Time: 13:40 - 14:05, January 23, 2025

Mpache: Interaction Aware Multi-level Cache Bypassing on GPUs

Mengyue Xi, Tianyu Guo, Xuanteng Huang, Zejia Lin, Xianwei Zhang

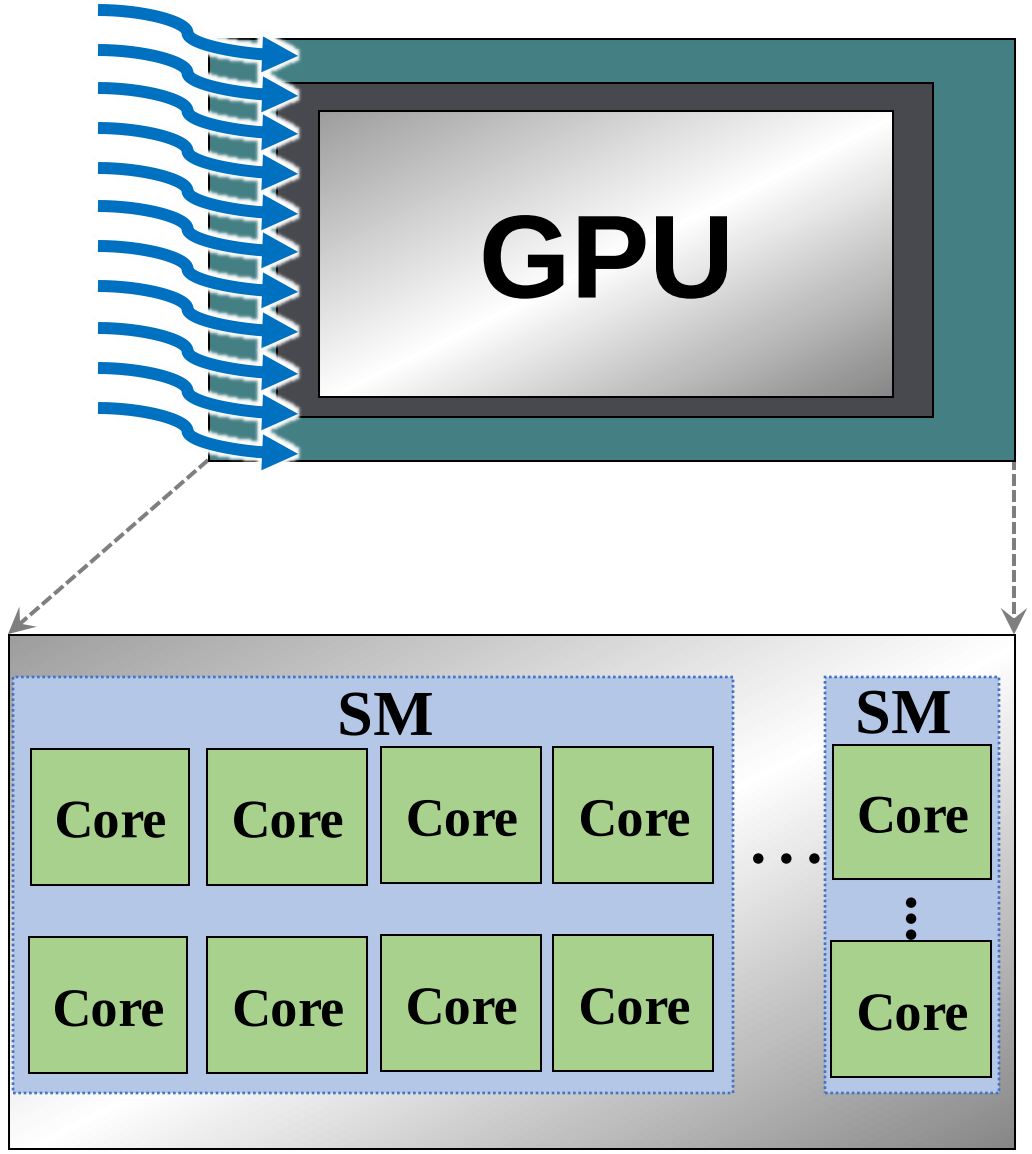
Email: ximy@mail2.sysu.edu.cn

SUN YAT-SEN UNIVERSITY

中山大學 SUN YAT-SEN UNIVERSITY

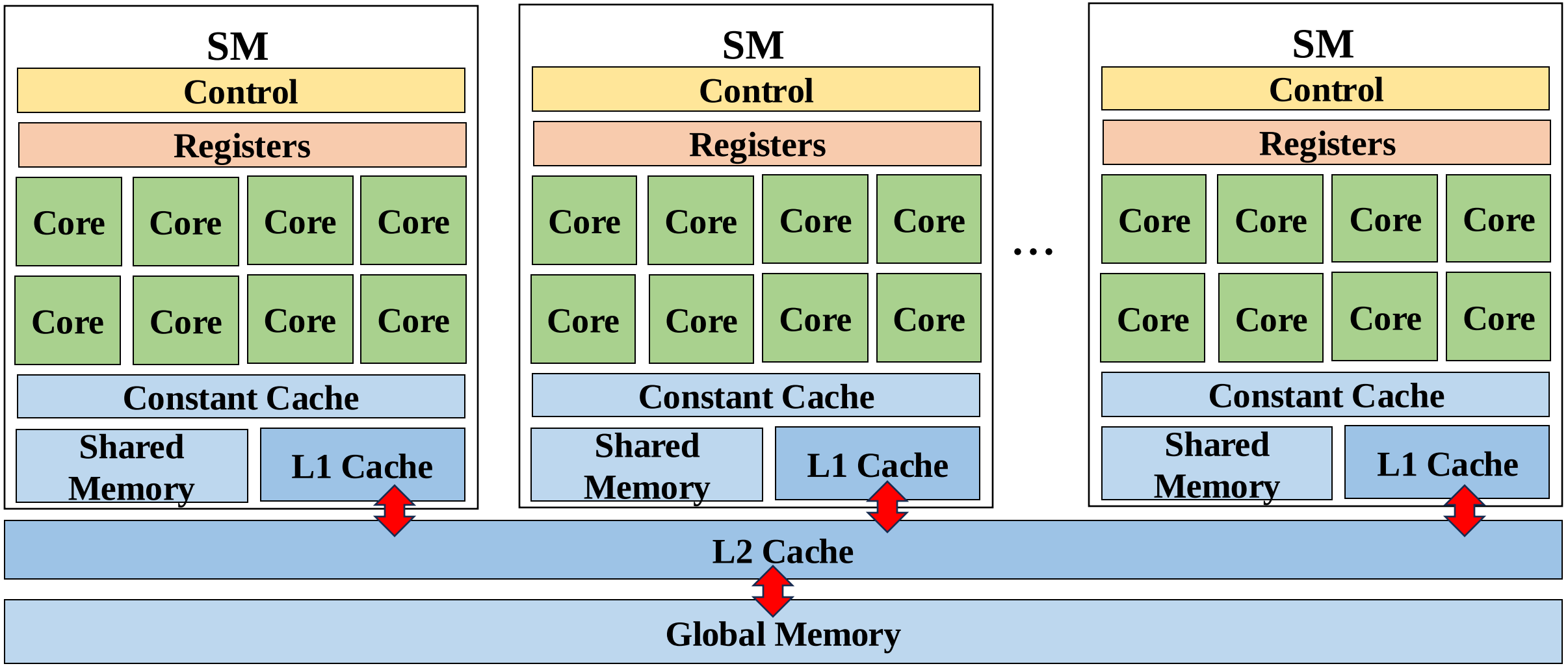
GPUs

- Designed for parallel computing
  - **Thousands of cores**
  - Executing **thousands of threads** simultaneously.
- Support diverse applications
  - Artificial intelligence (AI)
  - High-performance computing (HPC)
  - Graphics rendering
  - …

