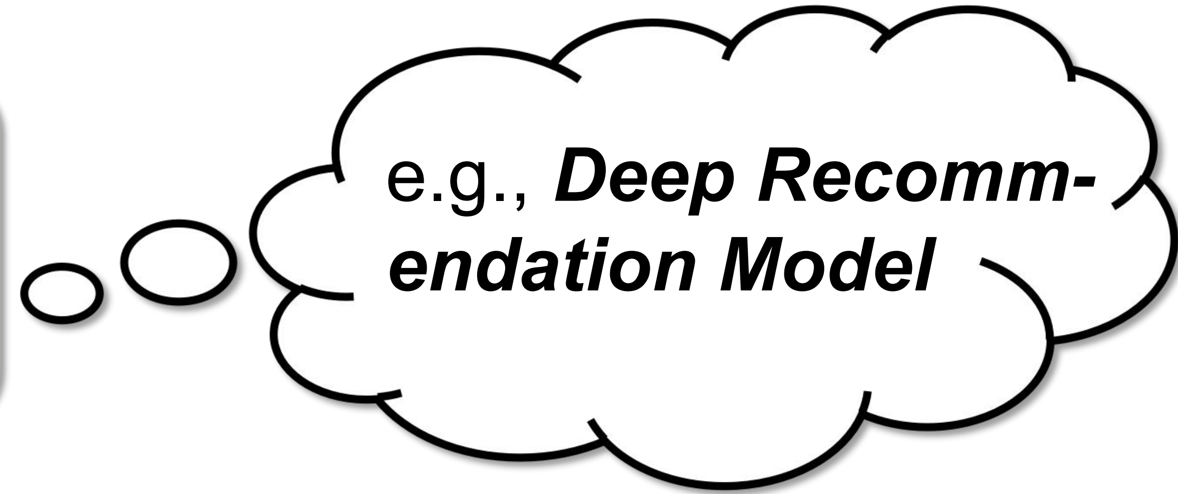


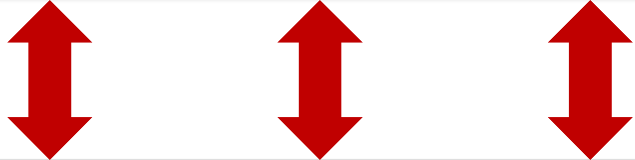
GPU Applications

Unprecedented Data Scale of GPU App.

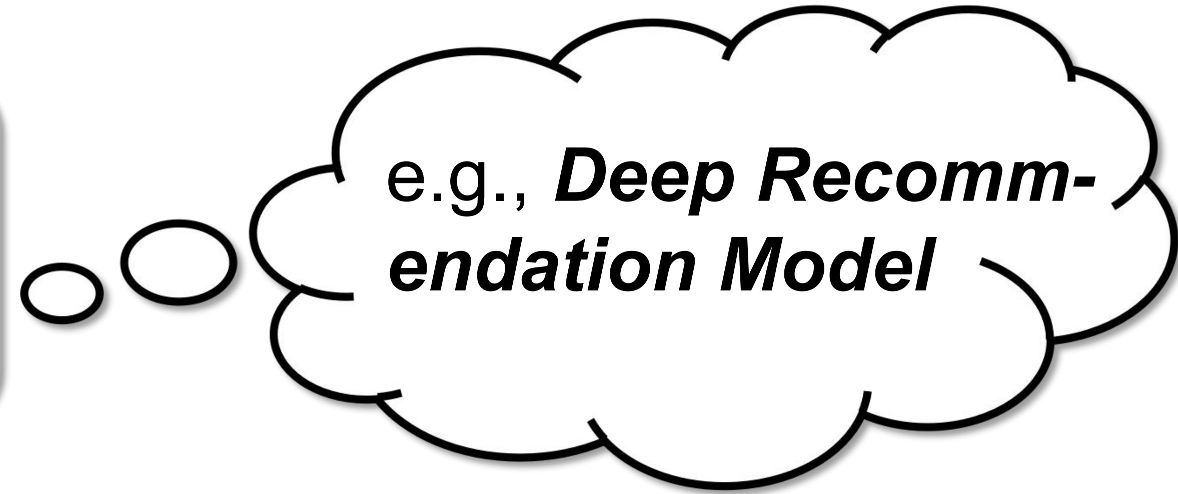


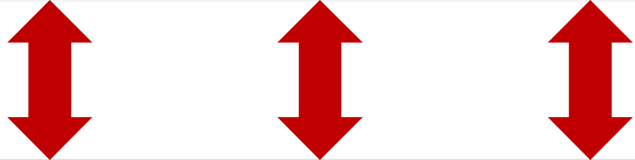
Unprecedented Data Scale of GPU App.

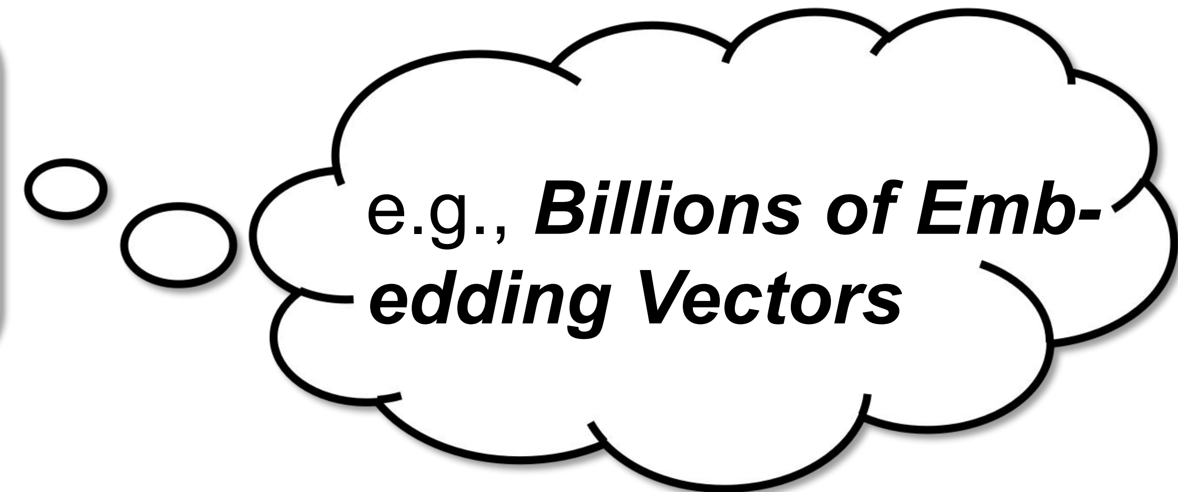




Unprecedented Data Scale of GPU App.



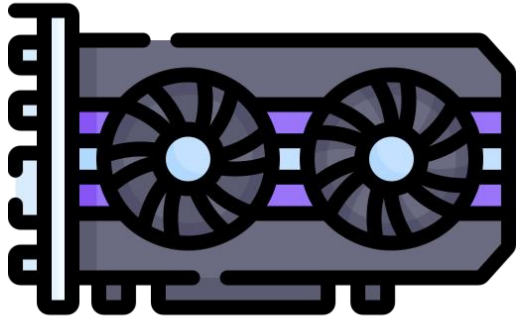




Existing Data Management of GPU App.

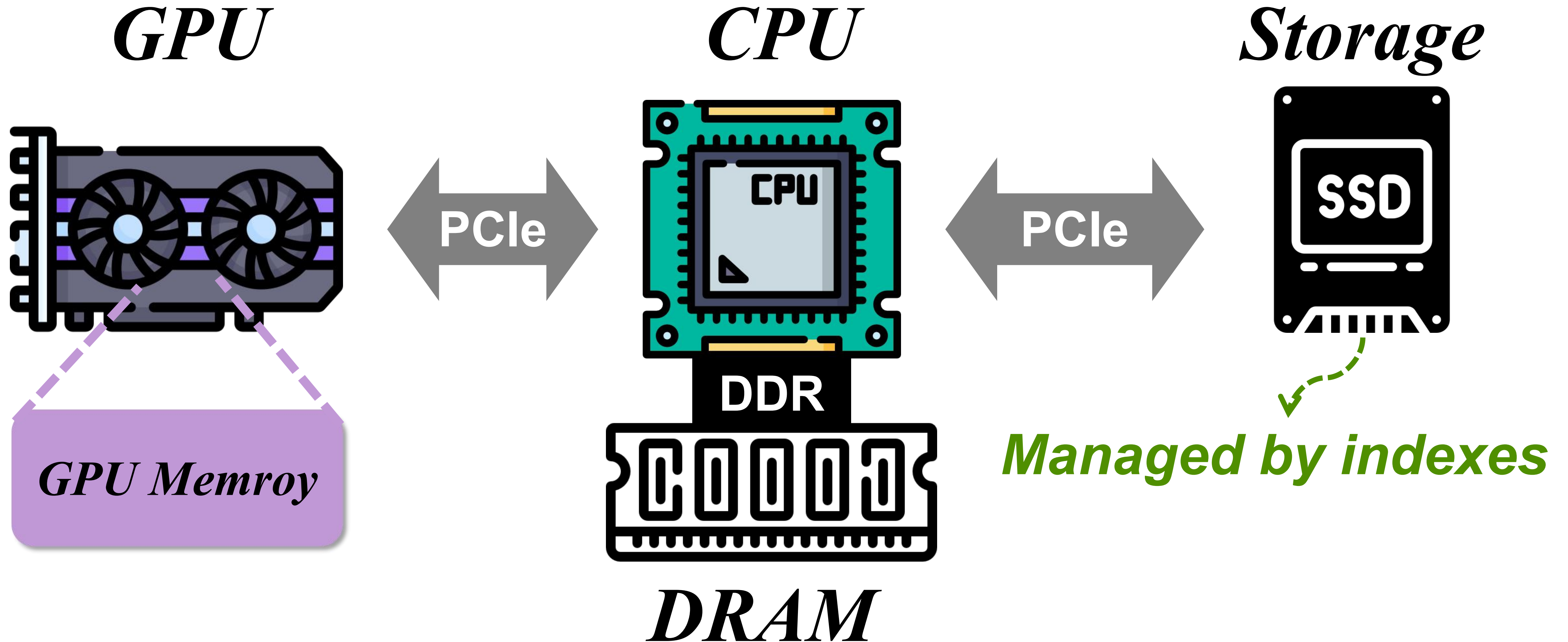
Existing Data Management of GPU App.

GPU

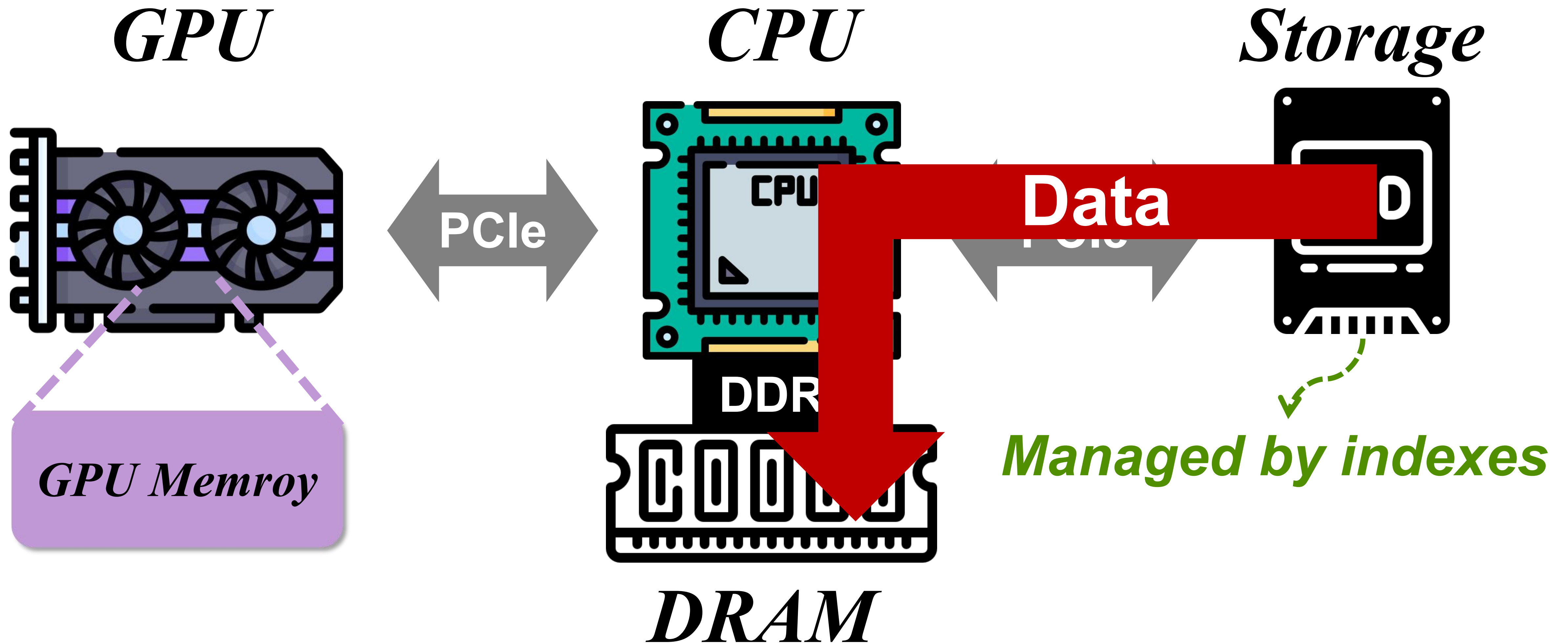


Storage

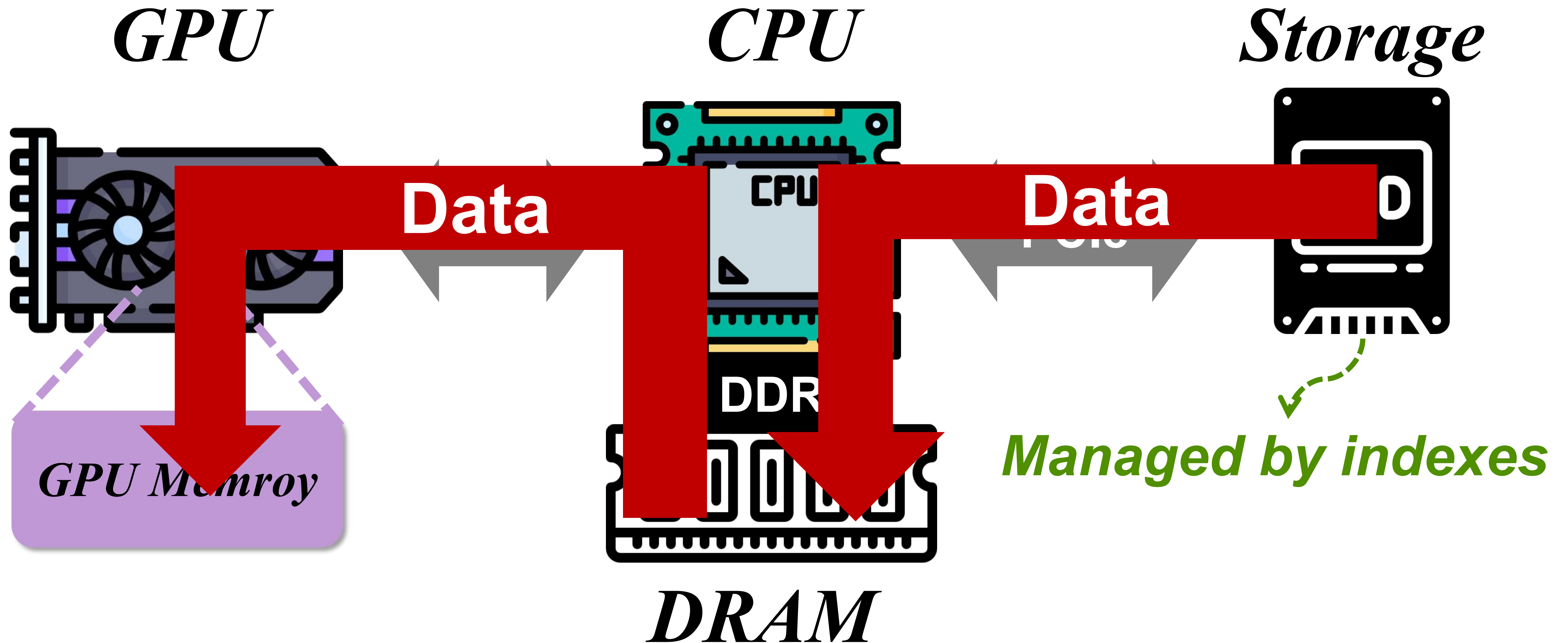
Existing Data Management of GPU App.



Existing Data Management of GPU App.



Existing Data Management of GPU App.



Existing Data Management of GPU App.

Existing Data Management of GPU App.

Existing Data Management of GPU App.

Large capacity and persistence

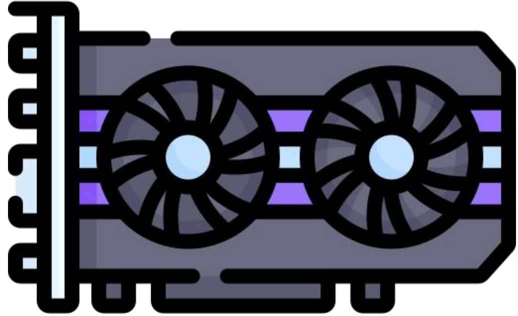
High overhead for data transfer

Extra CPU consumption

Direct Data Access from GPU

Direct Data Access from GPU

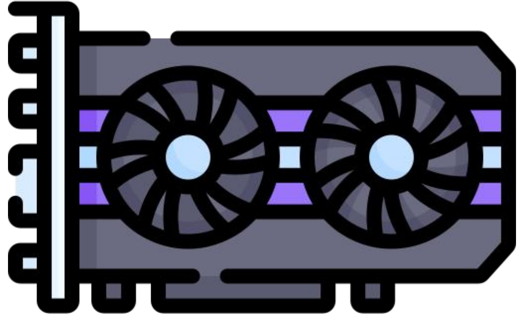
GPU



Storage

Direct Data Access from GPU

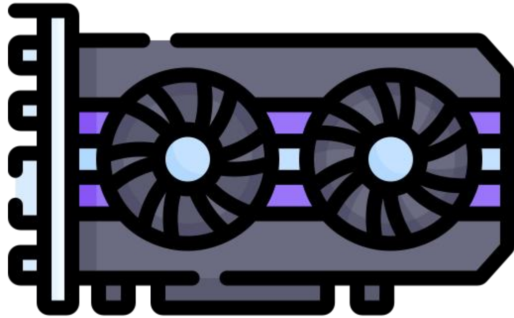
GPU



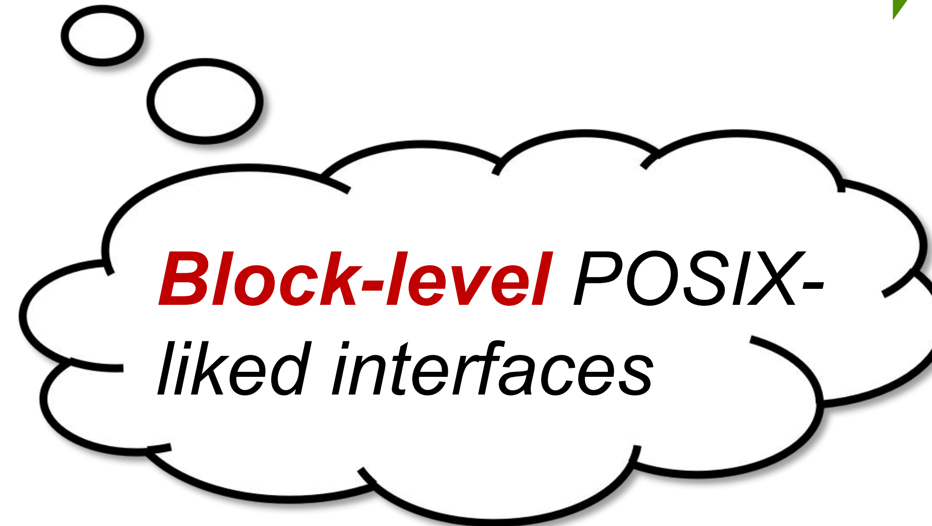
Storage

Direct Data Access from GPU

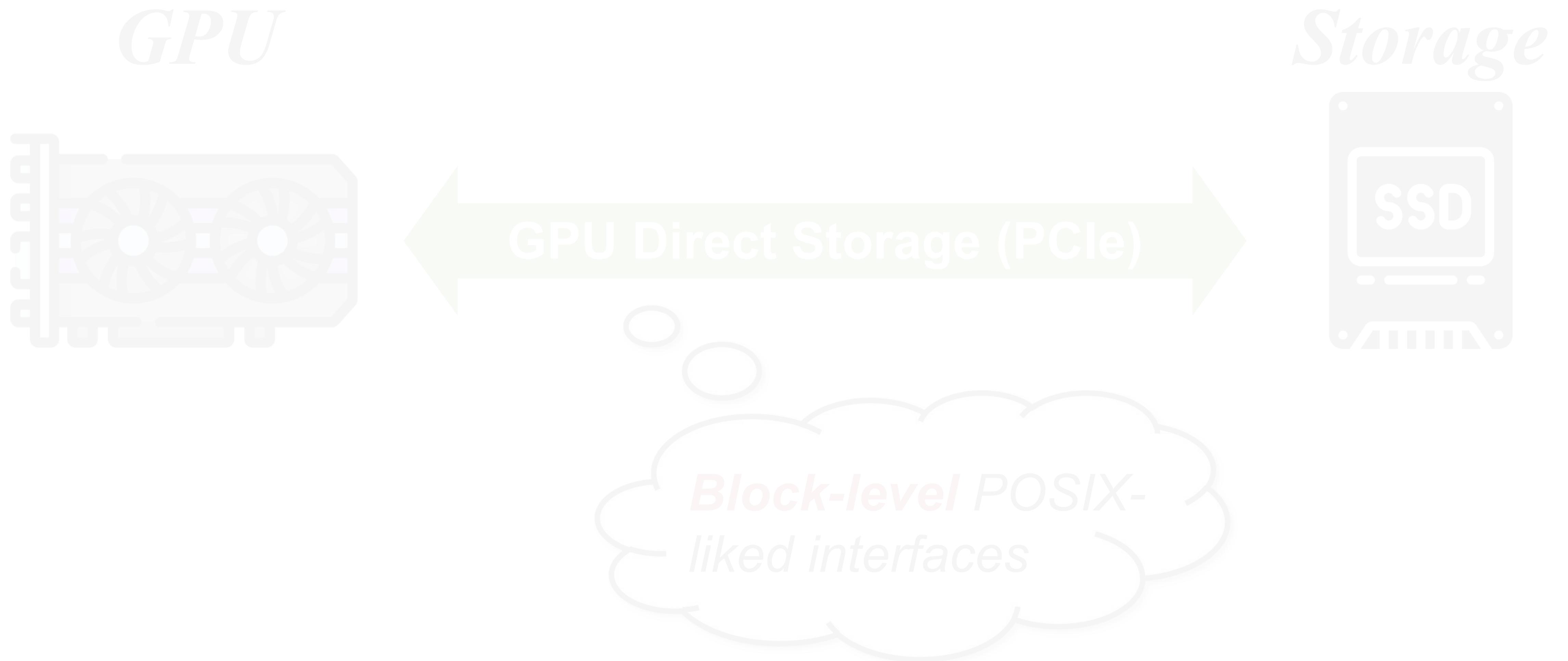
GPU



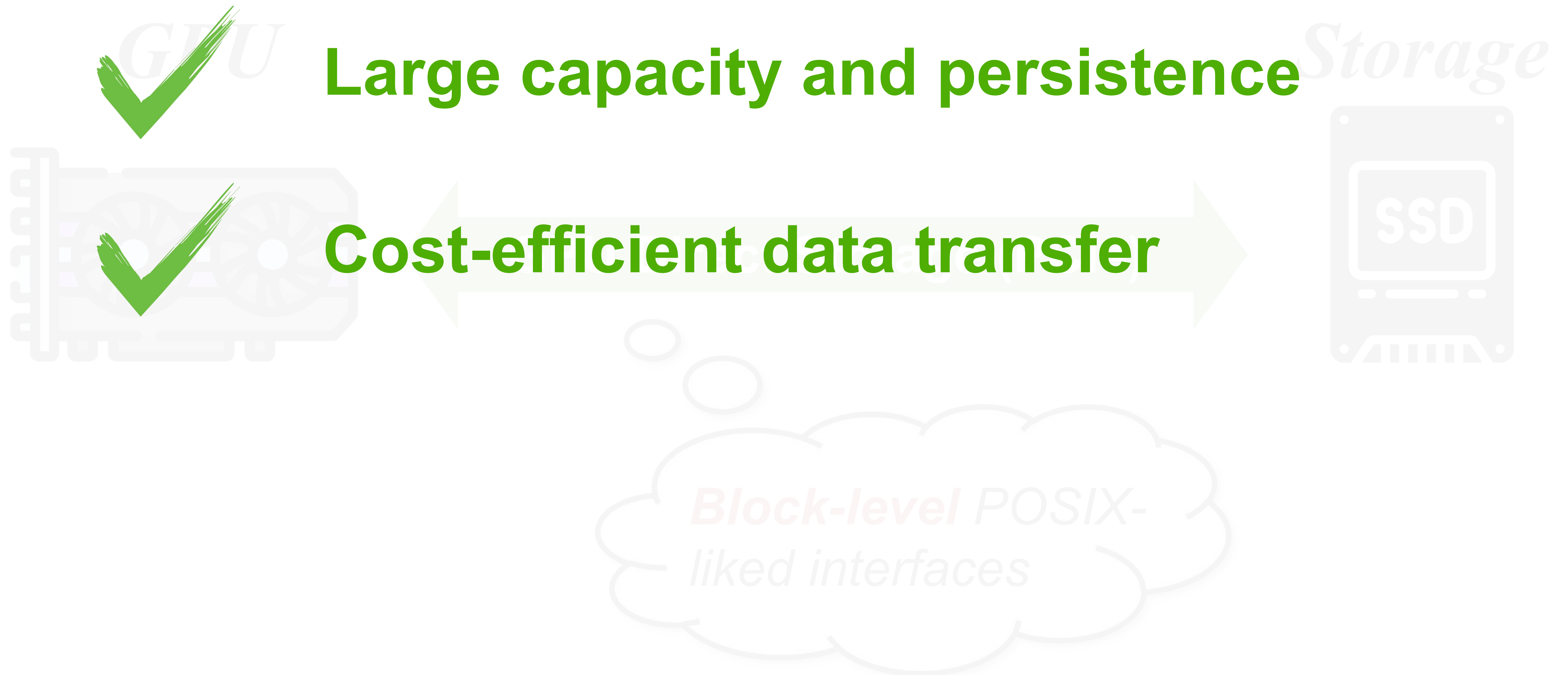
Storage



Direct Data Access from GPU



Direct Data Access from GPU



- Large capacity and persistence
- Cost-efficient data transfer

Direct Data Access from GPU

Large capacity and persistence

Cost-efficient data transfer

Hard to program data structure

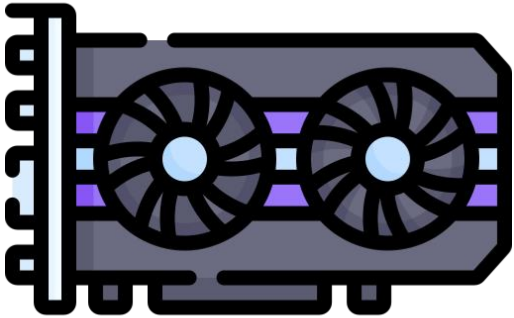
Transfer extraneous data

GPU with Persistent Memory (GPM)

[1] GPM: Leveraging Persistent Memory from a GPU [ASPLOS' 22]
[2] Scoped Buffered Persistency Model for GPUs [ASPLOS' 23]

GPU with Persistent Memory (GPM)

GPU



Persistent Memory

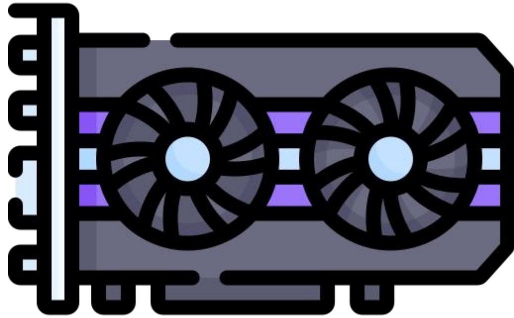
[1] GPM: Leveraging Persistent Memory from a GPU [ASPLOS' 22]
[2] Scoped Buffered Persistency Model for GPUs [ASPLOS' 23]

GPU with Persistent Memory (GPM)

- ✓ Large capacity
- ✓ Byte granularity
- ✓ Persistence
- ✓ High performance

...

GPU



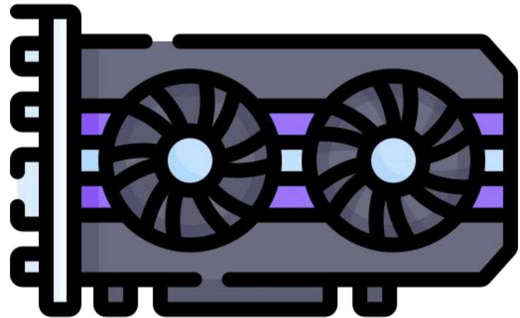
Persistent Memory

[1] GPM: Leveraging Persistent Memory from a GPU [ASPLOS' 22]
[2] Scoped Buffered Persistency Model for GPUs [ASPLOS' 23]

GPU with Persistent Memory (GPM)

- ✓ Large capacity ...
- ✓ Byte granularity
- ✓ Persistence
- ✓ High performance

GPU



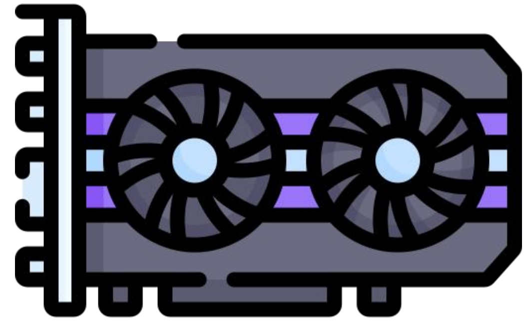
Persistent Memory

[1] GPM: Leveraging Persistent Memory from a GPU [ASPLOS' 22]
[2] Scoped Buffered Persistency Model for GPUs [ASPLOS' 23]

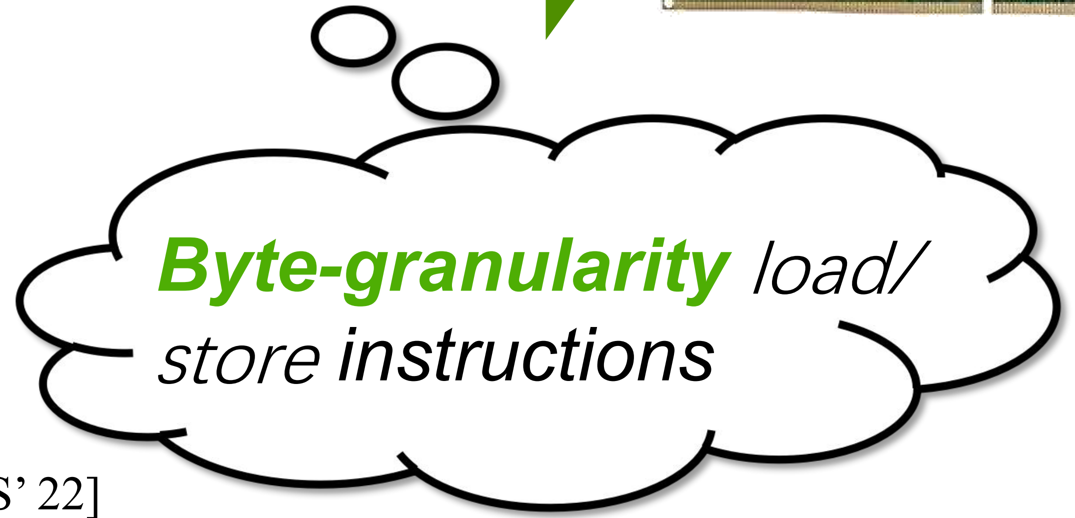
GPU with Persistent Memory (GPM)

- ✓ Large capacity ...
- ✓ Byte granularity
- ✓ Persistence
- ✓ High performance

GPU



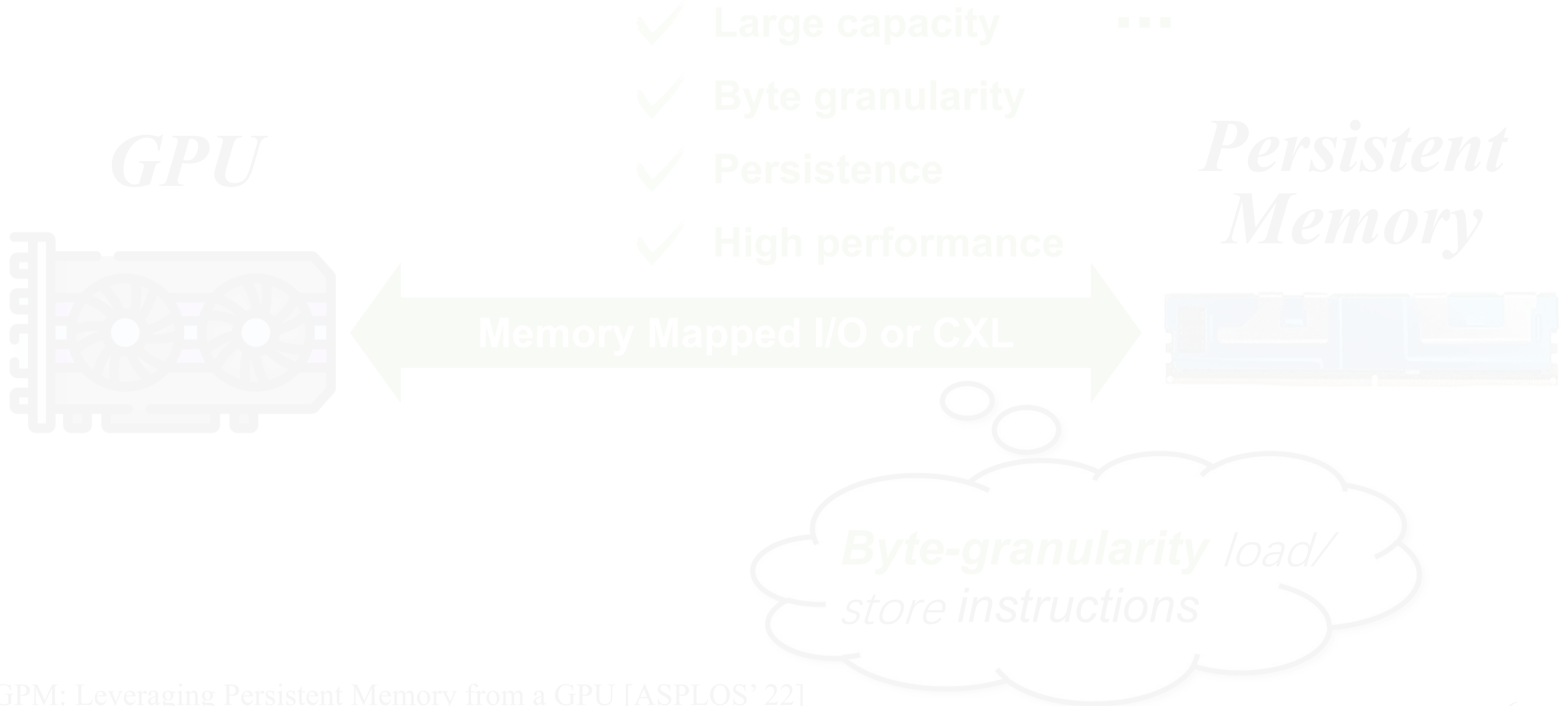
Persistent Memory



[1] GPM: Leveraging Persistent Memory from a GPU [ASPLOS' 22]

[2] Scoped Buffered Persistency Model for GPUs [ASPLOS' 23]

GPU with Persistent Memory (GPM)



[1] GPM: Leveraging Persistent Memory from a GPU [ASPLOS' 22]
[2] Scoped Buffered Persistency Model for GPUs [ASPLOS' 23]

GPU with Persistent Memory (GPM)

Large capacity and persistence

Cost-efficient and fine-grained data transfer

Easy to program data structure

[1] GPM: Leveraging Persistent Memory from a GPU [ASPLOS' 22]
[2] Scoped Buffered Persistency Model for GPUs [ASPLOS' 23]

