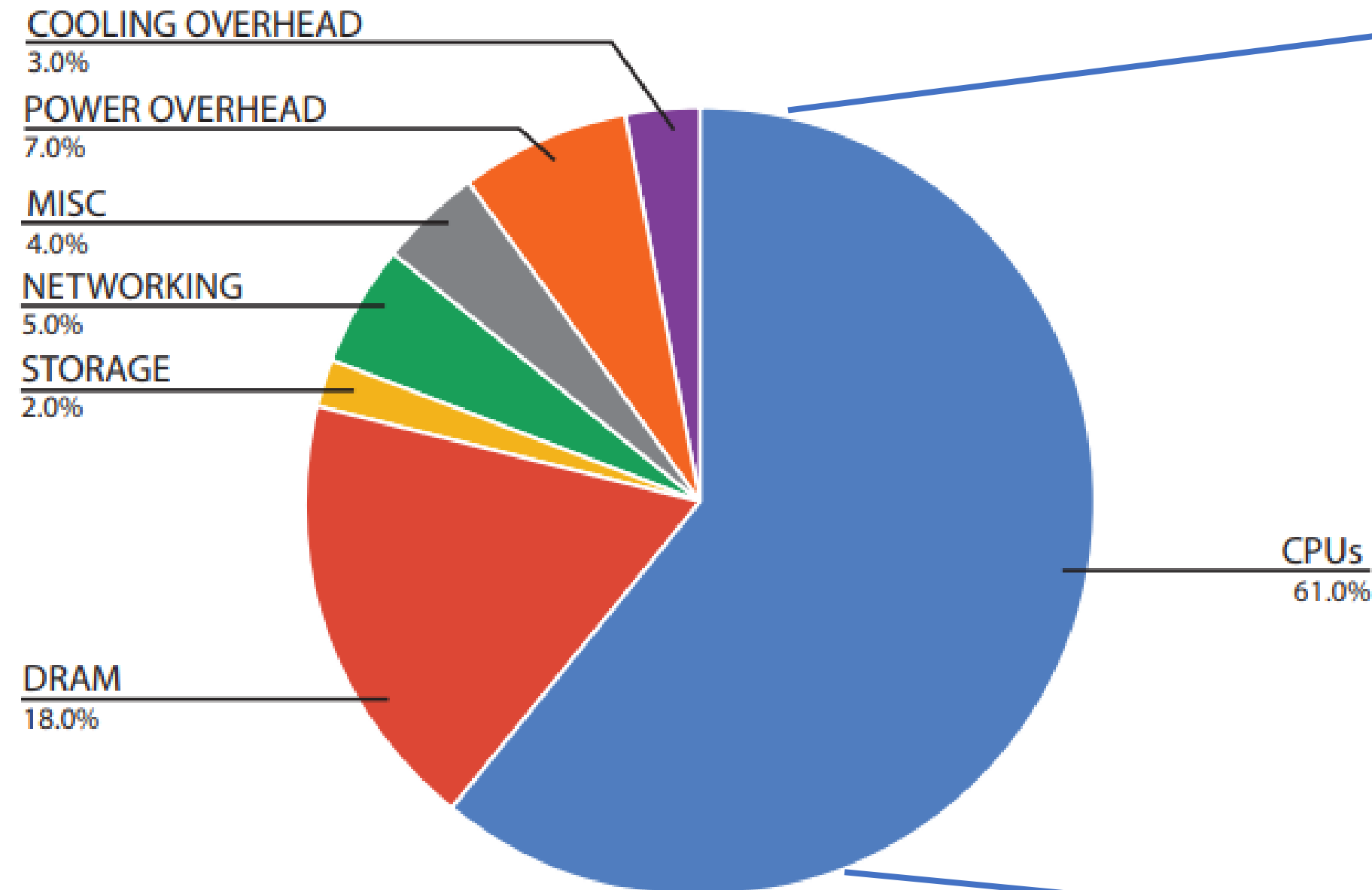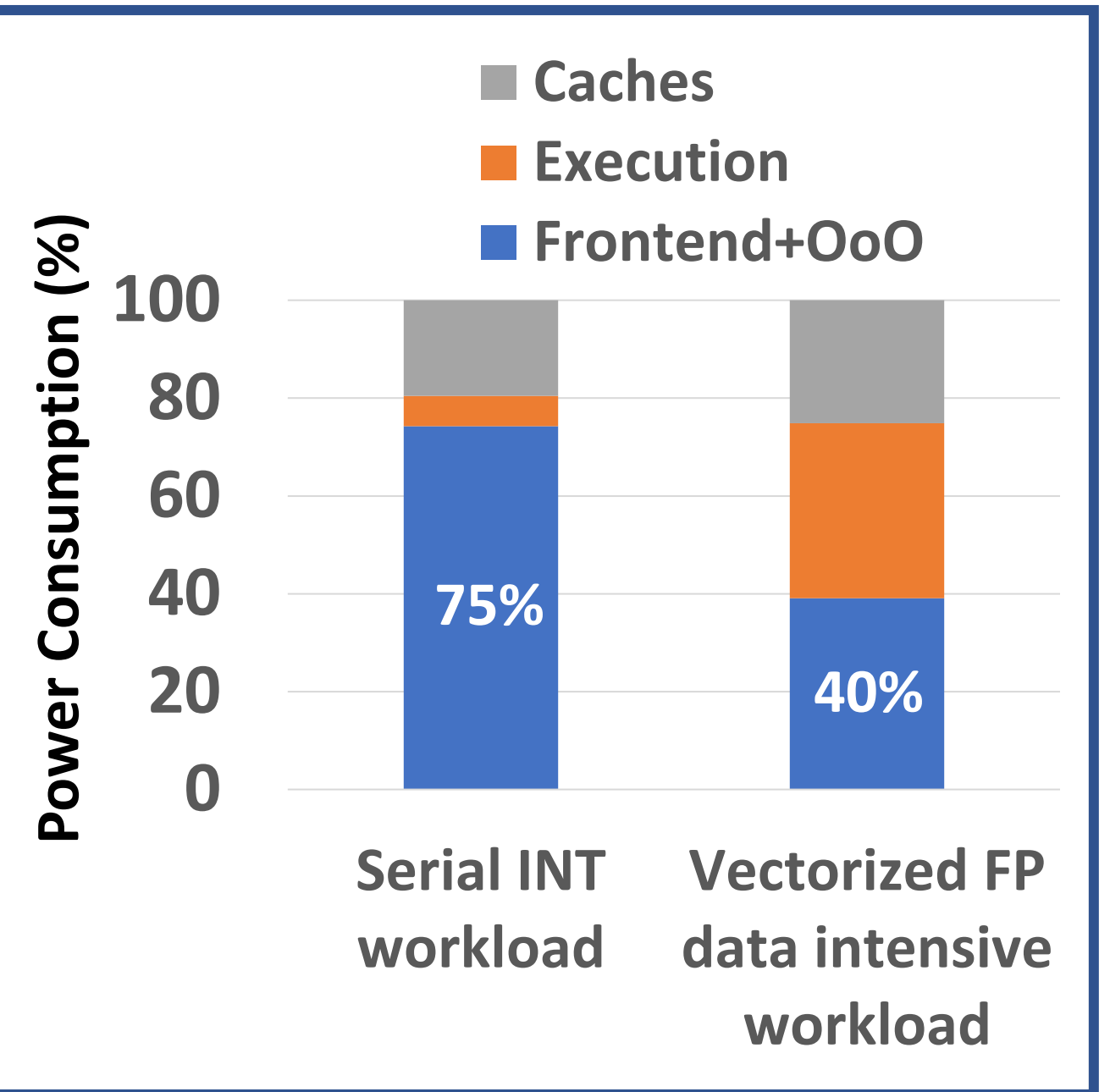# SIMR: Single Instruction Multiple Request Processing for Energy-Efficient Data Center Microservices

