ThreadFuser: A SIMT Analysis Framework for MIMD Programs

Ahmad Alawneh, Ni Kang, Mahmoud Khairy* , Timothy G. Rogers

PURDUE UNIVERSITY | Elmore Family School of Electrical and Computer Engineering

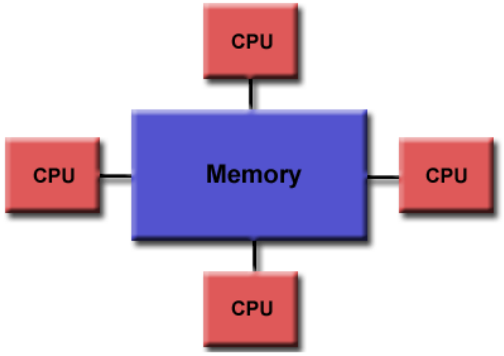
*Currently at AMD

MICRO 2024

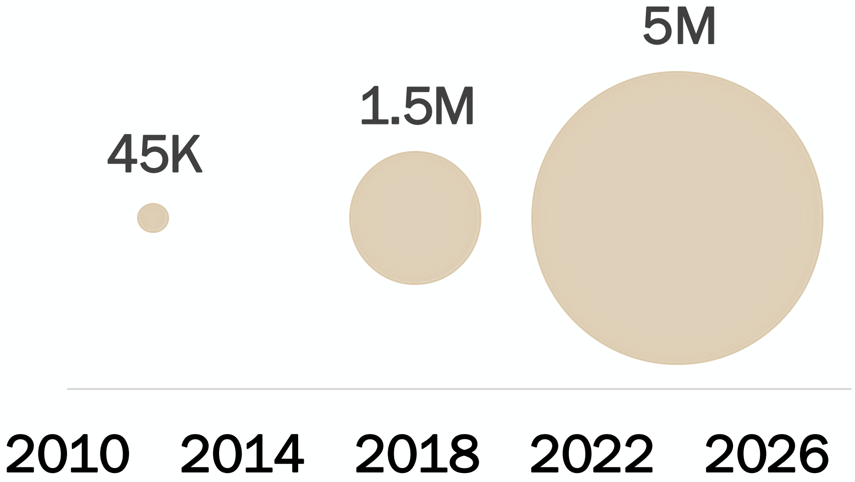
Expanding the Parallelism with SIMT

- Rising Demand for Parallelism
  - Traditional parallel programs use MIMD, primarily targeting CPUs
- Shifting Paradigms
  - The Slowing of Moore’s Law
  - CPUs face energy-efficiency limitations
- Emergence of Accelerators:
  - GPUs with SIMT are leading this trend

OpenMP



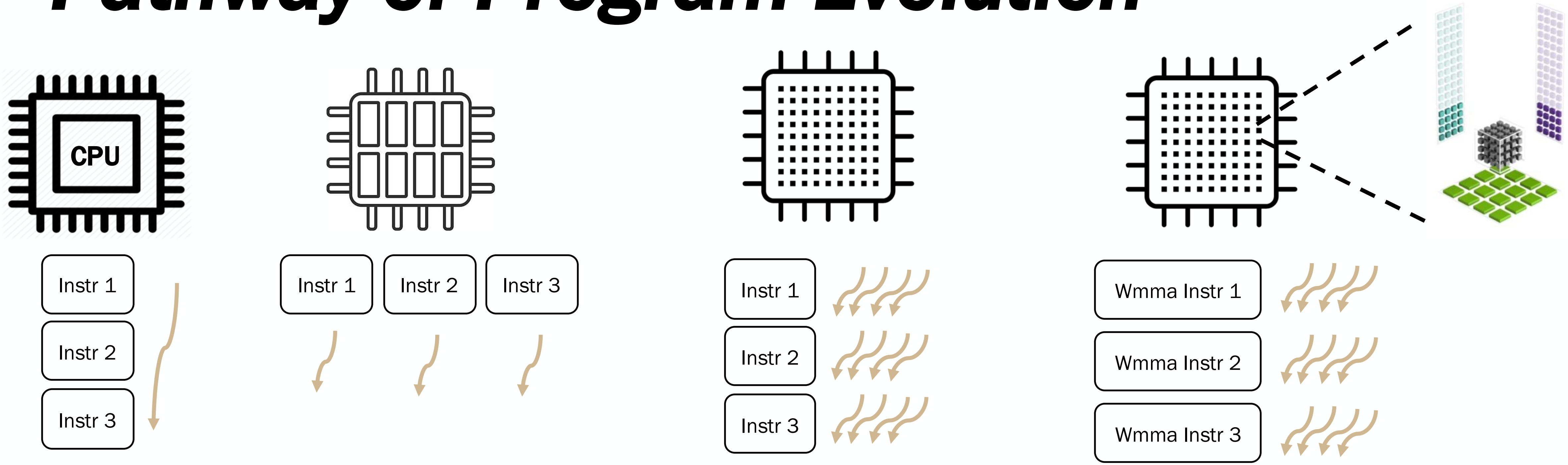
100X GPU DEVELOPERS



2024 NVIDIA Corporation Annual Review

SIMT: Single Instruction Multiple Threads

Pathway of Program Evolution

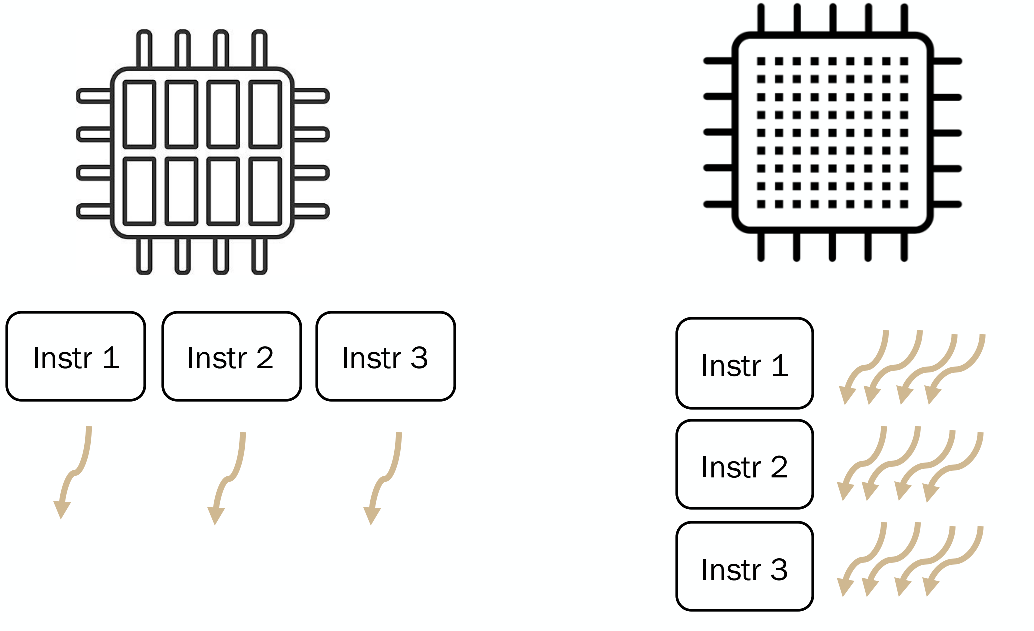


GEMM

What is Next?