Hash Index

Hash Index

Hash Index

Hash Index

Constant-scale point query

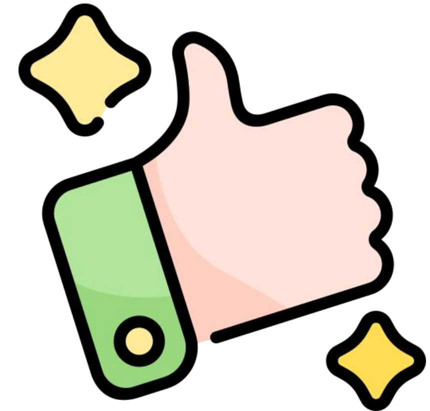
Good for parallel access

Hash Index

Constant-scale point query

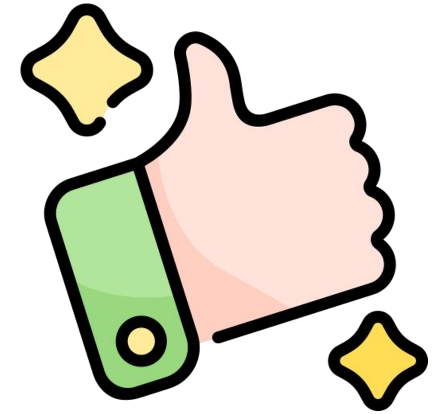
Good for parallel access

GPM Hash Index



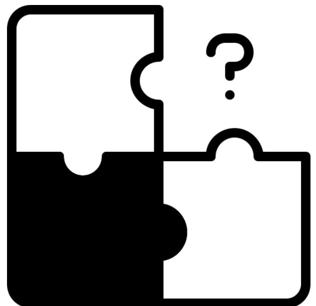
Hash Index

GPM Hash Index



Constant-scale point query

Good for parallel access



However, it is non-trivial to implement an efficient GPM hash index

Challenge 1: Agnostic to GPU Execution Manner

Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

Challenge 1: Agnostic to GPU Execution Manner

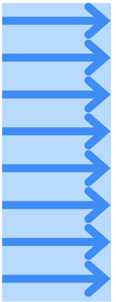
Warp Divergence

```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```

Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```

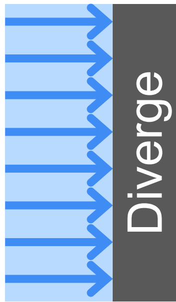


Time

Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```

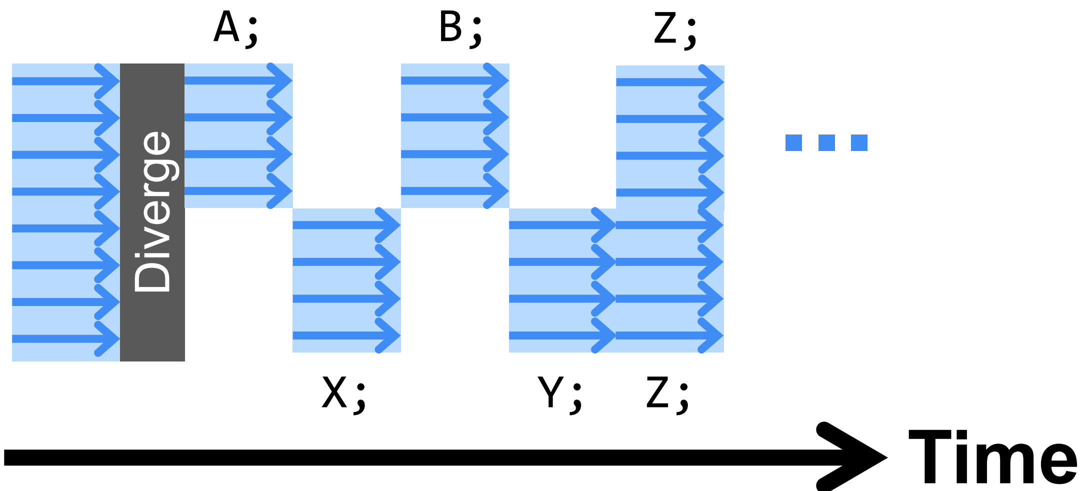


Time

Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

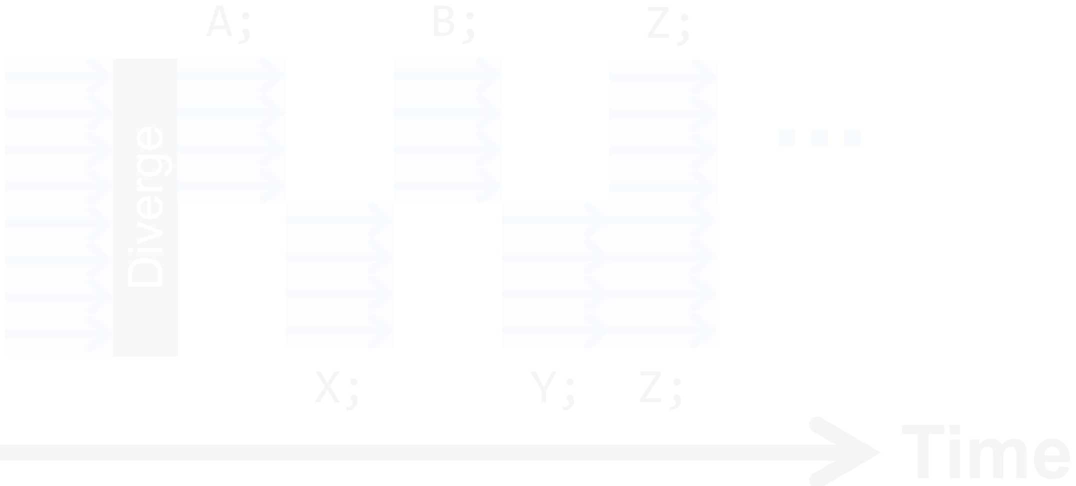
```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```



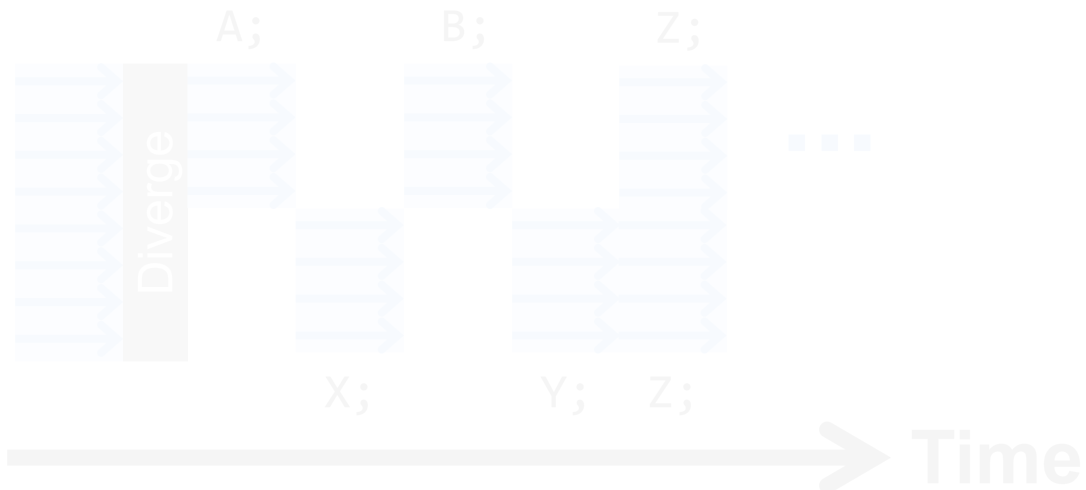
Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```

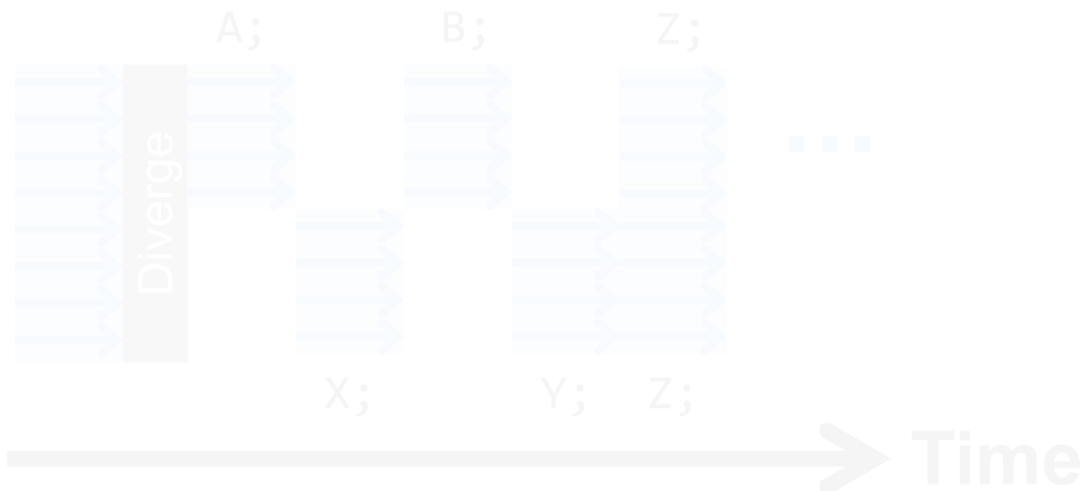


Challenge 1: Agnostic to GPU Execution Manner



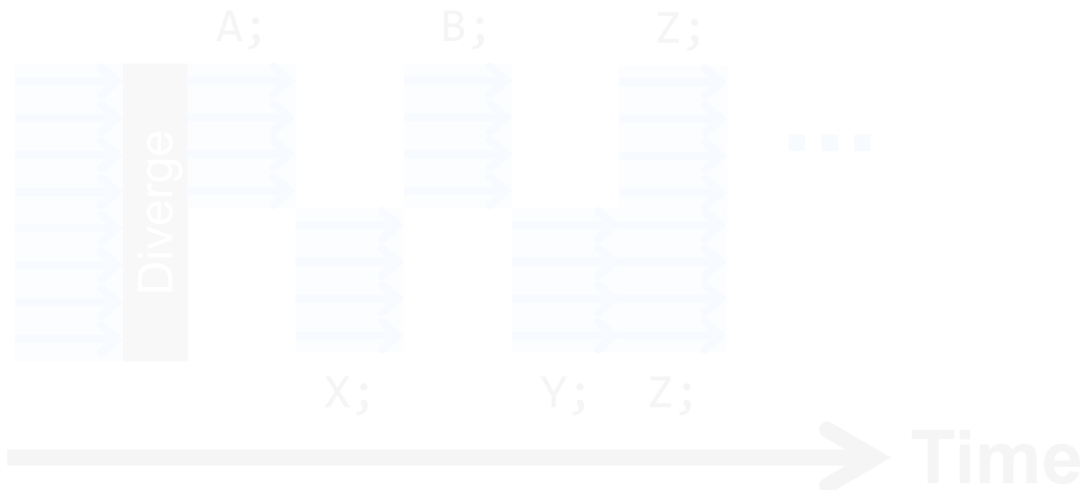
Coalesced memory accesses

Challenge 1: Agnostic to GPU Execution Manner



Coalesced memory accesses

Challenge 1: Agnostic to GPU Execution Manner

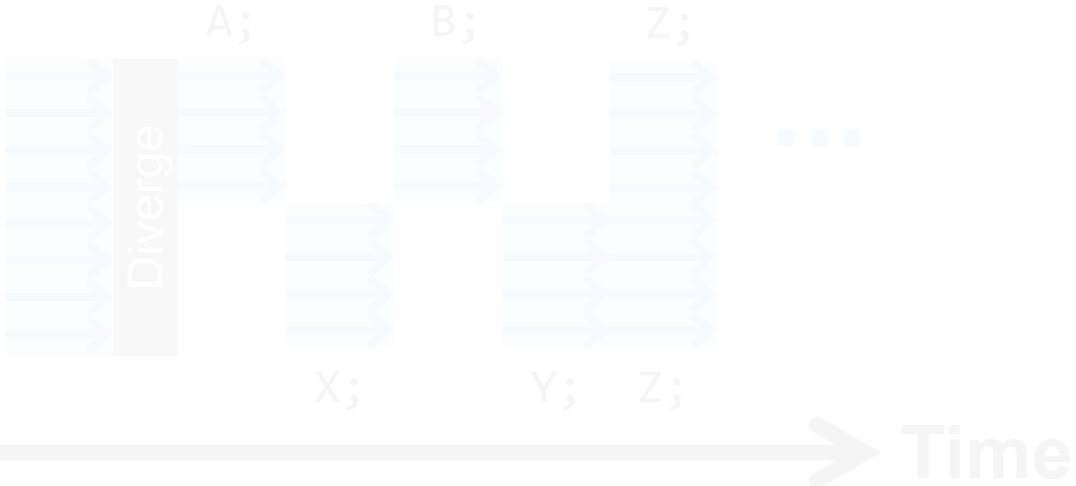


Coalesced memory accesses

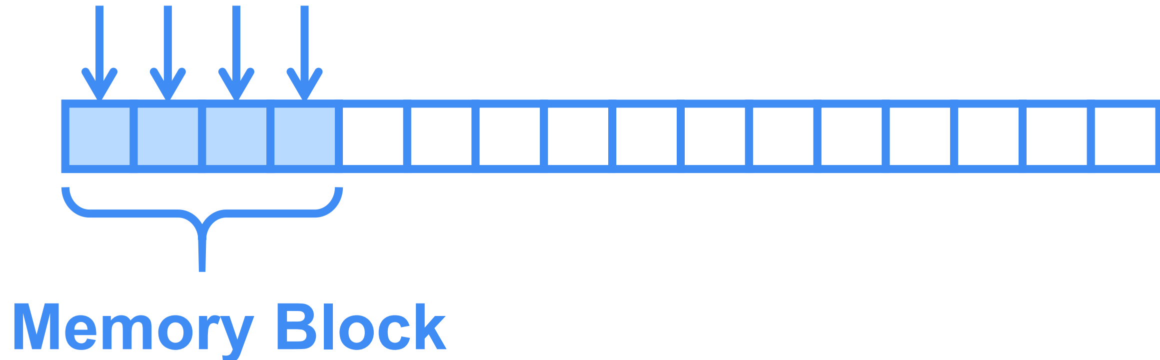
Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```

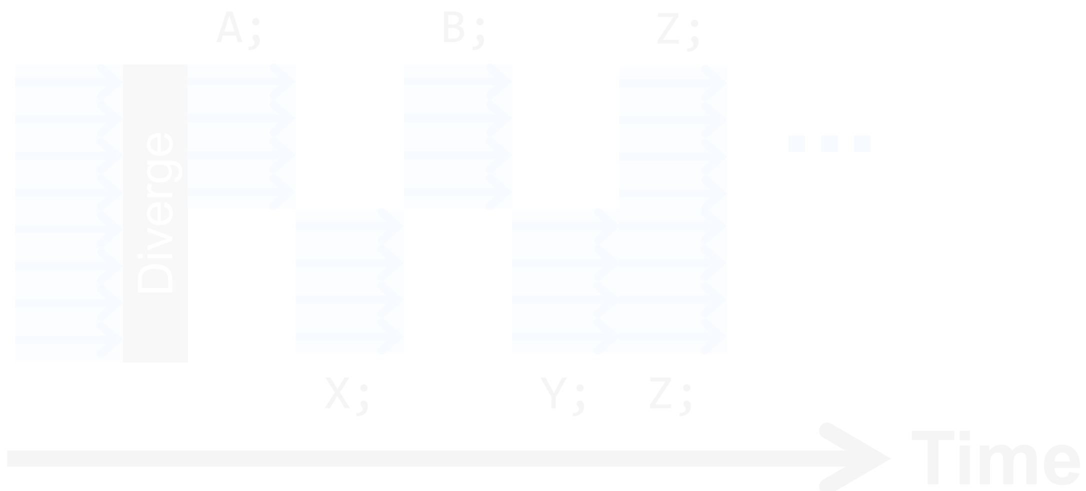


Coalesced memory accesses

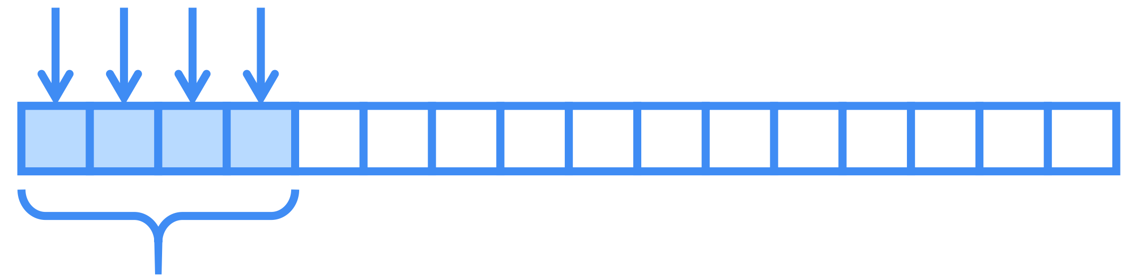


Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence



Coalesced memory accesses

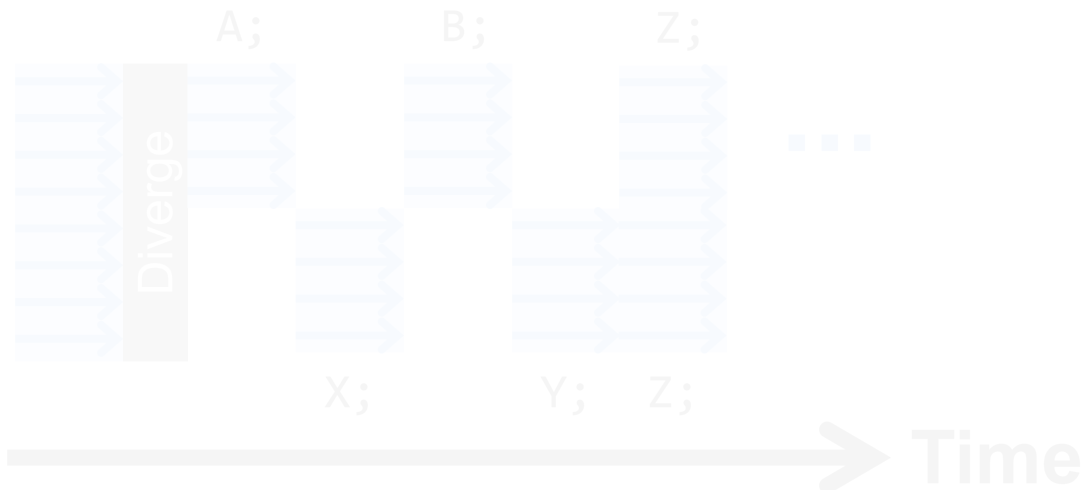


Uncoalesced memory accesses

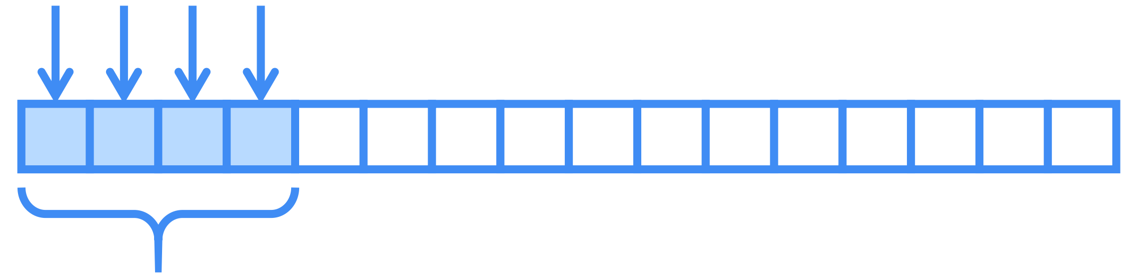
Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```

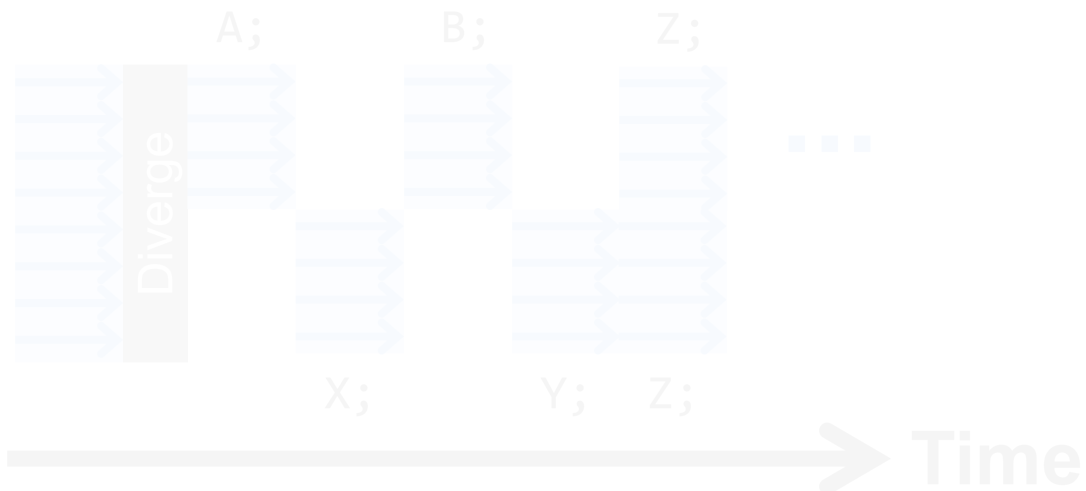


Coalesced memory accesses

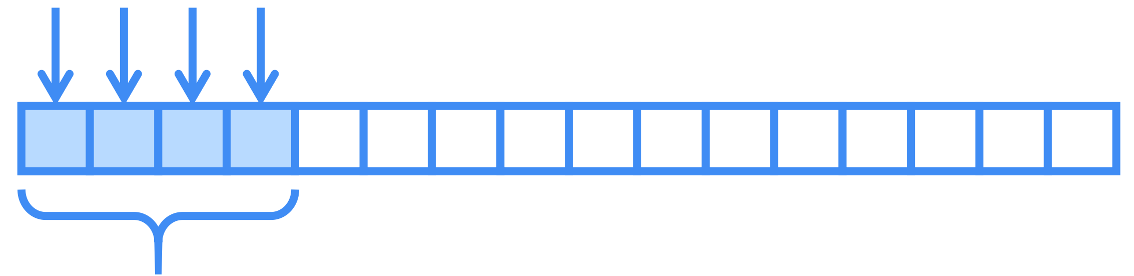


Uncoalesced memory accesses

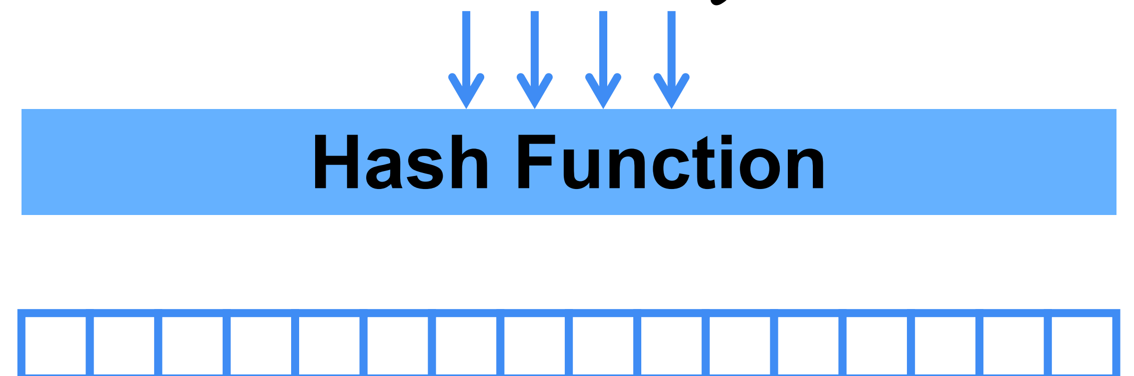
Challenge 1: Agnostic to GPU Execution Manner



Coalesced memory accesses



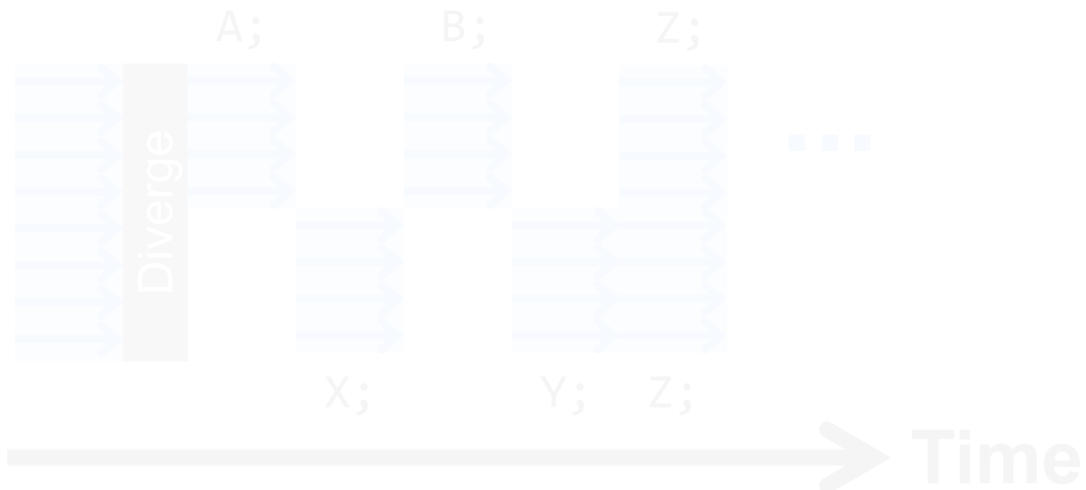
Uncoalesced memory accesses



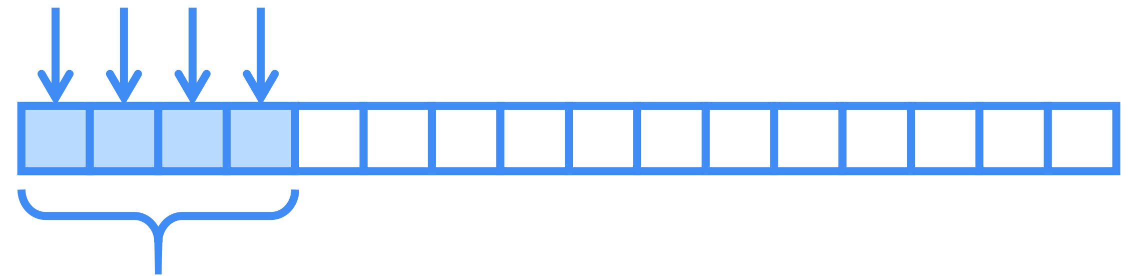
Challenge 1: Agnostic to GPU Execution Manner

Warp Divergence

```
if (thread_id < 4) {
    A;
    B;
} else {
    X;
    Y;
}
Z;
```

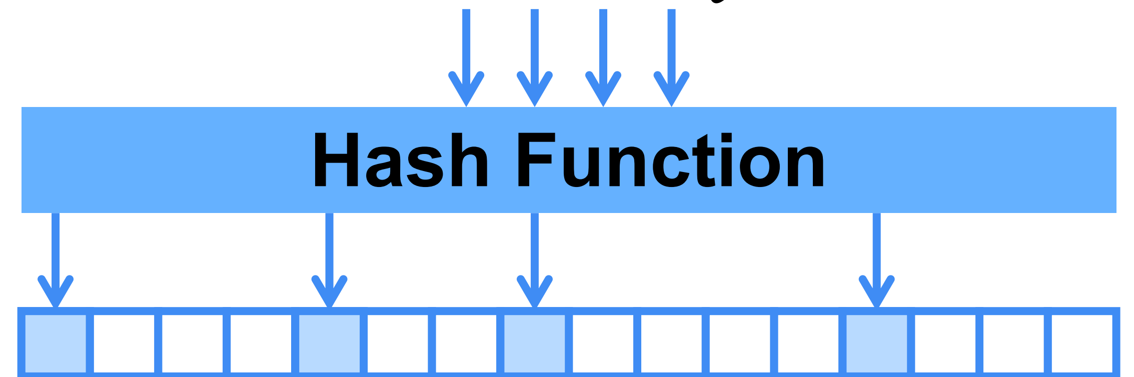


Coalesced memory accesses



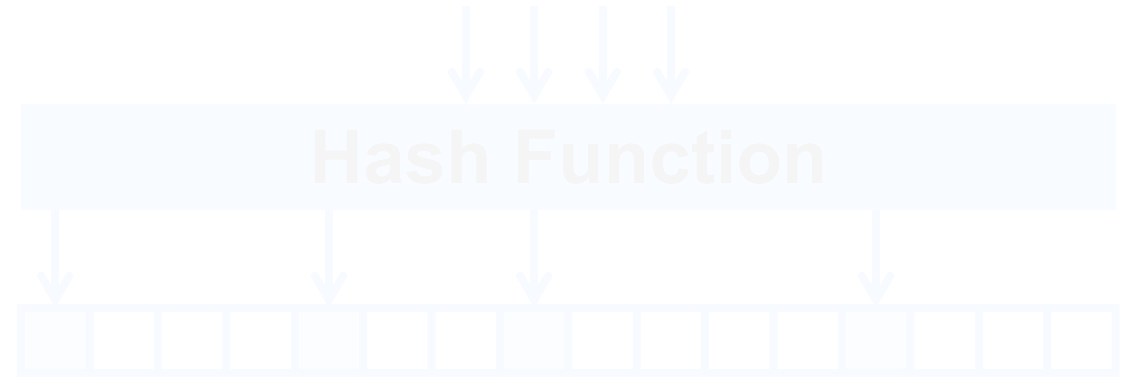
Memory Block

Uncoalesced memory accesses



Challenge 1: Agnostic to GPU Execution Manner

Severe ***warp divergence*** and ***uncoalesced memory accesses*** lead to **Performance Degradation**



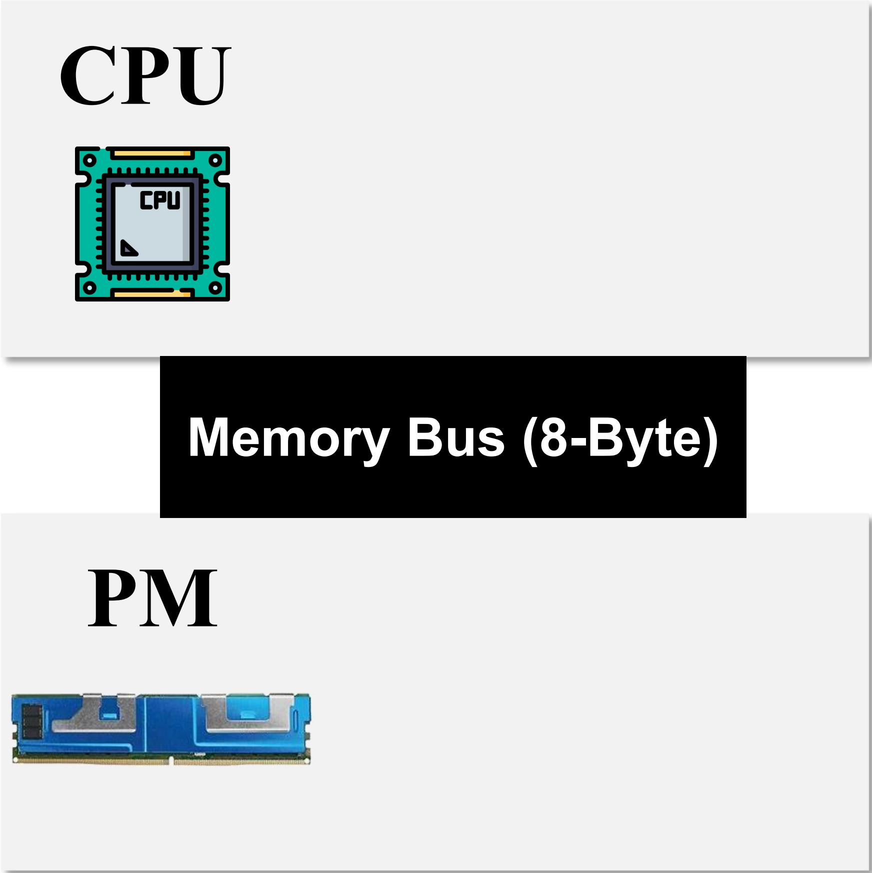
Challenge 2: Ensure Crash Consistency

Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee

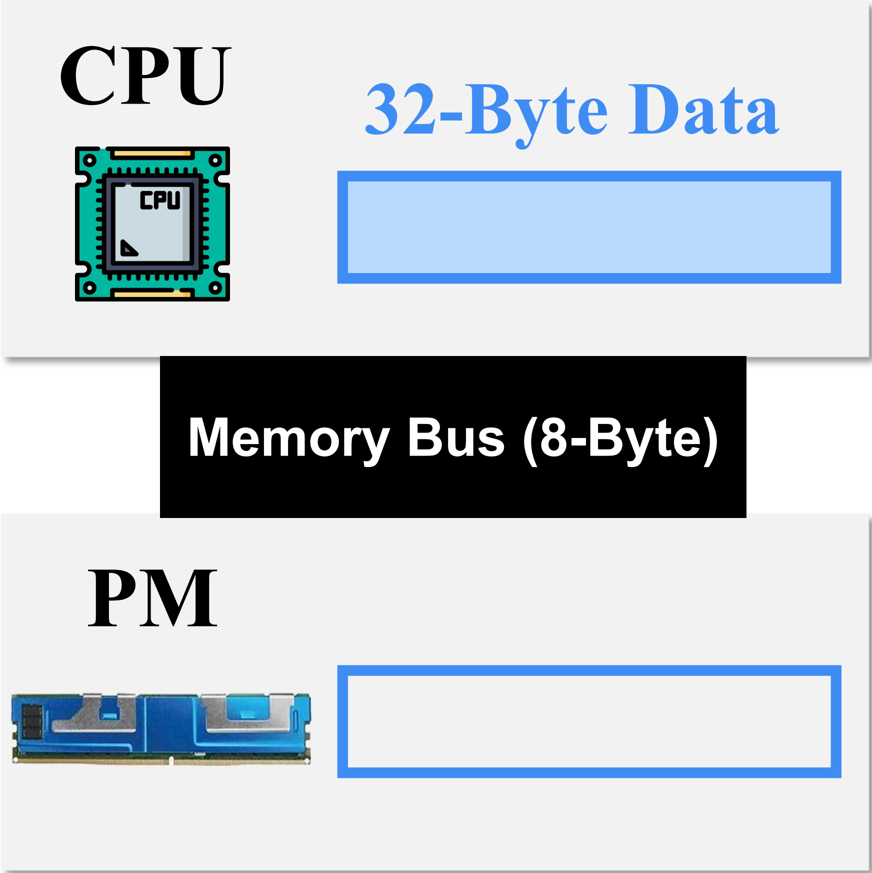
Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee



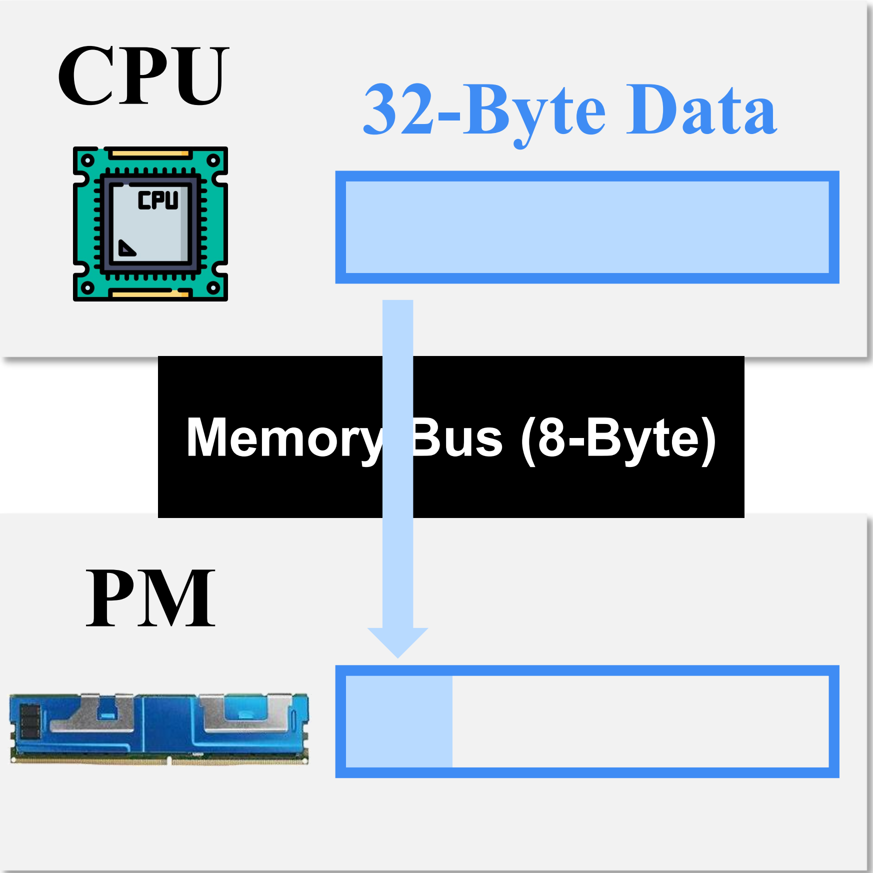
Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee



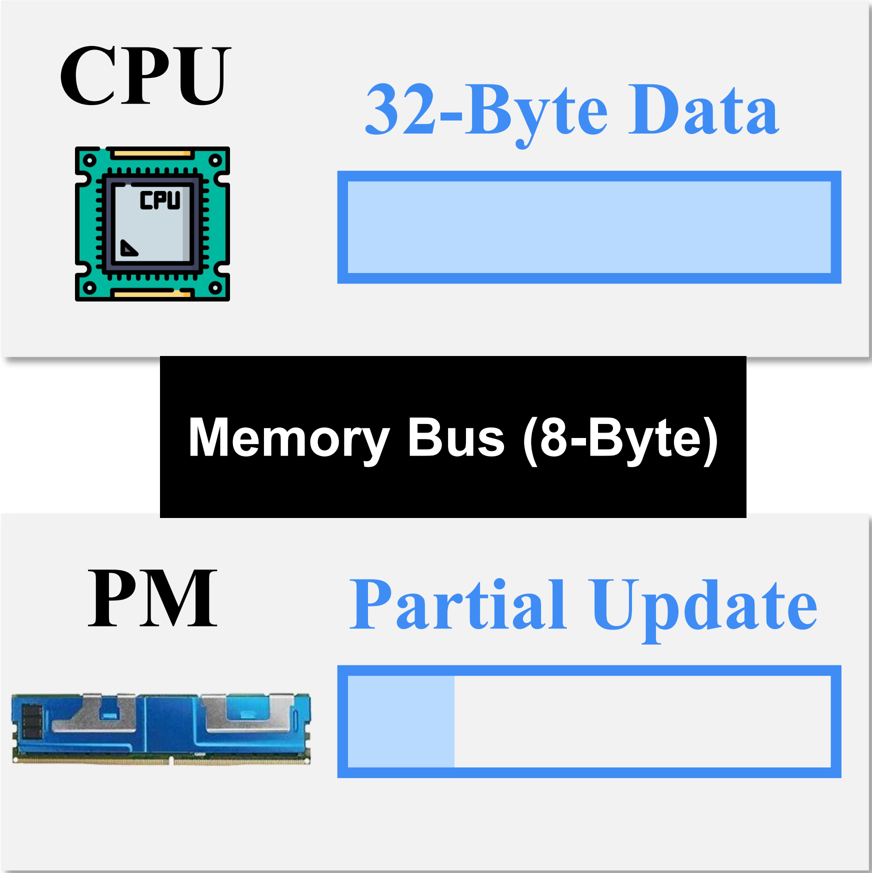
Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee



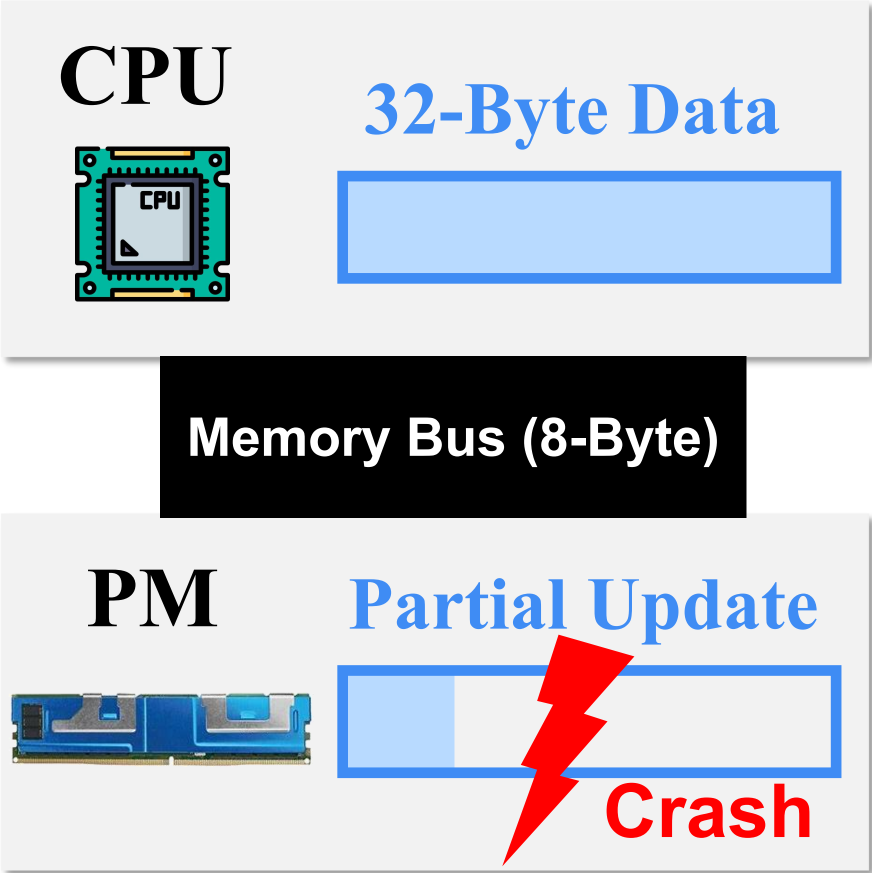
Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee



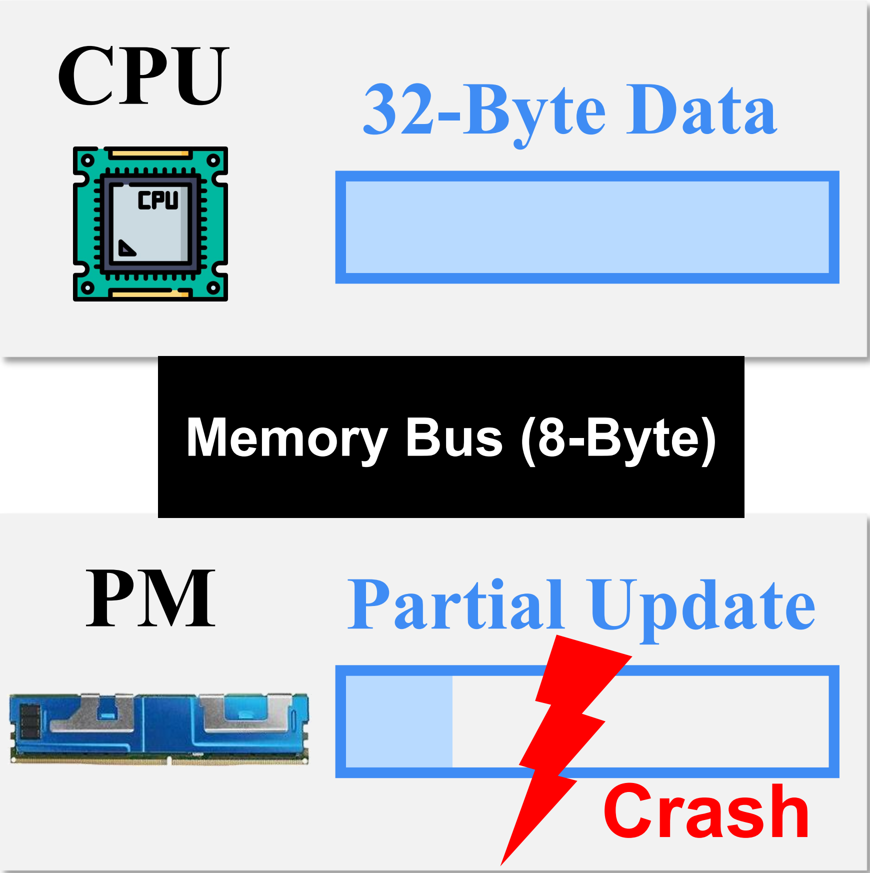
Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee



Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee

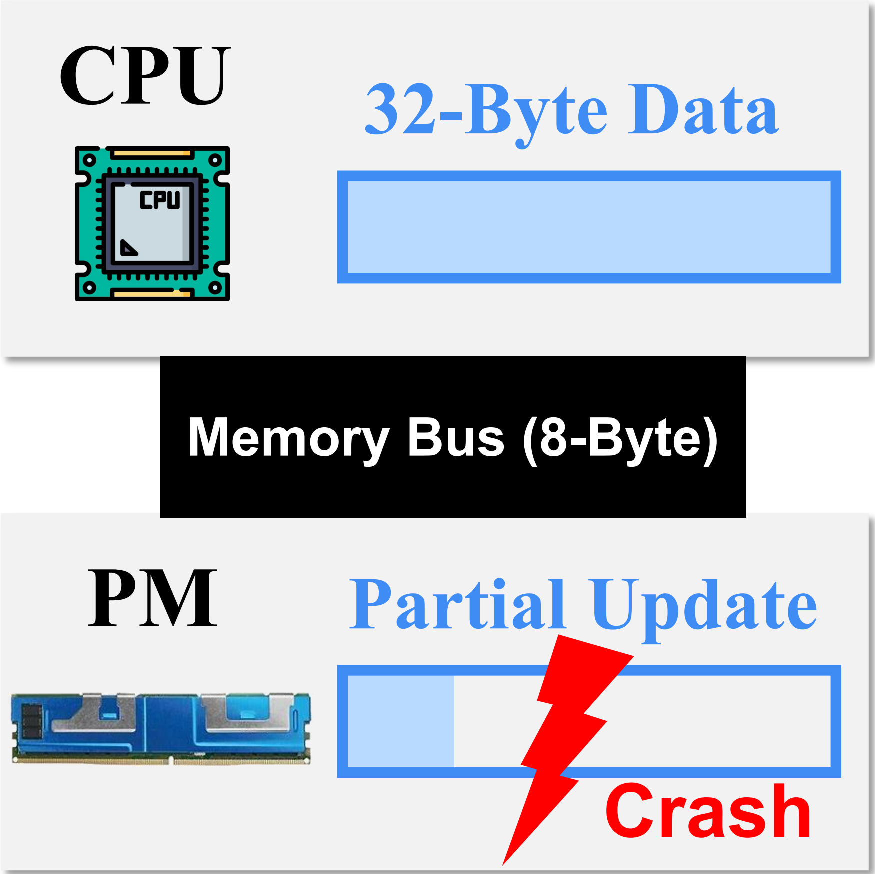


Data Inconsistency!

Challenge 2: Ensure Crash Consistency

Without Consistency Guarantee

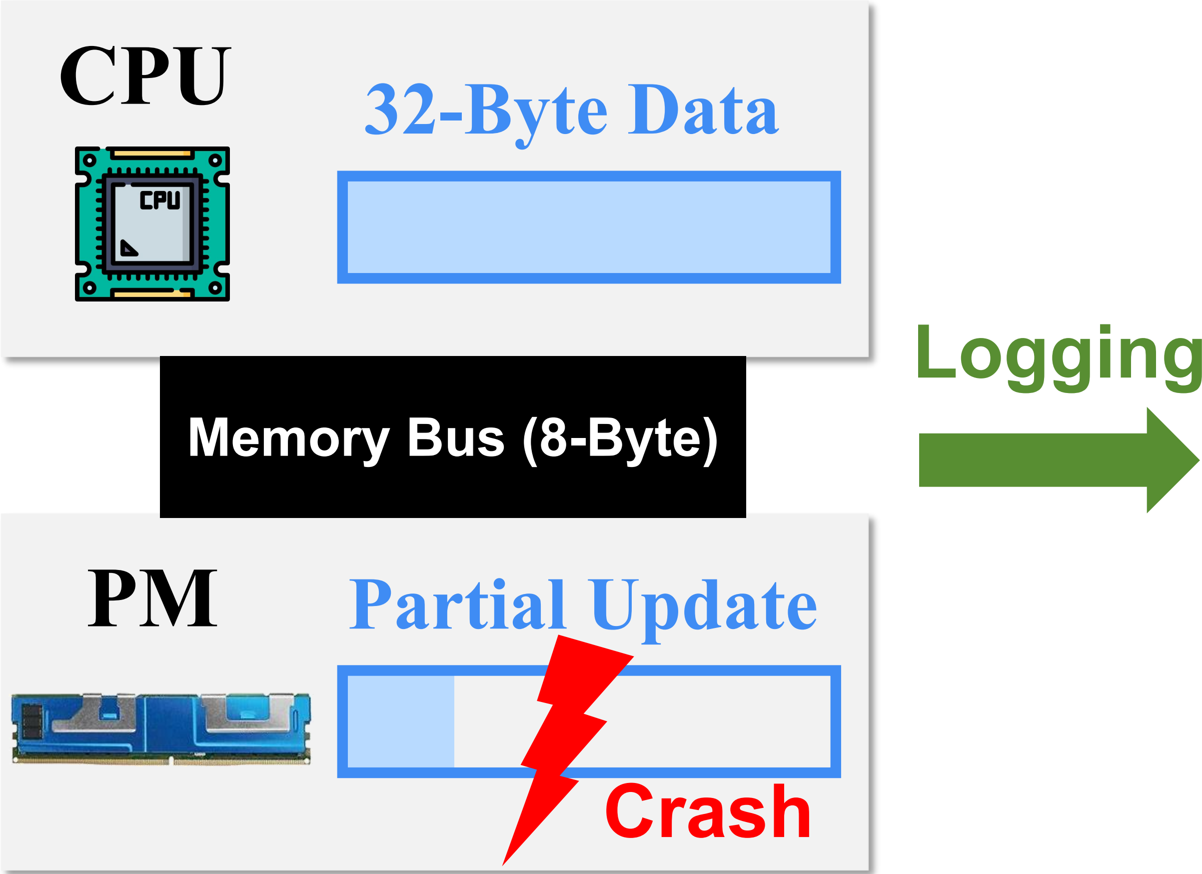
With Consistency Guarantee



Data Inconsistency!

Challenge 2: Ensure Crash Consistency

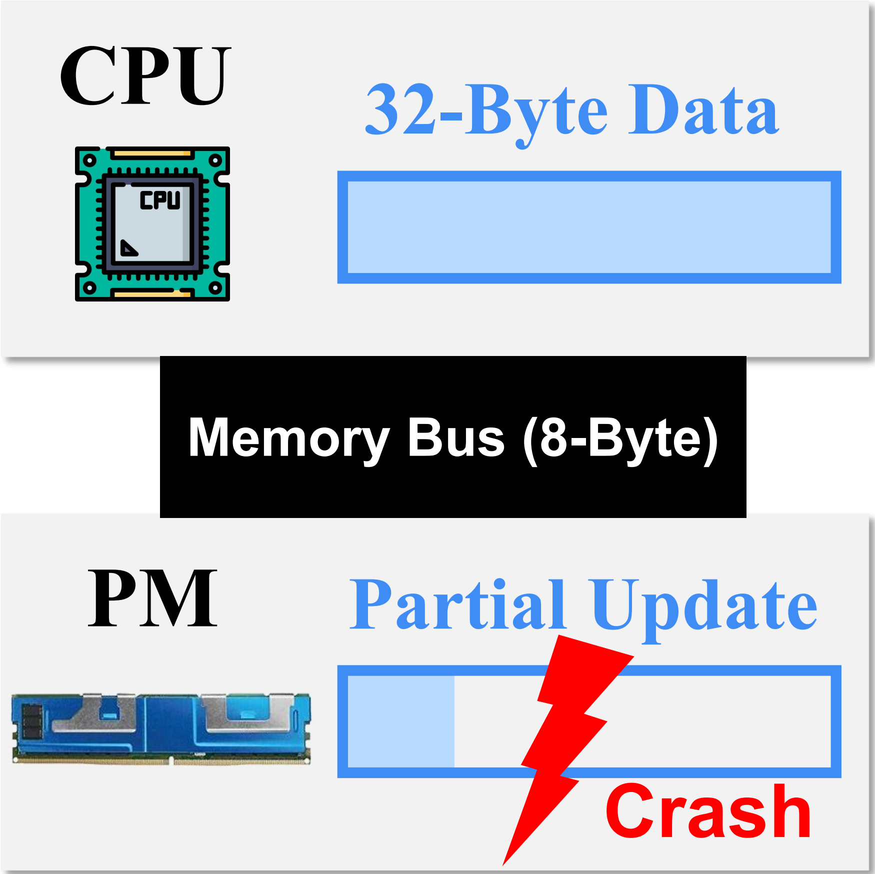
Without Consistency Guarantee | **With Consistency Guarantee**



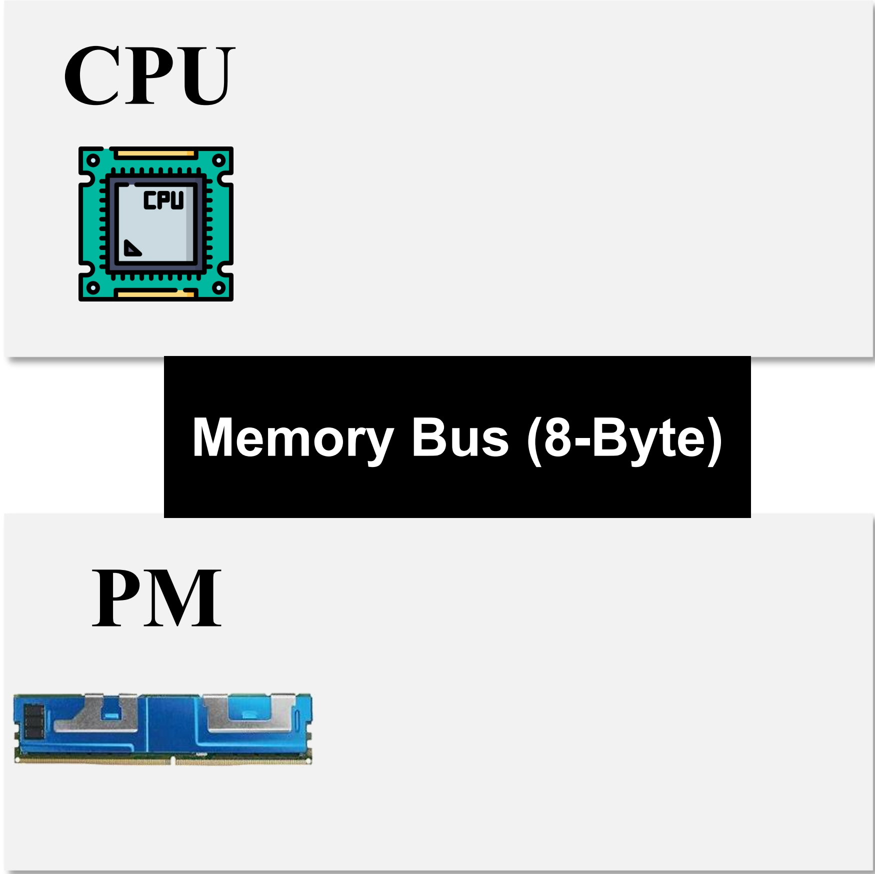
Logging

Data Inconsistency!

Challenge 2: Ensure Crash Consistency

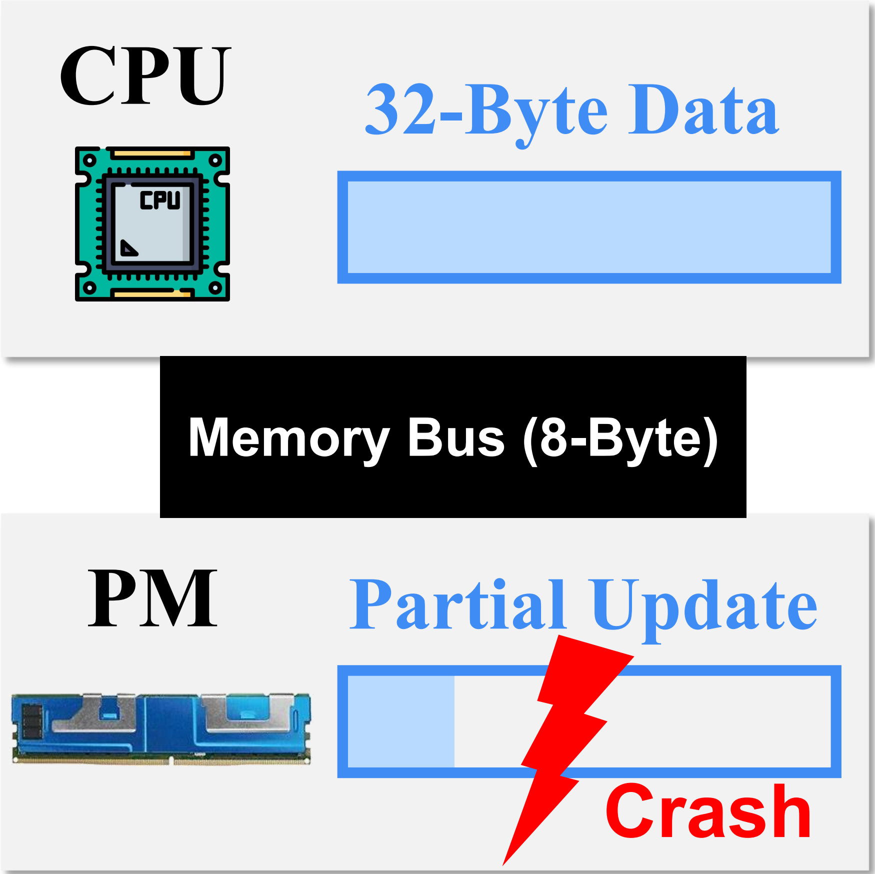


Logging

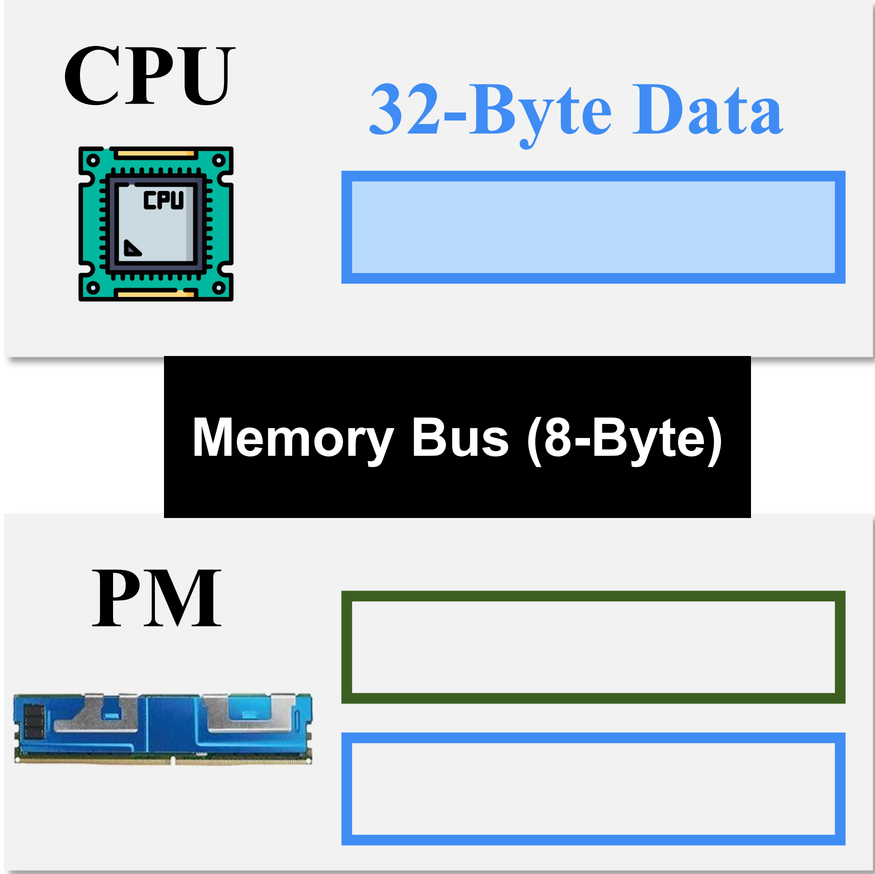


Data Inconsistency!

Challenge 2: Ensure Crash Consistency

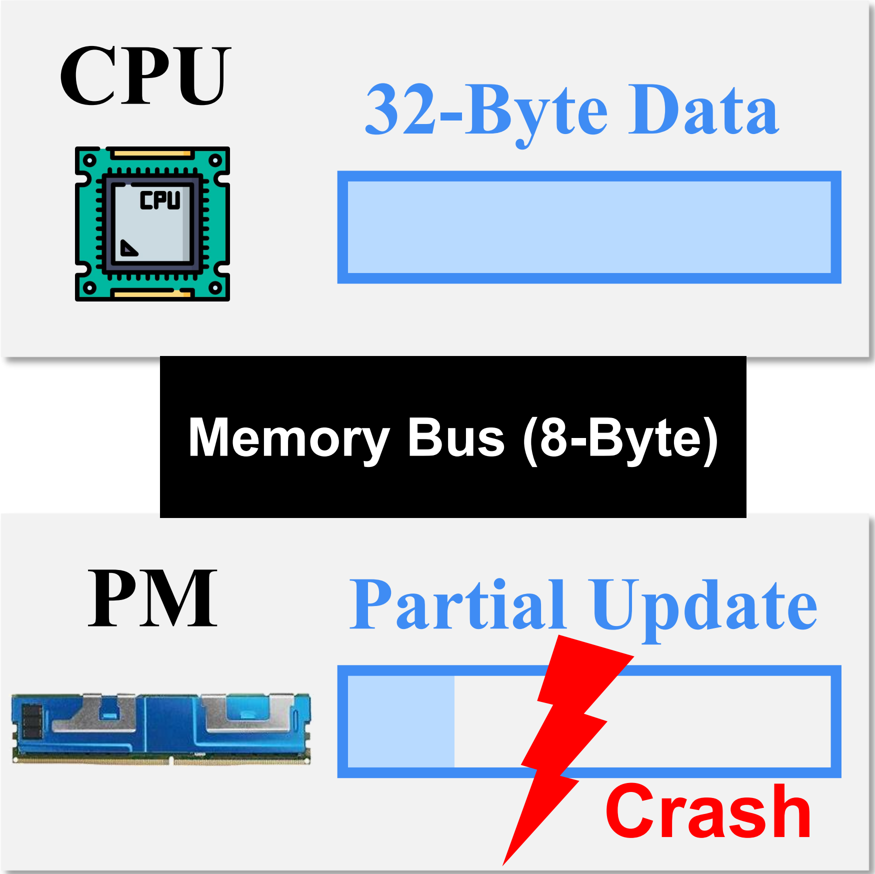


Logging

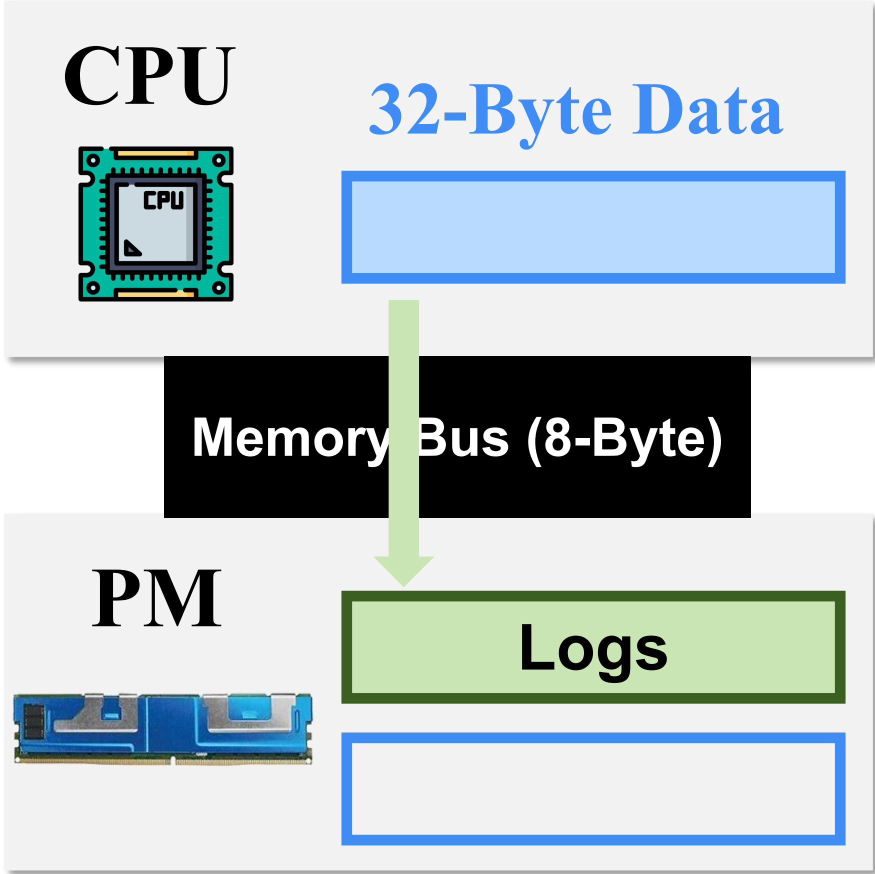


Data Inconsistency!

Challenge 2: Ensure Crash Consistency

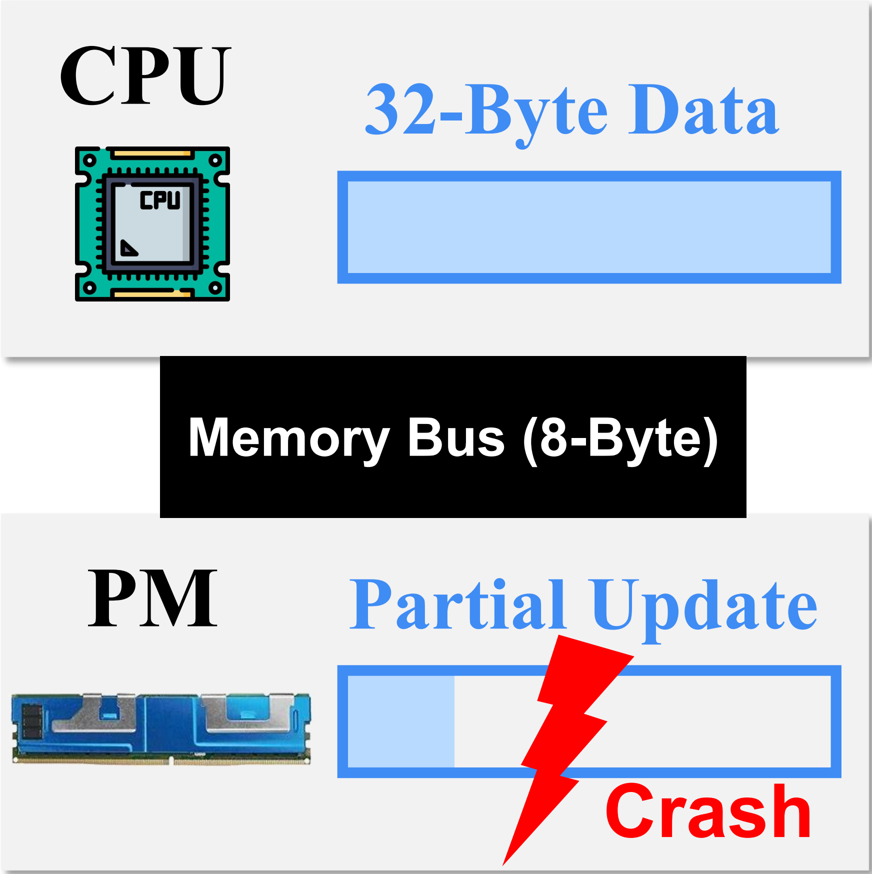


Logging

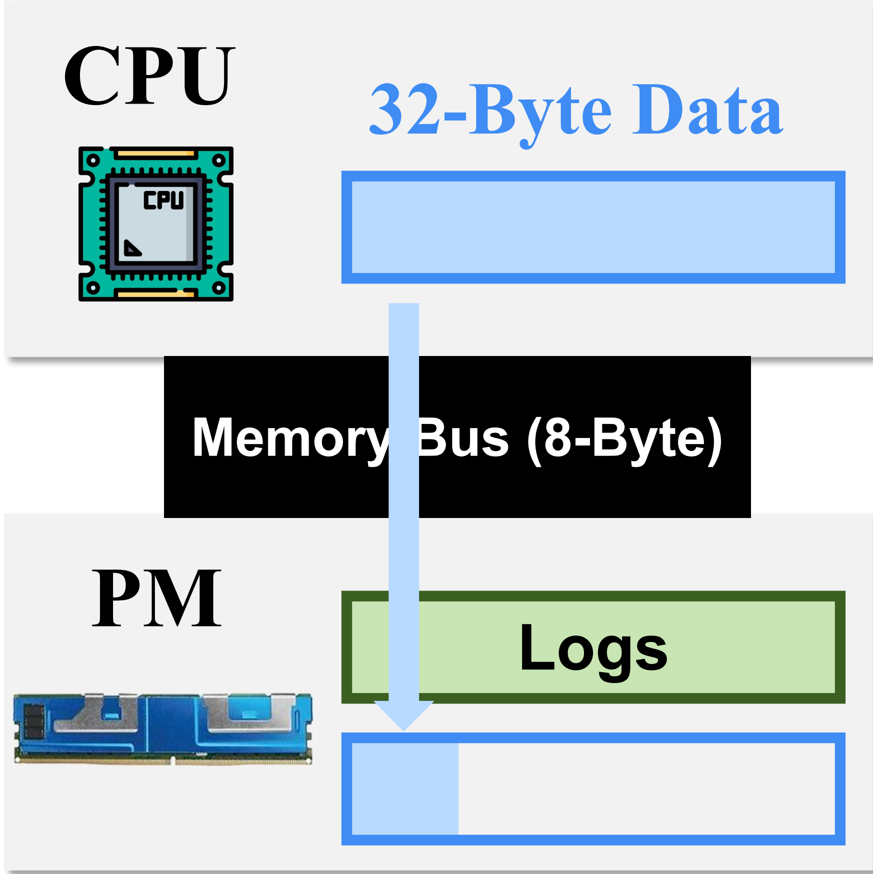


Data Inconsistency!

Challenge 2: Ensure Crash Consistency

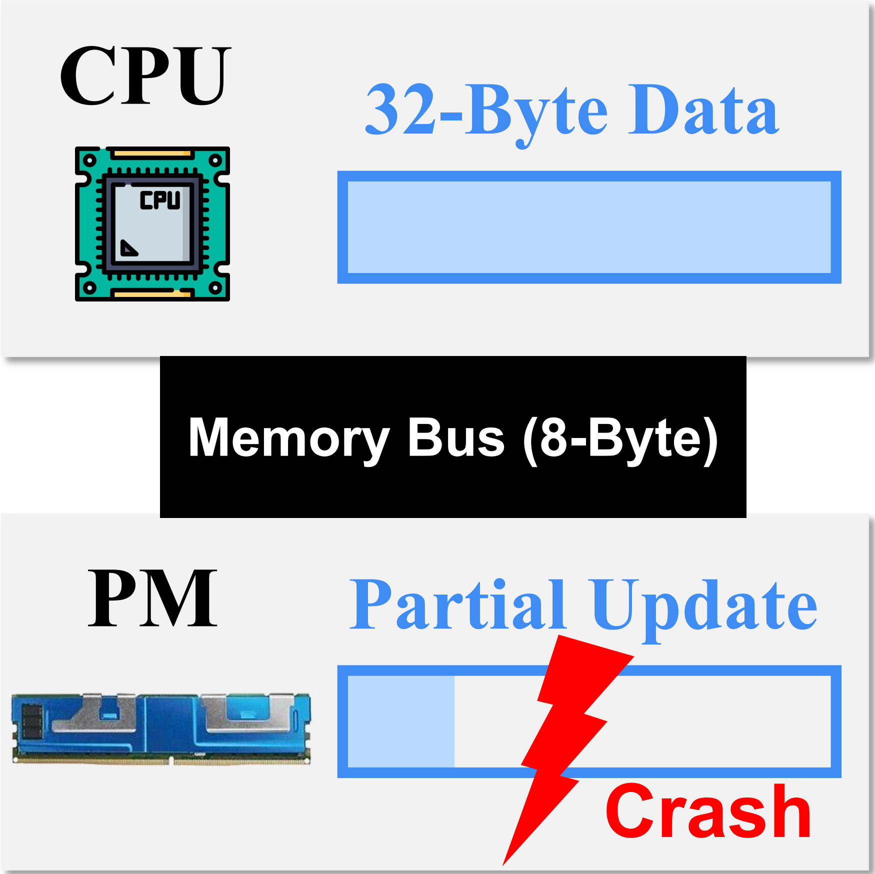


Logging

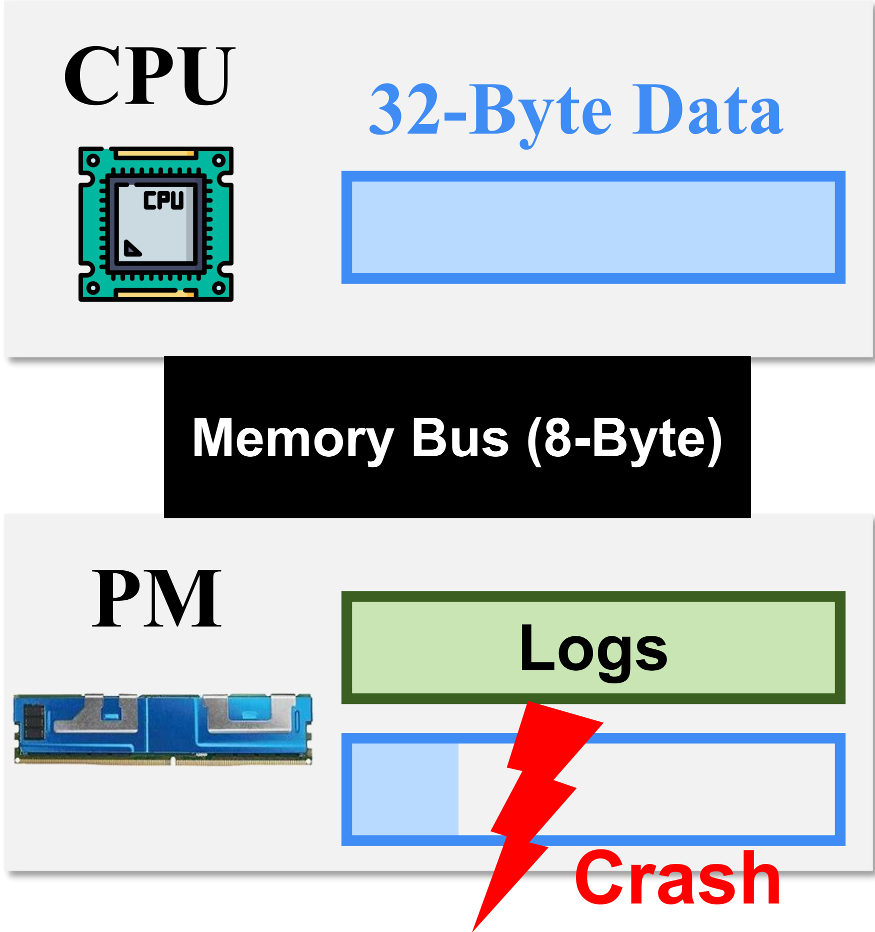


Data Inconsistency!