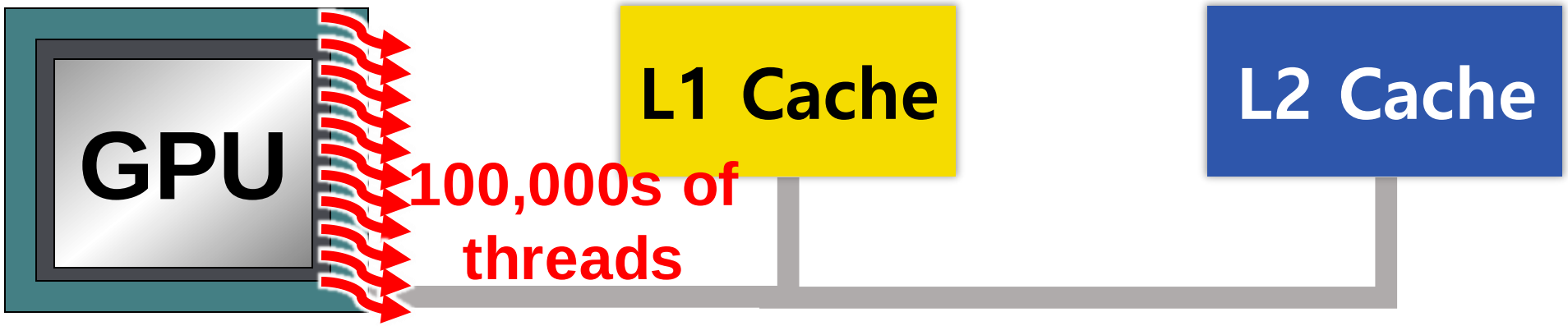


GPU Memory Hierarchy

- Using an Nvidia GPU as an example

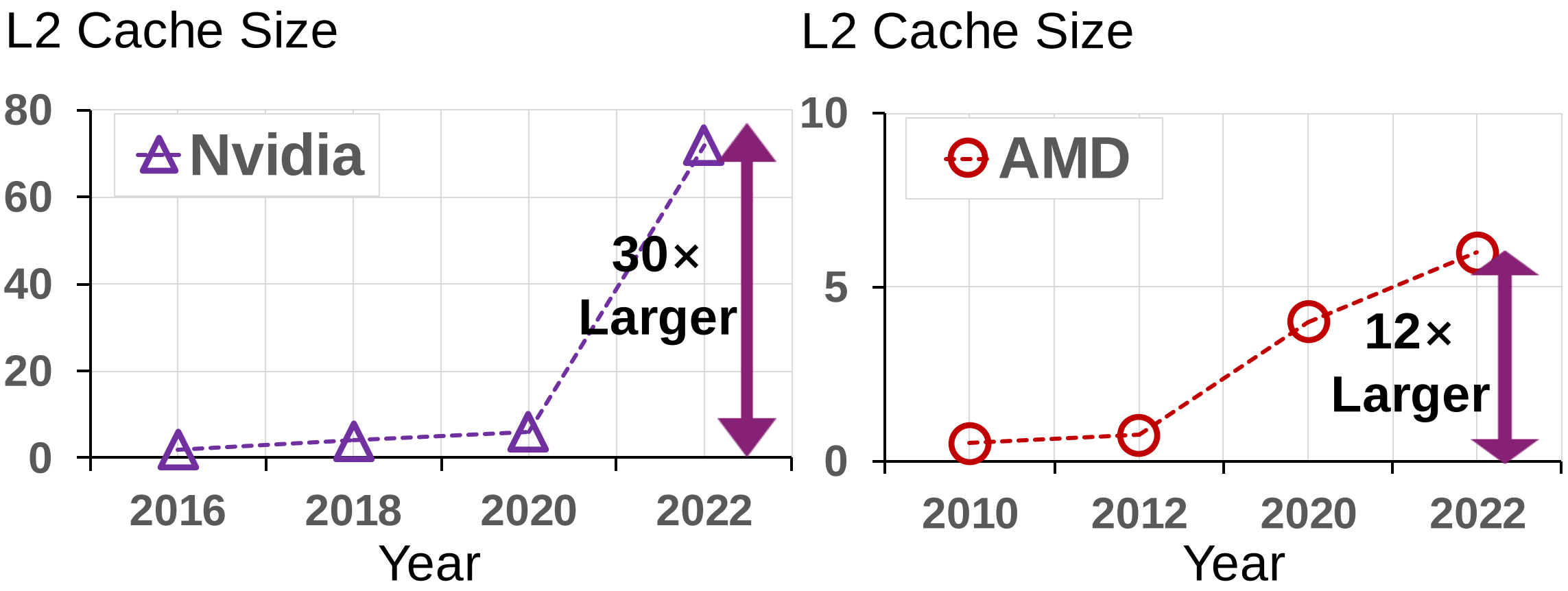


Cache Becomes Bottleneck



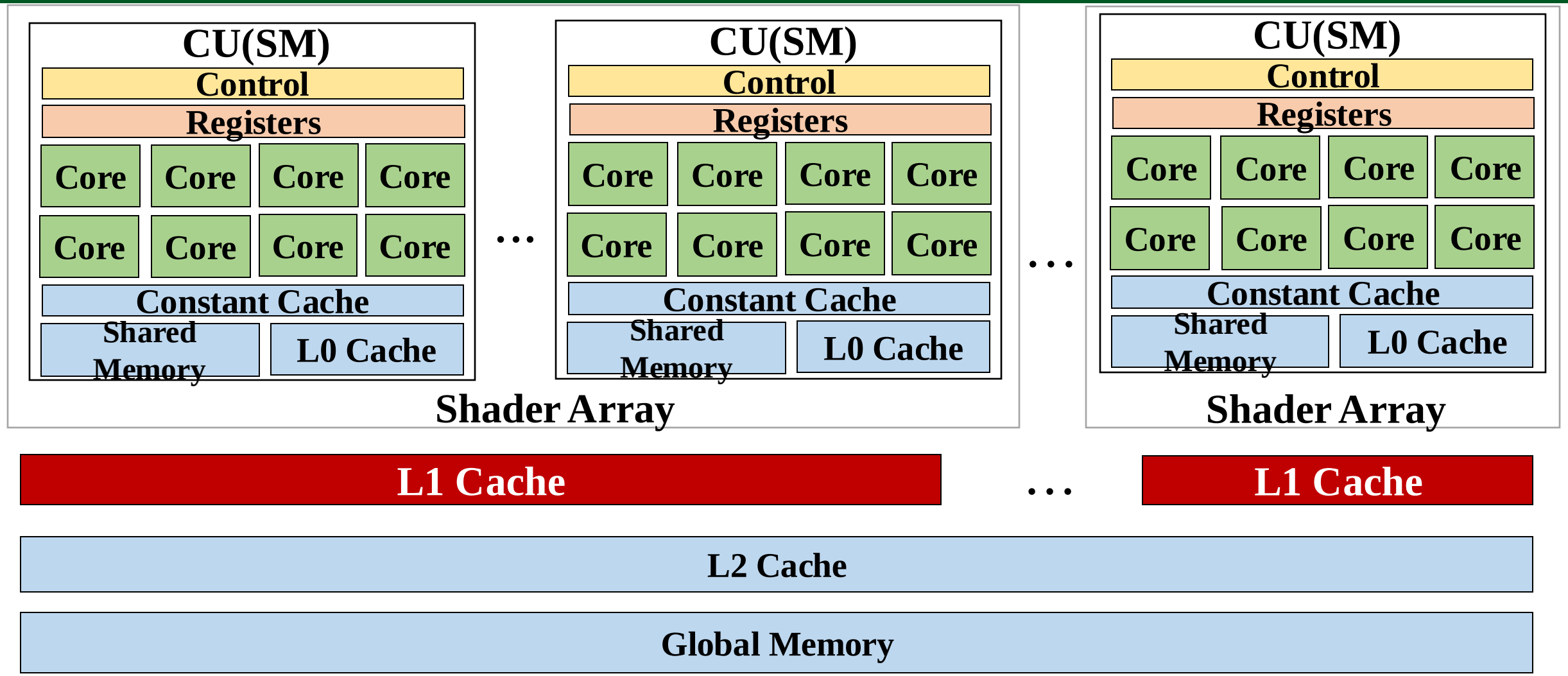
- Limited cache hit rates:
  - **Cache conflicts** caused by thousands of threads
  - **Irregular memory access patterns** reduce cache efficiency
- Cache pollution: **Streaming data** occupies cache space