Mahmoud Khairy*, Ahmad Alawneh, Aaron Barnes, and Timothy G. Rogers

Purdue University

*Currently at AMD Research

10/3/2022

Datacenter Power Breakdown

Datacenter Power Breakdown (from Google)

CPU Power Breakdown

25-45% of datacenter power is consumed in CPU's instruction supply (frontend & OoO)

Barroso, Luiz André, and Urs Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines." *Synthesis lectures on computer architecture.* 2018
Haj-Yihia, Jawad, et al. "Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems." *ACM TACO* 2016
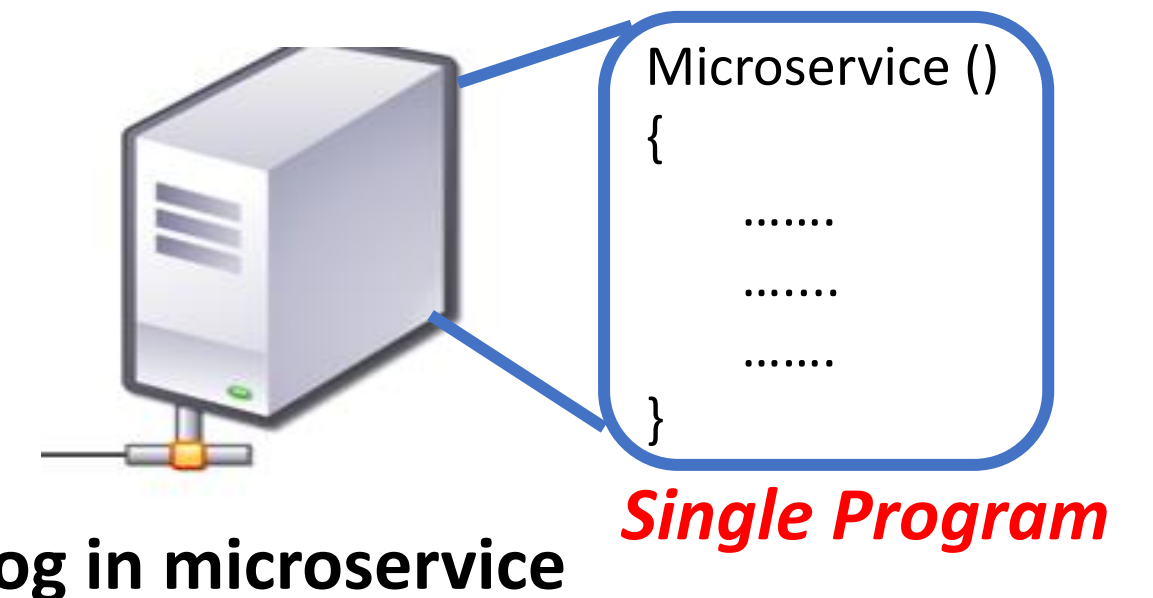*Caches power include dynamic L1/L2 cache power

1 Application, Million of Users

"Similar" Request-Level Parallelism
1000s of independent requests are all running the same code

Log-in reqs

("xyz", "1234")
("john", "5678")
("ma98", "4444")
("mah", "ko56")

Multiple Data

log in microservice

```
Microservice ()
{
    .......
    .......
    .......
}
```
Single Program

Private Datacenter

Public Datacenter

search reqs

("purdue univ")
("arsenal fc")
("elections 2024")
("stock today")

search microservice

Key Observation #1: Single Program Multiple Data (SPMD) are abundant in the datacenters

Server Workloads on GPU's

- **Key Idea:** Exploit SPMD by batching requests and run them on GPU's Single Instruction Multiple Thread (SIMT) or CPU's SIMD

- **Advantage**: Significant energy efficiency (throughput/watts) vs multi-threaded CPU

- **Drawbacks**:
  - (1) Hindering programmability (C++/PHP vs CUDA/OpenCL)
  - (2) Limited system calls support
  - (3) High service latency (10-6000x)
    - GPUs tradeoff single threaded optimizations (OoO, speculative execution, etc.) in favor of excessive multithreading
    - In SIMD, relying on branch predicates & fine grain context

