Morpheus

Extending the Last Level Cache Capacity in GPU Systems with Idle GPU Core Resources

Sina Darabi, Mohammad Sadrosadati, Negar Akbarzadeh,
Joël Lindegger, Mohammad Hosseini, Jisung Park, Juan Gómez-Luna,
Onur Mutlu, Hamid Sarbazi-Azad

October 3rd 2022
MICRO-55

Motivation & Goal

Memory-bound GPU applications
leave **some cores under-utilized**

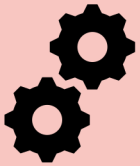
Our Goal:

Leverage the **under-utilized cores**
to **boost** the **performance** and **energy efficiency**
of **memory-bound applications**

Our Work

Morpheus

First technique that leverages some **GPU cores' private memories** to **extend** the total GPU **last-level cache (LLC) capacity**



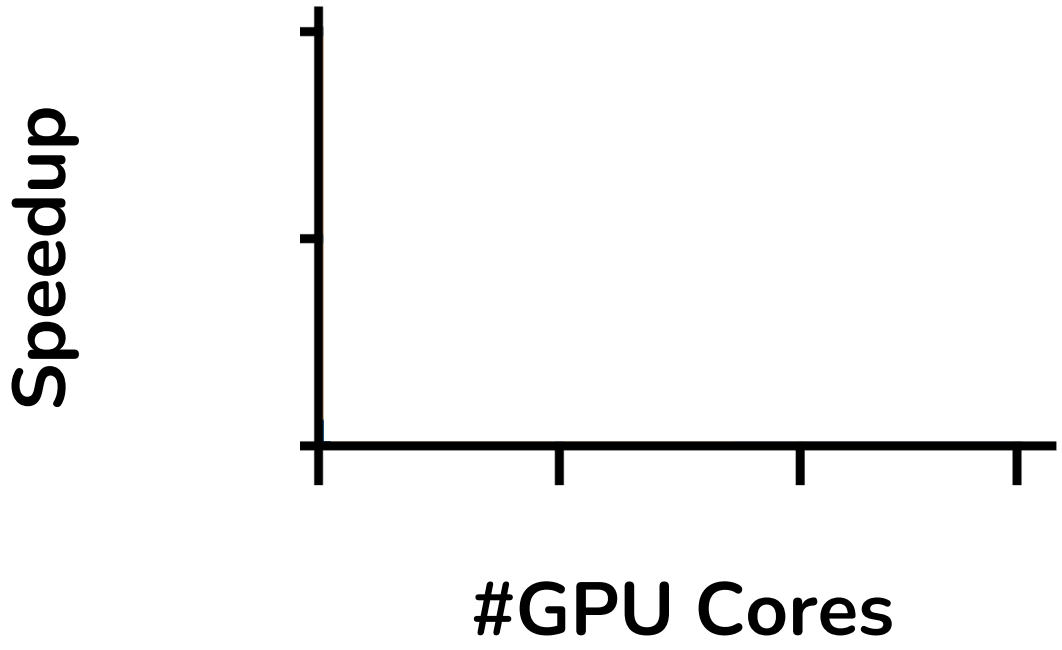
Morpheus employs **(1)** a **software helper kernel** in cores and **(2)** a **new hardware unit** in the LLC partitions

Morpheus significantly **improves performance** and **energy efficiency** of **memory-bound GPU applications**

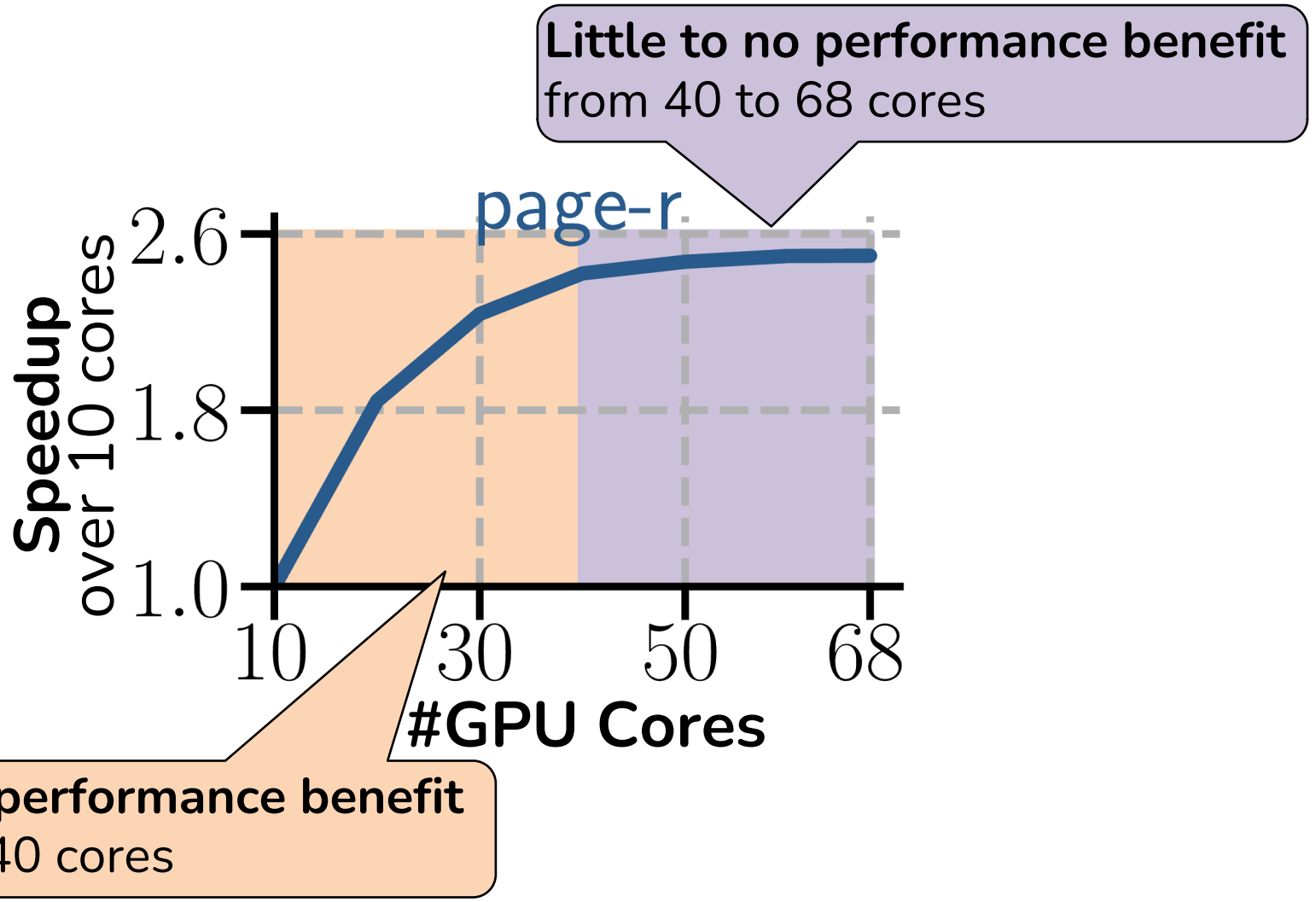
and **enables a 4x larger LLC** with the same LLC hardware

Outline

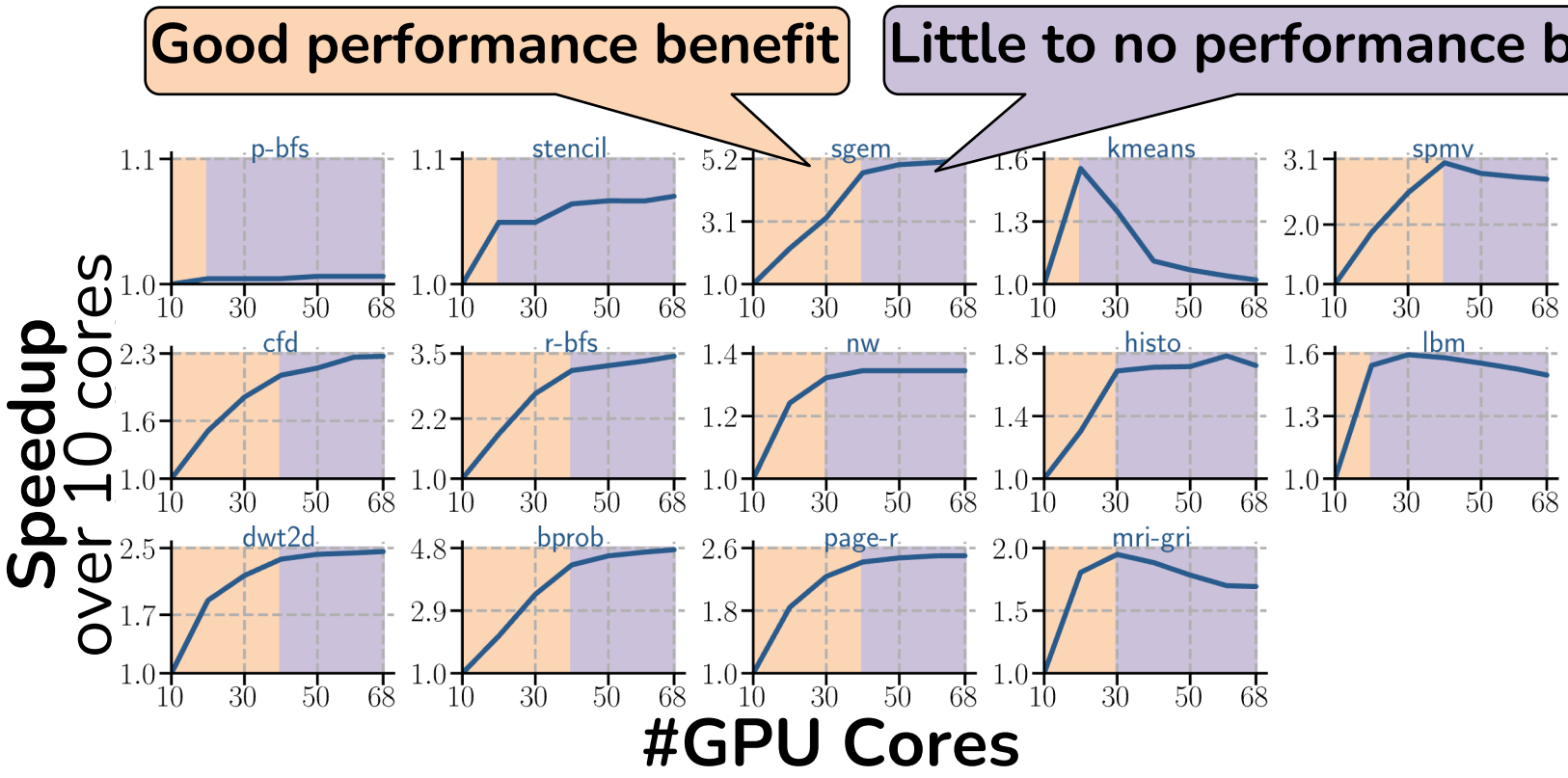
Memory-Bound GPU Applications



Memory-Bound GPU Applications



Memory-Bound GPU Applications

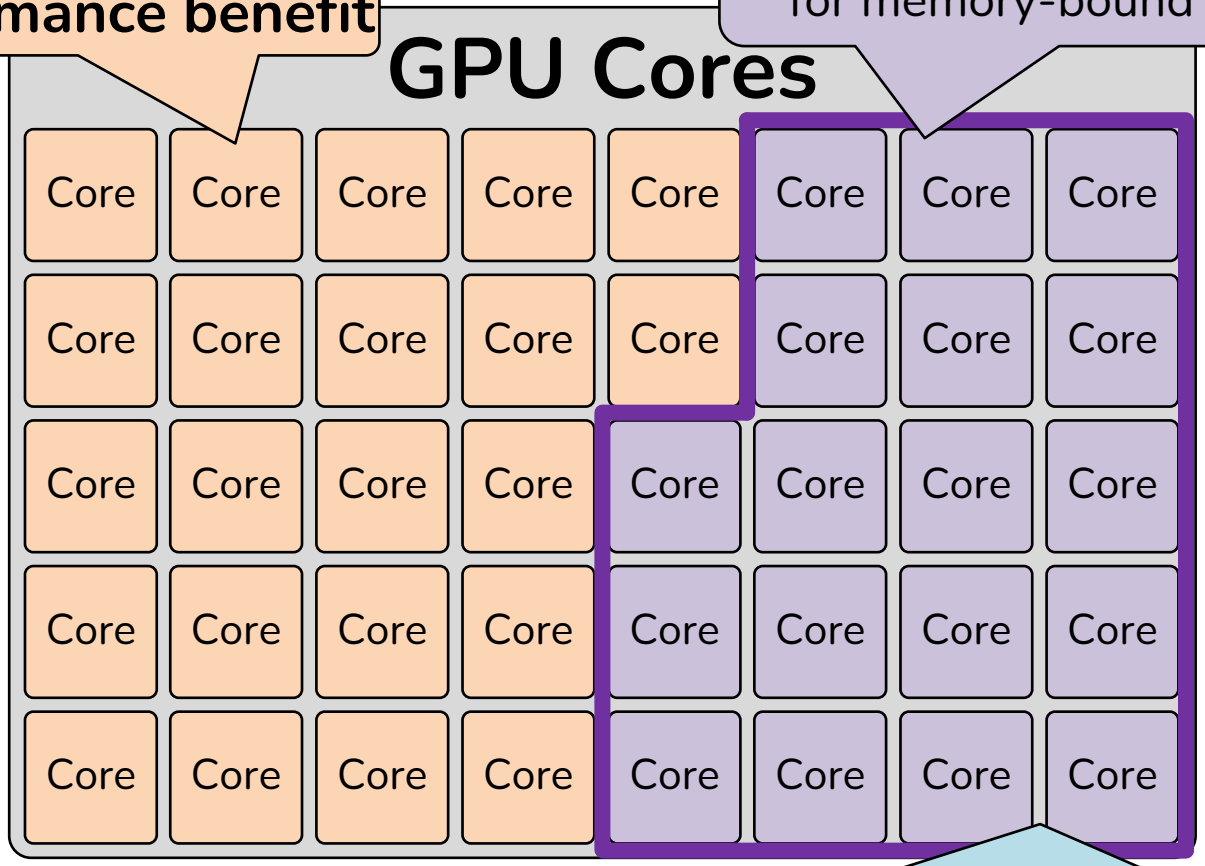


Conventional GPUs do **not benefit (significantly) from all cores** for **memory-bound** applications