Trend1: Enlarged Cache Capacity



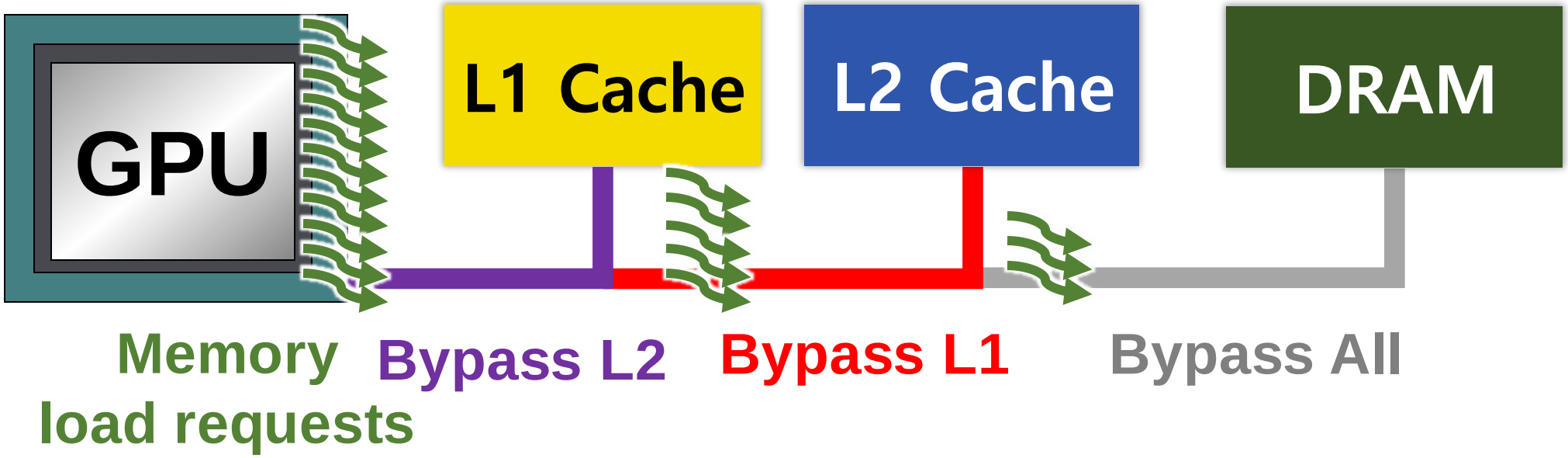
- Nvidia: **30x** increase from 2MB to 72MB
- AMD: **12x** increase from 512KB to 6MB