Single Thread → *MIMD/ SIMD* → *SIMT* → *Specialized Processing Cores*

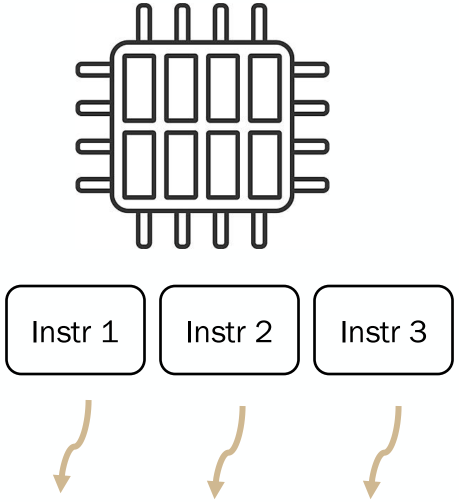
Challenges



MIMD/ SIMD

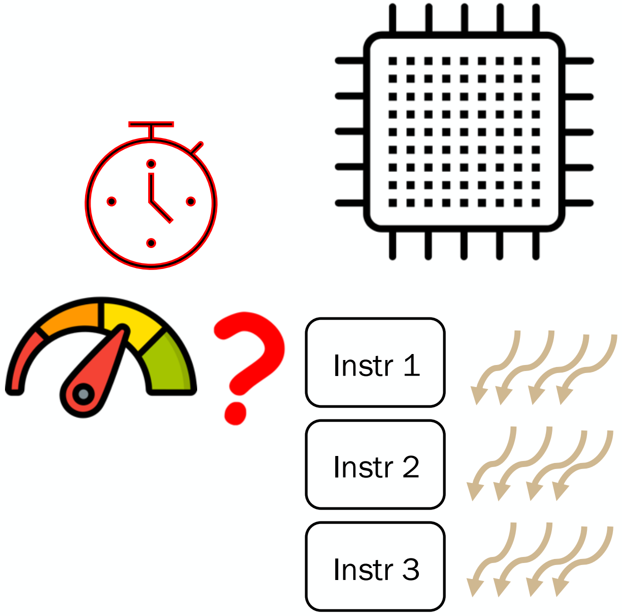
SIMT

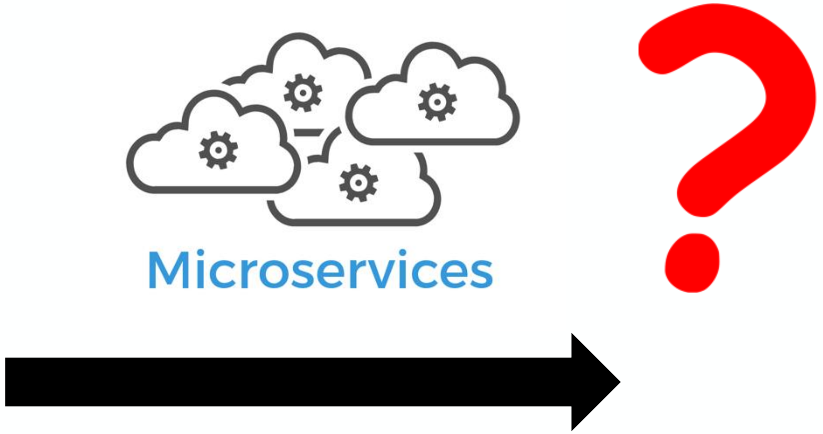
Challenges



- Porting effort is needed to exploit the SIMT accelerator
- Porting is risky
  - Time-consuming
  - Speedups are not guaranteed

- Hardware designers struggle to analyze the potential efficiency of SIMT hardware due to a lack of diverse software

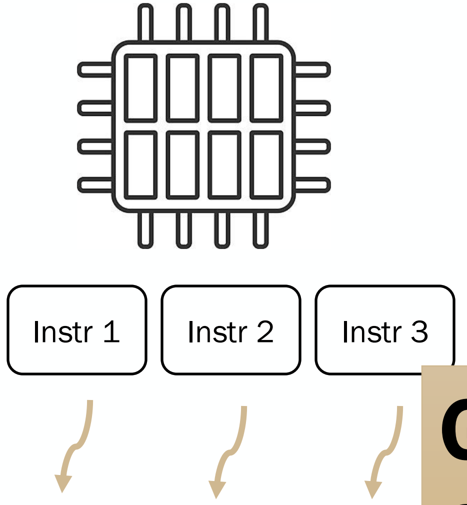




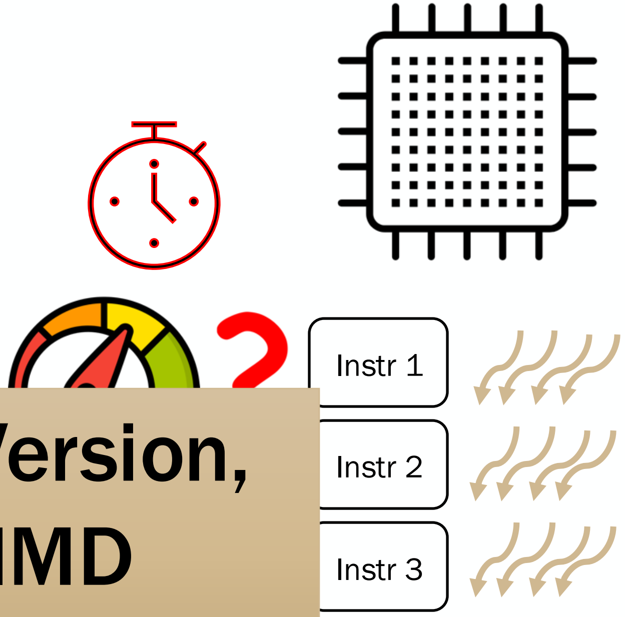
MIMD/ SIMD

SIMT

Challenges



- Porting effort is needed to exploit the SIMT accelerator
- Porting is risky
  - Time-consuming
  - Speedups are not guaranteed

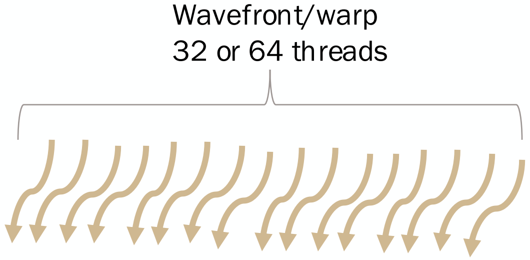


Goal: If we already have a parallel CPU Version, Can we analyze the performance of a MIMD application on SIMT hardware without porting?

MIMD/ SIMD

SIMT

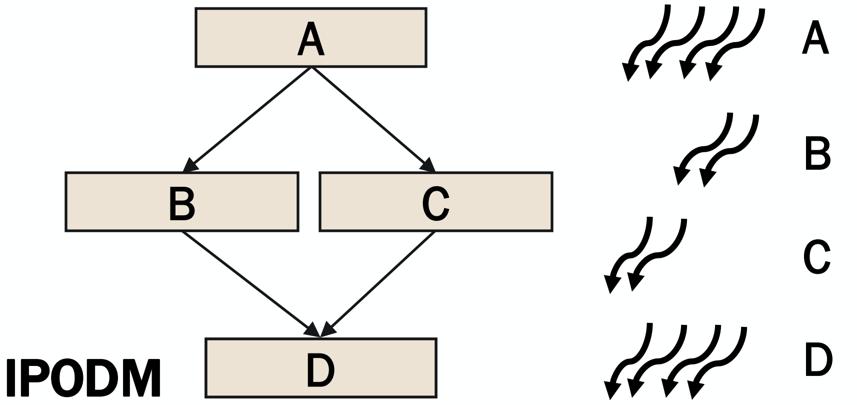
Single Instruction Multiple Thread (SIMT)



- Execution Model
    - Based on SPMD running on SIMD hardware
    - Threads grouped into warps/wavefronts
- CUDA, ROCm, and OpenCL are popular frameworks
- Efficiency Gains
    - **Pipeline:** Fetch, decode, and schedule once per warp
    - **Memory:** Coalesced accesses reduce traffic