Our Goal

Good performance benefit

Little to no performance benefit
for memory-bound applications



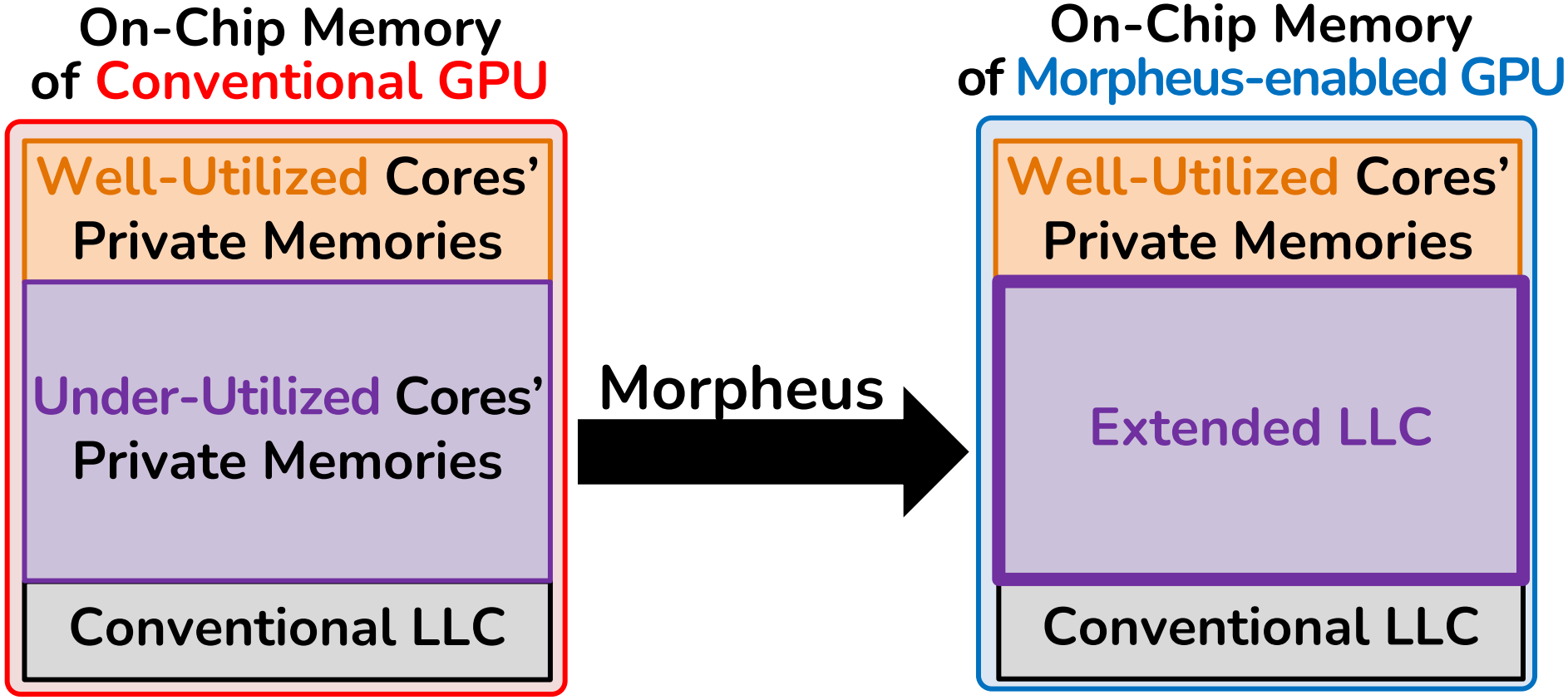
Our Goal

Leverage the **under-utilized cores**
to **boost** the **performance** and **energy efficiency**
of **memory-bound applications**

SAFARI

Outline

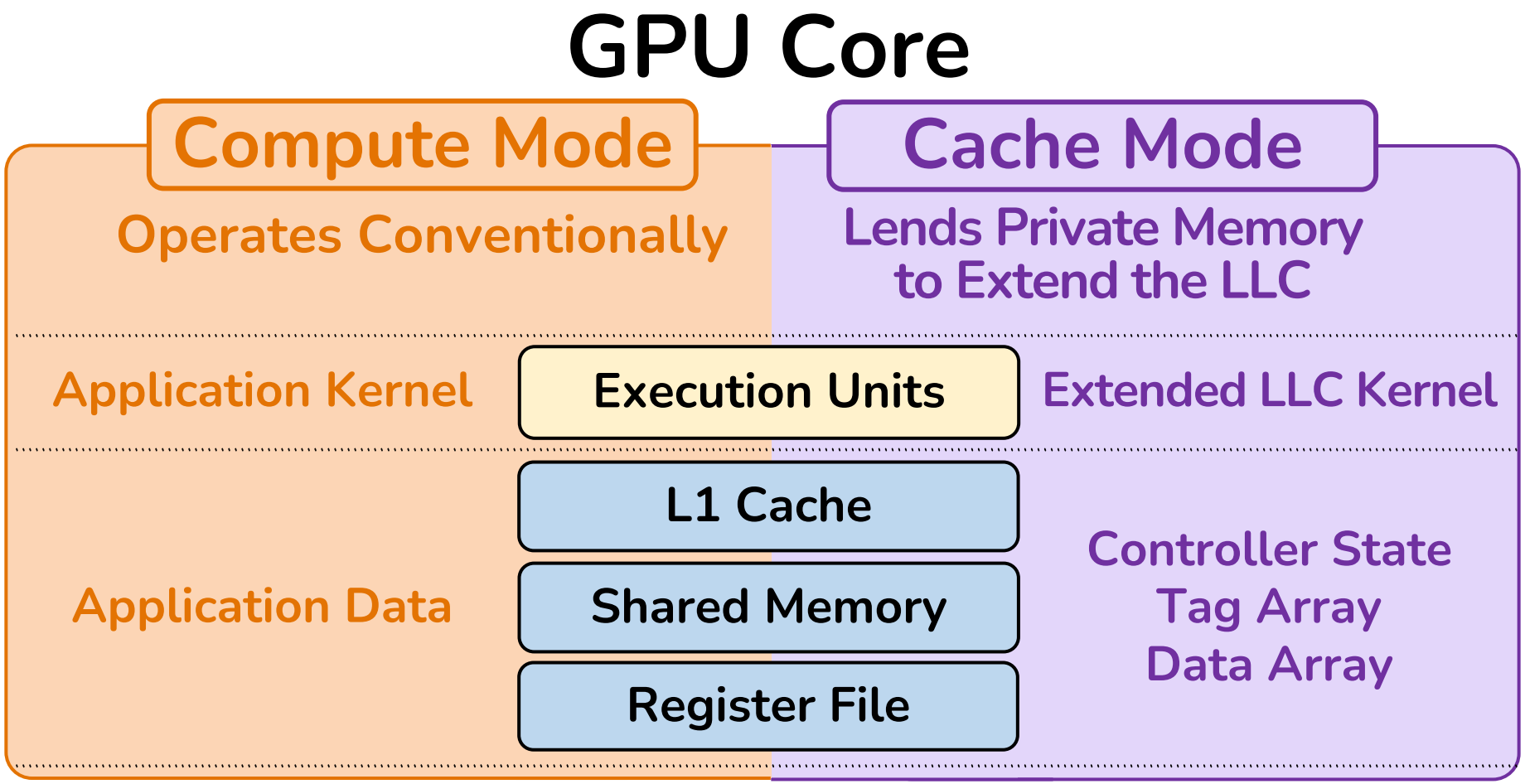
Key Idea



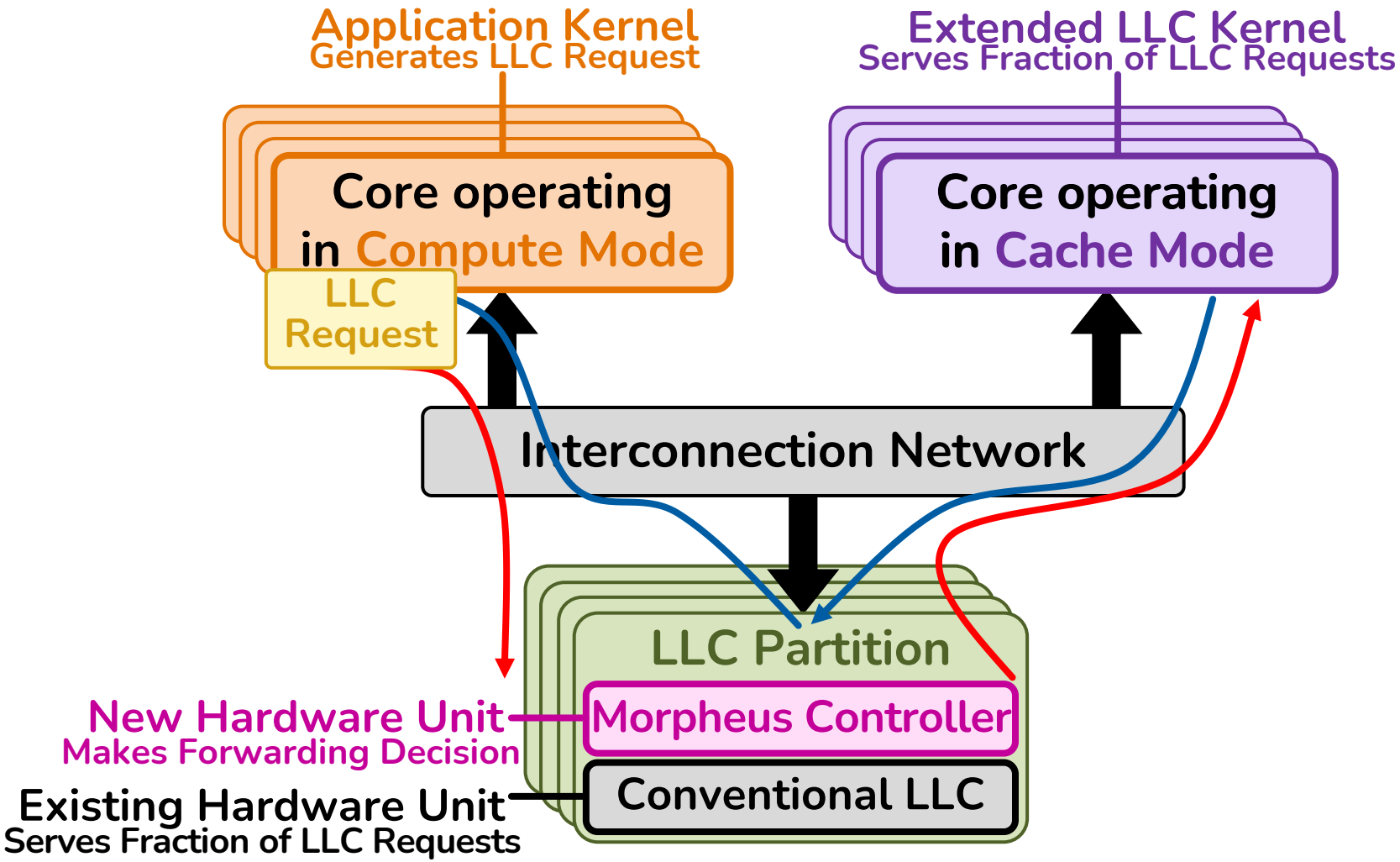
Morpheus' Key Idea:

Repurpose under-utilized GPU cores' private memory to extend the GPU's LLC capacity