Trend2: Deepened Cache Levels



- AMD: Introduces **three cache levels** in its RDNA architectures

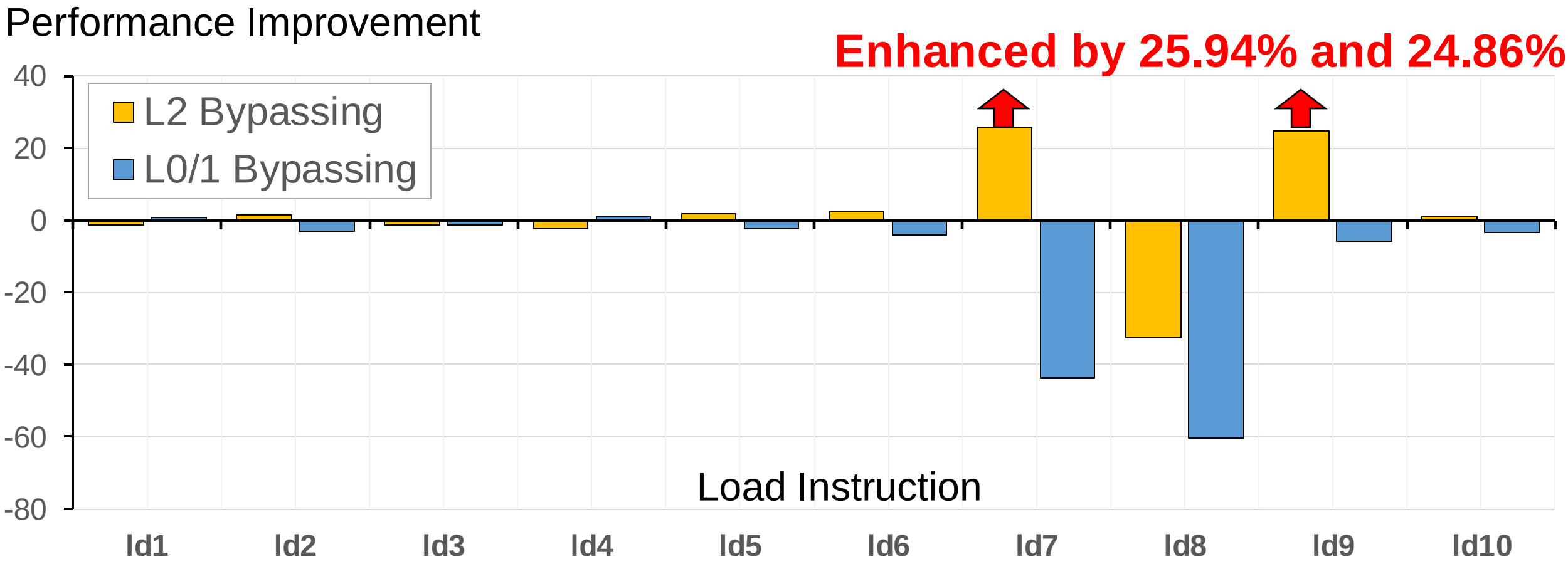
Optimization Opportunity: Bypass



→ Bypass a specific cache level for load accesses

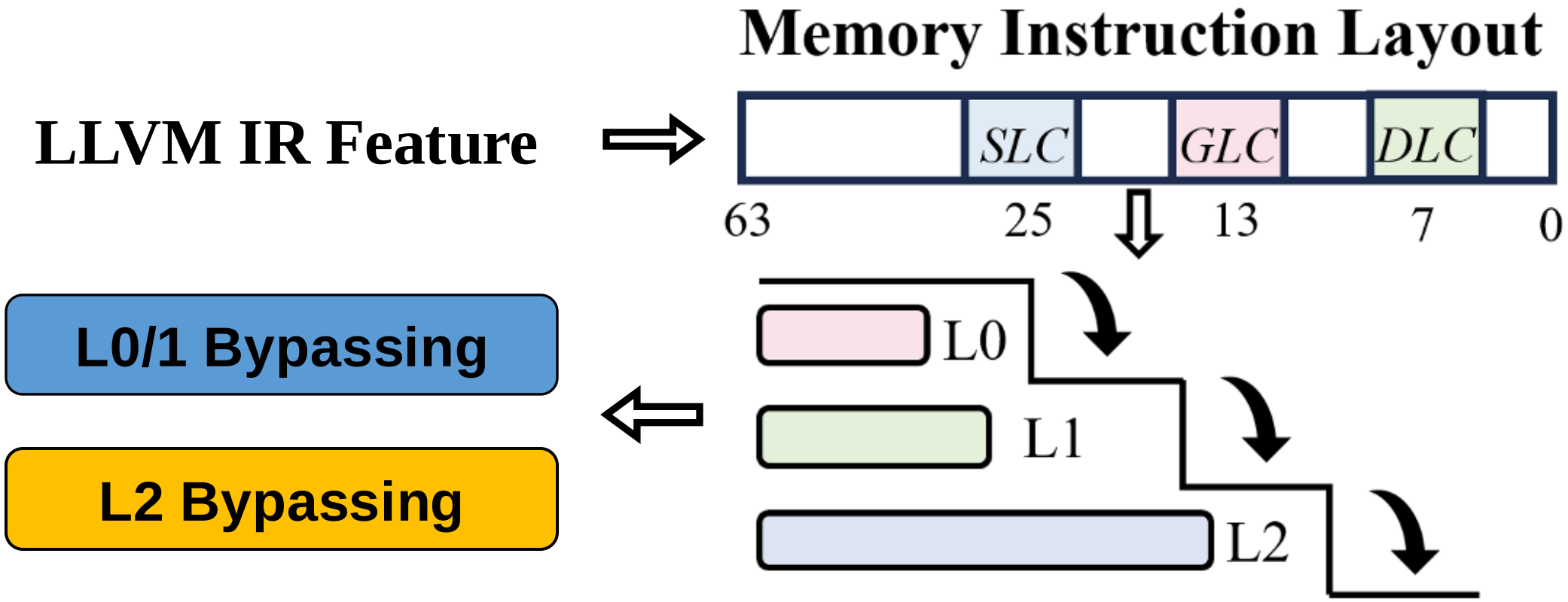
- Alleviate cache trashing and reduce cache conflicts
- Bypass unnecessary accesses and minimize cache pollution