Rhythm: Harnessing Data Parallel Hardware for Server Workloads

Sandeep R Agrawal
Duke University
sandeep@cs.duke.edu

Valentin Pistol
Duke University
pistol@cs.duke.edu

Jun Pang
Duke University
pangjun@cs.duke.edu

John Tran
NVIDIA
johntran@nvidia.com

David Tarjan *
NVIDIA

Alvin R Lebeck
Duke University
alvy@cs.duke.edu

MemcachedGPU: Scaling-up Scale-out Key-value Stores

Tayler H. Hetherington
The University of British Columbia
taylerh@ece.ubc.ca

Mike O'Connor
NVIDIA & UT-Austin
moconnor@nvidia.com

Tor M. Aamodt
The University of British Columbia
aamodt@ece.ubc.ca

ispc: A SPMD Compiler for High-Performance CPU Programming

Matt Pharr
Intel Corporation
matt.pharr@intel.com

William R. Mark
Intel Corporation
william.r.mark@intel.com

Recall: GPUs and SIMDs were designed to execute data parallel portion (i.e., loops) not the entire application
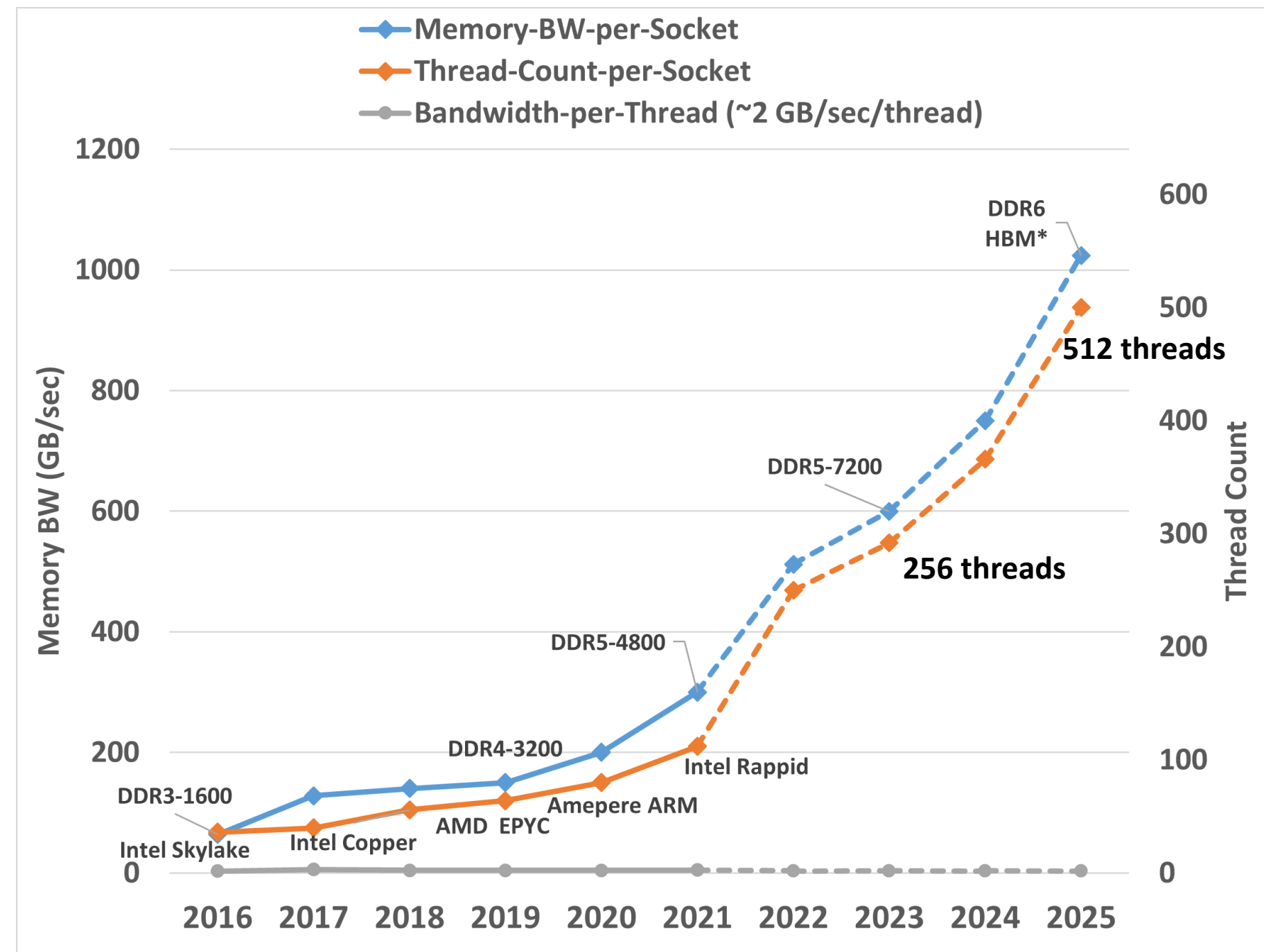
"Slower but energy-efficient wimpy cores only win for general data center workloads if their single-core speed is reasonably close to that of mid-range brawny cores"

Up to 2x slower latency should be tolerated by data center providers

Urs Hölzle
Google SVP

Barroso, Luiz André, and Urs Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines." *Synthesis lectures on computer architecture.* 2018
Hölzle, Urs. "Brawny cores still beat wimpy cores, most of the time." IEEE MICRO 2010

Off-Chip BW Scaling

Key Observation #2: There is available headroom to increase on-chip throughput (thread count) in the foreseeable future.
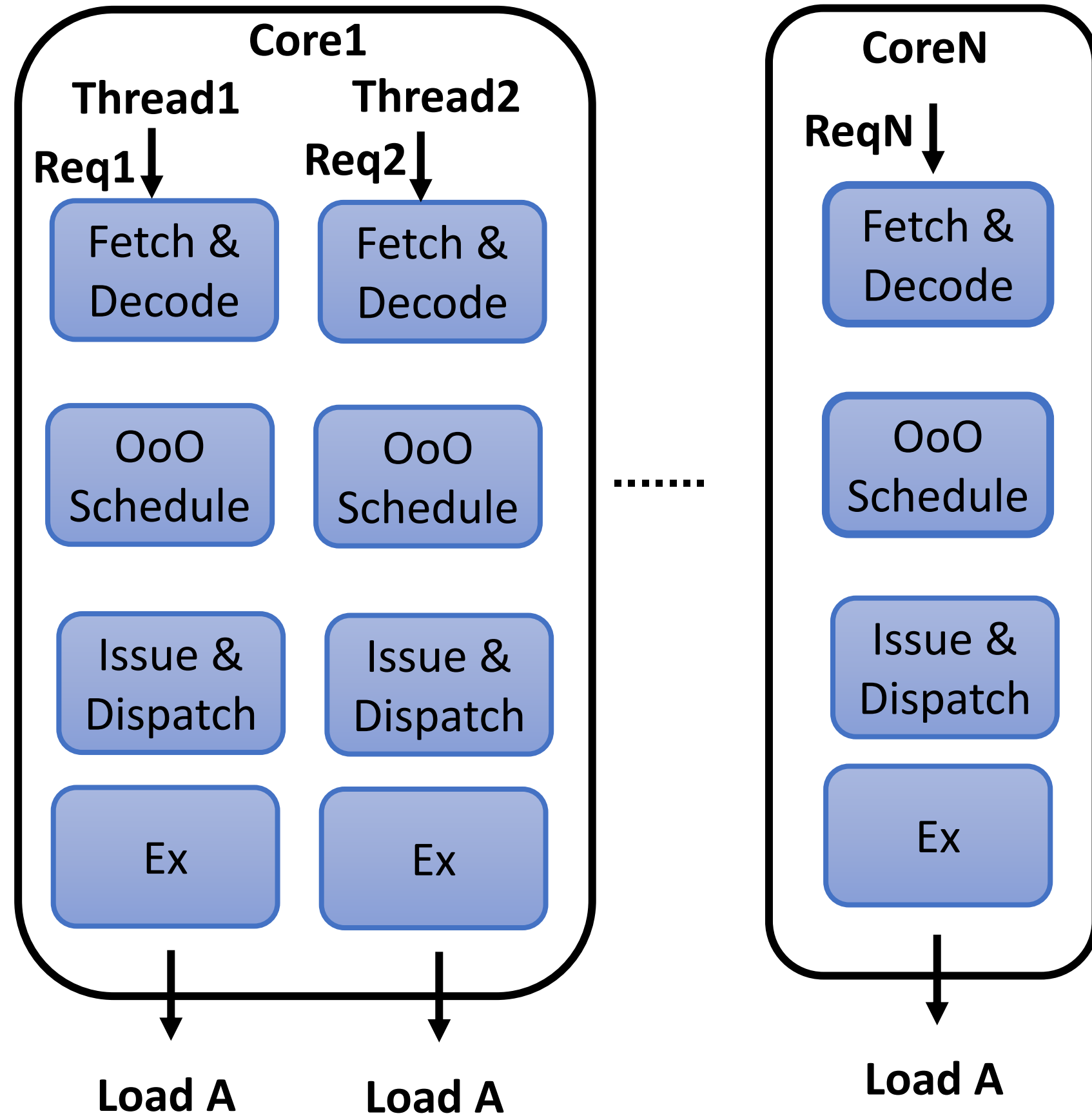
How to increase on-chip throughput of CPU?