Challenge 2: Ensure Crash Consistency

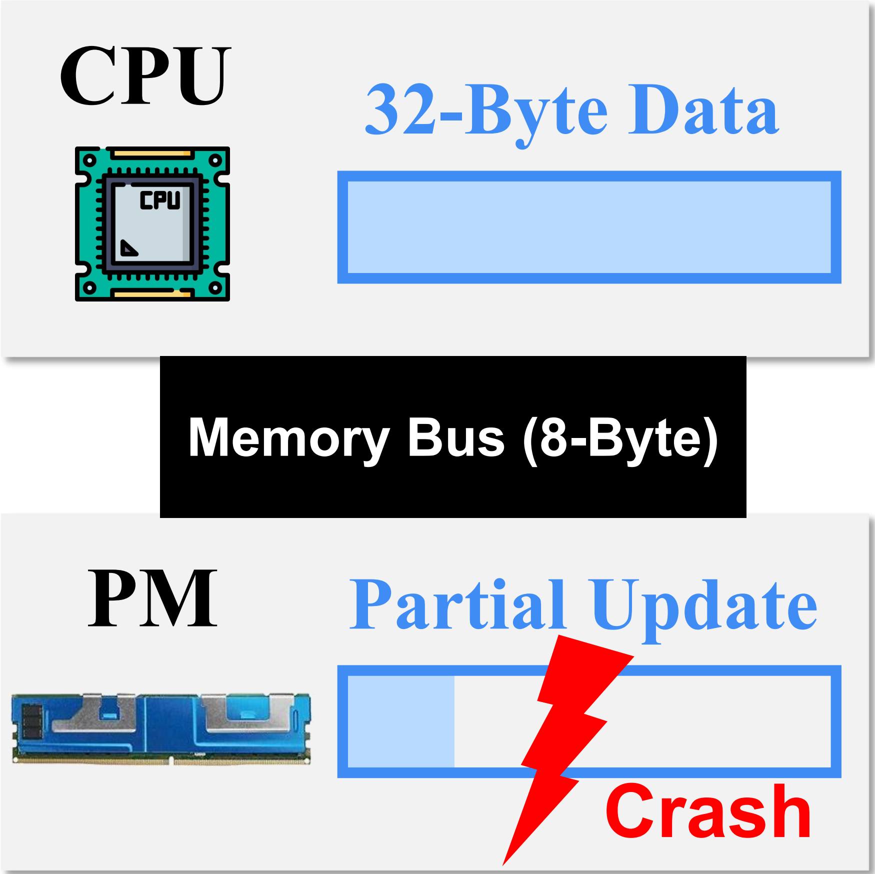


Logging

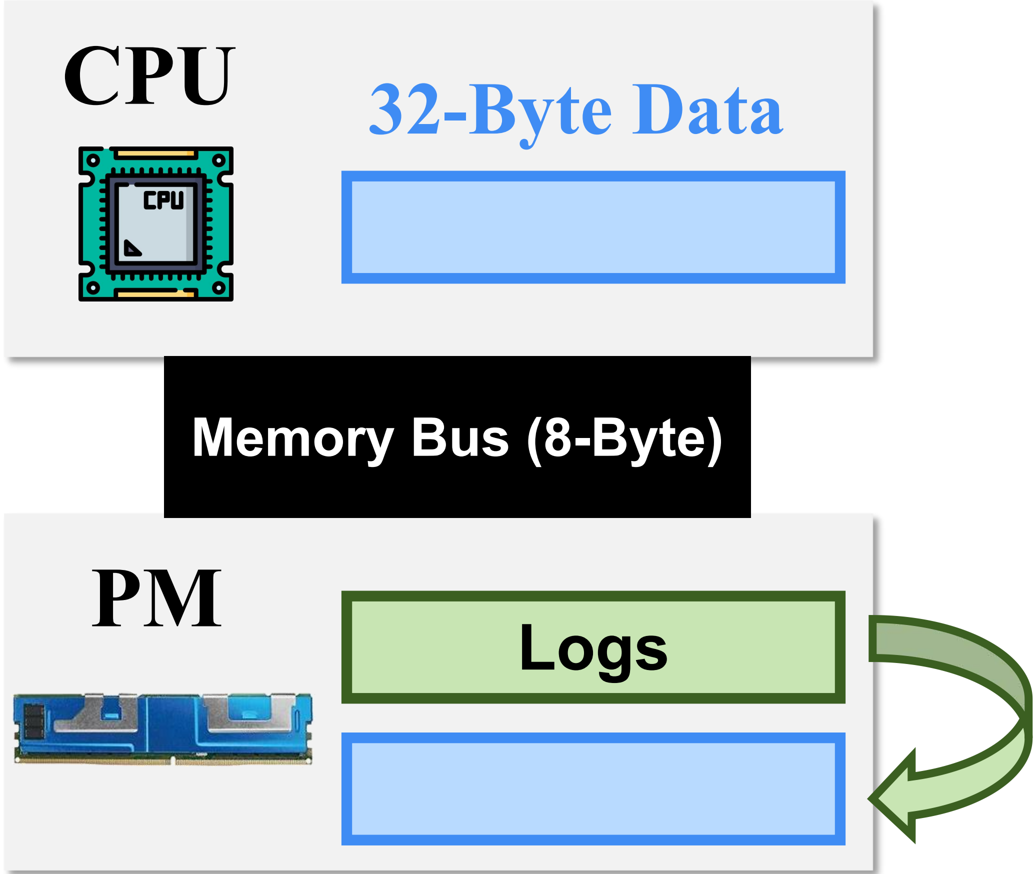


Data Inconsistency!

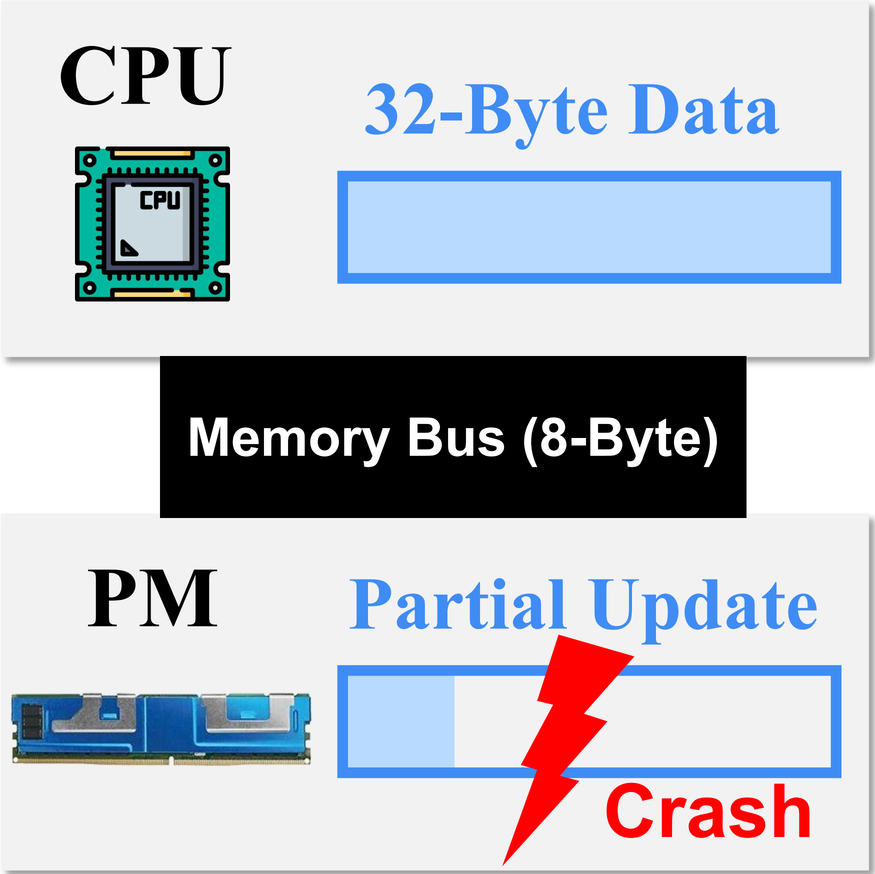
Challenge 2: Ensure Crash Consistency



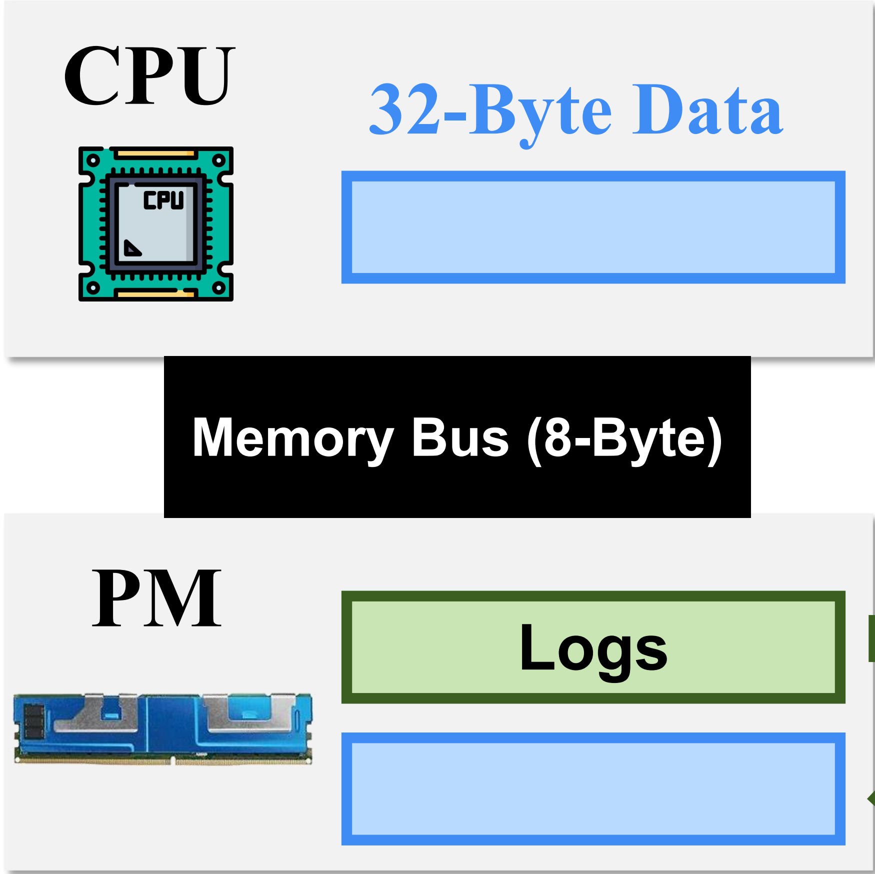
Logging



Challenge 2: Ensure Crash Consistency



Logging



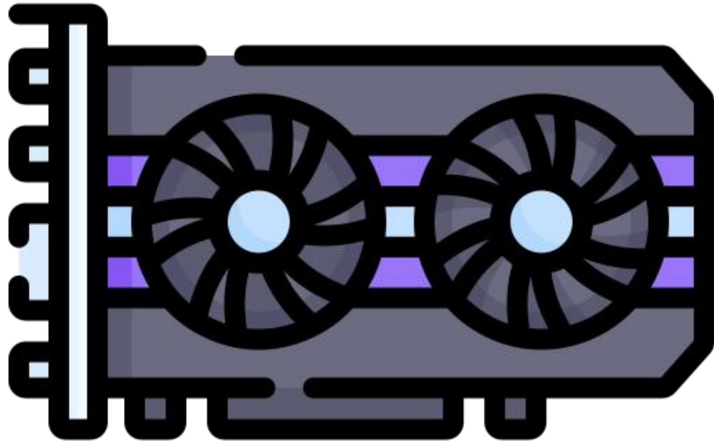
Challenge 2: Ensure Crash Consistency

Crash consistency guarantee introduces **High Overhead**

Challenge 3: Huge Bandwidth Gap

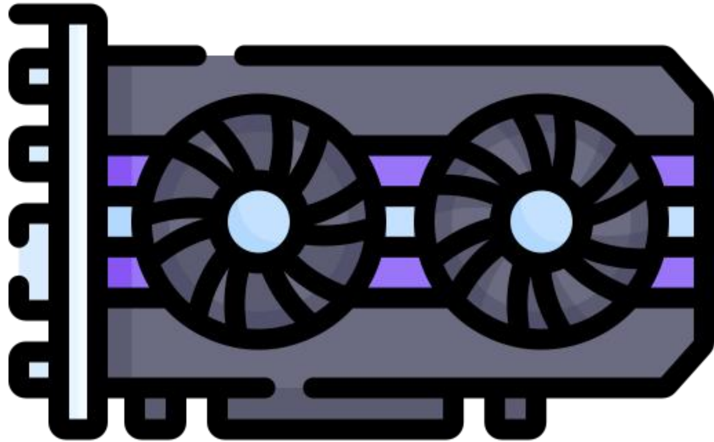
Challenge 3: Huge Bandwidth Gap

GPU

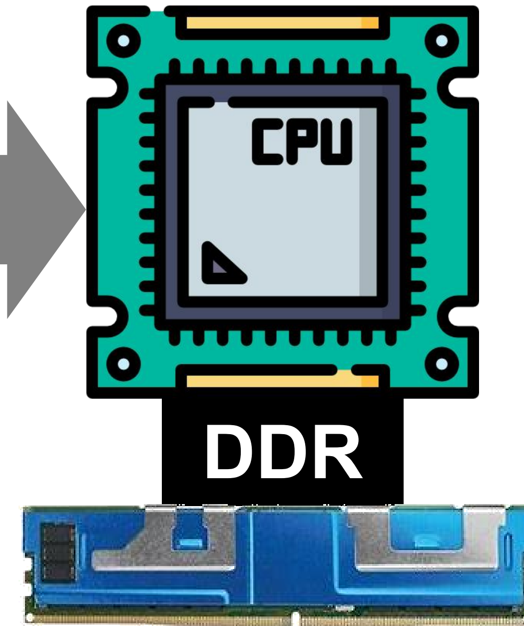


Challenge 3: Huge Bandwidth Gap

GPU



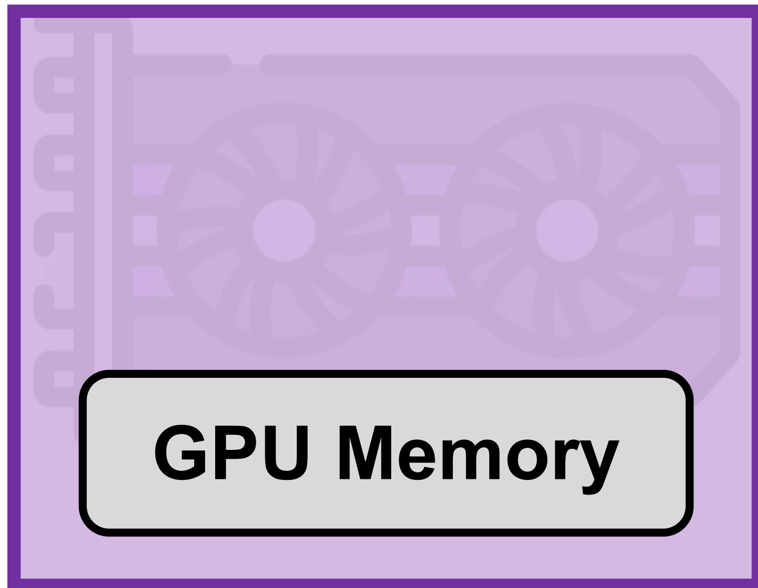
CPU



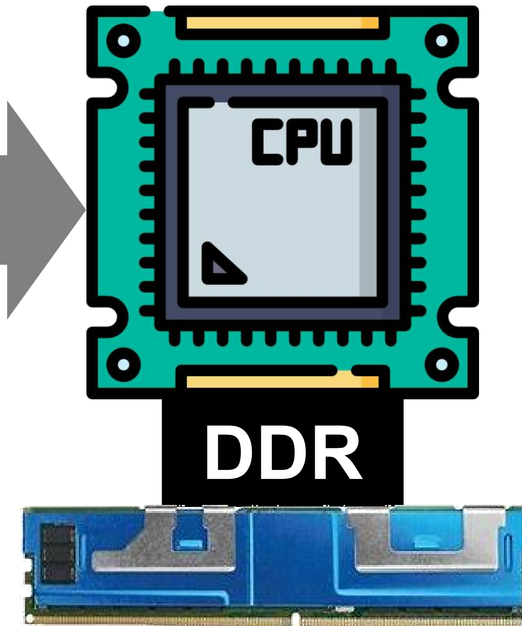
Persistent Memory

Challenge 3: Huge Bandwidth Gap

GPU



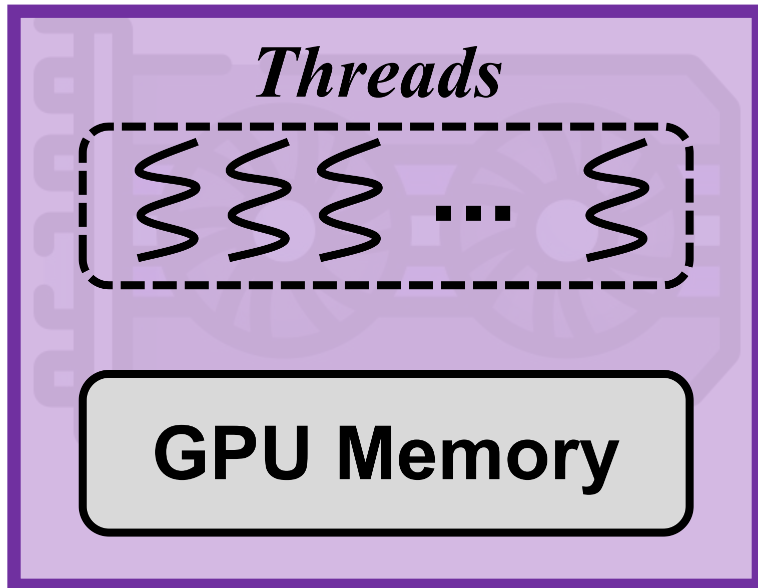
CPU



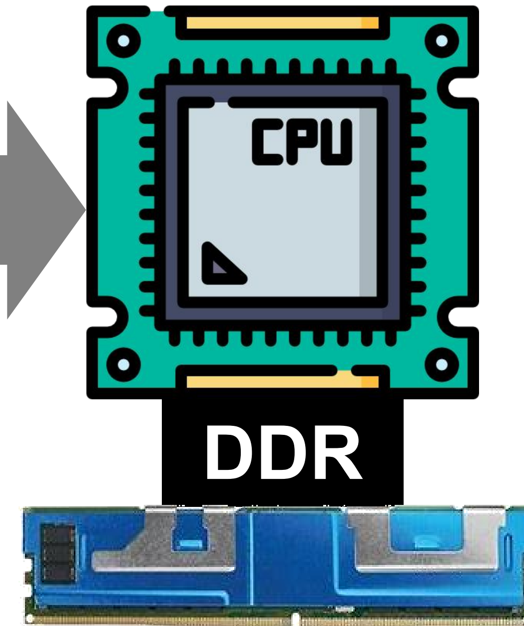
Persistent Memory

Challenge 3: Huge Bandwidth Gap

GPU



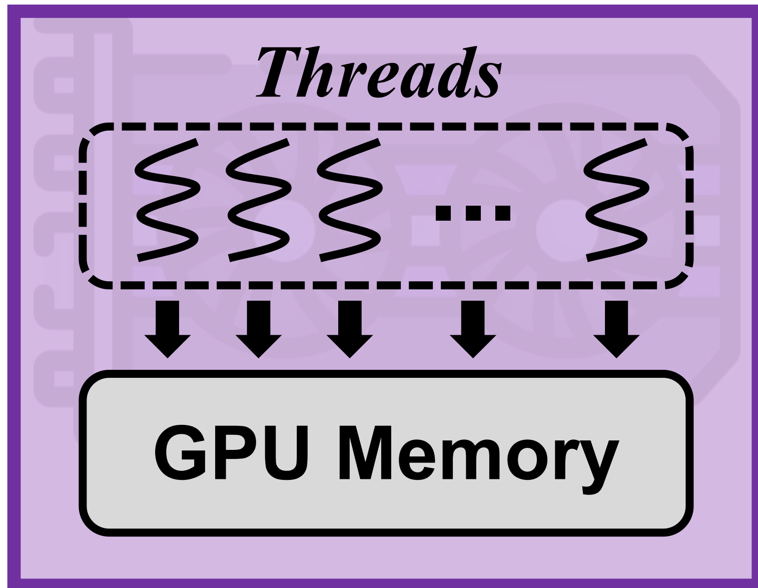
CPU



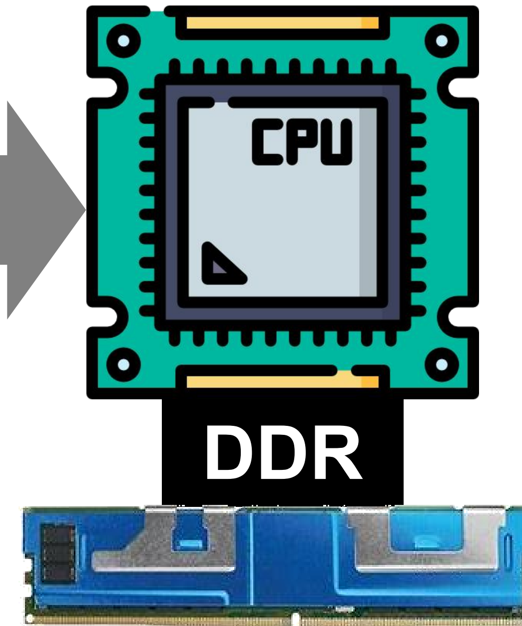
Persistent Memory

Challenge 3: Huge Bandwidth Gap

GPU



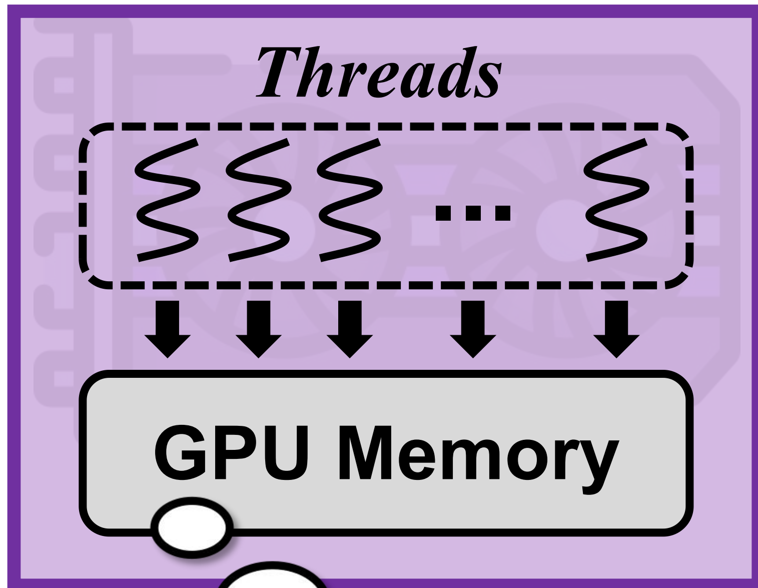
CPU



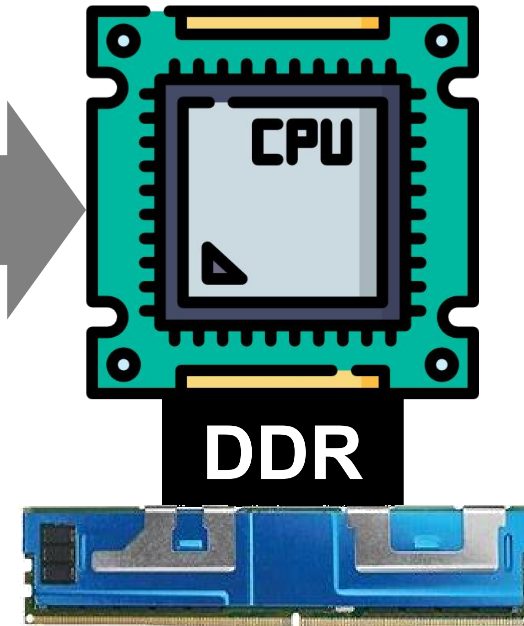
Persistent Memory

Challenge 3: Huge Bandwidth Gap

GPU



CPU

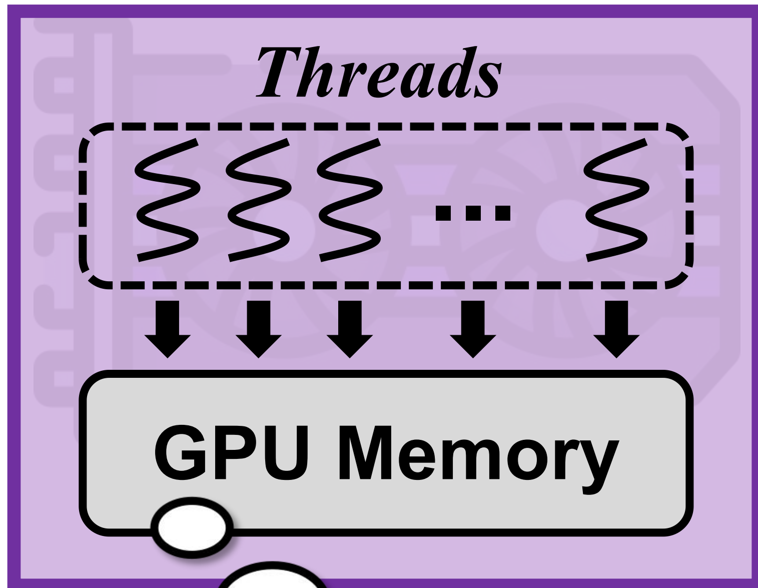


e.g., ***900 GB/s*** for NVIDIA V100

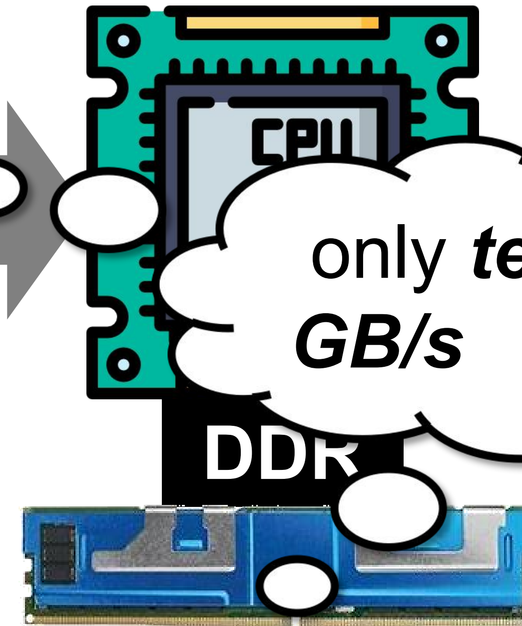
Persistent Memory

Challenge 3: Huge Bandwidth Gap

GPU



CPU

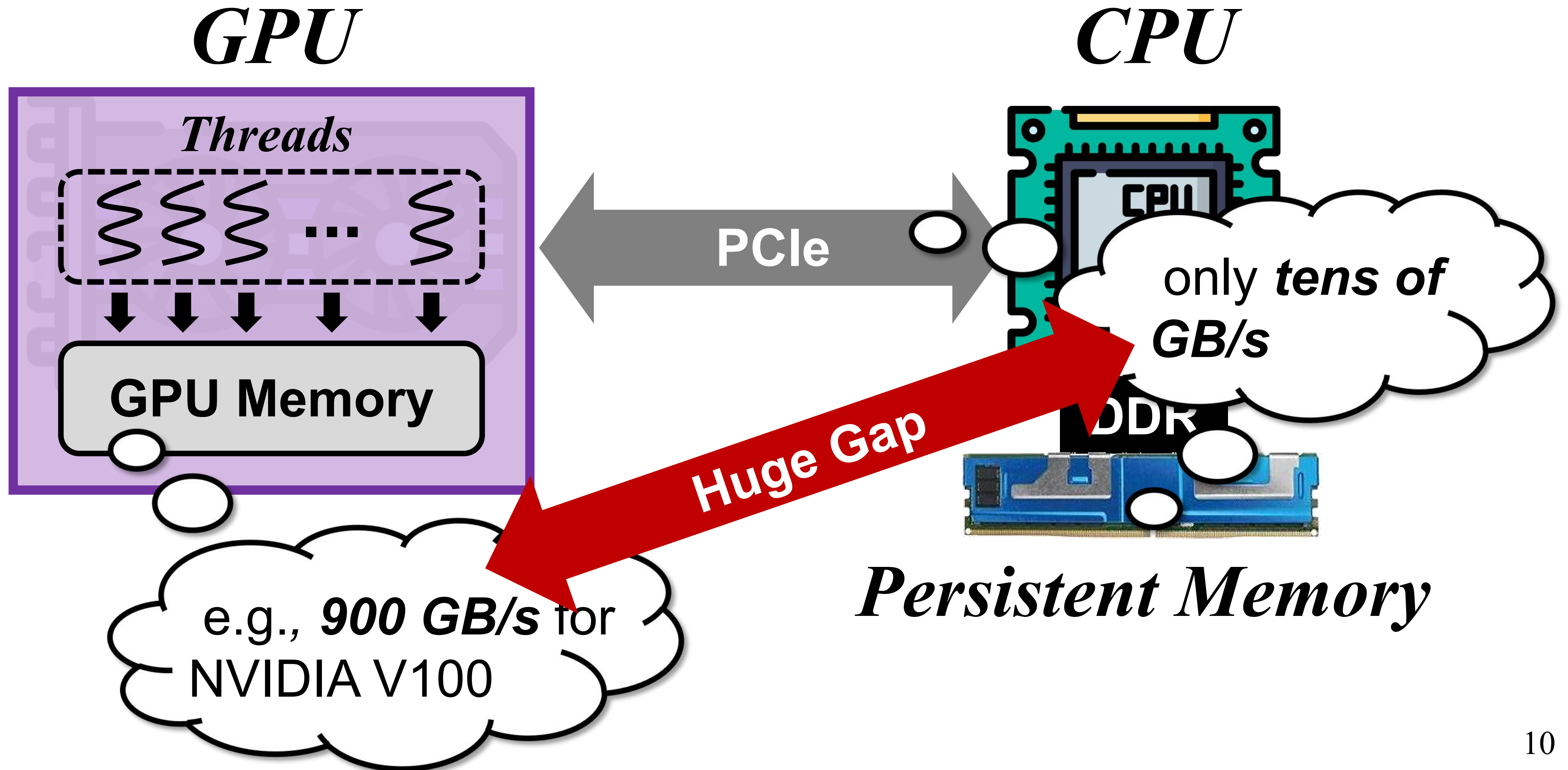


only ***tens of GB/s***

e.g., ***900 GB/s*** for NVIDIA V100

Persistent Memory

Challenge 3: Huge Bandwidth Gap



Challenge 3: Huge Bandwidth Gap

GPU

CPU

Threads

GPU Memory

Huge Gap

DDR

e.g., ***900 GB/s*** for NVIDIA V100

Persistent Memory

Huge bandwidth gap between PM and GPU **Limits the Utilization** of GPU's high parallelism

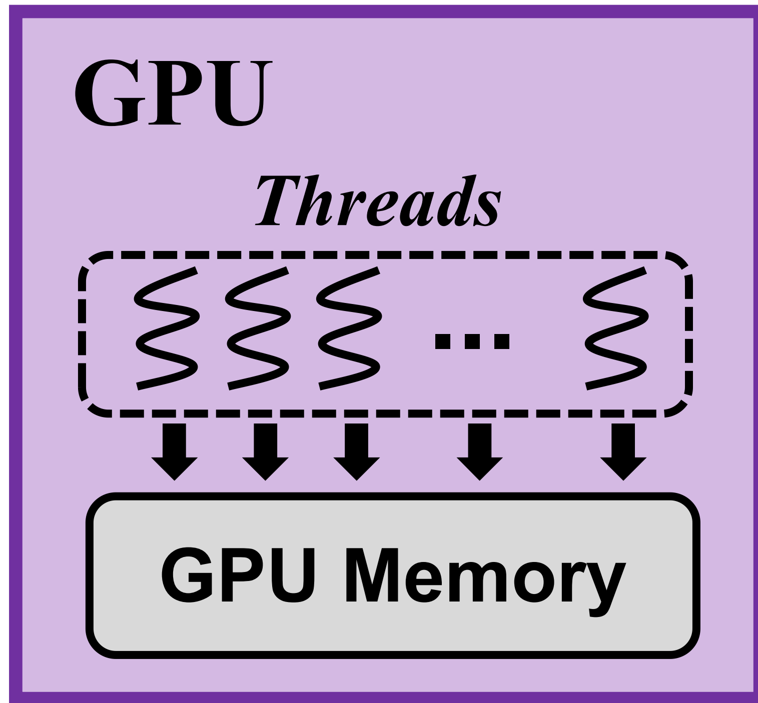
Our Solution: GPHash

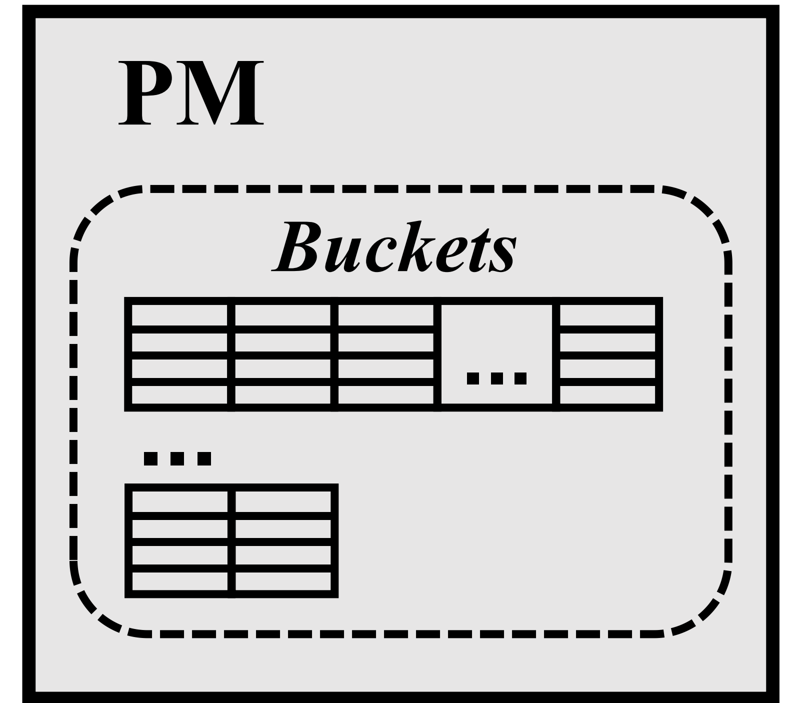
Our Solution: GPHash

- **GPHash:** ***An Efficient Hash Index for GPM Systems***

Our Solution: GPHash

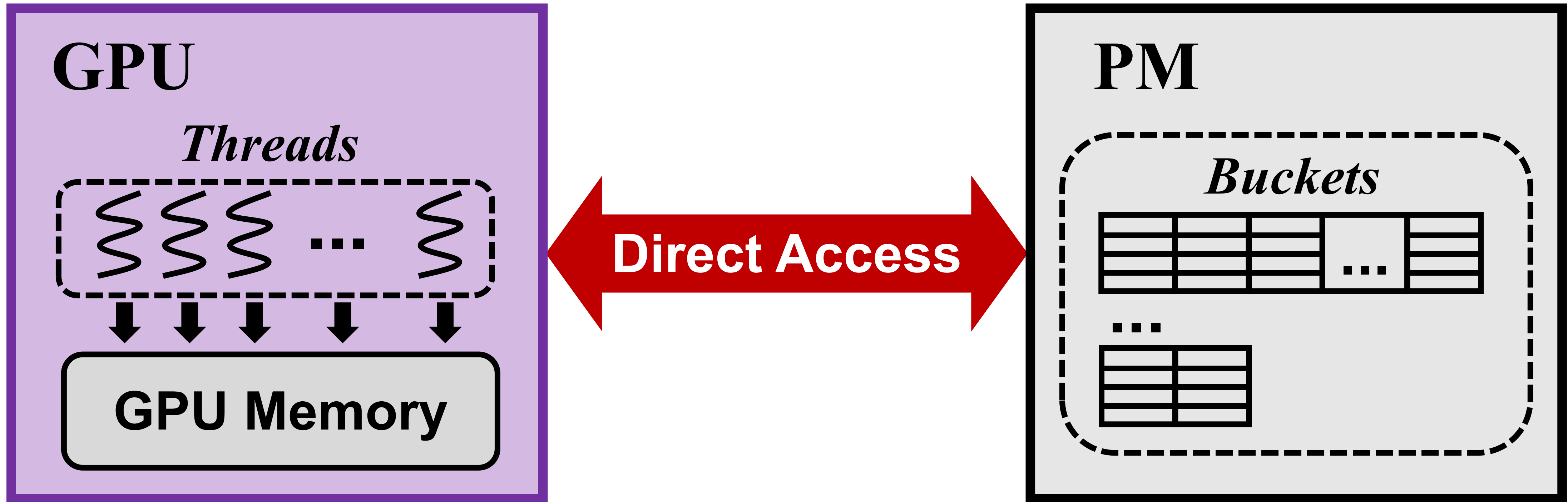
- **GPHash:** ***An Efficient Hash Index for GPM Systems***





