PURDUE
UNIVERSITY

Scalable and Energy-Efficient SIMT Systems for Deep Learning and Data Center Microservices

Mahmoud Khairy

PhD Candidate – Final Examination

abdallm@purdue.edu
https://mkhairy.github.io/

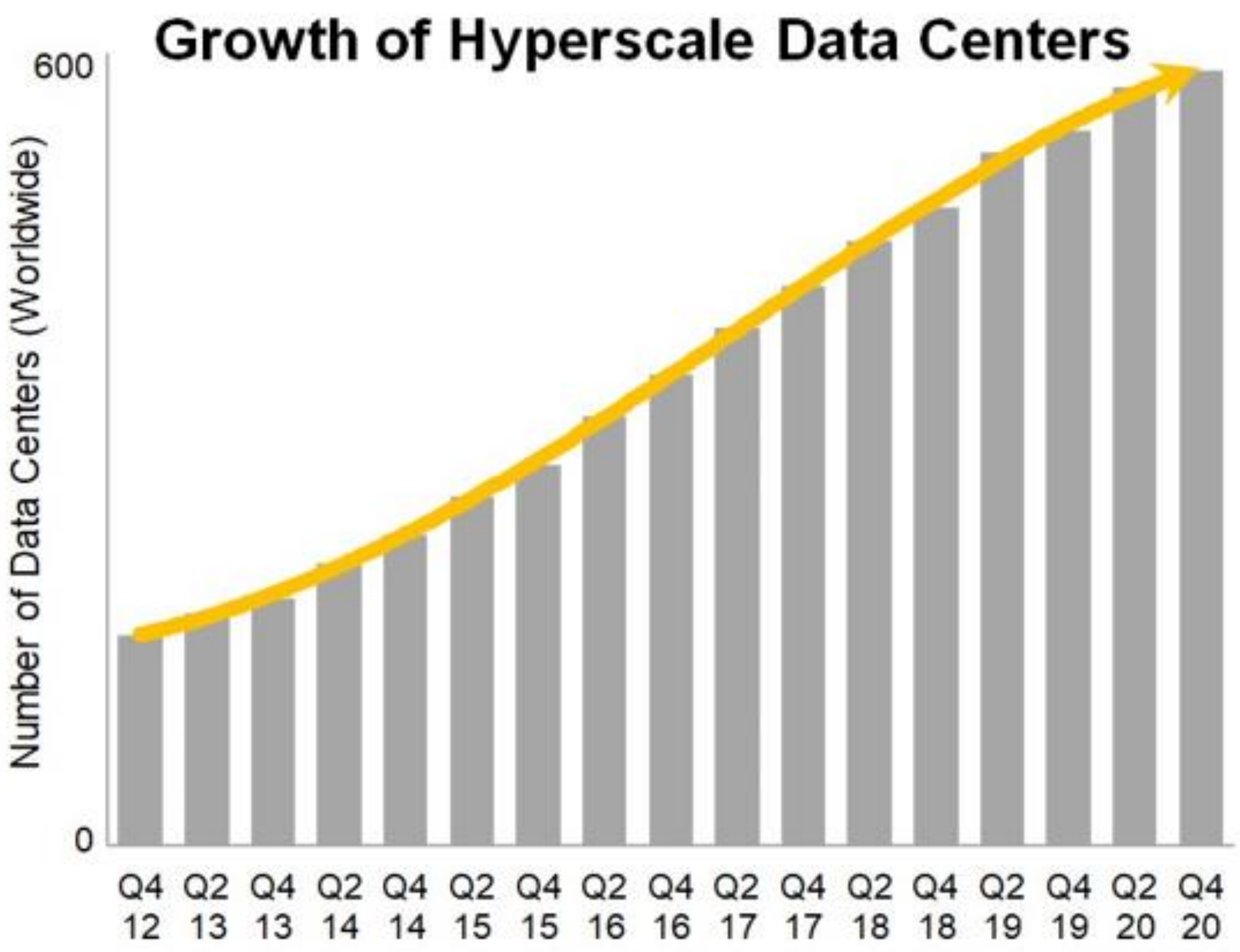
6/1/2022

Agenda

- Motivation and Thesis Summery (*5 mins*)
- LADM: Transparent Multi-GPU Scaling (*7 mins*)
- Accel-Sim: An Extensible GPU simulation framework (*5 mins*)
- RPU: A SIMT System for Data Center Microservices (*25 mins*)
    - Overview & Key Observations
    - RPU Hardware & Software Stack
    - Experimental Setup & Results
- Conclusions & Future Work (*3 mins*)
- Q&A (15+ mins)

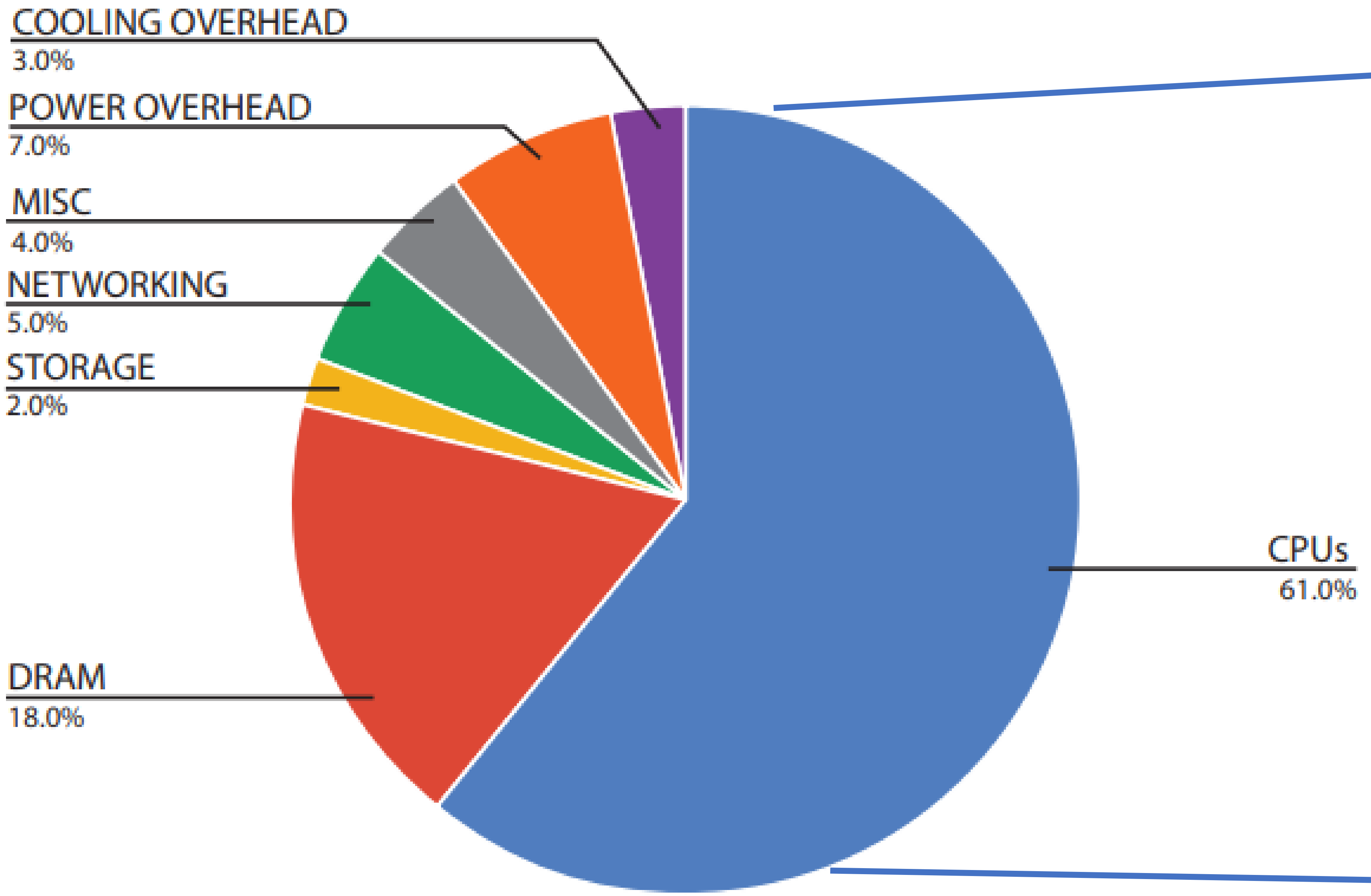
Growth of Hyperscale Data Centers

- The growth of hyperscale data centers has steadily increased in the last decade
- The next era of IoT and AI
- Challenges:
  - Slowing growth of Moore's law
  - High power consumption
  - Large carbon footprint
  - By 2030, the data centers will consume 10% of the total electricity demand

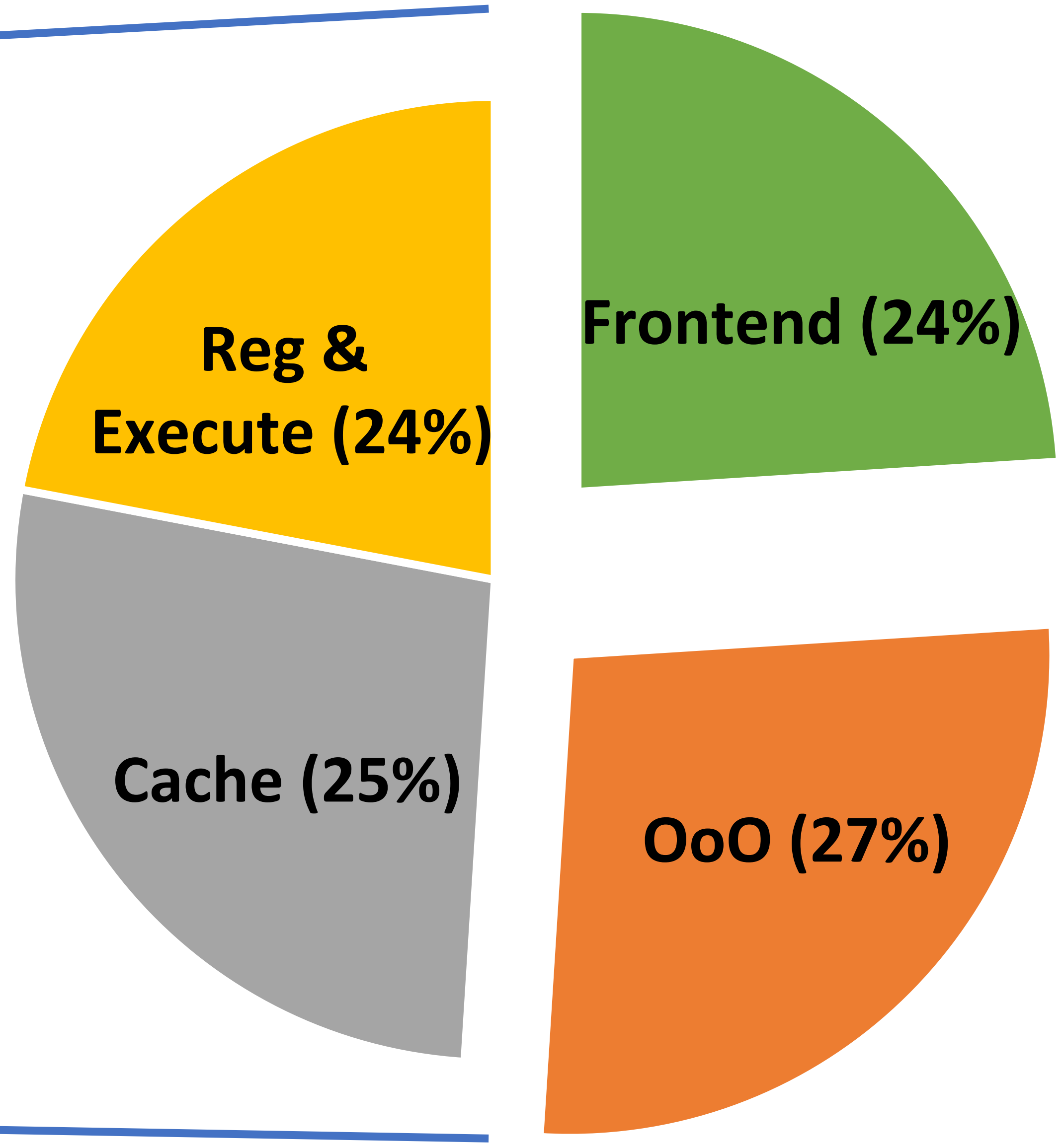


https://www.datacenterknowledge.com/cloud/analysts-there-are-now-more-500-hyperscale-data-centers-world
https://www.nature.com/articles/d41586-018-06610-y

Datacenter Power Breakdown



Datacenter Power Breakdown (from Google)



CPU Power Breakdown (Intel Skylake)

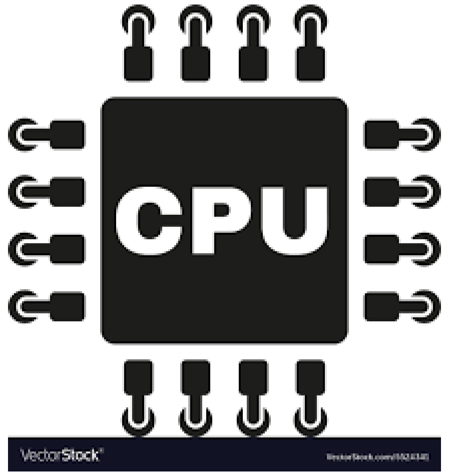
30% of datacenter power is consumed in CPU's instruction supply (frontend & OoO)

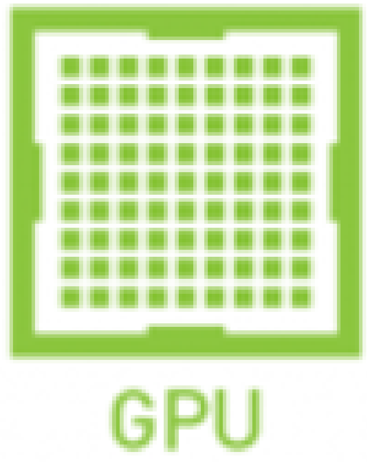
[1] Barroso, Luiz André, and Urs Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines." *Synthesis lectures on computer architecture*. 2018
[2] Haj-Yihia, Jawad, et al. "Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems." *ACM TACO* 2016
[3] Powell, Michael D., et al. "CAMP: A technique to estimate per-structure power at run-time using a few simple parameters." *HPCA 2009*

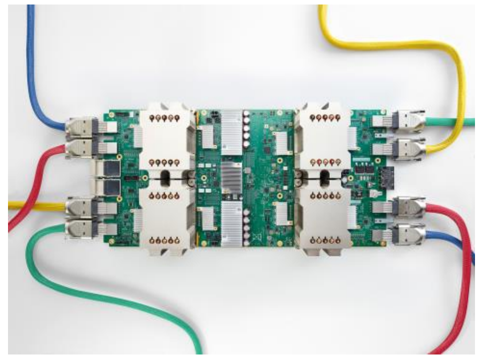
Datacenter Paradigm Shifts (HW-SW Codesign)

Software

Hardware

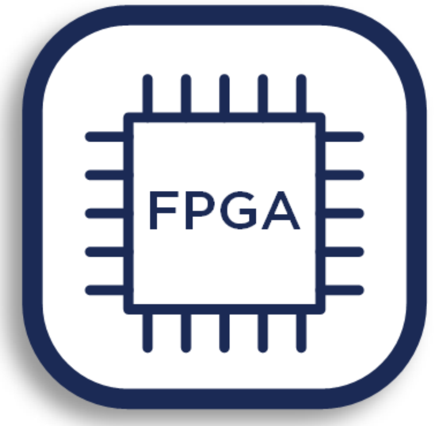






TPU

VCU

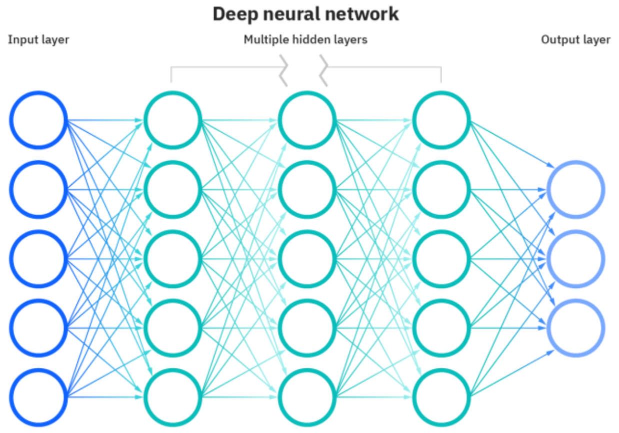


……

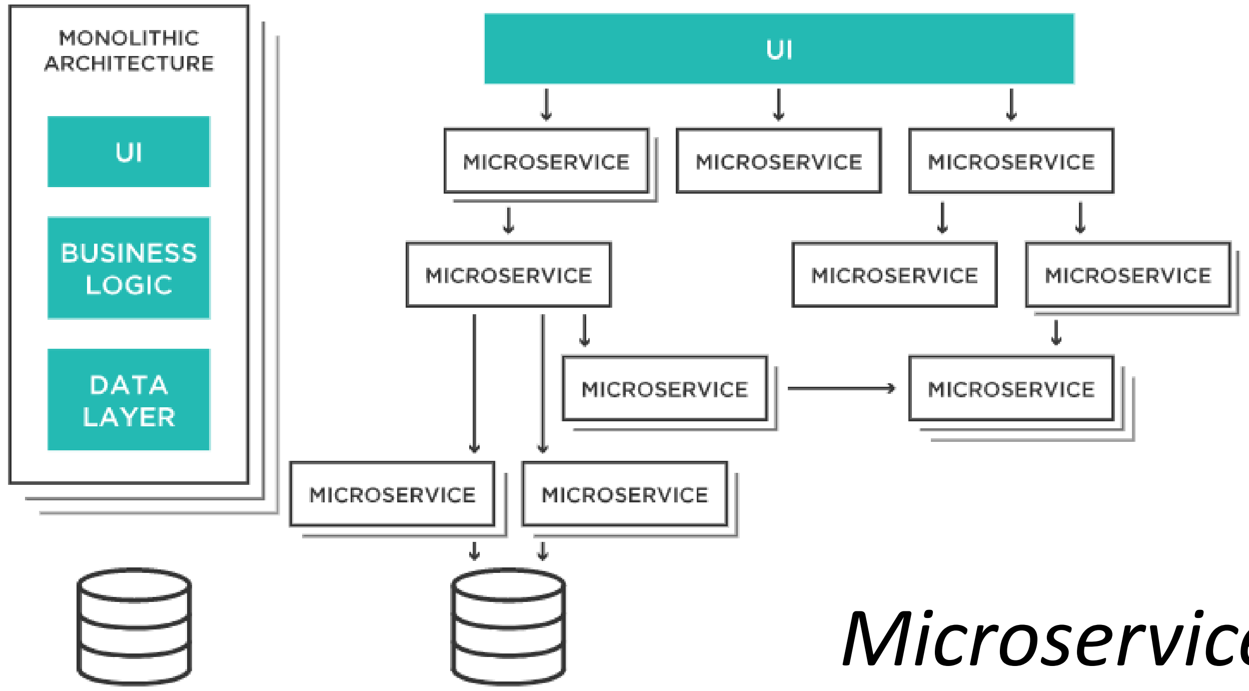
Accelerators

Datacenter Paradigm Shifts (HW-SW Codesign)

Software



Deep Learning



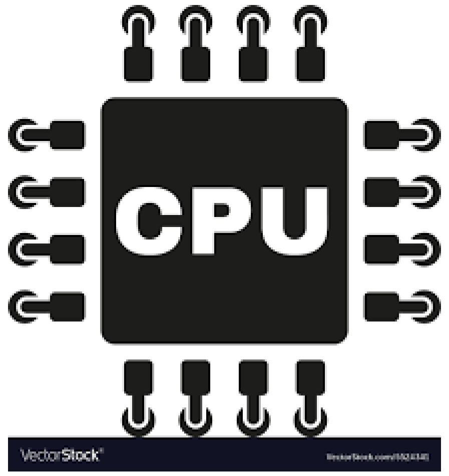
mongoDB

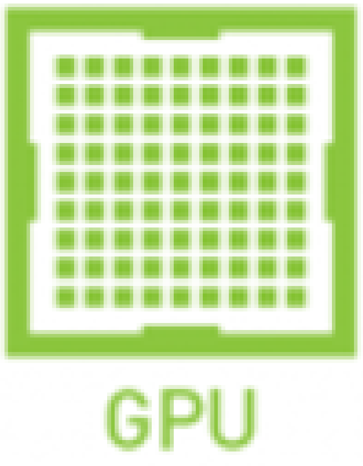
MEMCACHED

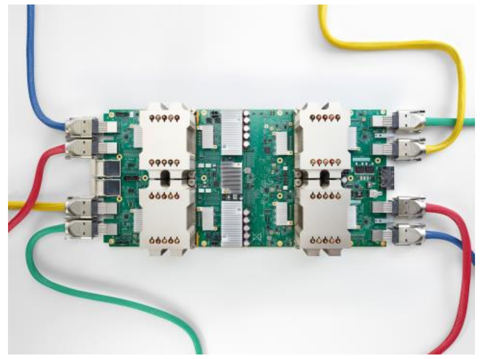
gRPC

Microservices

Hardware

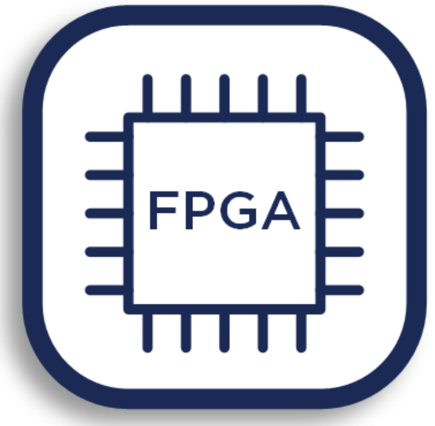






TPU

VCU



......

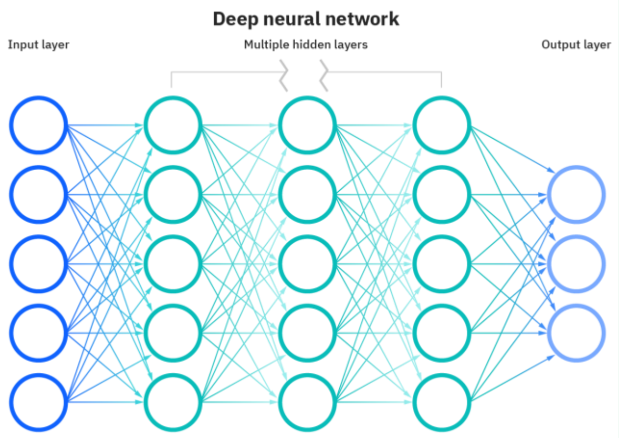
Accelerators

My Ph.D. Thesis Contributions

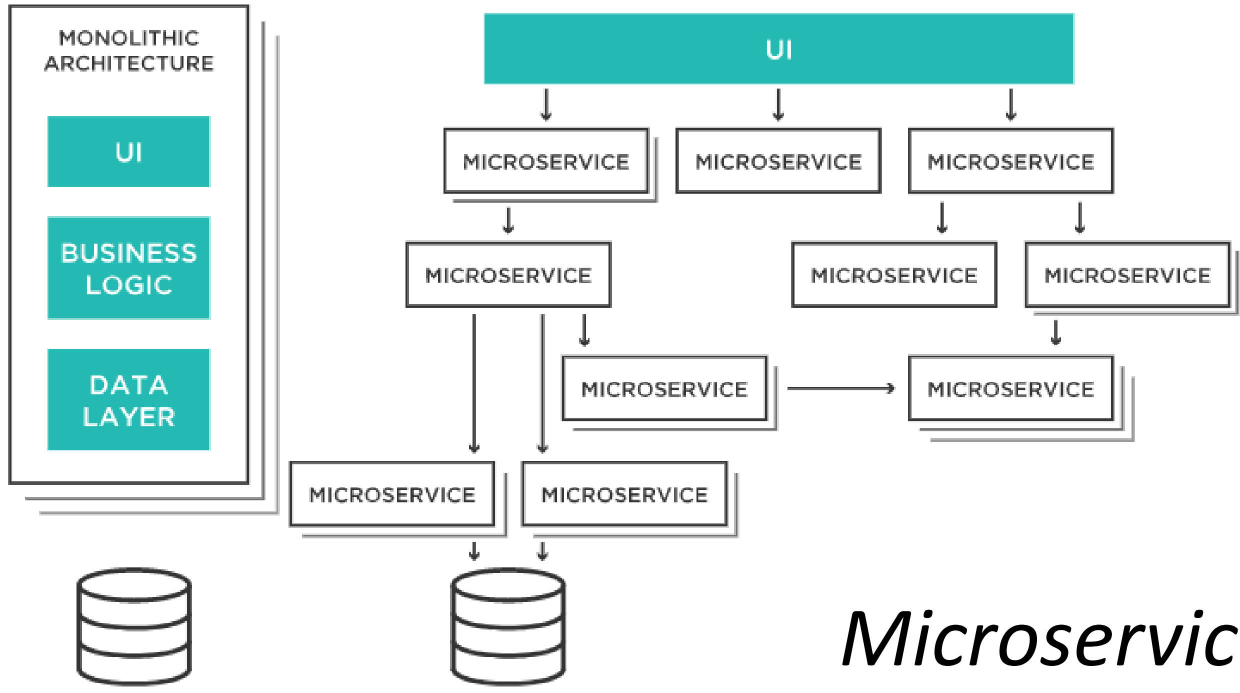
Energy-Efficient uService Processing
→ RPU [under review]

Software

Efficient Multi-GPU
→ LADM [MICRO'20]
→ [SigArch'21]



Deep Learning



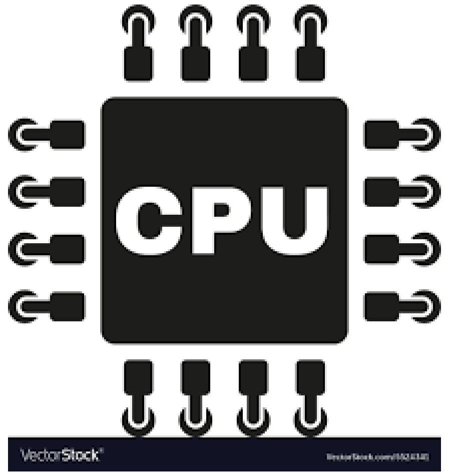
mongoDB

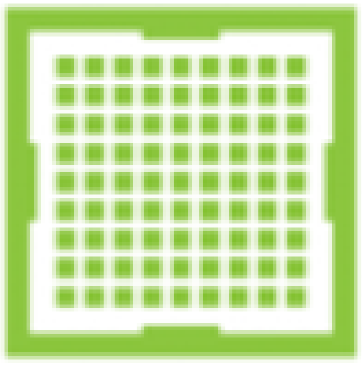
MEMCACHED

gRPC

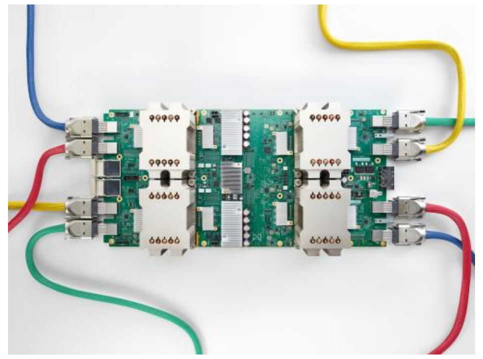
Microservices

Hardware



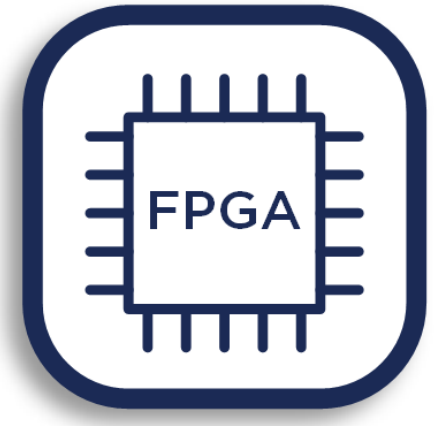


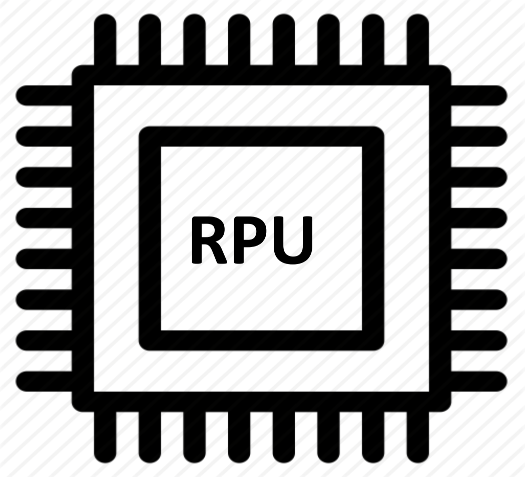
GPU



TPU

VCU





Accurate and Extensible Simulator
→ Accel-Sim [ISCA'20] [SIGMETRICS'18]

Accelerators

Thesis Statement (Verbatim)

- SIMT-based accelerators, like GPUs and my proposed RPUs, are promising solutions to achieve significant *energy efficiency* while still preserving *programmability* in the twilight of Moore’s Law.
- I propose three approaches to build next-generation scalable and energy-efficient SIMT systems:

  (1) Detect and optimize for each type of locality exist in the DL and HPC workloads to overcome NUMA effects,

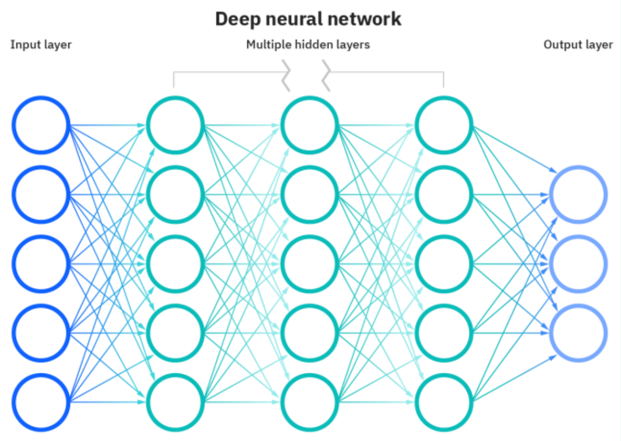
  (2) Exploit microservices execution similarity and eliminate redundancy to improve data center energy efficiency, and

  (3) Build extensible and validated SIMT simulation tools to keep-up with industrial changes.