Control-Flow and Memory Divergence in SIMT

Control Flow Divergence



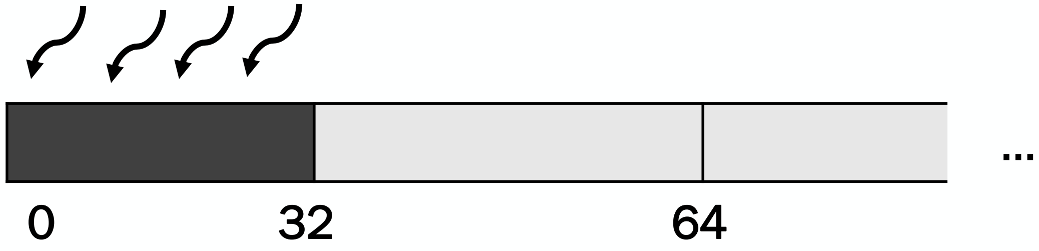
SIMT Stack

Rec point	Next BBL	Active Mask
-	D	1111
D	C	1100
D	B	0011

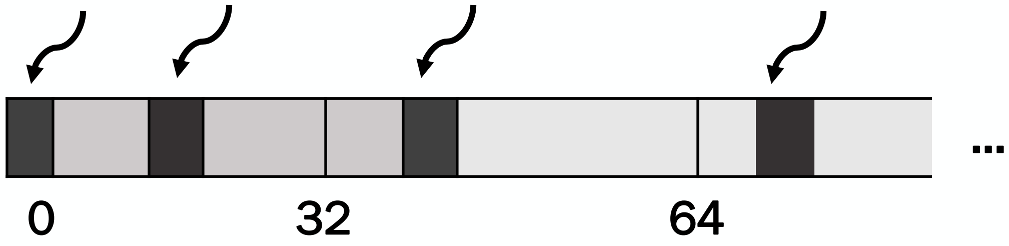
TOS ←

SIMT Efficiency

Memory Divergence

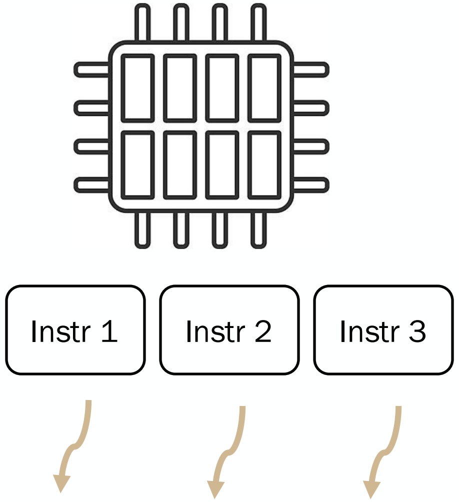


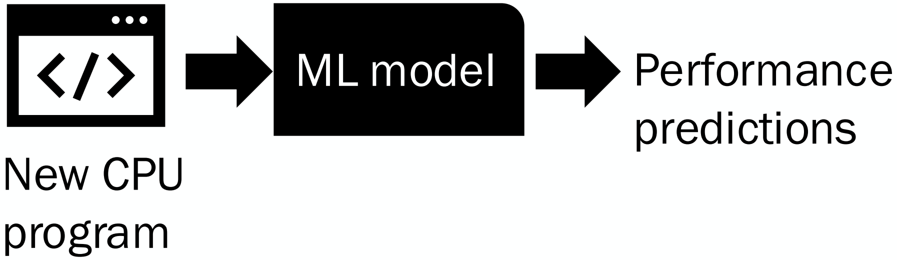
1 Memory Transaction

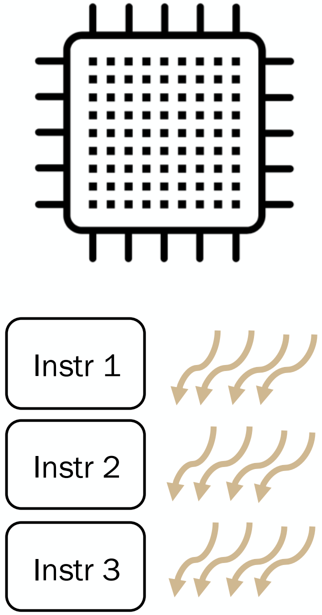


3 Memory Transactions

Previous Work







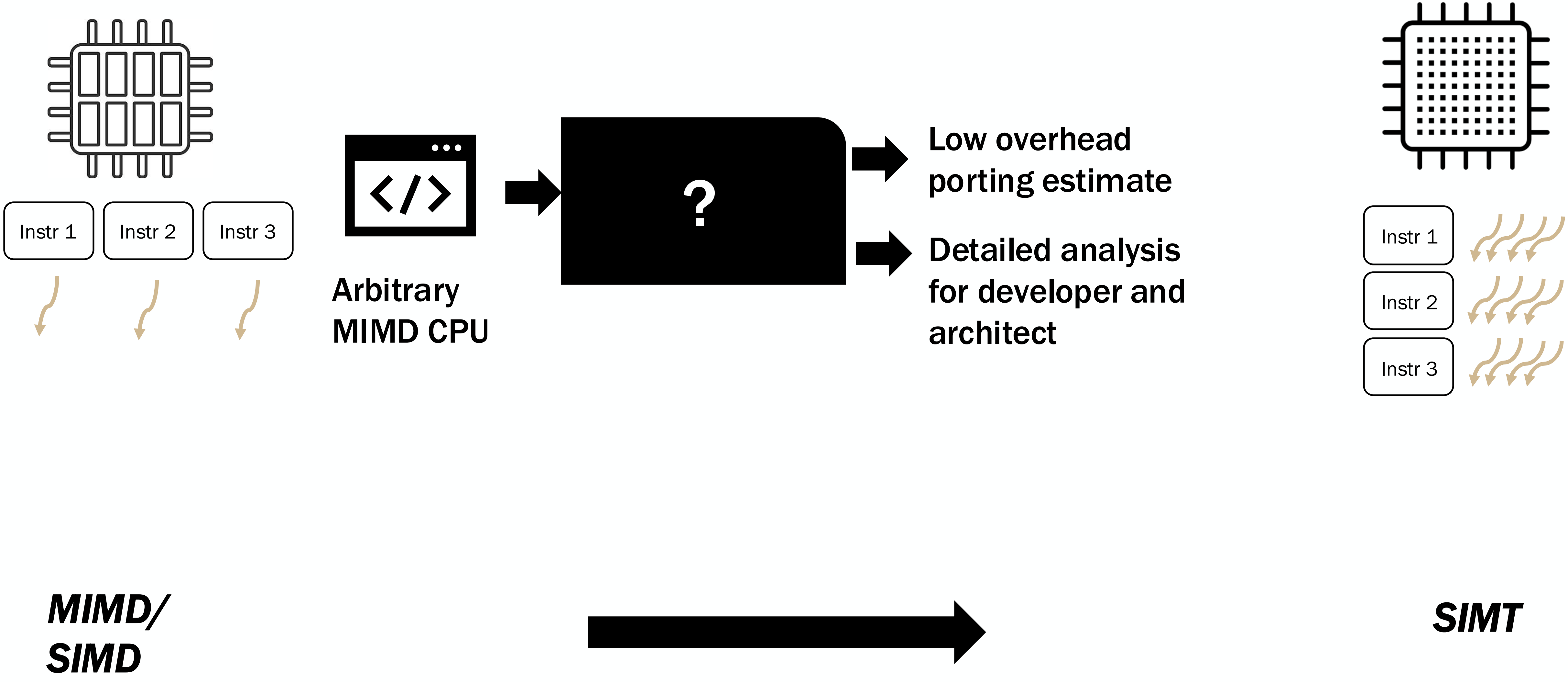
- **Prior Approaches:** Depend on machine learning models to predict execution time *