Example: Benefits of Cache Bypass



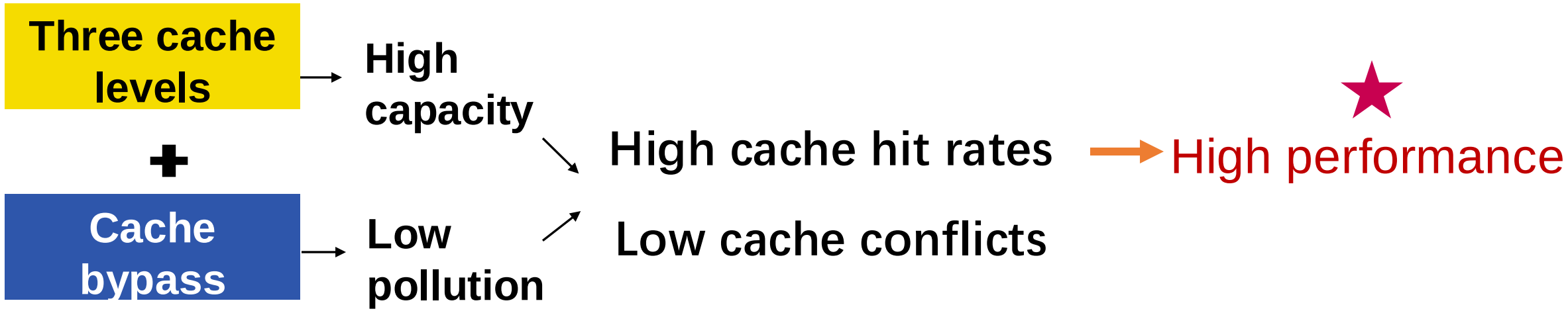
- SPMV: Bypassing **load7 and load9** at **L2 cache** yields **significant performance gains**

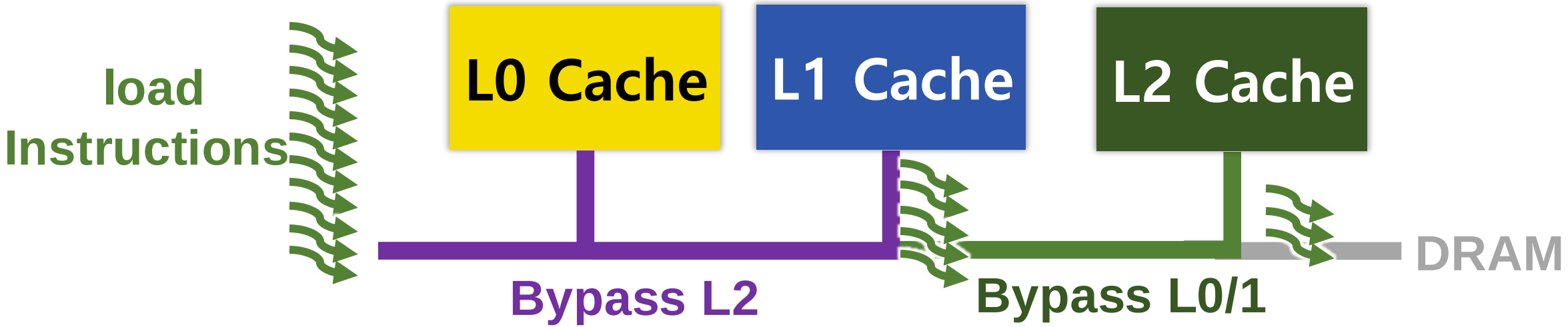
Software-based Cache Bypass



- AMD: **Instruction bits** and **LLVM features**
- Nvidia: **Instruction hints** and **L2 residency control**