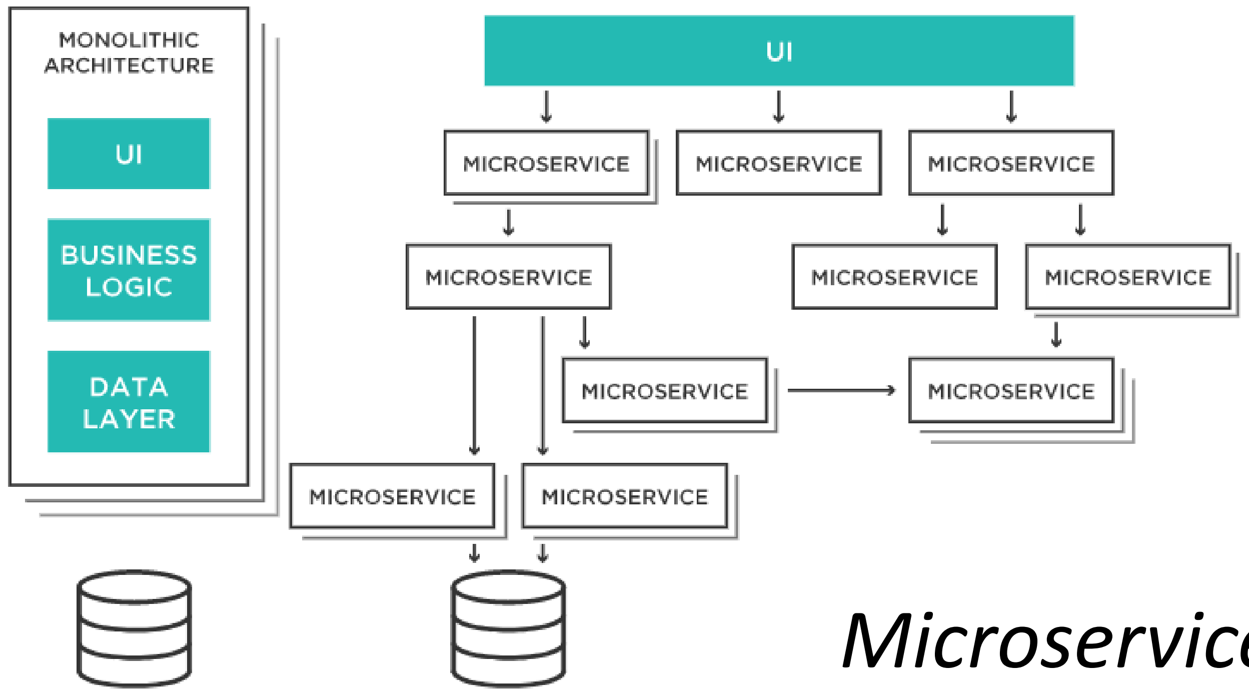
My Ph.D. Thesis Contributions

Software

Efficient Multi-GPU
→ LADM [MICRO'20]
→ [SigArch'21]



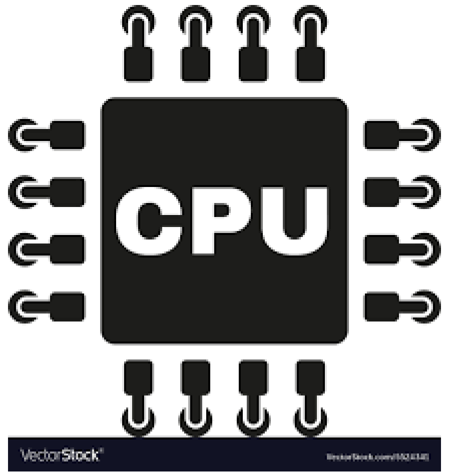
Deep Learning

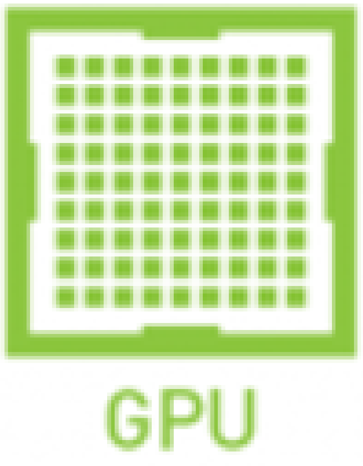


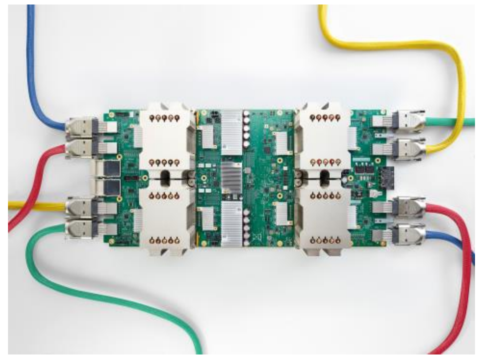
mongoDB. MEMCACHED gRPC

Microservices

Hardware

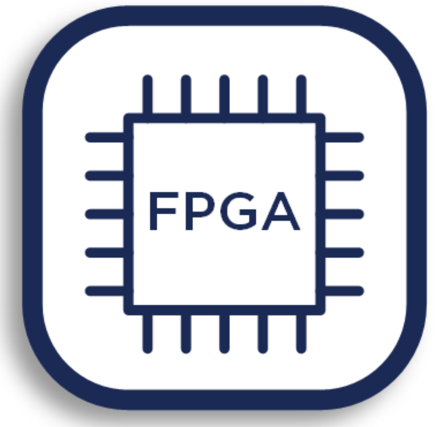


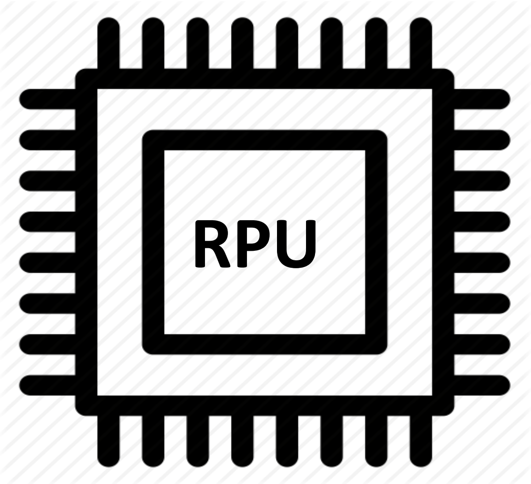




TPU

VCU



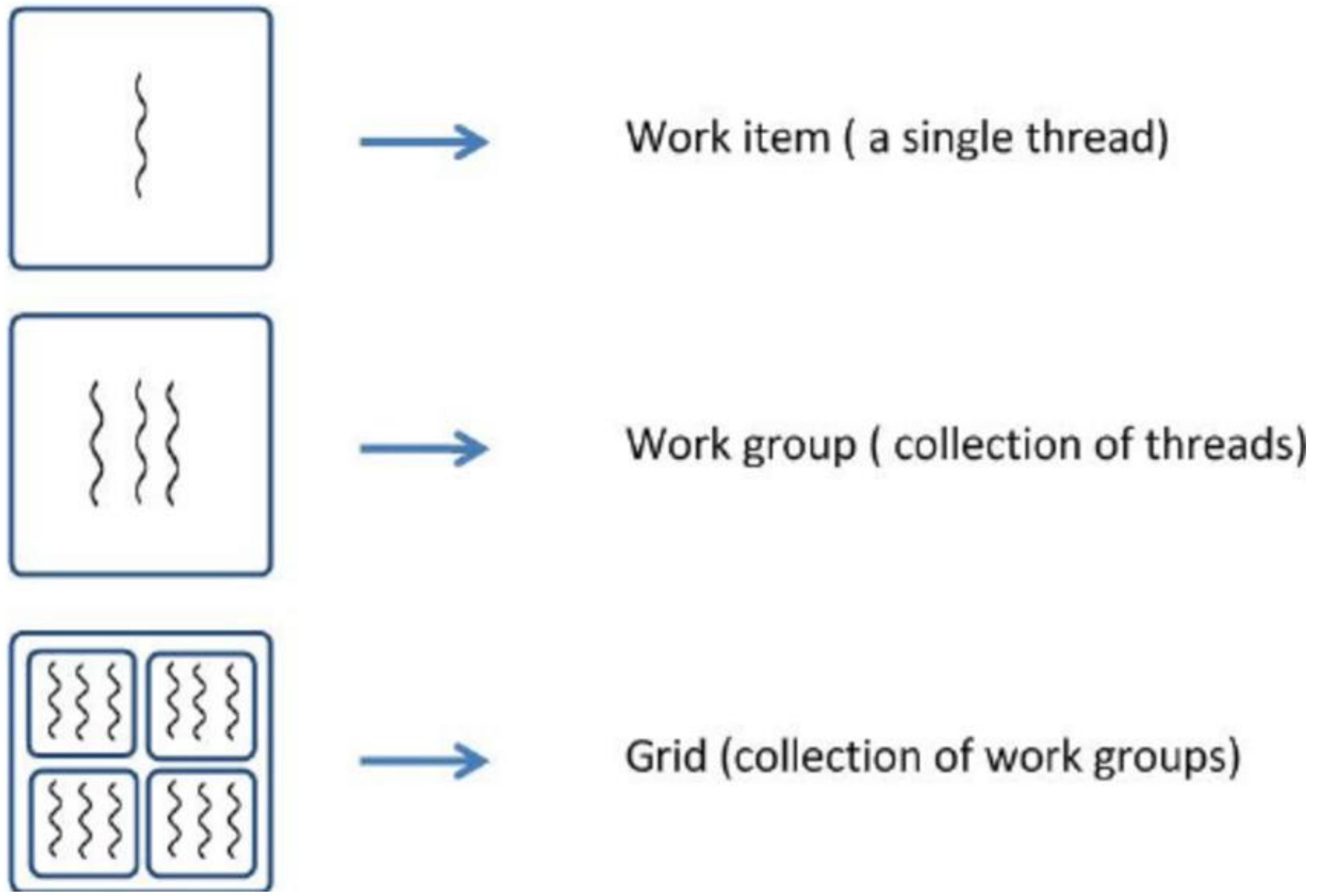


Accurate and Extensible Simulator
→ Accel-Sim [ISCA'20] [SIGMETRICS'18]

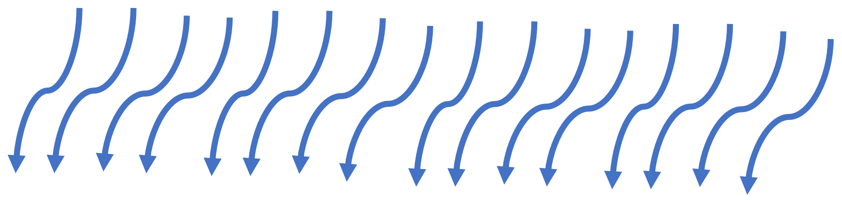
Accelerators

Single Instruction Multiple Thread (SIMT)

- GPGPU Programming Model
  - Single Program Multiple Data
  - Express parallelism in terms of fine-grain hierarchal threads

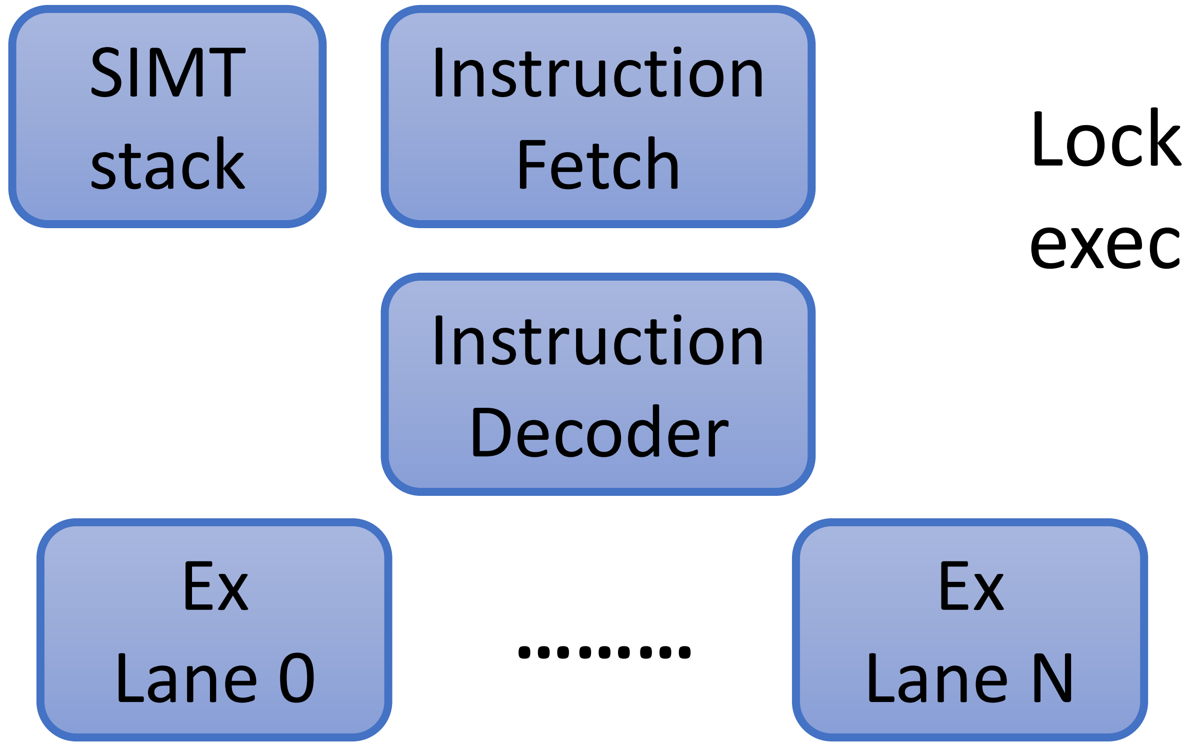

Work item (a single thread)
Work group (collection of threads)
Grid (collection of work groups)

- GPU Hardware:
  - Aggregate every 32/64 threads in a *warp*



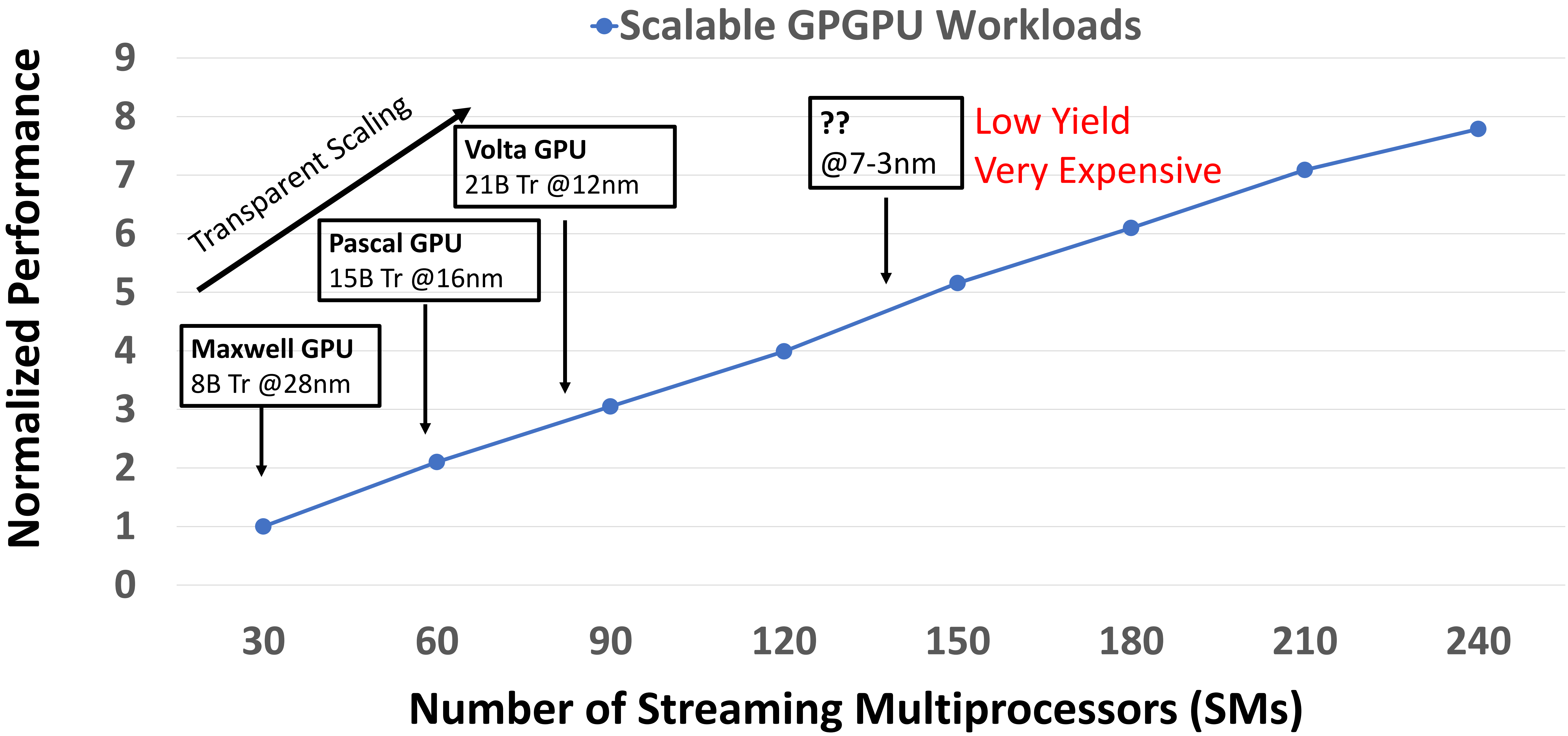
Warp of Threads

- SIMT = One Instruction, Multiple Threads



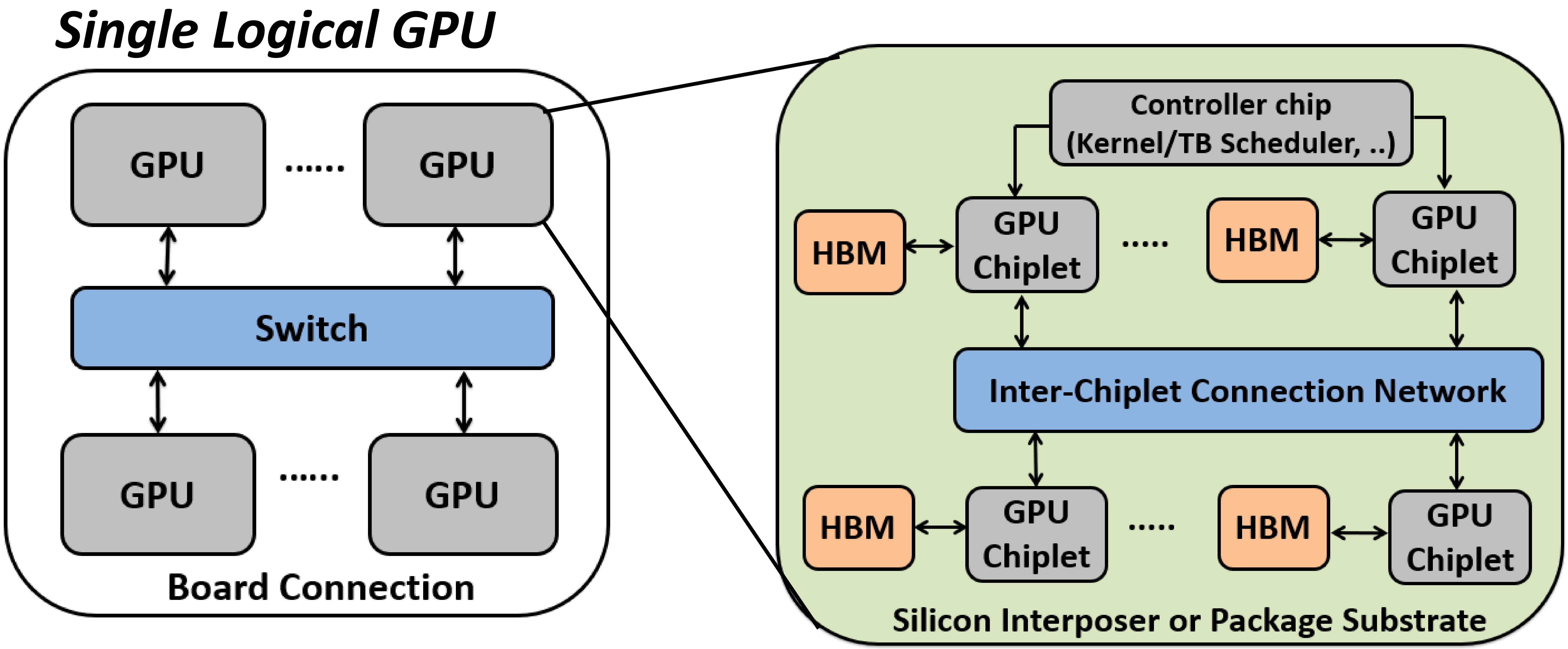
Lock-step execution

GPU Performance Scalability is at Risk



Scaling all the GPU resources: Increasing SMs, memory bandwidth and interconnection bandwidth.

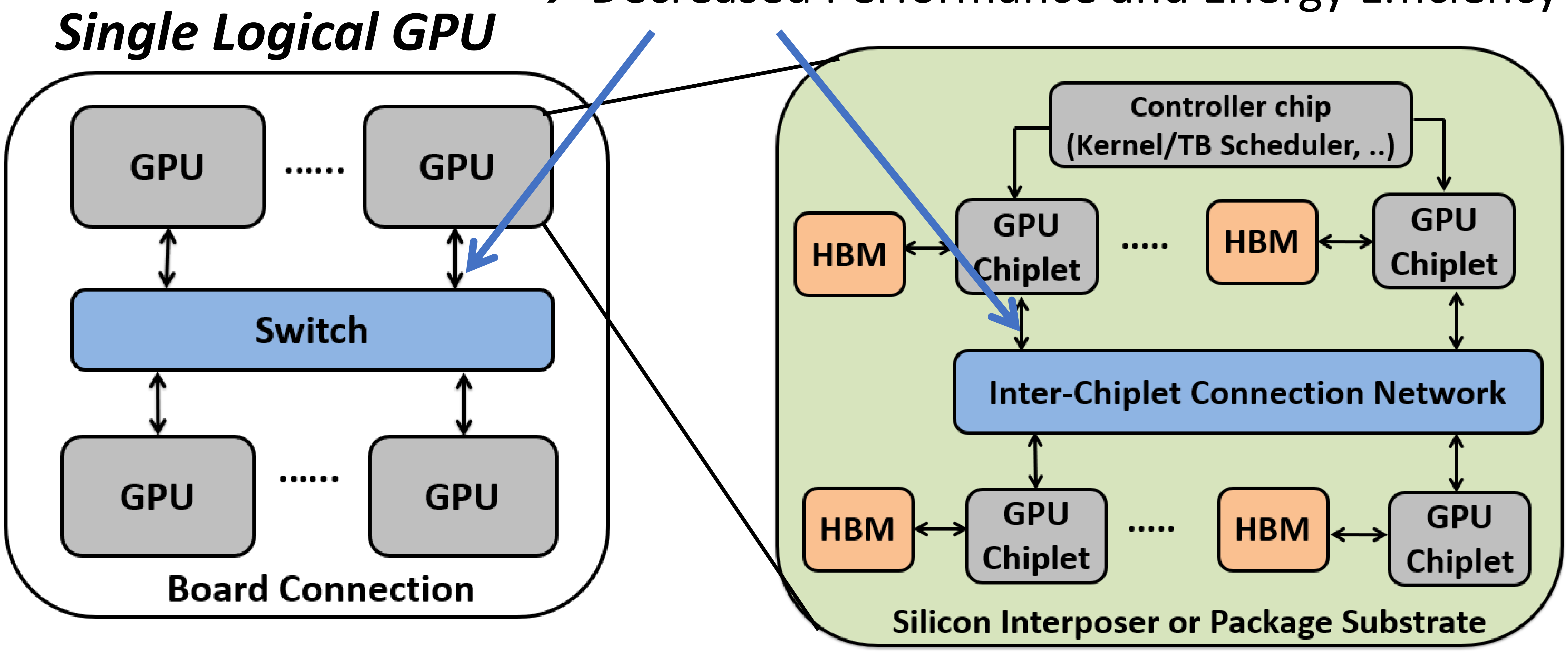
Hierarchical Multi-GPU Multi-Chiplet



Arunkumar, et al. "MCM-GPU: Multi-chip-module GPUs for continued performance scalability” ISCA 2017
Milic, et al. "Beyond the socket: NUMA-aware GPUs." MICRO 2017
Ren, Xiaowei, et al. "Hmg: Extending cache coherence protocols across modern hierarchical multi-gpu systems." HPCA 2020

Hierarchical Multi-GPU Multi-Chiplet

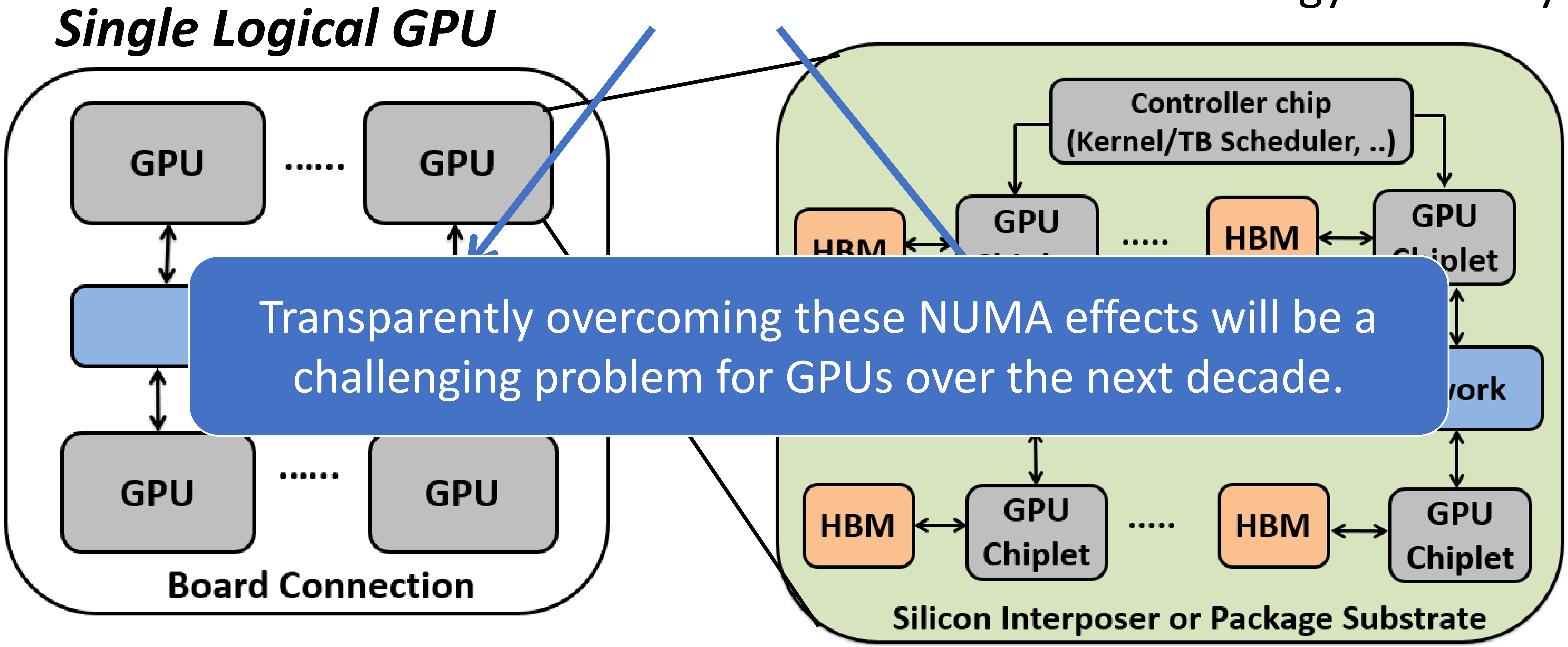
Non-Uniform Memory Access (NUMA)
→ Decreased Performance and Energy Efficiency



Arunkumar, et al. "MCM-GPU: Multi-chip-module GPUs for continued performance scalability" ISCA 2017
Milic, et al. "Beyond the socket: NUMA-aware GPUs." MICRO 2017
Ren, Xiaowei, et al. "Hmg: Extending cache coherence protocols across modern hierarchical multi-gpu systems." HPCA 2020

Hierarchical Multi-GPU Multi-Chiplet

Non-Uniform Memory Access (NUMA)
→ Decreased Performance and Energy Efficiency

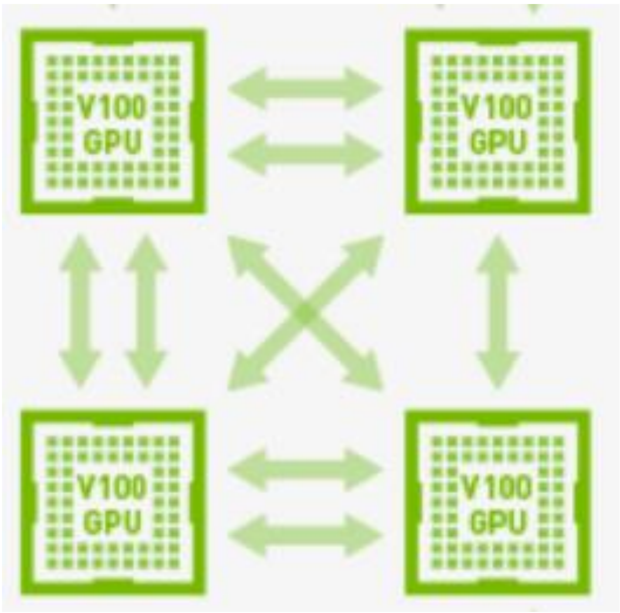


Transparently overcoming these NUMA effects will be a challenging problem for GPUs over the next decade.

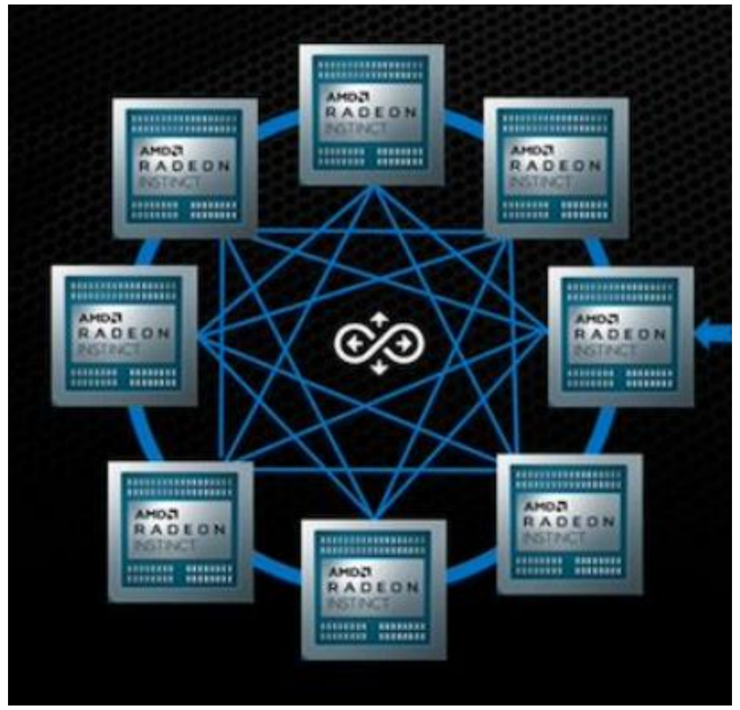
Arunkumar, et al. "MCM-GPU: Multi-chip-module GPUs for continued performance scalability" ISCA 2017
Milic, et al. "Beyond the socket: NUMA-aware GPUs." MICRO 2017
Ren, Xiaowei, et al. "Hmg: Extending cache coherence protocols across modern hierarchical multi-gpu systems." HPCA 2020

NUMA-GPU is already out there

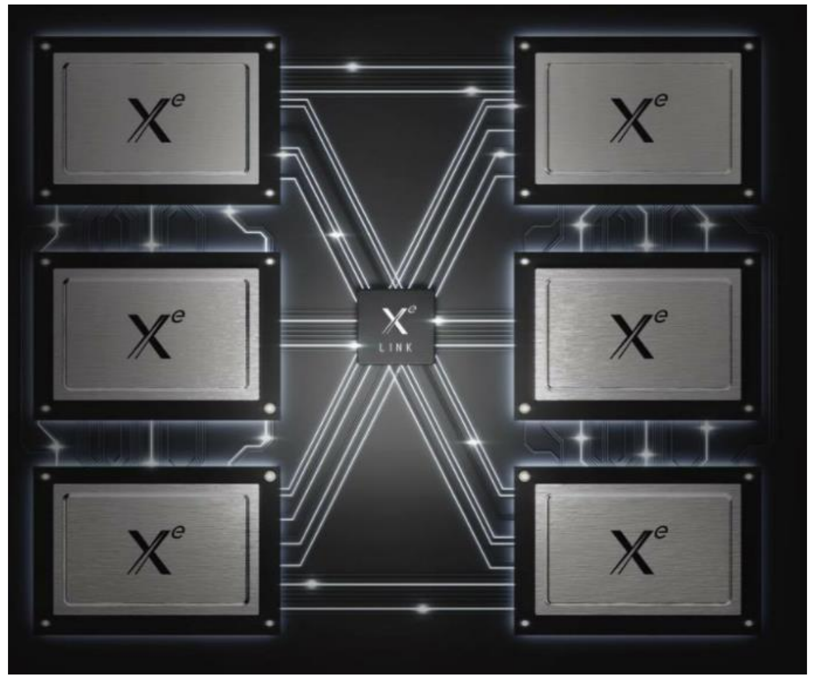
Socket-based Multi-GPU:



Nvidia DGX server
Nvlink (4-16 GPUs)



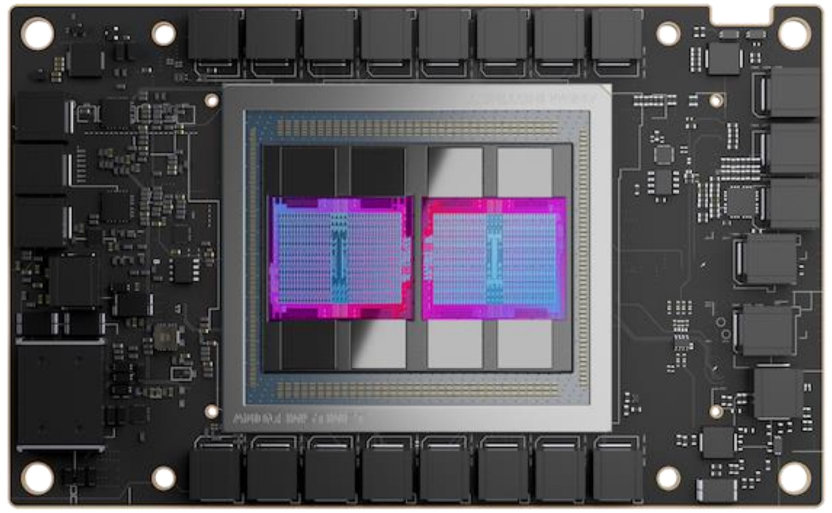
AMD GPU server
Infinity link (4-8 GPUs)



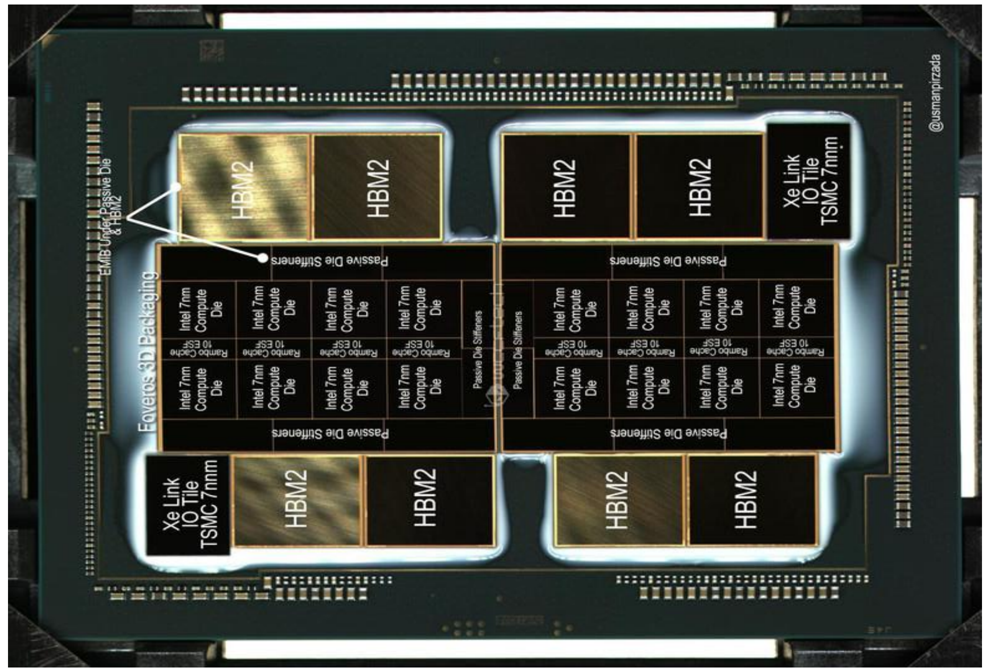
Intel Xe GPU server
Xe link (6 GPUs)

Multi-Chiplet GPU:

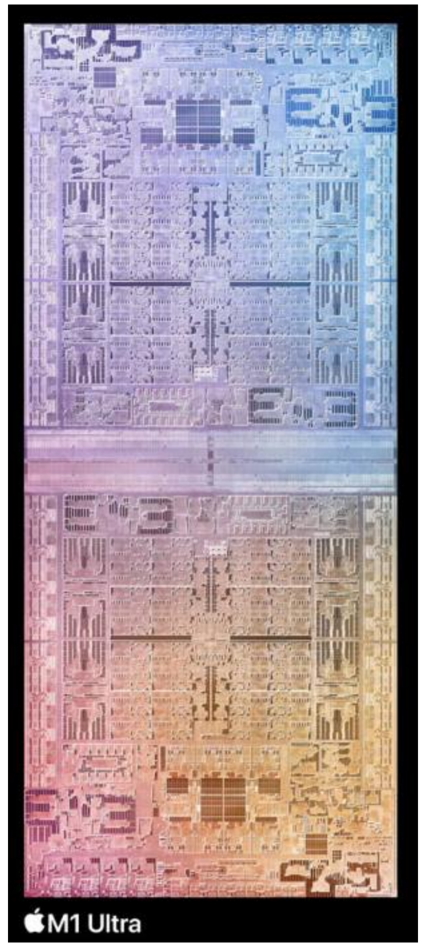
Nvidia Ampere (2 virtual GPU clusters)



AMD Instinct MI200



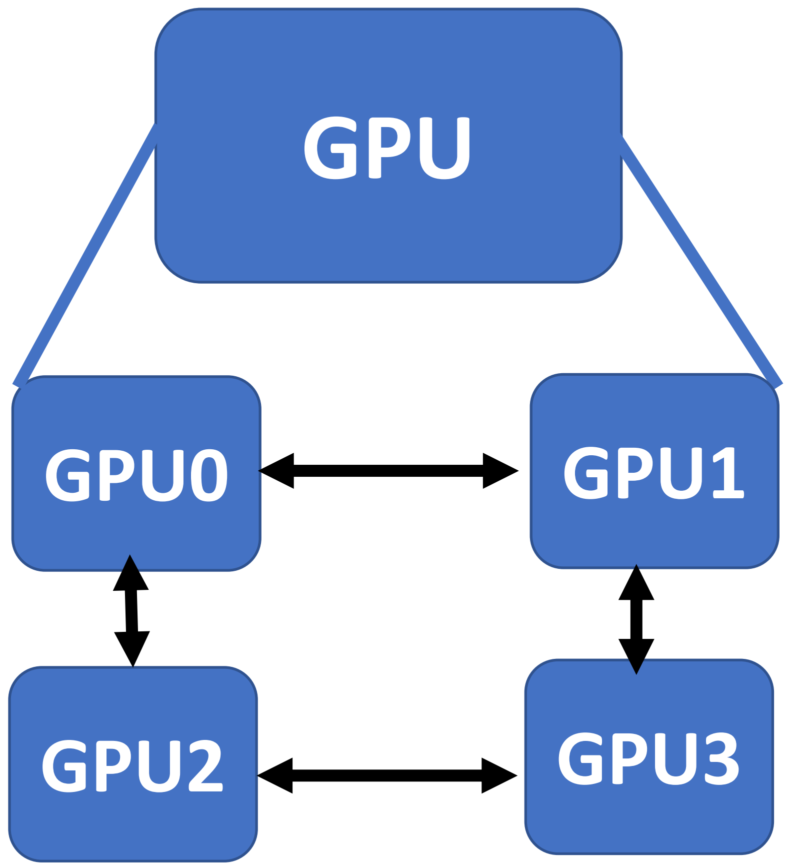
Intel Ponte Vecchio (8 tiles per GPU)



Apple M1 Ultra

NUMA Impact: Performance Loss

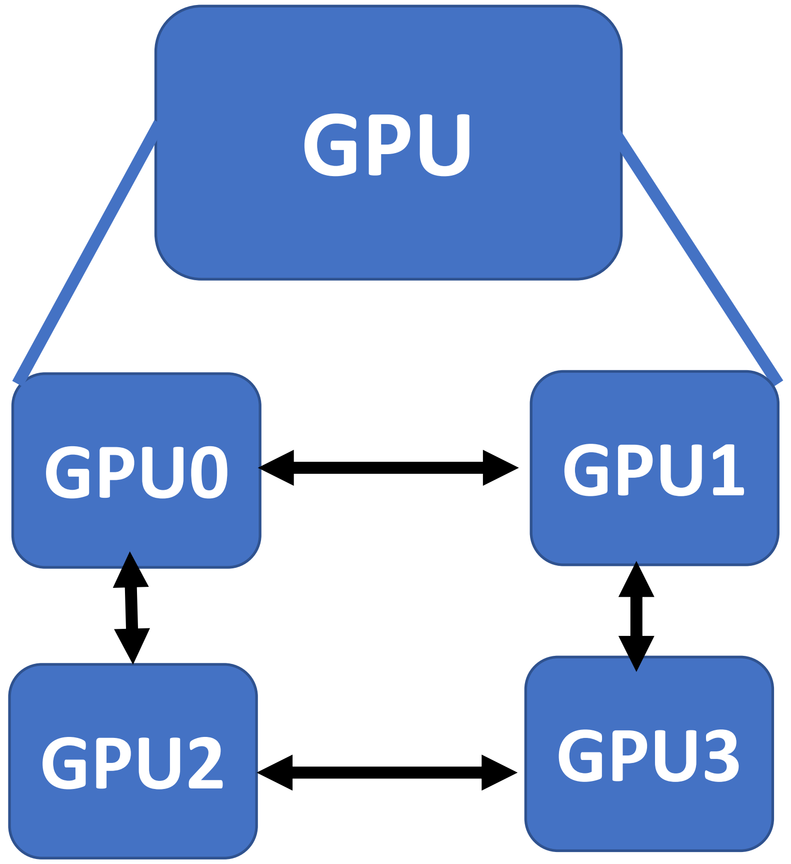
Monolithic GPU Chip
(256 SMs + 2.8 TB/sec BW)



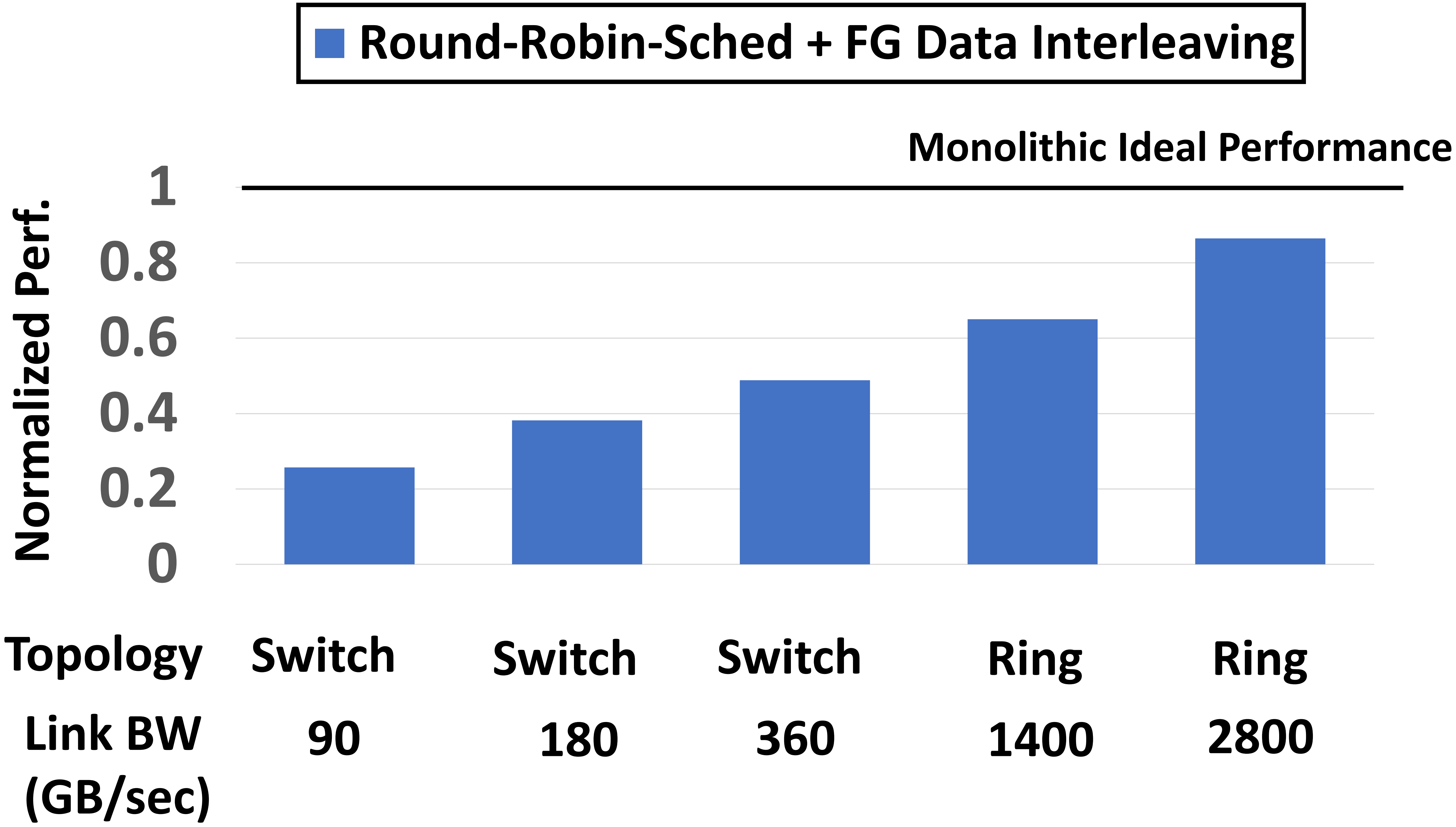
4 GPUs with each of
(64 SMs + 700 GB/sec BW)

NUMA Impact: Performance Loss

Monolithic GPU Chip
(256 SMs + 2.8 TB/sec BW)

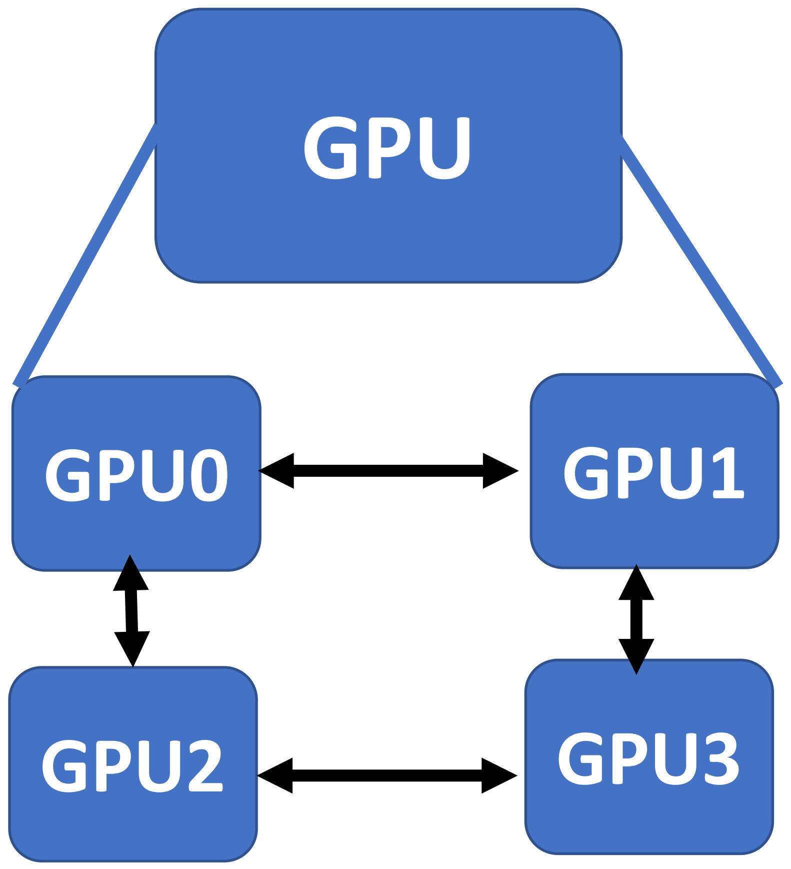


4 GPUs with each of
(64 SMs + 700 GB/sec BW)

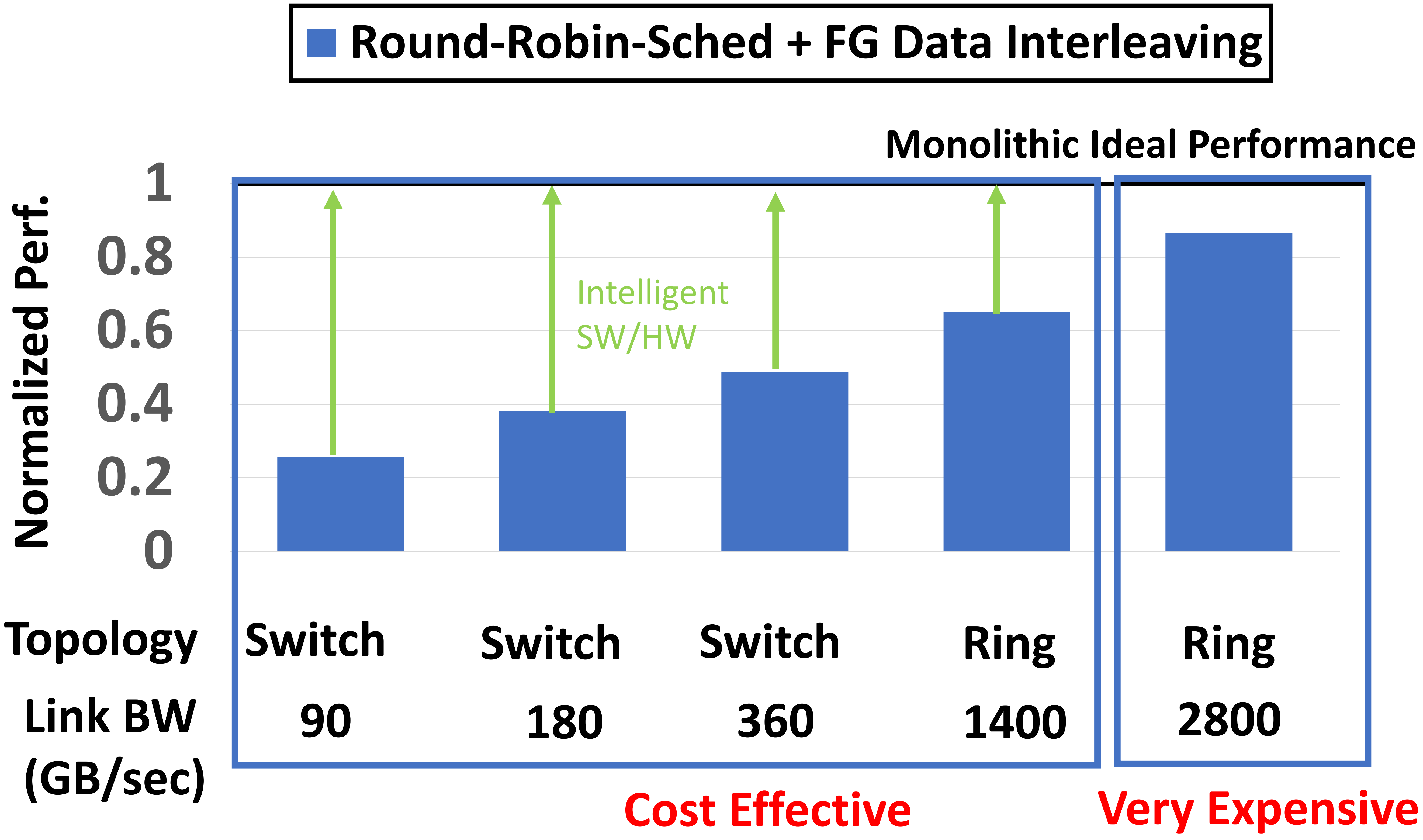


NUMA Impact: Performance Loss

Monolithic GPU Chip
(256 SMs + 2.8 TB/sec BW)

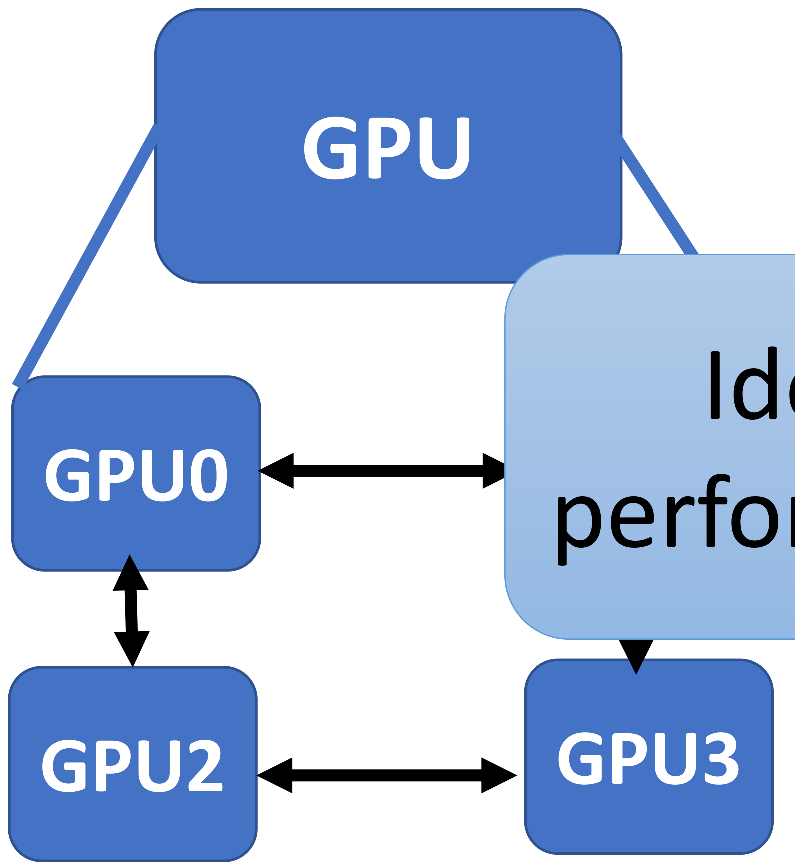


4 GPUs with each of
(64 SMs + 700 GB/sec BW)



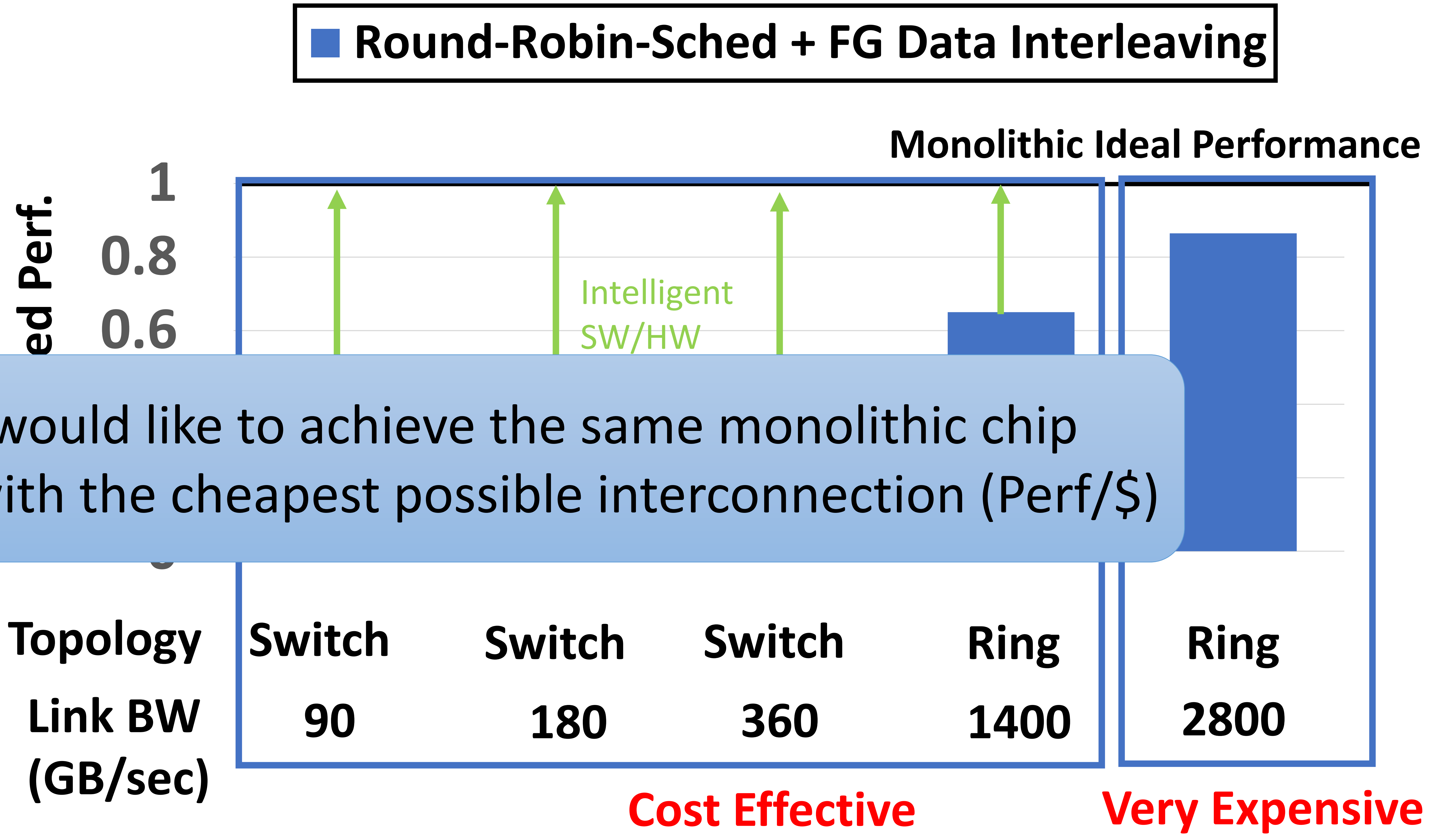
NUMA Impact: Performance Loss

Monolithic GPU Chip
(256 SMs + 2.8 TB/sec BW)



4 GPUs with each of
(64 SMs + 700 GB/sec BW)

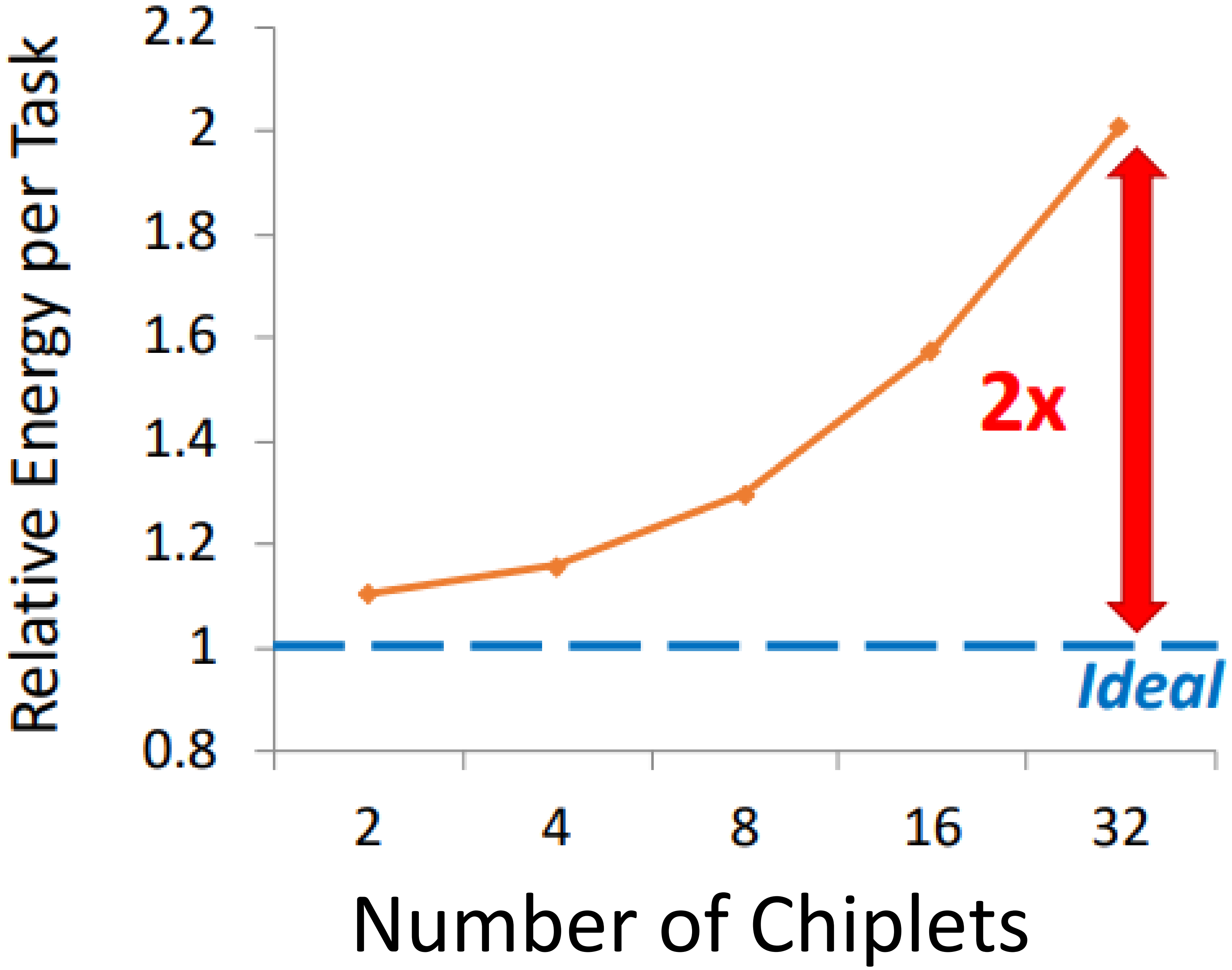
Ideally, we would like to achieve the same monolithic chip performance with the cheapest possible interconnection (Perf/$)



NUMA Impact: Decreased Energy Efficiency (Perf/Watt)

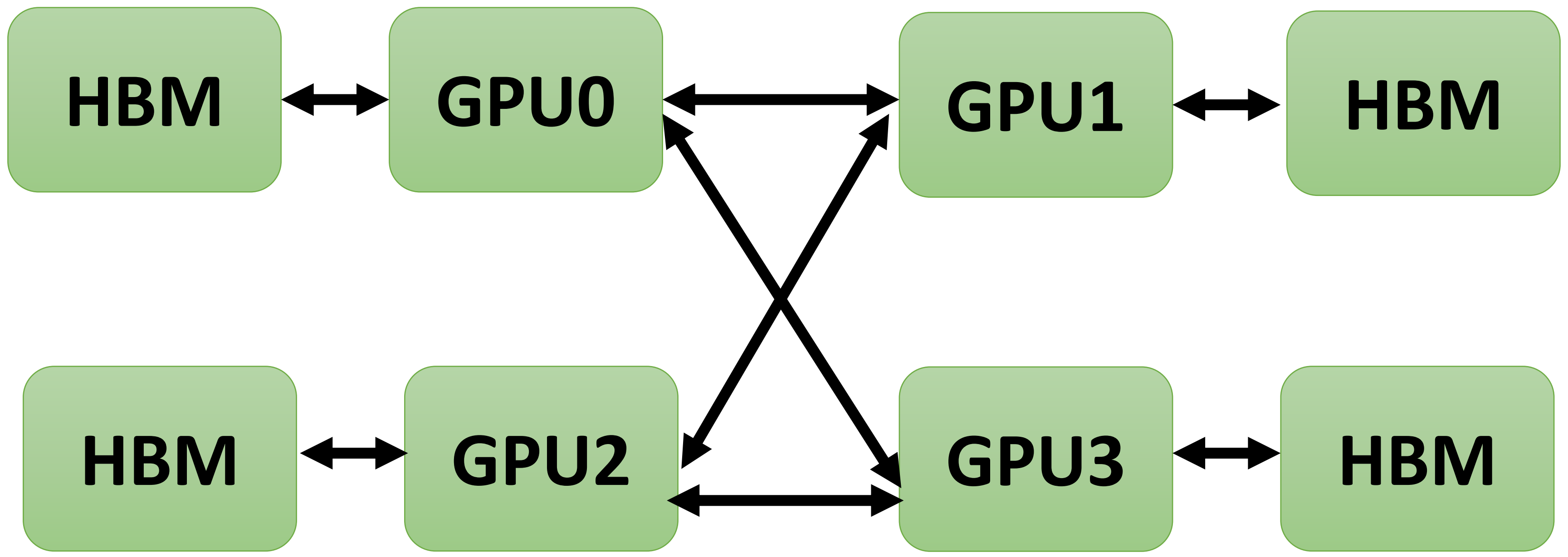
- Energy cost per task could double

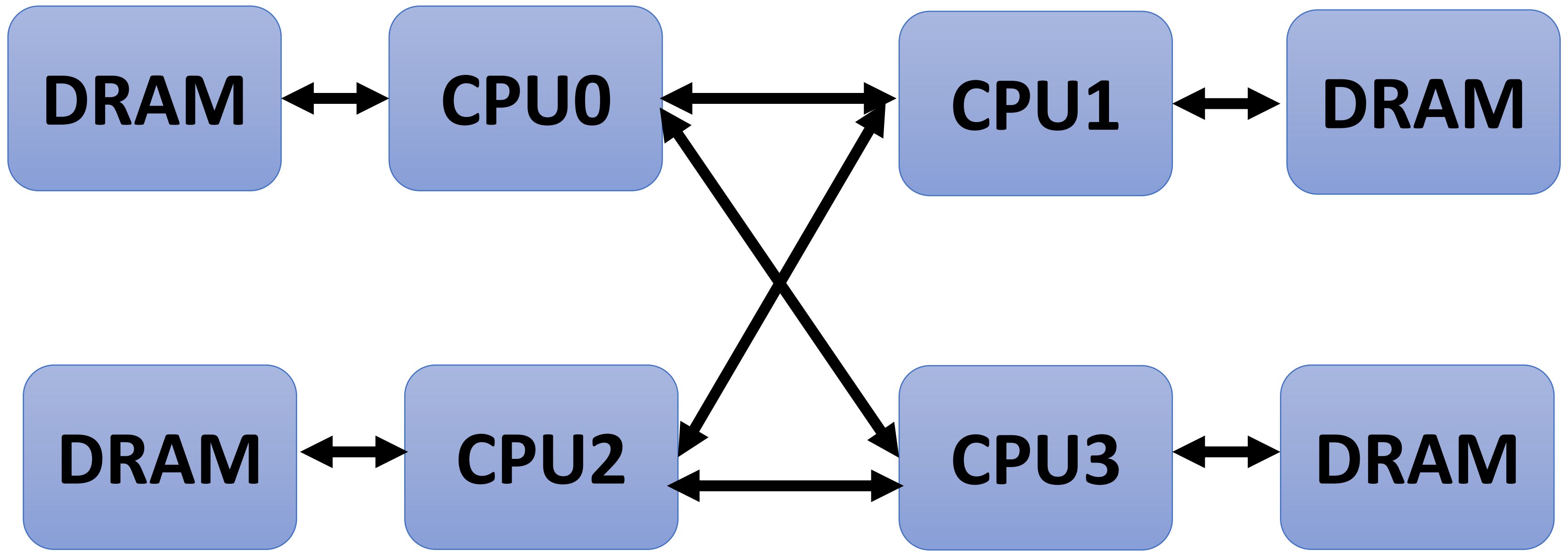
- 50% of the future GPU power is anticipated to be consumed on off-chiplet traffic



Arunkumar, et al. "Understanding the future of energy efficiency in multi-module gpus." HPCA 2019

Traditional NUMA Solutions

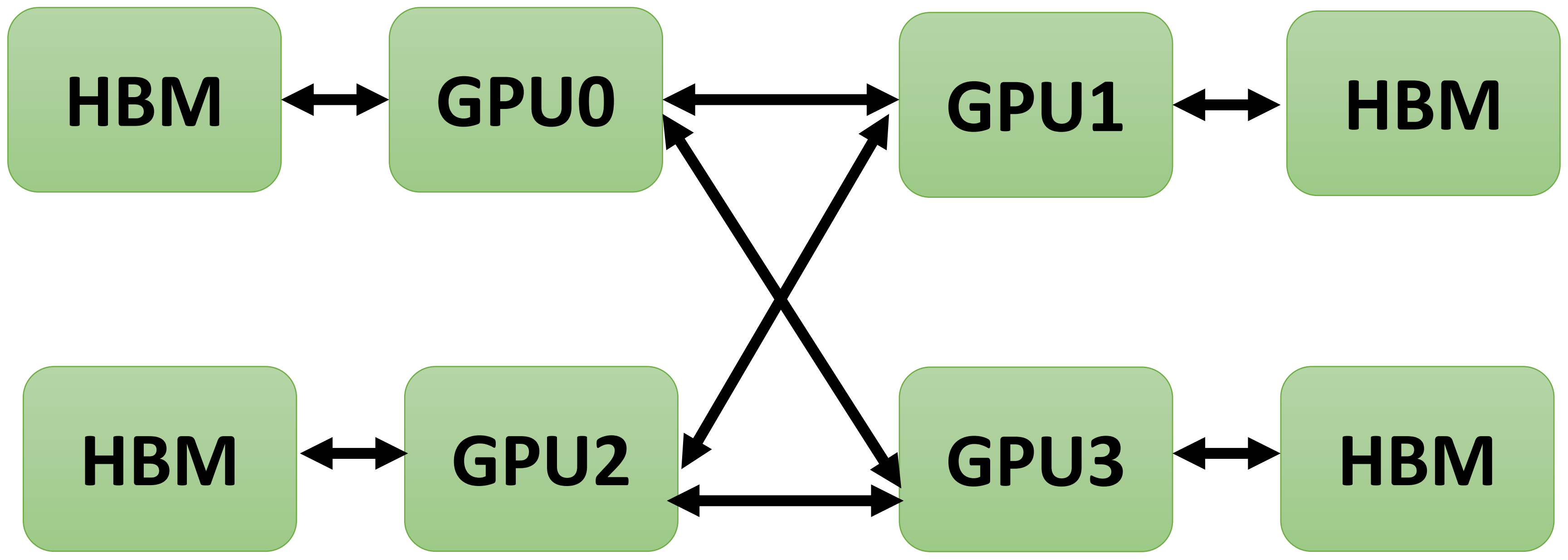


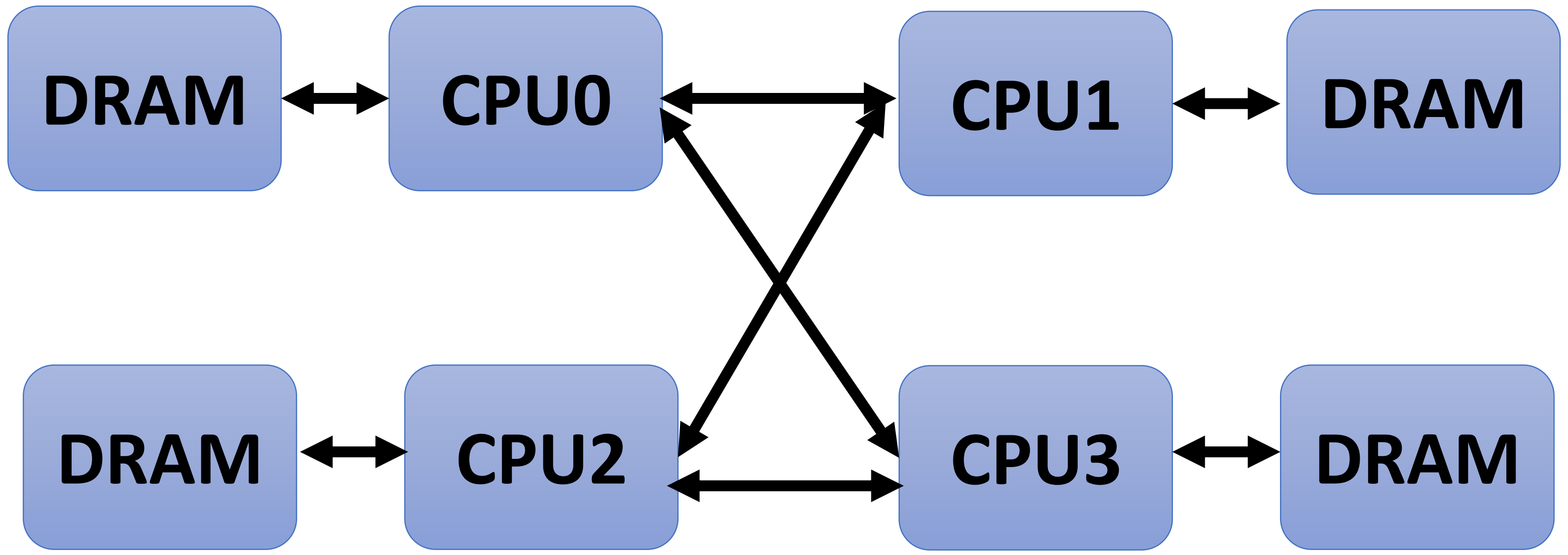


Reactive Solutions:
First-touch page placement
Page migration/duplication
Work redistribution