- **Limitations:**
  - Single Metric Focus
  - Small Codebase
  - No Architecture Exploration

* *1-Predicting Cross-Architecture Performance of Parallel Programs* (IPDPS 2024)
2- *Cross-architecture performance prediction (XAPP)* (MICRO 2015)

MIMD/ SIMD → ***SIMT***

Previous Work



Arbitrary MIMD CPU

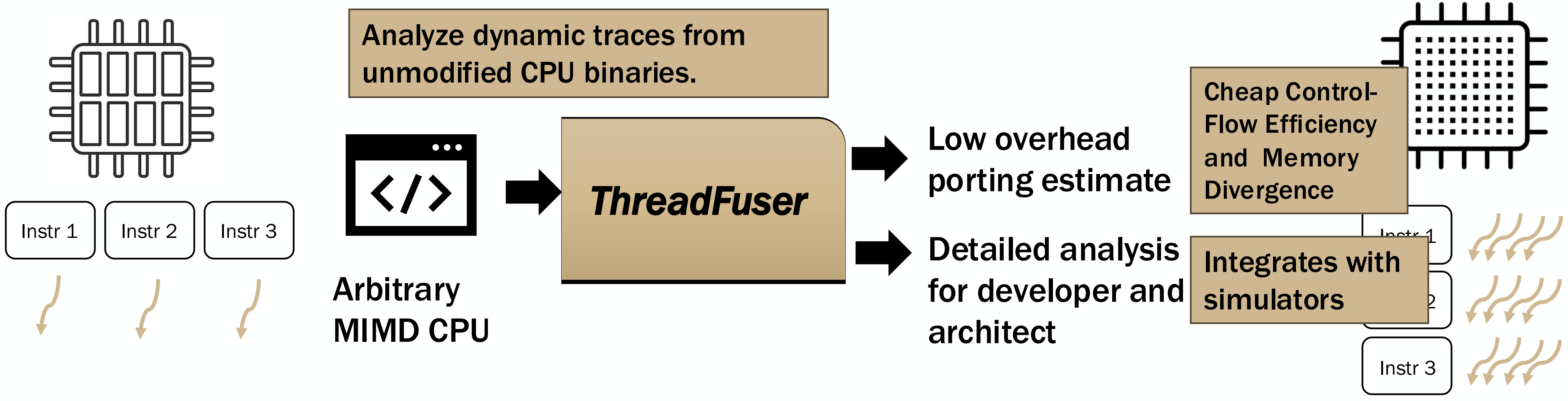
Low overhead porting estimate

Detailed analysis for developer and architect

MIMD/ SIMD

SIMT

Previous Work



Analyze dynamic traces from unmodified CPU binaries.

Arbitrary MIMD CPU

ThreadFuser

Low overhead porting estimate

Detailed analysis for developer and architect

Cheap Control-Flow Efficiency and Memory Divergence

Integrates with simulators

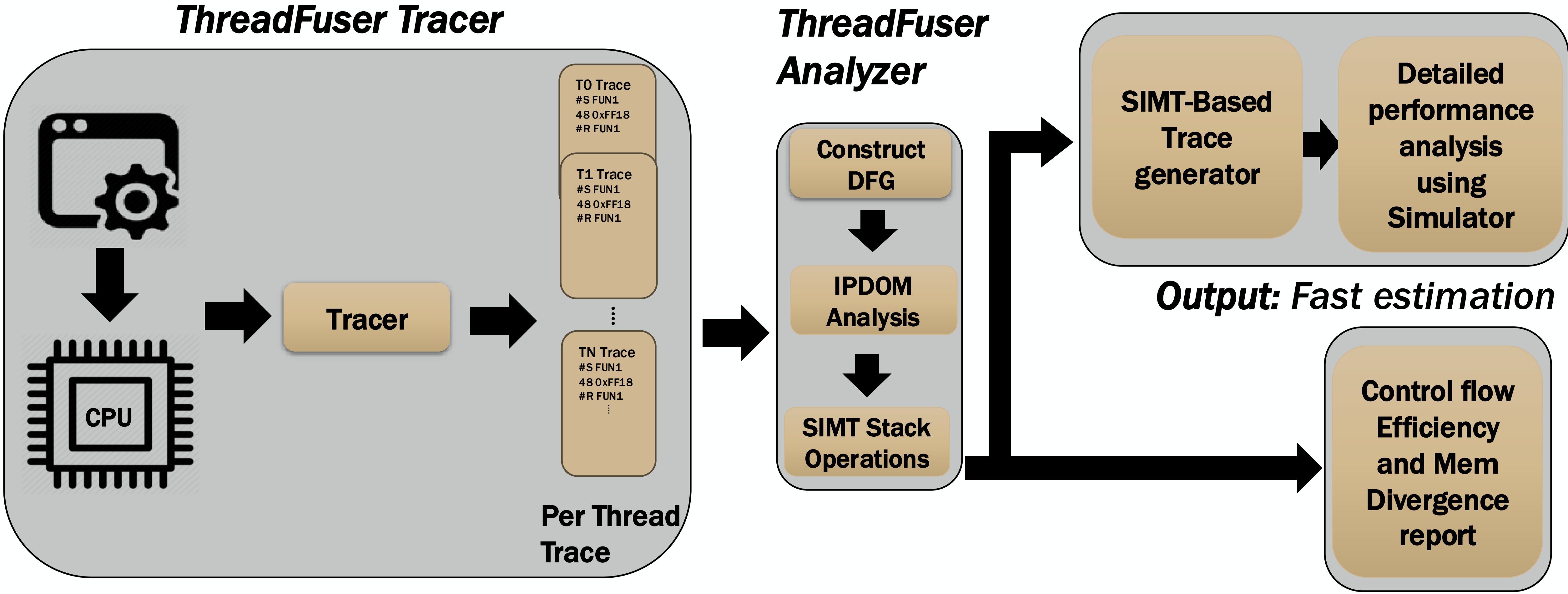
ThreadFuser: is an analysis framework that enables performance analysis of any MIMD CPU programs on SIMT hardware

MIMD/ SIMD

SIMT

SYSTEM OVERVIEW

Output: *Cycle-level performanc analysis using trace-driven simulator*



ThreadFuser Tracer

CPU

Tracer

T0 Trace
#S FUN1
48 0xFF18
#R FUN1

T1 Trace
#S FUN1
48 0xFF18
#R FUN1

TN Trace
#S FUN1
48 0xFF18
#R FUN1

Per Thread Trace

ThreadFuser Analyzer

Construct DFG

IPDOM Analysis

SIMT Stack Operations

SIMT-Based Trace generator

Detailed performance analysis using Simulator

Output: *Fast estimation*

Control flow Efficiency and Mem Divergence report

ThreadFuser Tracer and Analyzer

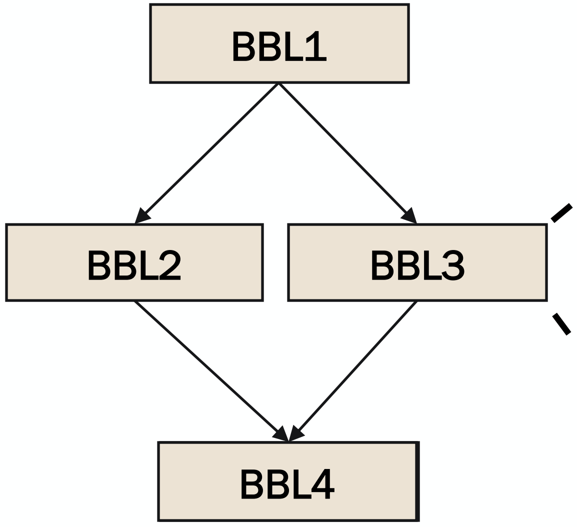
- Traces Outputs:
  - Per-thread Instructions trace
  - Memory access records
- Analyzer Components:
  - Per-function Control Flow Graph (CFG)
  - Perform IPDOM analysis
  - SIMT Stack Operations
  - Generates SIMT-based traces

