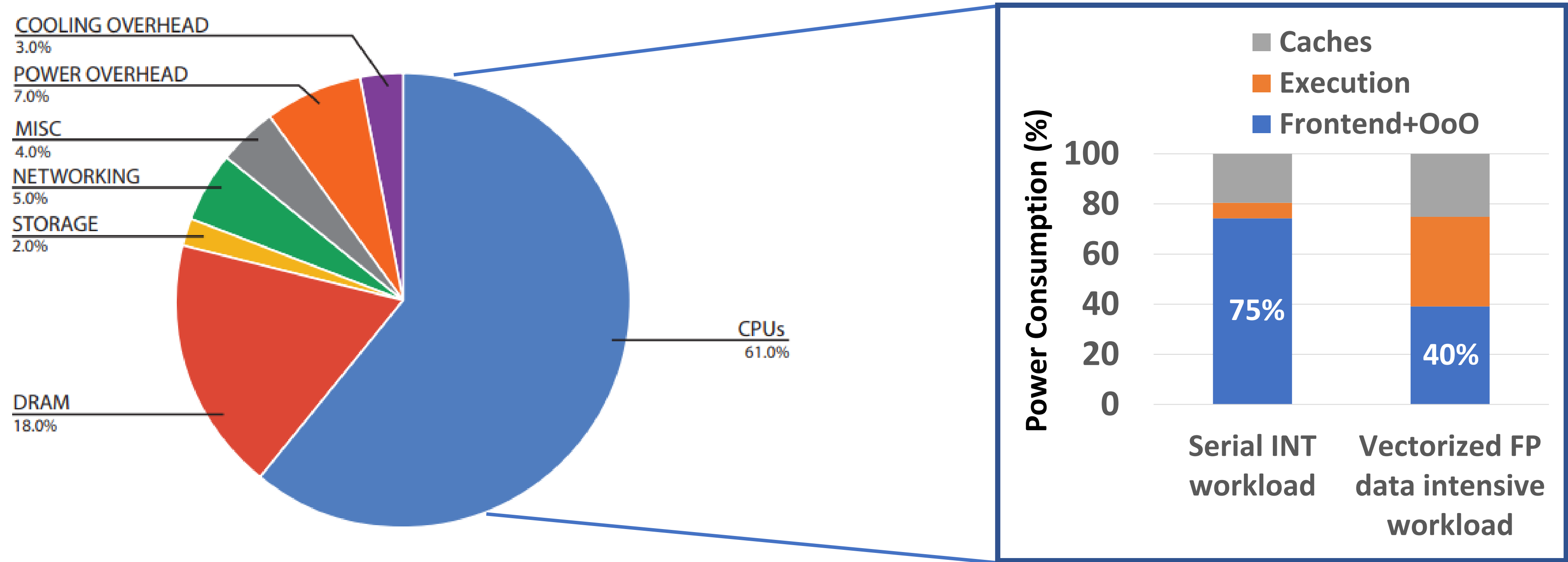


SIMR: Single Instruction Multiple Request Processing for Energy-Efficient Data Center Microservices

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

Purdue University

Datacenter Power Breakdown



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

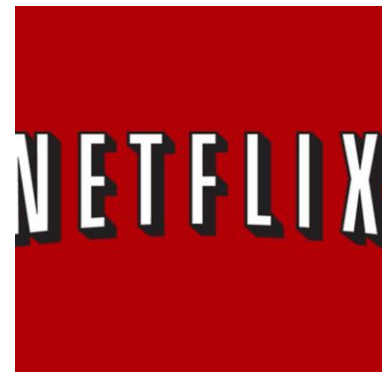
1 Application, Million of Users

Google

facebook

Private Datacenter

Uber



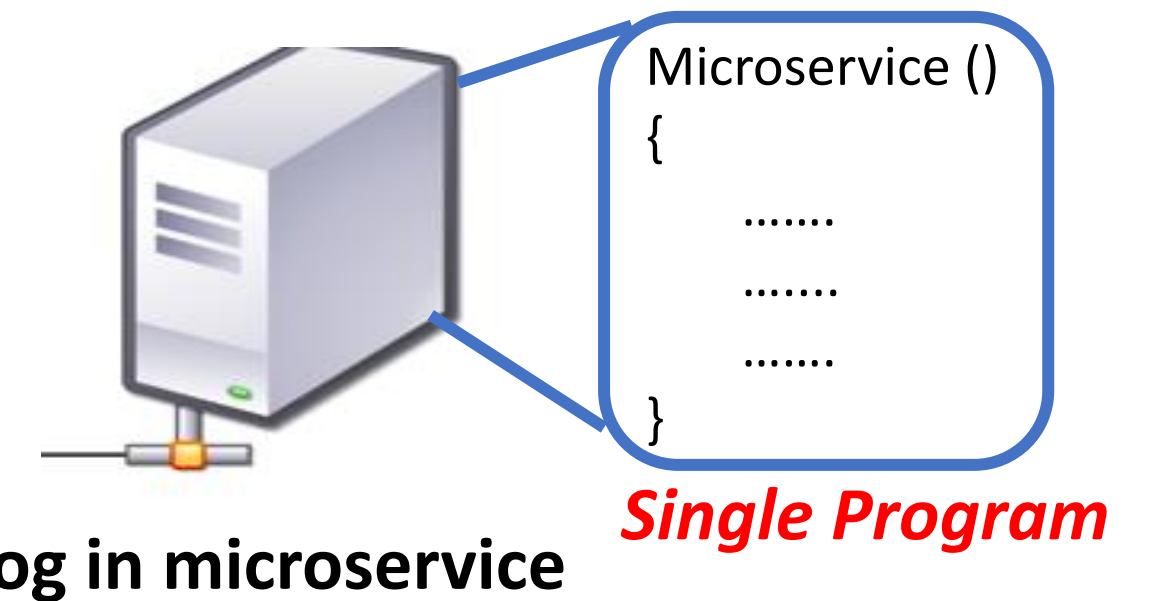
Public Datacenter

“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



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



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

Rhythm, ASPLOS 2014

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

MemcachedGPU, SoCC 2015

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

ispc, InPar 2012

Recall: GPUs and SIMDs were designed to execute data parallel portion (i.e., loops) not the entire application

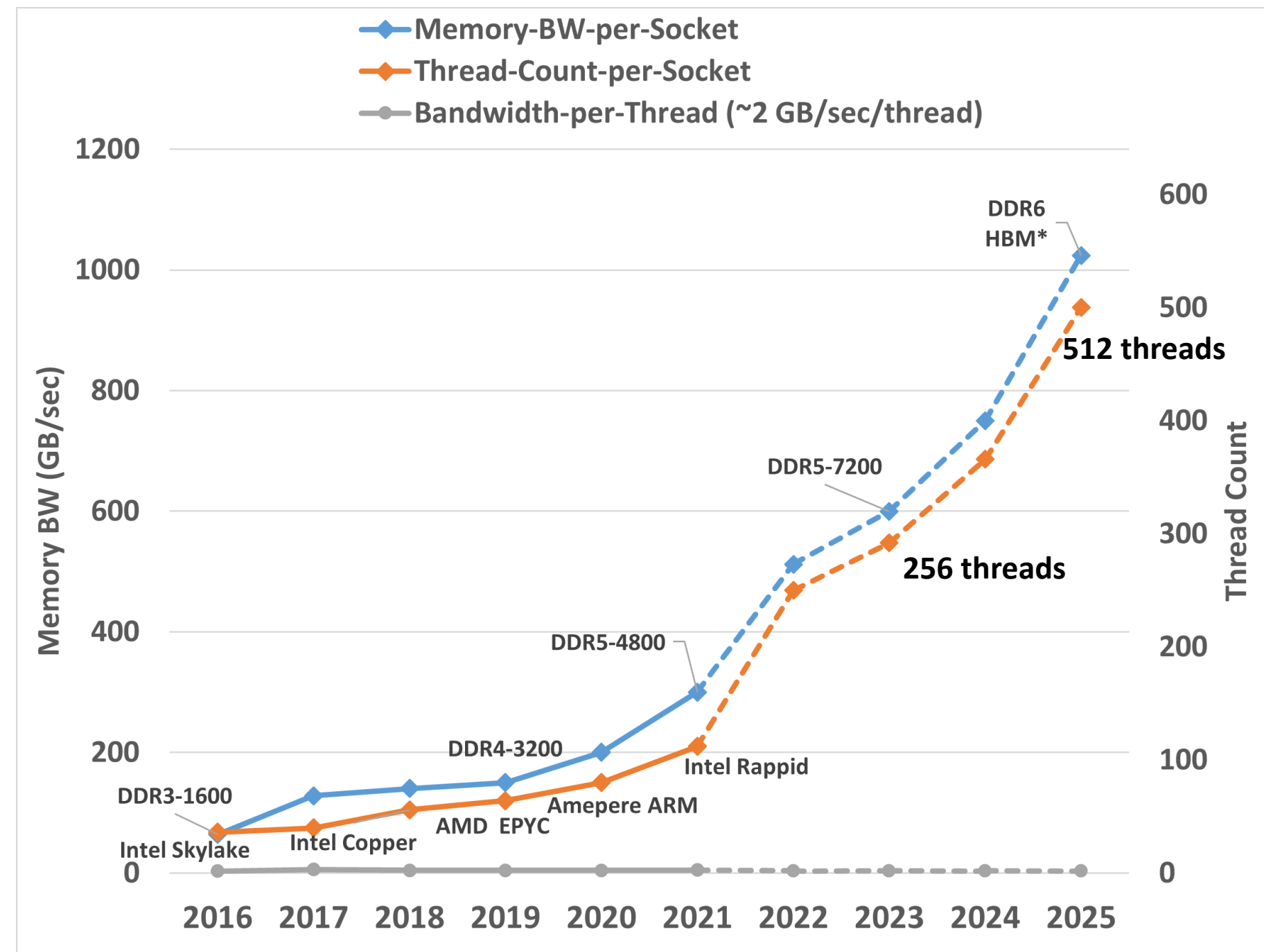
“Slower but energy-efficient wimpy cores only win for general data center workloads if their single-core speed is reasonably close to that of mid-range brawny cores”