- Direction#1 (industry standard): Add more Chiplets + Cores + SMT ✖

- Direction#2 (this work): Move to *SIMT* ✔
    - More energy efficient (throughput/watts)
    - Cost-effective (throughput/area)
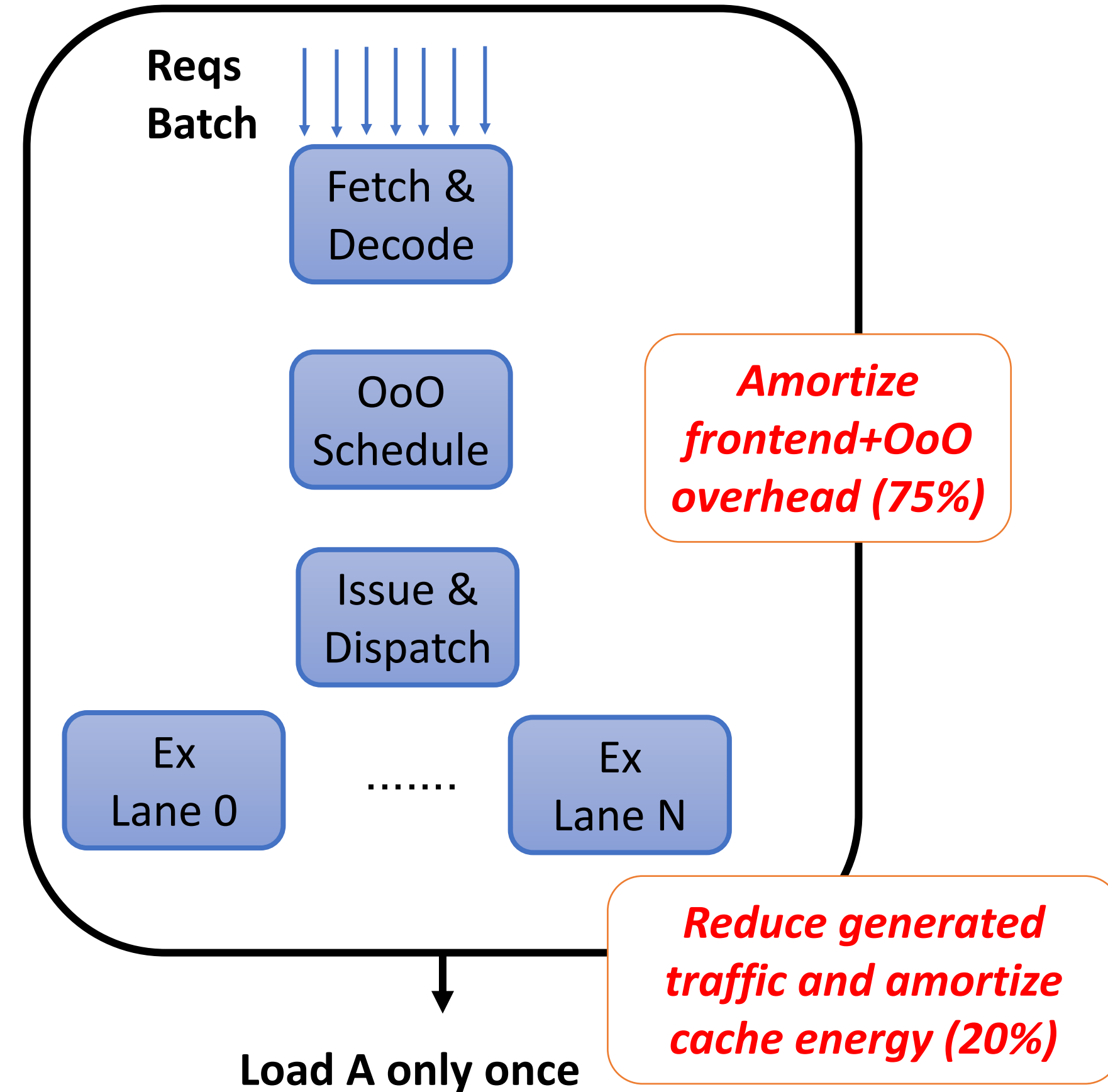   - Better scalability