GPU Core Execution Modes in Morpheus

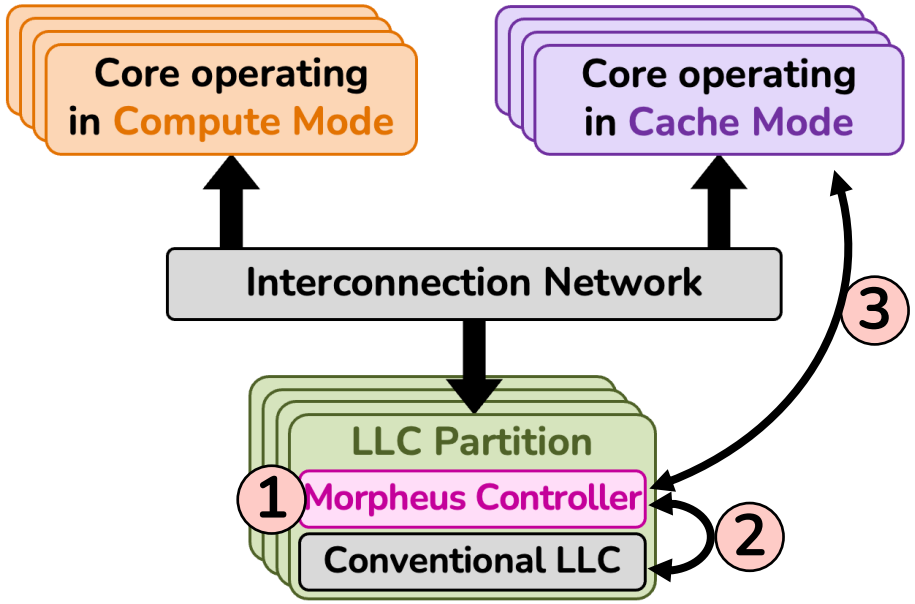


Serving an LLC Request in Morpheus



Outline

Morpheus Controller

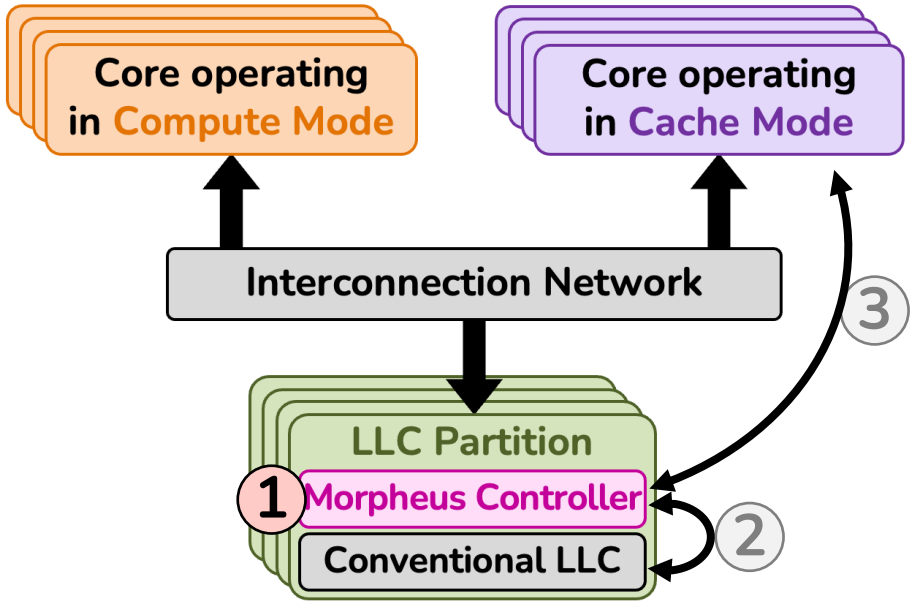


1 Forwarding Decision

2 Query Conventional LLC

3 Query Extended LLC

Morpheus Controller



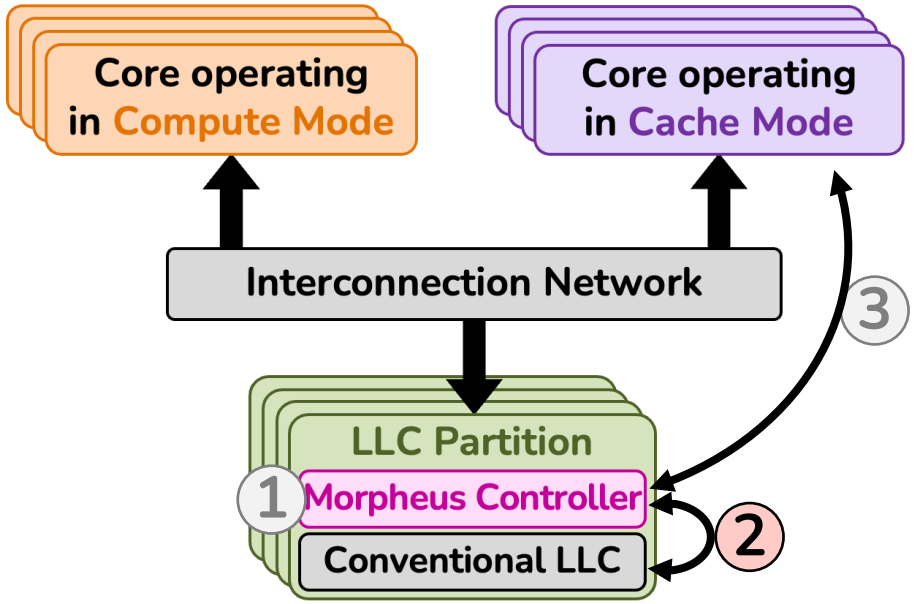
1 **Forwarding Decision**

***Address Separation*, Combinational Logic**

2 Query Conventional LLC

3 Query Extended LLC

Morpheus Controller



1 **Forwarding Decision**

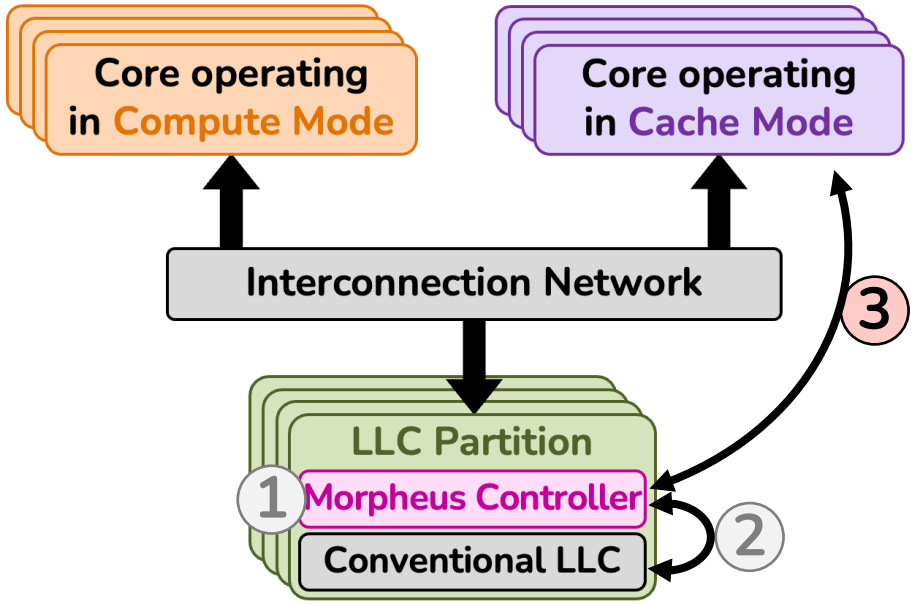
Address Separation, Combinational Logic

2 **Query Conventional LLC**

Physically Co-Located, Simple Forwarding

3 **Query Extended LLC**

Morpheus Controller



1 **Forwarding Decision**

Address Separation, Combinational Logic

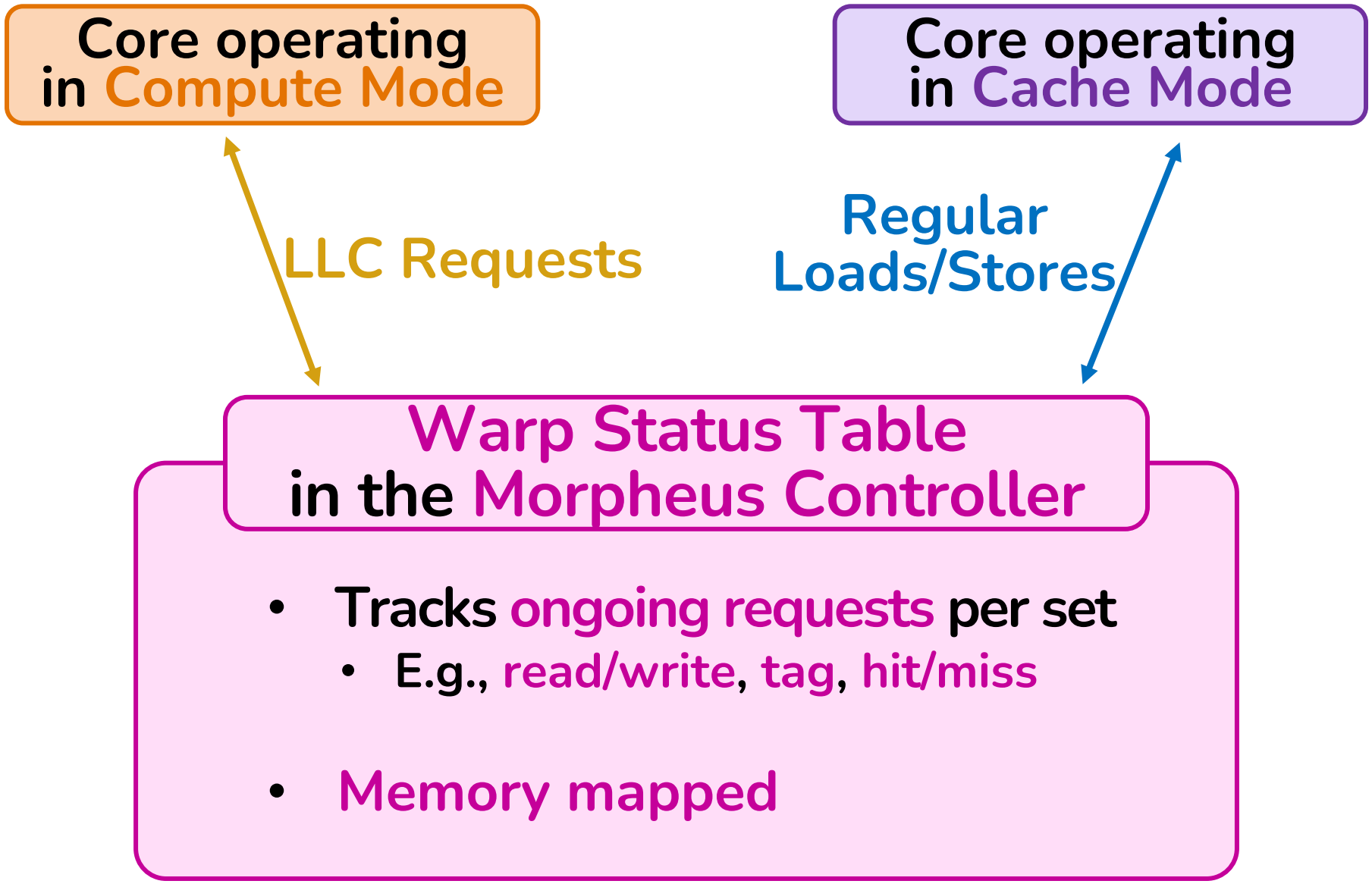
2 **Query Conventional LLC**

Physically Co-Located, Simple Forwarding

3 **Query Extended LLC**

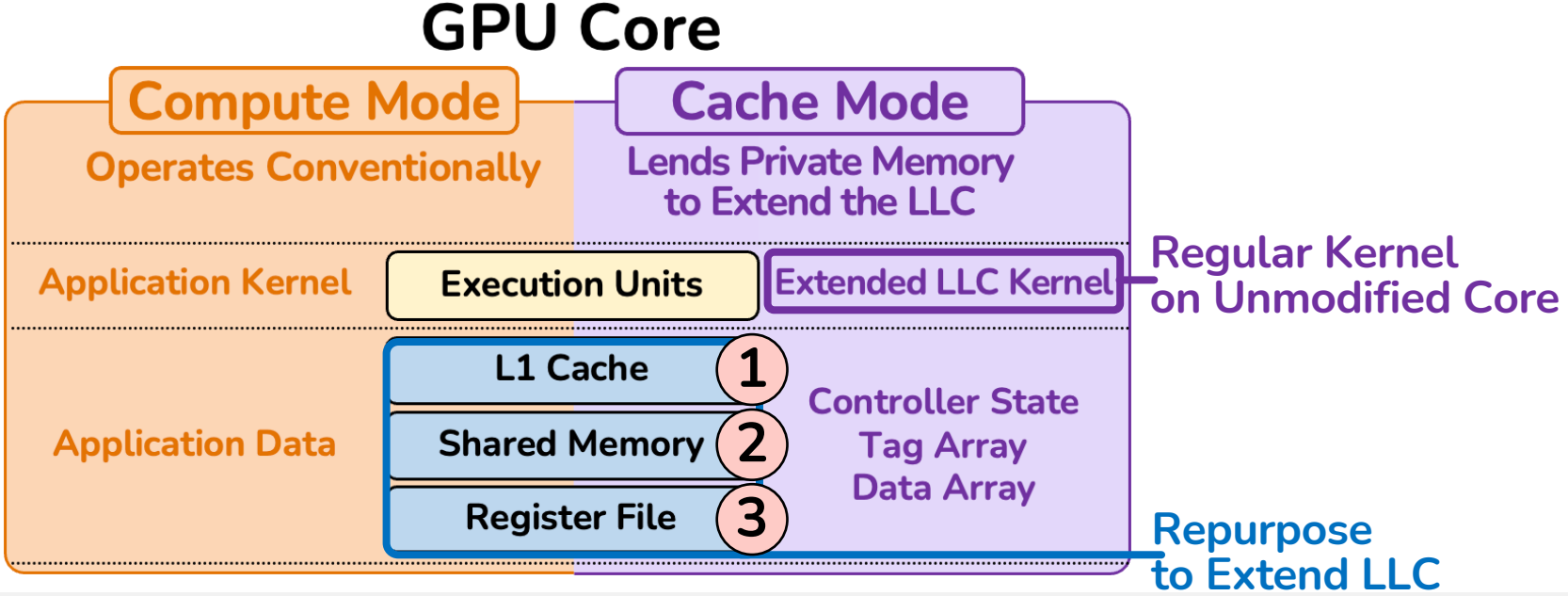
Warp Status Table

Morpheus Controller Warp Status Table



Outline