T0 Trace

#S FUN1
BBL1
BBL2
BBL4
#R FUN1

T1 Trace

#S FUN1
BBL1
BB3
BBL4
#R FUN1



CFG

SIMT Stack Status

Rec point	Next BBL	# inst.	Active Mask	Fun Stack
-	BBL4	2	11	Fun1
BBL4	BBL3	8	01	Fun1
BBL4	BBL2	4	10	Fun1

Experimental Setup

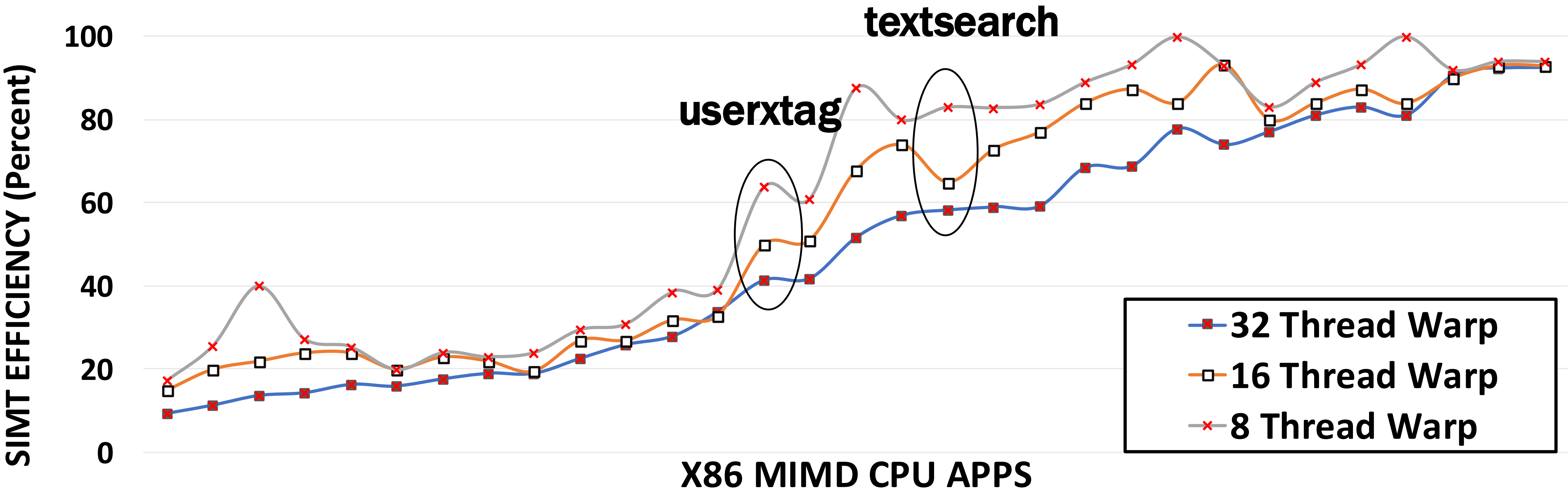
- 36 workloads across task-parallel and data-parallel architectures

Benchmark suite	Parallelism
Rodinia	data-parallel
Parapoly	data-parallel
µsuite	task-parallel
DeathStarBench	task-parallel
ParSec 3.0	Task and data parallel

- Intel’s Pin tool to build tracer
- Correlated SIMT efficiency and memory divergence against Nvidia H100
  - 11 applications with identical CPU and GPU implementations
- Accel-Sim for performance simulation (x86 support added)
  - In task parallel microservices application, we batch requests to the same microservice and run them on SIMT

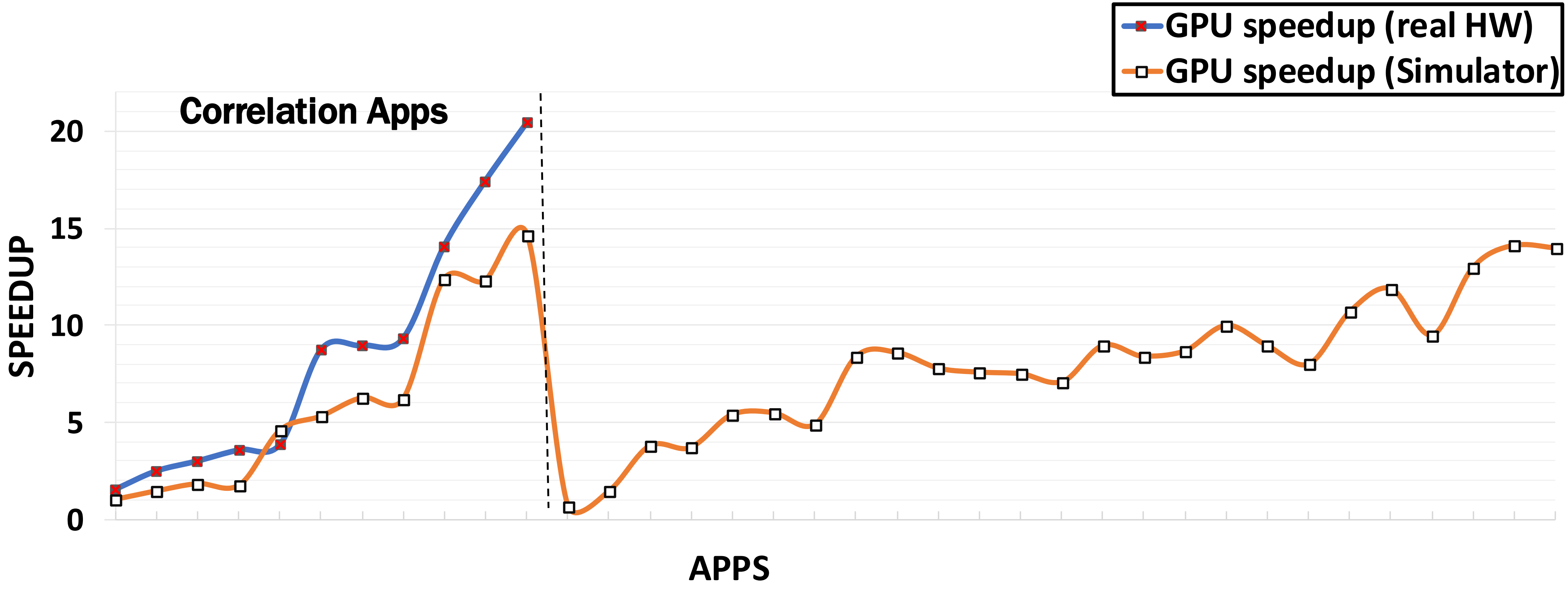
Use Case: Quick Porting Estimation

- **For Developers:**
  - Provides SIMT efficiency projections, helping to assess the potential effort required for porting
- **For Architects:**
  - Provides insights for designing future SIMT architectures and accelerators suited to diverse applications



Use Case: Detailed Performance Analysis

- Full cycle level analysis using generated SIMT-based traces and trace-driven simulation



Conclusion

Summary:

- A Framework for comprehensive performance analysis of any MIMD CPU binary on SIMT hardware

Impact:

- Enables performance evaluation on SIMT hardware without code porting
- Supports SIMT hardware design by offering efficiency insights across diverse applications