GPUs with Cache Bypass





→ Identify the optimal bypass strategy

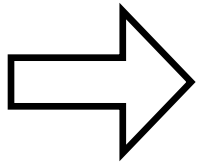
Mpache: Interaction-Aware Cache Bypass

Code

Clang
Fronted

LLVM Pass Injection

Mpache



Executable
File

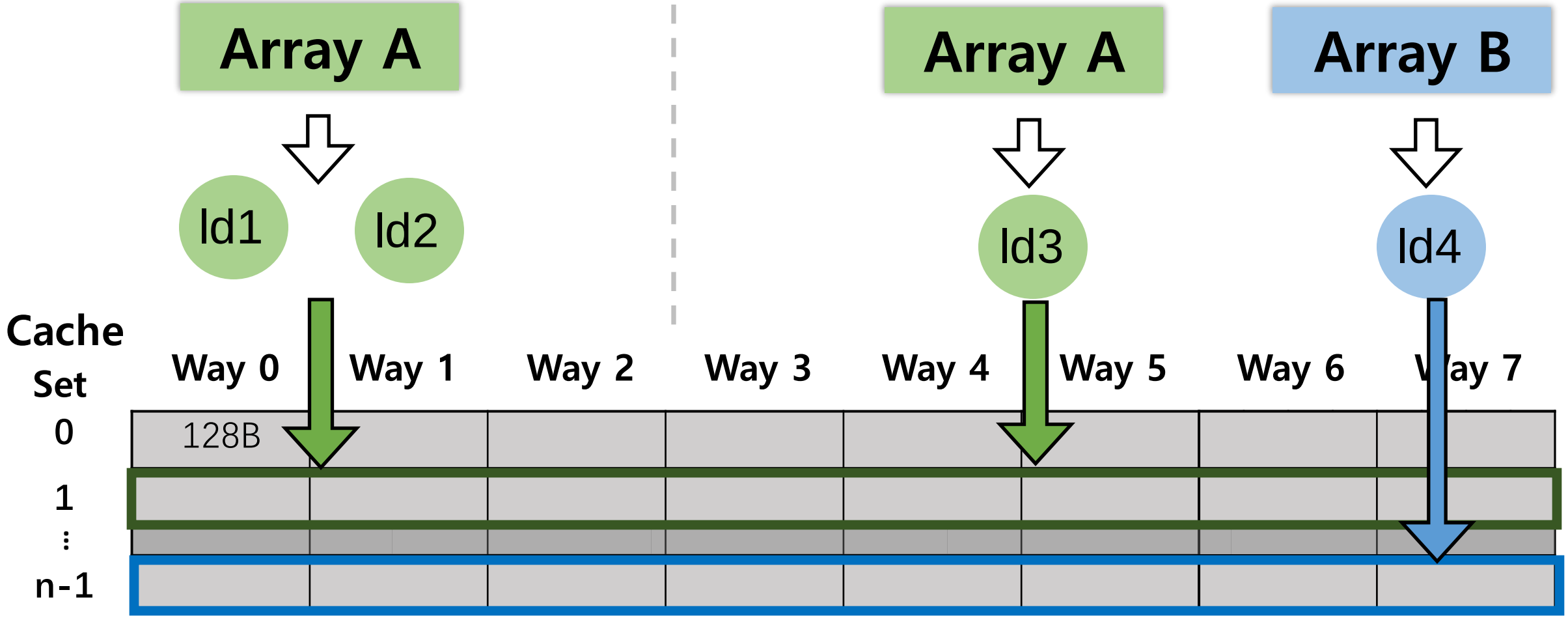
Assign one strategy to

each load IR/Instruction

- **Cache**
- **Bypass L0/1**
- **Bypass L2**

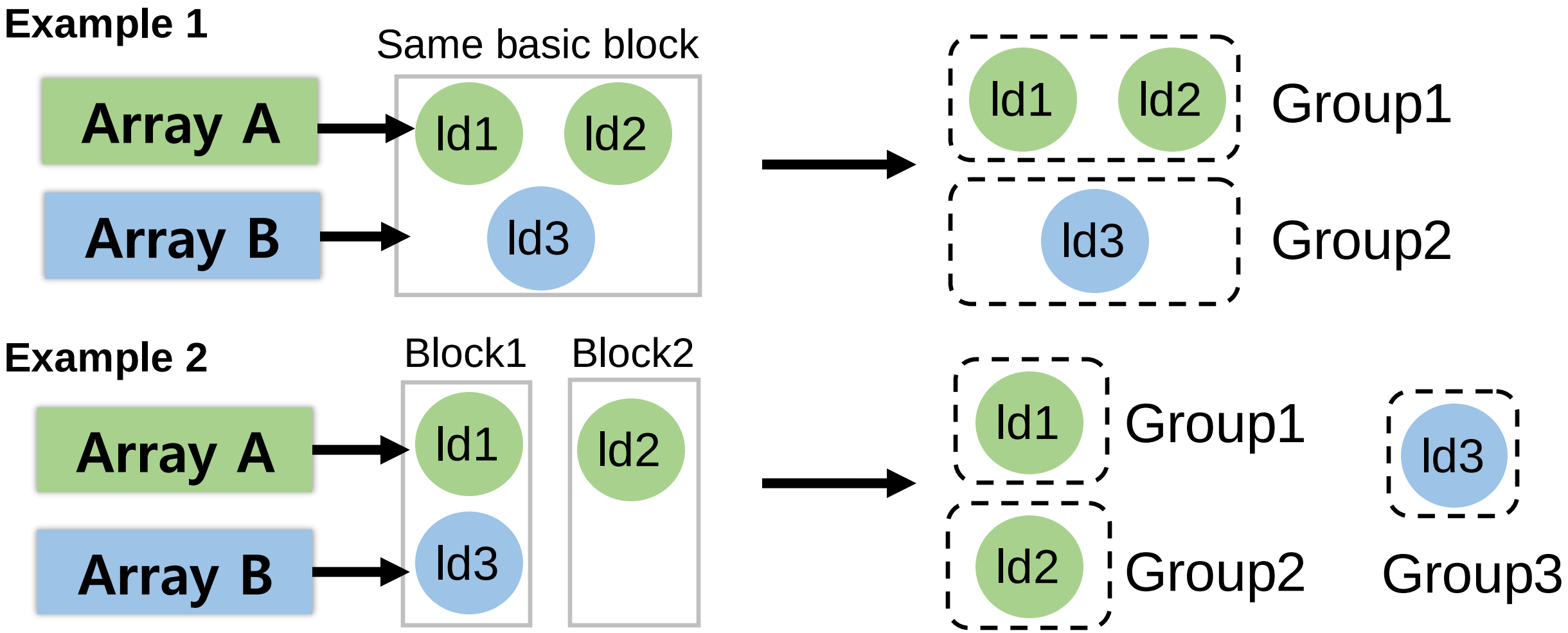