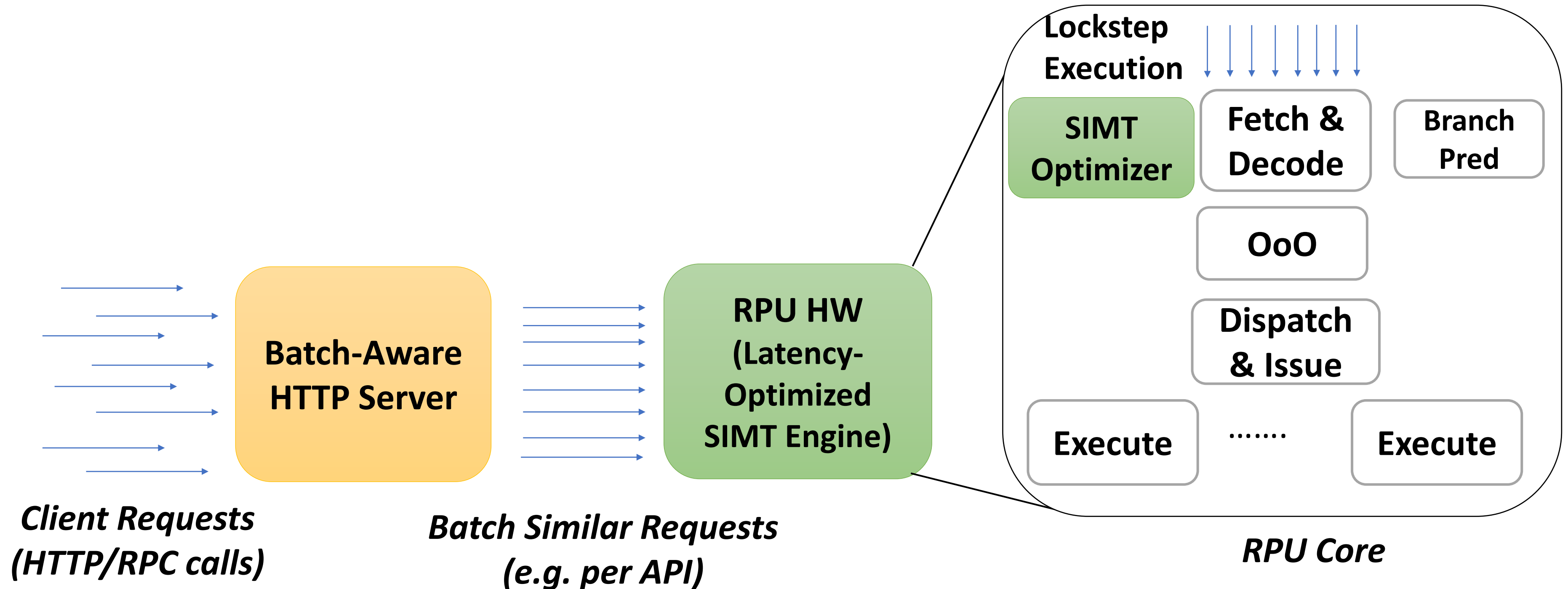*"Let's bring SIMT efficiency to the CPU world!"*

SIMT Efficiency

CPU Multi-Core with Simultaneous Multi-Threading

Request Processing Unit (RPU)
SIMT+OoO Architecture

Amortize frontend+OoO overhead (75%)

Reduce generated traffic and amortize cache energy (20%)

SIMR System Overview

CPU vs GPU vs RPU

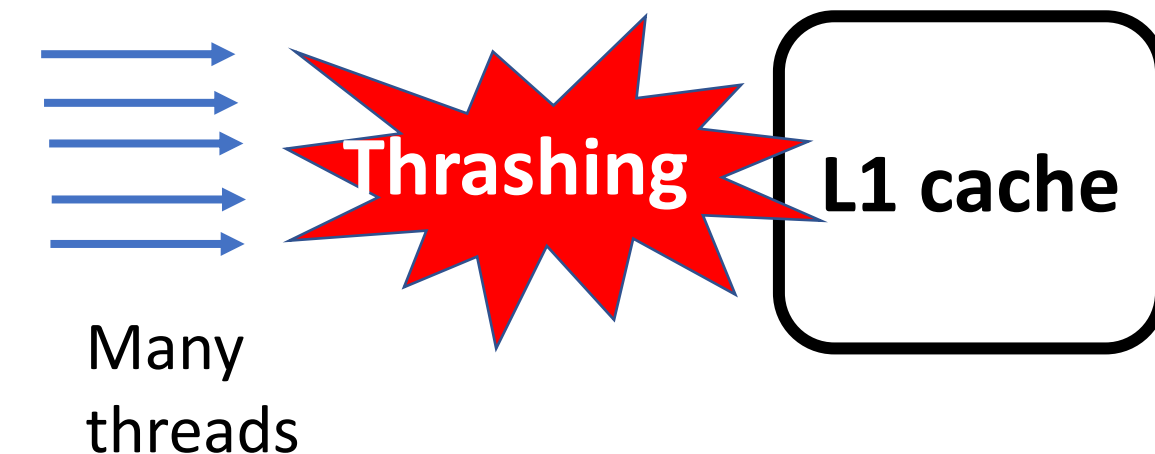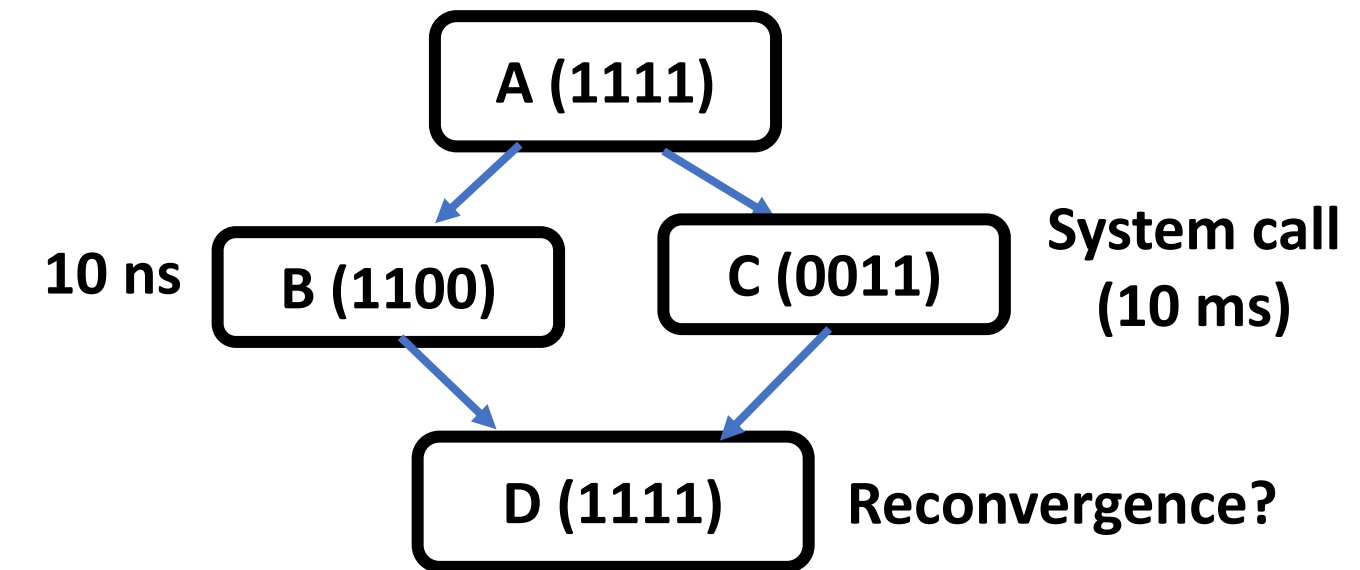| Metric	CPU	GPU	RPU
Core model	OoO	In-Order	OoO
Programming	General-Purpose	CUDA/OpenCL	General-Purpose
ISA	x86/ARM	HSAIL/PTX	x86/ARM
System Calls Support	Yes	No	Yes
Thread grain	Coarse grain	Fine grain	Coarse grain
Threads per core	Low (1-8)	Massive (2K)	Moderate (8-32)
Thread model	SMT	SIMT	SIMT
Consistency	Variant	Weak+NMCA*	Weak+NMCA*
Interconnect	Mesh/Ring	Crossbar	Crossbar