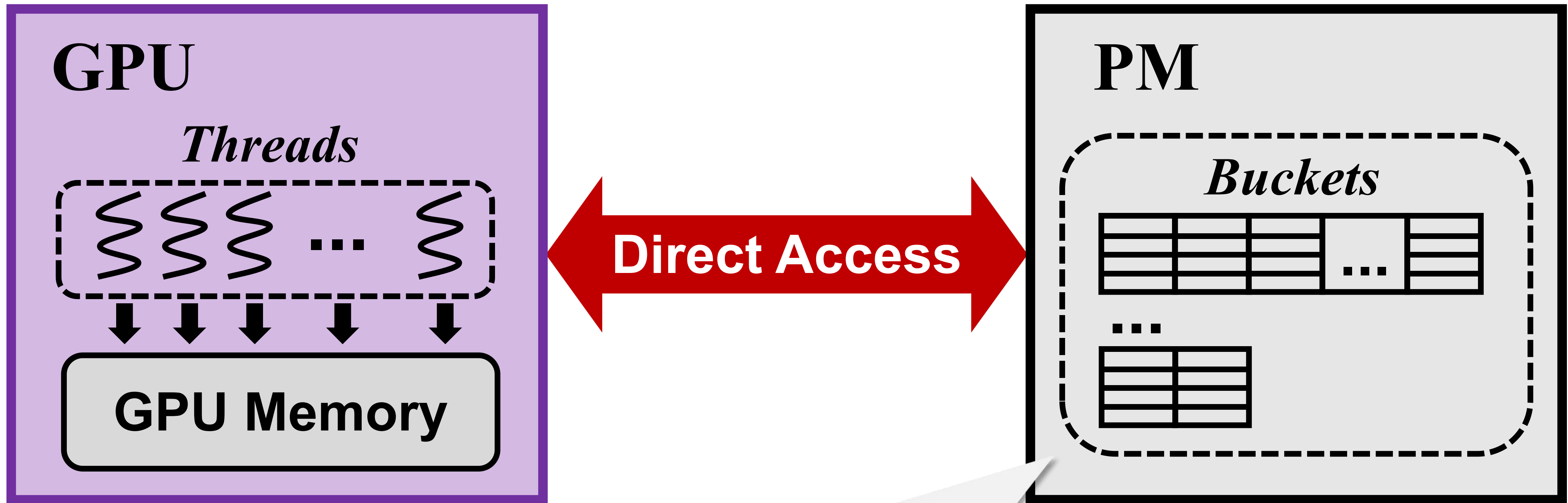
Our Solution: GPHash

- **GPHash:** ***An Efficient Hash Index for GPM Systems***



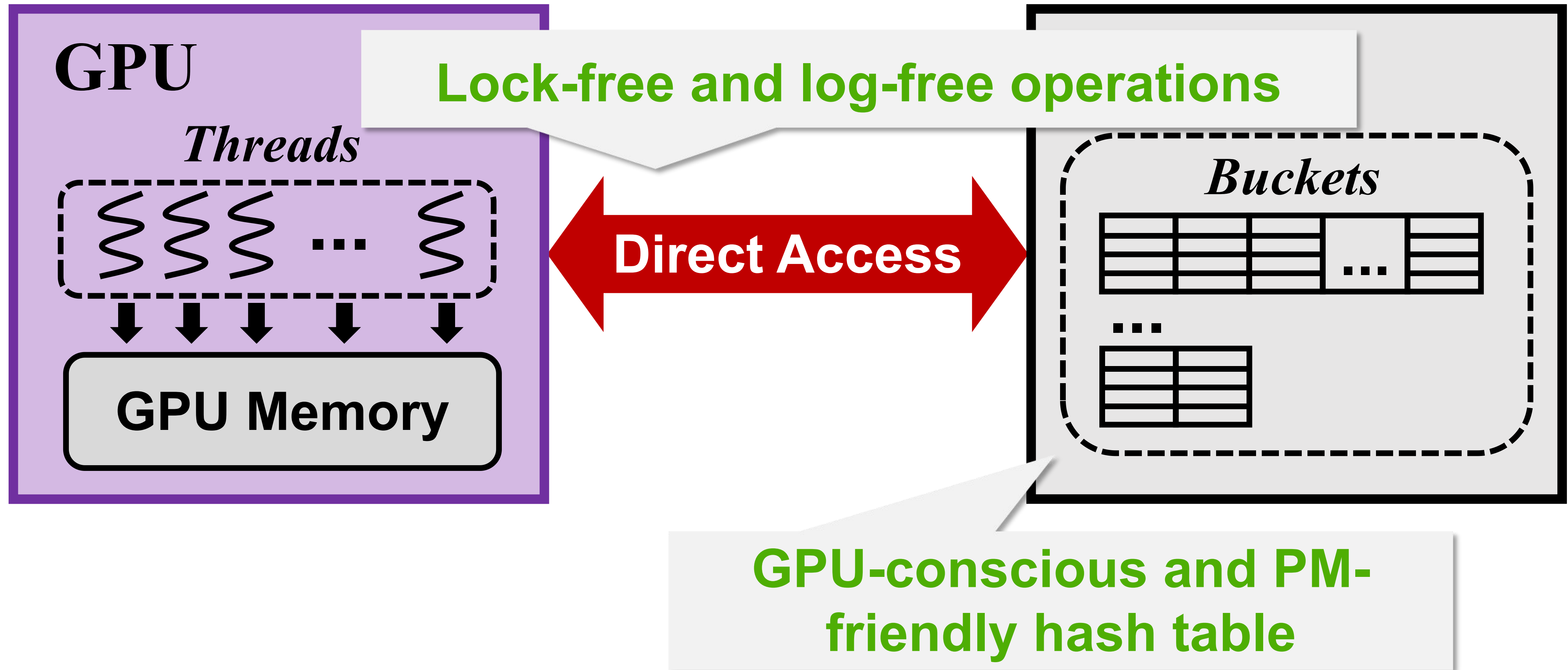
Our Solution: GPHash

➢ **GPHash:** ***An Efficient Hash Index for GPM Systems***



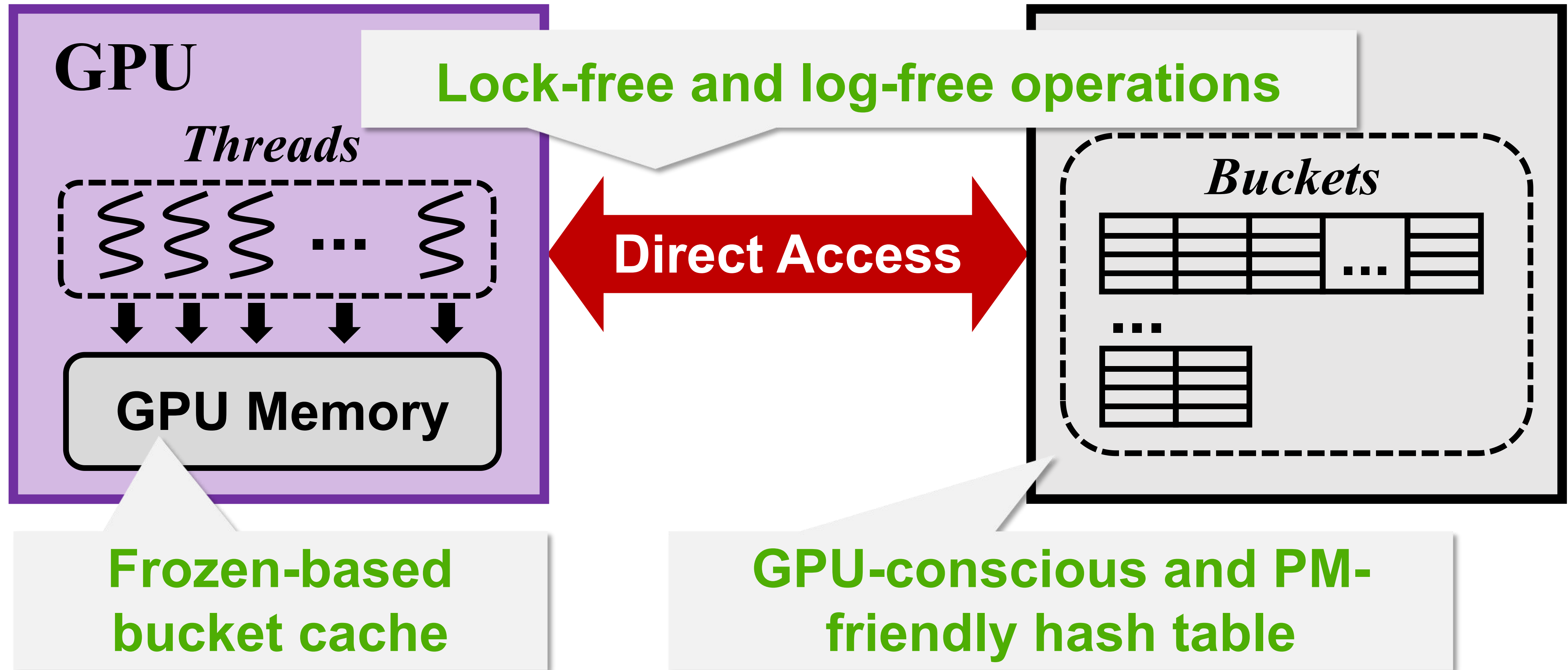
Our Solution: GPHash

- **GPHash:** ***An Efficient Hash Index for GPM Systems***



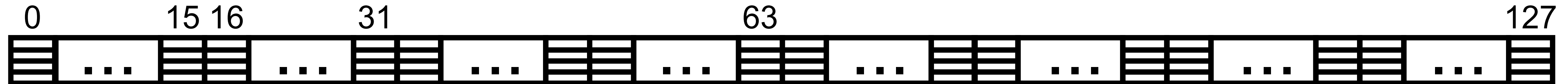
Our Solution: GPHash

➢ **GPHash:** ***An Efficient Hash Index for GPM Systems***

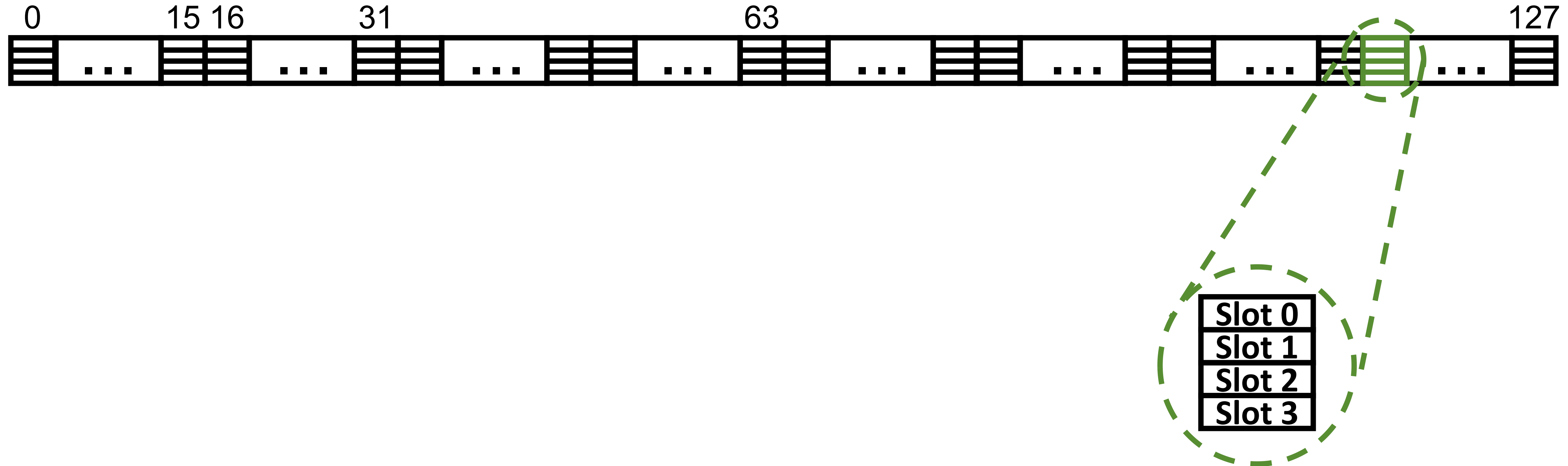


GPU-Conscious and PM-Friendly Hash Table

GPU-Conscious and PM-Friendly Hash Table

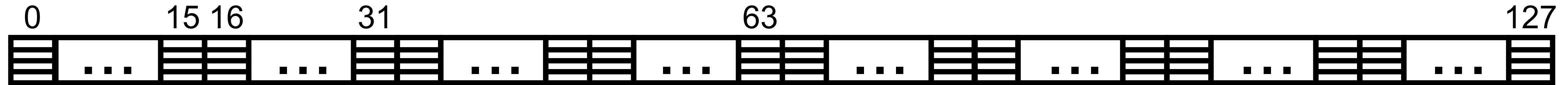


GPU-Conscious and PM-Friendly Hash Table

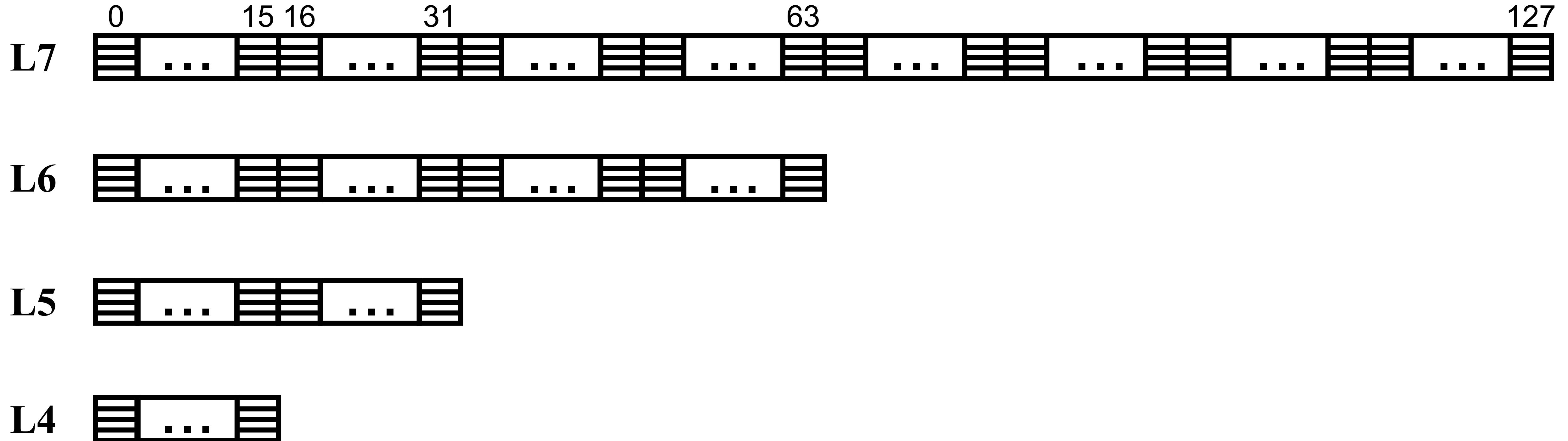


- *Slot Associativity*

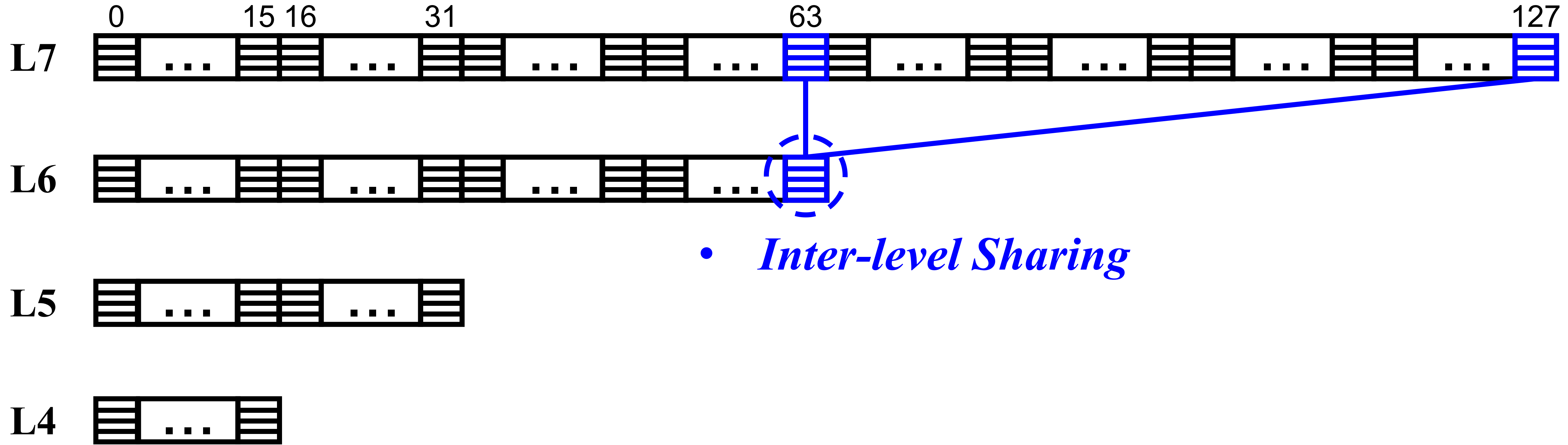
GPU-Conscious and PM-Friendly Hash Table



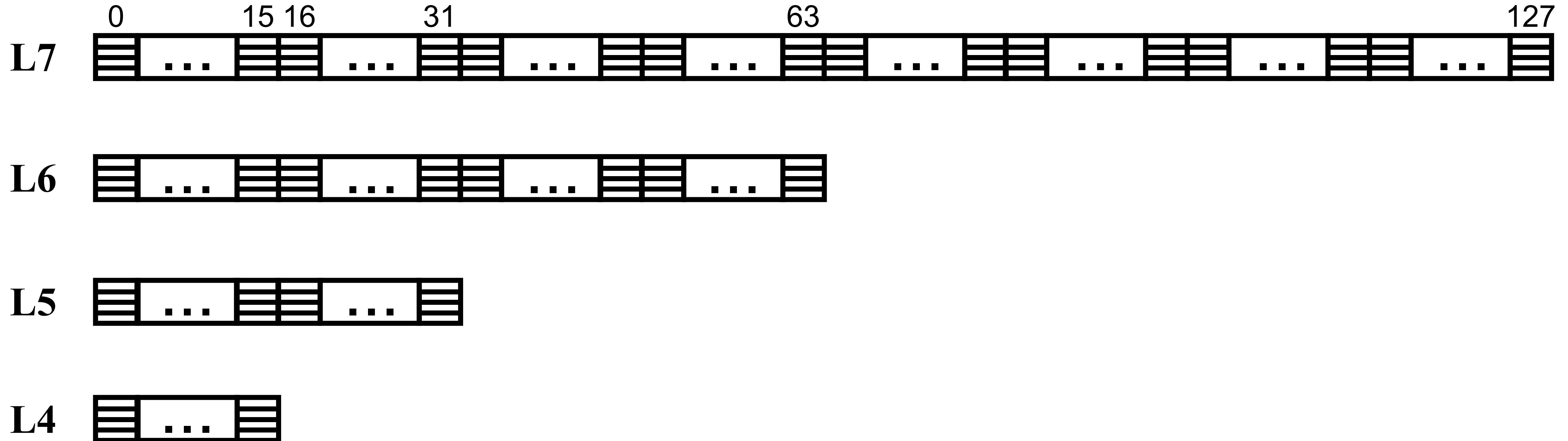
GPU-Conscious and PM-Friendly Hash Table



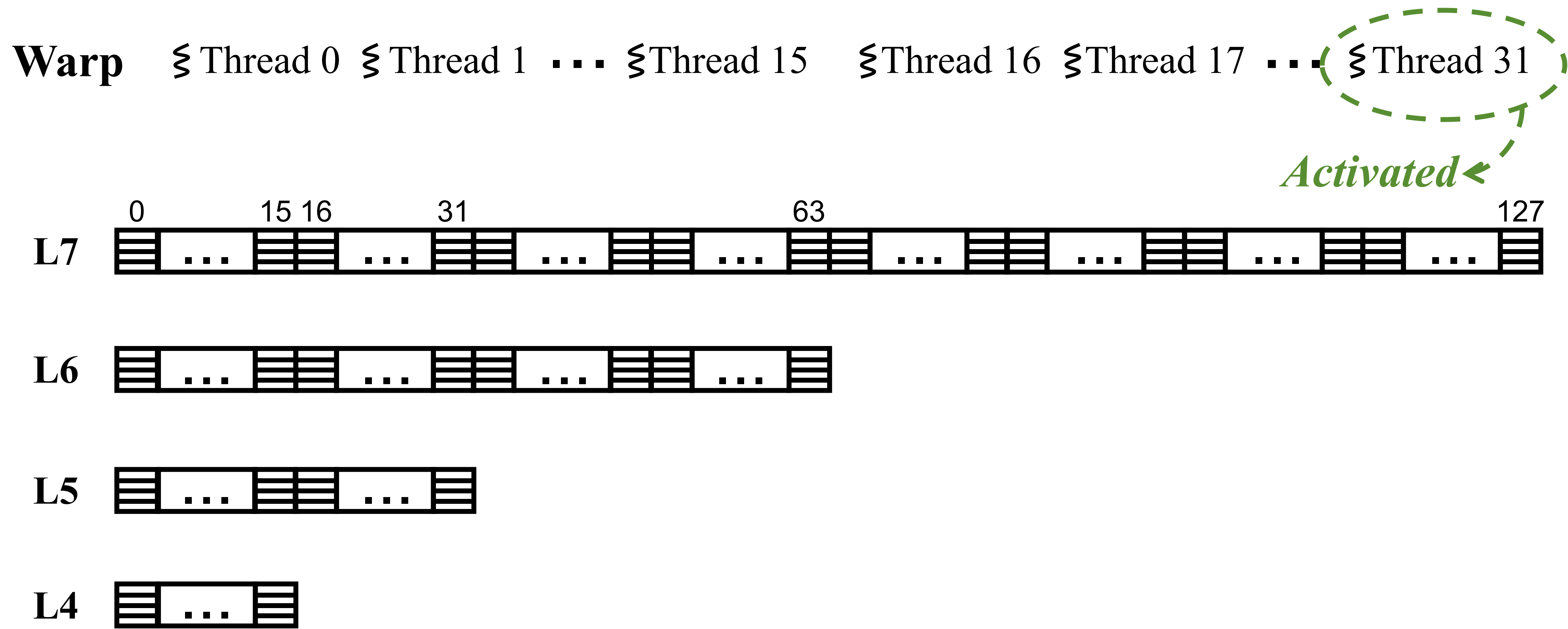
GPU-Conscious and PM-Friendly Hash Table



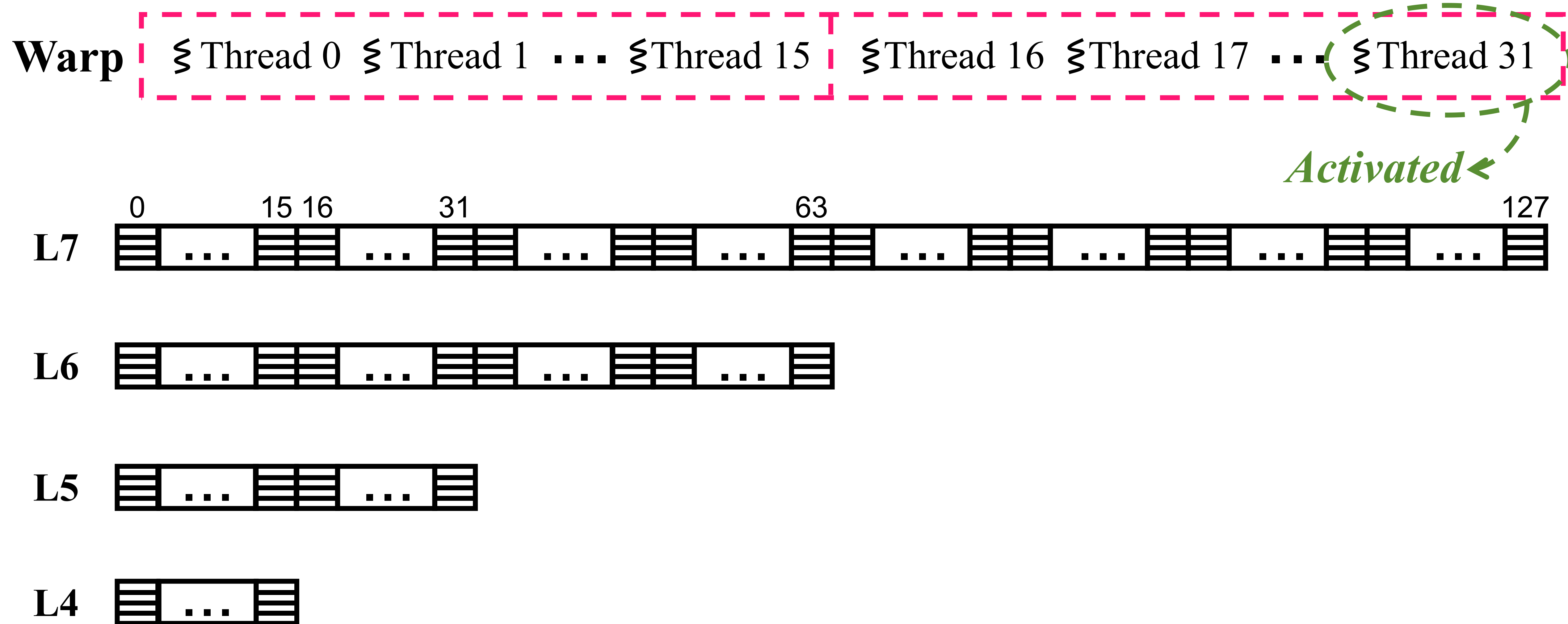
GPU-Conscious and PM-Friendly Hash Table



GPU-Conscious and PM-Friendly Hash Table



GPU-Conscious and PM-Friendly Hash Table



GPU-Conscious and PM-Friendly Hash Table

Warp | Thread 0 | Thread 1 ... Thread 15 | Thread 16 | Thread 17 ... Thread 31

$Hash_1(Key_{31})$

$Hash_2(Key_{31})$

Activated

0 15 16 31 63 127

L7

L6

L5

L4

Candidate buckets in one-shot warp access

GPU-Conscious and PM-Friendly Hash Table

Warp | Thread 0 | Thread 1 ... Thread 15 | Thread 16 | Thread 17 ... Thread 31

Activated

$Hash_1(Key_{31})$

$Hash_2(Key_{31})$

0 15 16 31 63 127

L7

L6

L5

L4

Candidate buckets in one-shot warp access

- *Multiple Hash Locations*
- *One-Shot Warp Access*

GPU-Conscious and PM-Friendly Hash Table

Warp: Thread 0, Thread 1 ... Thread 15 | Thread 16, Thread 17 ... Thread 31

Hash₁(Key₃₁)   *Hash₂(Key₃₁)*   *Activated*

0 15 16 31 63 127

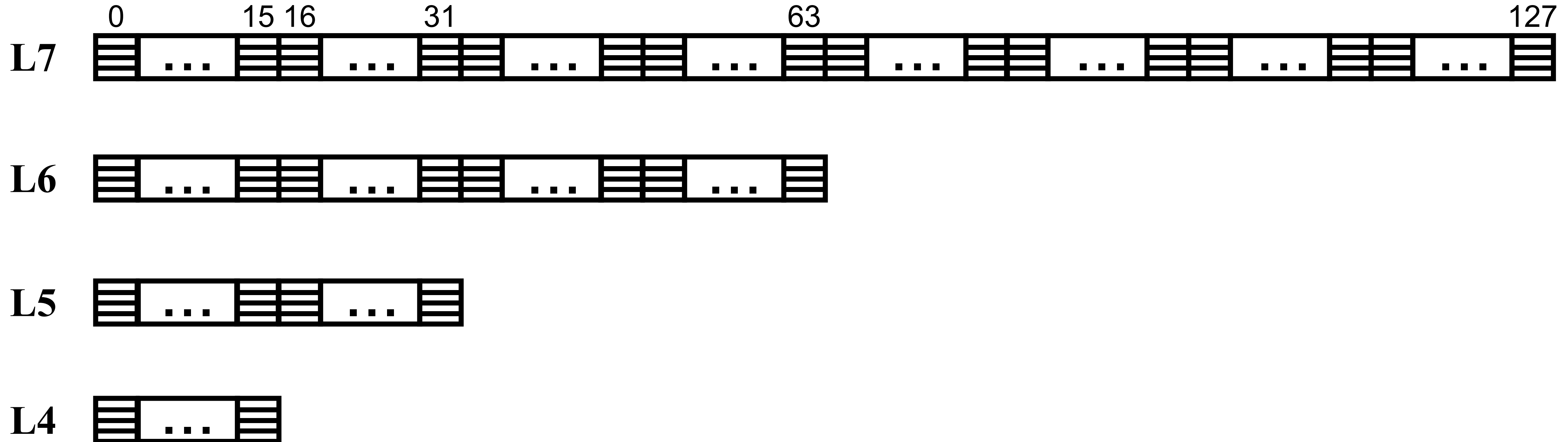
L7
L6
L5
L4

Candidate buckets in one-shot warp access

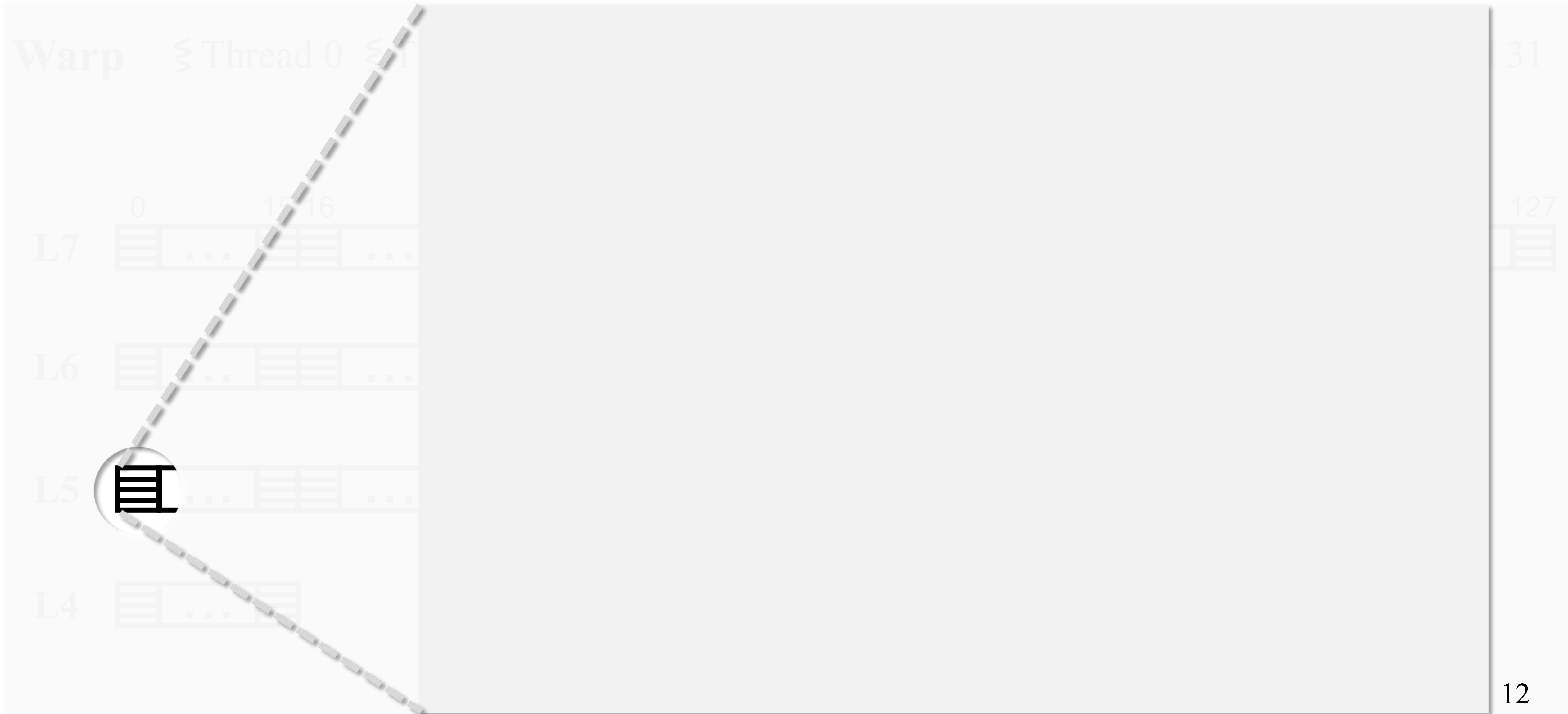
- *Multiple Hash Locations*
- *One-Shot Warp Access*

32x → 1x

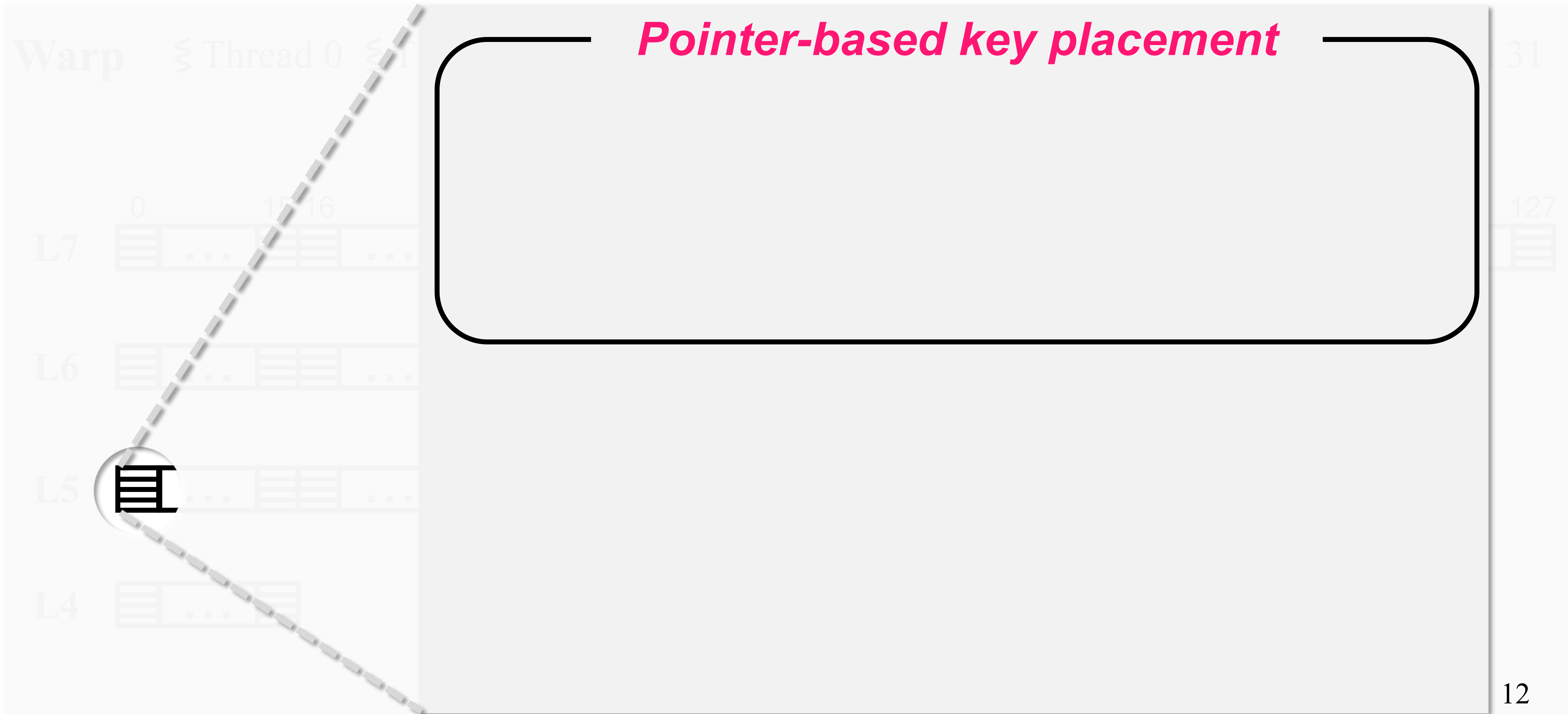
GPU-Conscious and PM-Friendly Hash Table