Questions

Backup

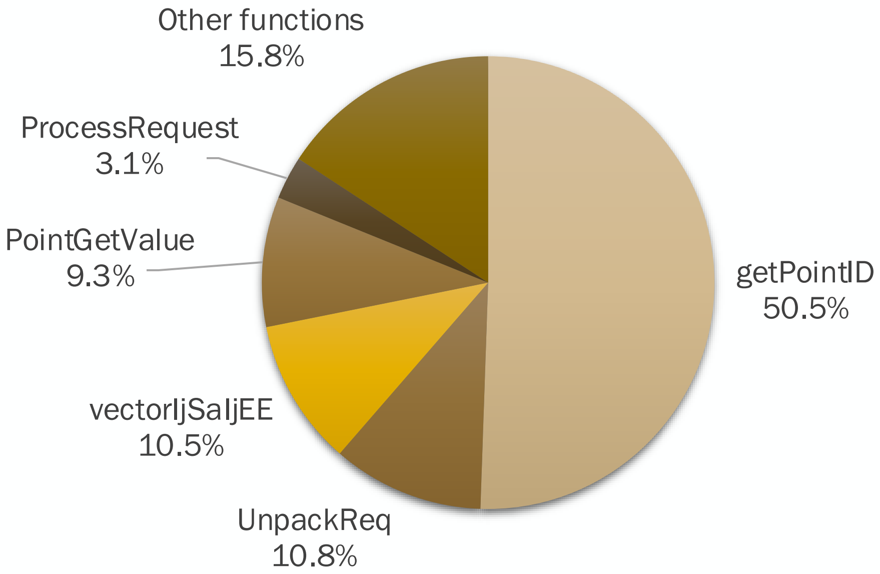
Use Case: Source Code Analysis

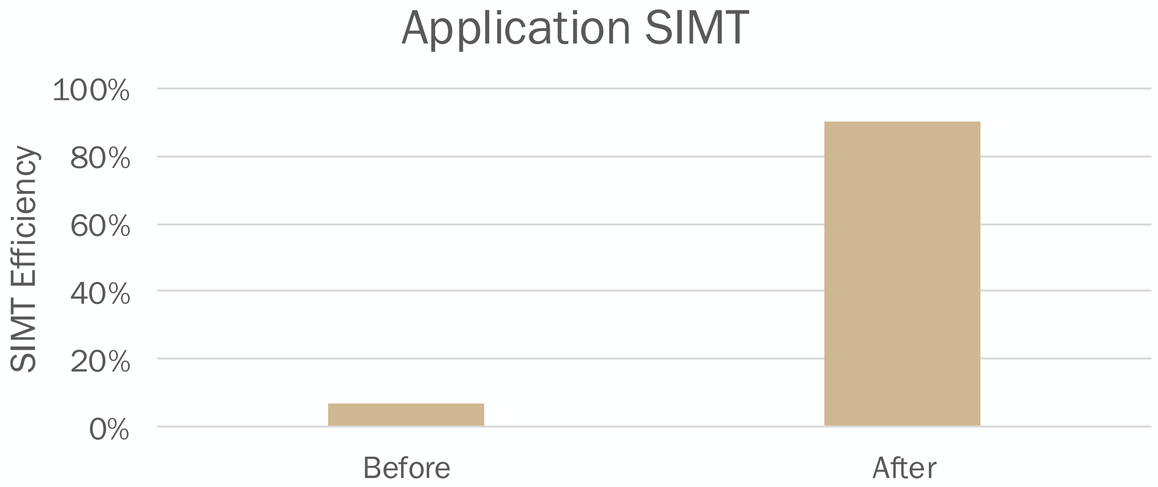
- ThreadFuser diagnoses low SIMT efficiency in ported MIMD implementations when source code is available.
- Example: *HDSearch-Midtier.*

```
for (; table != table_end; ++table) {
        for (; xor_mask != xor_mask_end; ++xor_mask) {
                sub_key = key ^ (*xor_mask);
                ...
                for (int j = 0; j < 10; j++) {
                        point_id_vec->push_back(point);
                }
        }
}
```

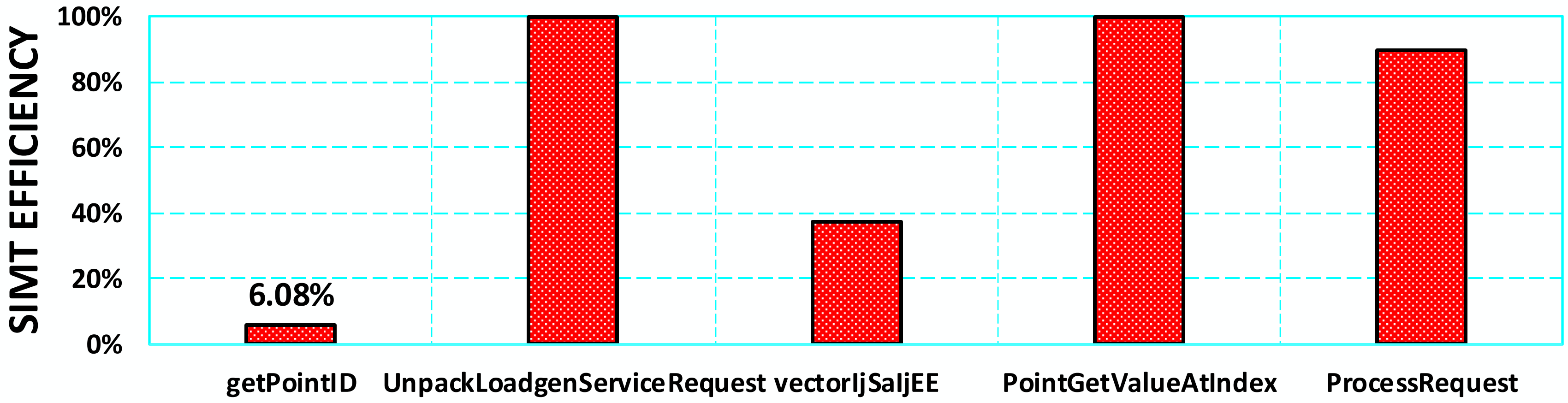
ThreadFuser reports 6% efficiency for getPointID function