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





Urs Hölzle
Google SVP

Off-Chip BW Scaling



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

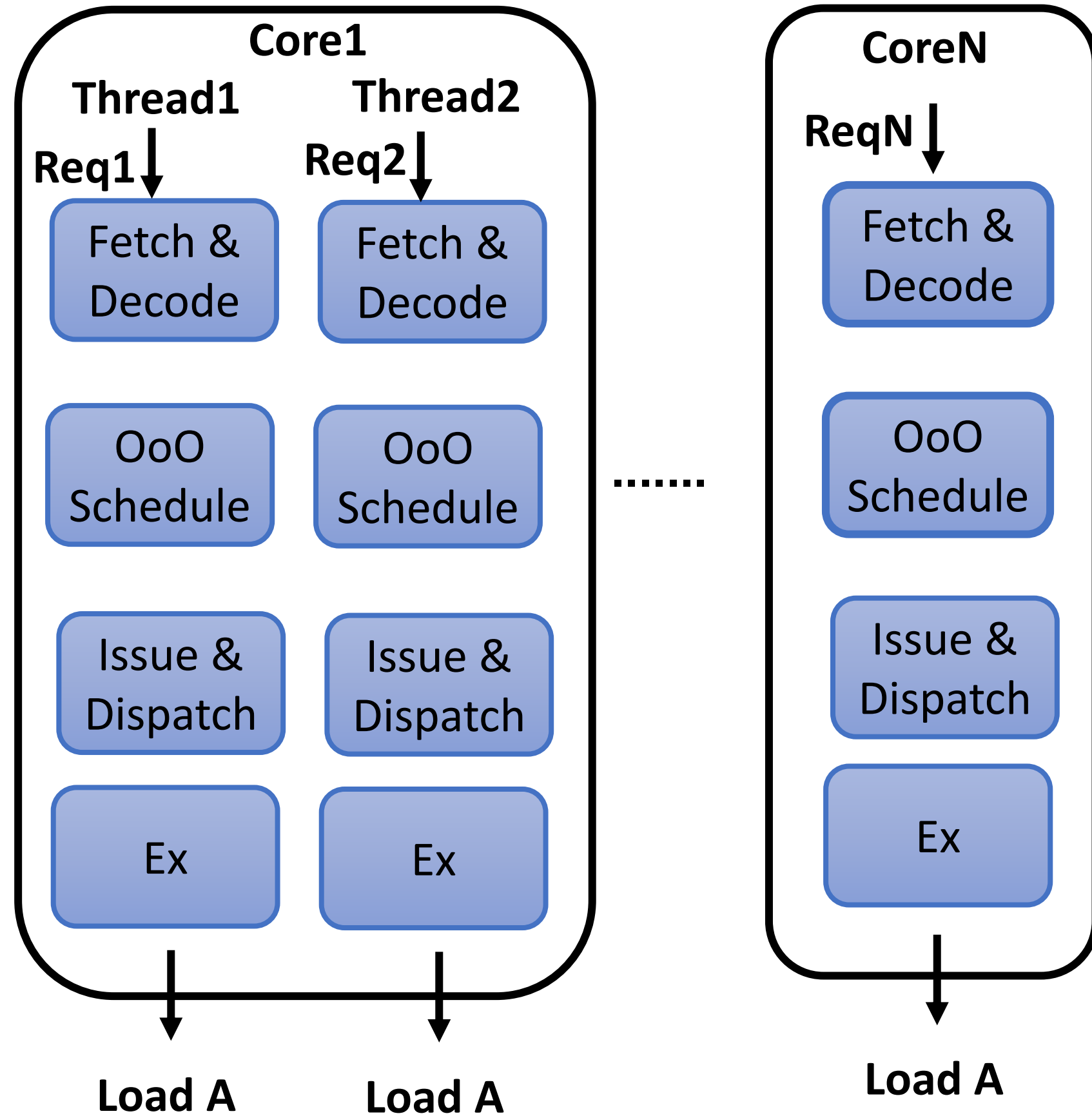
How to increase on-chip throughput of CPU?

- 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

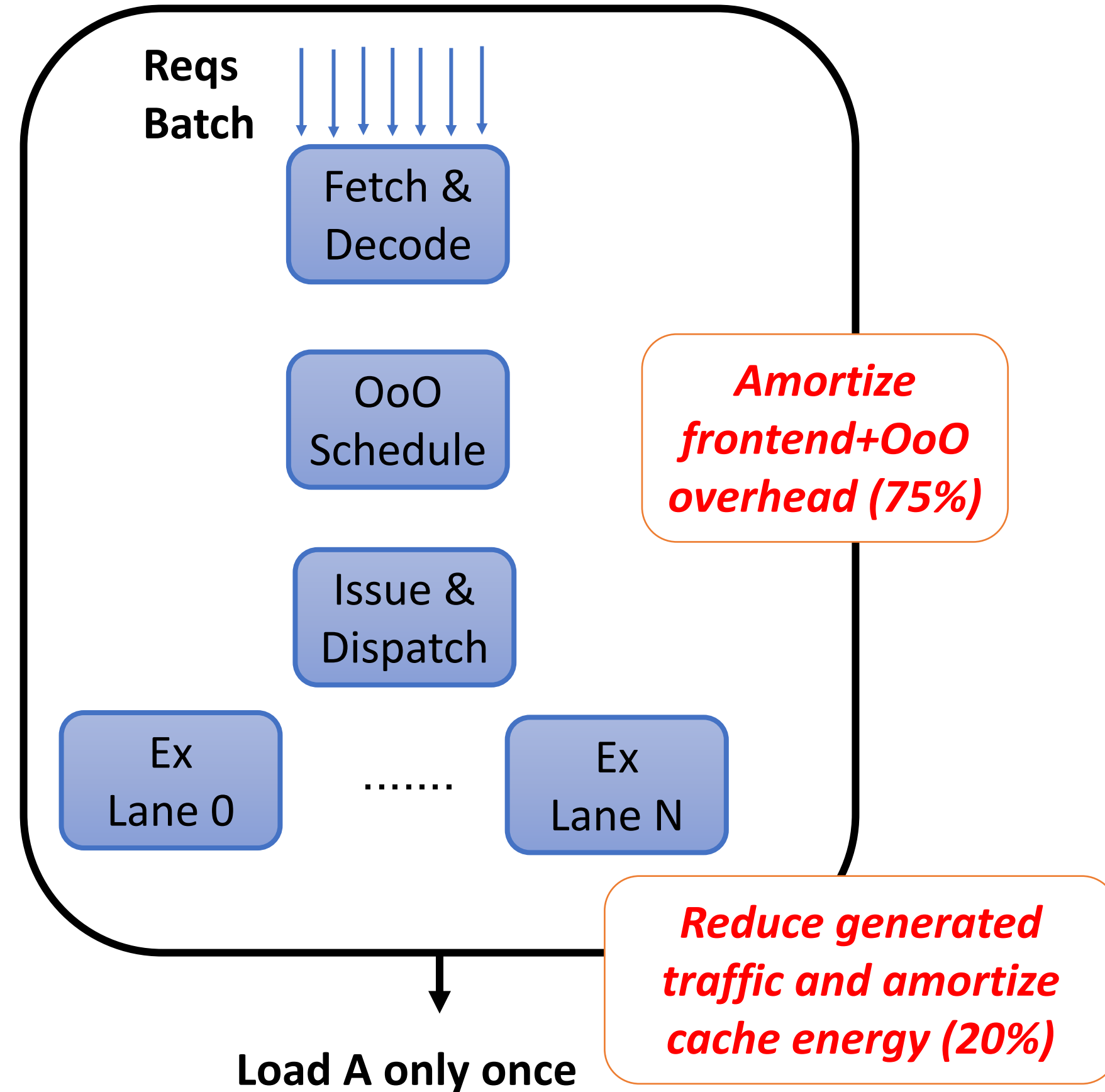
“Let’s bring SIMD efficiency to the CPU world!”