Extended LLC Kernel



1 **Query L1 Cache**

Load/Store Instructions

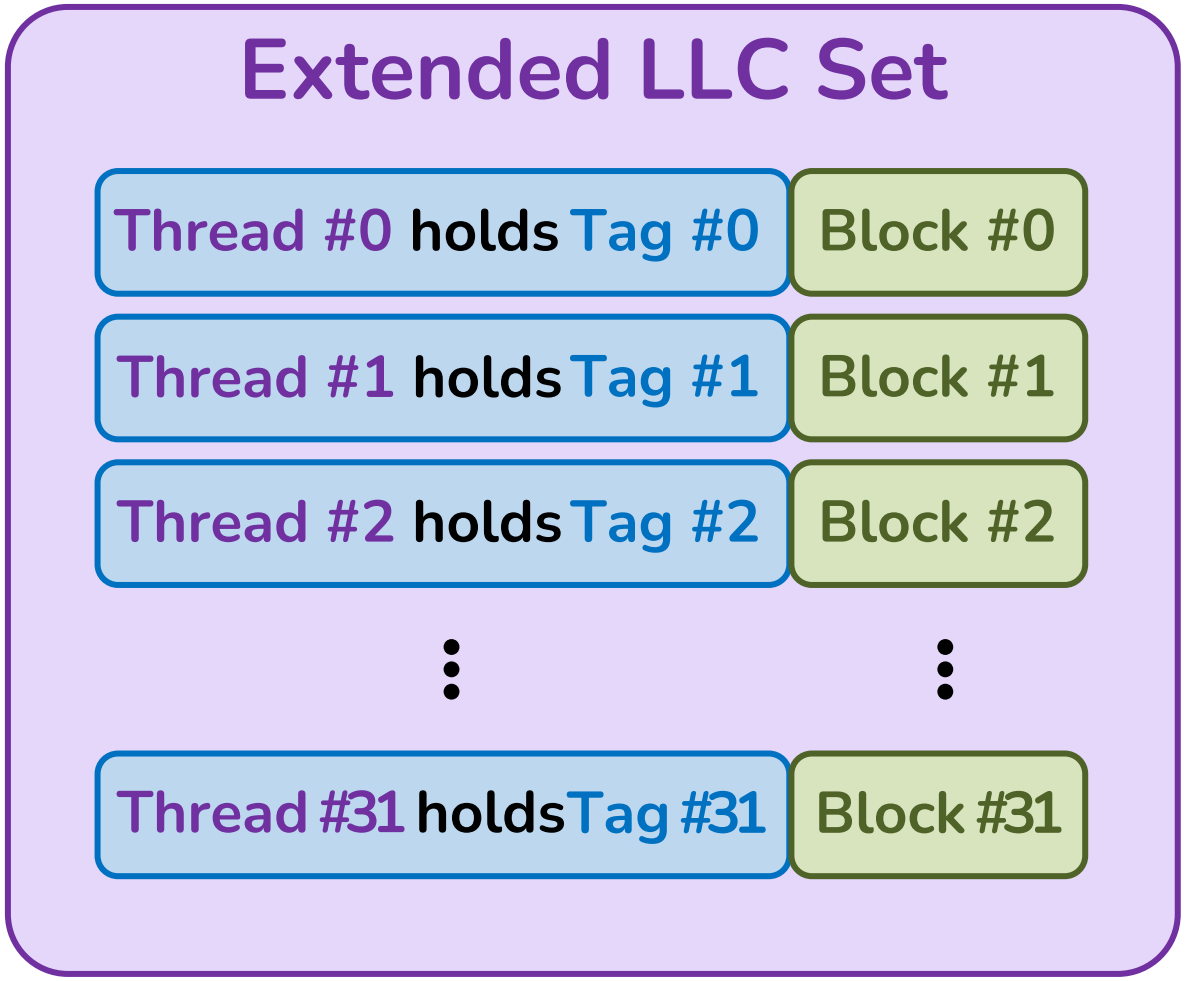
2 **Query Shared Memory**

Software Engineering

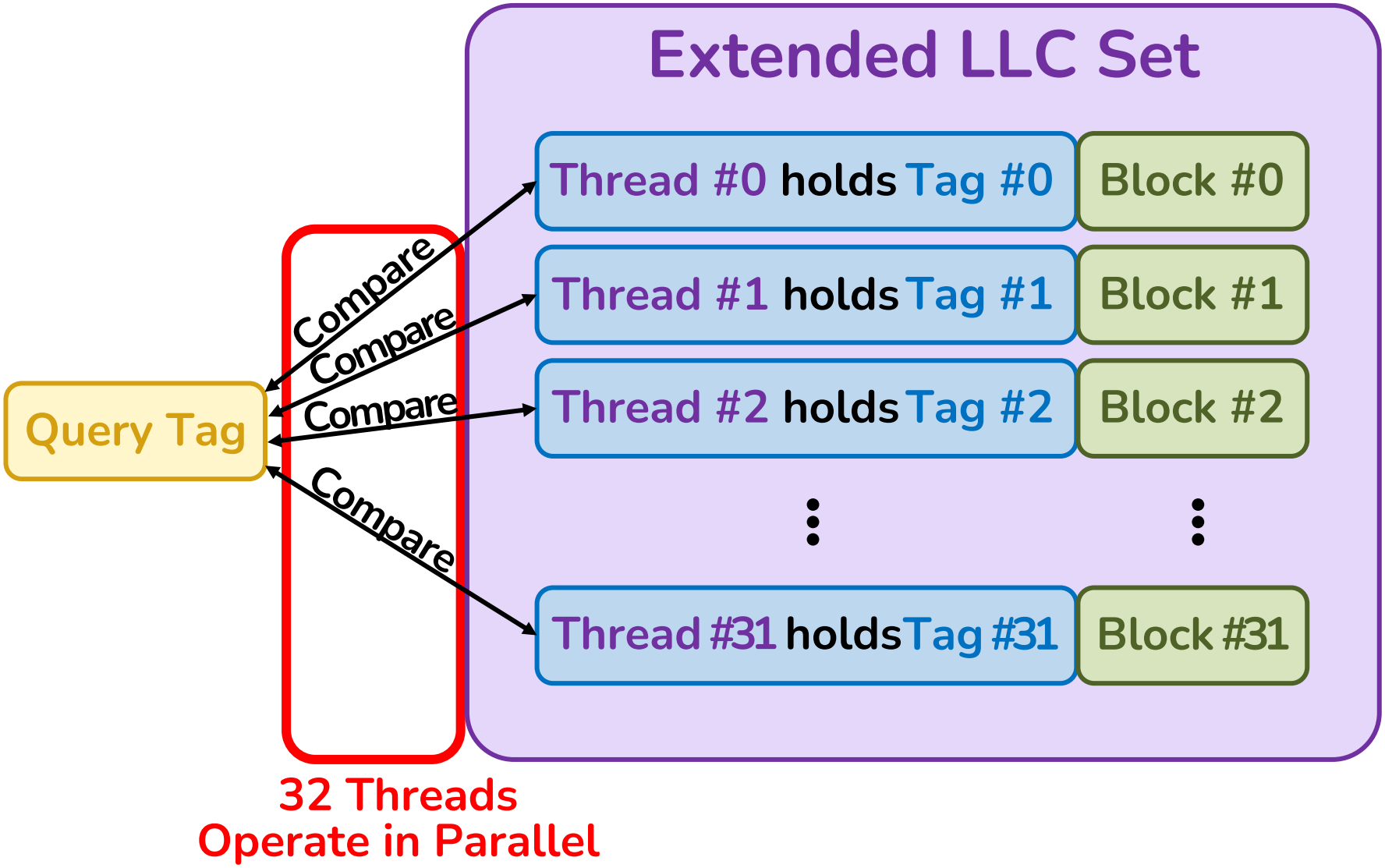
3 **Query Register File**

Data Layout across Threads for Parallelism

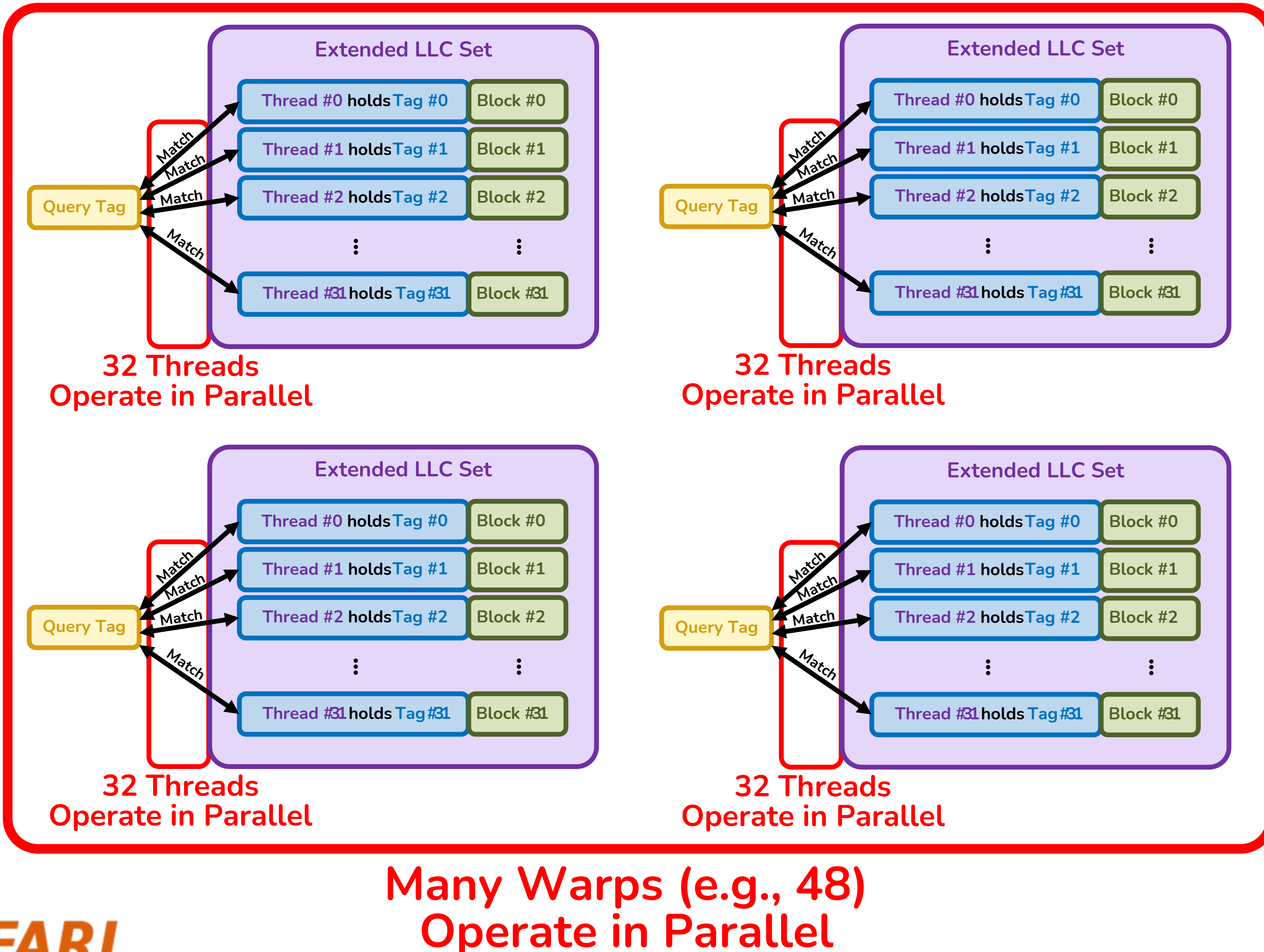
Extended LLC via Register File



Extended LLC via Register File



Extended LLC via Register File



SAFARI

More in the Paper: Mechanisms

- **Detailed mechanisms**
  - Morpheus Controller
  - Extended LLC kernel

- **Optimization techniques**
  - Hit/miss prediction
  - Indirect-MOV
  - Cache compression applied to Morpheus

Morpheus on arXiv

Outline

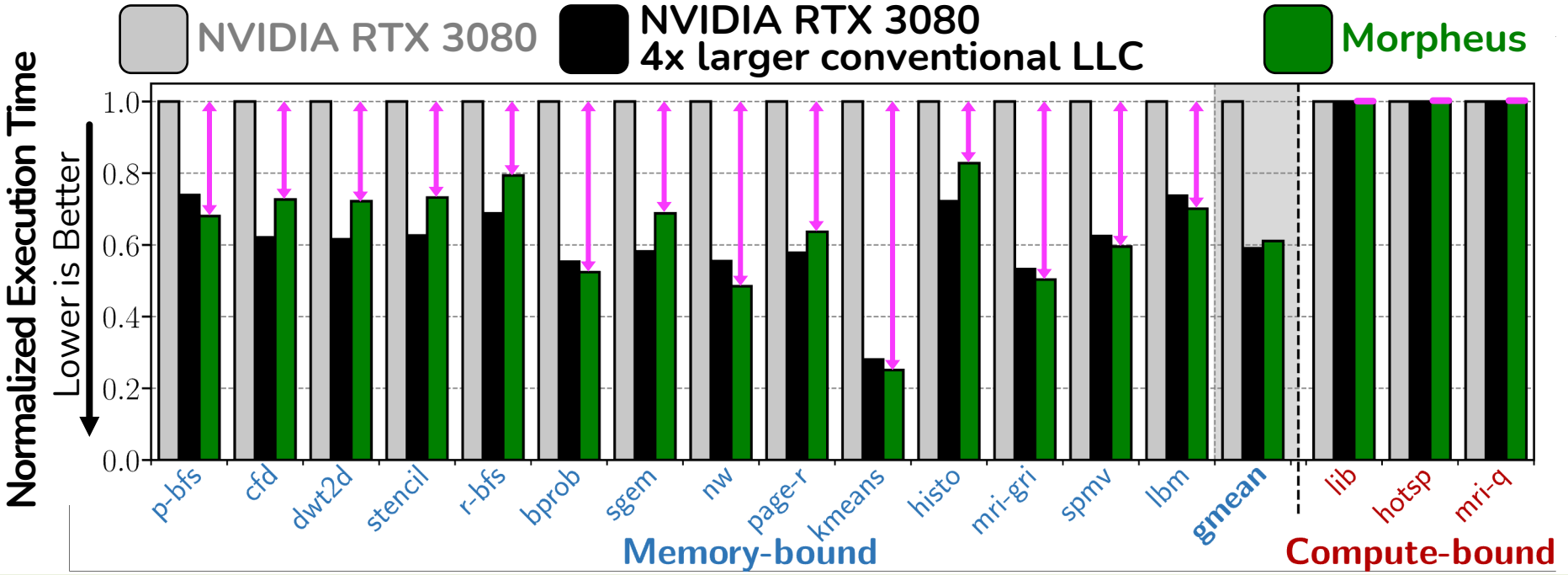
Methodology

- **17 representative applications**
  - 14 memory-bound
  - 3 compute-bound

- **Experimentally characterize the *extended LLC***
  - On a real NVIDIA RTX 3080, as the *extended LLC kernel* requires no special hardware
  - Capacity, latency, bandwidth, energy/byte of the *extended LLC*