Zheng et al, “Towards High Performance Paged Memory for GPUs ”, HPCA’16
Young et al., “Combining HW/SW Mechanisms to Improve NUMA Performance of Multi-GPU Systems”, MICRO’18

Traditional NUMA Solutions





Substantial Overhead

Reactive Solutions:

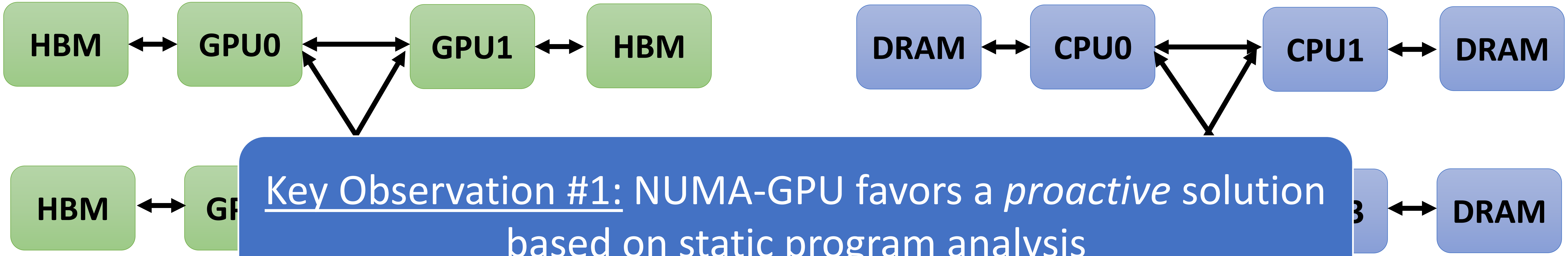
First-touch page placement ➡ PCIe bottleneck, No Context Switching support

Page migration/duplication ➡ Limited GPU Memory capacity

Work redistribution ➡ Massive threads’ context

Zheng et al, “Towards High Performance Paged Memory for GPUs ”, HPCA’16
Young et al., “Combining HW/SW Mechanisms to Improve NUMA Performance of Multi-GPU Systems”, MICRO’18

Traditional NUMA Solutions



Key Observation #1: NUMA-GPU favors a *proactive* solution based on static program analysis

Substantial Overhead

Reactive Solutions:

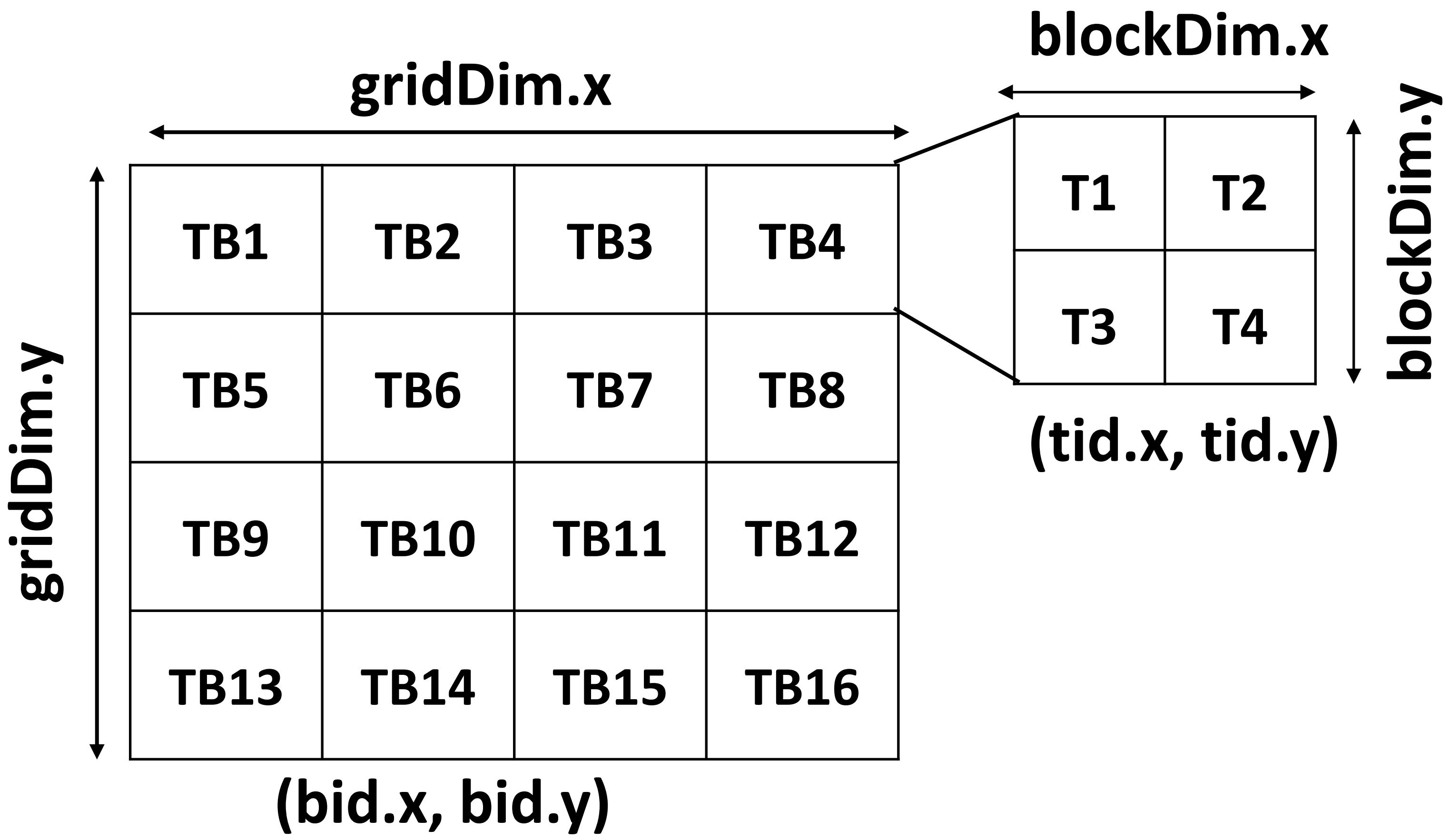
First-touch page placement ➡ PCIe bottleneck, No Context Switching support

Page migration/duplication ➡ Limited GPU Memory capacity

Work redistribution ➡ Massive threads’ context

Zheng et al, “Towards High Performance Paged Memory for GPUs ”, HPCA’16
Young et al., “Combining HW/SW Mechanisms to Improve NUMA Performance of Multi-GPU Systems”, MICRO’18

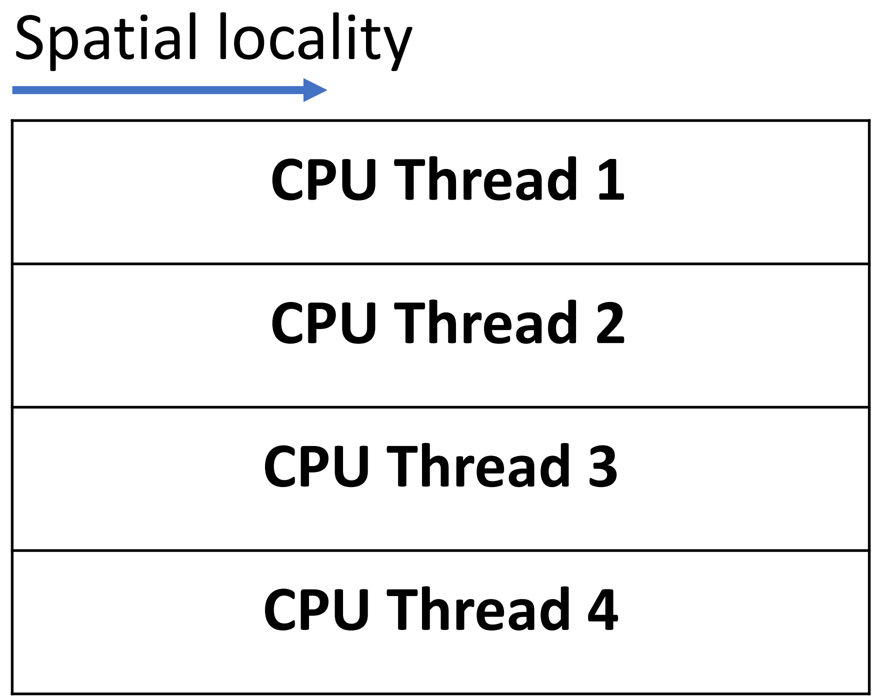
GPU vs CPU Programming Model



TB1	TB2	TB3	TB4
TB5	TB6	TB7	TB8
TB9	TB10	TB11	TB12
TB13	TB14	TB15	TB16

T1	T2
T3	T4

GPU hierarchical fine-grain threads

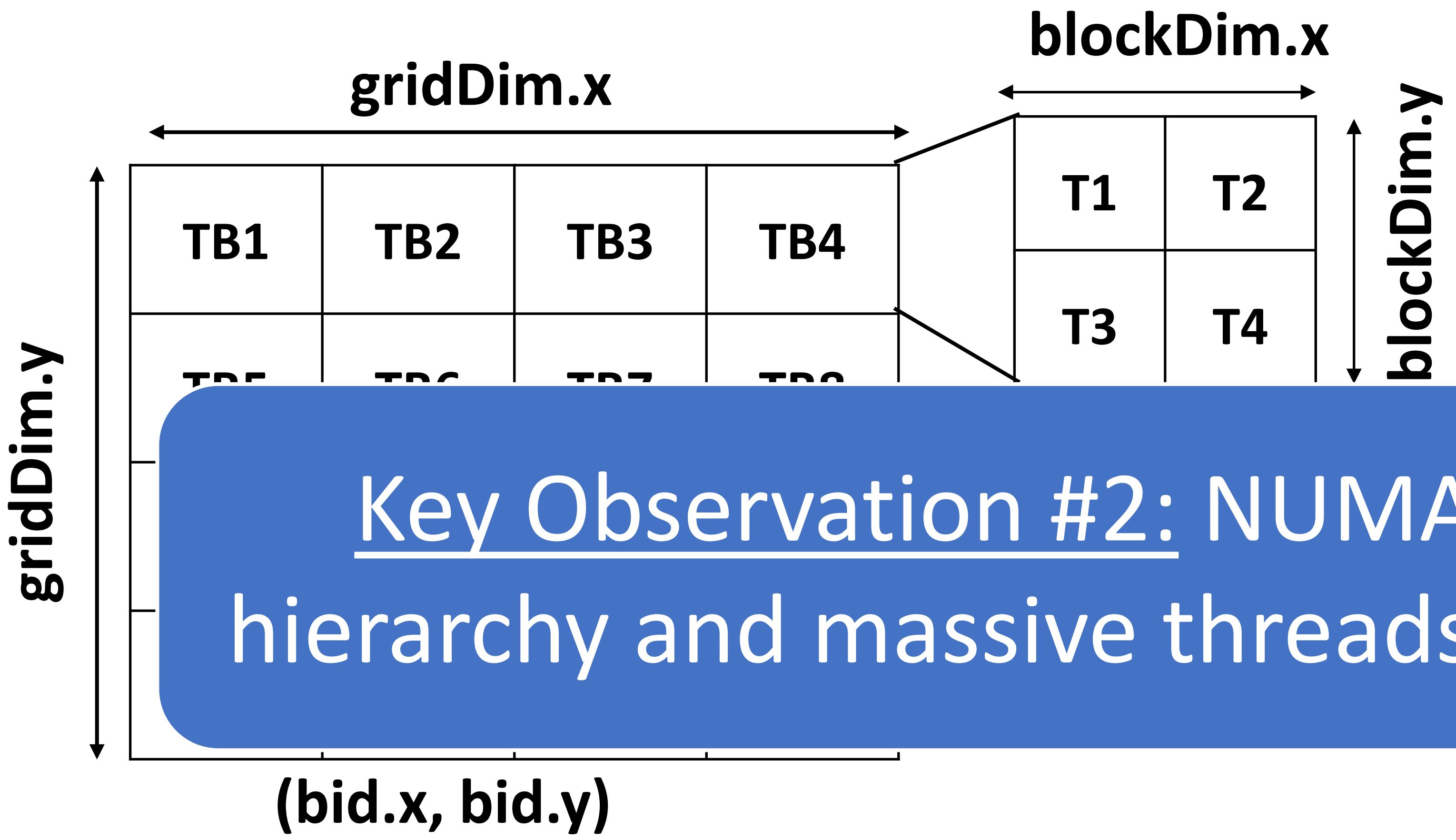


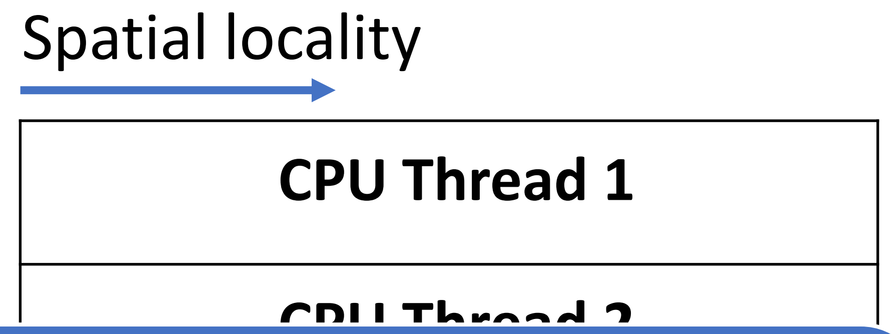
CPU Thread 1
CPU Thread 2
CPU Thread 3
CPU Thread 4

CPU flat coarse-grain threads

+ Scheduling at thread-block level
+ Expressive Thread IDs
+ Low work/spatial locality per thread

GPU vs CPU Programming Model





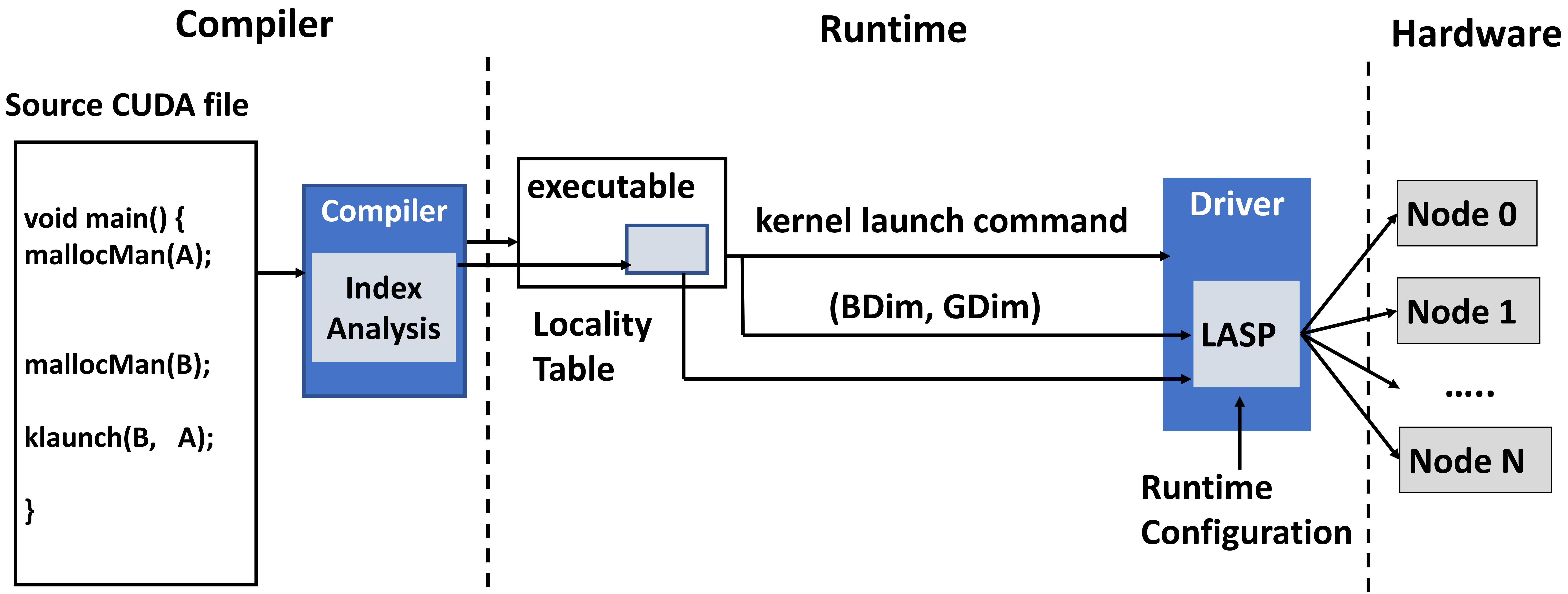
Key Observation #2: NUMA-GPU should consider the hierarchy and massive threads of GPU programming model

GPU hierarchical fine-grain threads

CPU flat coarse-grain threads

+ Scheduling at thread-block level
+ Expressive Thread IDs
+ Low work/spatial locality per thread

Locality-Aware Data Management (LADM)



```
void main() {
mallocMan(A);


mallocMan(B);

klaunch(B,   A);

}
```

Threadblock-Centric Static Index Analysis

Locality-Aware Scheduling and Placement (LASP)

LADM [*MICRO'20*]

- **Key Idea:** LADM exploits a *threadblock-centric index analysis* to optimize runtime threadblock scheduling, data placement and cache policy.

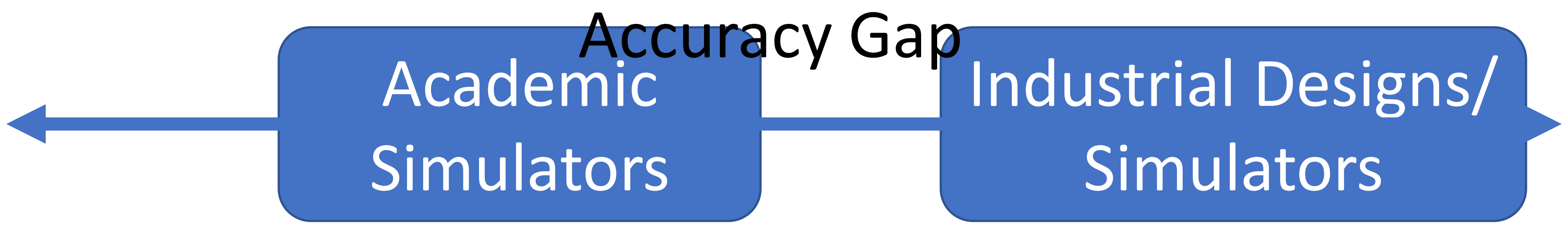
- **Key Results:** LADM decreases inter-GPU memory traffic by 4x and comes within 83% of ideal monolithic performance while using limited and cheap interconnect technology.

More details can be found in the thesis & our MICRO'20 paper

Architecture Simulators

- Simulation is commonly used to estimate the effectiveness of a new architectural design idea.

- The simulation tools used by industry are often not released for open use.



Accuracy Gap

Academic Simulators

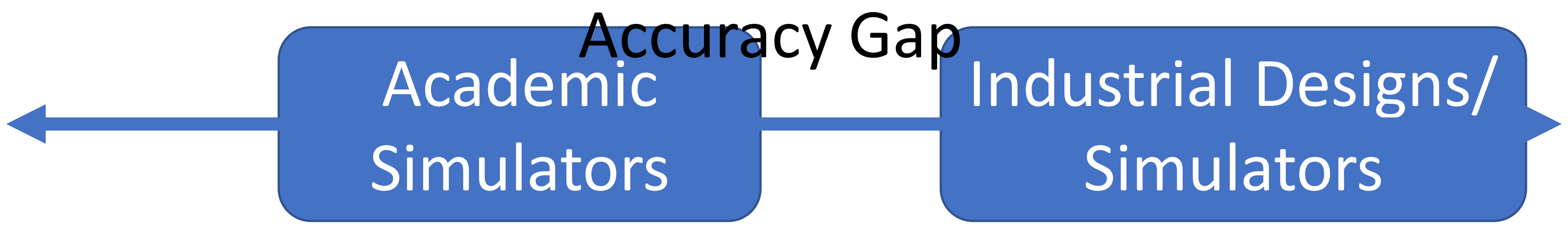
Industrial Designs/ Simulators

Incorrect baseline assumptions
→ unrealistic issues or incorrect conclusions ☹

Architecture Simulators

- Simulation is commonly used to estimate the effectiveness of a new architectural design idea.
- The simulati ed for open use.

Research cannot look ahead, if its baseline assumptions are too far behind



Incorrect baseline assumptions
→ unrealistic issues or incorrect conclusions ☹

GPU Accelerators are Evolving Rapidly

2009 2011 2013 2015 2017 2019 2022

Fermi

- mISA sm20
- Caches/Atomics
- Dual warp scheduler

Kepler

- mISA sm30
- DP unit
- Dynamic Parallelism
- Dual issue

Maxwell

- mISA sm50
- Subcore model

Pascal

- mISA sm60
- Unified memory
- HBM
- FP16 support
- Streaming l1 cache

Volta

- mISA sm70
- Scoped synchronization
- Tensor cores & INT unit
- Independent threads SIMT
- Cooperative Groups
- Unified adaptive cache

Turing

- mISA sm75
- New tensor cores
- RT-cores
- UDP cores

Ampere

- mISA sm80
- Sparse tensor cores
- Asynchronous copy and barriers
- HBM2

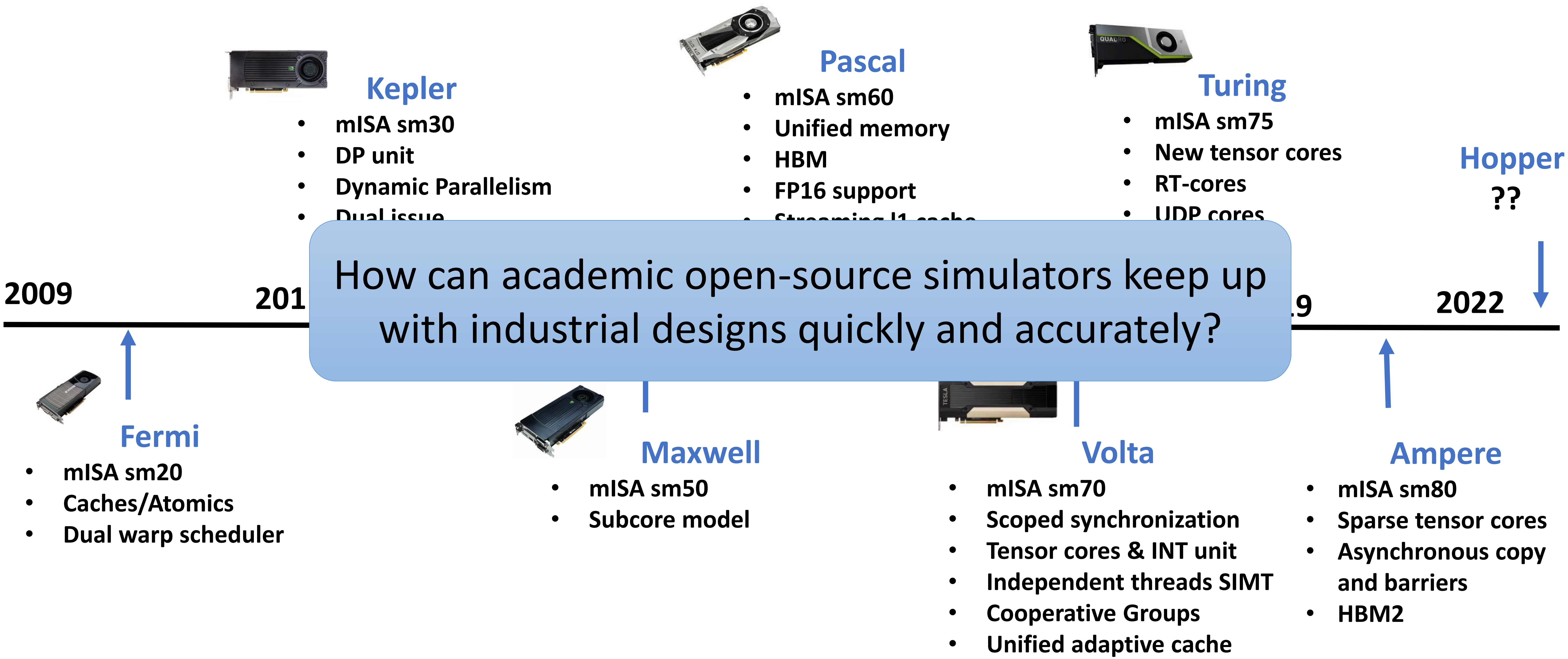
Hopper

??

New machine ISA and architecture designs every 1-2 years!

We show here an example of Nvidia GPU. Similar trend was observed for other GPU vendors.

GPU Accelerators are Evolving Rapidly



2009

Fermi
- mISA sm20
- Caches/Atomics
- Dual warp scheduler

201

Kepler
- mISA sm30
- DP unit
- Dynamic Parallelism
- Dual issue

Maxwell
- mISA sm50
- Subcore model

Pascal
- mISA sm60
- Unified memory
- HBM
- FP16 support
- Streaming L1 cache

Volta
- mISA sm70
- Scoped synchronization
- Tensor cores & INT unit
- Independent threads SIMT
- Cooperative Groups
- Unified adaptive cache

Turing
- mISA sm75
- New tensor cores
- RT-cores
- UDP cores

9

Ampere
- mISA sm80
- Sparse tensor cores
- Asynchronous copy and barriers
- HBM2

2022

Hopper
??

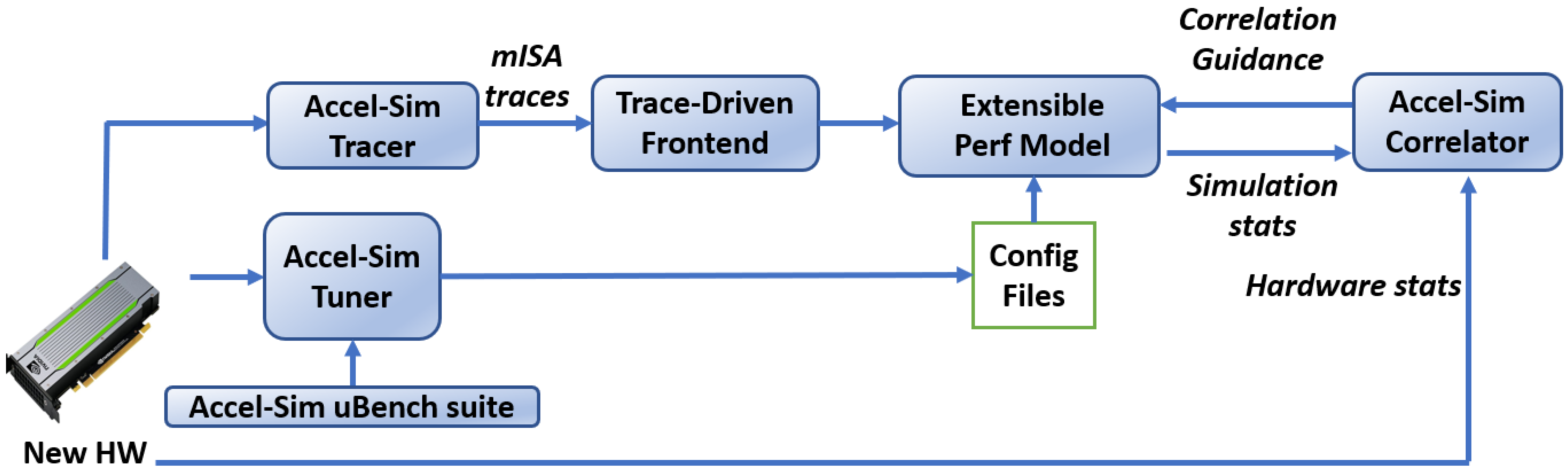
How can academic open-source simulators keep up with industrial designs quickly and accurately?

New machine ISA and architecture designs every 1-2 years!

We show here an example of Nvidia GPU. Similar trend was observed for other GPU vendors.

Accel-Sim [*ISCA'20*]

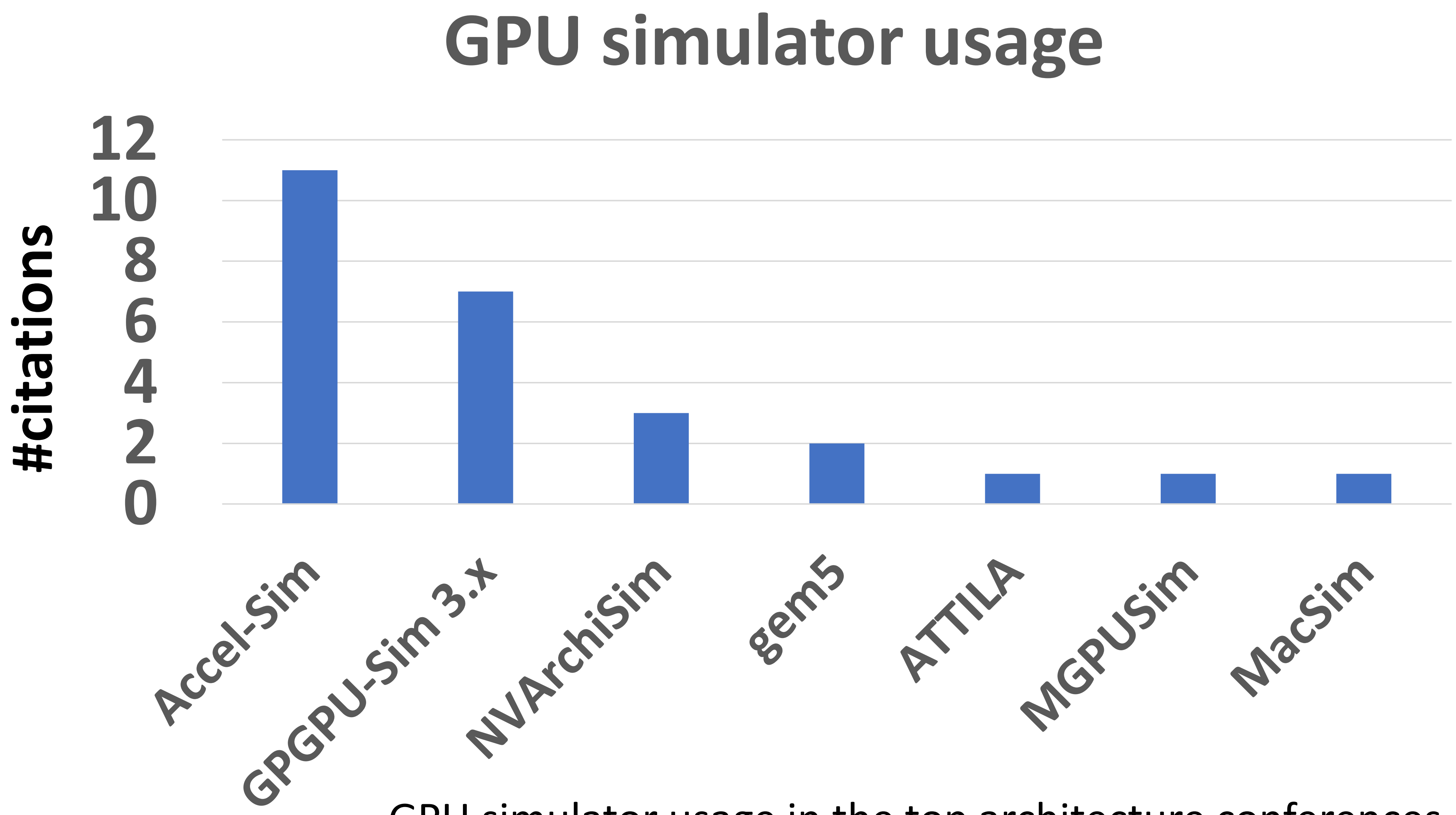
- Accel-Sim introduces a simulation framework to help solve the problem of keeping simulators up-to-date with contemporary designs.