SIMT Efficiency

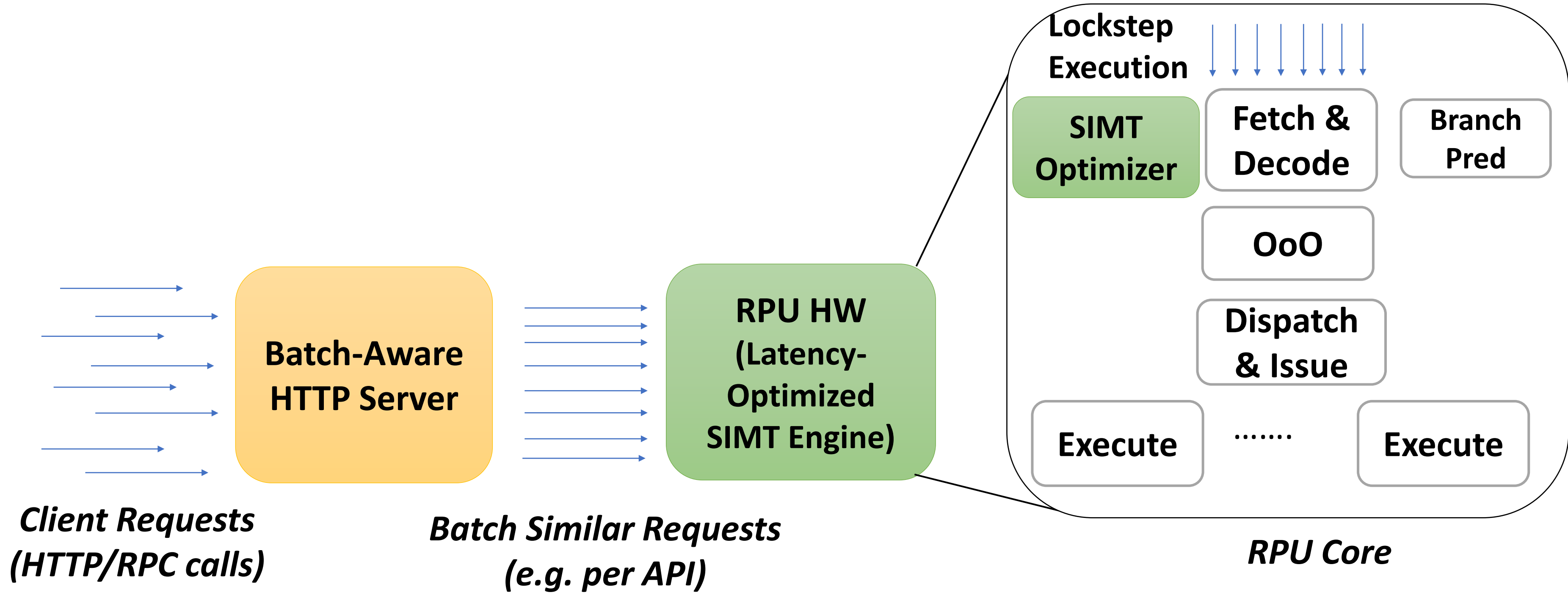
CPU Multi-Core with Simultaneous Multi-Threading



Request Processing Unit (RPU)
SIMT+OoO Architecture



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

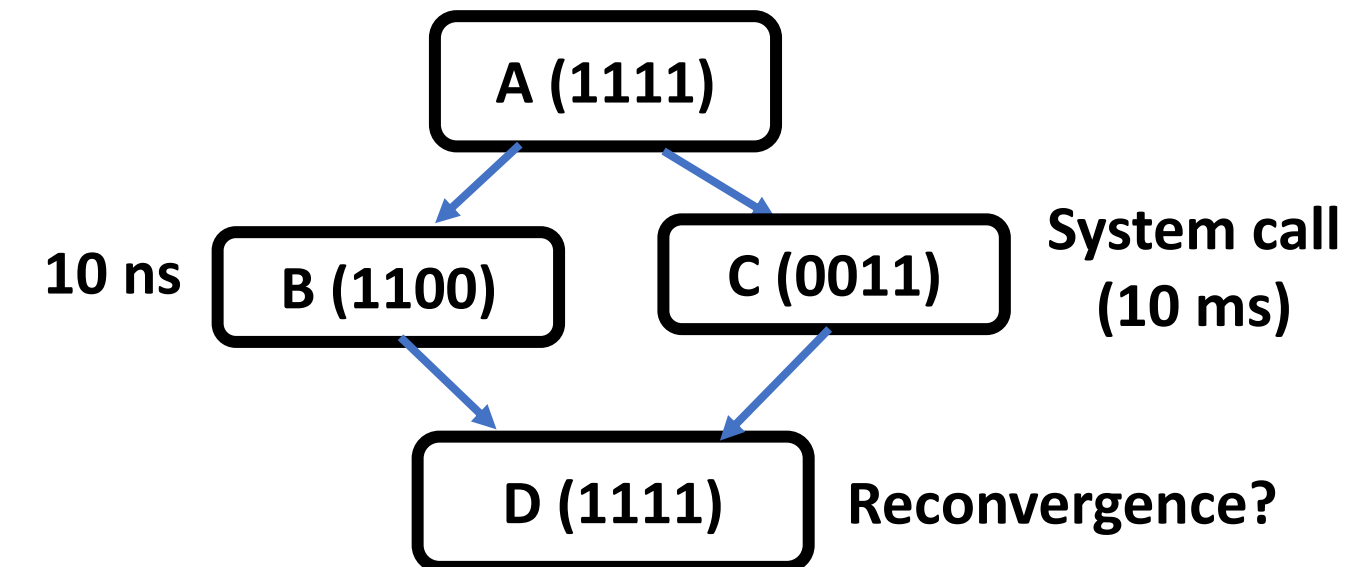
& SIMT efficiency and memory model scalability of the GPU

*NMCA: non-multi copy atomicity

RPU's Challenges

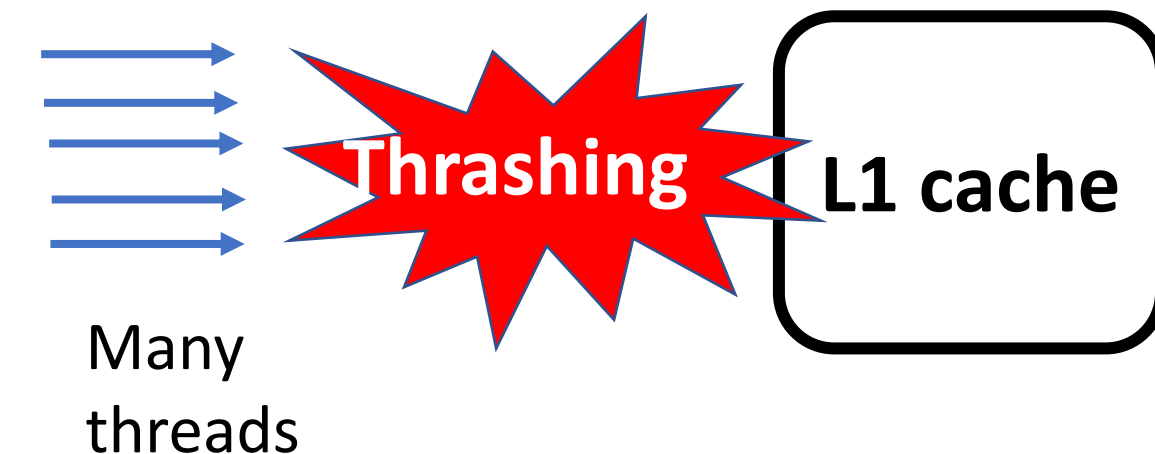
- Control Divergence

- Challenge: Control divergence with high latency path
- Solution: Optimized batching & System-level batch split