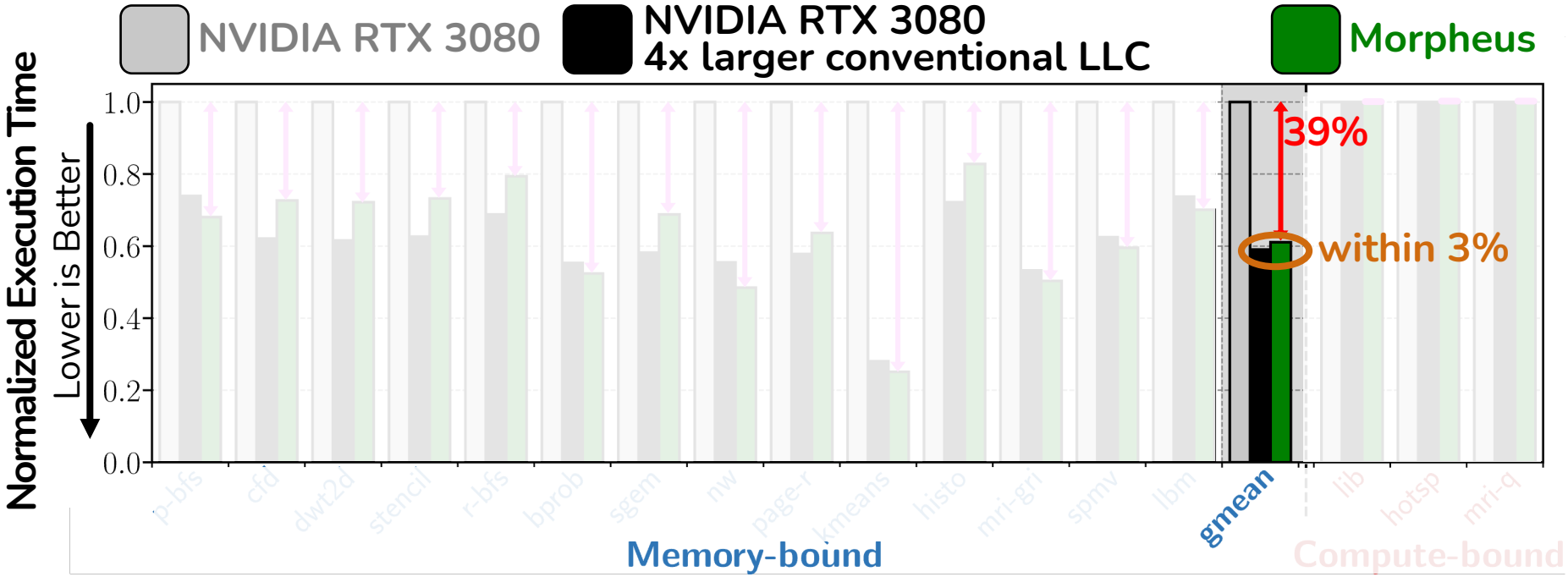
- **Simulate a Morpheus-enabled GPU using AccelSim** [Khairy+ ISCA'20]
  - Performance, energy efficiency, bandwidth utilization

Morpheus Performance



Morpheus performs **better** than or **equal** to the baseline for **all** applications

Morpheus Performance

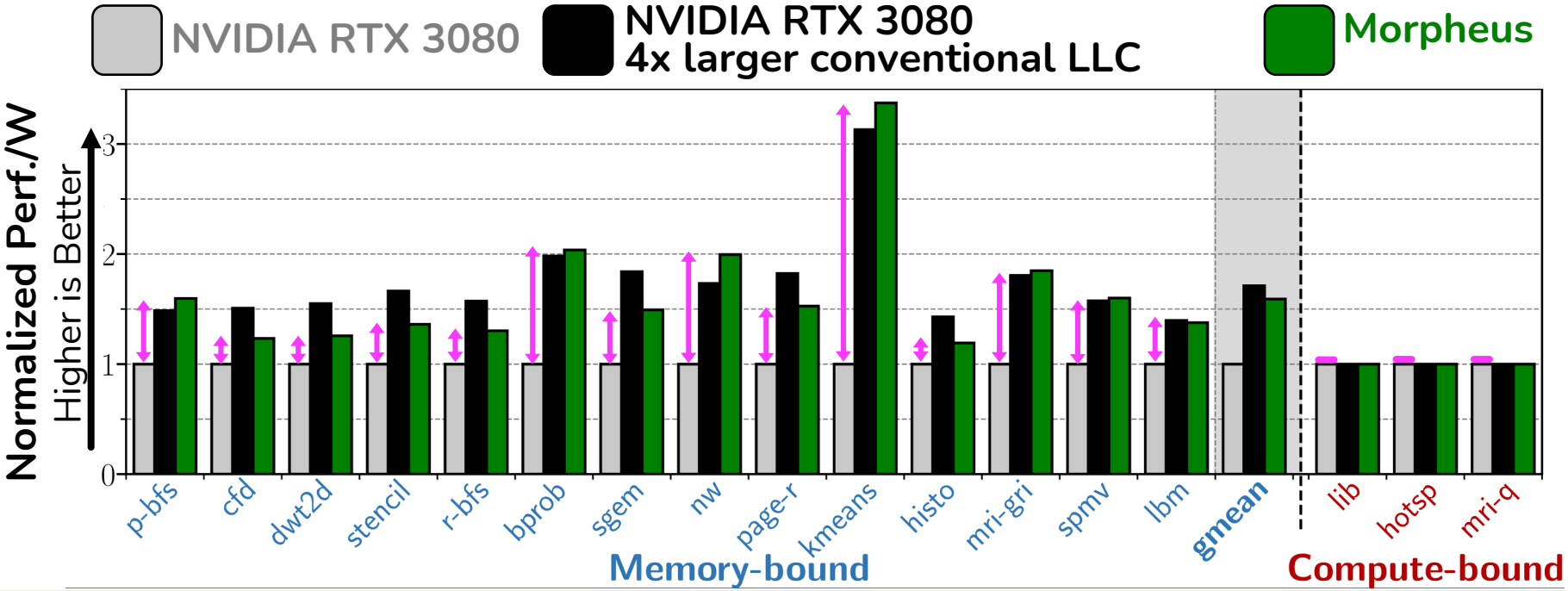


Morpheus performs **better** than or **equal** to the baseline for **all** applications

Morpheus **accelerates memory-bound** applications by **39%** on average

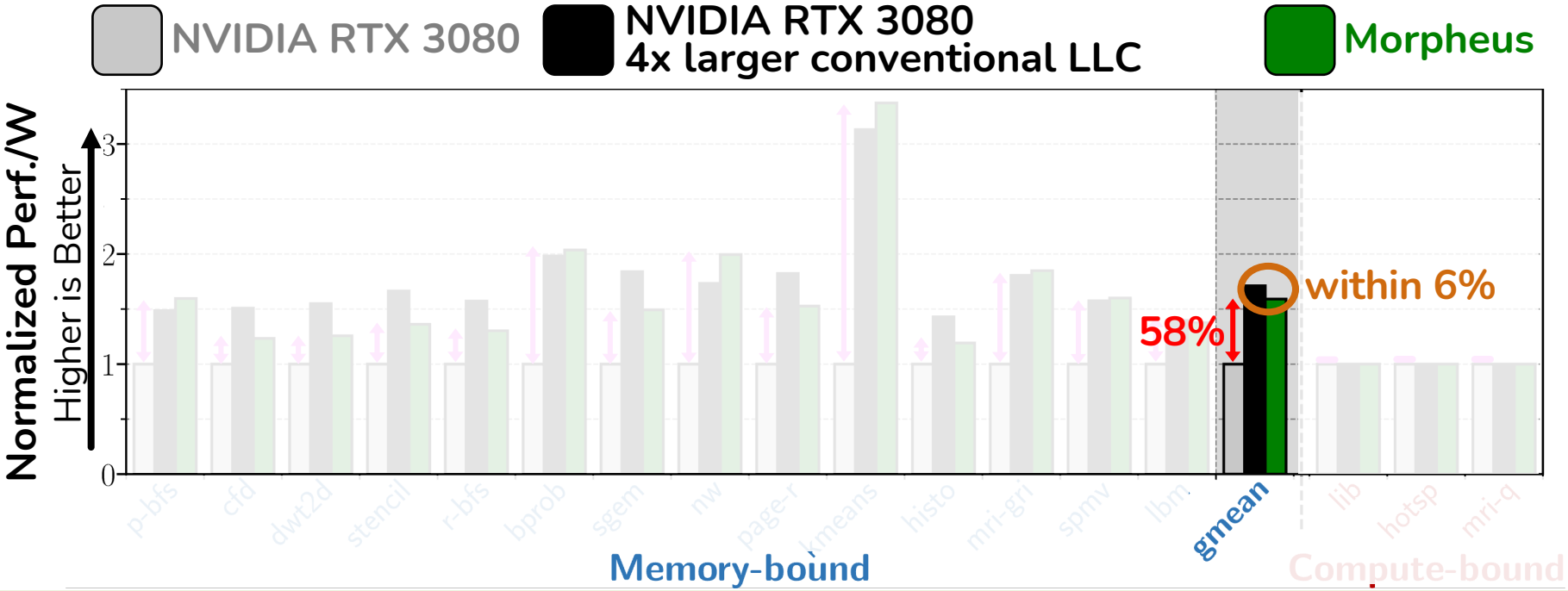
Morpheus **performs within 3%** of a **4x larger** conventional **LLC**

Morpheus Energy Efficiency



Morpheus uses **less** or the **same** energy as the baseline for **all** applications

Morpheus Energy Efficiency



Morpheus uses **less** or the **same** energy as the baseline for **all** applications

Morpheus **improves** energy efficiency by **58%** on average for **memory-bound** applications

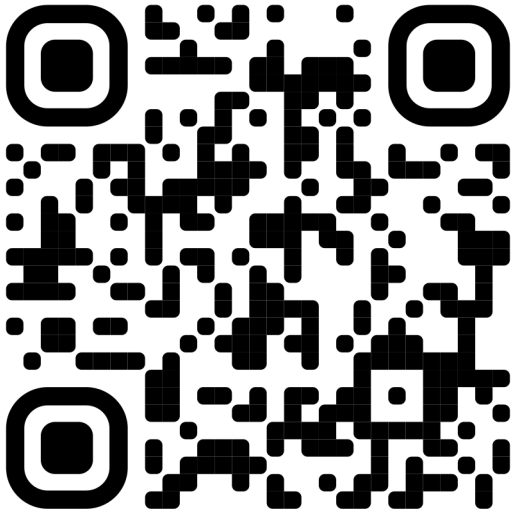
Morpheus' energy efficiency is **within 6%** of a **4x larger** conventional **LLC**

More in the Paper: Evaluation

Morpheus: Extending the Last Level Cache Capacity in GPU Systems Using Idle GPU Core Resources

Sina Darabi[†] Mohammad Sadrosadati[§] Negar Akbarzadeh[†] Joël Lindegger[§] Mohammad Hosseini[‡]
Jisung Park[§∇] Juan Gómez-Luna[§] Onur Mutlu[§] Hamid Sarbazi-Azad[†‡]

[†]*Sharif University of Technology* [§]*ETH Zürich*
[‡]*Institute for Research in Fundamental Sciences (IPM)* [∇]*POSTECH*



More in the Paper: Evaluation

- Characterization of extended LLC on a real GPU
- Performance and energy comparisons with more baselines
- Total LLC throughput with Morpheus
- On-chip and off-chip memory bandwidth utilization
- Storage and power overhead analysis

Outline

Summary

Motivation	**Memory-bound** GPU applications leave some **cores under-utilized**
Goal	**Leverage** the **under-utilized cores** to boost the **performance** and **energy efficiency** of **memory-bound applications**
Morpheus	**First** technique that leverages some **GPU cores' private memories** to **extend** the total GPU **last-level cache capacity**
Key Results	Morpheus outperforms state-of-the-art GPU architectures for memory-bound applications • **39% performance** improvement (relative to RTX 3080) • **58% energy efficiency** improvement (relative to RTX 3080) • Morpheus performs **within 3%** of a **4x larger conventional LLC** • Morpheus provides a **4x larger LLC** with the same LLC hardware
Conclusion	Morpheus **effectively repurposes** the **private memory** of under-utilized cores to **extend the GPU LLC capacity**