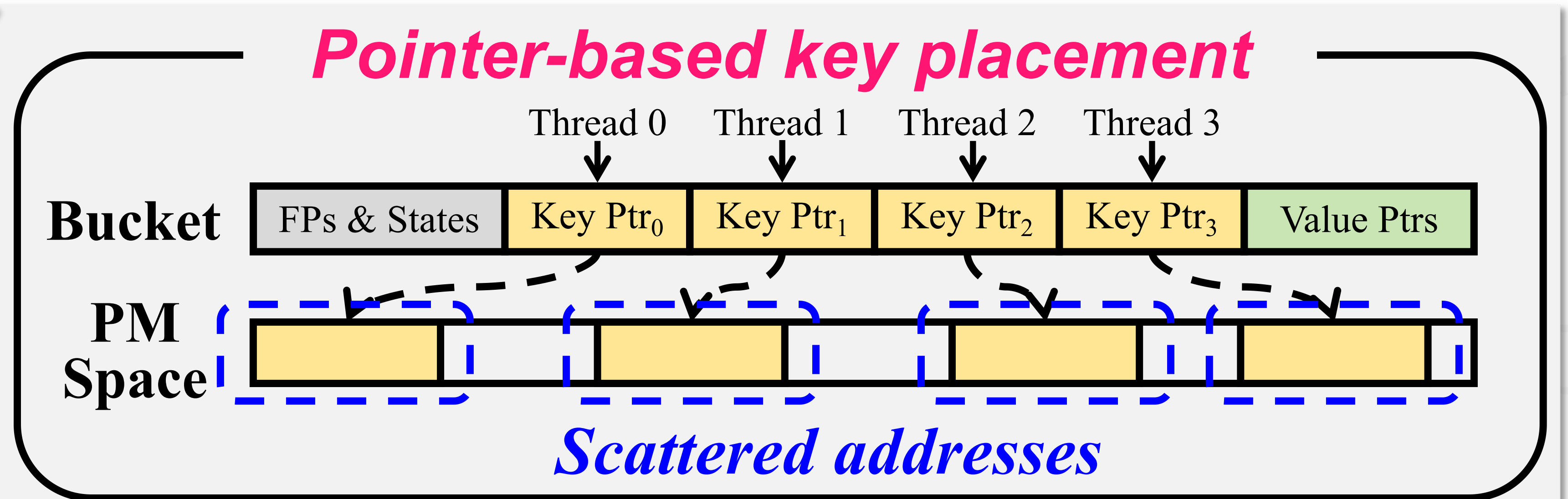
GPU-Conscious and PM-Friendly Hash Table



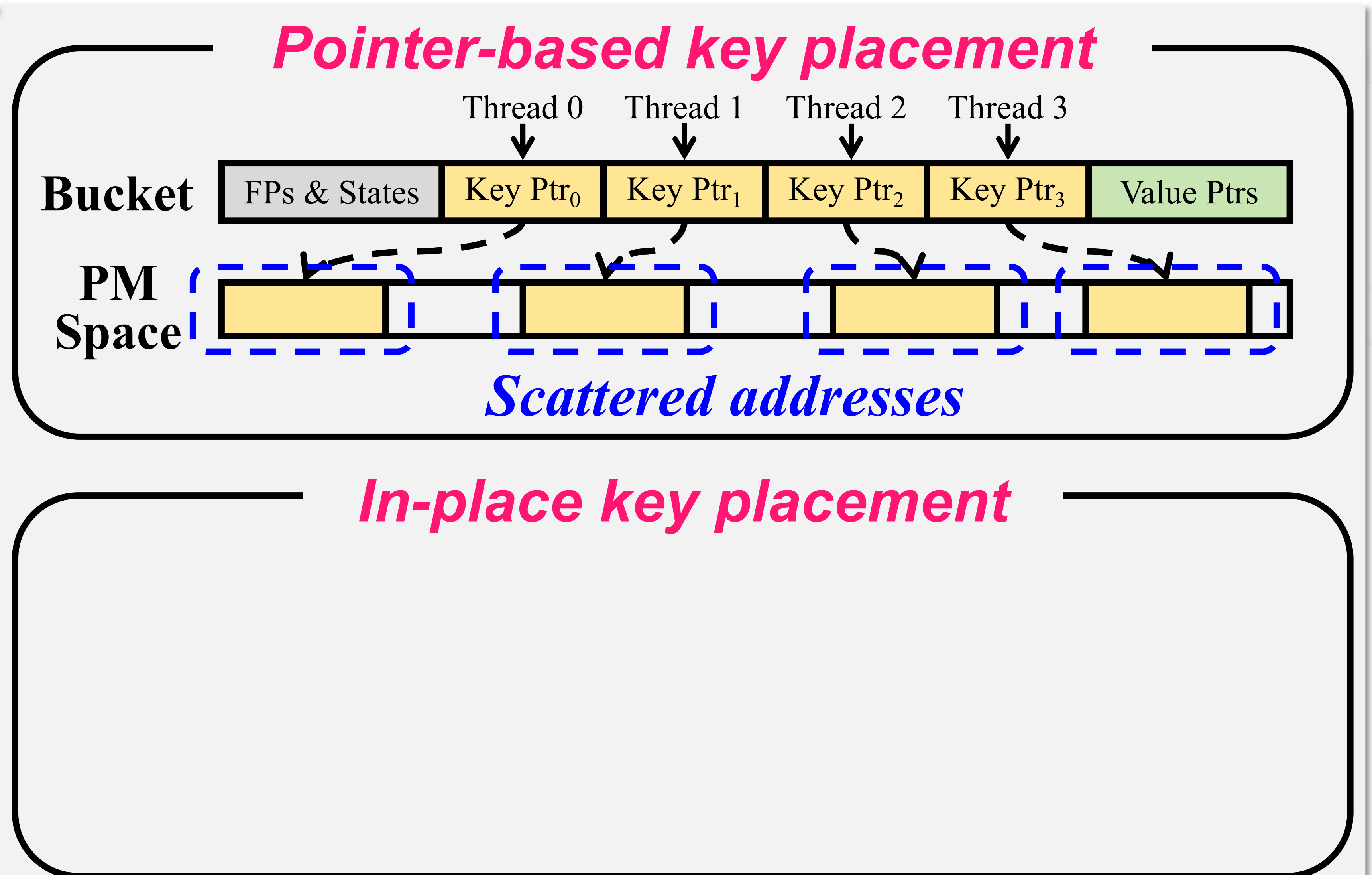
GPU-Conscious and PM-Friendly Hash Table



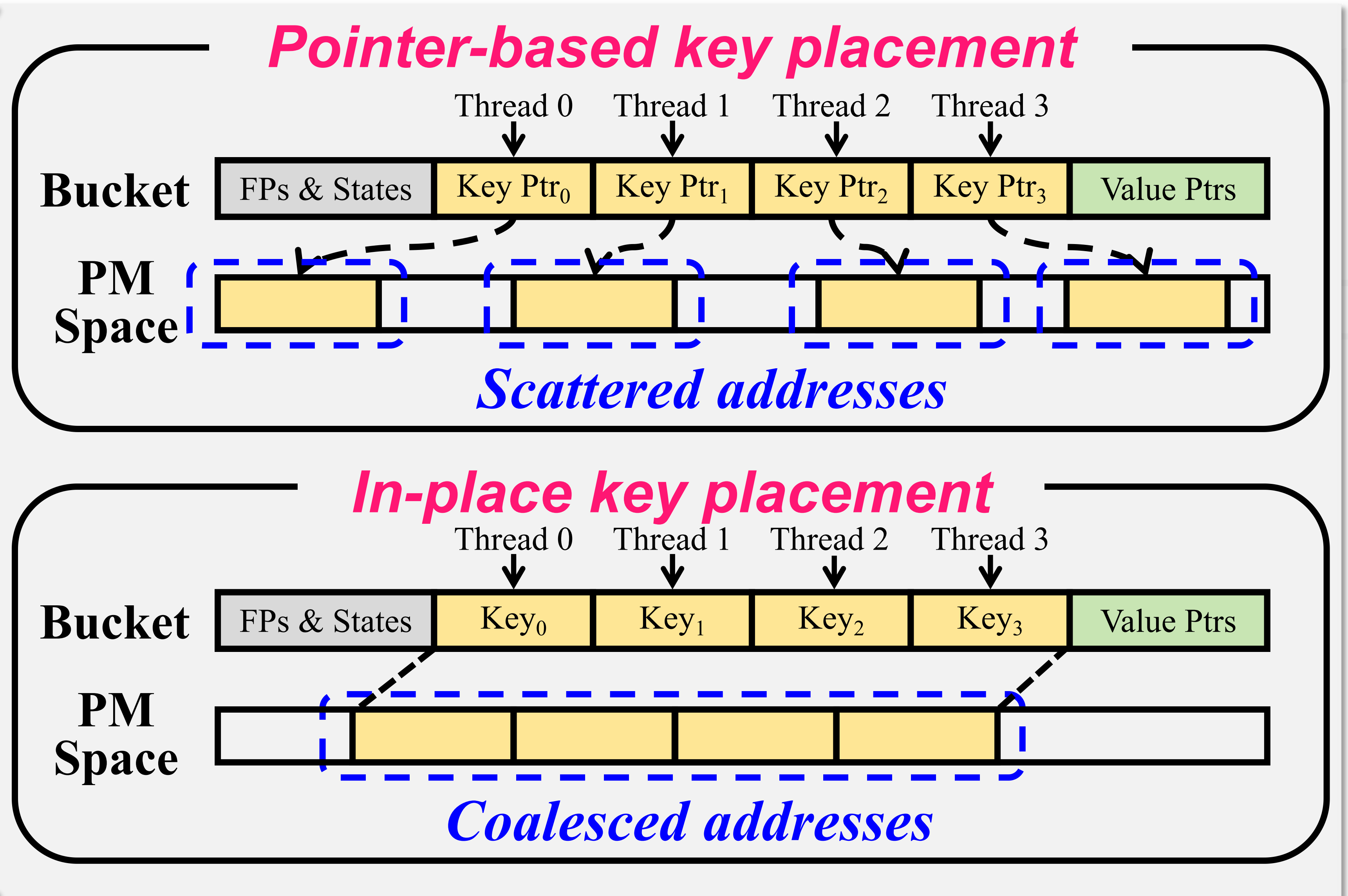
GPU-Conscious and PM-Friendly Hash Table



GPU-Conscious and PM-Friendly Hash Table

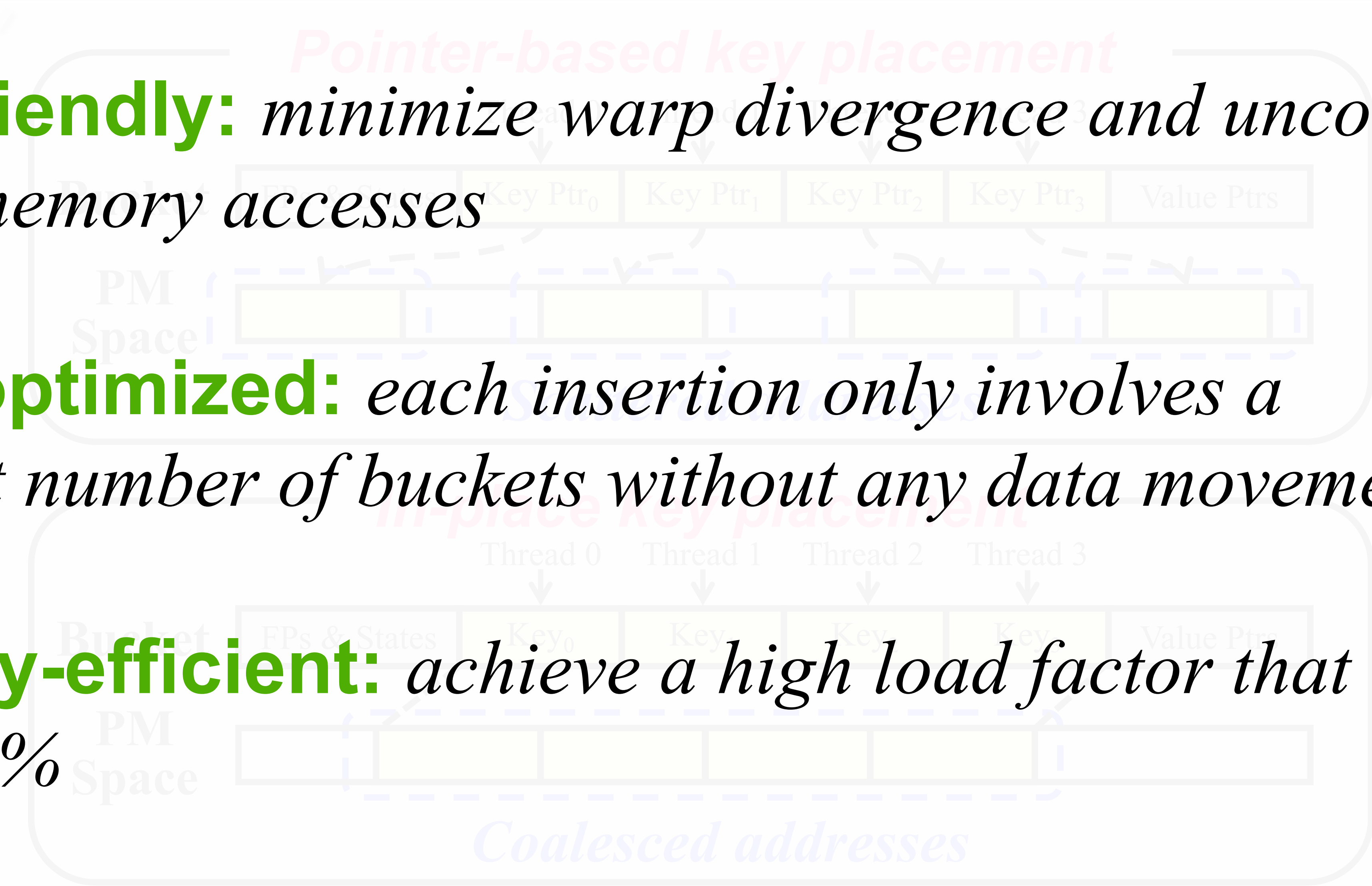


GPU-Conscious and PM-Friendly Hash Table



GPU-Conscious and PM-Friendly Hash Table

GPU-Conscious and PM-Friendly Hash Table

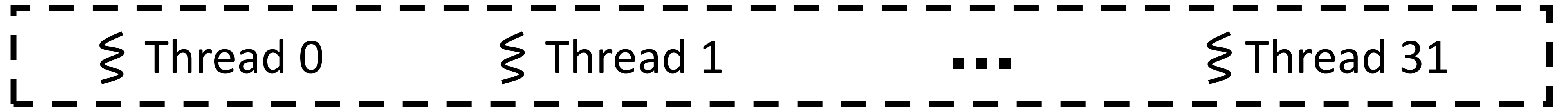


- **GPU-friendly:** *minimize warp divergence and uncoalesced memory accesses*
- **Write-optimized:** *each insertion only involves a constant number of buckets without any data movement*
- **Memory-efficient:** *achieve a high load factor that is up to 92%*

Lock-Free and Log-Free Insertion

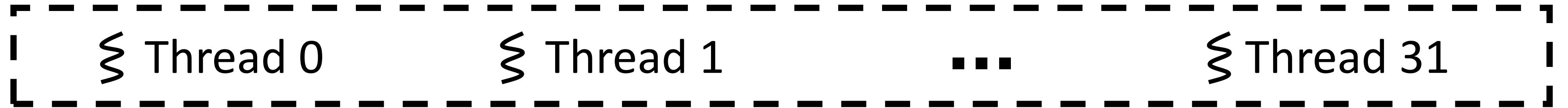
Lock-Free and Log-Free Insertion

Warp



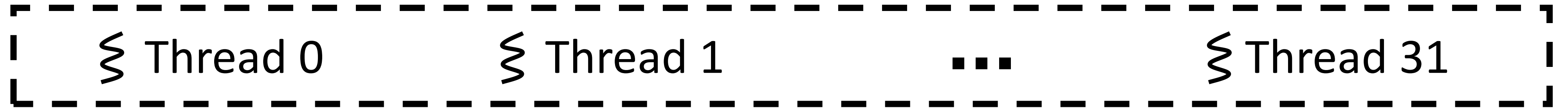
Lock-Free and Log-Free Insertion

Warp



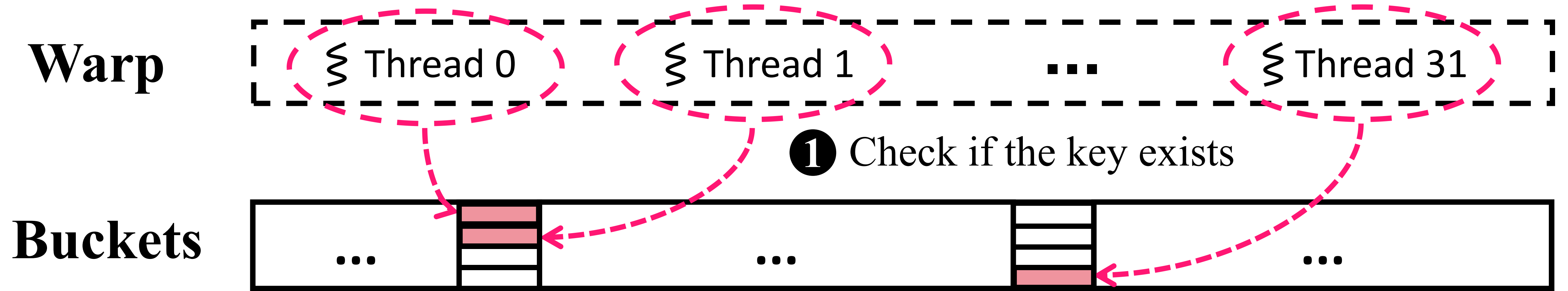
Buckets

Lock-Free and Log-Free Insertion

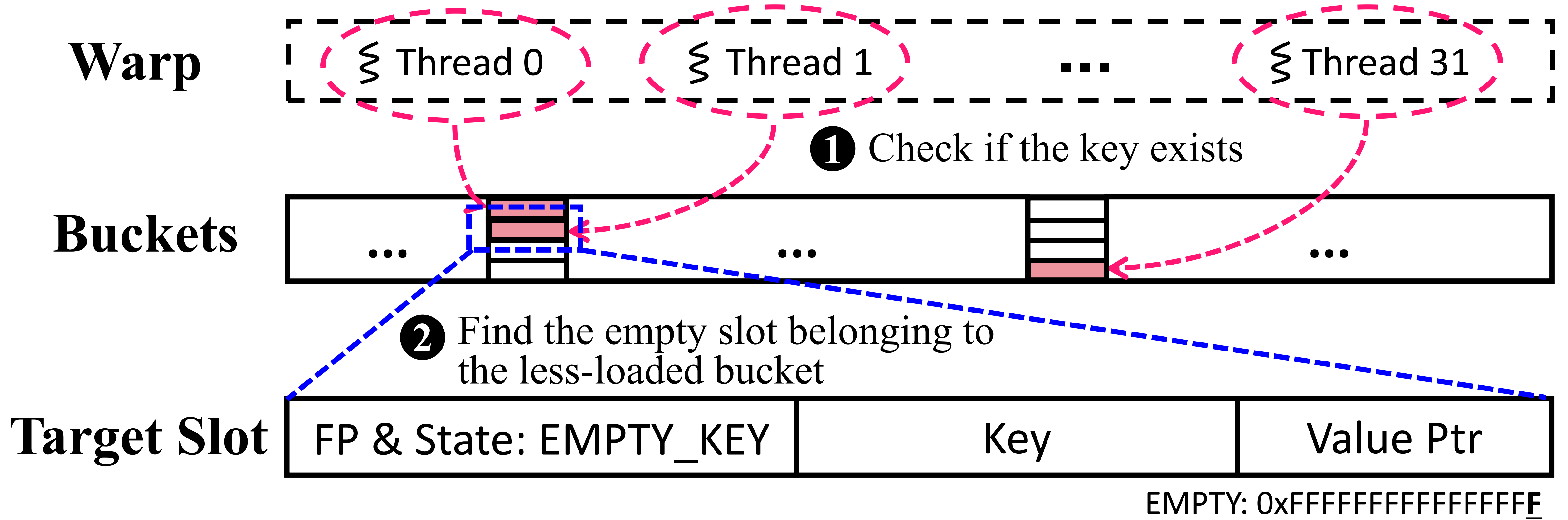


❶ Check if the key exists

Lock-Free and Log-Free Insertion

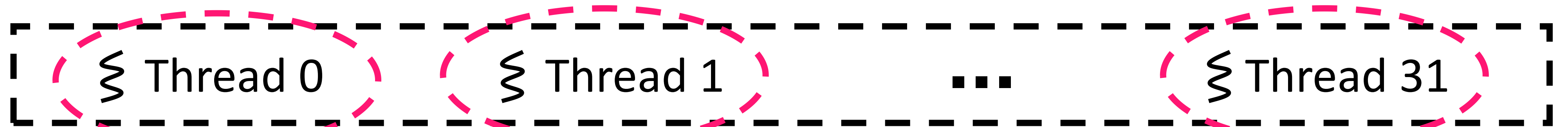


Lock-Free and Log-Free Insertion



Lock-Free and Log-Free Insertion

Warp



❶ Check if the key exists

Buckets

❷ Find the empty slot belonging to the less-loaded bucket

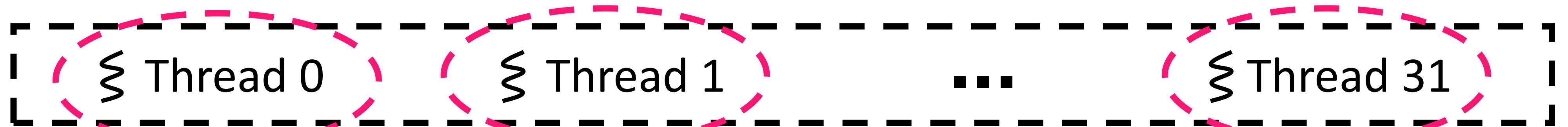
Target Slot

EMPTY: 0xFFFFFFFFFFFFFFFF

❸ Set the State to INSERTING using AtomicCAS operation

Lock-Free and Log-Free Insertion

Warp



❶ Check if the key exists

Buckets

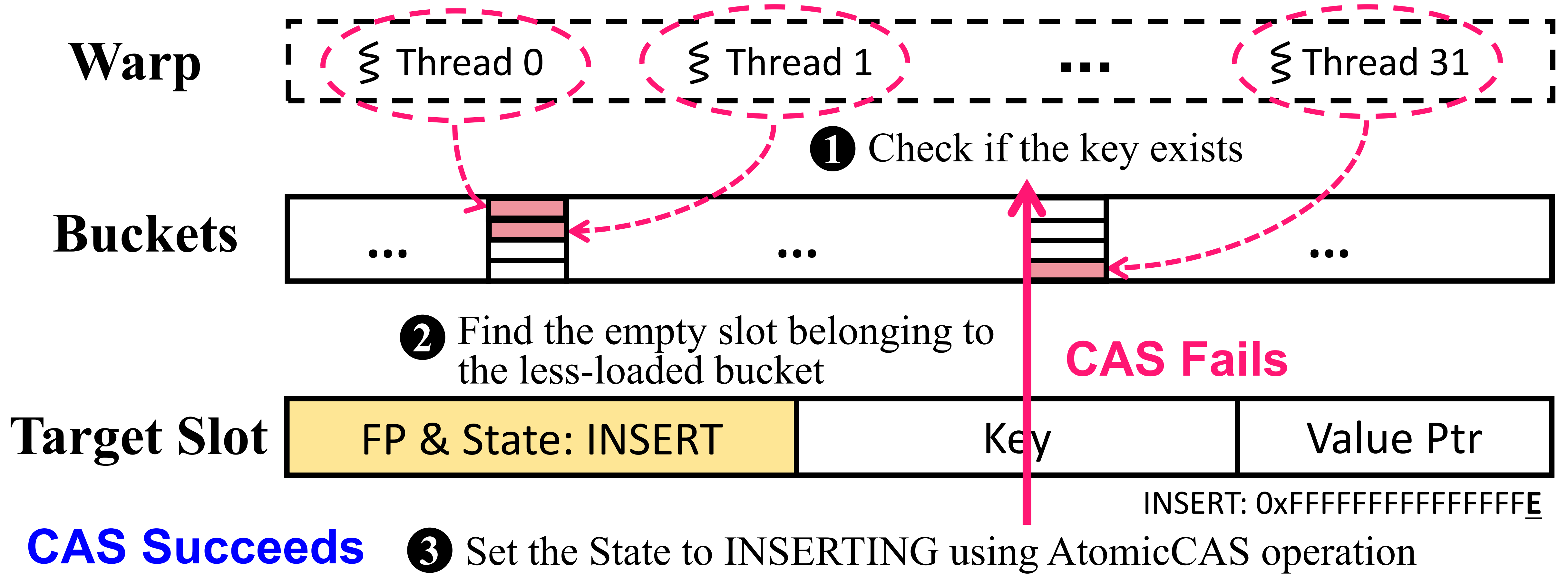
❷ Find the empty slot belonging to the less-loaded bucket

Target Slot

INSERT: 0xFFFFFFFFFFFFFFFE

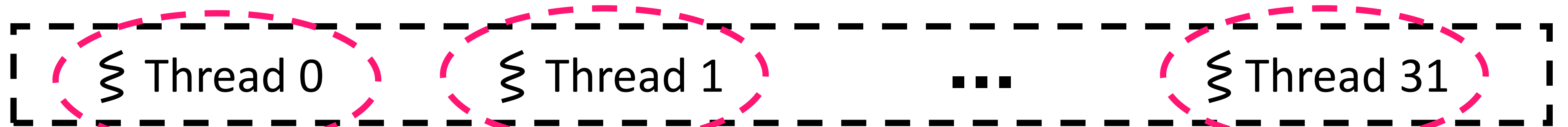
CAS Succeeds ❸ Set the State to INSERTING using AtomicCAS operation

Lock-Free and Log-Free Insertion



Lock-Free and Log-Free Insertion

Warp



❶ Check if the key exists

Buckets

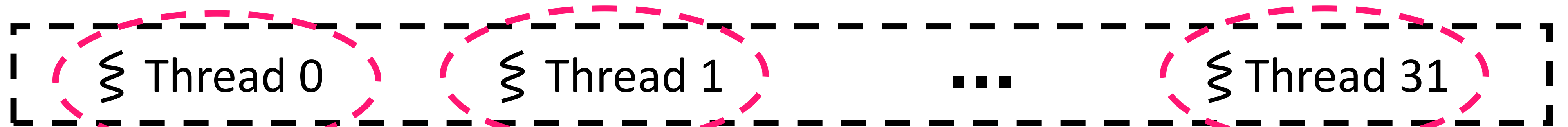
❷ Find the empty slot belonging to the less-loaded bucket

Target Slot

INSERT: 0xFFFFFFFFFFFFFFFE

Lock-Free and Log-Free Insertion

Warp



❶ Check if the key exists

Buckets

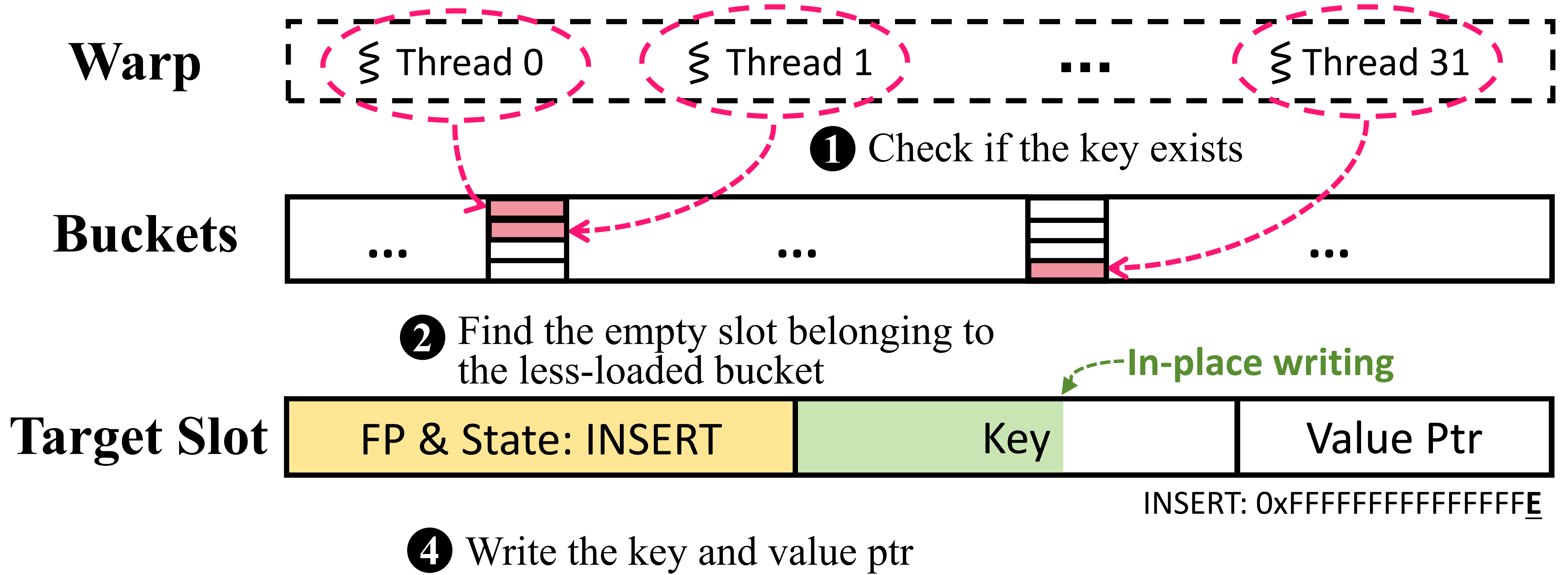
❷ Find the empty slot belonging to the less-loaded bucket