- Memory Divergence

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



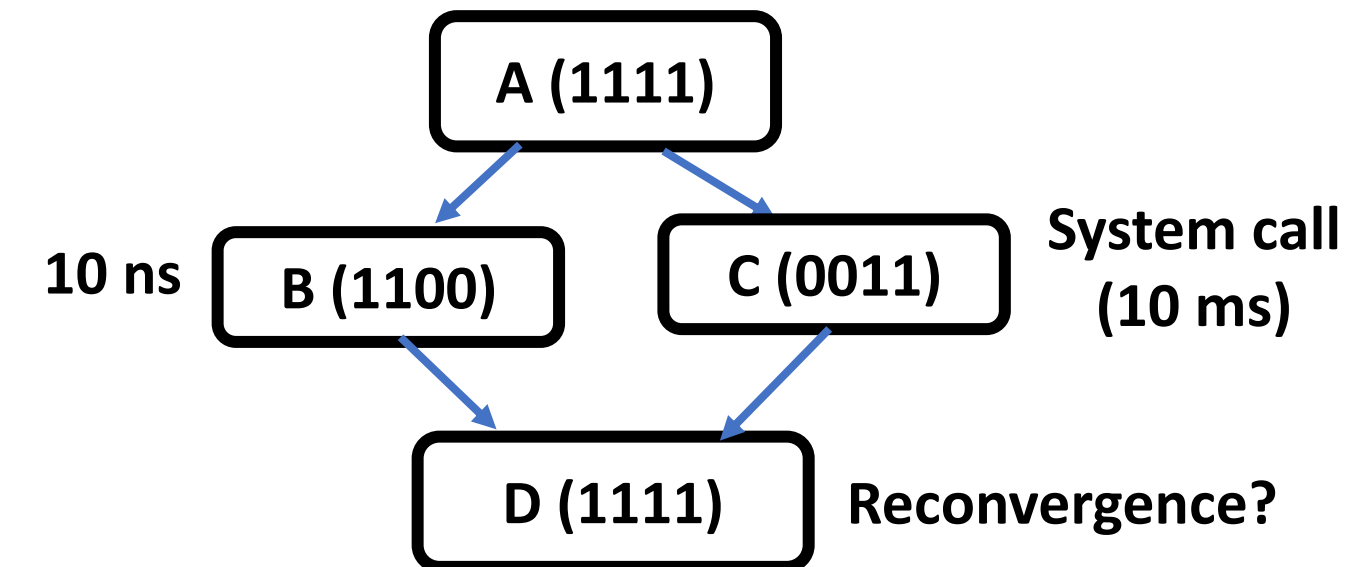
- Larger execution units & cache resources

- Challenge: Higher instruction execution & L1 hit latency
- Solution: Exploit low IPC, less generated traffic and employ sub-batching interleaving

RPU's Challenges

- Control Divergence

- Challenge: Control divergence with high latency path
- Solution: Optimized batching & System-level batch split



- Memory

- Challenge: High memory access latency
- Solution: Batch tuning, stack/memory coalescing and SIMR-aware memory allocation

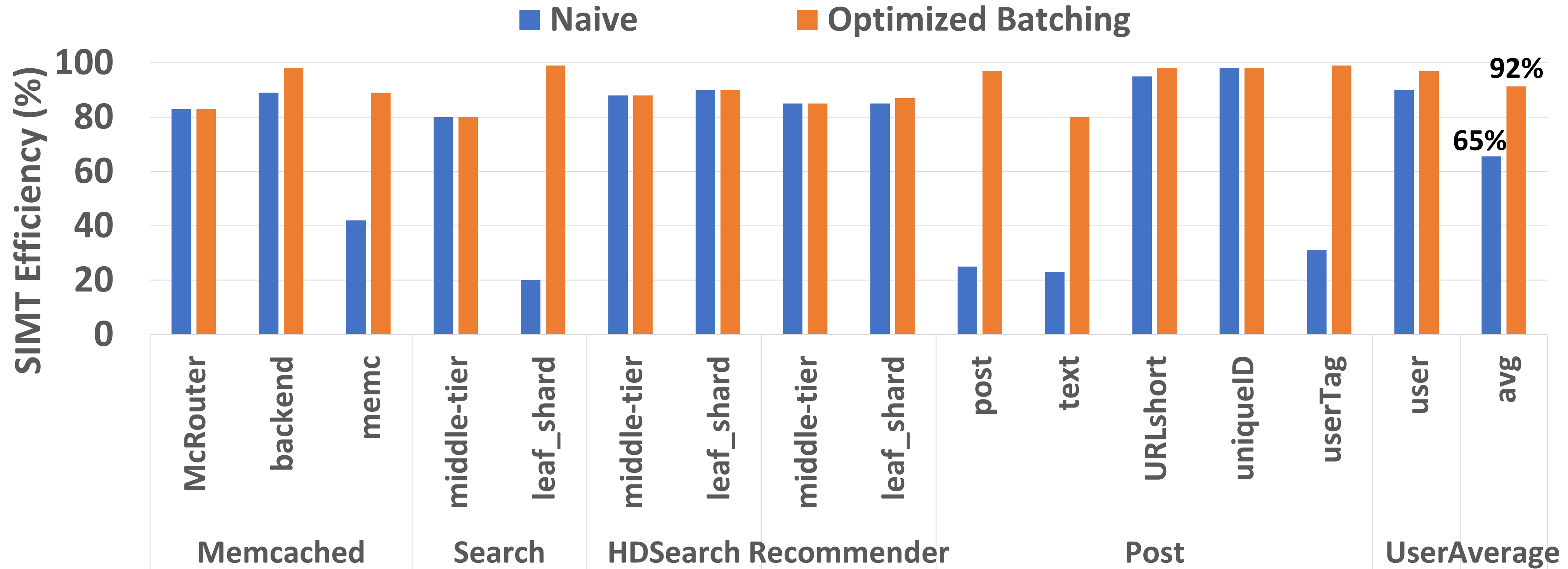
Read more details in the paper on how we address these challenges