Instructions Distribution per Function

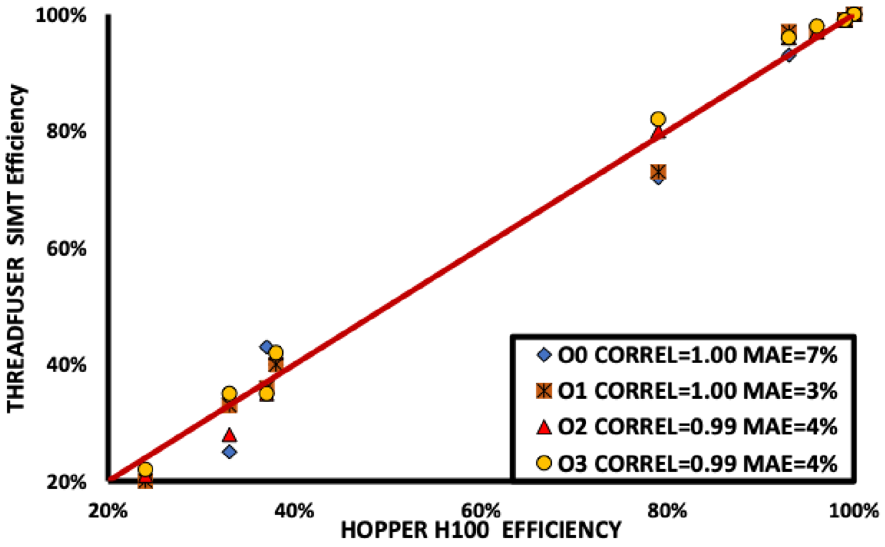


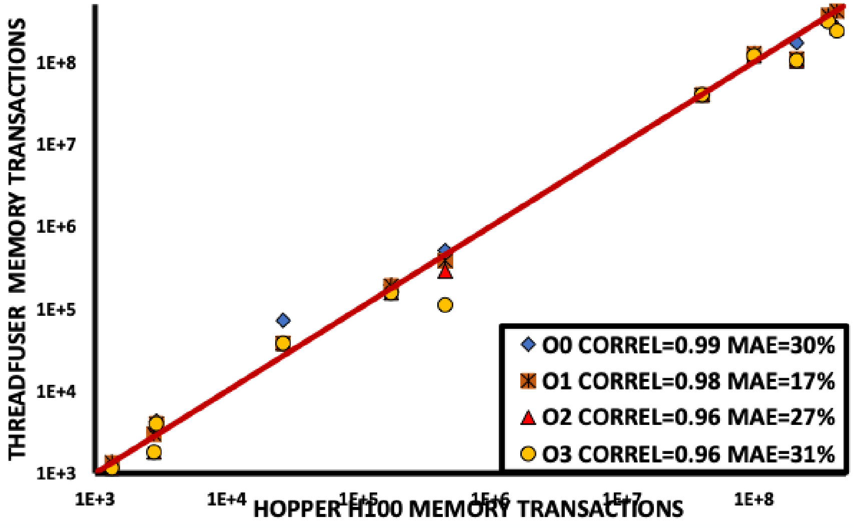


Use Case: Source Code Analysis

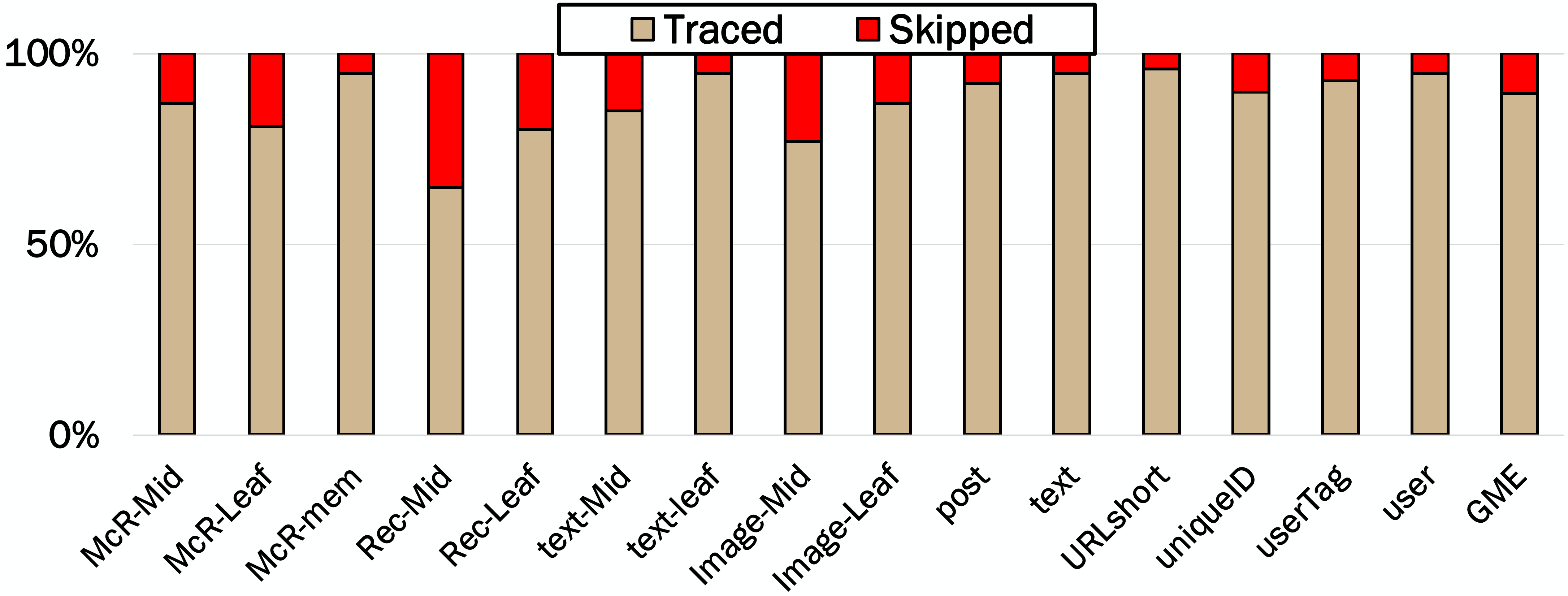


Correlation

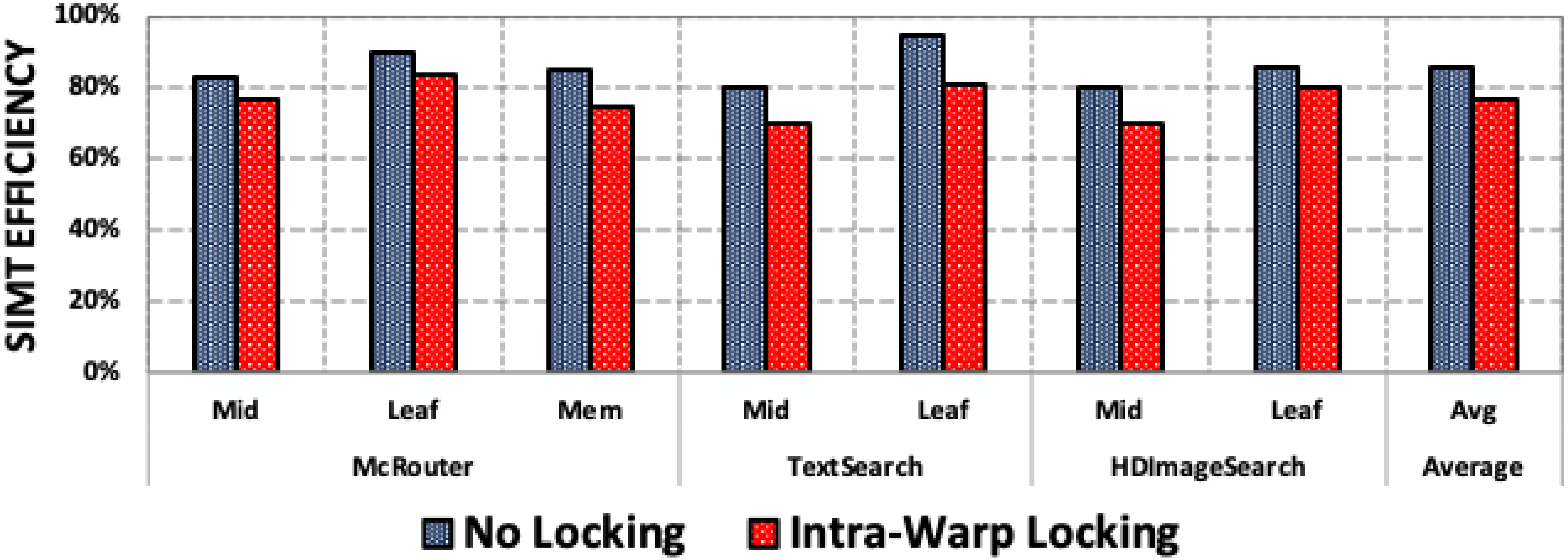




Synchronization study



Synchronization study



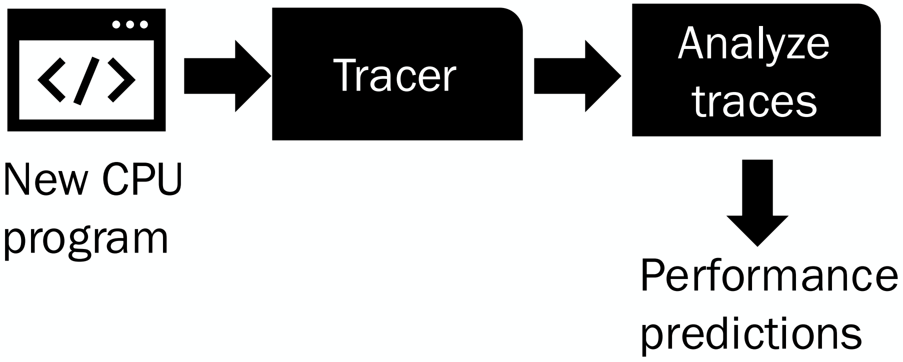
ThreadFuser: A SIMT Analysis Framework for MIMD Programs

- **ThreadFuser:** is an analysis framework that predicts the performance of MIMD CPU programs on SIMT hardware.