Morpheus

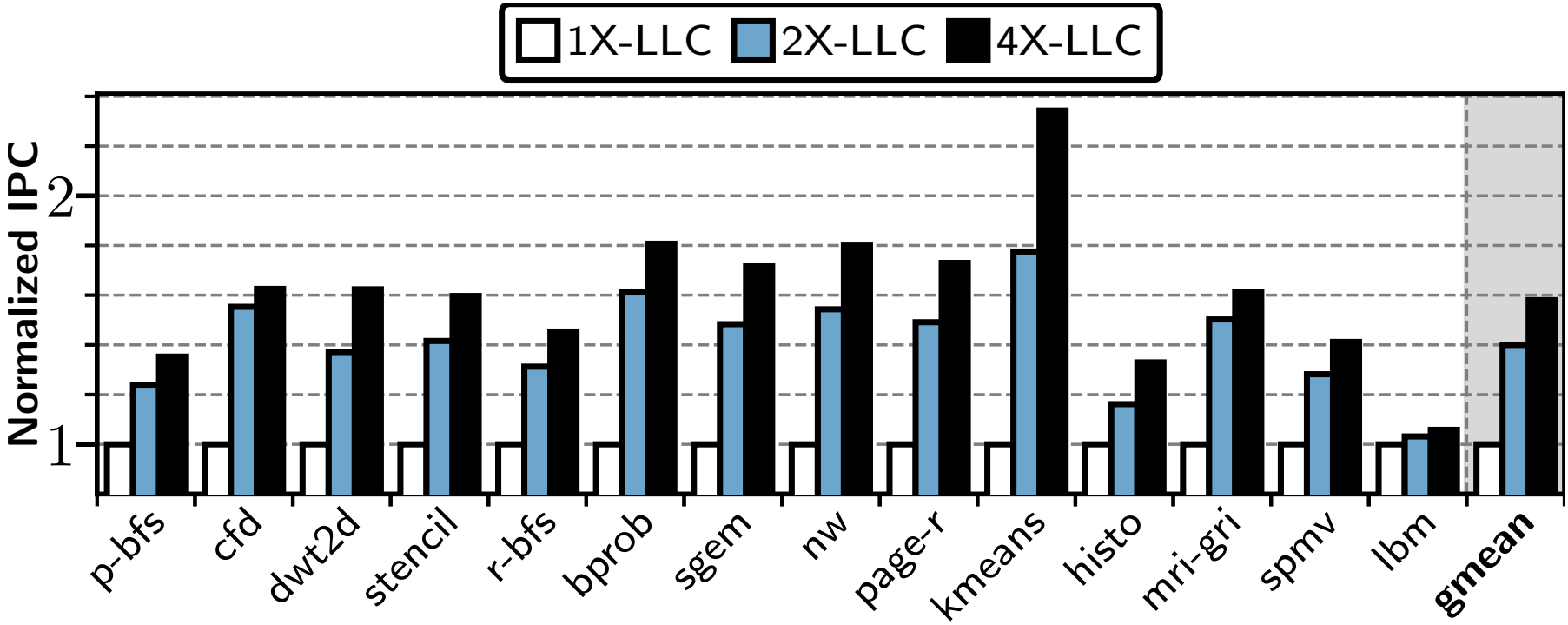
Extending the Last Level Cache Capacity in GPU Systems with Idle GPU Core Resources

Sina Darabi, Mohammad Sadrosadati, Negar Akbarzadeh,
Joël Lindegger, Mohammad Hosseini, Jisung Park, Juan Gómez-Luna,
Onur Mutlu, Hamid Sarbazi-Azad

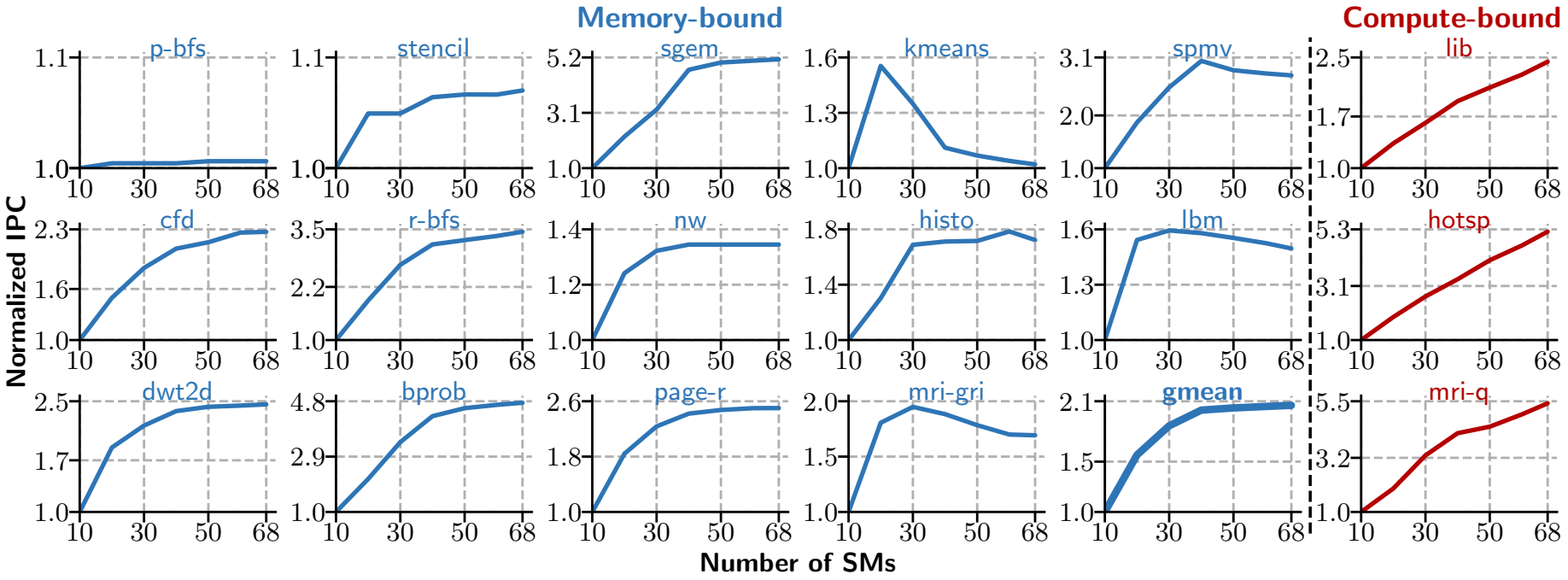
October 3rd 2022
MICRO-55

Backup Slides

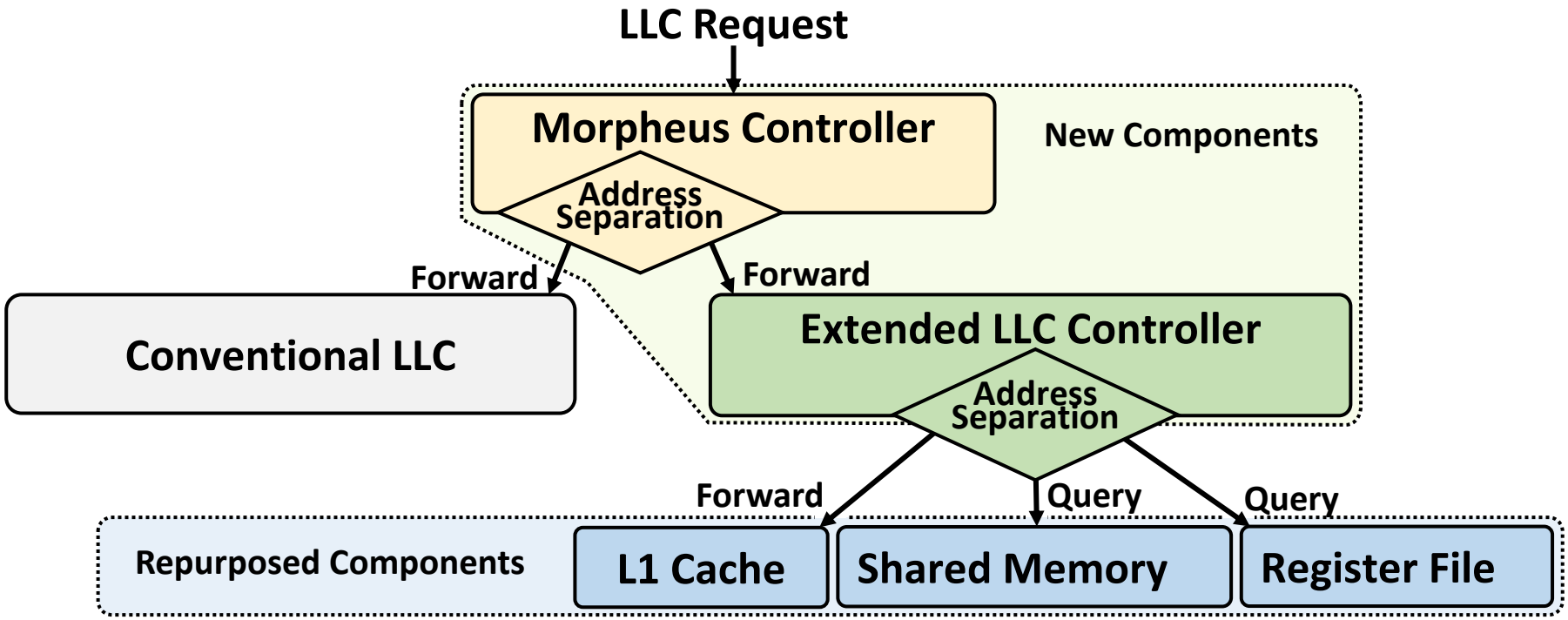
Performance Benefits of a Larger LLC



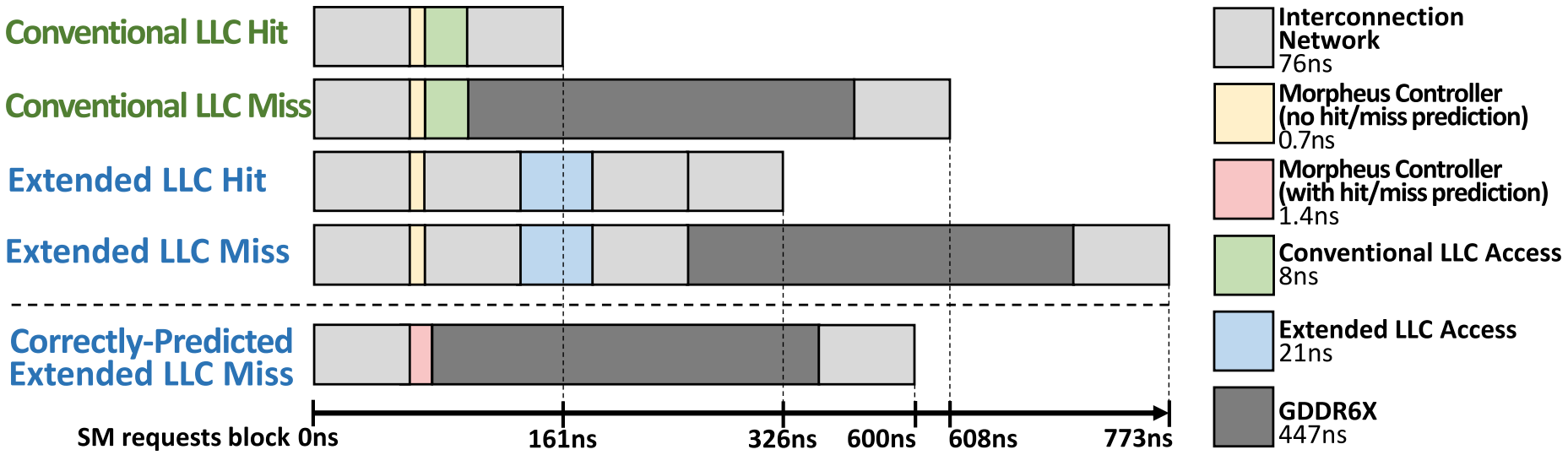
Memory-bound vs. Compute-bound Applications