- Larger execution units & cache resources

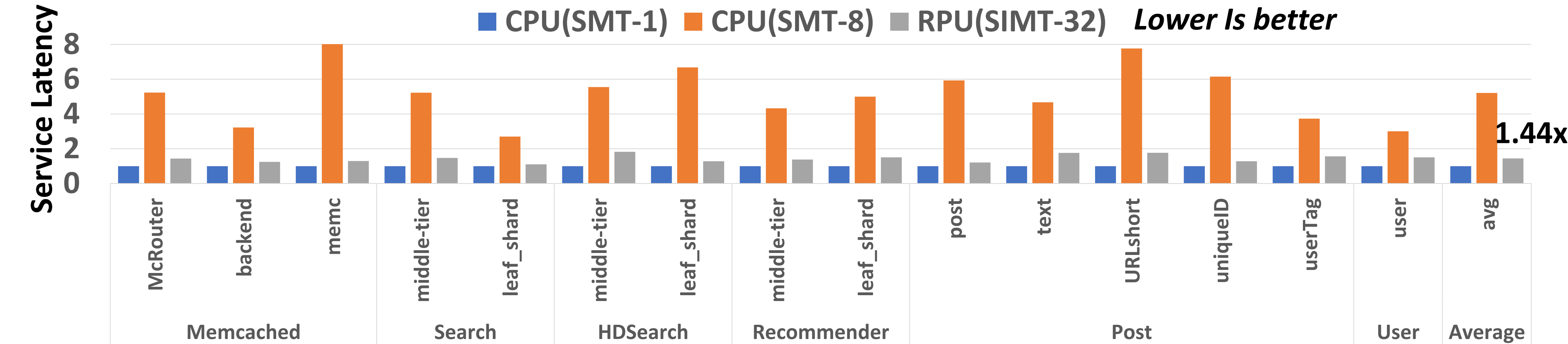
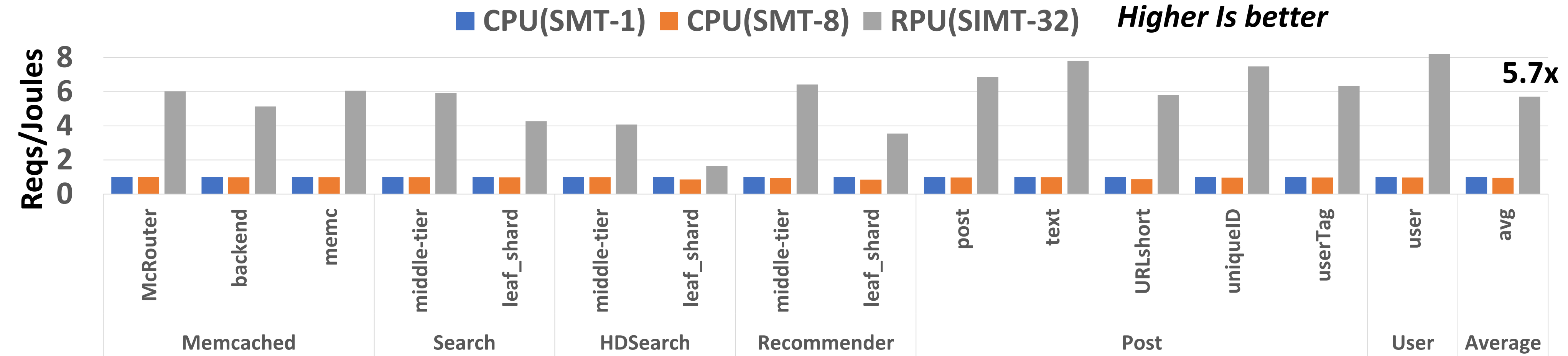
- Challenge: Higher instruction execution & L1 hit latency
- Solution: Exploit low IPC, less generated traffic and employ sub-batching interleaving

SIMT Control Efficiency



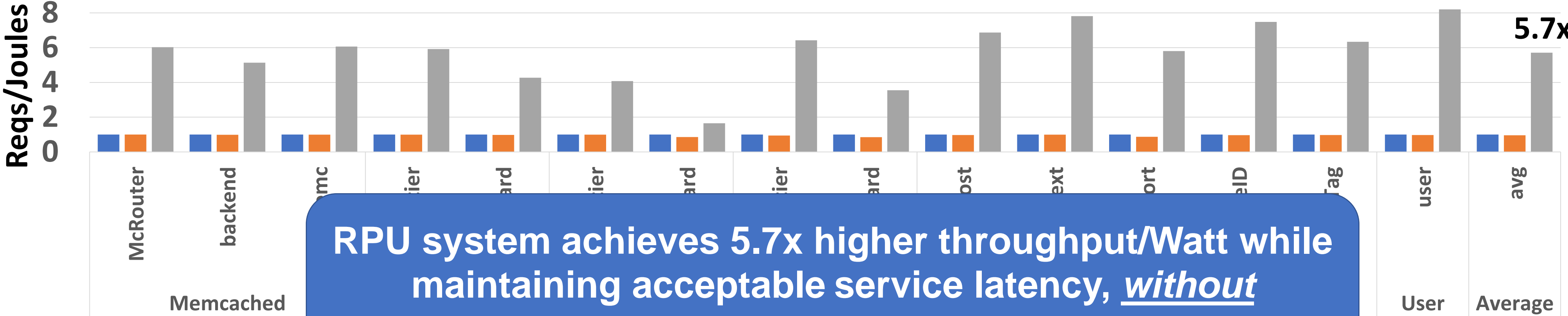
Notes: (1) Batch Size = 32 & #batches=75, (2) System Calls are not traced, (3) $\text{SIMT Eff} = \text{scalar-instructions} / (\text{batch-instructions} * \text{batch-size})$, (4) fine-grain locking are assumed. Other assumptions are included in the paper.

Efficiency and Service Latency Results (Simulation)

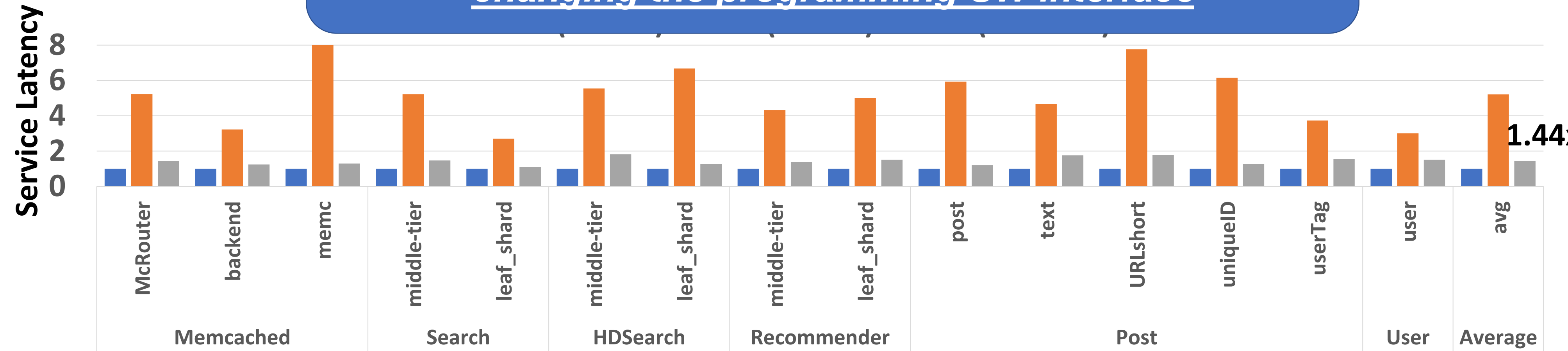


Efficiency and Service Latency Results (Simulation)

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

μ Suite: A Benchmark Suite for Microservices

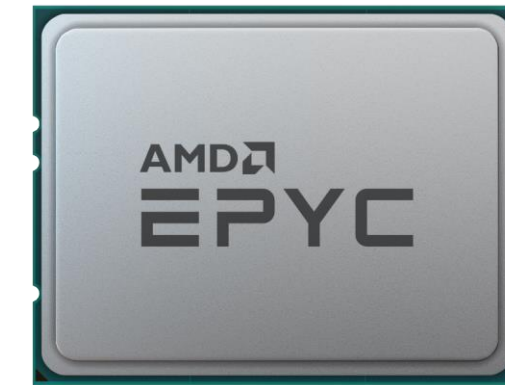
We are very interested in evaluating SIMT control efficiency in proprietary production microservices.

Google
facebook

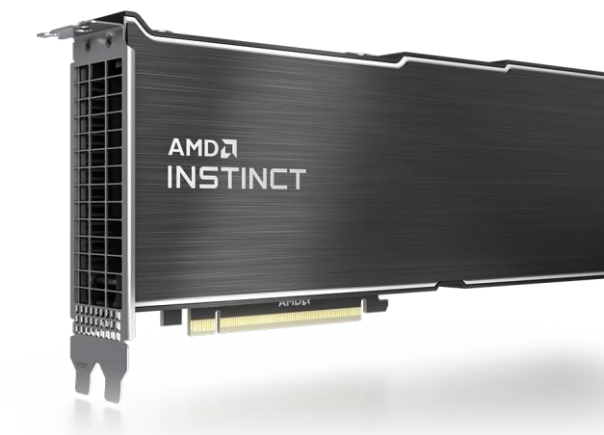
Thank You!

Q&A?