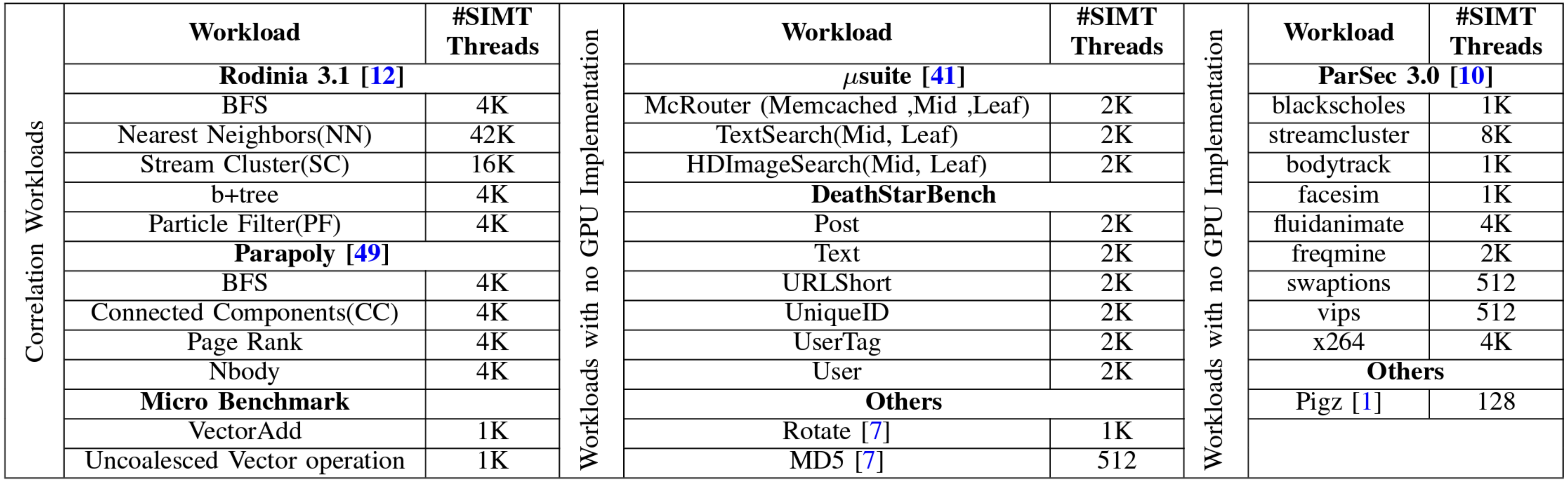
- Analyze dynamic traces from unmodified CPU binaries.

- Offers insights into control flow efficiency, memory divergence, and synchronization on SIMT hardware.

- **ThreadFuser** integrates for cycle-level analysis with simulators like Accel-Sim.



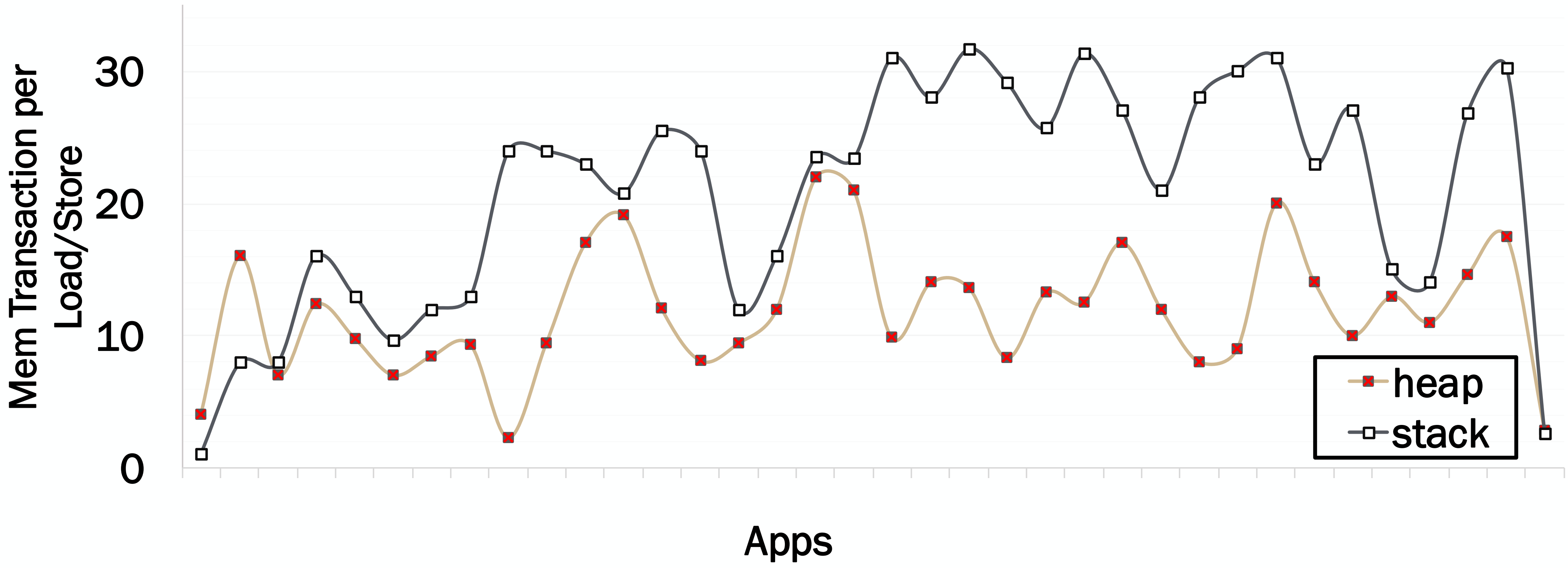
Workloads



	Workload	#SIMT Threads		Workload	#SIMT Threads		Workload	#SIMT Threads
Correlation Workloads	**Rodinia 3.1 [12]**		Workloads with no GPU Implementation	**μsuite [41]**		Workloads with no GPU Implementation	**ParSec 3.0 [10]**	
	BFS	4K		McRouter (Memcached ,Mid ,Leaf)	2K		blackscholes	1K
	Nearest Neighbors(NN)	42K		TextSearch(Mid, Leaf)	2K		streamcluster	8K
	Stream Cluster(SC)	16K		HDImageSearch(Mid, Leaf)	2K		bodytrack	1K
	b+tree	4K		**DeathStarBench**			facesim	1K
	Particle Filter(PF)	4K		Post	2K		fluidanimate	4K
	Parapoly [49]			Text	2K		freqmine	2K
	BFS	4K		URLShort	2K		swaptions	512
	Connected Components(CC)	4K		UniqueID	2K		vips	512
	Page Rank	4K		UserTag	2K		x264	4K
	Nbody	4K		User	2K		**Others**	
	Micro Benchmark			**Others**			Pigz [1]	128
	VectorAdd	1K		Rotate [7]	1K			
	Uncoalesced Vector operation	1K		MD5 [7]	512			

TABLE I: Studied Workloads. #SIMT threads is the number of threads simulated by ThreadFuser

Memory Transactions


Mem Transaction per Load/Store
Apps
heap
stack