Target Slot

INSERT: 0xFFFFFFFFFFFFFFFE

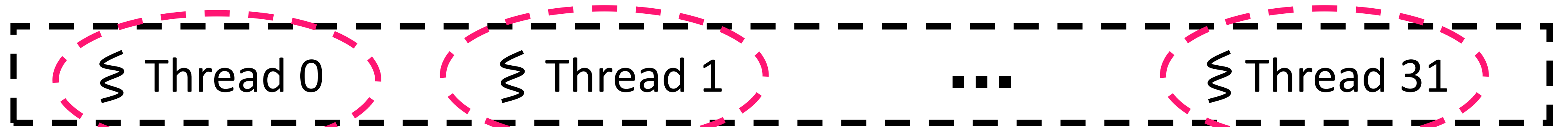
❹ Write the key and value ptr

Lock-Free and Log-Free Insertion



Lock-Free and Log-Free Insertion

Warp



❶ Check if the key exists

Buckets

❷ Find the empty slot belonging to the less-loaded bucket

In-place writing

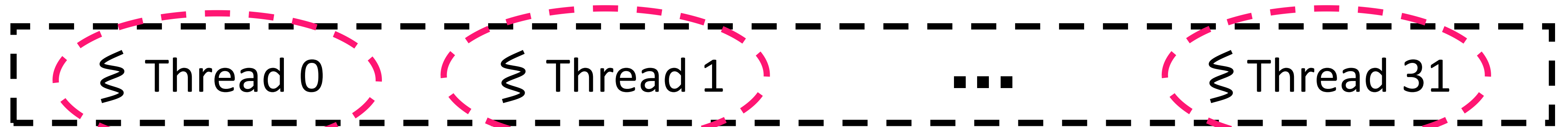
Target Slot

INSERT: 0xFFFFFFFFFFFFFFFE

❺ Set the FP to the hash value of the key

Lock-Free and Log-Free Insertion

Warp



❶ Check if the key exists

Buckets

❷ Find the empty slot belonging to the less-loaded bucket

In-place writing

Target Slot

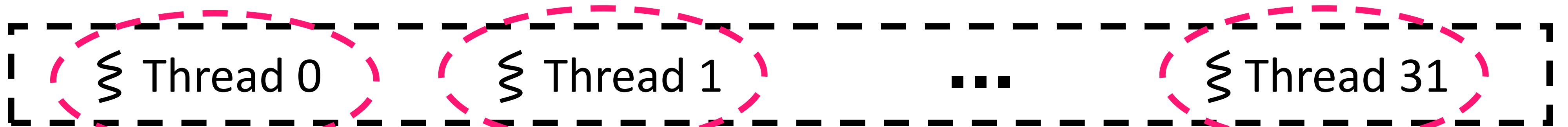
INSERT: 0xFFFFFFFFFFFFFFFE

❺ Set the FP to the hash value of the key

Target Slot

Lock-Free and Log-Free Insertion

Warp



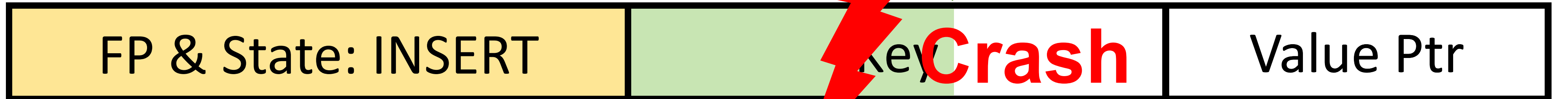
❶ Check if the key exists

Buckets

❷ Find the empty slot belonging to the less-loaded bucket

In-place writing

Target Slot

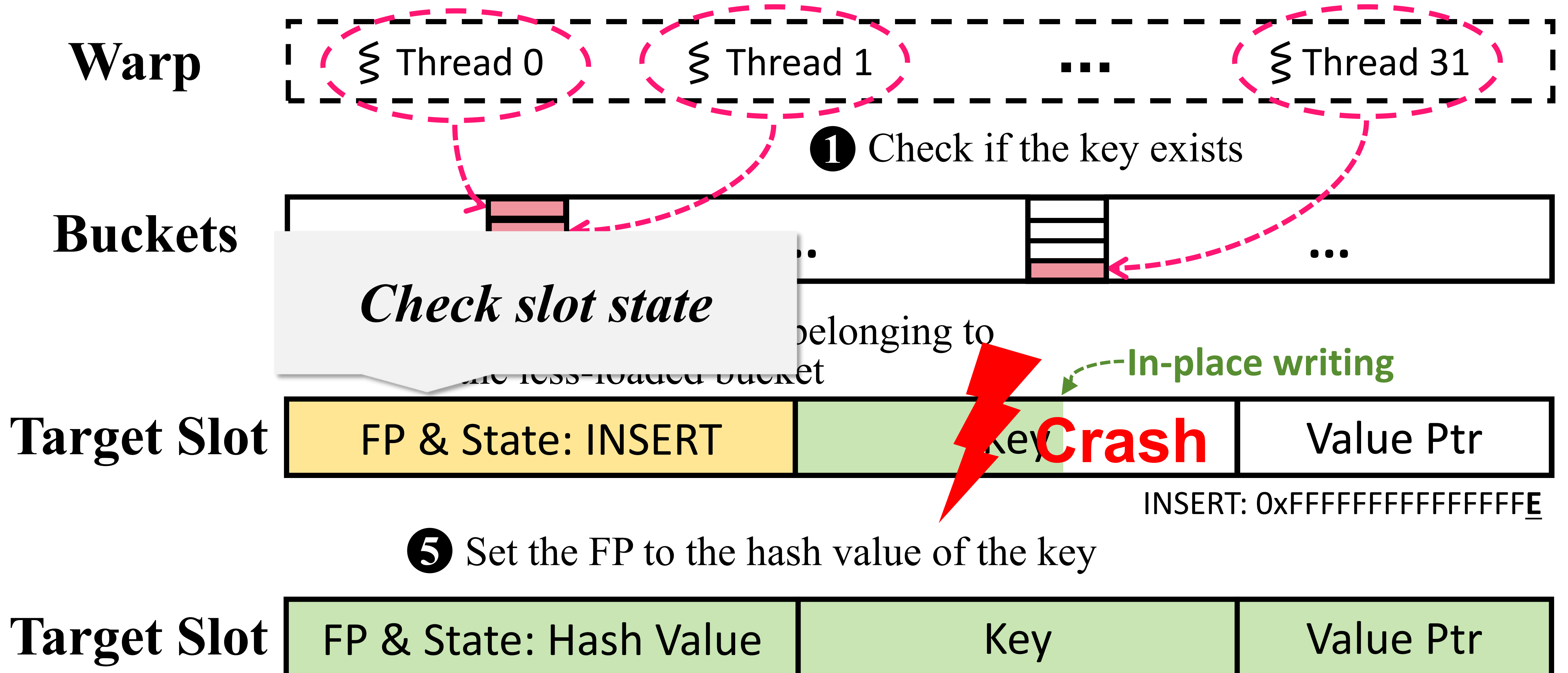


INSERT: 0xFFFFFFFFFFFFFFFE

❺ Set the FP to the hash value of the key

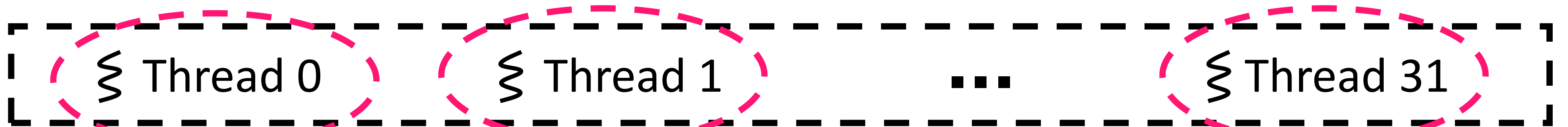
Target Slot

Lock-Free and Log-Free Insertion



Lock-Free and Log-Free Insertion

Warp



❶ Check if the key exists

Buckets

❷ Find the empty slot belonging to the less-loaded bucket

Recover

Target Slot

EMPTY: 0xFFFFFFFFFFFFFFFF

❺ Set the FP to the hash value of the key

Target Slot

Frozen-Based Bucket Cache

Frozen-Based Bucket Cache

GPU

PM

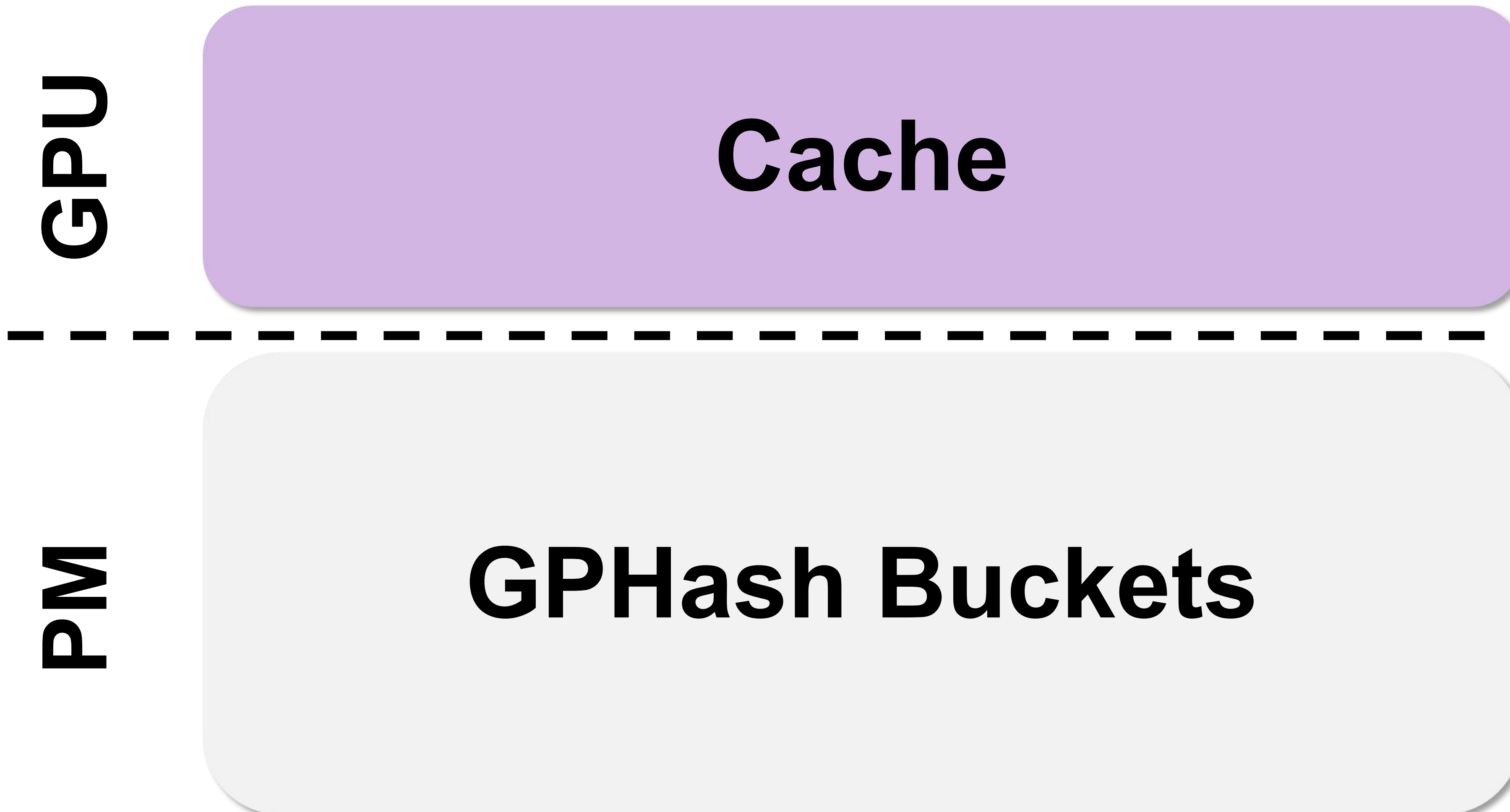
Frozen-Based Bucket Cache

GPU

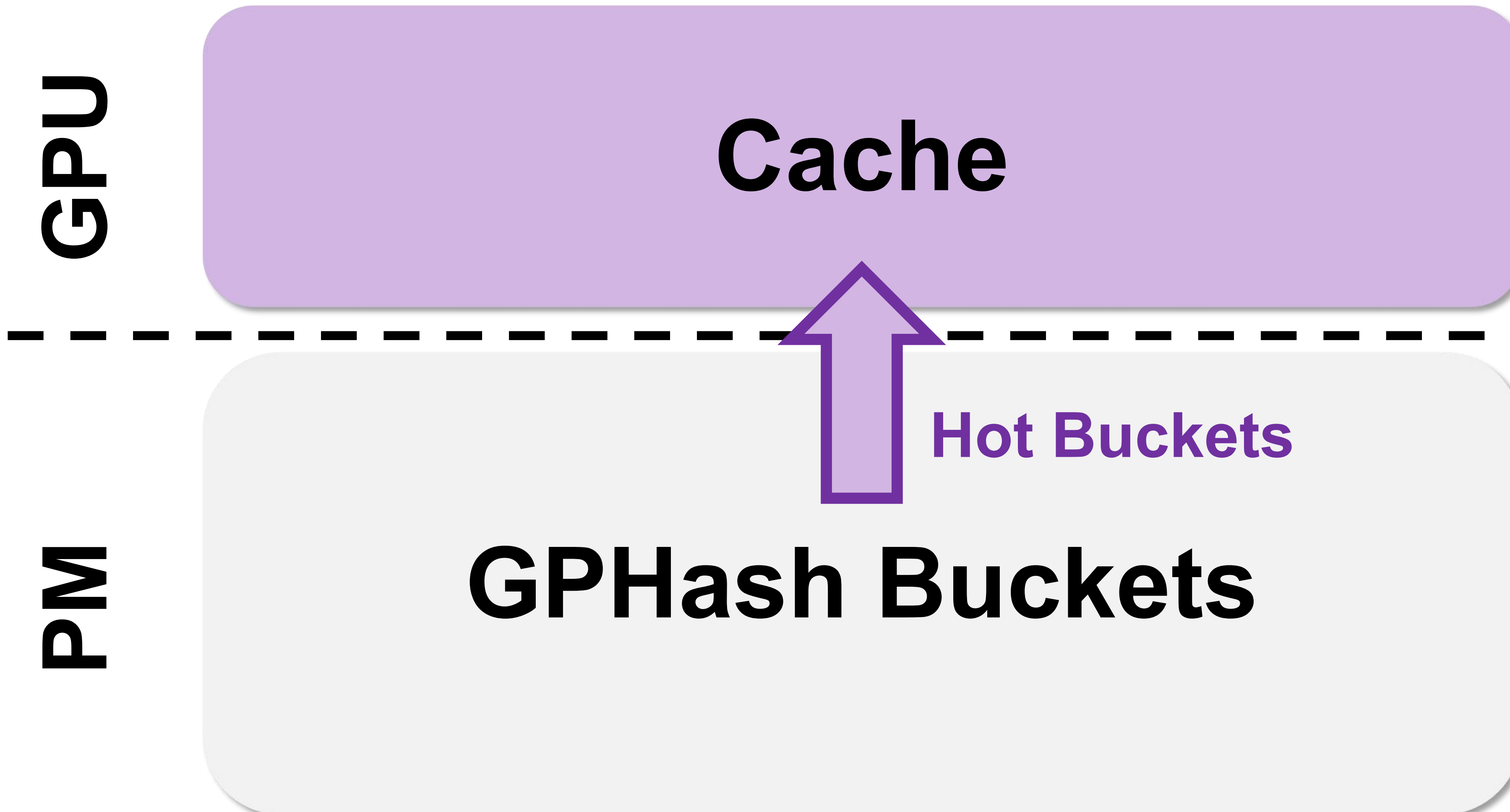
PM

GPHash Buckets

Frozen-Based Bucket Cache



Frozen-Based Bucket Cache



Frozen-Based Bucket Cache

GPU

PM

Frozen-Based Bucket Cache

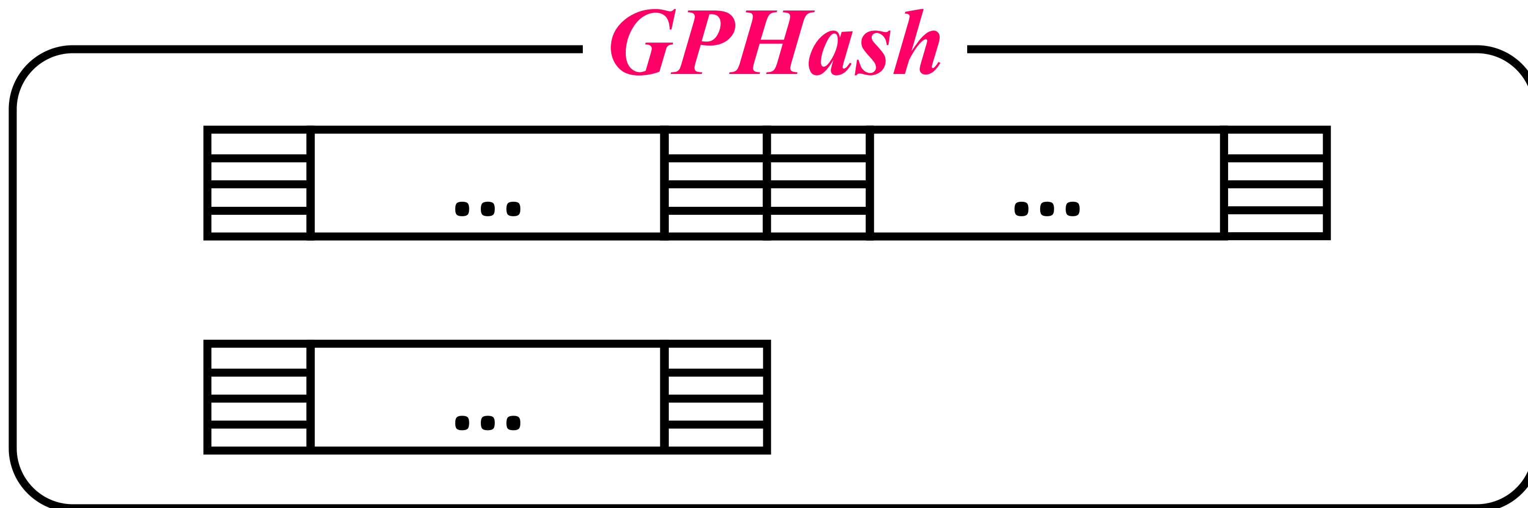
GPU

PM

Frozen-Based Bucket Cache

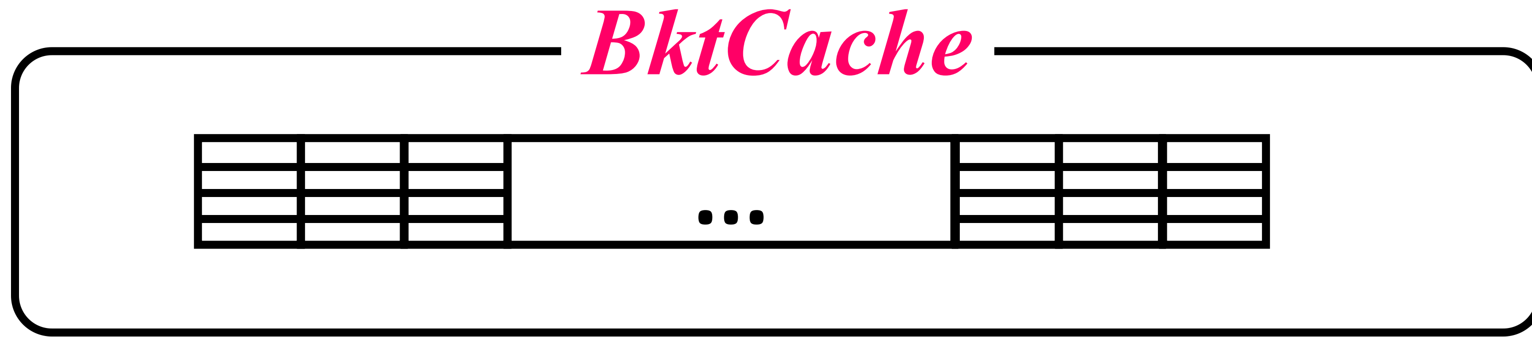
GPU

PM

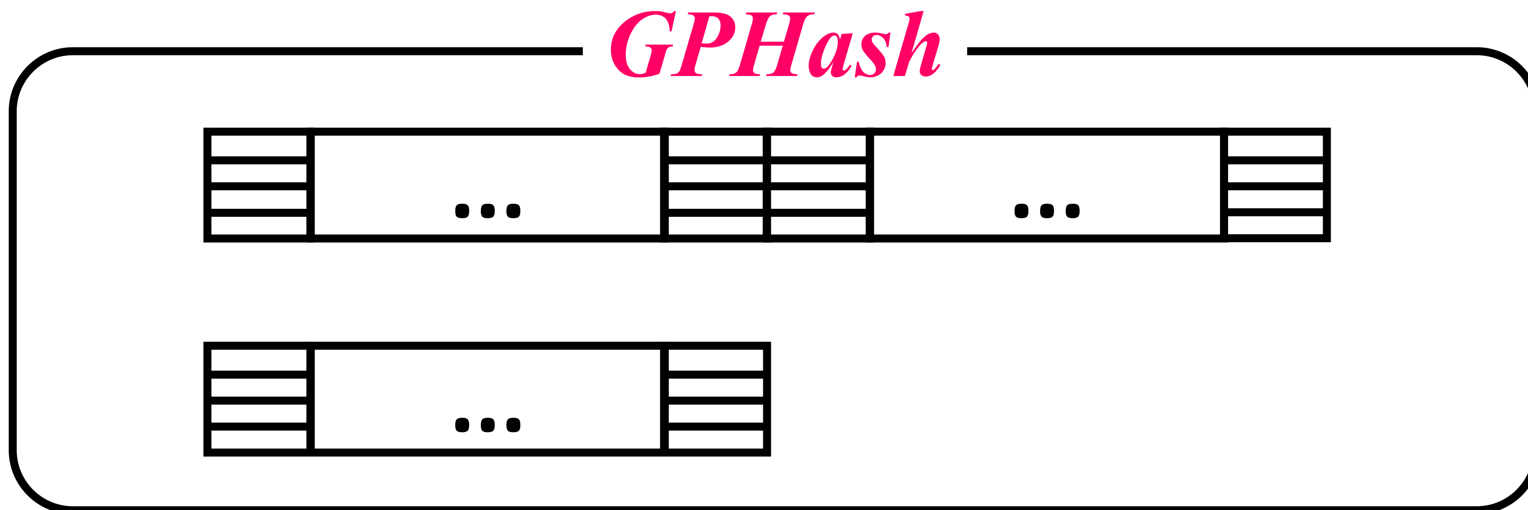


Frozen-Based Bucket Cache

GPU



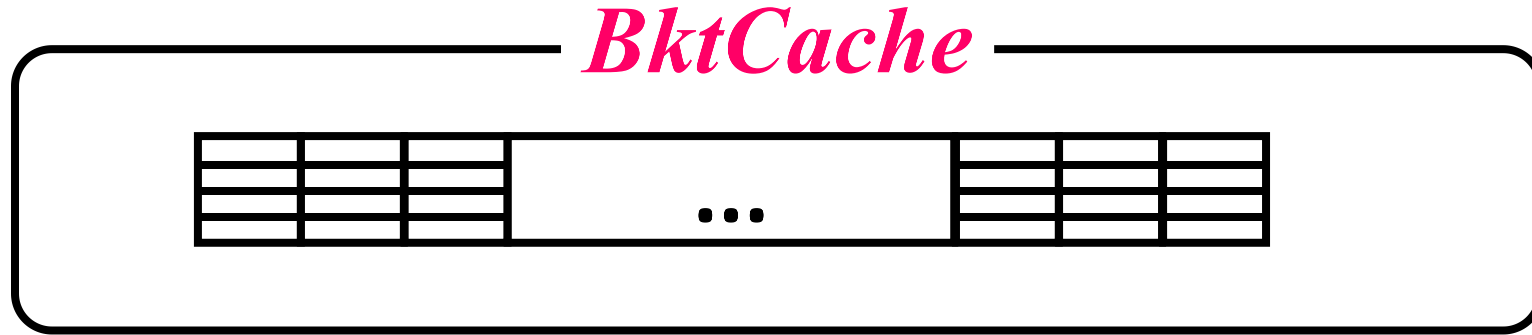
PM



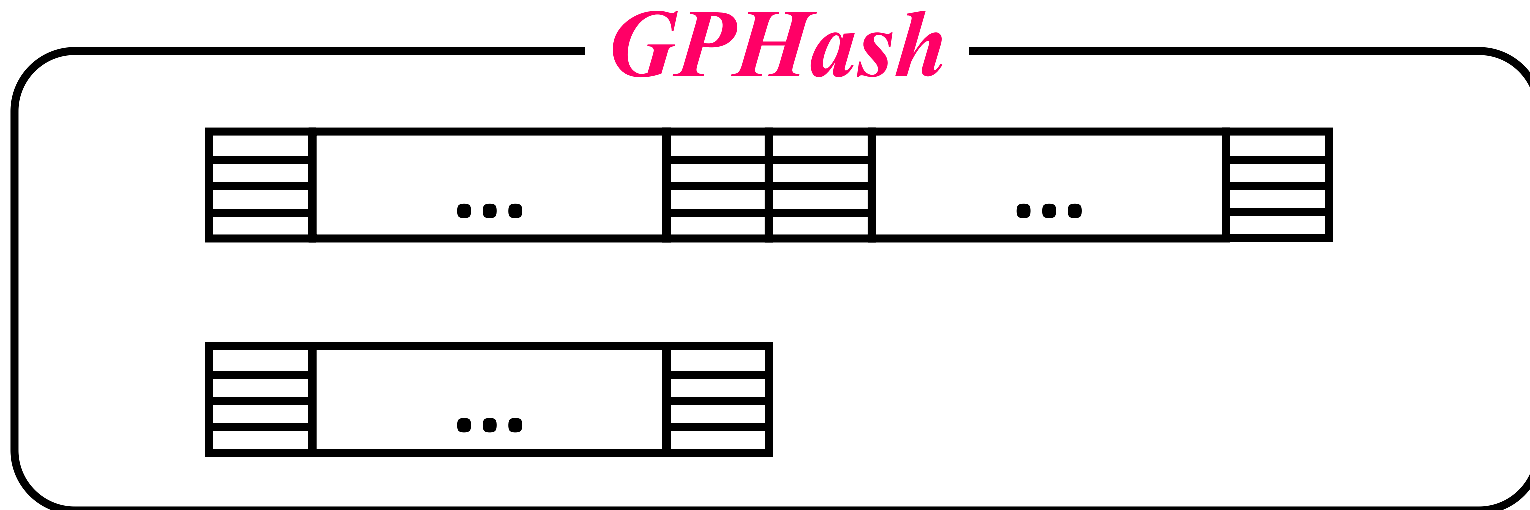
Frozen-Based Bucket Cache

Thread 0 Thread 1 ... Thread 31

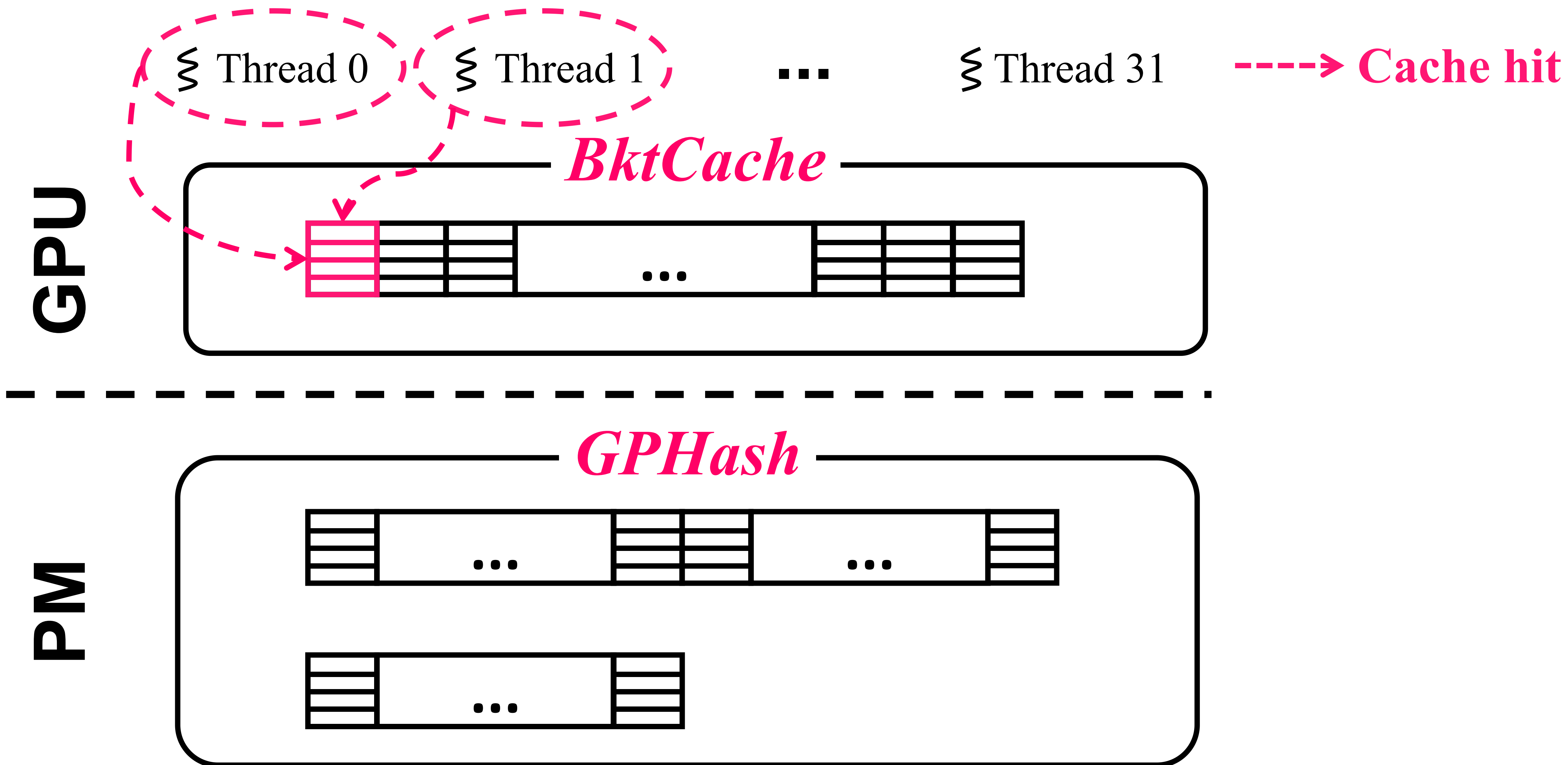
GPU



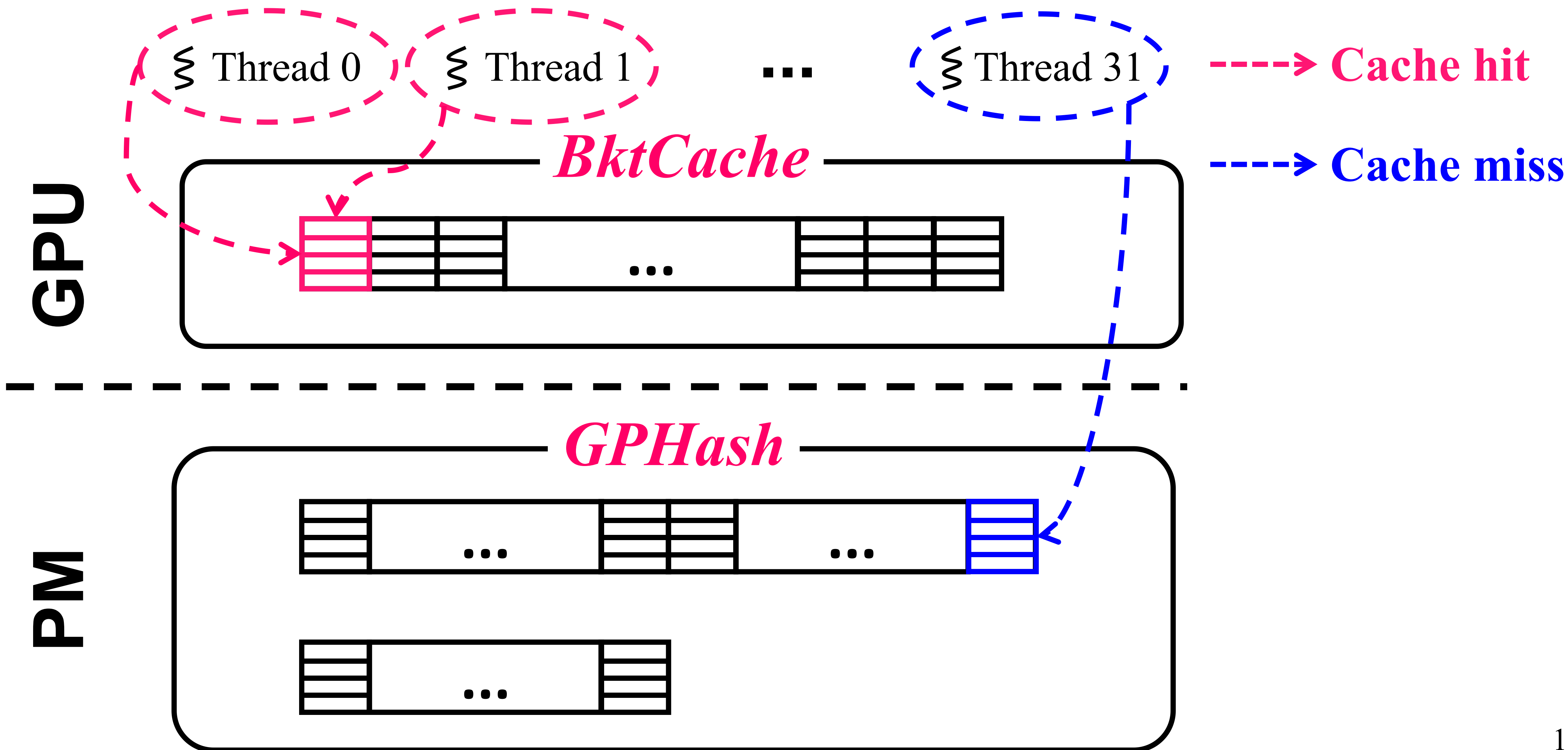
PM



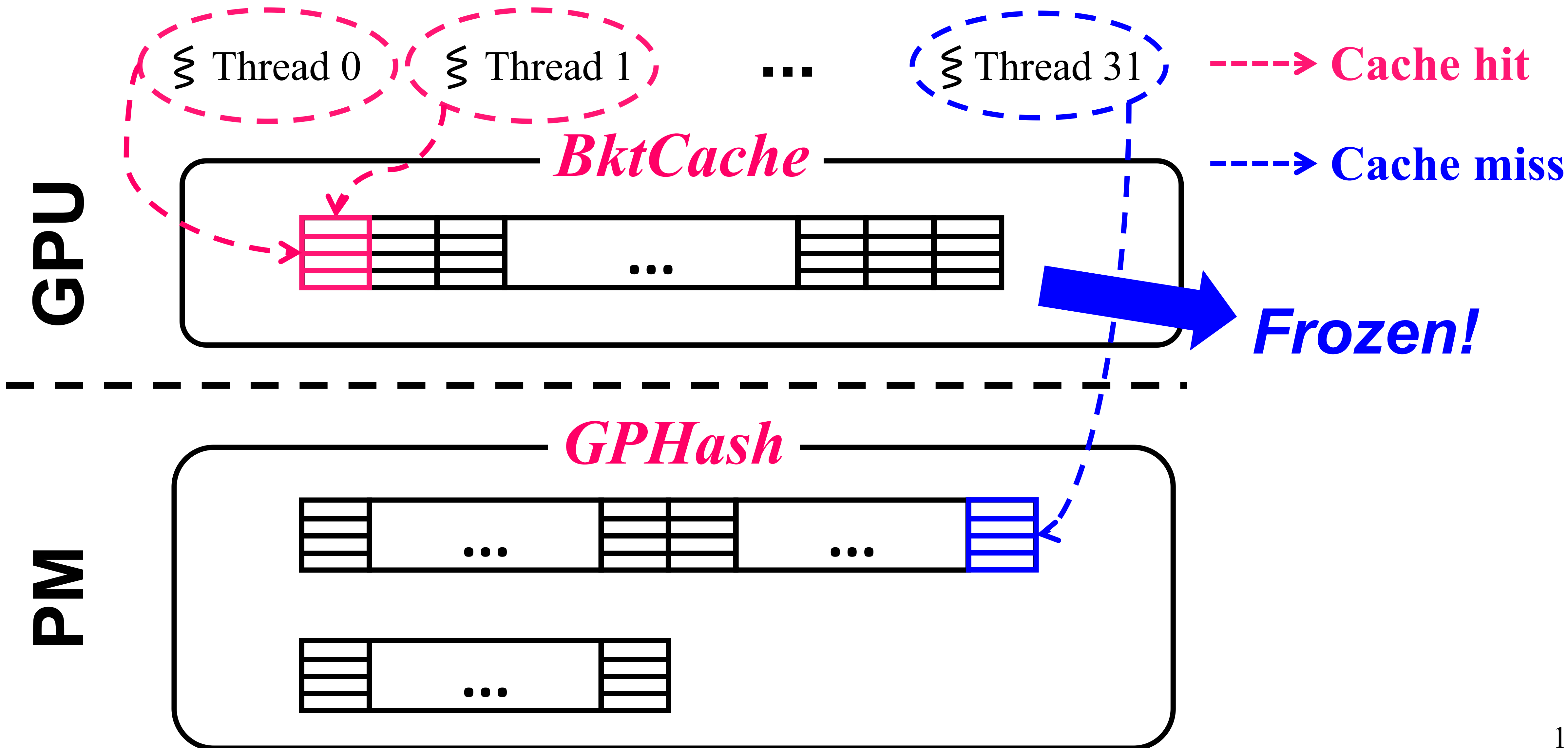
Frozen-Based Bucket Cache



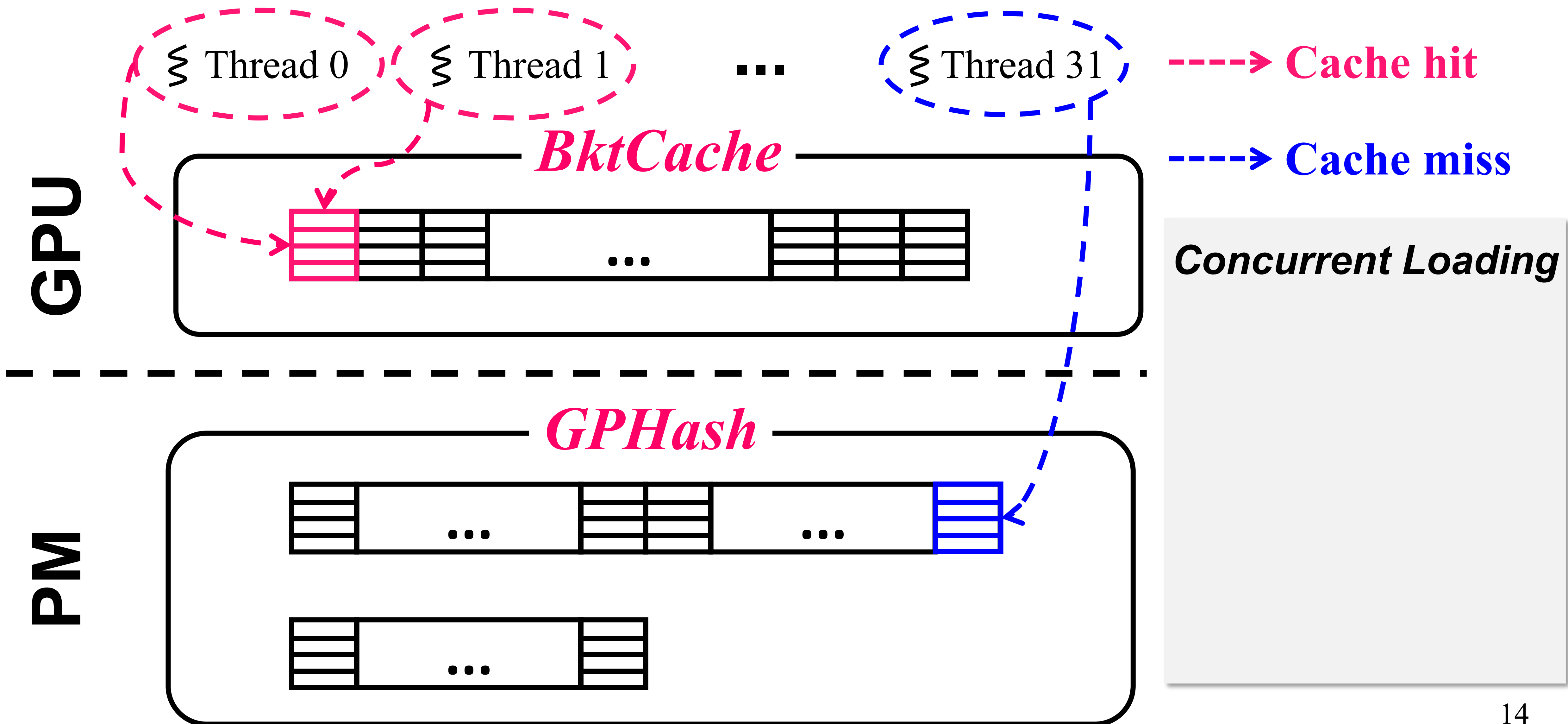
Frozen-Based Bucket Cache



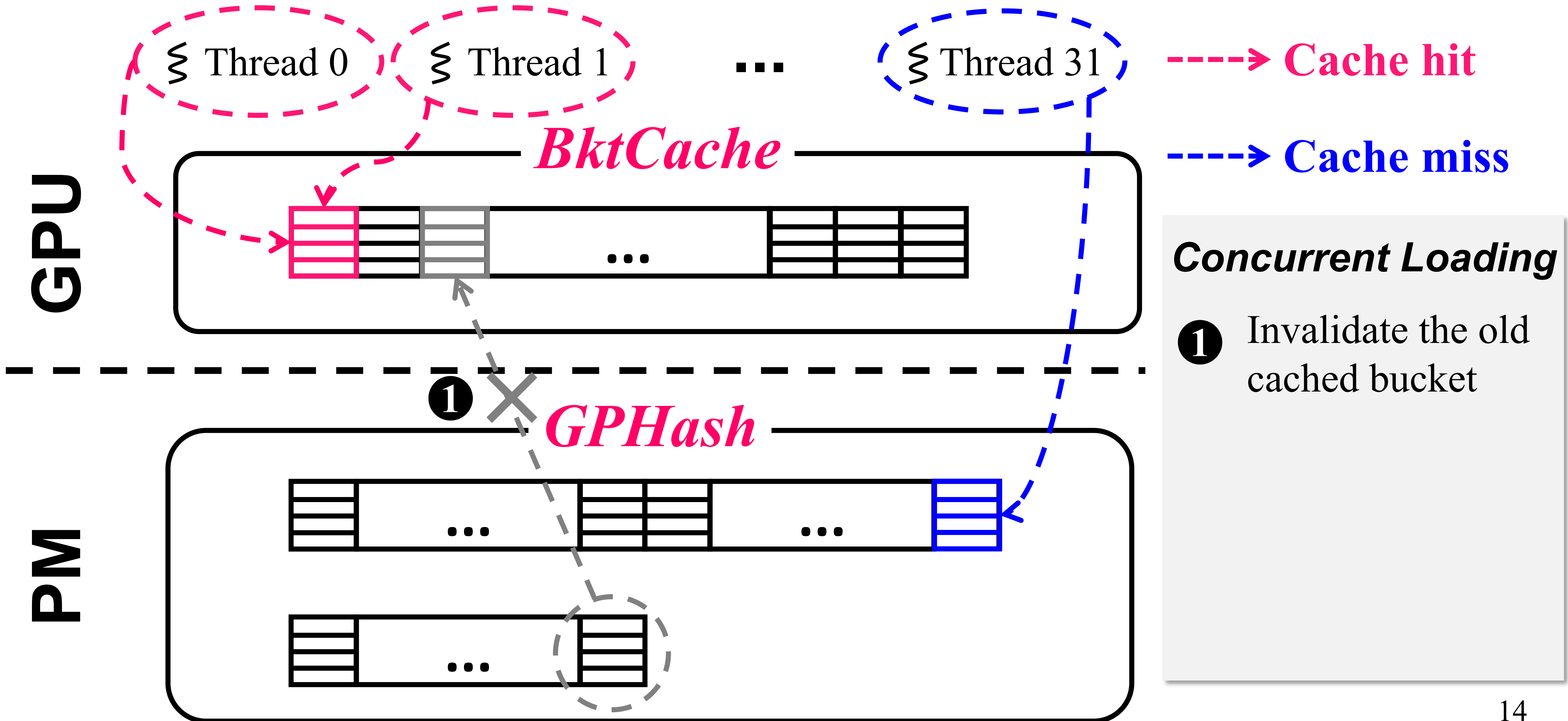
Frozen-Based Bucket Cache



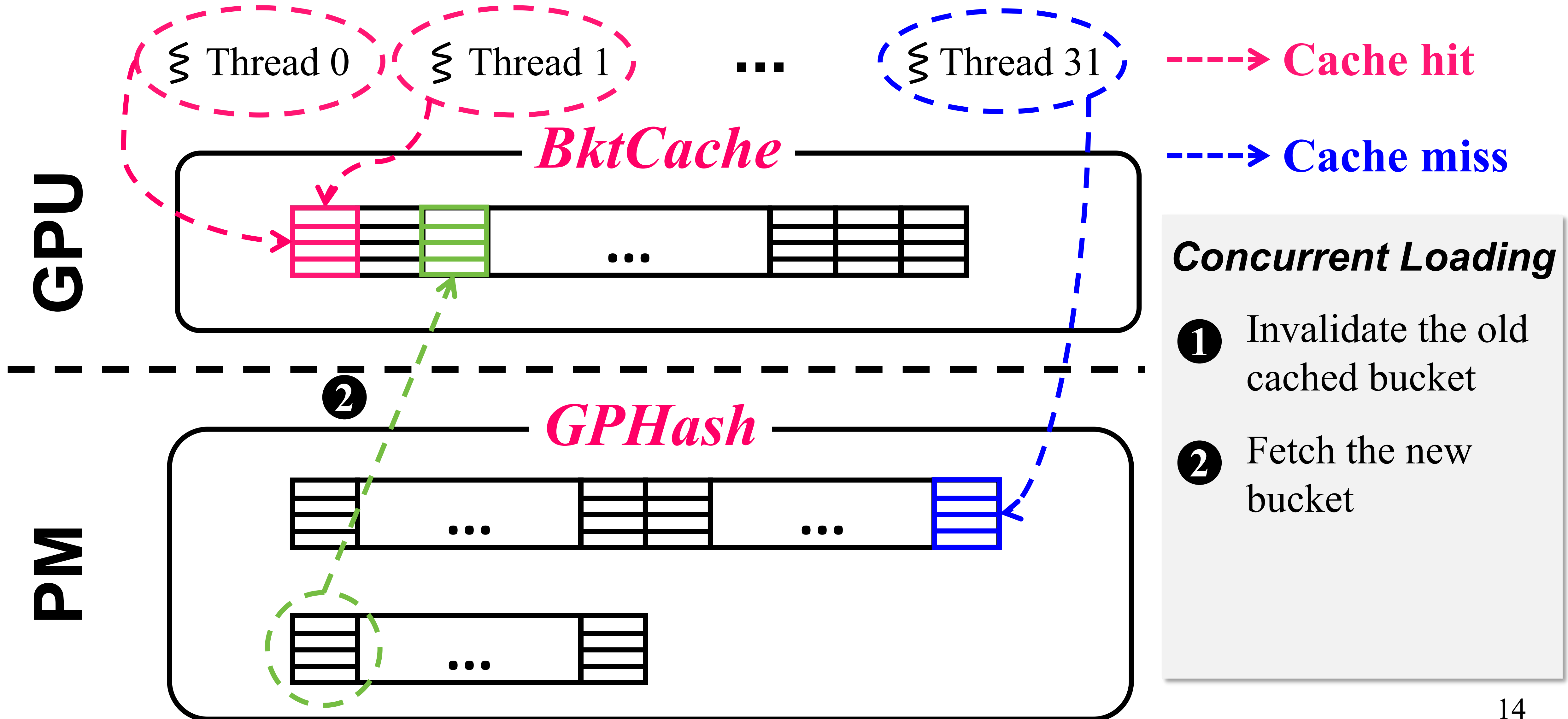
Frozen-Based Bucket Cache



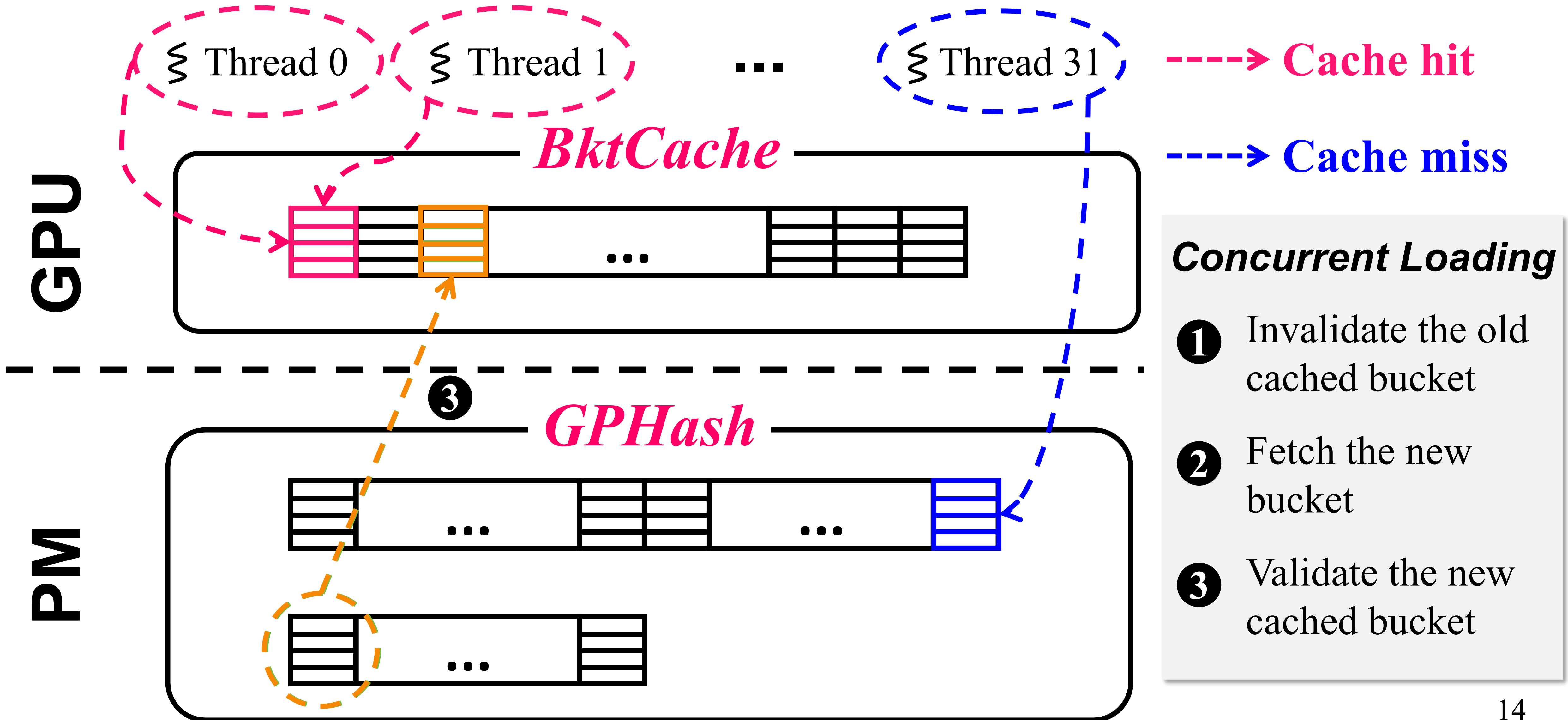
Frozen-Based Bucket Cache



Frozen-Based Bucket Cache



Frozen-Based Bucket Cache



More Details

More Details

- Warp-cooperative Execution Manner
- More Index Operations
- Bucket Caching Granularity
- …

More Details

- Warp-cooperative Exec-ution Manner
- More Index Operations
- Bucket Caching Granularity
- …

More Details

- Warp-cooperative Execution Manner
- More Index Operations
- Bucket Caching Granularity
- ...

Evaluation

Evaluation

➢ Platform

- 1 V100 GPU + 768 GB Intel Optane DC PM (6 × 128 GB)

Evaluation

- **Platform**
  - 1 V100 GPU + 768 GB Intel Optane DC PM (6 × 128 GB)
- **Comparisons**
  - CPU-assisted approaches[1]: *Clevel[ATC'20] , Dash[VLDB'20], and SEPH[OSDI'23]*