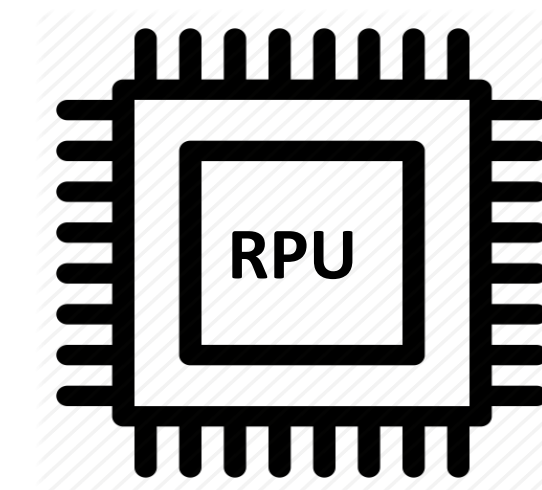
Instruction level parallelism (ILP) &
Thread level parallelism (TLP)



Data level parallelism (DLP)

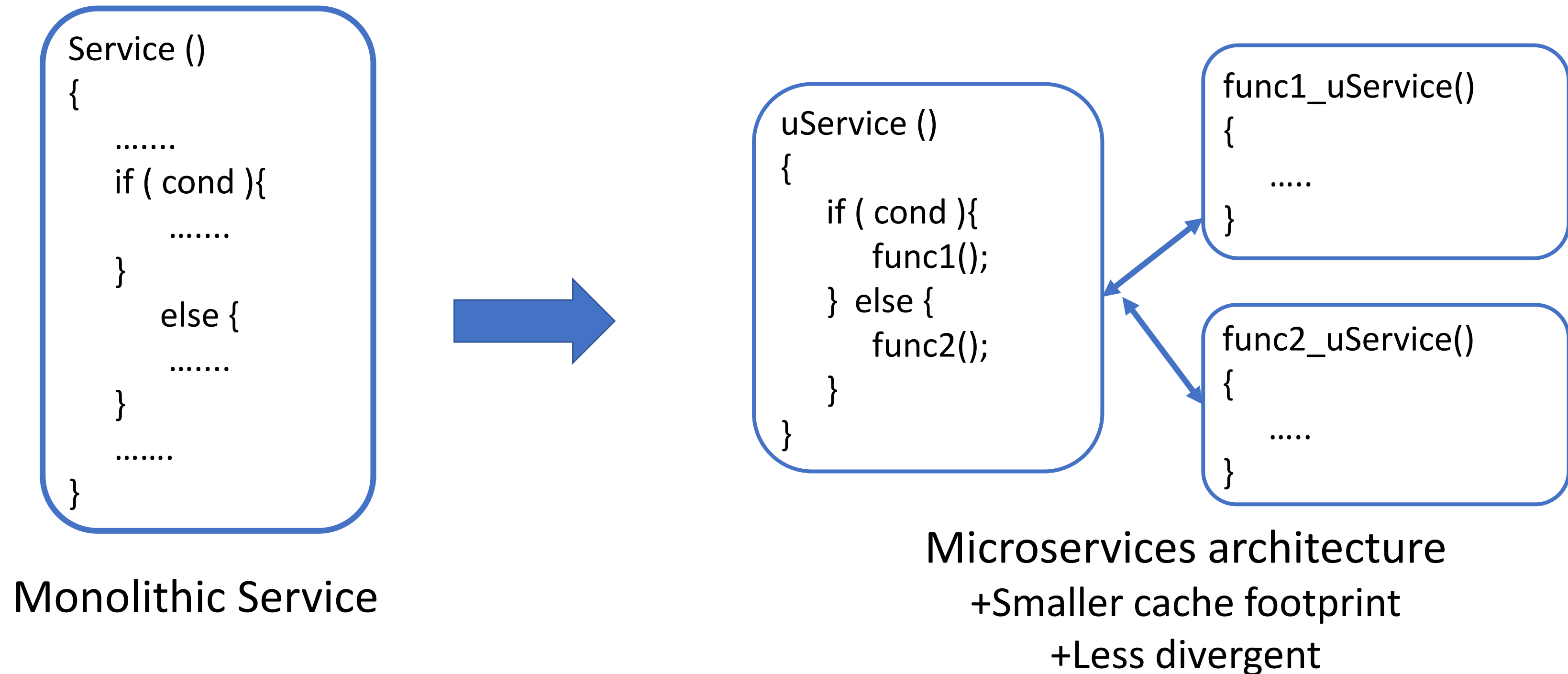


Request level parallelism (RLP)



Backup Slides

SIMT-friendly Microservices



Key Observation#3: Microservices reduce the per-thread cache requirement and minimize control-flow variations between concurrent threads

Batching Optimization

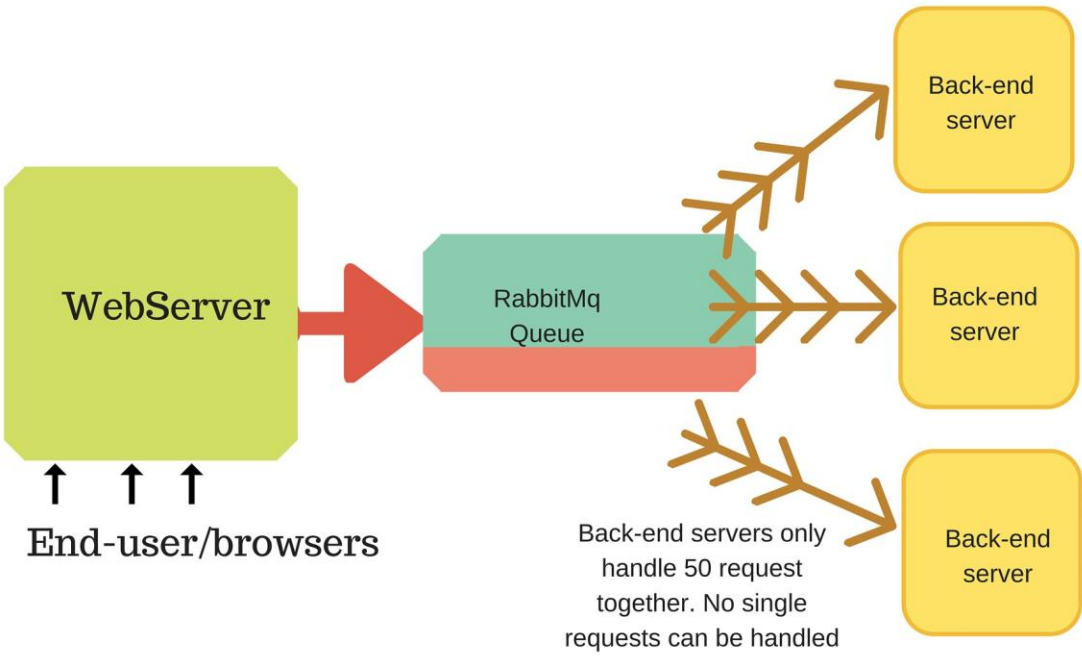
From Google’s Production DL Inference

Production						MLPerf 0.7		
DNN	ms	batch	DNN	ms	batch	DNN	ms	batch
MLP0	7	200	RNN0	60	8	Resnet50	15	16
MLP1	20	168	RNN1	10	32	SSD	100	4
CNN0	10	8	BERT0	5	128	GNMT	250	16
CNN1	32	32	BERT1	10	64			

Table 5. Latency limit in ms and batch size picked for TPUv4i.