The RPU takes advantage of the latency optimizations and programmability of the CPU

& SIMT efficiency and memory model scalability of the GPU

*NMCA: non-multi copy atomicity

Ren, Xiaowei, et al. "HMG: Extending cache coherence protocols across modern hierarchical multi-GPU systems." HPCA 2020
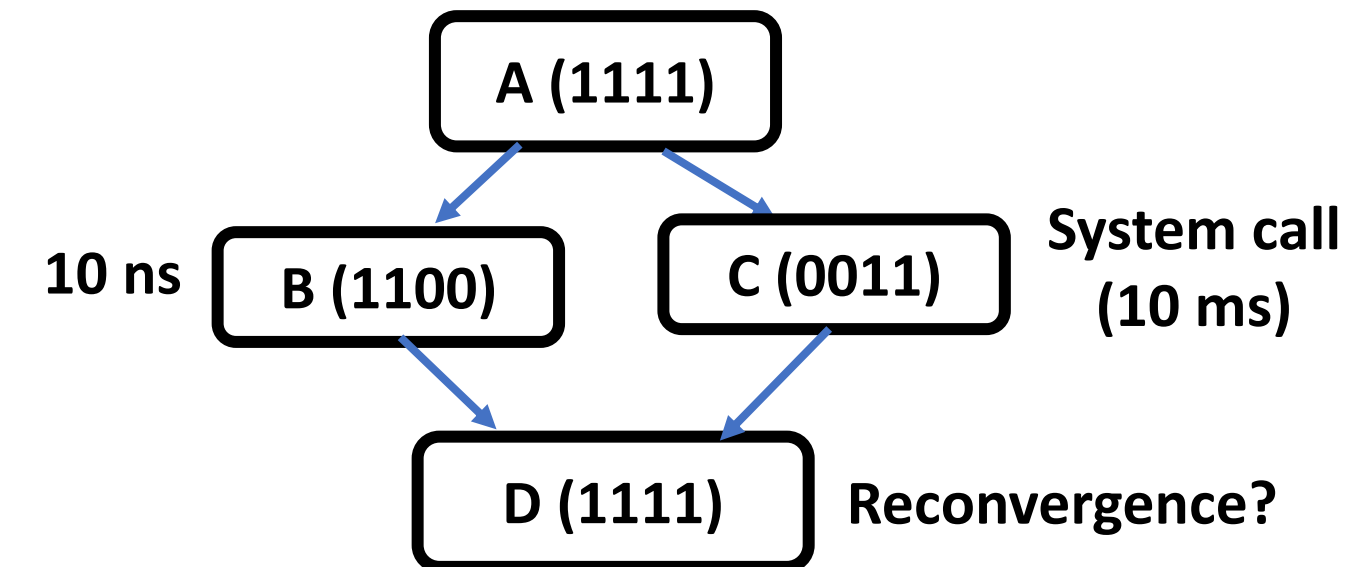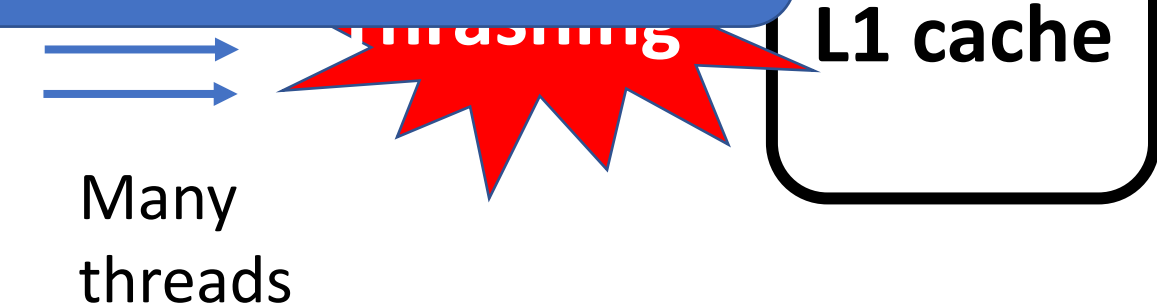
RPU's Challenges

- Control Divergence
  - Challenge: Control divergence with high latency path
  - Solution: System-level batch split

- Memory Divergence
  - Challenge: Cache contention & bank conflicts
  - Solution: Batch tuning, stack/memory coalescing and SIMR-aware memory allocation

- Larger execution units & cache resources
  - Challenge: Higher instruction execution & L1 hit latency
  - Solution: Exploit low IPC and less generated traffic

A (1111)

10 ns **B (1100)** **C (0011)** **System call (10 ms)**

D (1111) **Reconvergence?**

Thrashing **L1 cache**

Many threads

RPU's Challenges

- Control Divergence
  - Challenge: Control divergence with high latency path
  - Solution: System-level batch split

A (1111)

10 ns   B (1100)   C (0011)   System call (10 ms)

D (1111)   Reconvergence?