- Key Results: Modeling and validating against five generations of NVIDIA GPUs ranging from Kepler to Ampere with correlation > 0.97 in all instances.

Accel-Sim Popularity/Impact

https://accel-sim.github.io/



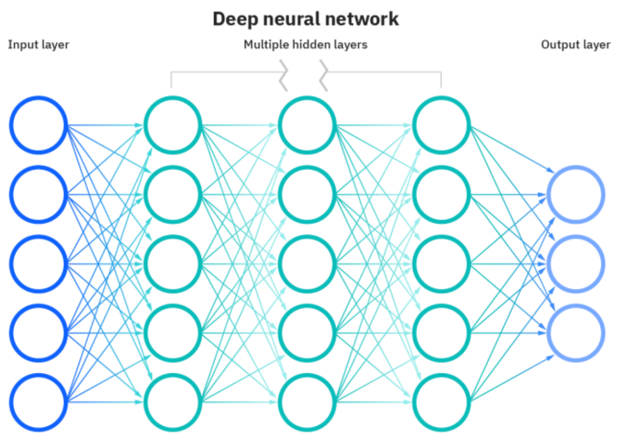
GPU simulator usage in the top architecture conferences (MICRO, ISCA, HPCA, ASPLOS) since June 2019

- The most widely used GPU simulator by the research community since its release
- Usage beyond academia: Sandia National Labs, LLNL, some industrial companies & startups (e.g. Rivos startup among others)

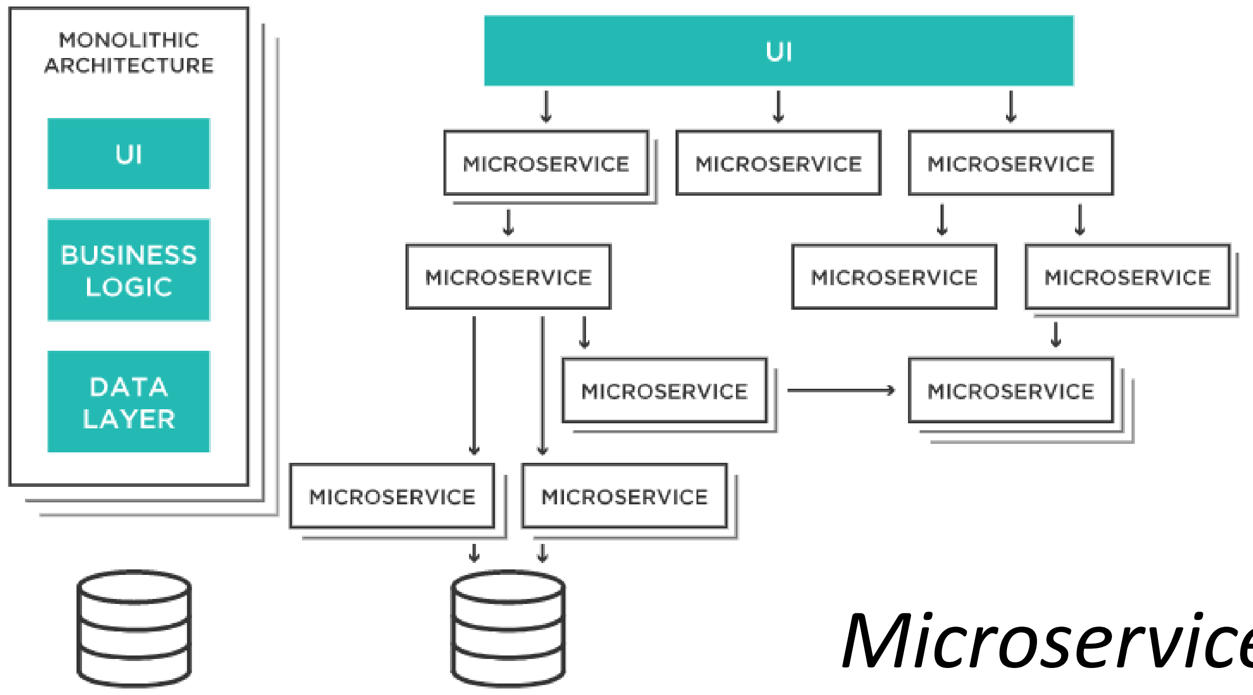
My Ph.D. Thesis Contributions

Efficient uService Processing
→ RPU [under review]

Software



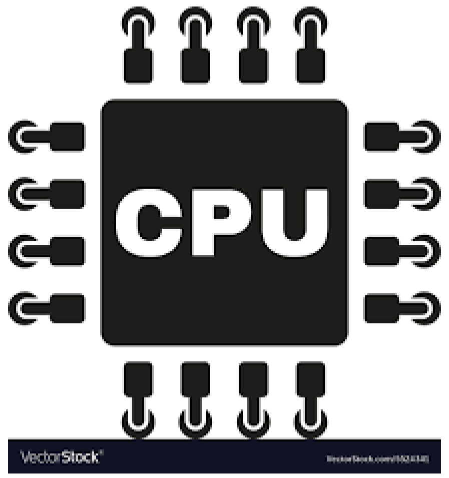
Deep Learning

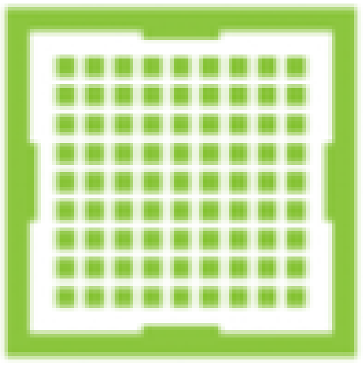


mongoDB
MEMCACHED
gRPC

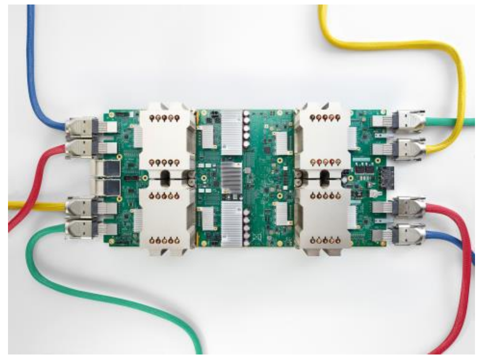
Microservices

Hardware



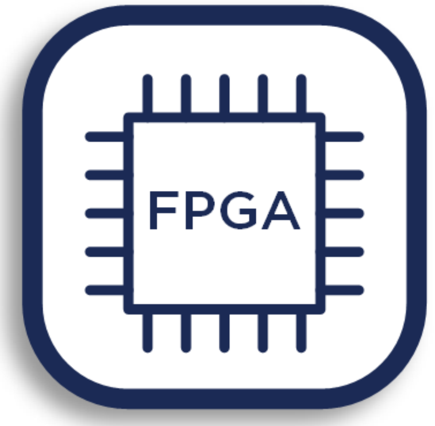


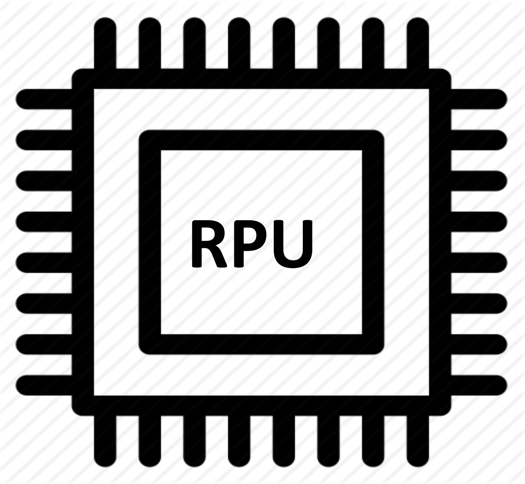
GPU



TPU

VCU





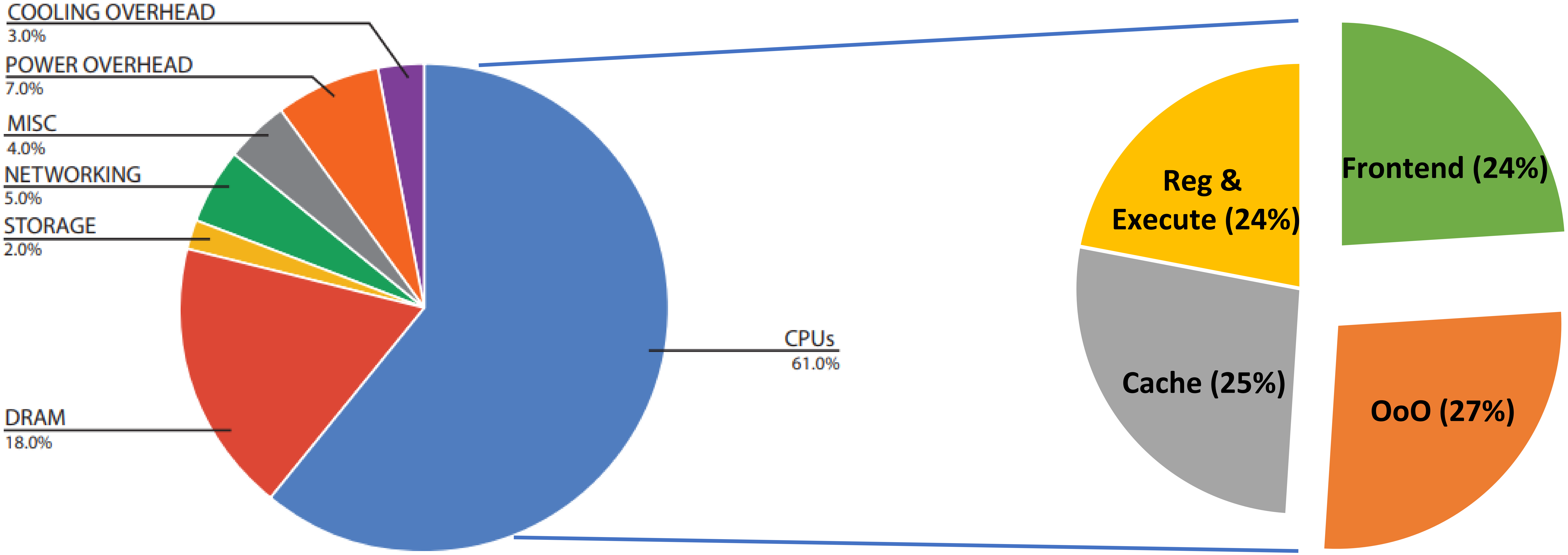
Accelerators

Request Processing Unit (RPU): Single Instruction Multiple Request Processing for Energy-Efficient Data Center Microservices

[under review at a top tier conference]

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

Recall: Datacenter Power Breakdown



Datacenter Power Breakdown (from Google)

CPU Power Breakdown (Intel Skylake)

30% of datacenter power is consumed in CPU's instruction supply (frontend & OoO)

[1] 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
[2] 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
[3] Powell, Michael D., et al. "CAMP: A technique to estimate per-structure power at run-time using a few simple parameters." *HPCA 2009*

1 Application, Million of Users

Google

facebook

Private Datacenter

Uber

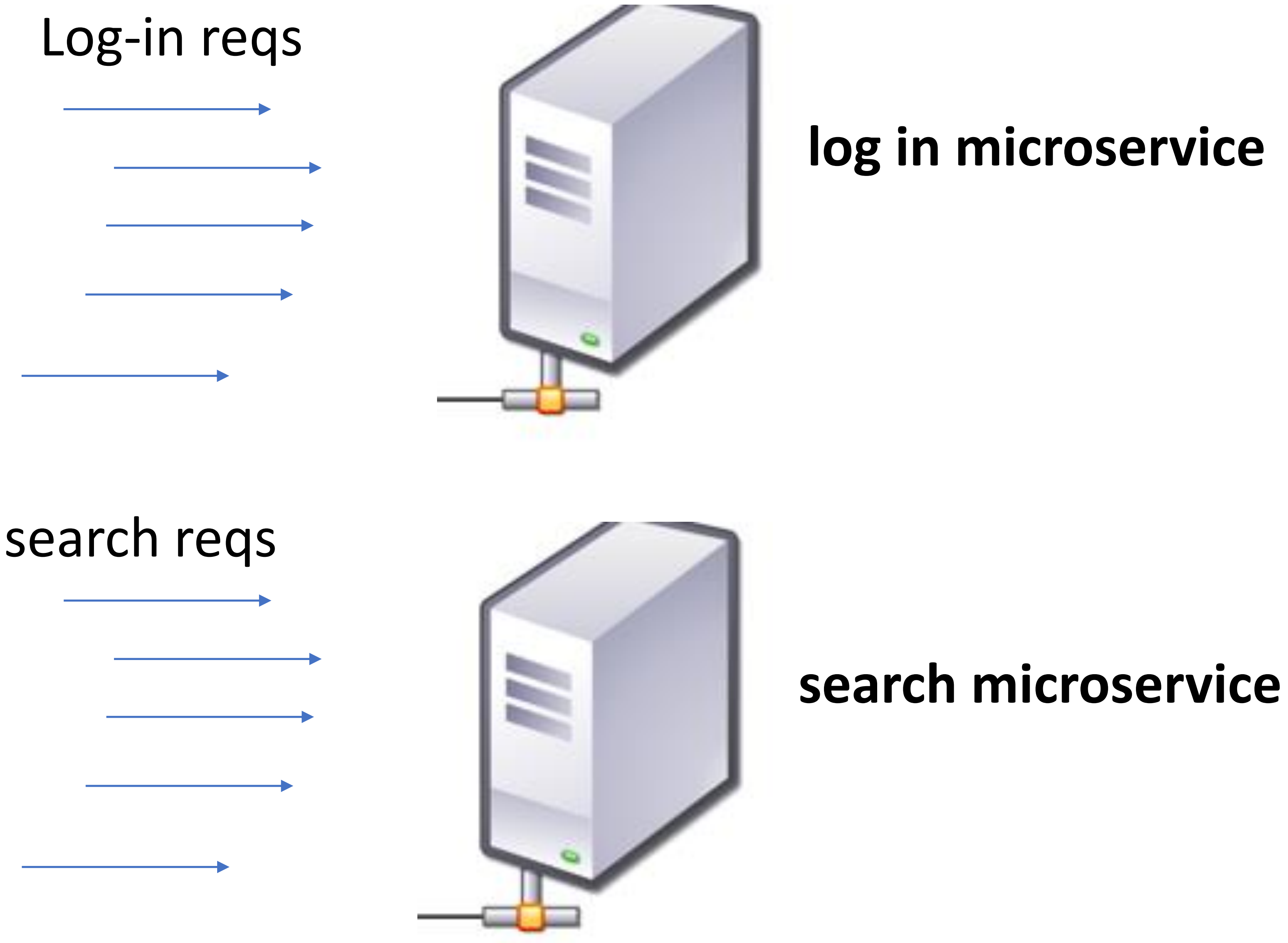
NETFLIX

AWS

Public Datacenter

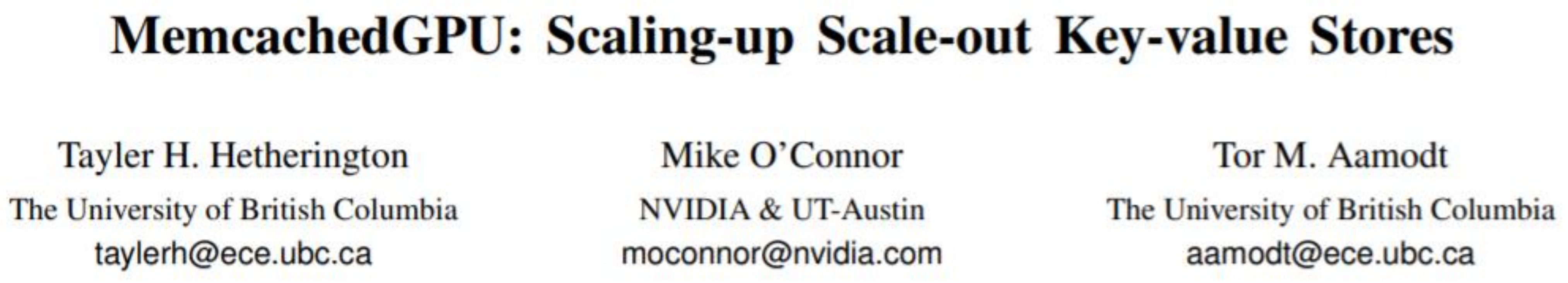
"Similar" Request-Level Parallelism

1000s of independent requests are all running the same code



Key Observation#1: Single Program Multiple Data (SPMD) are abundant in the cloud, either in private or public datacenters

Server Workloads on GPU’s SIMT



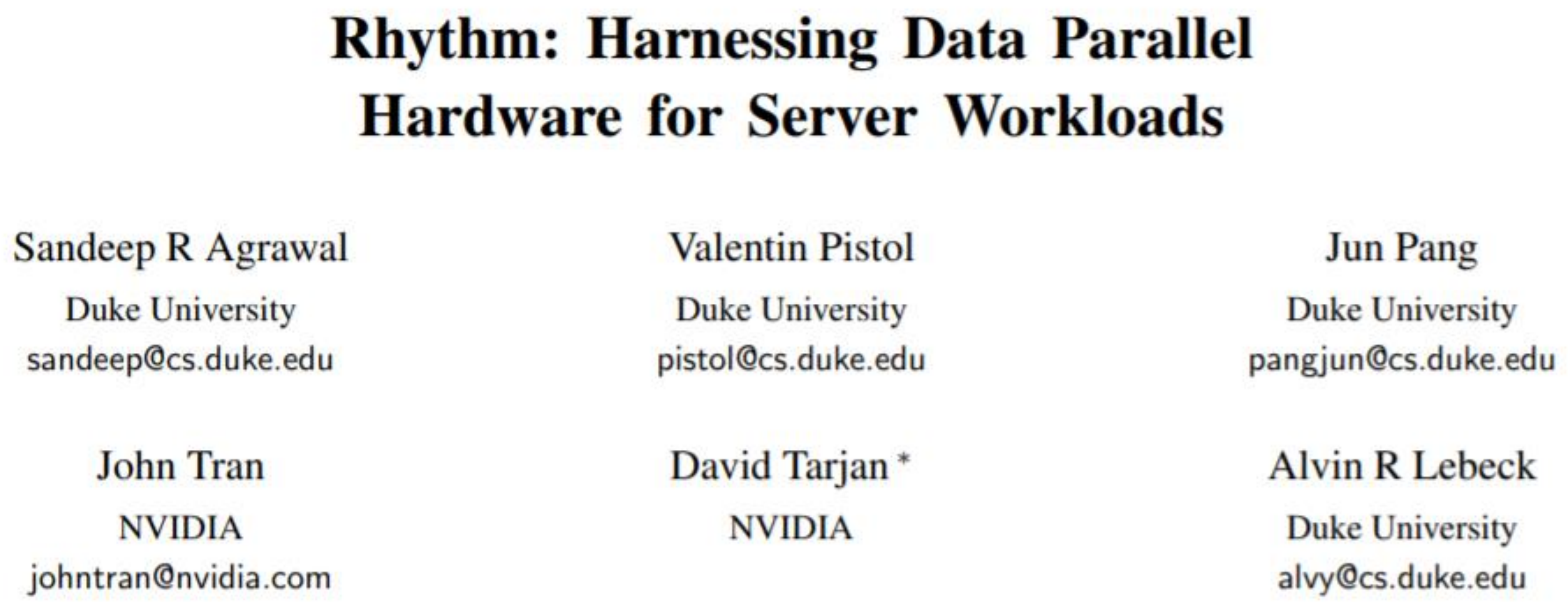
MemcachedGPU: Scaling-up Scale-out Key-value Stores

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

Memcached on GPU [SoCC’2015]



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

SPEC-web on GPU [ASPLOS’2014]

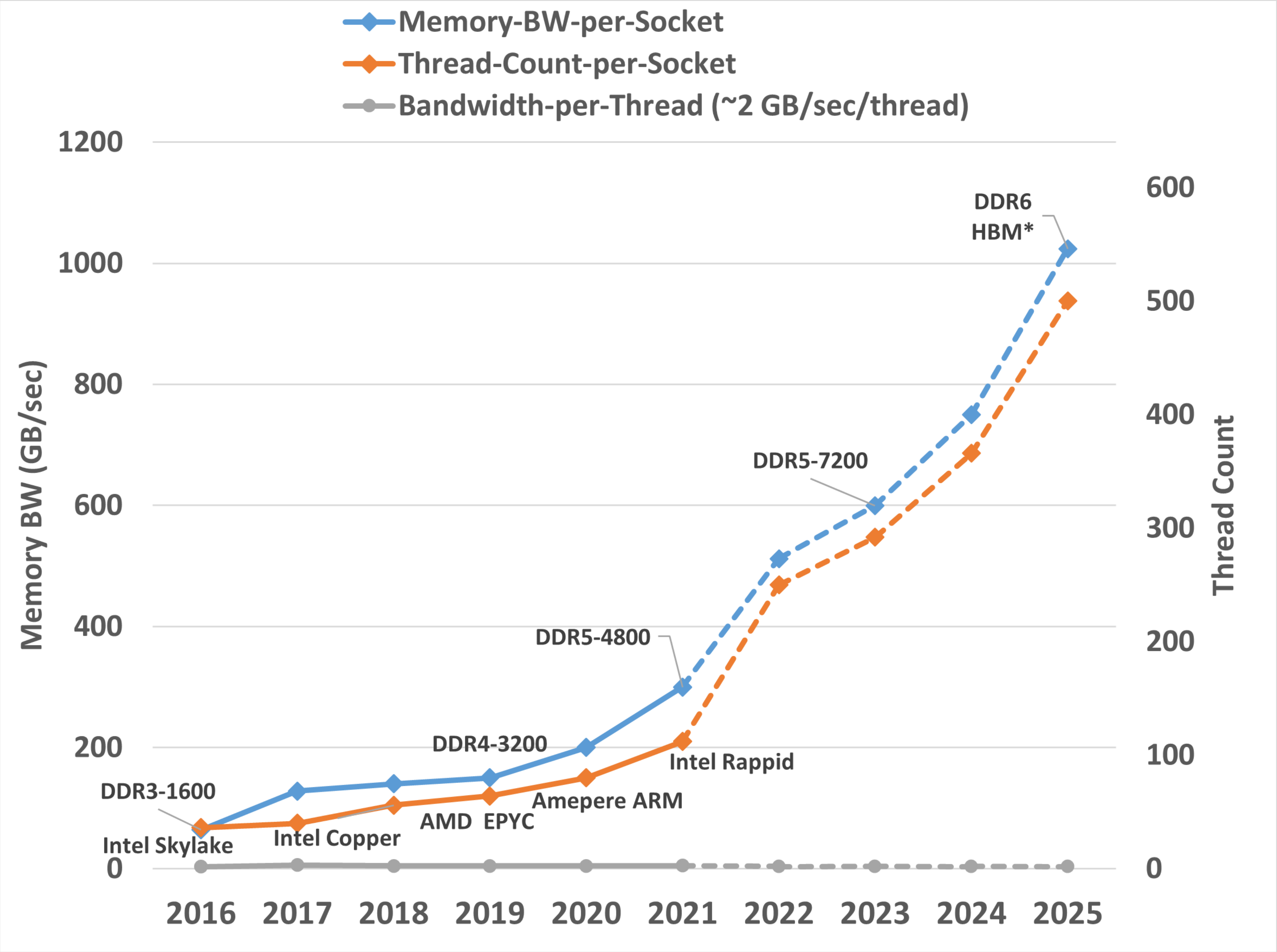
- **Key Idea:** batch requests and run on GPU’s SIMT
- **Advantages**: Significant Energy Efficiency (throughput/watts) vs CPU
- **Drawbacks**:
  - (1) Hindering Programmability (C++/PHP vs CUDA)
  - (2) Limited System Calls Support (CPU-GPU communication)
  - <u>(3) High service latency</u>
    - In Rhythm [ASPLOS’14], GPU TITANX reports 6000X slower latency than CPU
    - In MemcachedGPU [SoCC’15], GPU was 10X slower than CPU

“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”

Urs Hölzle
Google SVP

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.

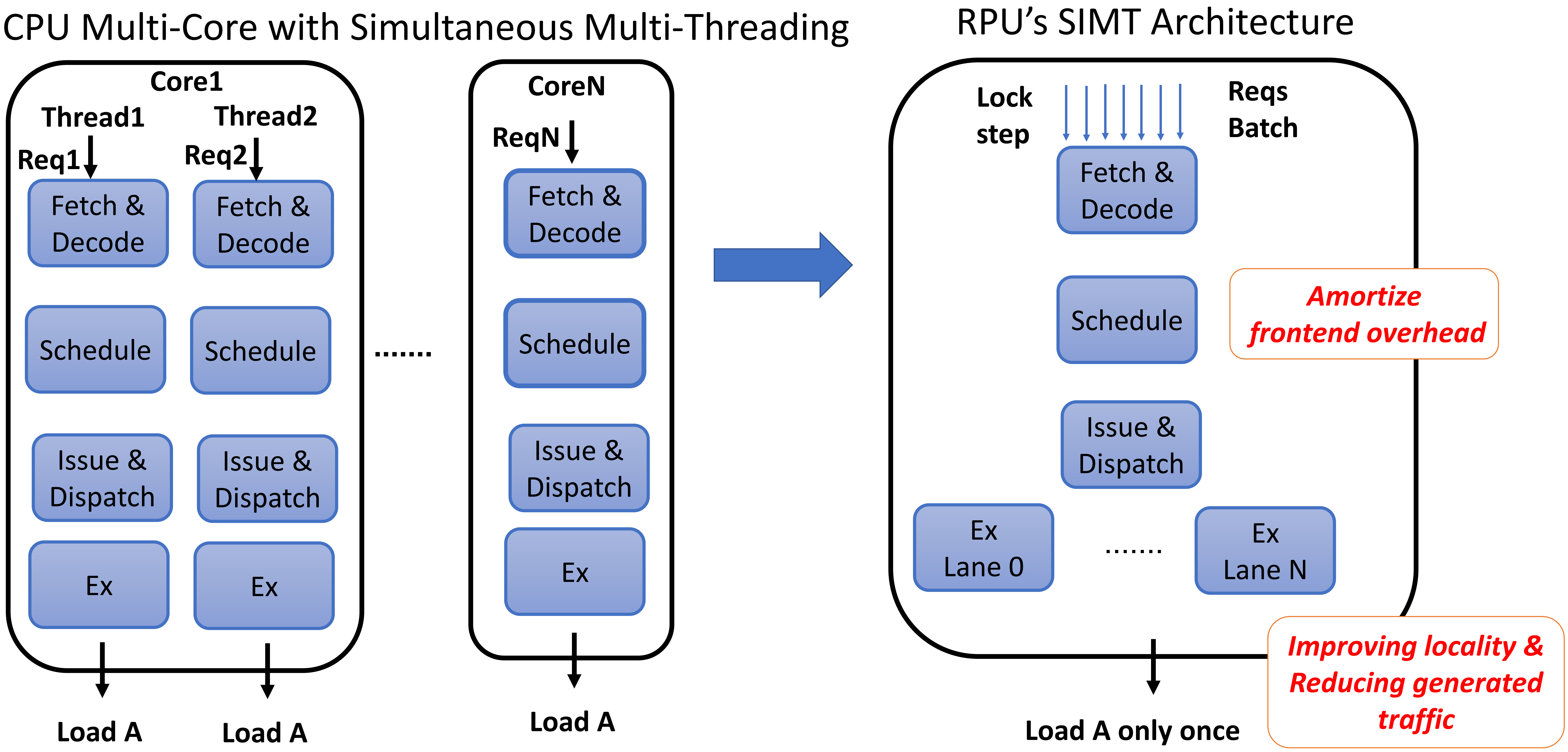
How to increase on-chip throughput of CPU?

- 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 the SIMT efficiency to the CPU world!”

SIMT Efficiency



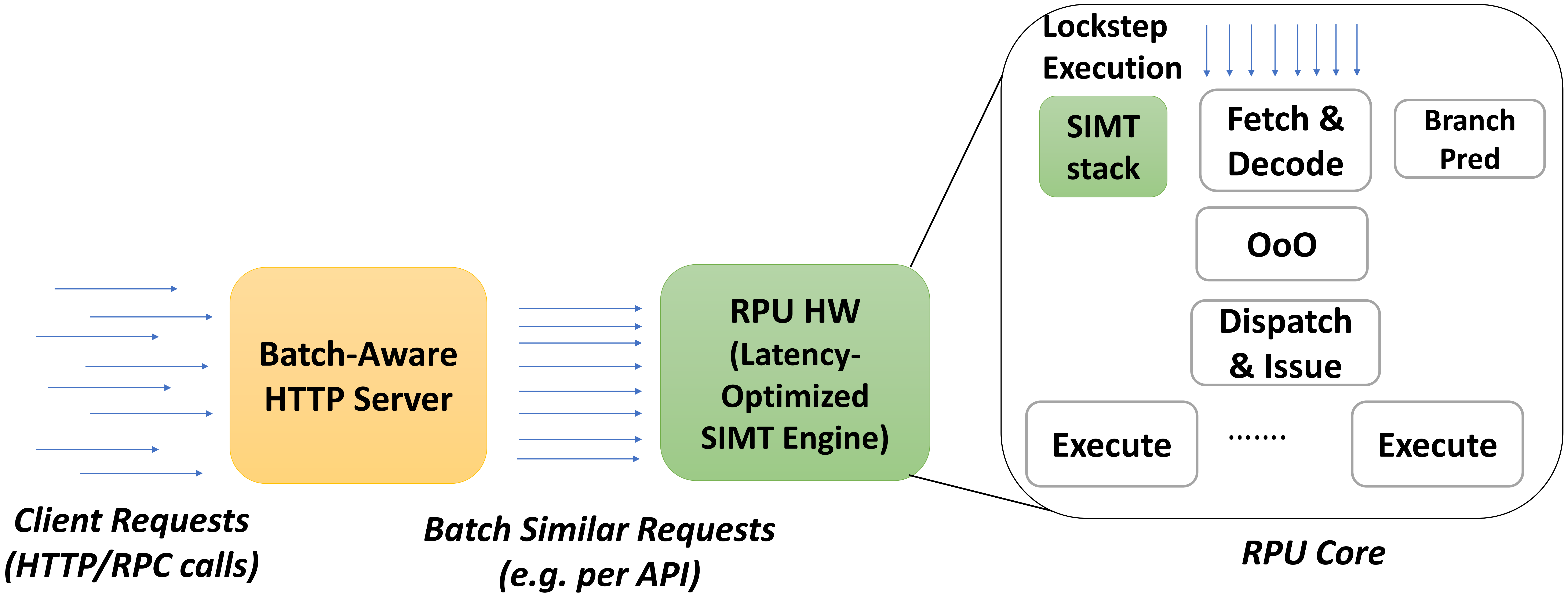
CPU Multi-Core with Simultaneous Multi-Threading

RPU's SIMT Architecture

Amortize frontend overhead

Improving locality & Reducing generated traffic

RPU Overview



CPU vs GPU vs RPU

Metric	CPU	GPU	RPU
Core model	OoO	In-Order	OoO
Freq	High	Moderate	High
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
TLP per core	Low (1-8)	Massive (2K)	Moderate (8-32)
Thread model	SMT	SIMT	SIMT
Consistency	Variant	Weak+NMCA*	Weak+NMCA*
Coherence	Complex	Relaxed Simple	Relaxed Simple
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

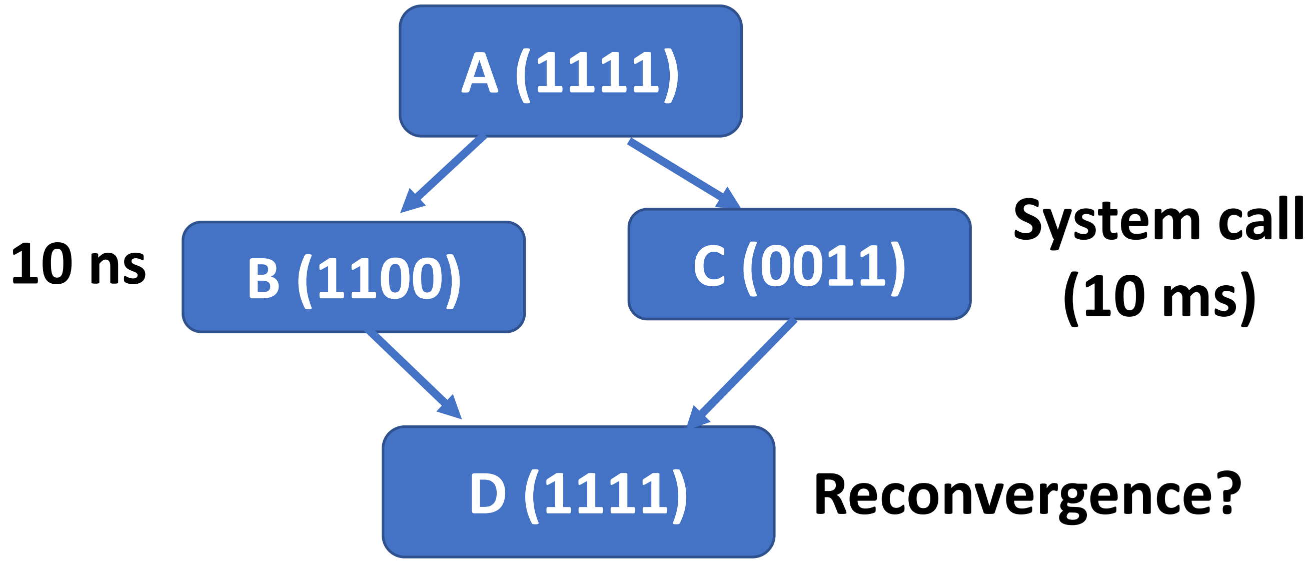
Hechtman, Blake A., et al. "QuickRelease: A throughput-oriented approach to release consistency on GPUs." HPCA 2014

RPU Executive Summary

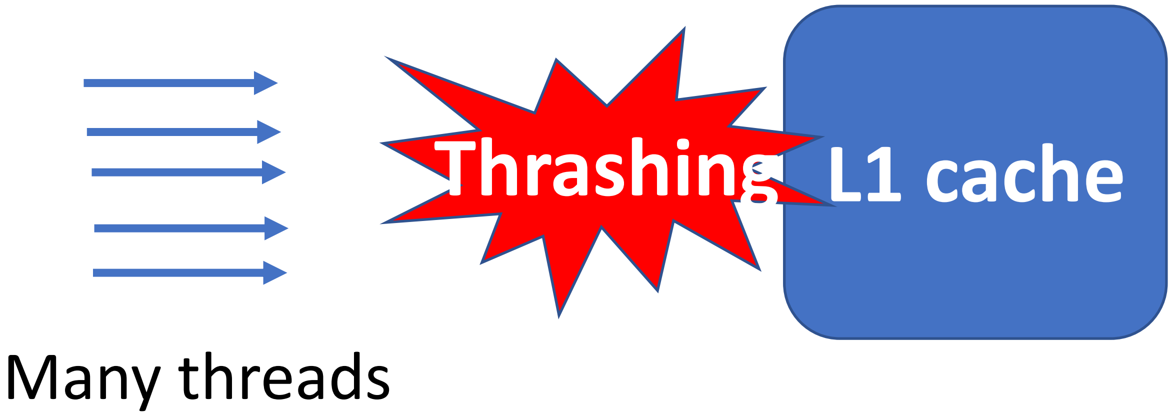
- *Request Similarity* is abundant in the data center.
- We start with *OoO CPU* design and then turns it to *SIMT execution* to maximize chip utilization and exploit the similarity.
- We co-design the software stack to support *batching* and awareness of SIMT execution.