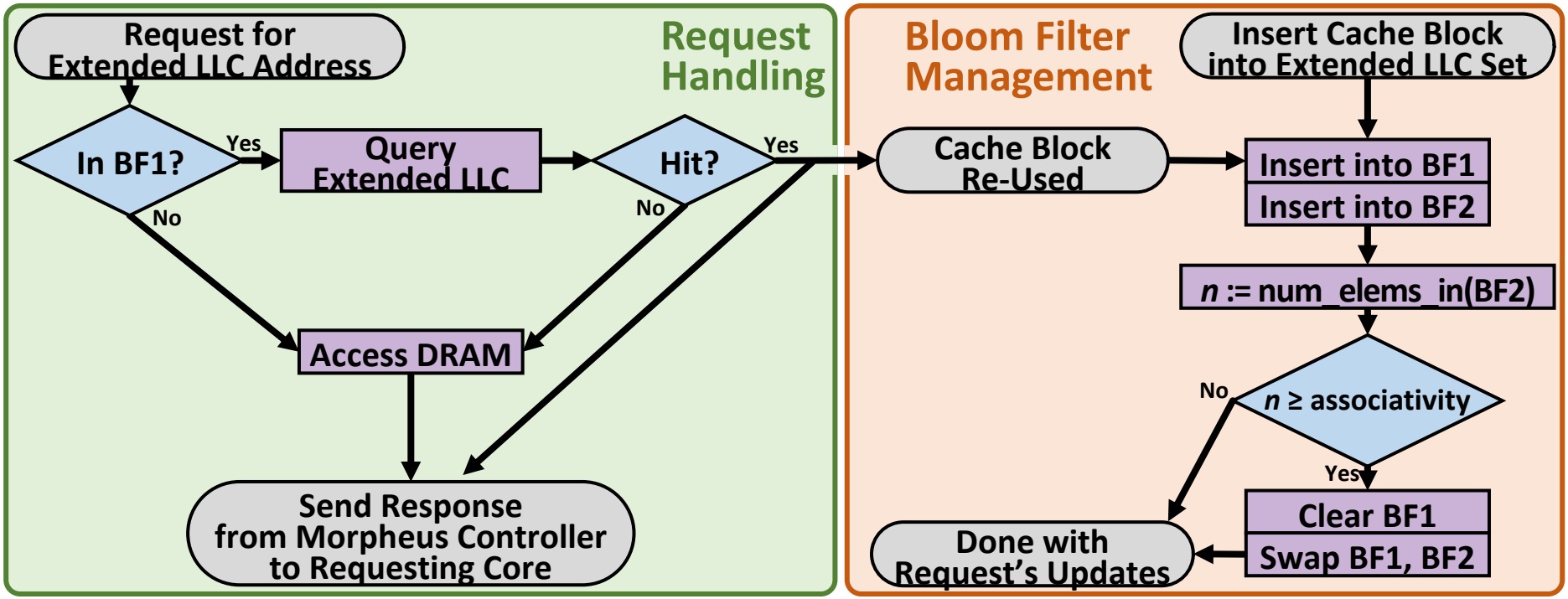
Morpheus' Logical Data Flow



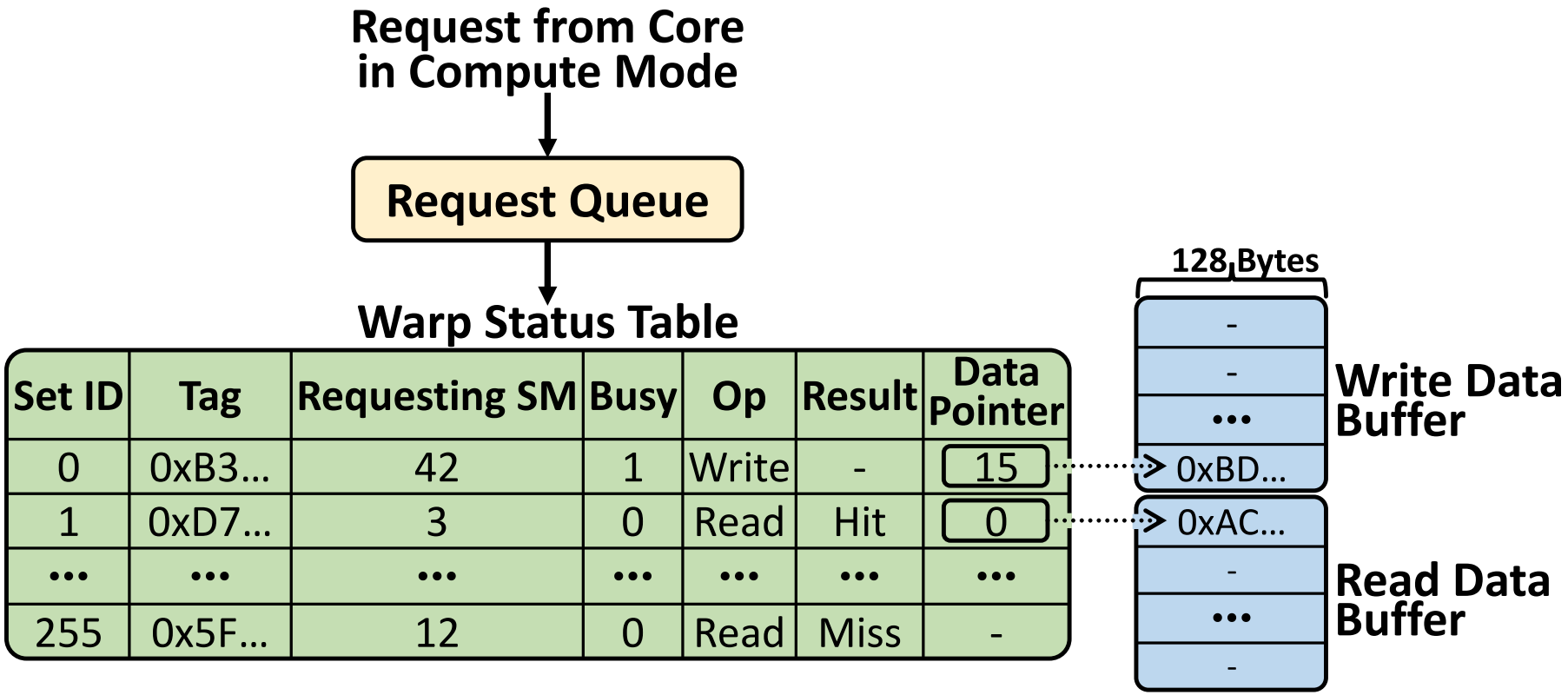
LLC Timelines



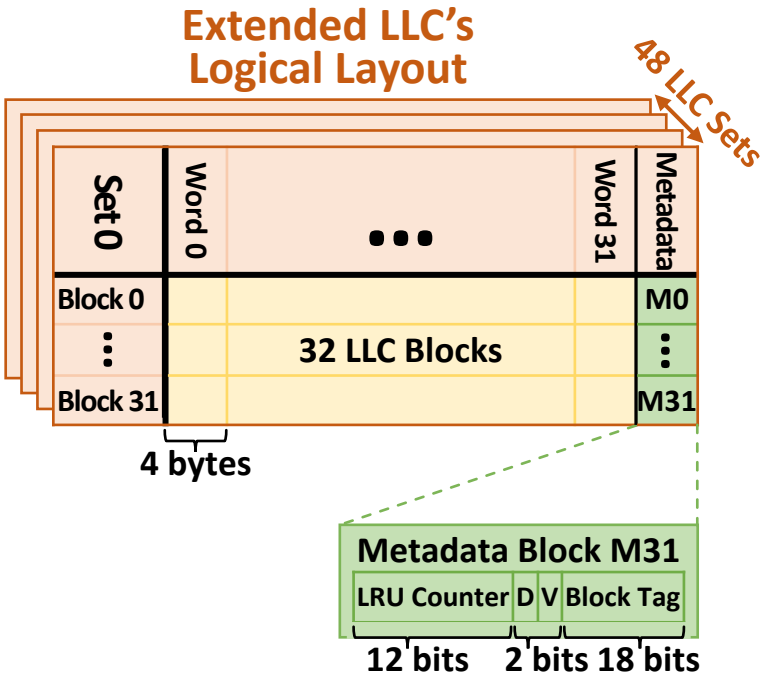
Hit/Miss Predictor

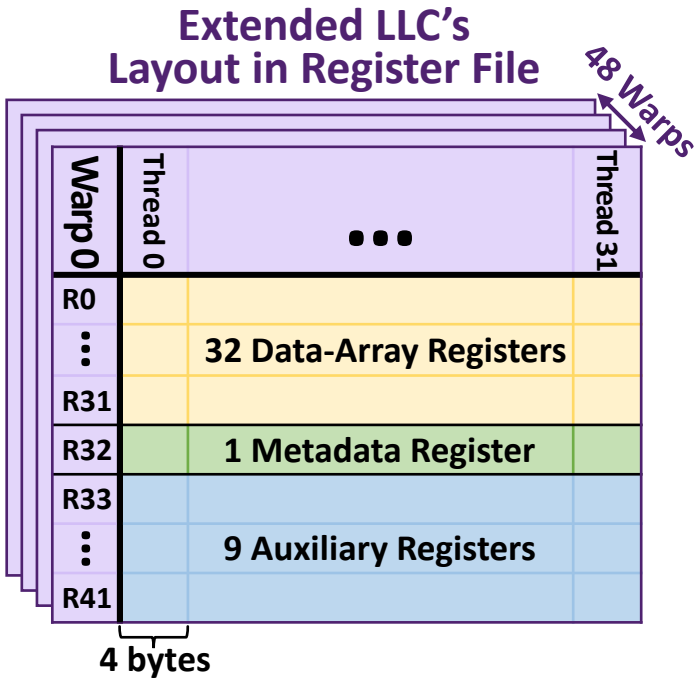


Extended LLC Query Logic Unit

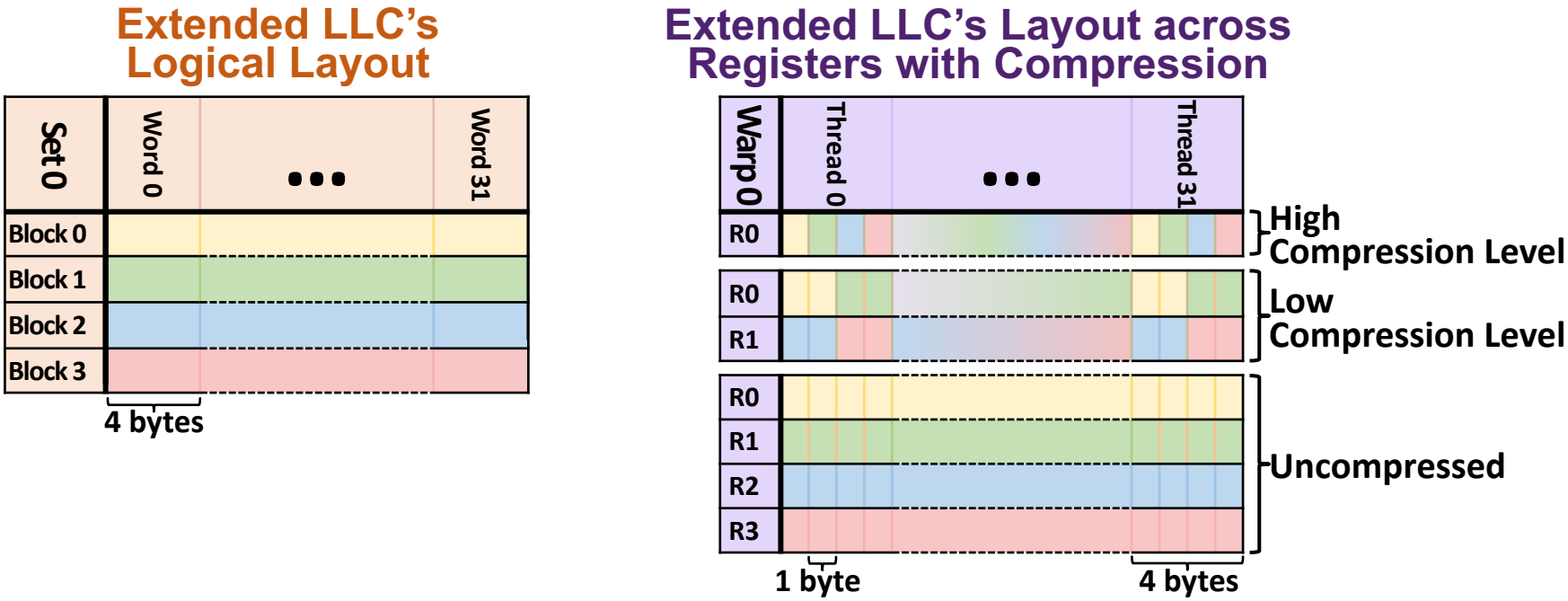


Registerfile Layout





Cache Compression



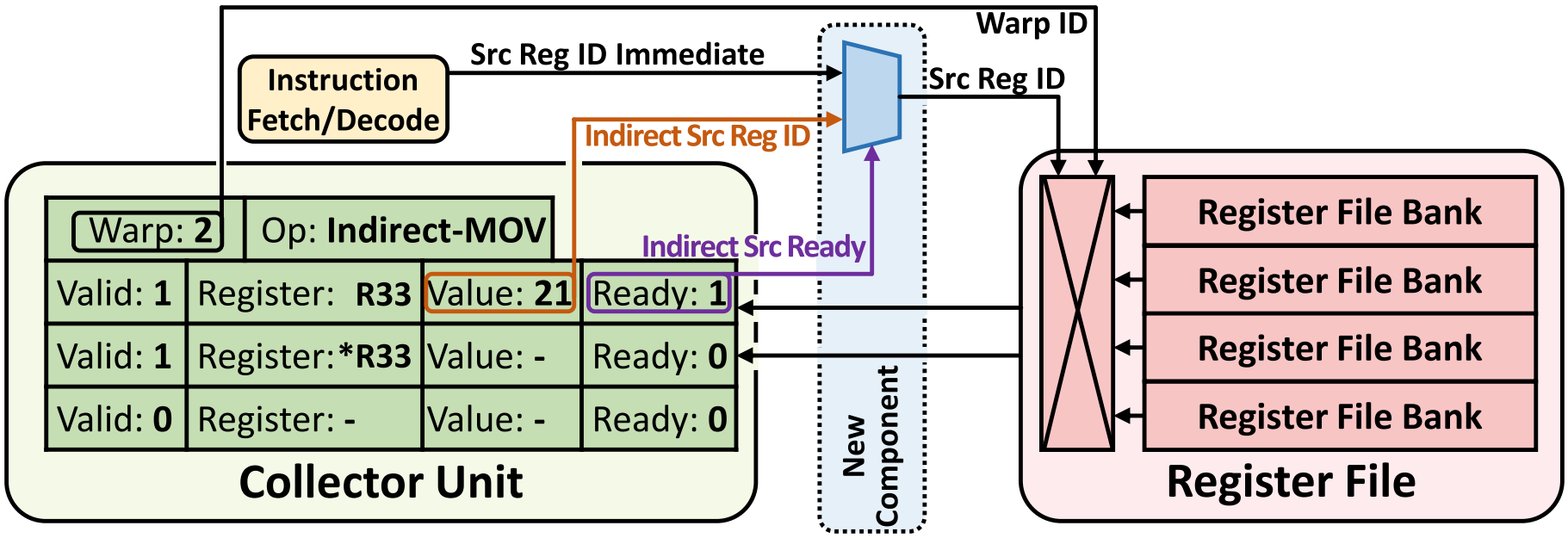
Indirect-MOV Algorithm

Algorithm 2 Indirect-MOV Algorithm

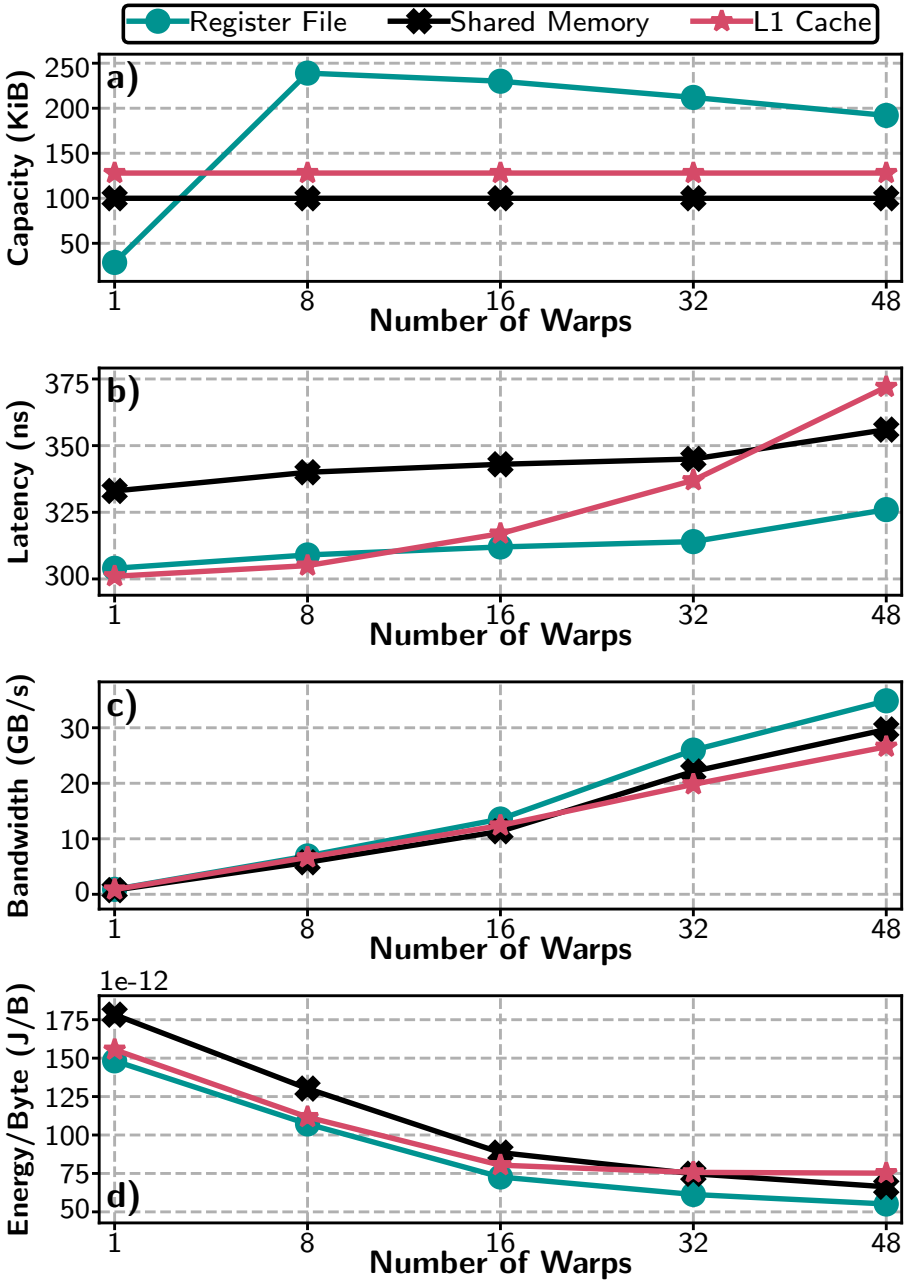
Input: BLOCK_INDEX:int (R_{aux_3})
Output: Requested Extended LLC Block (R_{aux_0})

1: **procedure** INDIRECT-MOV //The goal is to implement register indirect access, reading from a register whose index is determined by accessing the value in another register. This procedure is critical for accessing data-array registers in the extended LLC kernel.
2: T_{list} : *.Branch Targets* $L_0, L_1, L_2, \ldots, L_{31}$; //Define 32 branch targets, each is allocated to access a specific register index.
3: *@p brx.idx* R_{aux_3}, T_{list}; //Branch to label L_i specified by the target LLC block index i=R_{aux_3}
4: L_0 :
5: *MOV* R_0, R_{aux0} //Access data-array register R_0 if target LLC block index is 0
6: *return*
7: L_1 :
8: *MOV* R_1, R_{aux0} //Access data-array register R_1 if target LLC block index is 1
9: *return*
10: ...
11: L_{31} :
12: *MOV* R_{31}, R_{aux0}//Access data-array register R_{31} if target LLC block index is 31
13: *return*
14: **end procedure**

Indirect-MOV Instruction



Extended LLC Characterization



Simulation Parameters

- NVIDIA RTX 3080

Parameter	Value
Number of SMs	68
Scheduler	Two-Level [18, 38]
GPU Memory Interface	320-bit GDDR6X [39]
GPU Memory Capacity	10 GiB
Conventional LLC Capacity	5 MiB
L1/Shared-Memory Capacity	128 KiB per SM
Register File Capacity	256 KB per SM

Number of Compute Mode Cores

Table 3: Number of GPU cores executing application threads for different evaluated systems (#available GPU cores is 68).

Application	p-bfs	cfd	dwt2d	stencil	r-bfs	bprob	sgem	nw	page-r	kmeans	histo	mri-gri	spmv	lbm	lib	hotsp	mri-q
IBL	68	68	68	68	68	68	68	68	68	24	53	34	42	34	68	68	68
Morpheus-Basic	32	42	42	50	34	39	48	18	42	37	47	36	44	32	68	68	68
Morpheus-ALL	40	55	54	56	37	41	54	26	46	47	52	43	47	36	68	68	68

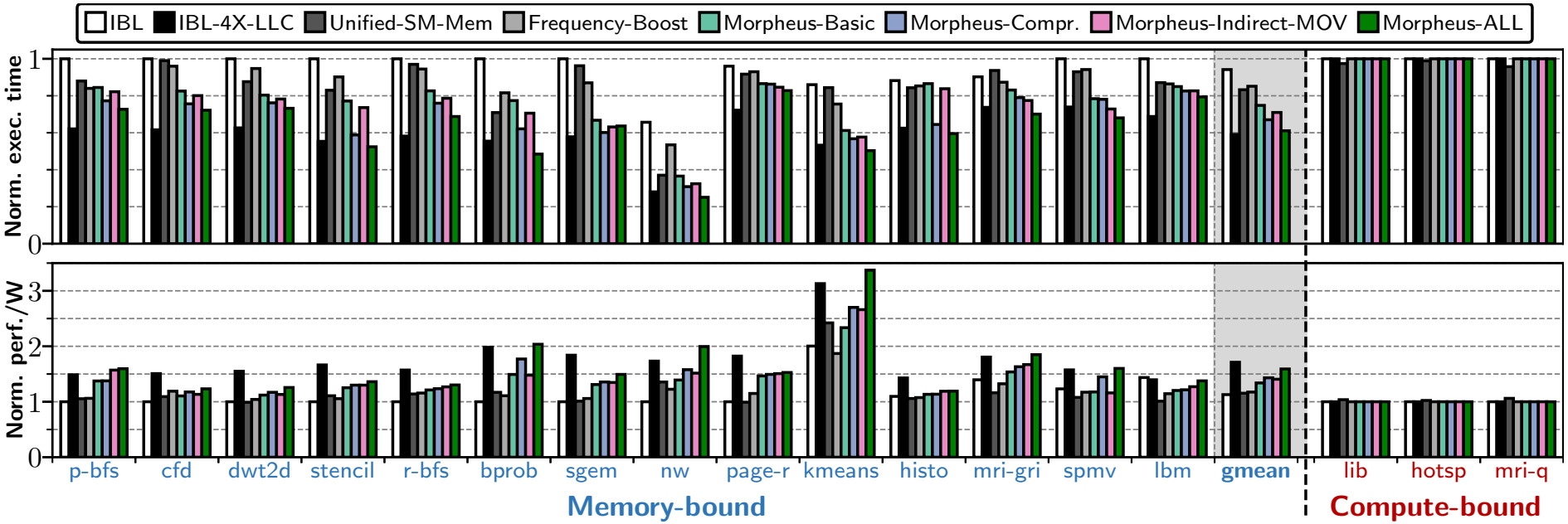