- Memory Di...
  - C...
  - S... Batch tuning, stack coalescing and SIMR-aware memory allocation

Read more details in the paper on how we address these challenges
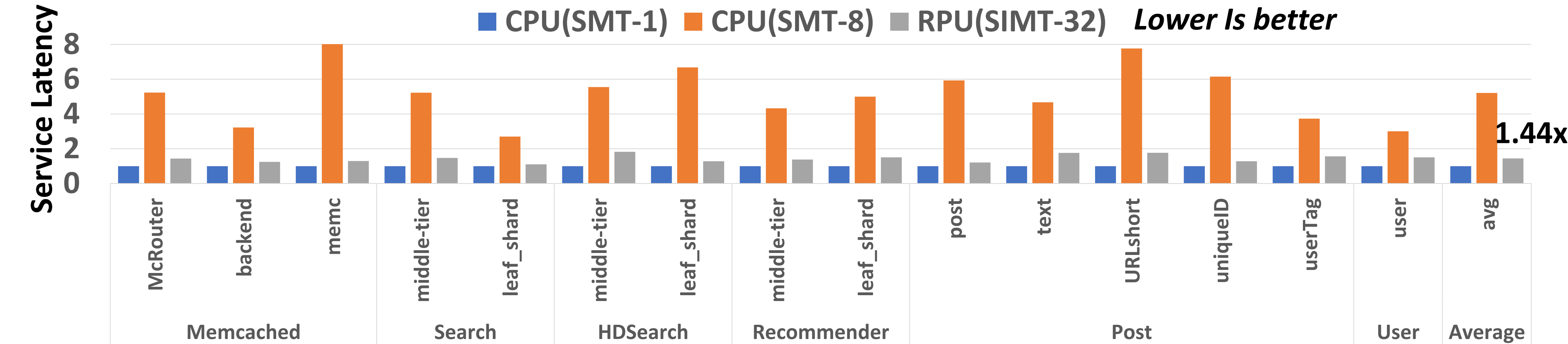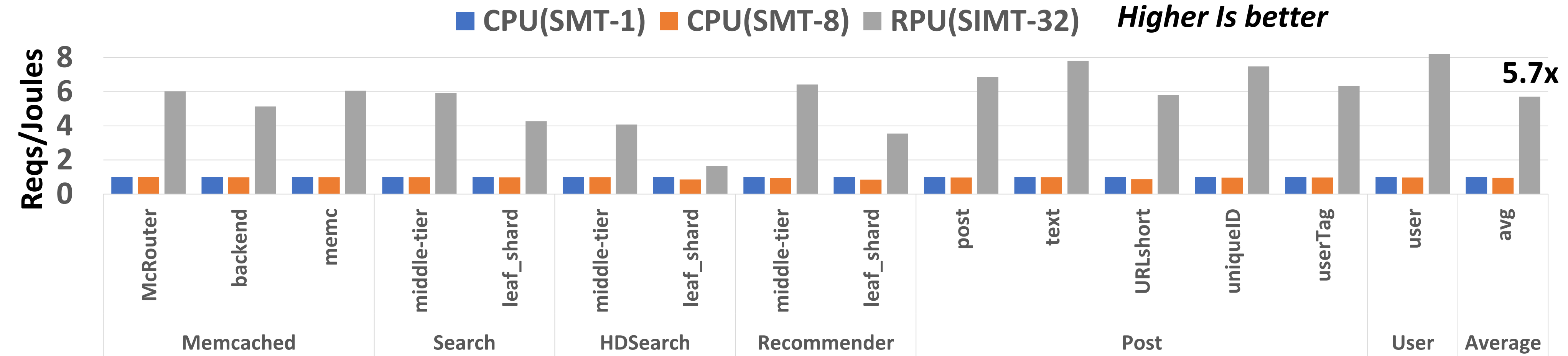
Thrashing   L1 cache

Many threads

- Larger execution units & cache resources
  - Challenge: Higher instruction execution & L1 hit latency
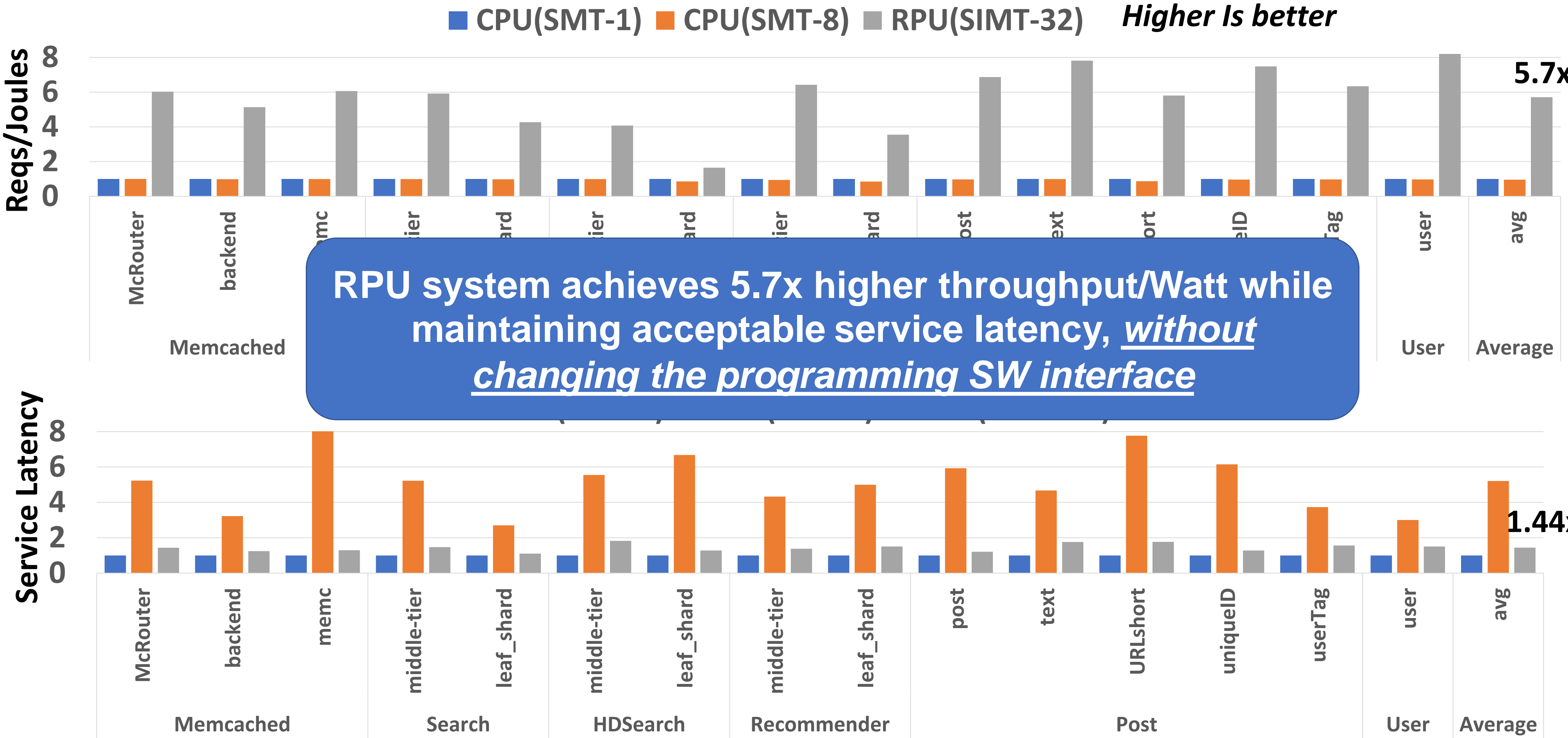 - Solution: Exploit low IPC and less generated traffic