Deep Dive into RPU’s Challenges

- Control Divergence
  - Control divergence wit high latency branch

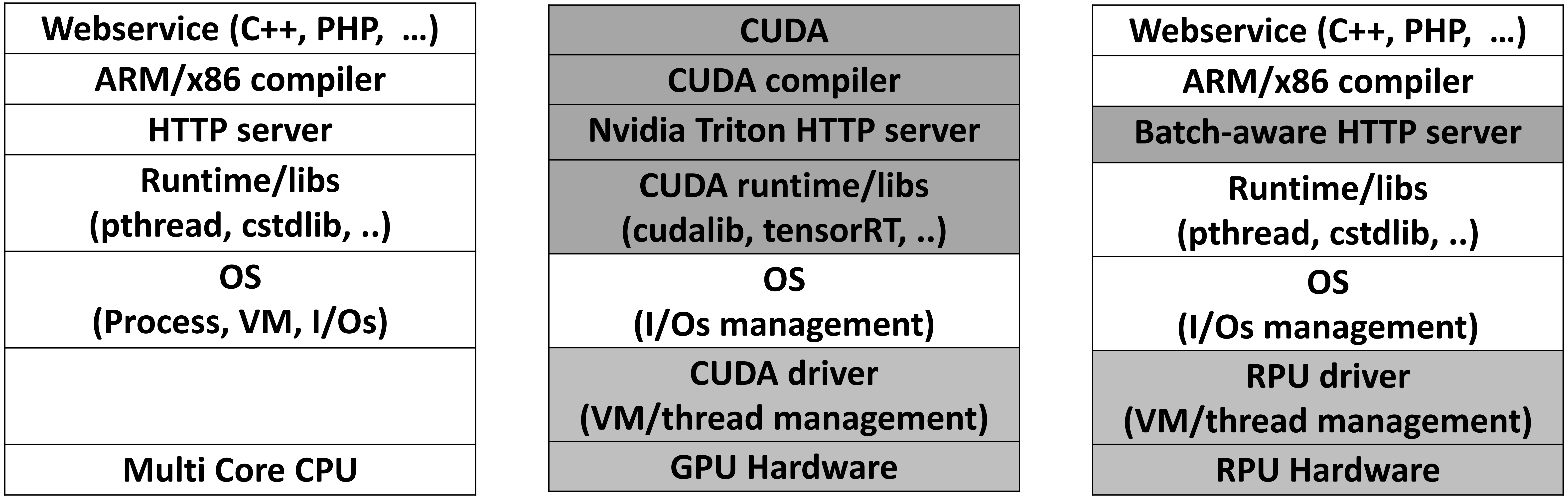


- Memory Divergence
  - Cache Contention & Bank Conflicts



- Higher instruction execution & L1 hit latency
  - Due to larger execution units & cache resources at the backend

HW/SW Stack



Webservice (C++, PHP, …)

ARM/x86 compiler

HTTP server

Runtime/libs (pthread, cstdlib, ..)

OS (Process, VM, I/Os)

Multi Core CPU

CUDA

CUDA compiler

Nvidia Triton HTTP server

CUDA runtime/libs (cudalib, tensorRT, ..)

OS (I/Os management)

CUDA driver (VM/thread management)

GPU Hardware

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

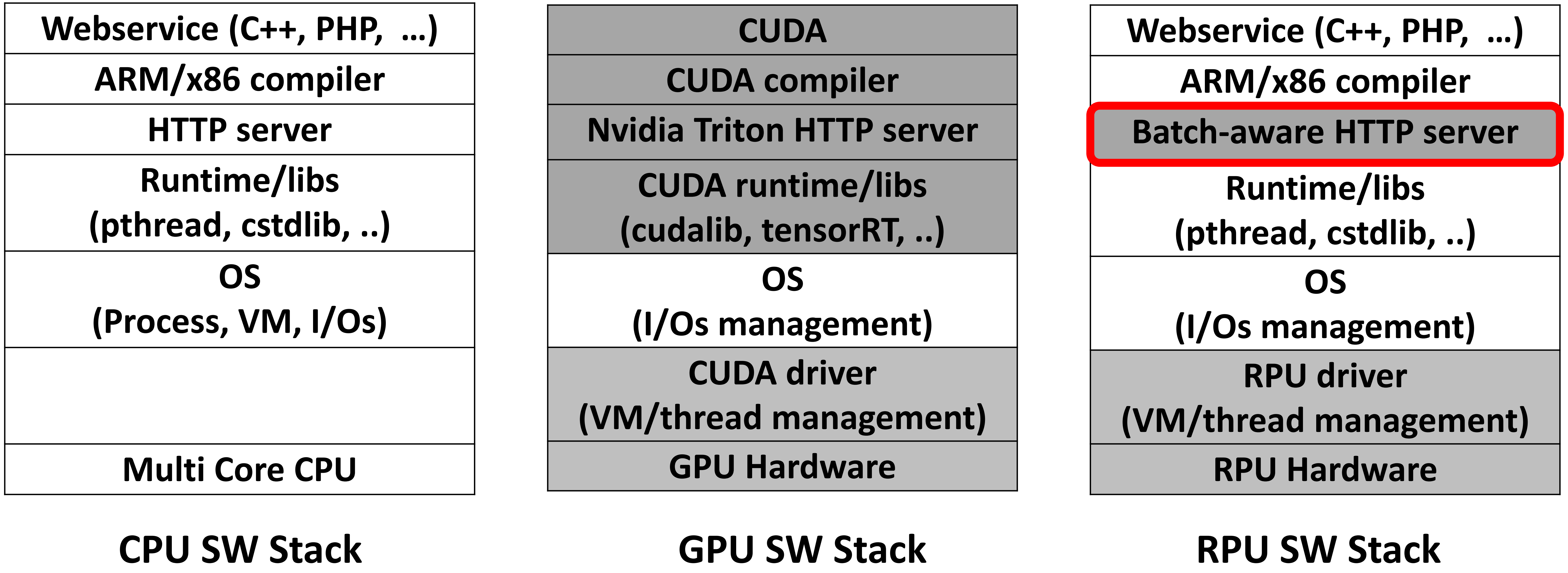
CPU SW Stack **GPU SW Stack** **RPU SW Stack**

→ For RPU, we keep the SW programming interface as in the CPU

→ RPU is binary backward compatible with CPU webservices.

→ Some VM&process management system calls are reimplemented in the RPU driver to be batch-aware

HW/SW Stack



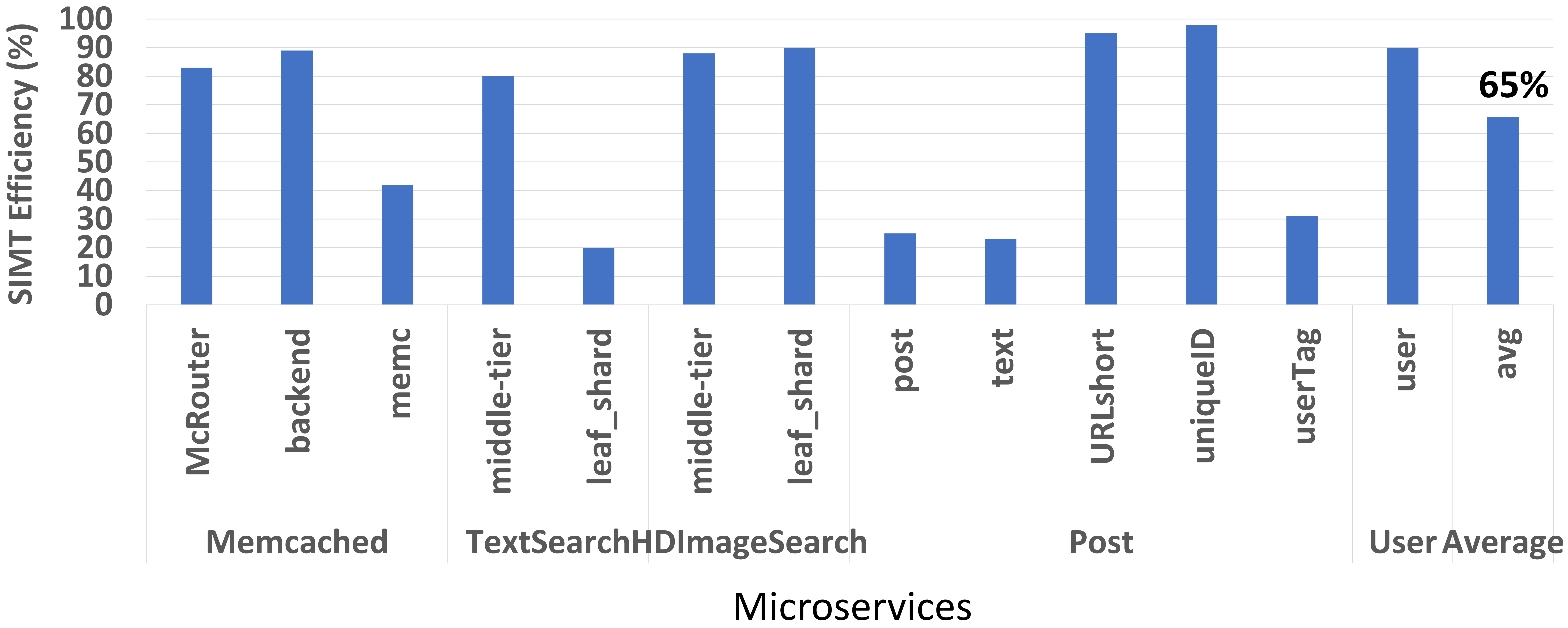
CPU SW Stack	GPU SW Stack	RPU SW Stack
Webservice (C++, PHP, …)	CUDA	Webservice (C++, PHP, …)
ARM/x86 compiler	CUDA compiler	ARM/x86 compiler
HTTP server	Nvidia Triton HTTP server	Batch-aware HTTP server
Runtime/libs (pthread, cstdlib, ..)	CUDA runtime/libs (cudalib, tensorRT, ..)	Runtime/libs (pthread, cstdlib, ..)
OS (Process, VM, I/Os)	OS (I/Os management)	OS (I/Os management)
	CUDA driver (VM/thread management)	RPU driver (VM/thread management)
Multi Core CPU	GPU Hardware	RPU Hardware

→ For RPU, we keep the SW programming interface as in the CPU

→ RPU is binary backward compatible with CPU webservices.

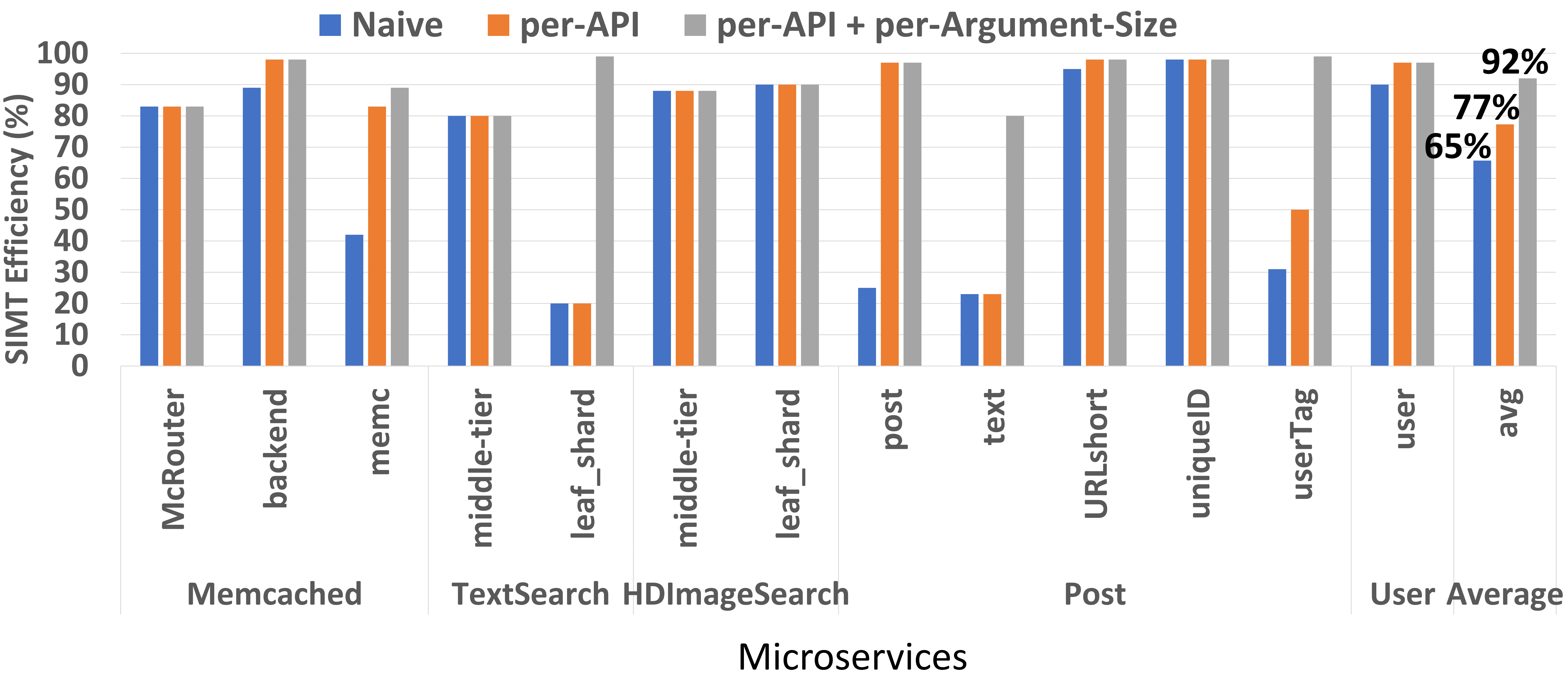
→ Some VM&process management system calls are reimplemented in the RPU driver to be batch-aware

SIMT Control Efficiency

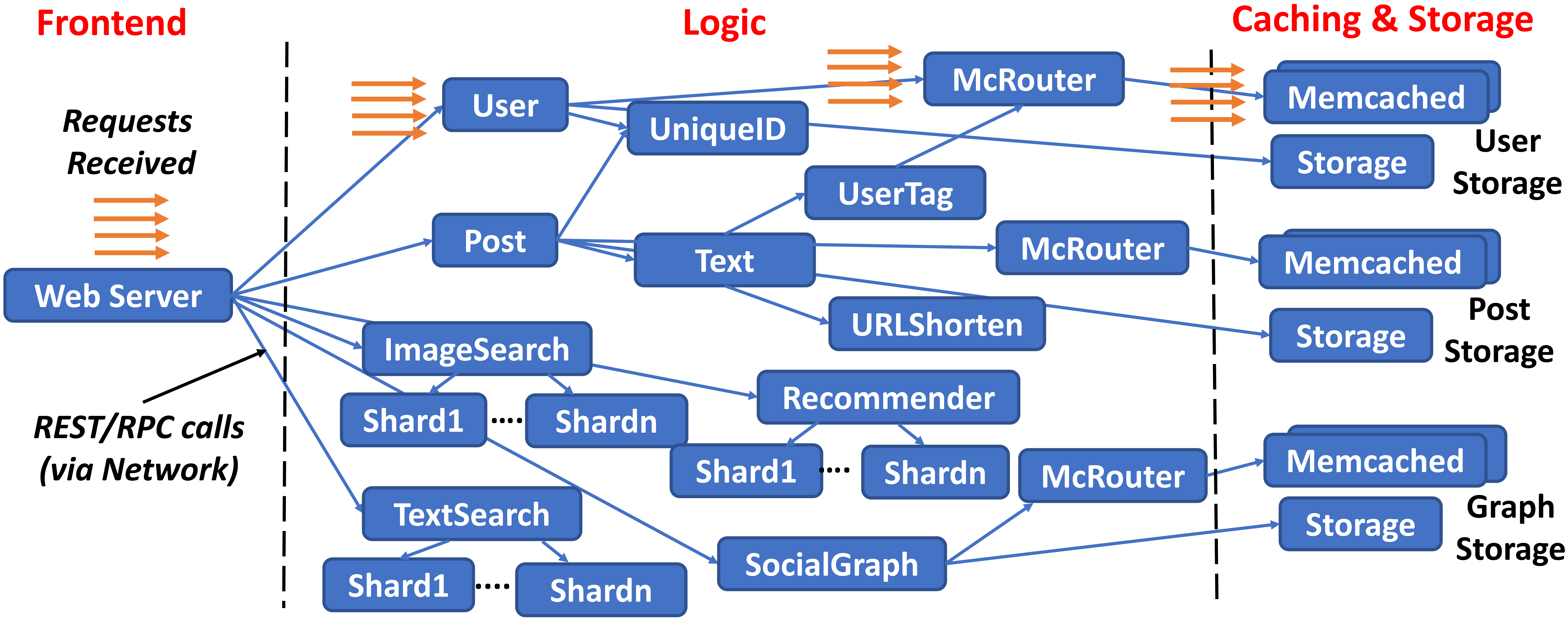


Notes: (1) Batch Size = 32, (2) System Calls are not included, (3) SIMT Eff = scalar-instructions / (batch-instructions * batch-size), (4) fine-grain locking are assumed

SIMT Control Efficiency (Optimized)

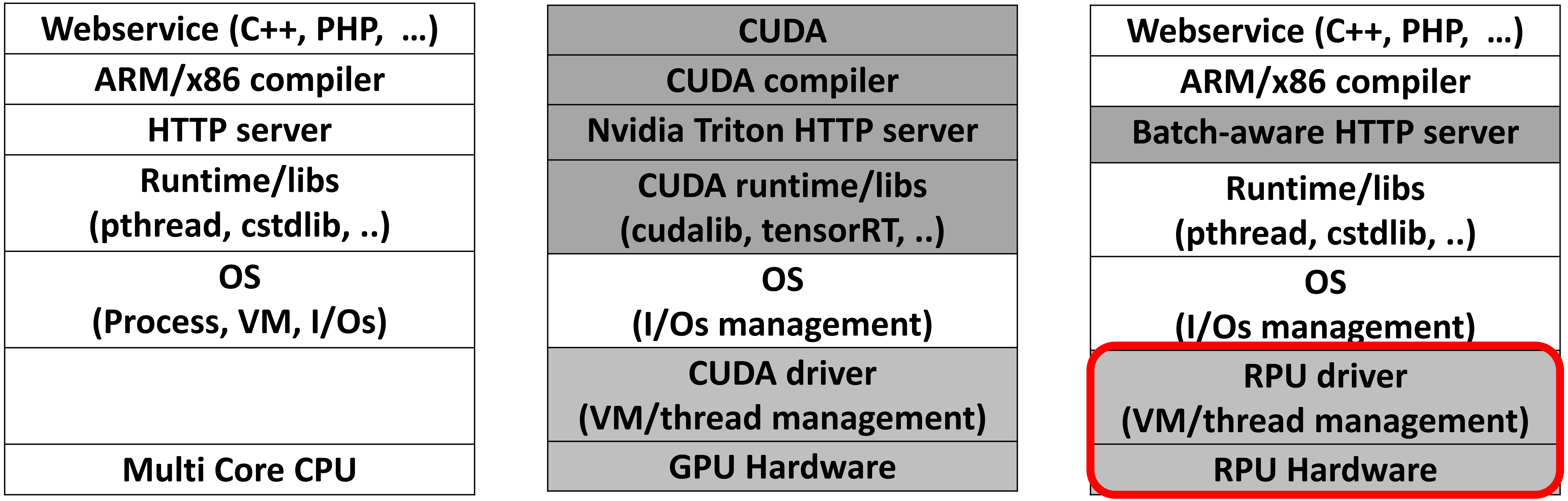


System-Level RPU Batching



Key Observation: Batching is heavily employed in the data center (DL inference, Memcached, ..)
→ Instead of batching individual microservices, we propose batching in all microservices in the graph

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

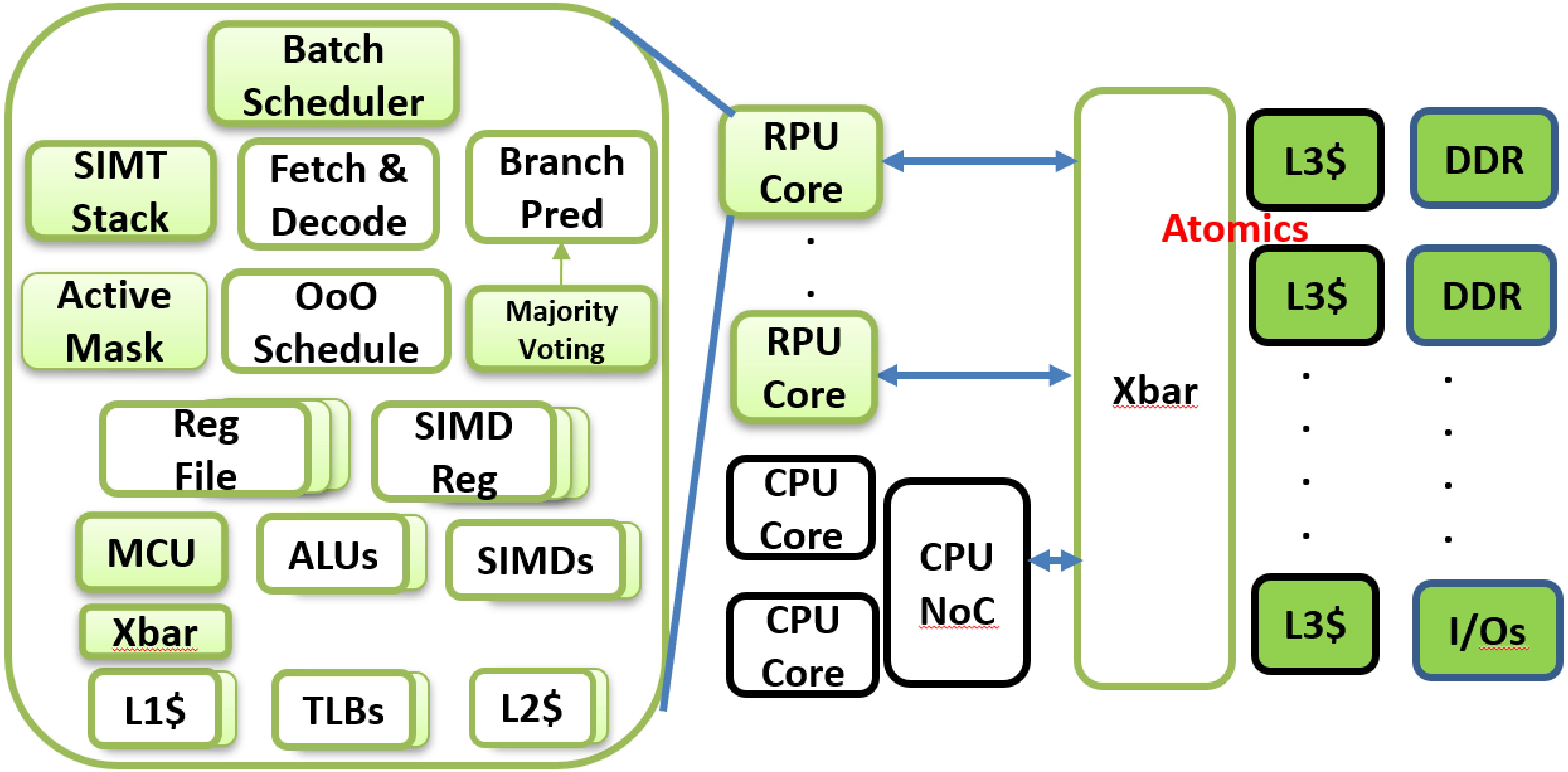
CUDA
CUDA compiler
Nvidia Triton HTTP server
CUDA runtime/libs (cudalib, tensorRT, ..)
OS (I/Os management)
CUDA driver (VM/thread management)
GPU Hardware

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

RPU HW

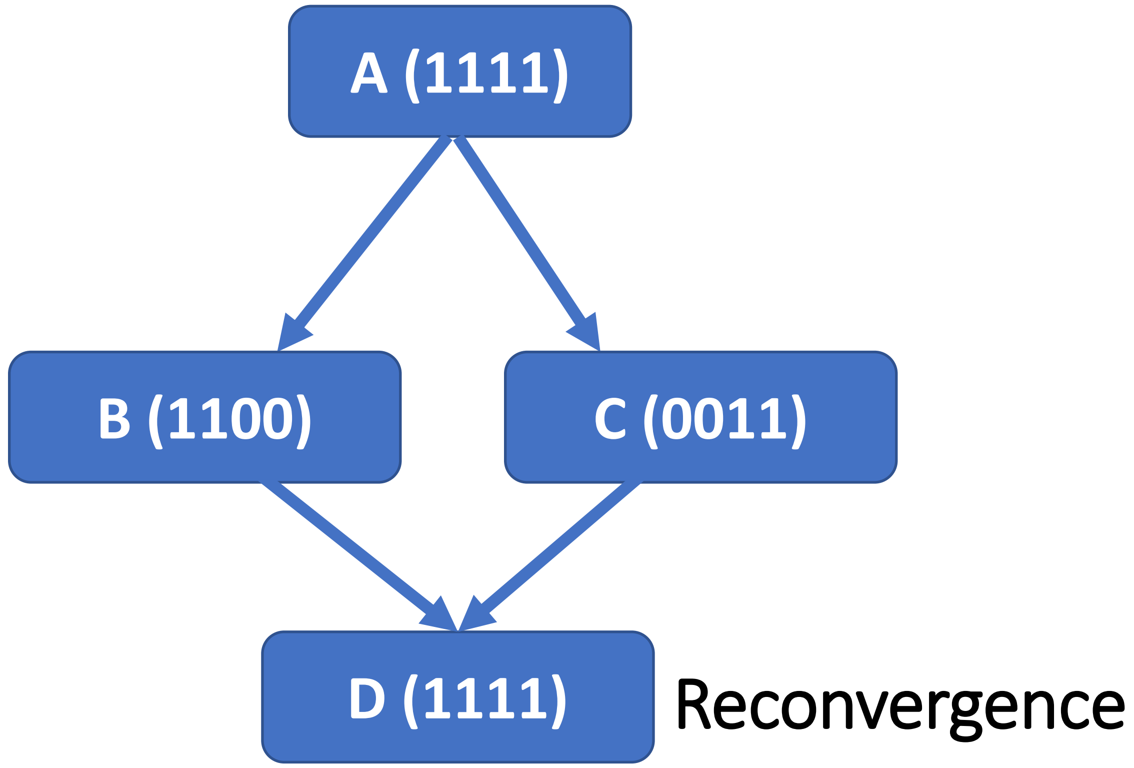


Batch Scheduler
SIMT Stack
Fetch & Decode
Branch Pred
Active Mask
OoO Schedule
Majority Voting
Reg File
SIMD Reg
MCU
ALUs
SIMDs
Xbar
L1$
TLBs
L2$
RPU Core
RPU Core
CPU Core
CPU Core
CPU NoC
Xbar
Atomics
L3$
L3$
L3$
DDR
DDR
I/Os

Control Divergence Handling

```
1. / / BBA Basic Block "A"
2. if ( x > 0)
3. {
4.      / / BBB
5. }
6. else
7.  {
8.     / / BBC
9.  }
10. / / BBD
```

Divergent code example



Control Flow with Active Mask

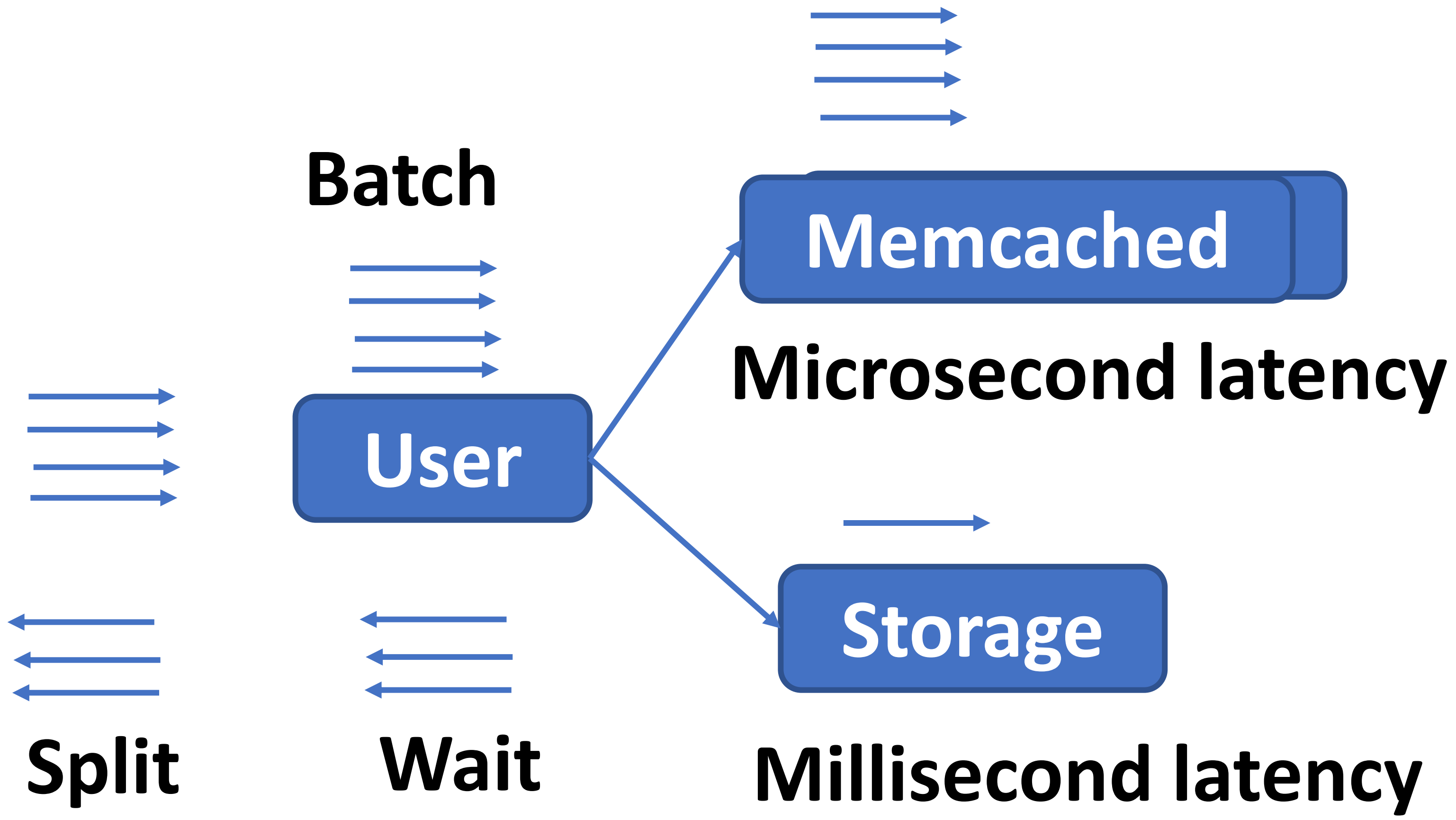
PC	RPC	Active Mask
D	...	1111
C	D	0011
B	D	1100

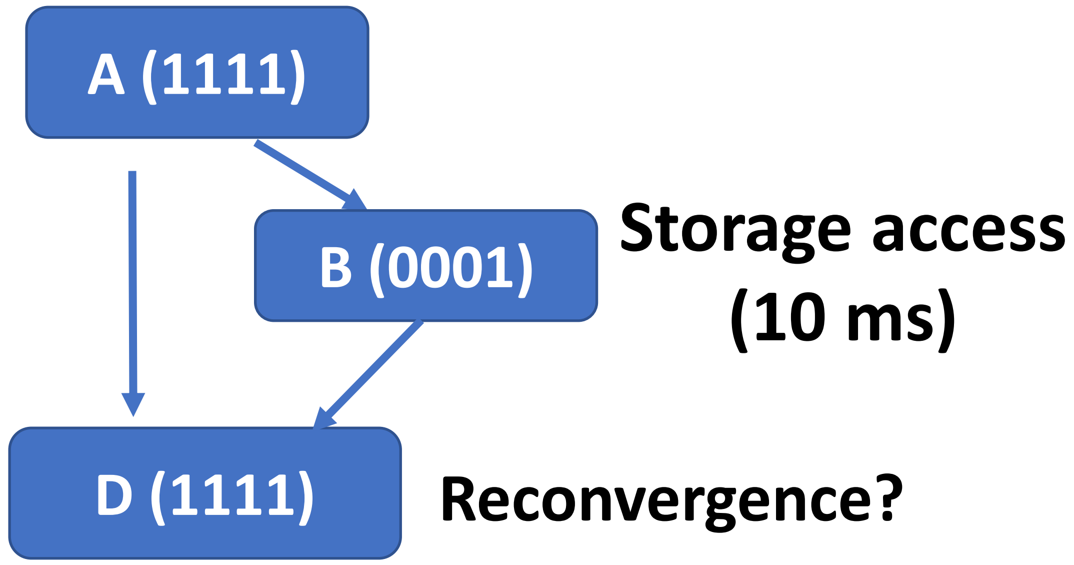
HW SIMT stack after line#2

Serialize divergent paths

System-Level Batch Splitting

```
Procedure get_user(int userid)
   /* first try the cache */
  data = memcached_fetch("userrow:" + userid)
  if not data        /* SIMT Divergence*/
       /* not found : request database */
       data = db_select("SELECT * FROM users
        WHERE userid = ?", userid)
       /* then store in cache until next get */
       memcached_add("userrow:" + userid, data)
   end       /* SIMT Reconvergence Point*/
   return data
```