  - GPM-enabled approaches: *Clevel-GPM and SlabHash[IPDPS'18]-GPM*

[1] For fair evaluation, we use PM to store the data and leverage PM hash indexes to manage the data in PM

Evaluation

➢ Platform

- 1 V100 GPU + 768 GB Intel Optane DC PM (6 × 128 GB)

➢ Comparisons

- CPU-assisted approaches[1]: *Clevel[ATC'20] , Dash[VLDB'20], and SEPH[OSDI'23]*
- GPM-enabled approaches: *Clevel-GPM and SlabHash[IPDPS'18]-GPM*

➢ Workloads

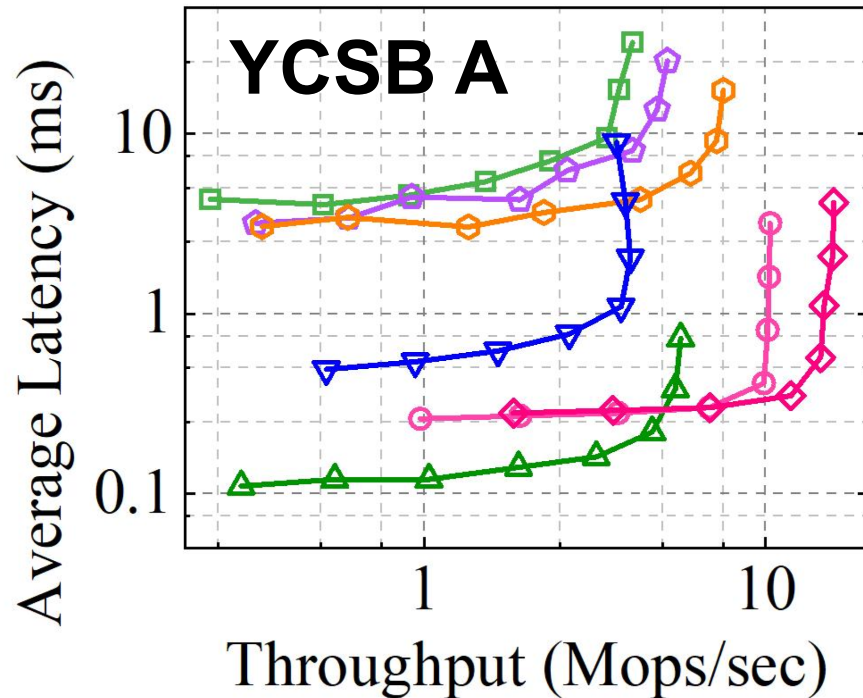
- YCSB workloads: *8-byte and 32-byte keys, 128-byte values*
- Real-world workloads: *DLRM and PageRank*

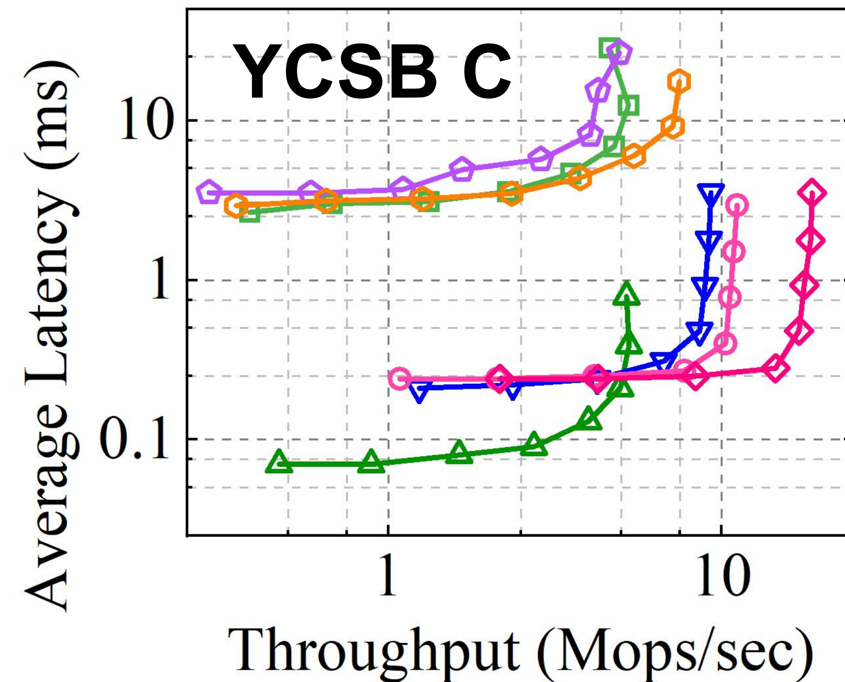
[1] For fair evaluation, we use PM to store the data and leverage PM hash indexes to manage the data in PM

End-to-End Performance

End-to-End Performance

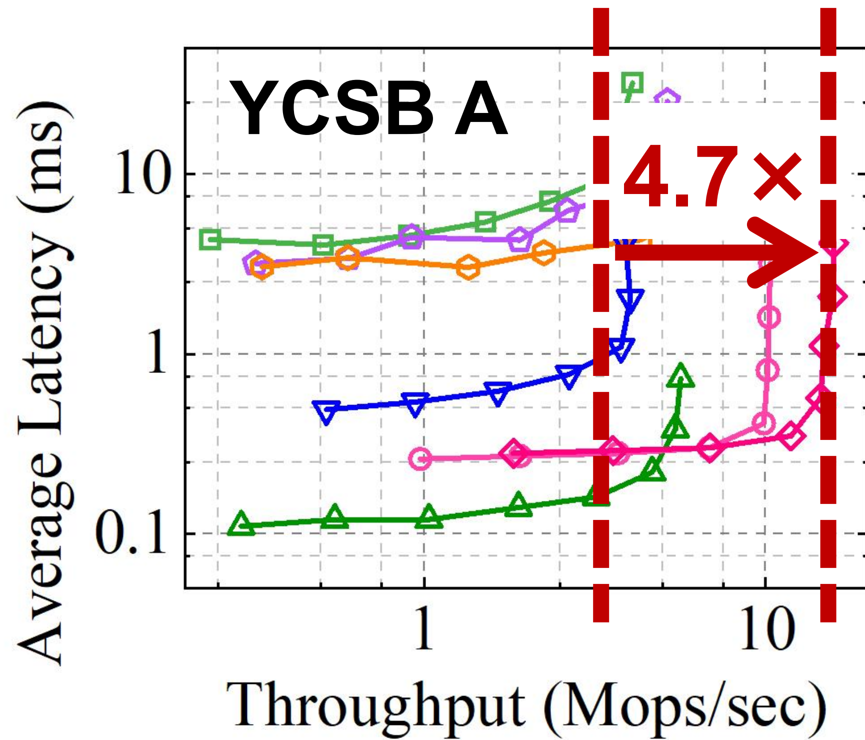
Clevel Dash SEPH Clevel-GPM SlabHash-GPM GPHash w/o cache GPHash

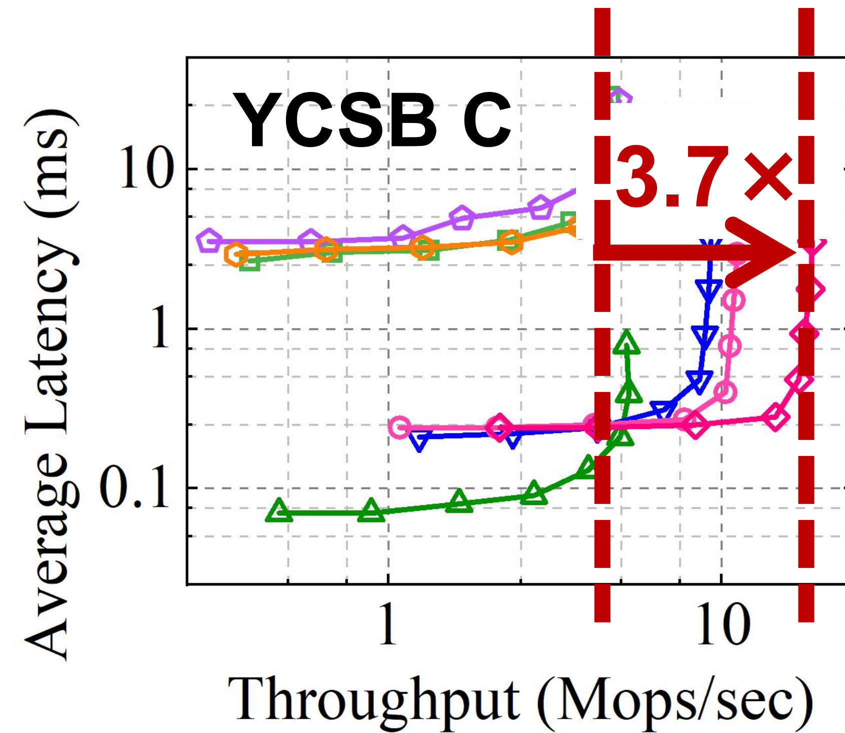




End-to-End Performance

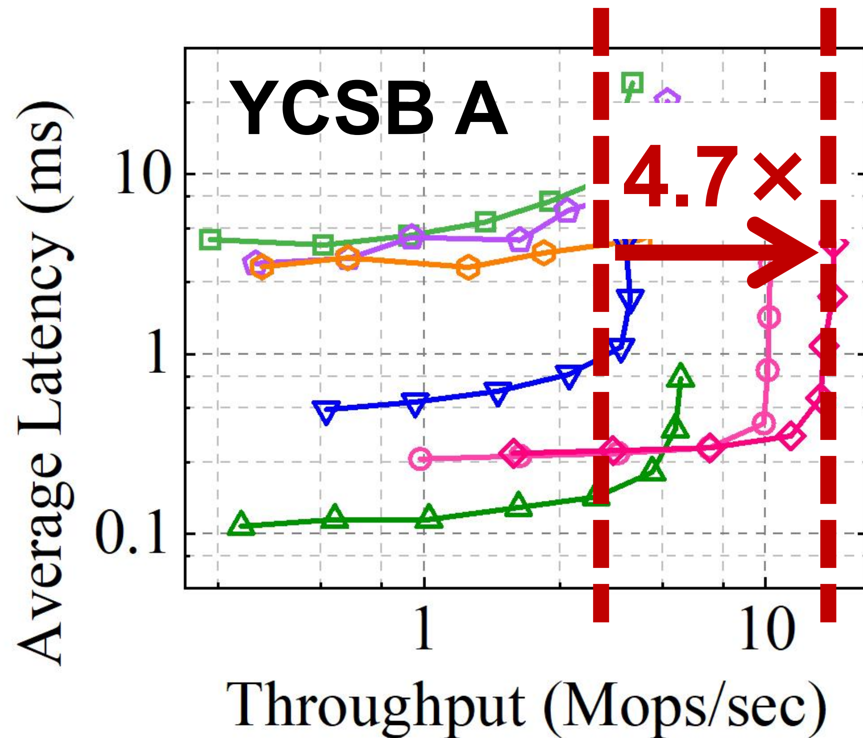
Clevel Dash SEPH Clevel-GPM SlabHash-GPM GPHash w/o cache GPHash

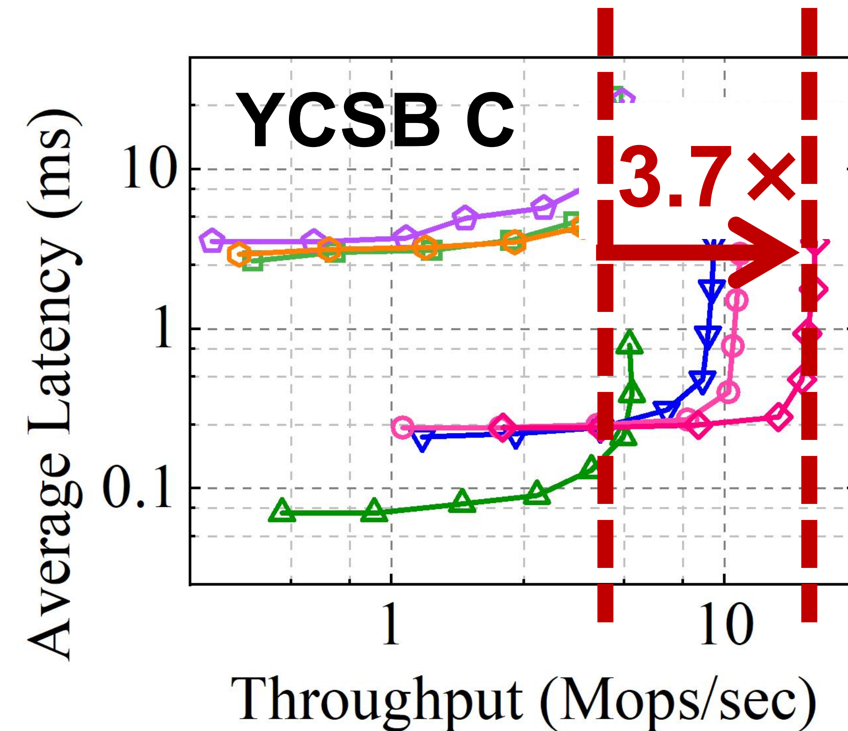




End-to-End Performance

Clevel Dash SEPH Clevel-GPM SlabHash-GPM GPHash w/o cache GPHash

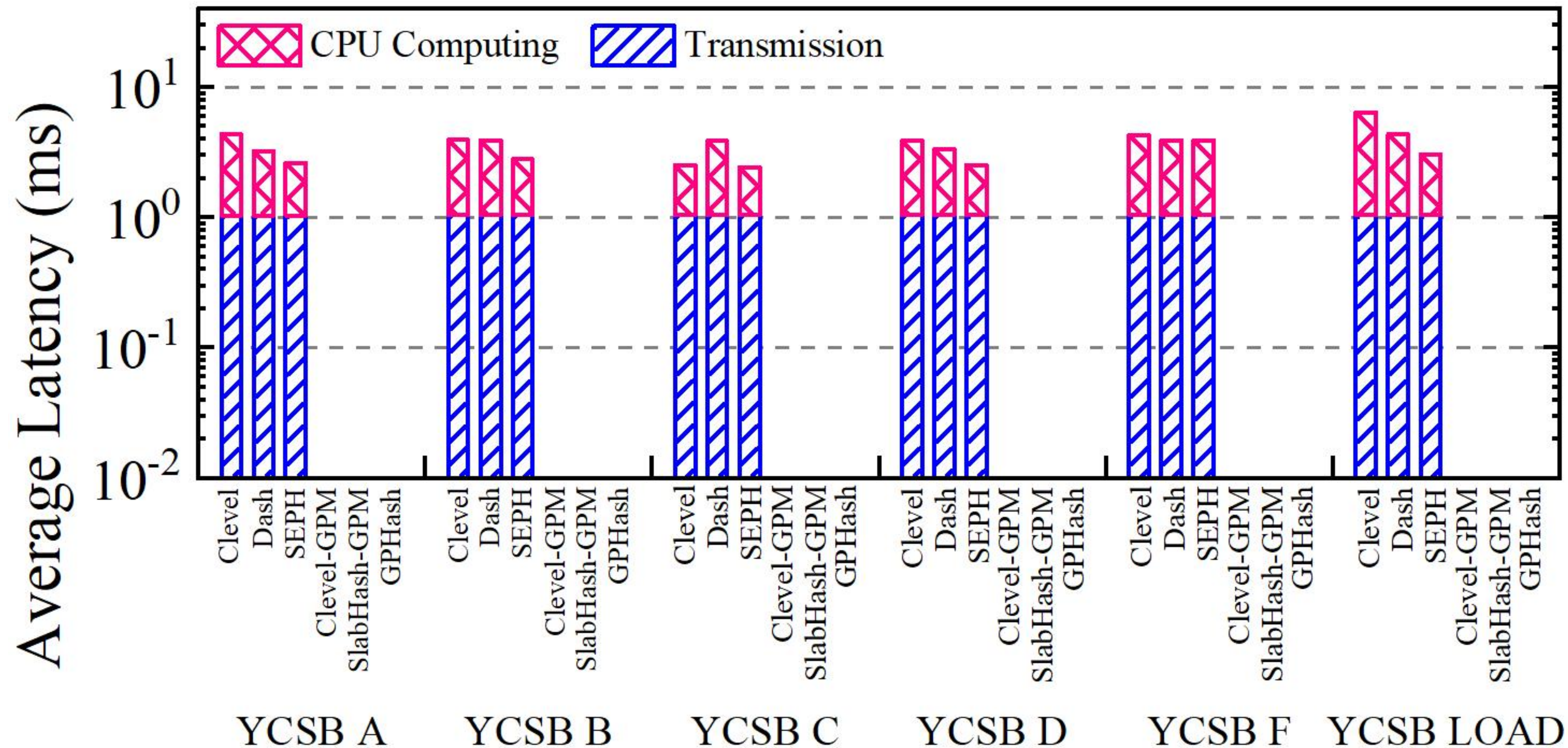




GPHash improves the throughput by **1.9~6.3×**

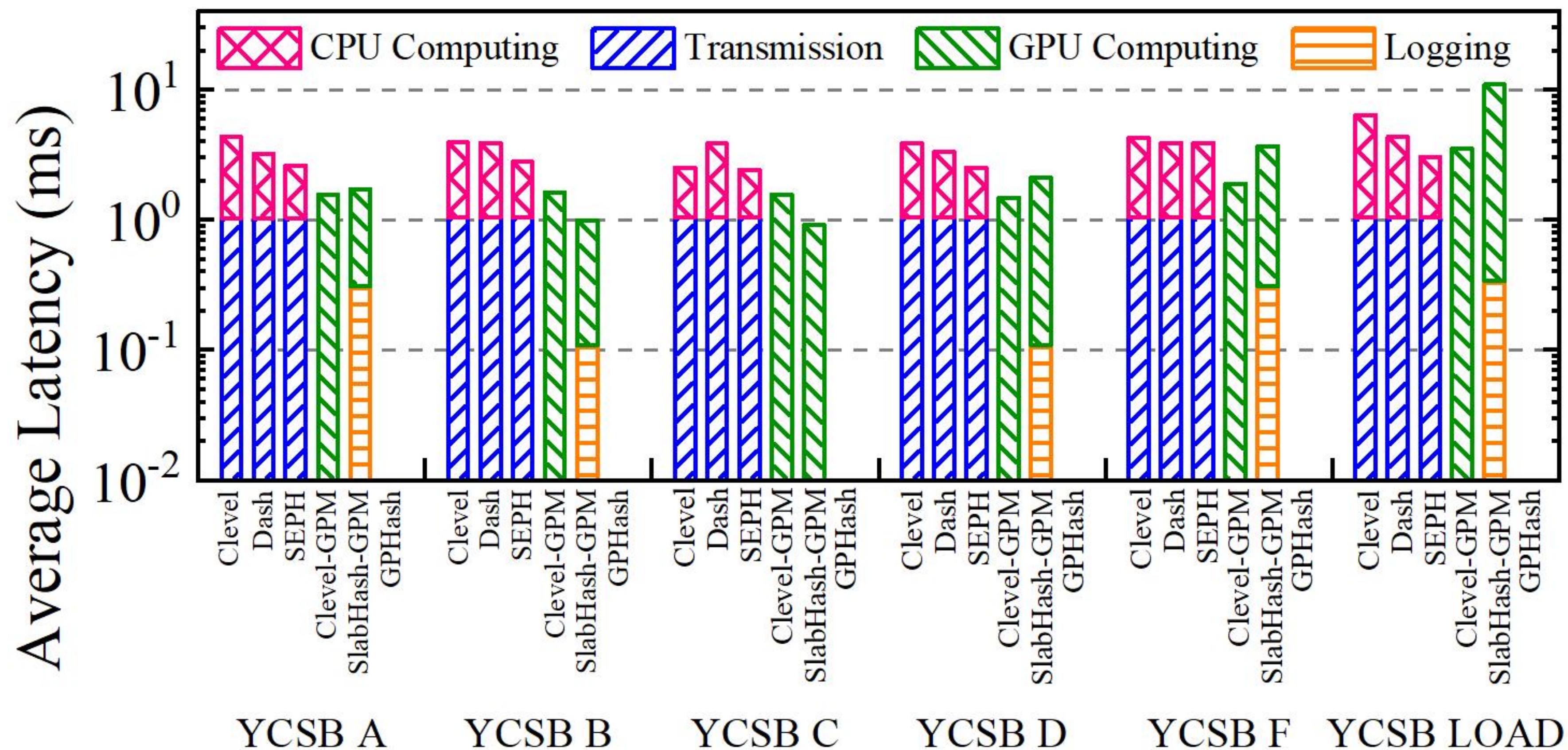
Latency Breakdown

Latency Breakdown



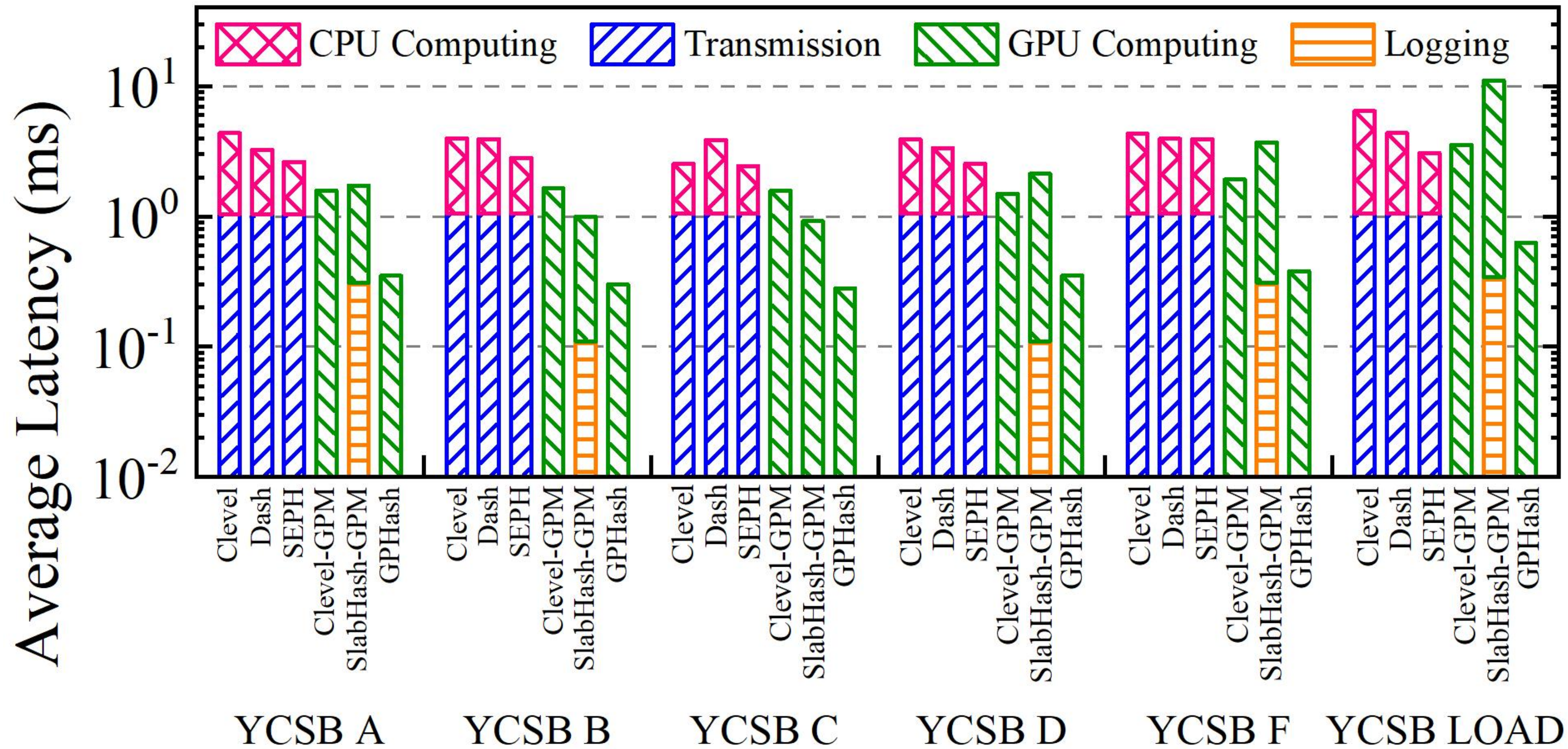
CPU-assisted approaches suffer from ***high transmission cost***

Latency Breakdown



Naive GPM-enabled approaches suffer from ***severe warp divergence*** and ***high-overhead consistency guarantee***

Latency Breakdown



GPHash fully leverages the ***high parallelism of GPU*** and provides a ***low-overhead consistency guarantee***

Conclusion

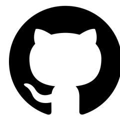
Conclusion

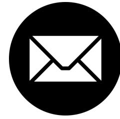
- Inefficient GPU data management on large-scale data
  - ***High overhead for data transfer***
  - ***Extra CPU consumption***

Conclusion

- Inefficient GPU data management on large-scale data
  - ***High overhead for data transfer***
  - ***Extra CPU consumption***
- **GPHash:** an efficient GPM-enabled hash index
  - ***GPU-conscious and PM-friendly hash table***
  - ***Lock-free and log-free operations***
  - ***Frozen-based bucket cache***

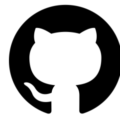
Conclusion

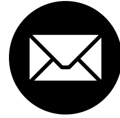

https://github.com/LighT-chenml/GPHash


chenml@hust.edu.cn

- Inefficient GPU data management on large-scale data
  - ***High overhead for data transfer***
  - ***Extra CPU consumption***
- **GPHash:** an efficient GPM-enabled hash index
  - ***GPU-conscious and PM-friendly hash table***
  - ***Lock-free and log-free operations***
  - ***Frozen-based bucket cache***

Conclusion

 https://github.com/LighT-chenml/GPHash

 chenml@hust.edu.cn

- Inefficient GPU data management on large-scale data
  - ***High overhead for data transfer***
  - ***Extra CPU consumption***
- **GPHash:** an efficient GPM-enabled hash index
  - ***GPU-conscious and PM-friendly hash table***
  - ***Lock-free and log-free operations***
  - ***Frozen-based bucket cache***

Thank you! Q&A