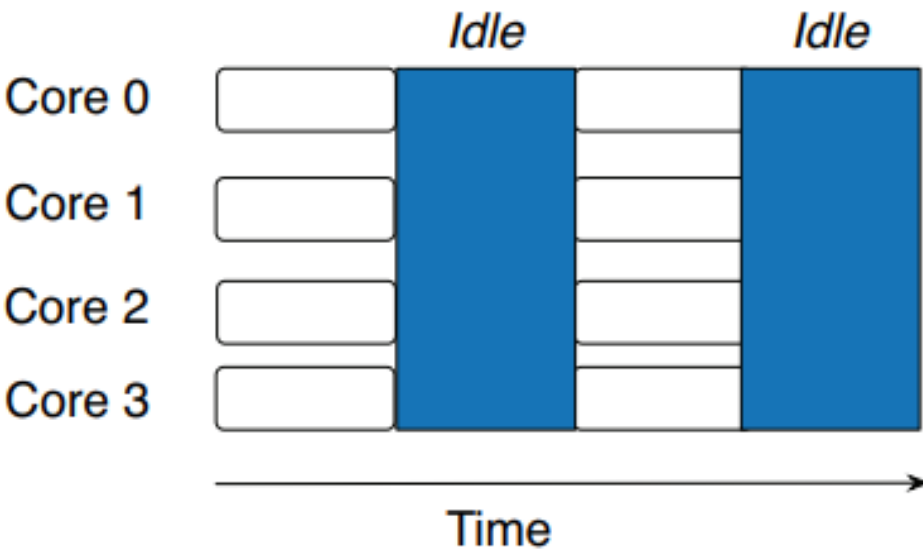
DL Inference Batching

Memcached servers



Network Batching

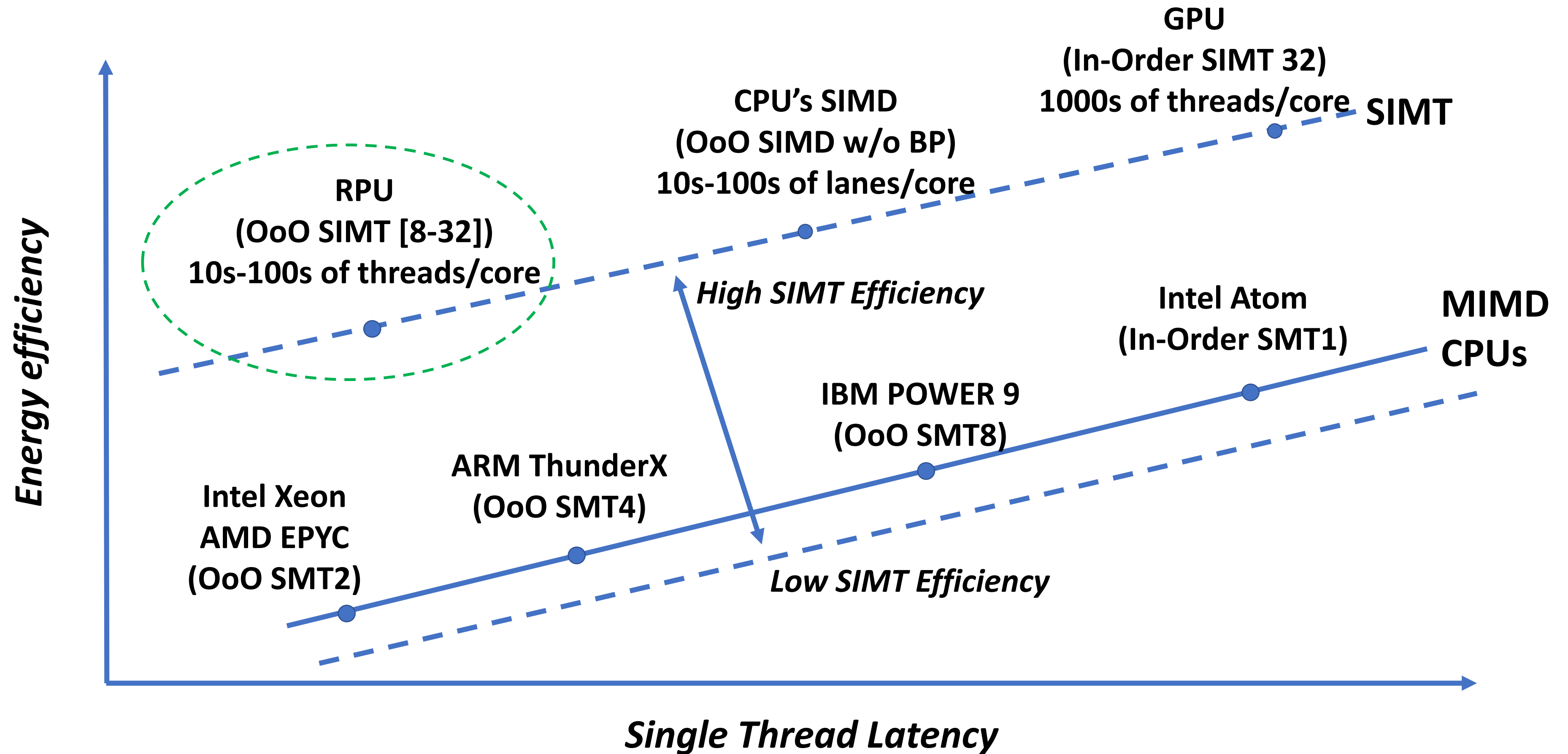
Power management



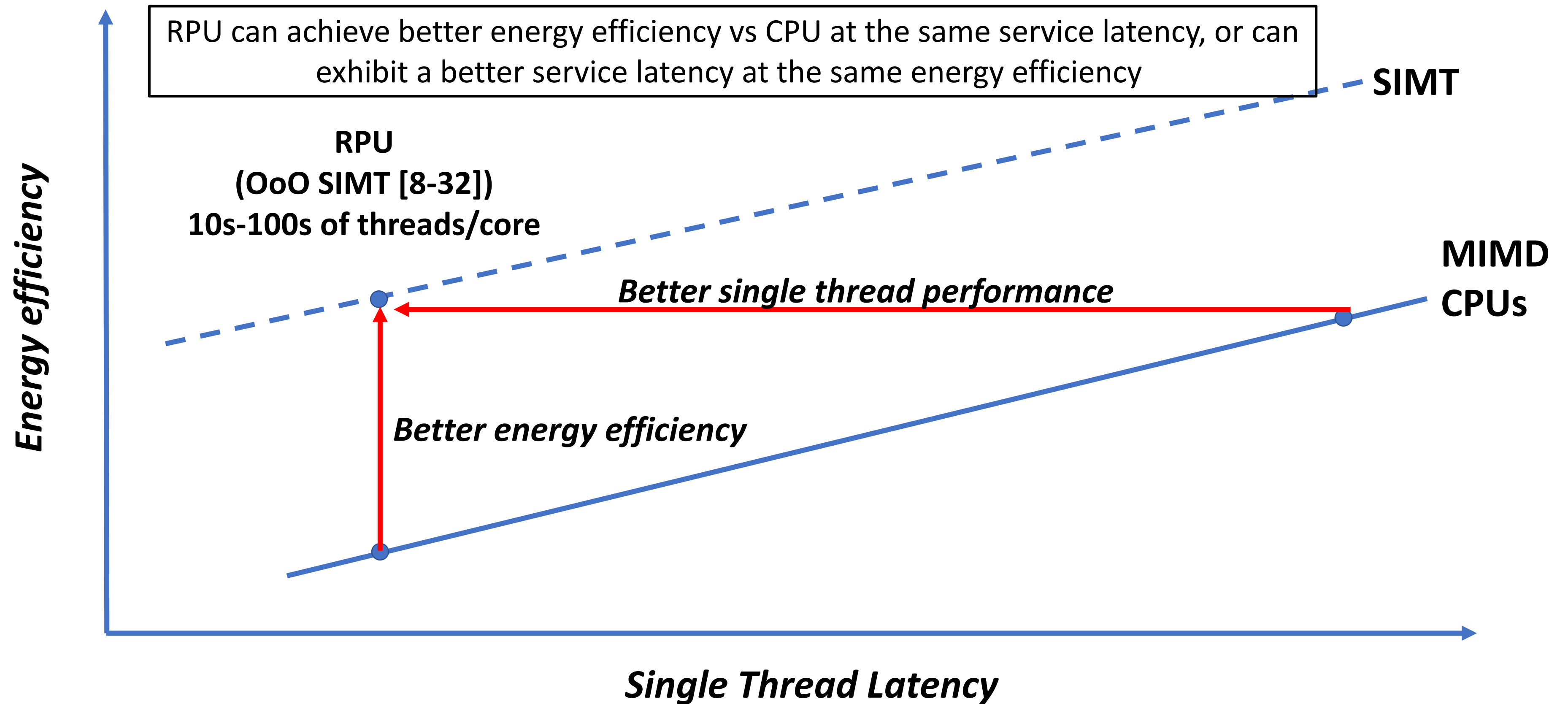
Batching for deep sleep

Key Observation#4: Modern data centers already rely on request batching heavily

Latency & Energy-Efficiency Tradeoff



Latency & Energy-Efficiency Tradeoff



HW/SW Stack

Webservice (C++, PHP, ...)
ARM/x86 compiler
HTTP server
Runtime/libs (pthread, cstdlib, ..)
OS (Process, VM, I/Os)
Multi Core CPU

CPU SW Stack

CUDA
CUDA compiler
Nvidia Triton HTTP server
CUDA runtime/libs (cudalib, tensorRT, ..)
OS (I/Os management)
CUDA driver (VM/thread management)
GPU Hardware