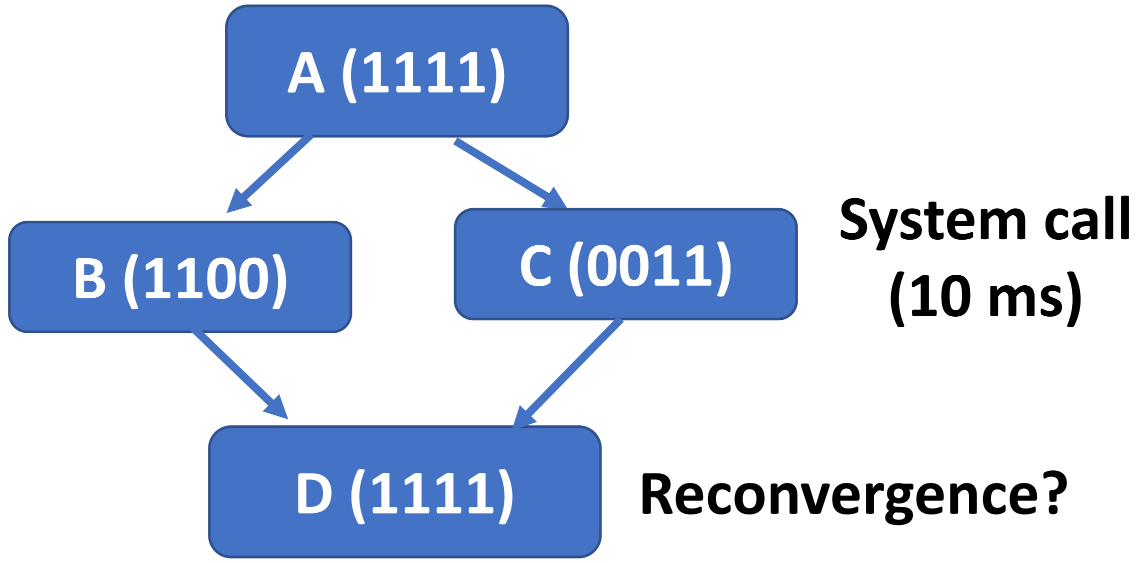


Control Flow with Active Mask

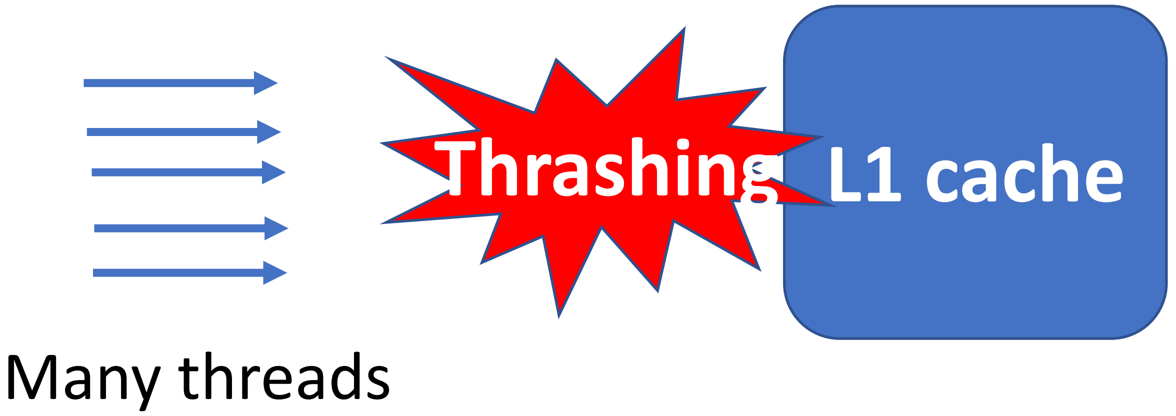
Deep Dive into RPU’s Challenges

- Control Divergence
  - Control divergence wit high latency branch

✓

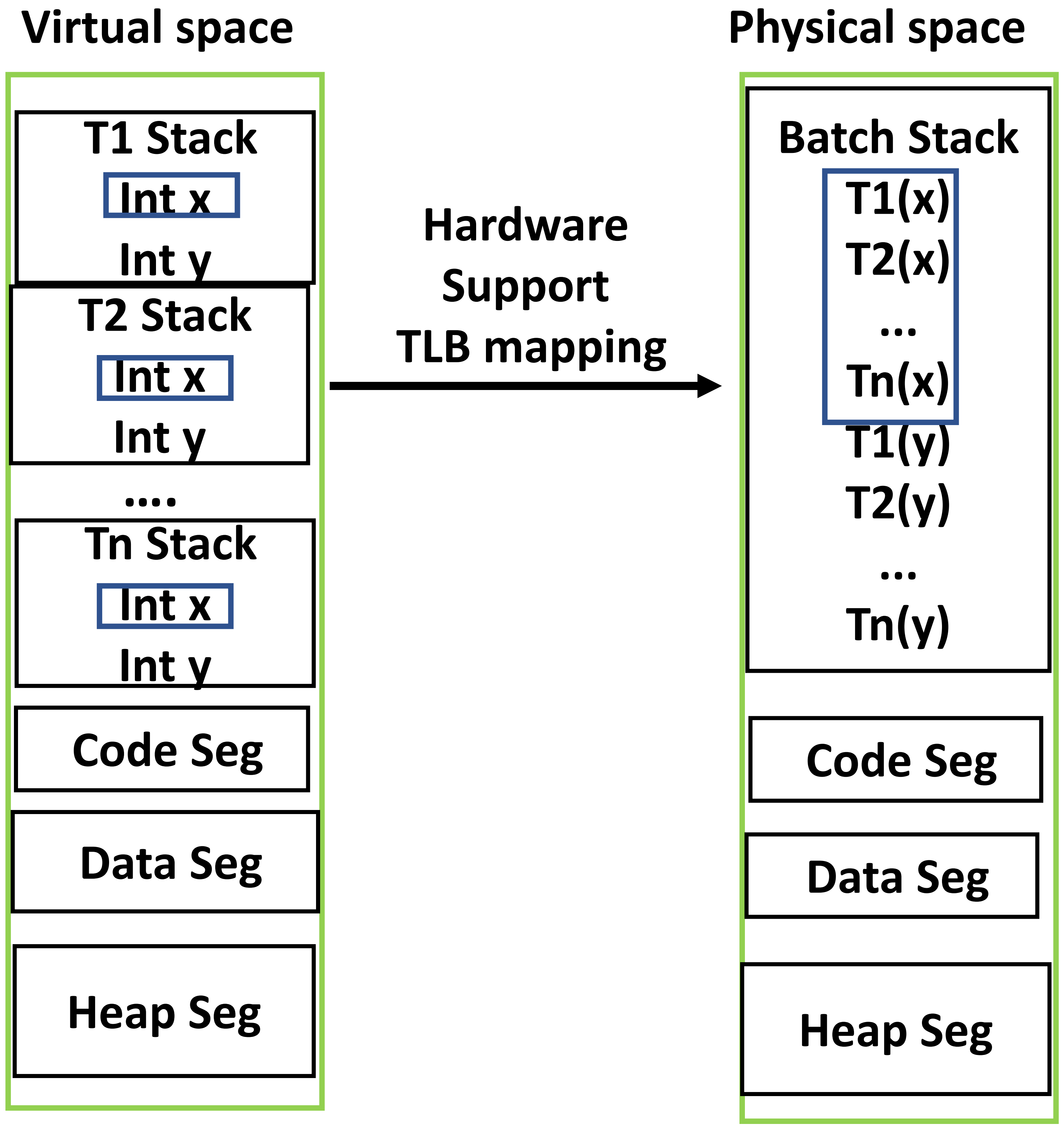


- Memory Divergence
  - Cache Contention & Bank Conflicts

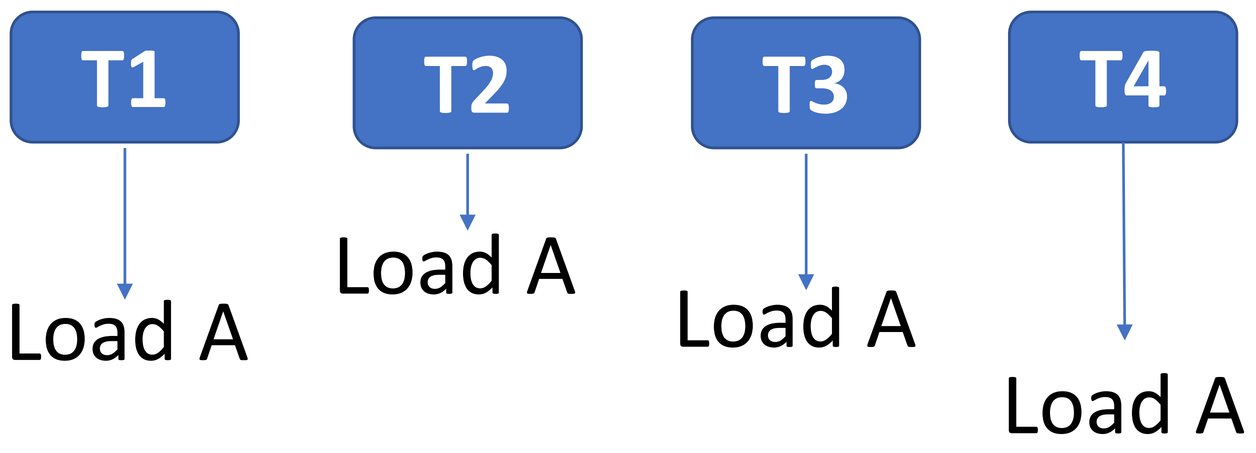


- Higher instruction execution & L1 hit latency
  - More execution units & cache resources at the backend

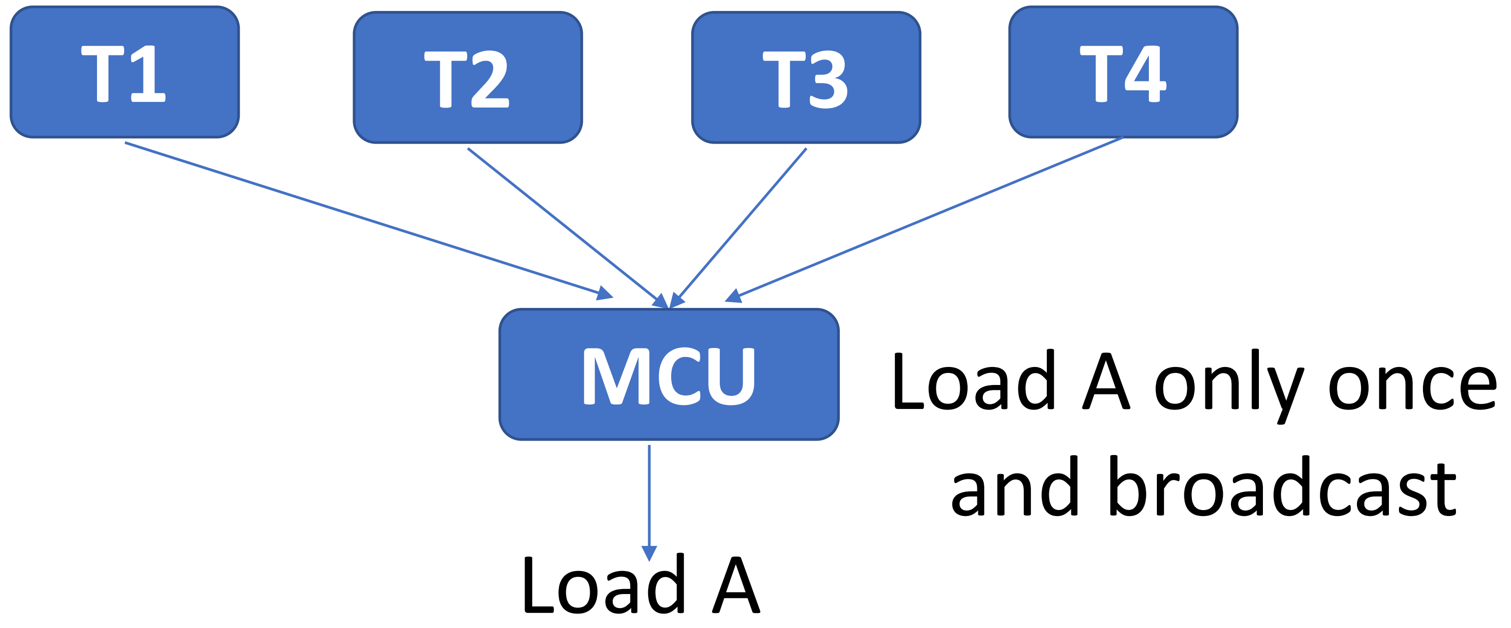
Memory Coalescing Optimizations



Stack segment coalescing with data interleaving



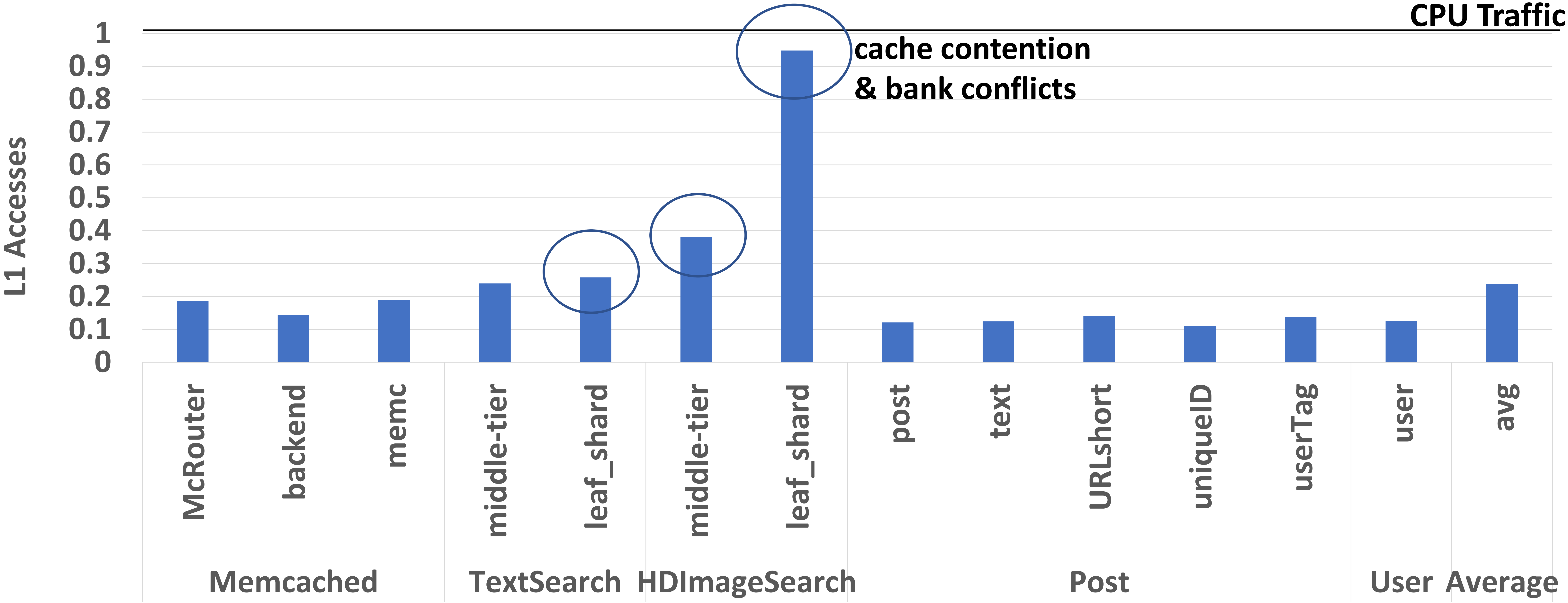
Independent threads execution (CPU)



SIMT execution with MCU

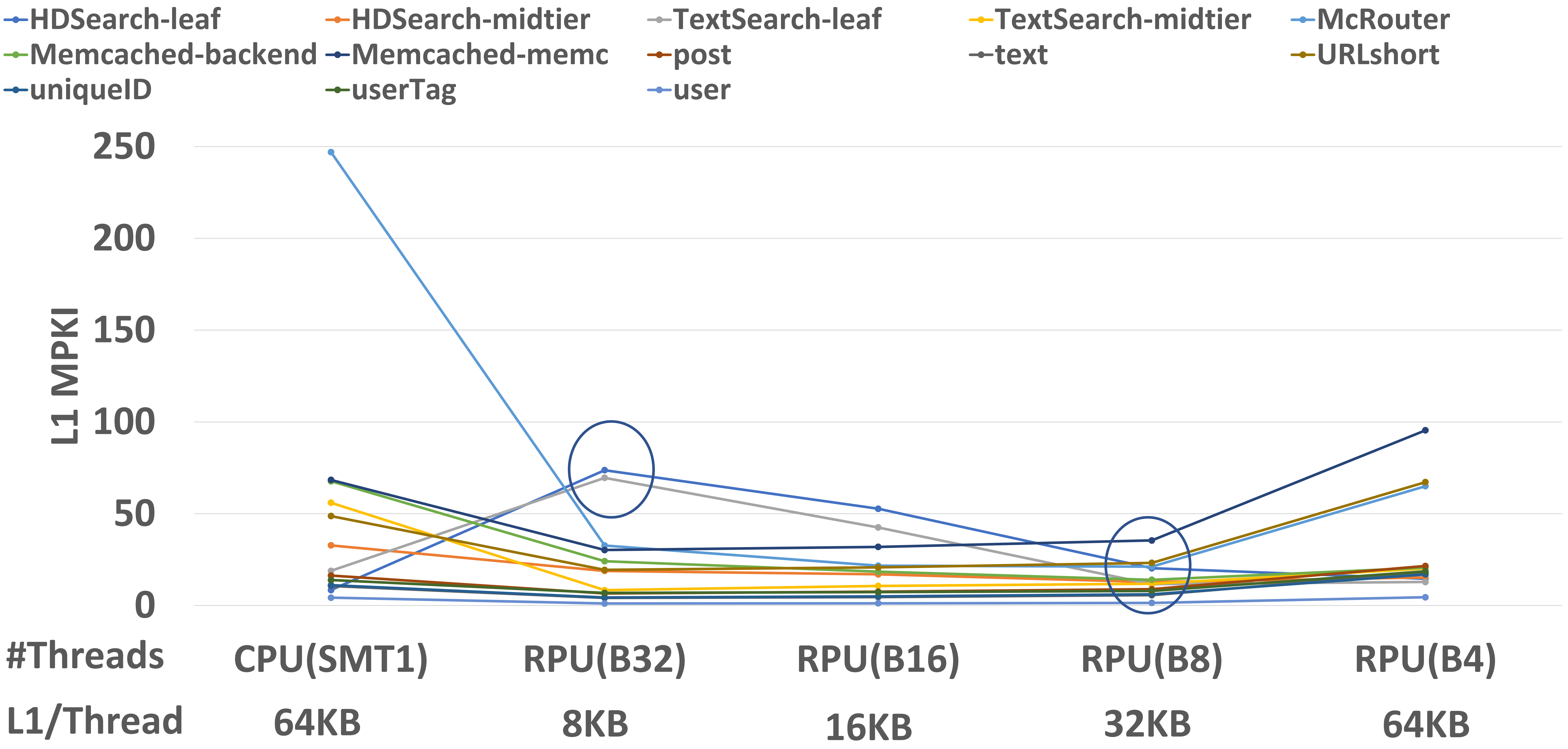
HW memory coalescing unit (MCU) for Heap & Data segments

Traffic Reduction

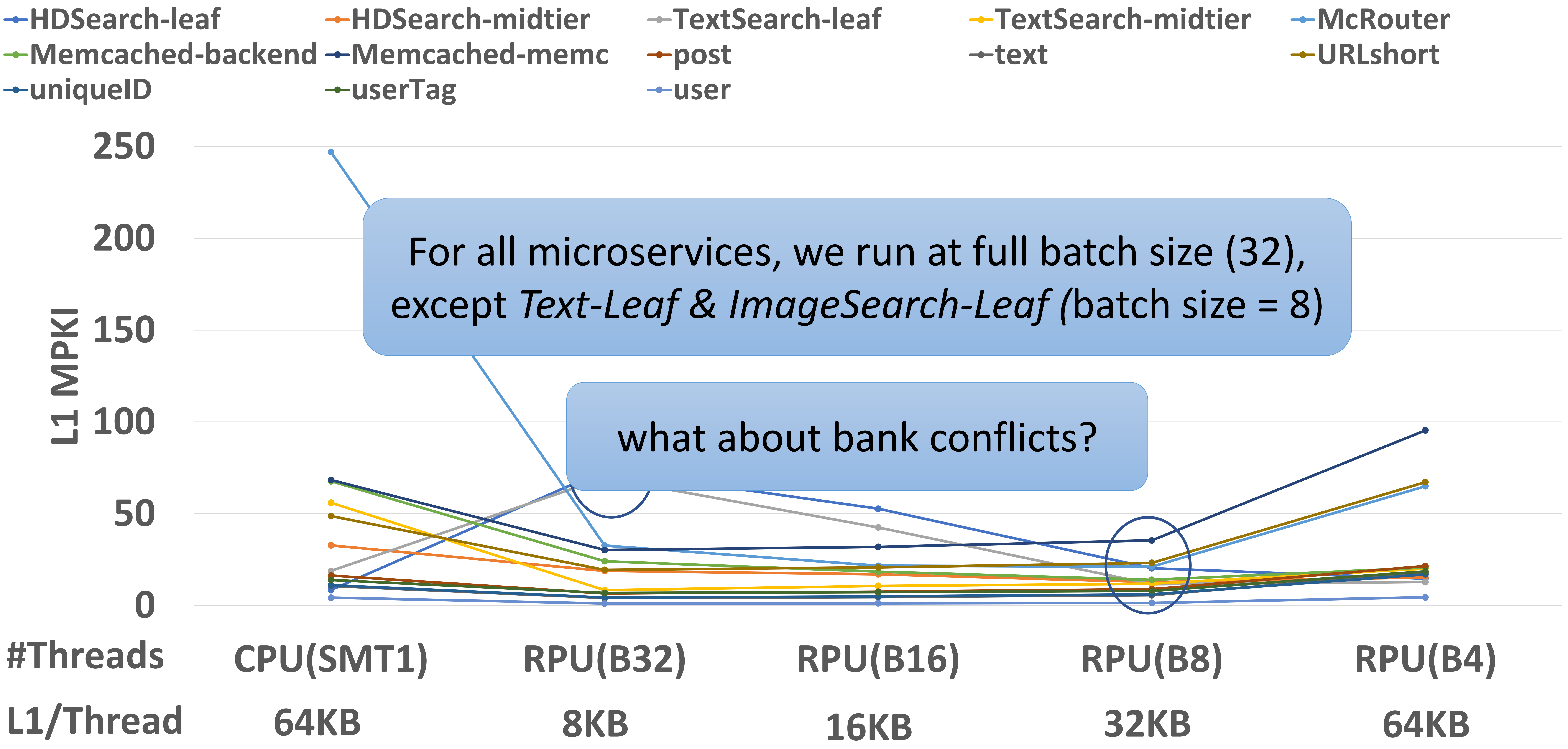


→ **4x traffic reduction compared to CPU**

Batch Size Tuning to Alleviate Cache Contention



Batch Size Tuning to Alleviate Cache Contention



For all microservices, we run at full batch size (32), except *Text-Leaf & ImageSearch-Leaf* (batch size = 8)

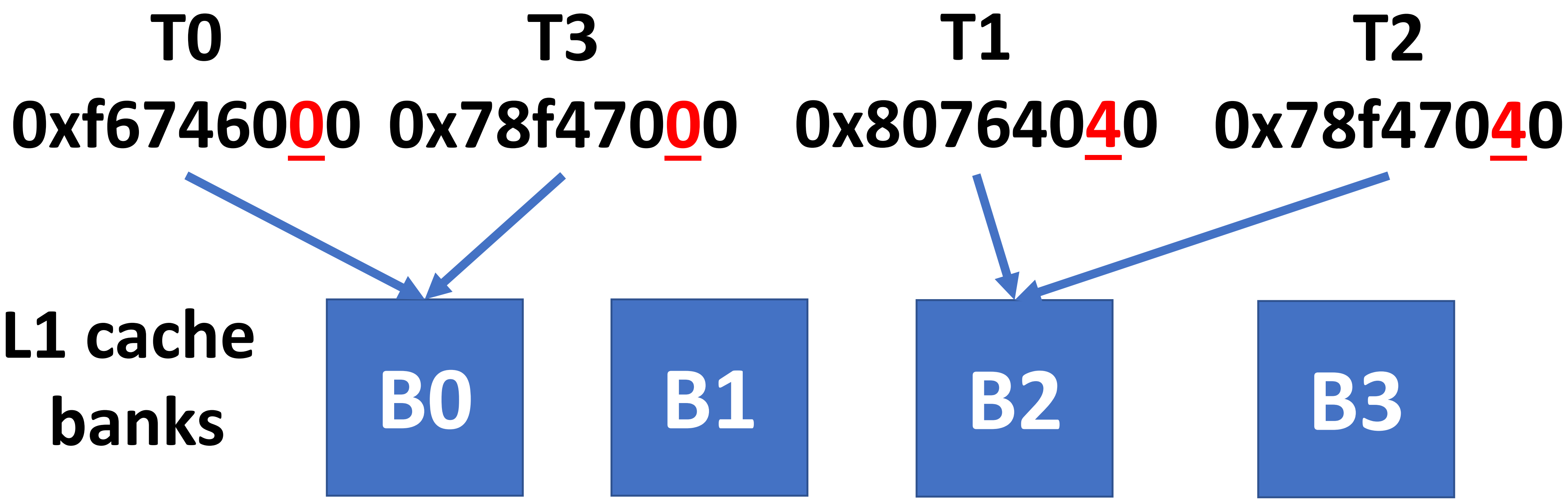
what about bank conflicts?

SIMT-Agnostic Memory Allocator

```
Microservice ()
  //Create a private temporary array in the
  // heap segment
  int* temp = new int[n];
   ...........
   for(int i=0; i<n; i++)
      temp[i] = i;      //Write to the temp
   ...........
for(int i=0; i<n; i++)
      sum += temp[i];  //Read from the temp
  ...........
```

Assume data are interleaved every 32B

temp **array address**



T0 0xf6746000

T3 0x78f47000

T1 0x80764040

T2 0x78f47040

L1 cache banks

B0 **B1** **B2** **B3**

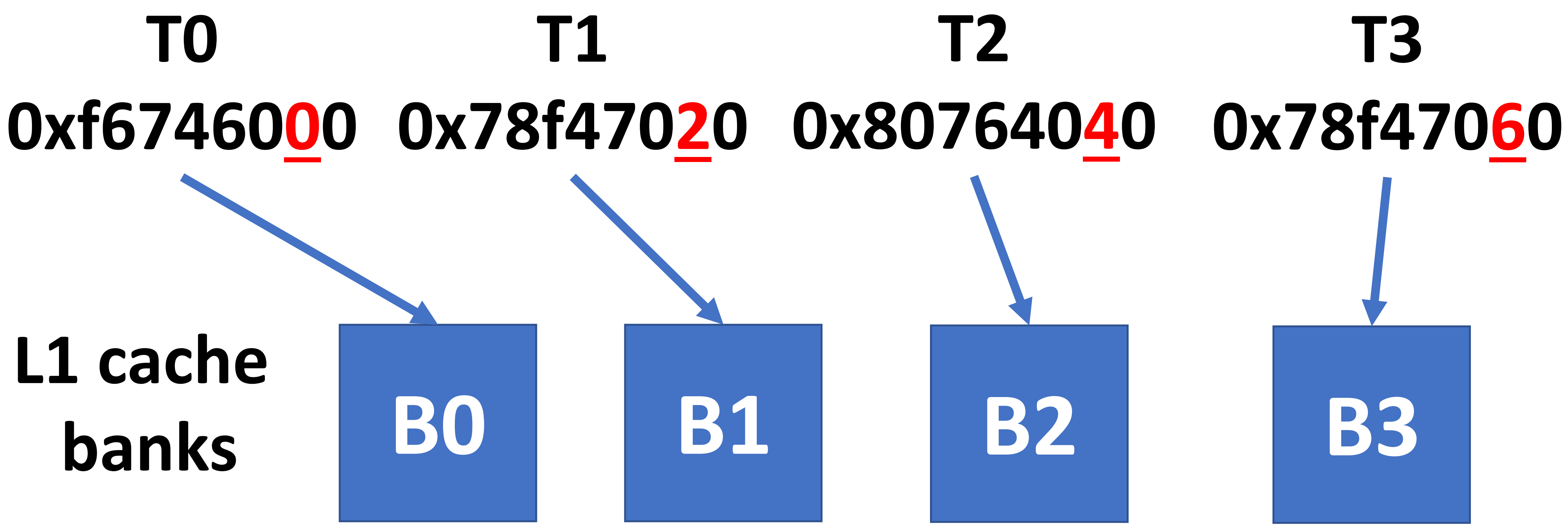
Severe Bank Conflicts

C++ SIMT-Agnostic Memory Allocator

SIMT-Aware Memory Allocator

```
Microservice ()
  //Create a private temporary array in the
  // heap segment
  int* temp = new int[n];
   ..........
   for(int i=0; i<n; i++)
      temp[i] = i;      //Write to the temp
   ..........
for(int i=0; i<n; i++)
      sum += temp[i];  //Read from the temp
 ..........
```

Assume data are interleaved every 32B



T0 0xf6746000

T1 0x78f47020

T2 0x80764040

T3 0x78f47060

L1 cache banks

B0 B1 B2 B3

Conflict-free

C++ SIMT-Aware Memory Allocator

→ ensures start_address%(n*tid) = 0

Evaluation

- Analytical Model
- Simulation-based evaluation
  - Chip-level evaluation
  - System-level evaluation

Energy Efficiency of CPU vs RPU (Analytical Model)

$$\frac{\boldsymbol{CPU\ Energy}}{\boldsymbol{RPU\ Energy}} = \frac{\boldsymbol{Execution\ Energy + Memory\ system\ Energy + Front\_OoO\ Energy + Static\ Energy}}{\boldsymbol{Execution\ Energy + (1-r)\,(Memory\ system\ Energy) + \frac{1}{n\,*\,eff}}\underbrace{\boldsymbol{[\,Front\_OoO\ Energy + r*Memory\ system\ Energy + Static\ Energy]}}_{\textit{Amortized factors = 50-80\%}}}$$

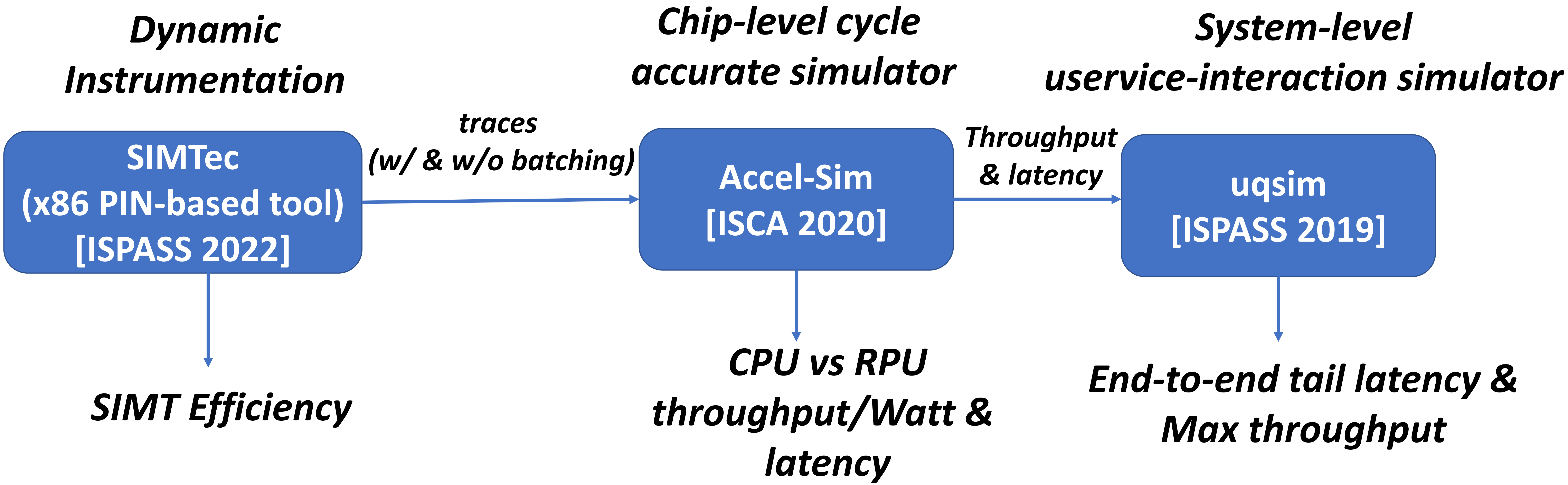
data locality ratio =75%

batch size (n) = 8-32

SIMT Efficiency=92%

→ *an anticipated 2-5x energy efficiency gain can be achieved with RPU vs CPU*

Experimental Setup



Dynamic Instrumentation

SIMTec (x86 PIN-based tool) [ISPASS 2022]

traces (w/ & w/o batching)

SIMT Efficiency

Chip-level cycle accurate simulator

Accel-Sim [ISCA 2020]

Throughput & latency

CPU vs RPU throughput/Watt & latency

System-level uservice-interaction simulator

uqsim [ISPASS 2019]