Interactions: Reuse and Contention

- Load from same array: **Reuse** and **Contention**
- Load from different arrays: Less **Contention**

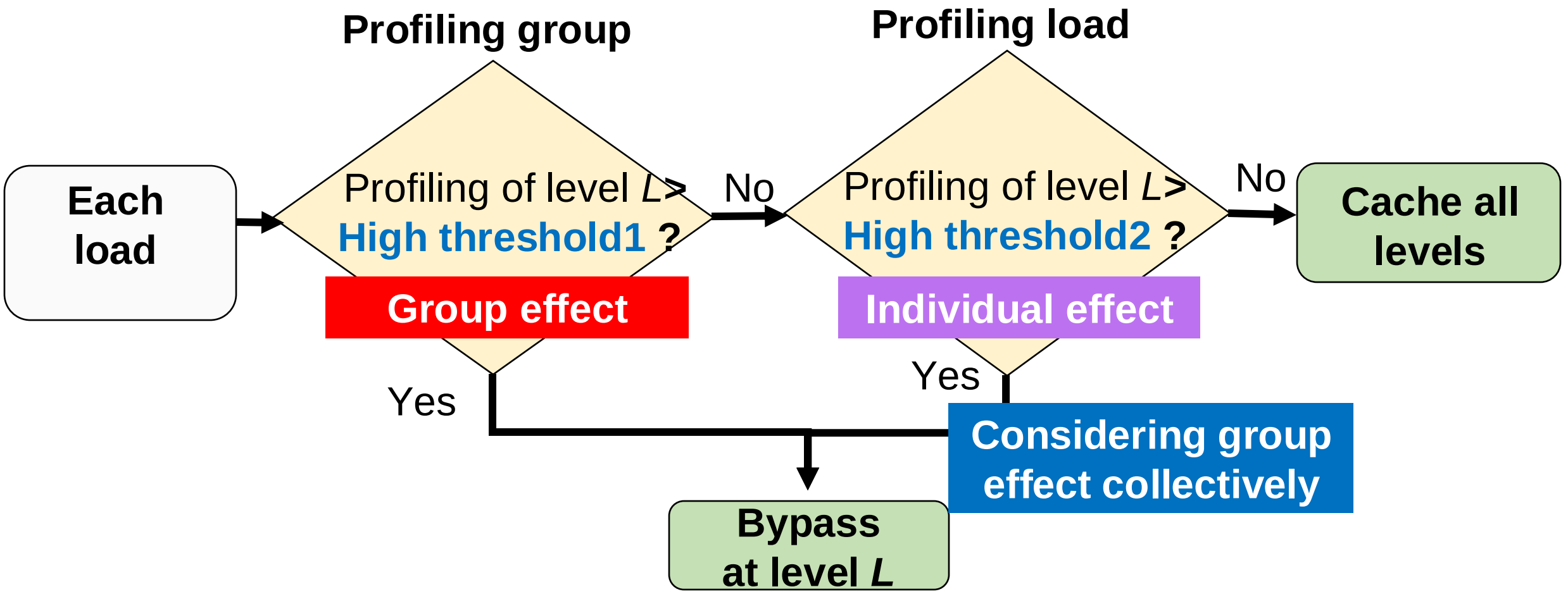


Load Group

- **Group loads** based on the **referenced arrays** & **basic blocks**



Load Bypass



- Bypass **a group** only when **advantageous**
- Bypass **a load** while considering both **group and individual effects**