Morpheus LLC Capacity (NVIDIA RTX 3080)

- **Conventional LLC:** 5 MiB

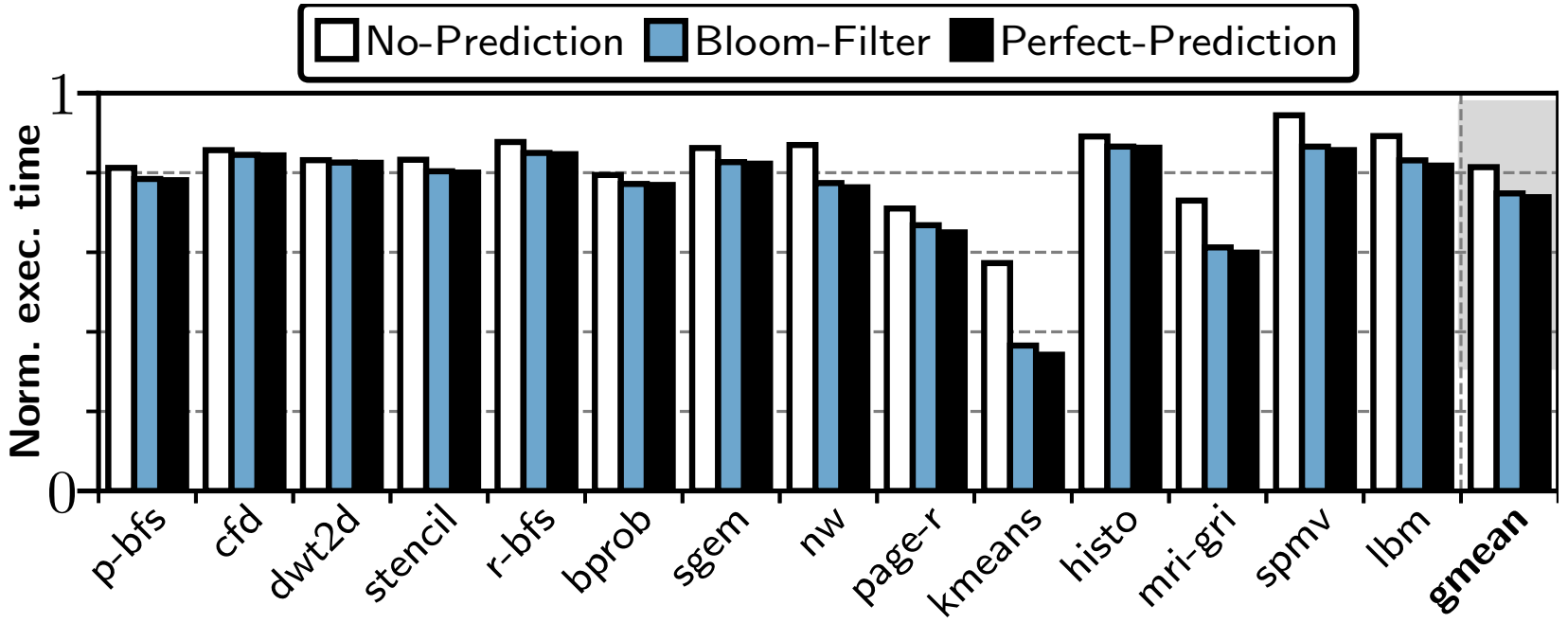
- **Extended LLC Per Core in Cache Mode:** 328 KiB
  - 128 KiB from L1
  - 256 KiB from Register File
    - » 200 KiB useable to extended LLC data

- **Number of Cache Mode Cores:** Application Dependent
  - E.g. 50 for Needleman-Wunsch (nw) => 16 MiB
  - Theoretically up to 67 => 21.4 MiB

Morpheus Energy and Power



Hit/Miss Prediction Benefits



Evaluated Applications

Application	Name	Type
Breadth-First Search [6]	p-bfs	Memory-bound
Computational fluid dynamics [5]	cfd	Memory-bound
Discrete Wavelet Transform (2D) [5]	dwt2d	Memory-bound
Stencil [6]	stencil	Memory-bound
Breadth-First Search [5]	r-bfs	Memory-bound
Back Propagation [5]	bprob	Memory-bound
sgemm [6]	sgem	Memory-bound
Needleman-Wunsch [5]	nw	Memory-bound
Page Rank [40]	page-r	Memory-bound
K-means [5]	kmeans	Memory-bound
Histogram [6]	histo	Memory-bound
Magnteic Resonance Imaging-Gridding [6]	mri-gri	Memory-bound
Sparse-Matrix Dense-Vector Multiplication [6]	spmv	Memory-bound
Lattice-Boltzmann [6]	lbm	Memory-bound
LIBOR Monte Carlo [41]	lib	Compute-bound
HotSpot [5]	hotsp	Compute-bound
Magnetic Resonance Imaging - Q [6]	mri-q	Compute-bound

Extended LLC Tag Lookup

Algorithm 1 Extended LLC Tag Lookup – Register File

Input: Extended LLC Request's Tag (R_{aux_0})
Output: HIT:bool, and if HIT=True, BLOCK_INDEX:int (R_{aux_3})

```
1:  procedure TAG LOOKUP//executed by an extended LLC kernel warp of 32 threads
2:      R_aux1 ← Valid(R_M) //ensure the block is valid
3:      R_aux1 ← R_aux1 && (R_aux0 == Tag(R_M)) //match request tag to metadata
4:      R_aux2 ← __ballot_sync(0xffffffff, R_aux1) //share R_aux1 between all
    threads as a 32-bit vector
5:      if (R_aux2) then //one of the bits is non-zero because there was a hit
6:          R_aux3 ← __ffs(R_aux2) − 1 //get the 0-based index of the non-zero bit
7:          HIT ← True
8:          BLOCK_INDEX ← R_aux3
9:          if (thread_idx == R_aux3) then //reset the LRU counter of the hit block
10:             LRU_Counter(R_M) ← 0xfff
11:         else//decrement the LRU counters of all other blocks
12:             LRU_Counter(R_M) ← LRU_Counter(R_M) − 1
13:         end if
14:     else
15:         HIT ← False
16:     end if
17: end procedure
```