SIMT Control Efficiency
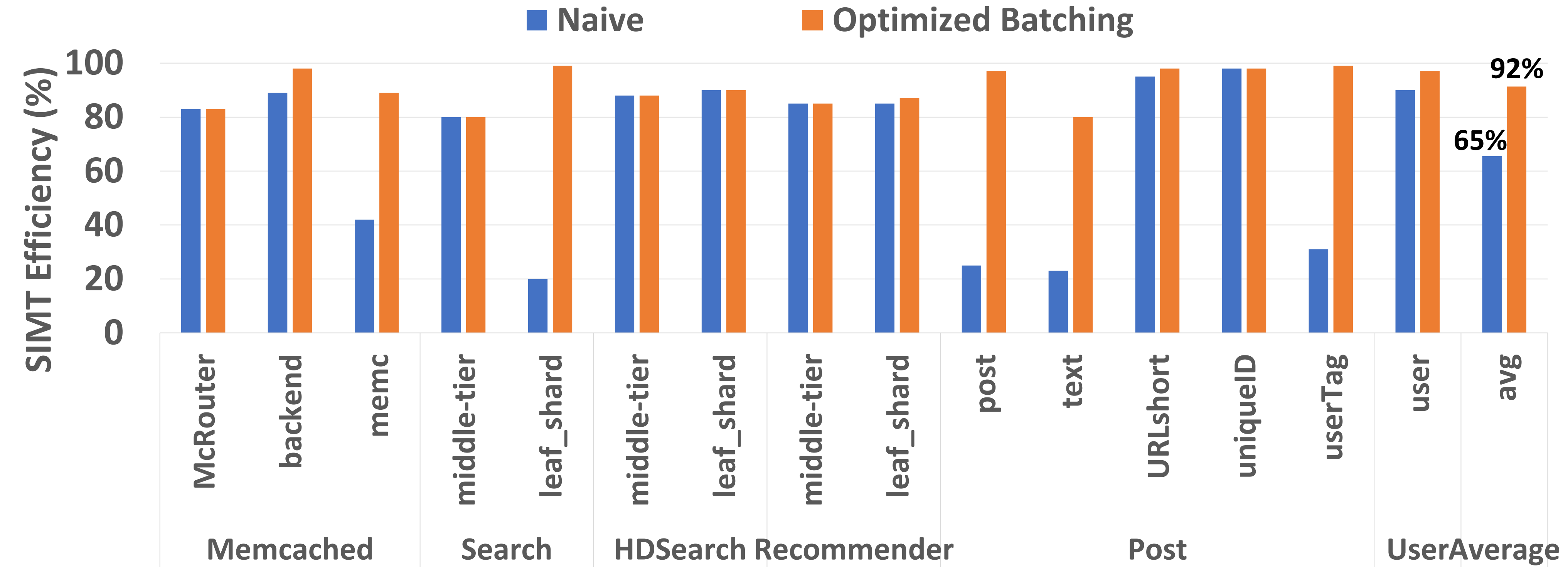
Efficiency and Service Latency Results (Simulation)

Efficiency and Service Latency Results (Simulation)

■ CPU(SMT-1) ■ CPU(SMT-8) ■ RPU(SIMT-32)   *Higher Is better*

RPU system achieves 5.7x higher throughput/Watt while maintaining acceptable service latency, _without changing the programming SW interface_

Summary

- *Request Similarity* is abundant in the data center.

- We start with *OoO CPU* design and augment it with *SIMT execution* to maximize chip utilization and exploit the similarity.

- We co-design the software stack to support *batching* and awareness of SIMT execution.

SIMT efficiency is high in the open-source microservices we study.

DeathStarBench

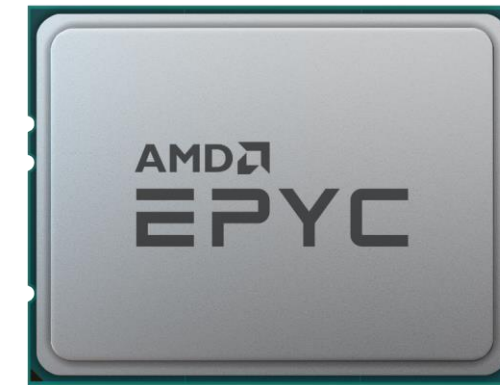μSuite: A Benchmark Suite for Microservices

We are very interested in evaluating SIMT control efficiency in proprietary production microservices.
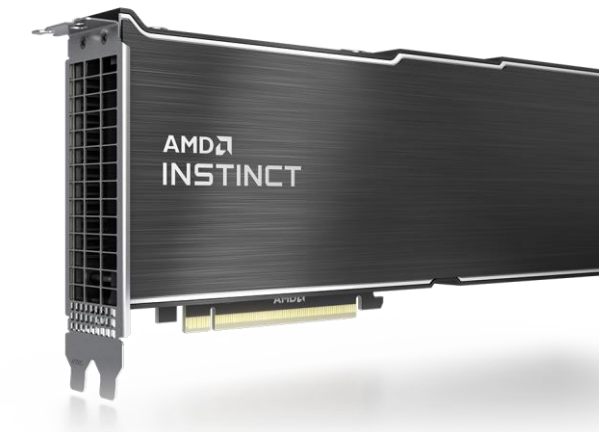
Google

facebook

Thank You!
Q&A?

Instruction level parallelism (ILP) &
Thread level parallelism (TLP)

Data level parallelism (DLP)

Request level parallelism (RLP)