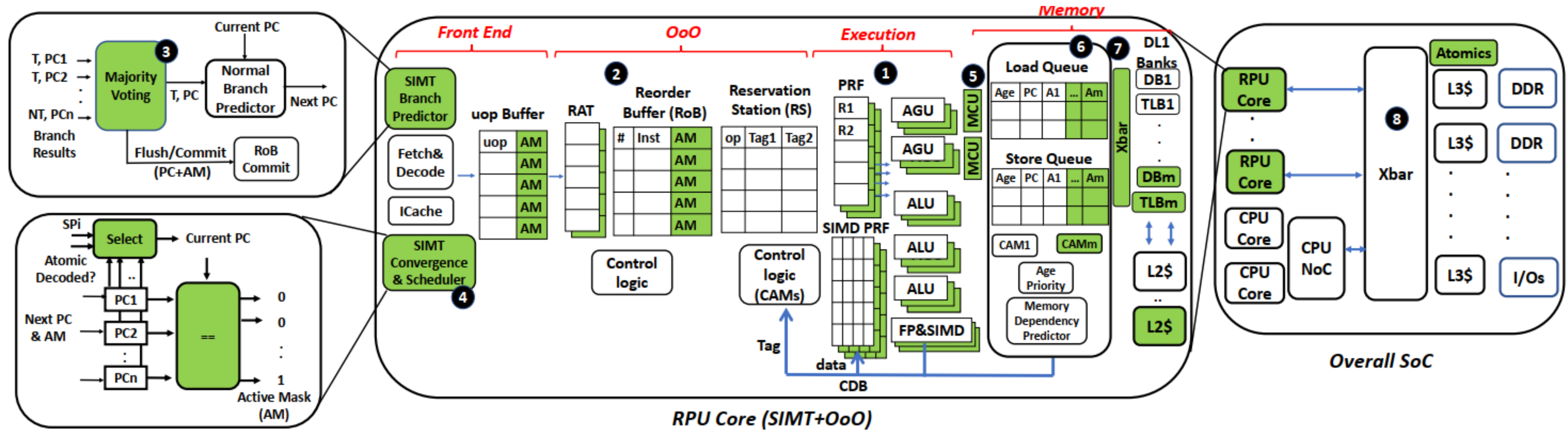
GPU SW Stack

Webservice (C++, PHP, ...)
ARM/x86 compiler
Batch-aware HTTP server
Runtime/libs (pthread, cstdlib, ..)
OS (I/Os management)
RPU driver (VM/thread management)
RPU Hardware

RPU SW Stack

- For RPU, we keep the SW programming interface as in the CPU
- Some VM&process management system calls are reimplemented in the RPU driver to be batch-aware

RPU HW



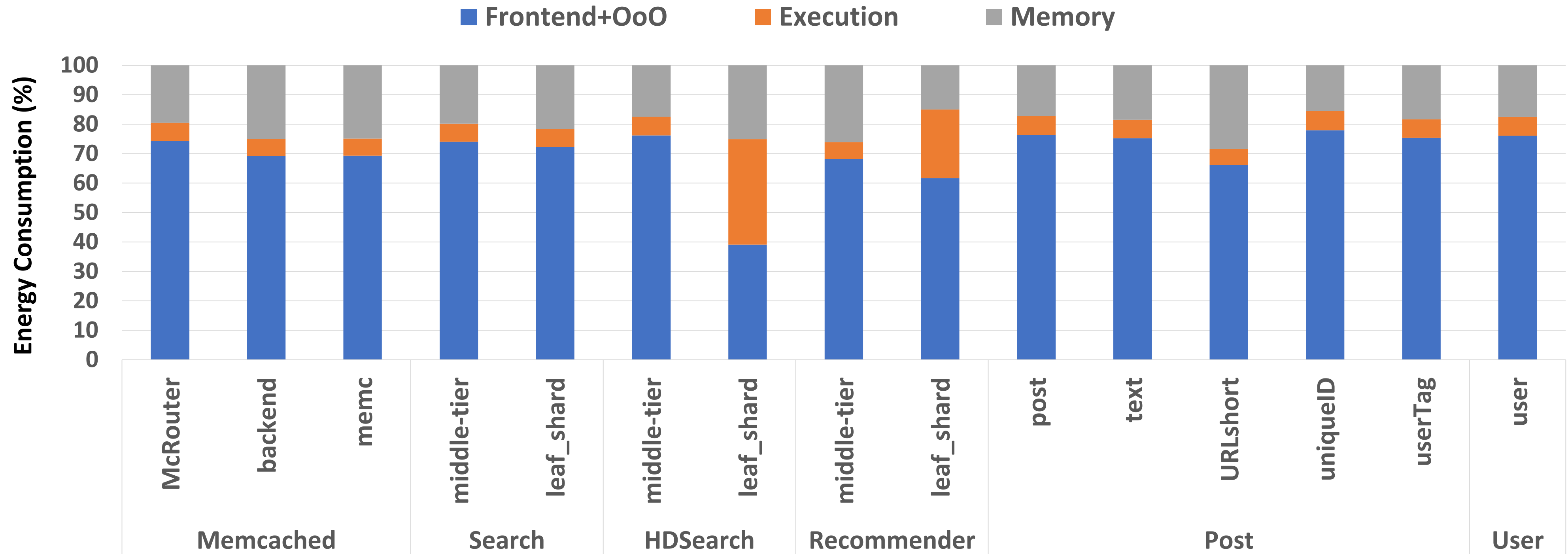
Energy Efficiency of CPU vs RPU (Analytical Model)

$$\frac{\text{CPU Energy}}{\text{RPU Energy}} = \frac{\text{Execution Energy} + \text{Memory system Energy} + \text{Front_OoO Energy} + \text{Static Energy}}{\text{Execution Energy} + (1 - r) (\text{Memory system Energy}) + \frac{1}{n * \text{eff}} [\text{Front_OoO Energy} + r * \text{Memory system Energy} + \text{Static Energy}] + \text{SIMT_Overhead}}$$

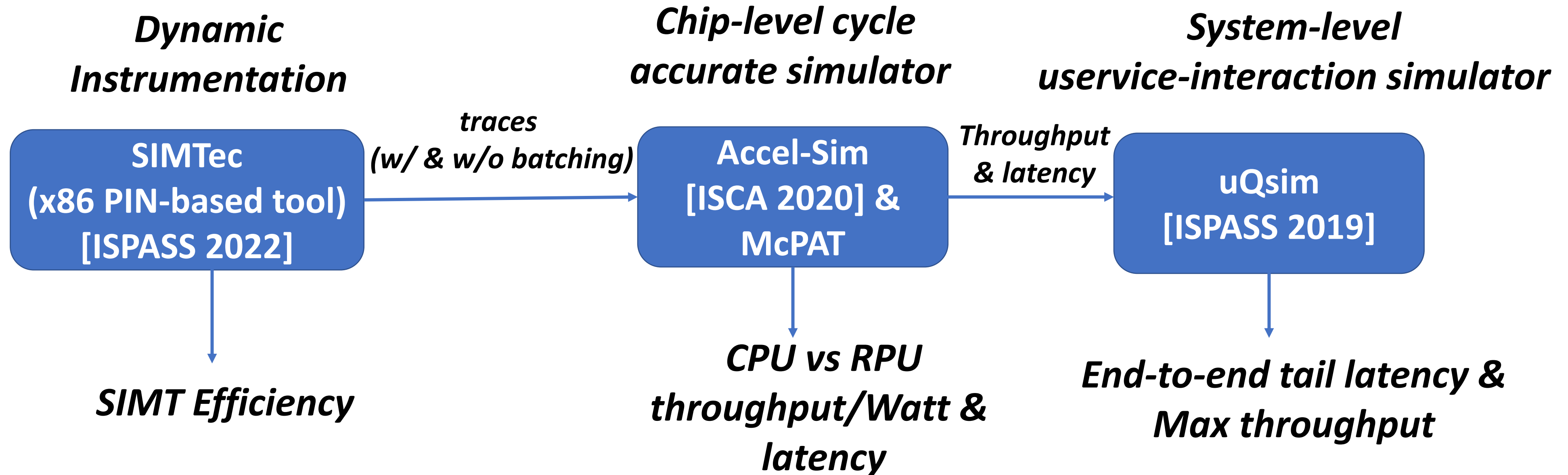
batch size (n) = 8-32
SIMT Efficiency=92%
data locality ratio =75%
Amortized factors = 50-90%
Larger L1/L2 MCUs Active mask etc.

→ an anticipated 2-10x energy efficiency gain can be achieved with RPU vs CPU

CPU Dynamic Energy Breakdown



Experimental Setup



Workloads: Social Network Microservices

Microsuite [IISWC 2018], DeathStarBench [ASPLOS 2020] and In-house benchmarks

Libraries: c++ stdlib, Intel MKL, OpenSSL, FLANN, Pthread, zlib, protobuf, gRPC and MLPack, ...