Experiment Setup

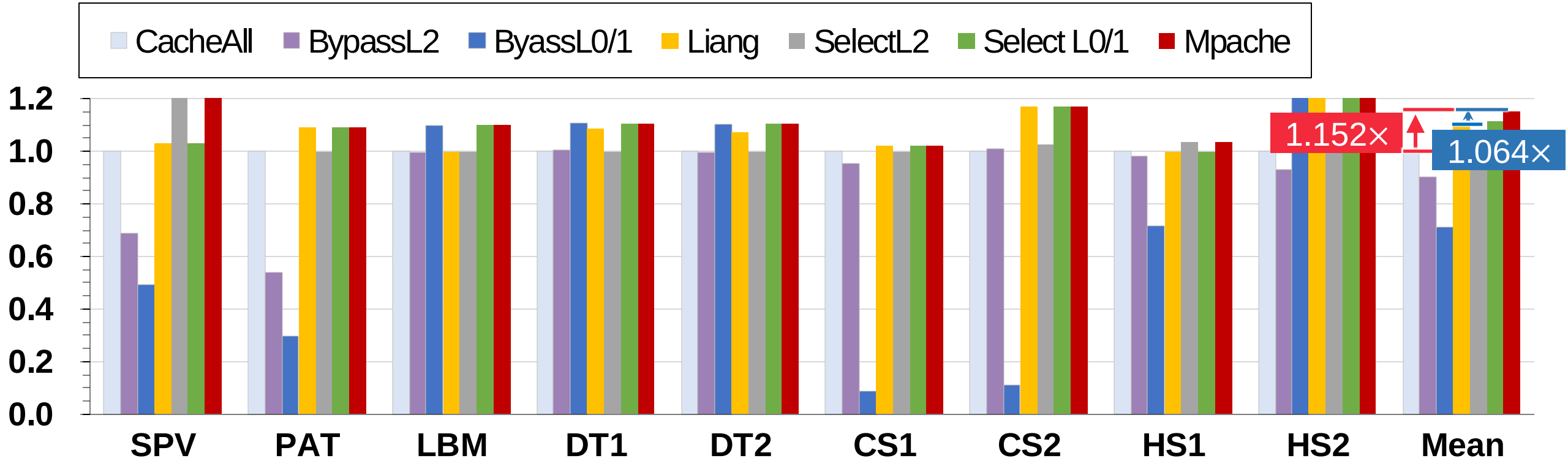
- Platforms
  - AMD Radeon RX 6900 XT
  - ROCm 5.5.0
  - LLVM 14.0.0
- Schemes
  - CacheAll
  - BypassL2 & ByassL0/1
  - SelectL2 & Select L0/1
  - Liang *[TCAD'18]*
  - Mpache

- Workload: 9 kernels from different domains from Rodinia, Parboil and CUDA Examples.

Kernels	*abbr.*	Kernels	*abbr.*
spmv	SPV	particlefilter	PAT
hybridsort-1	HS1	dct8x8_1	DT1
hybridsort-2	HS2	dct8x8_2	DT2
convolutionSeparable-1	CS1	lbm	LBM
convolutionSeparable-2	CS2		

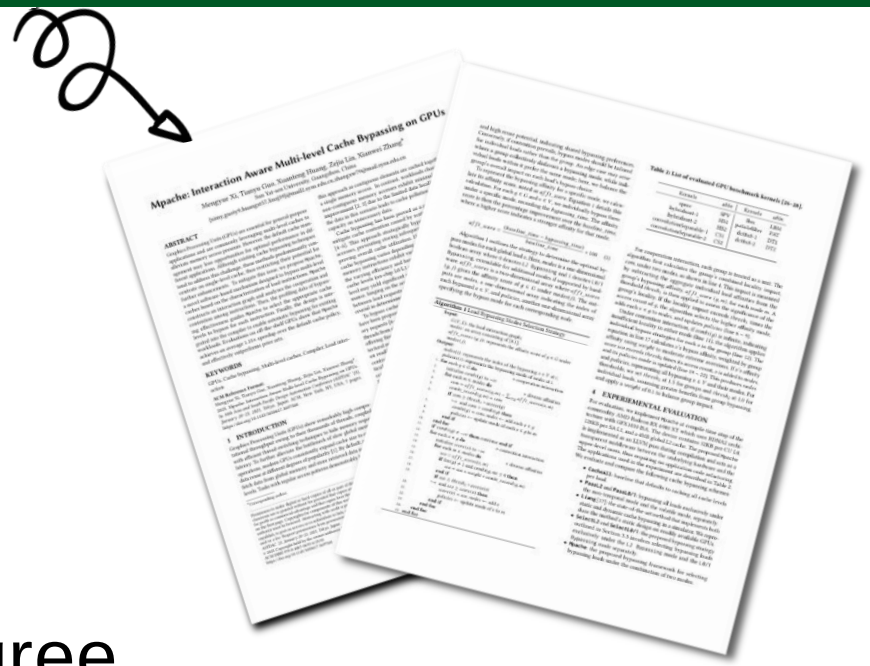
Performance Evaluation

- Average speedup **1.152x** compared to the default cache policy
- Outperforms Liang by **6%**



More in the Paper

- Load interaction analysis
- Detailed algorithm for load bypass
- Cache bypass control at the compiler level
- Sensitivity study
  - Tow thresholds for controlling bypass degree
  - Weight to balance balance group and individual effects
- Hardware cache hit rates
- Discussion with Nvidia GPU



Conclusion

- **Cache inefficiency** is observed in GPGPU applications
- **Cache bypass** can help alleviate this issue
- We propose **Mpache**, a **compiler-based cache bypass manager**
  - **Group loads** and analyze interactions
  - **Profile** loads and groups across different cache levels
  - Determine the bypass policy for each **group and individual load**
- Mpache outperforms the default cache policy and SOTA

中山大學
SUN YAT-SEN UNIVERSITY

Thank You!

Mengyue Xi, Tianyu Guo, Xuanteng Huang, Zejia Lin, Xianwei Zhang