End-to-end tail latency & Max throughput

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, …

Khairy, Mahmoud, et al. "Accel-Sim: An extensible simulation framework for validated GPU modeling." *ISCA 2020*
Zhang, Yanqi, Yu Gan, and Christina Delimitrou. "uqSim: Scalable and Validated Simulation of Cloud Microservices." ISPASS 2019
Alawneh, Ahmad , et al. "A SIMT Analyzer for Multi-Threaded CPU Applications." ISPASS 2022
Sriraman, Akshitha, and Thomas F. Wenisch. "μ suite: a benchmark suite for microservices." IISWC 2018
Gan, Yu, et al. "An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems.“ ASPLOS 2019

Simulation Configuration

- Baseline: Single threaded CPU and SMT8 CPU
- RPU: SIMT-32 (1 batch)

- We ensure both CPU and RPU have the same pipeline configuration, frequency, and memory resources/thread for SMT8 and our RPU

- CPU & RPU are TDP equivalent at the same technology node

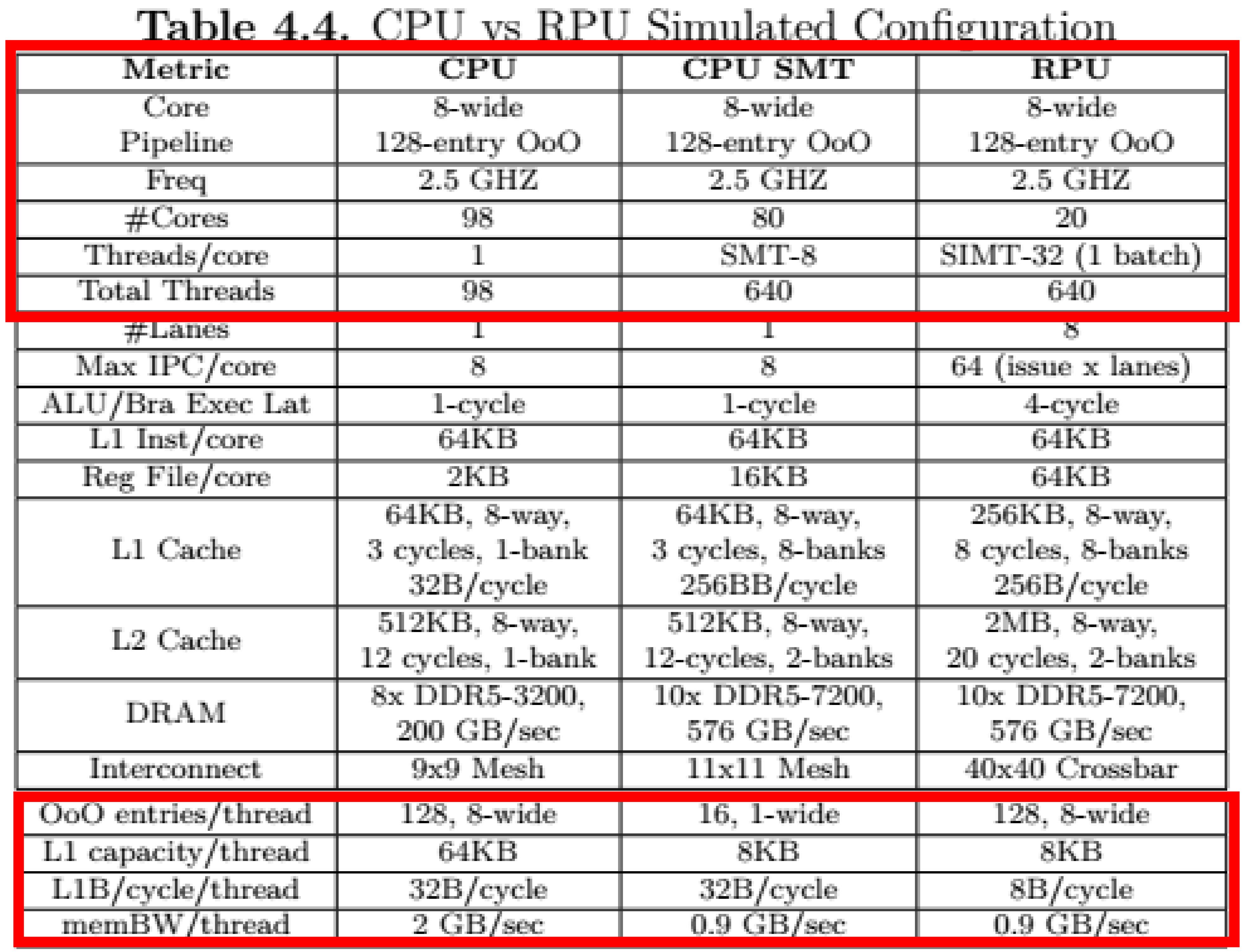
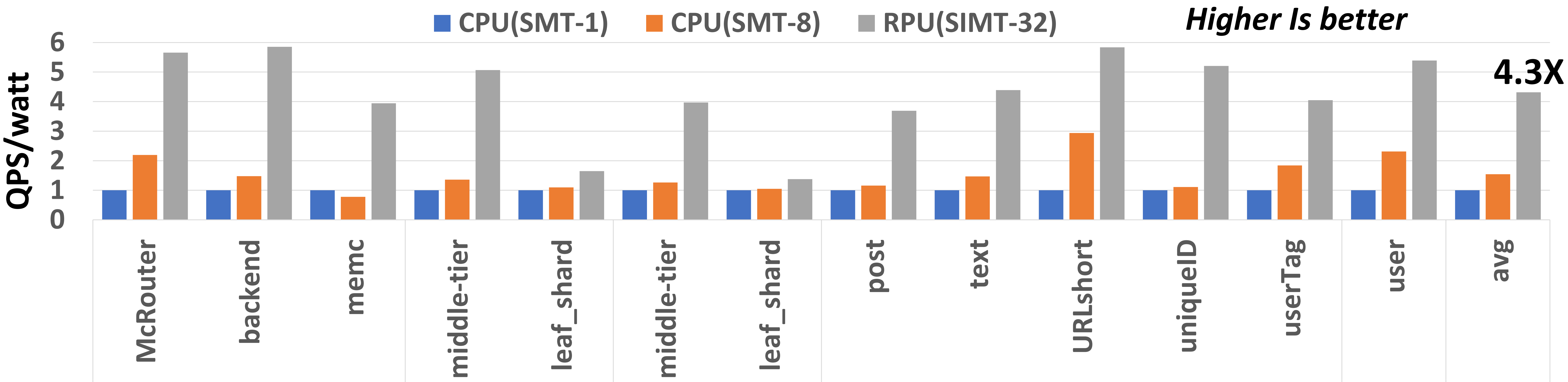


Table 4.4. CPU vs RPU Simulated Configuration

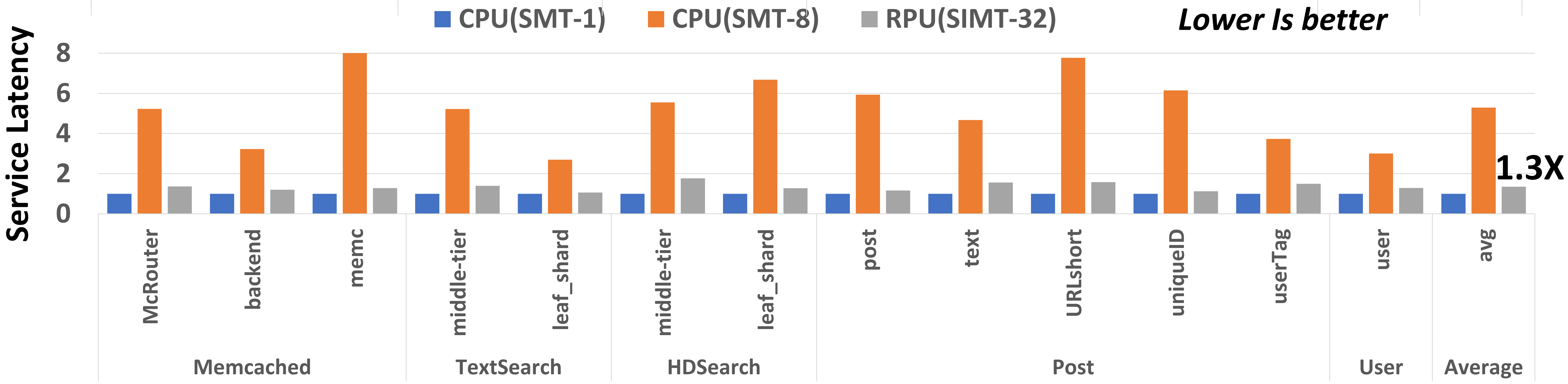
Metric	CPU	CPU SMT	RPU
Core Pipeline	8-wide 128-entry OoO	8-wide 128-entry OoO	8-wide 128-entry OoO
Freq	2.5 GHZ	2.5 GHZ	2.5 GHZ
#Cores	98	80	20
Threads/core	1	SMT-8	SIMT-32 (1 batch)
Total Threads	98	640	640
#Lanes	1	1	8
Max IPC/core	8	8	64 (issue x lanes)
ALU/Bra Exec Lat	1-cycle	1-cycle	4-cycle
L1 Inst/core	64KB	64KB	64KB
Reg File/core	2KB	16KB	64KB
L1 Cache	64KB, 8-way, 3 cycles, 1-bank 32B/cycle	64KB, 8-way, 3 cycles, 8-banks 256BB/cycle	256KB, 8-way, 8 cycles, 8-banks 256B/cycle
L2 Cache	512KB, 8-way, 12 cycles, 1-bank	512KB, 8-way, 12-cycles, 2-banks	2MB, 8-way, 20 cycles, 2-banks
DRAM	8x DDR5-3200, 200 GB/sec	10x DDR5-7200, 576 GB/sec	10x DDR5-7200, 576 GB/sec
Interconnect	9x9 Mesh	11x11 Mesh	40x40 Crossbar
OoO entries/thread	128, 8-wide	16, 1-wide	128, 8-wide
L1 capacity/thread	64KB	8KB	8KB
L1B/cycle/thread	32B/cycle	32B/cycle	8B/cycle
memBW/thread	2 GB/sec	0.9 GB/sec	0.9 GB/sec

Chip-level Results (Accel-Sim Simulation)



Higher Is better

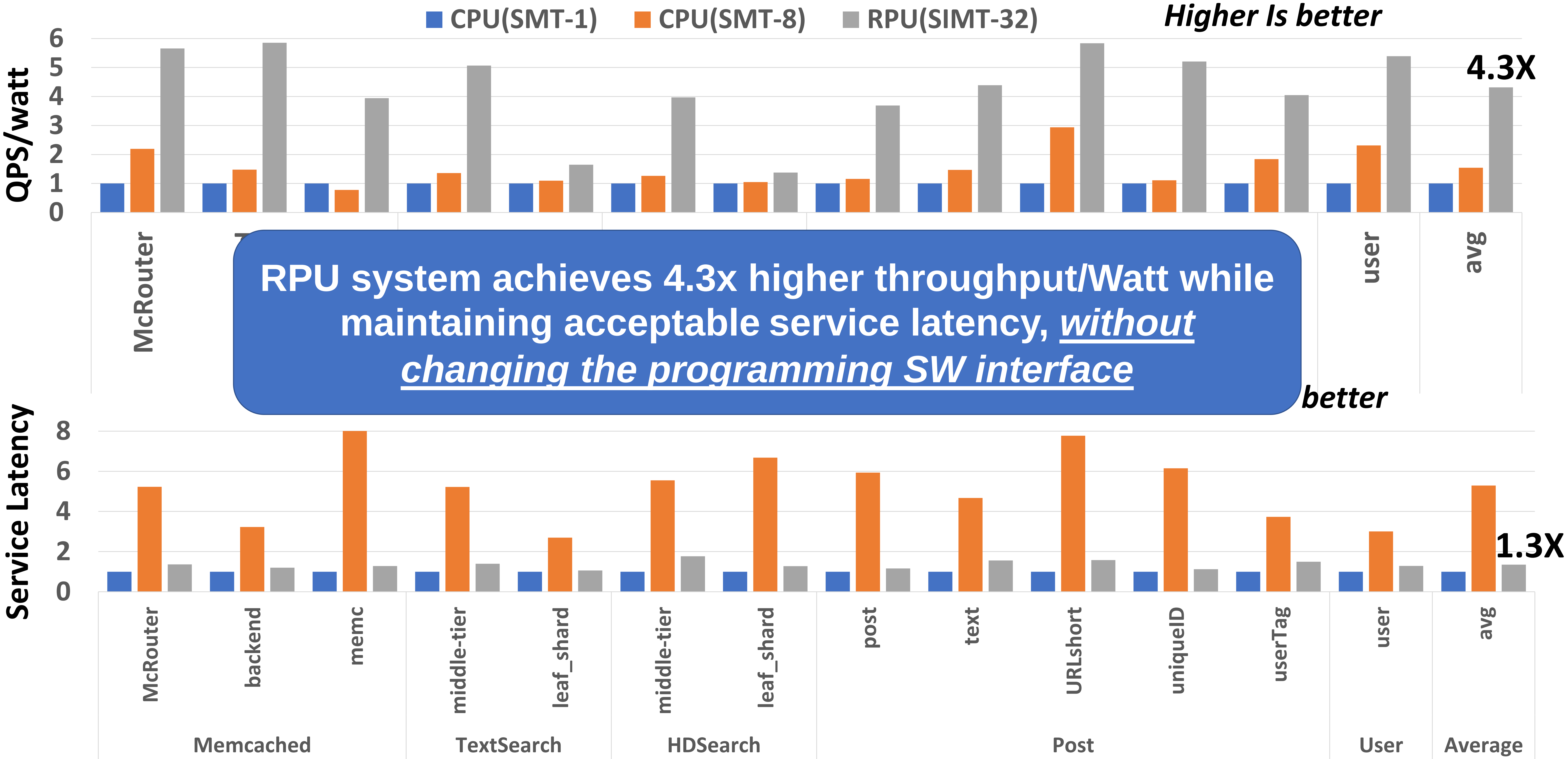
4.3X



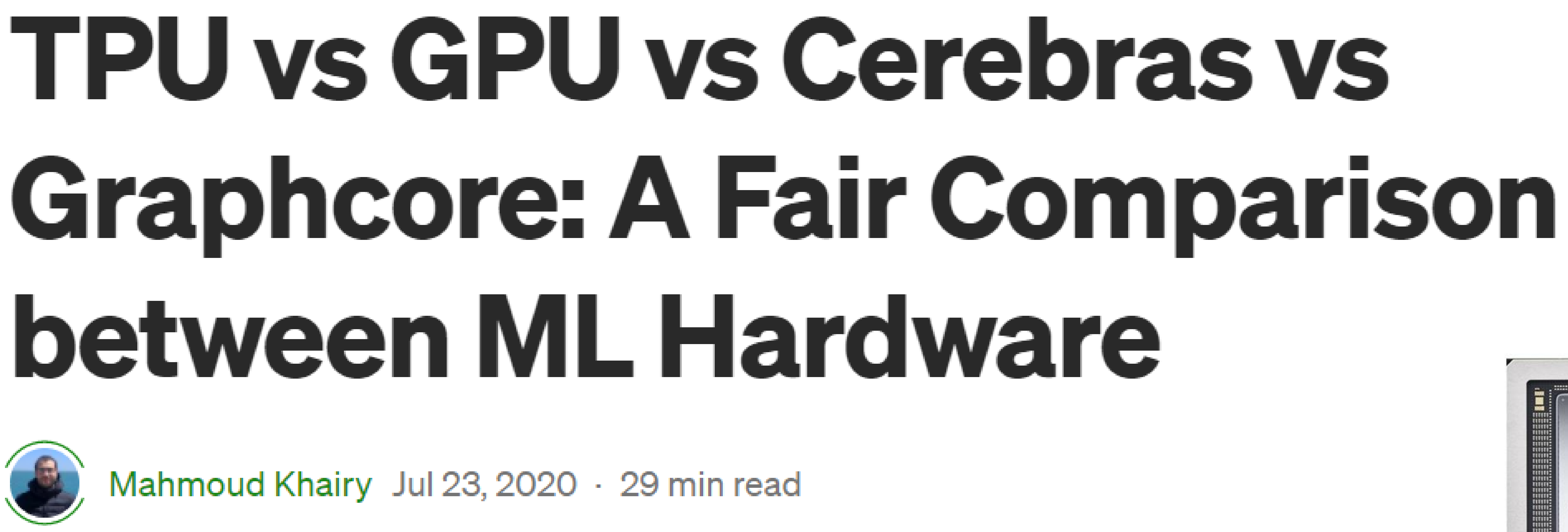
Lower Is better

1.3X

Chip-level Results (Accel-Sim Simulation)

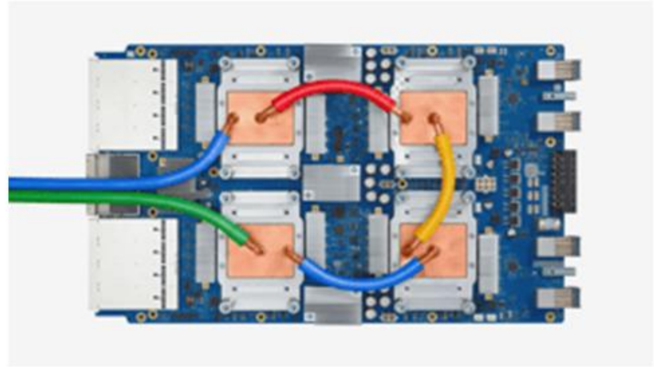


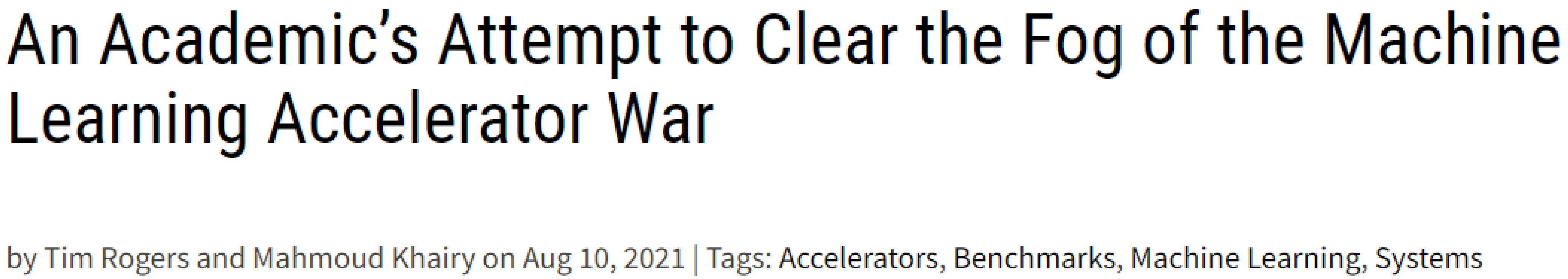
RPU system achieves 4.3x higher throughput/Watt while maintaining acceptable service latency, ***without changing the programming SW interface***



TPU vs GPU vs Cerebras vs Graphcore: A Fair Comparison between ML Hardware

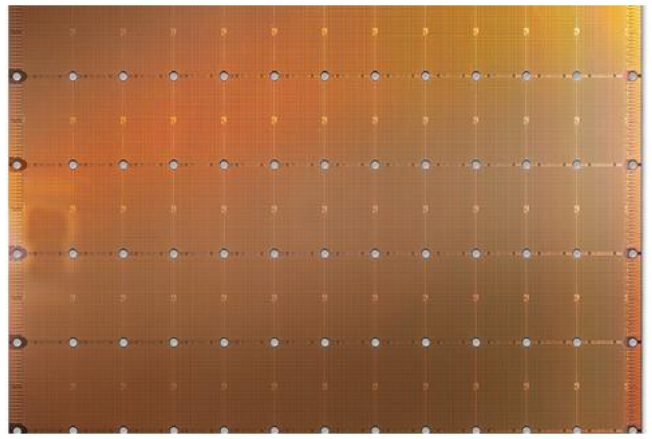
Mahmoud Khairy Jul 23, 2020 · 29 min read





An Academic's Attempt to Clear the Fog of the Machine Learning Accelerator War

by Tim Rogers and Mahmoud Khairy on Aug 10, 2021 | Tags: Accelerators, Benchmarks, Machine Learning, Systems



https://khairy2011.medium.com/tpu-vs-gpu-vs-cerebras-vs-graphcore-a-fair-comparison-between-ml-hardware-3f5a19d89e38
https://www.sigarch.org/an-academics-attempt-to-clear-the-fog-of-the-machine-learning-accelerator-war/

ML Hardware Startup Explosion

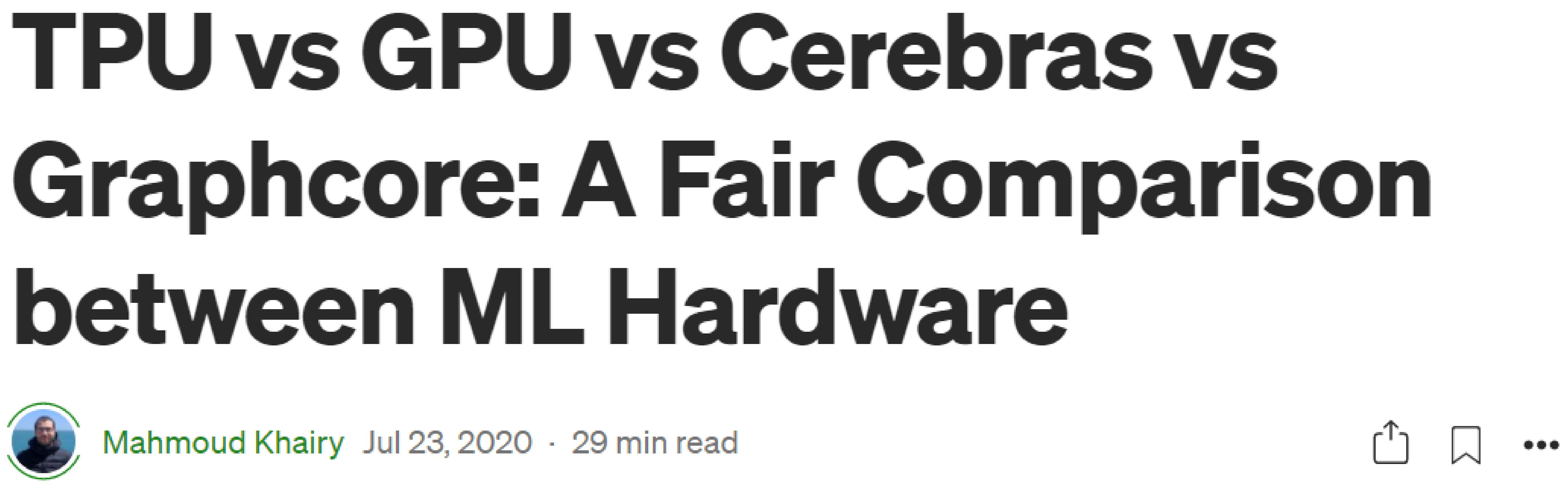
- 1.2B investment in 2017
- AI chip market is anticipated to be 90B in 2025 (train + inference)

NVIDIA
AMD
intel
Nervana
habana
An Intel Company
cerebras
Google
groq
GRAPHCORE
SambaNova
SYSTEMS
HUAWEI
Tenstorrent
CORNAMI
Intelligent Computing

How to Fairly Evaluate Existing Solutions?

- MLPerf only shows training time (i.e. performance), which is tricky!
- Proposed Solution:
    - Apples-to-apples comparison
    - Focusing on efficiency metrics
        - Performance per Dollar per Watt per Unit
    - Trying to reduce the batch size effect
    - Design philosophy (Data vs Model parallelism)

Read More Details in the Article



TPU vs GPU vs Cerebras vs Graphcore: A Fair Comparison between ML Hardware

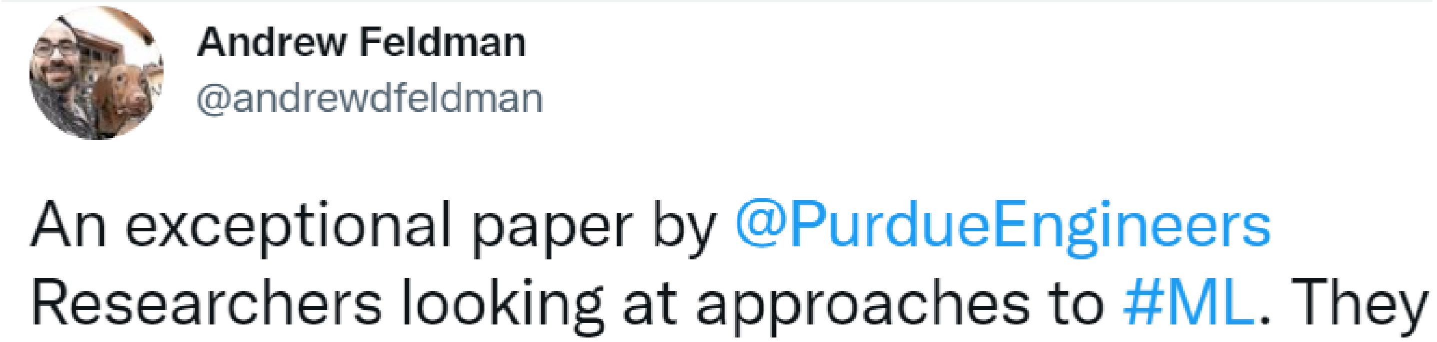
Mahmoud Khairy Jul 23, 2020 · 29 min read

https://khairy2011.medium.com/tpu-vs-gpu-vs-cerebras-vs-graphcore-a-fair-comparison-between-ml-hardware-3f5a19d89e38

Recognitions & Acknowledgements

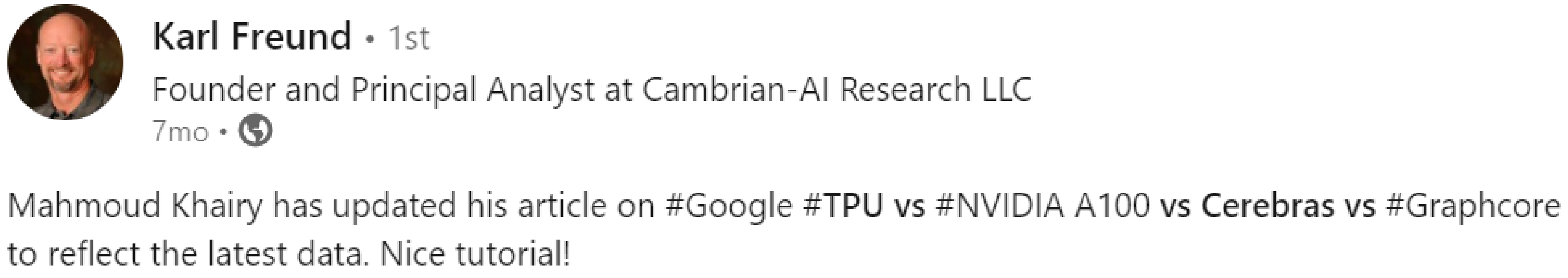
David PATTERSON

Nice article!



Andrew Feldman
@andrewdfeldman

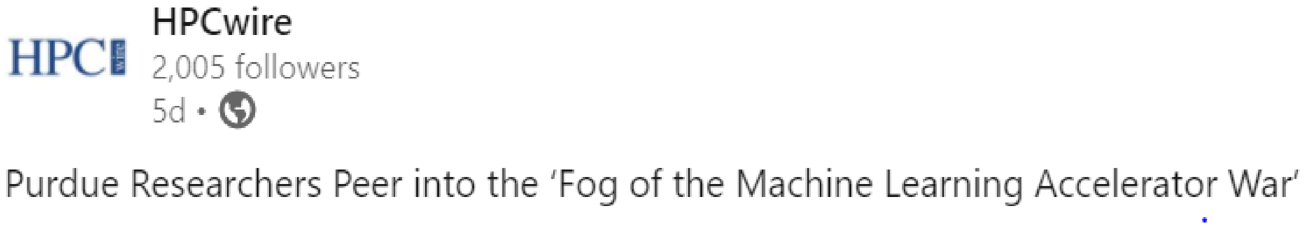
An exceptional paper by @PurdueEngineers Researchers looking at approaches to #ML. They



Karl Freund • 1st
Founder and Principal Analyst at Cambrian-AI Research LLC
7mo •

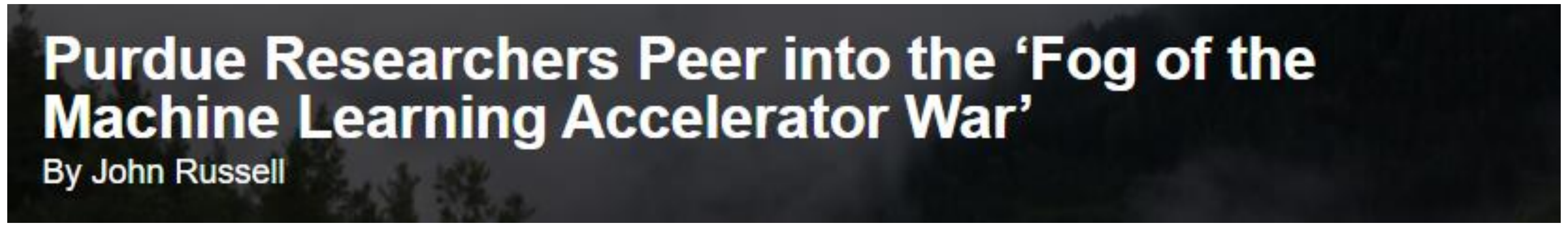
Mahmoud Khairy has updated his article on #Google #**TPU** **vs** #NVIDIA A100 **vs Cerebras vs** #Graphcore to reflect the latest data. Nice tutorial!

Mike Mantor · 1st
Corporate Fellow

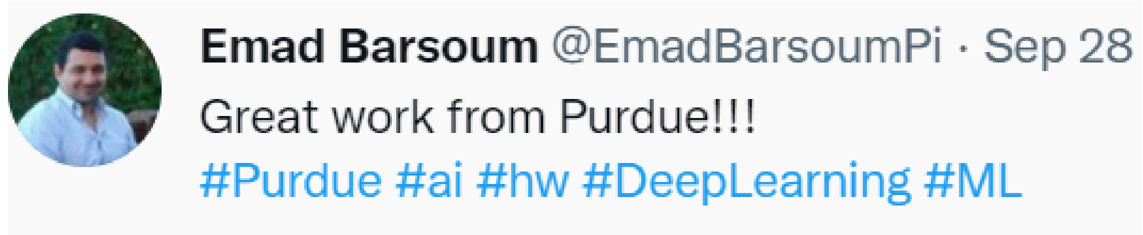


HPCwire
2,005 followers
5d •

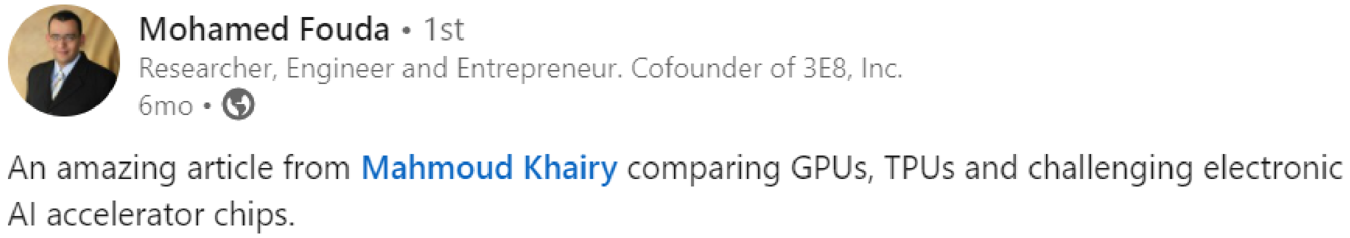
Purdue Researchers Peer into the 'Fog of the Machine Learning Accelerator War'



Purdue Researchers Peer into the 'Fog of the Machine Learning Accelerator War'
By John Russell

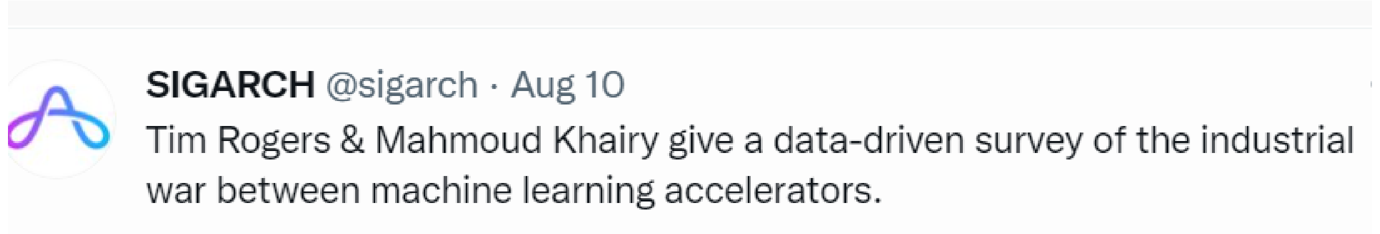


Emad Barsoum @EmadBarsoumPi · Sep 28
Great work from Purdue!!!
#Purdue #ai #hw #DeepLearning #ML



Mohamed Fouda • 1st
Researcher, Engineer and Entrepreneur. Cofounder of 3E8, Inc.
6mo •

An amazing article from Mahmoud Khairy comparing GPUs, TPUs and challenging electronic AI accelerator chips.



SIGARCH @sigarch · Aug 10
Tim Rogers & Mahmoud Khairy give a data-driven survey of the industrial war between machine learning accelerators.

HPC wire
Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

Cambrian AI Research
explore the explosion

Morgan Lewis

Sutter Hill Ventures

MLPerf

cerebras

AMD

Google

Other Publications

- Vijay Kandiah, Scott Peverelle, **Mahmoud Khairy**, Amogh Manjunath, Junrui Pan, Timothy G. Rogers, Tor Aamodt, Nikos Hardavellas "AccelWattch: A Power Modeling Framework for Modern GPUs." **MICRO 2021**
- Cesar Avalos, **Mahmoud Khairy**, Roland N. Green, Mathias Payer, Timothy G. Rogers. "Principal Kernel Analysis: A Tractable Methodology to Simulate Scaled GPU Workloads." **MICRO 2021**
- Jain Akshay*, **Mahmoud Khairy***, Timothy G. Rogers, *First Coauthors "A Quantitative Evaluation of Contemporary GPU Simulation Methodology." **SIGMETRICS 2018**
- **……..**

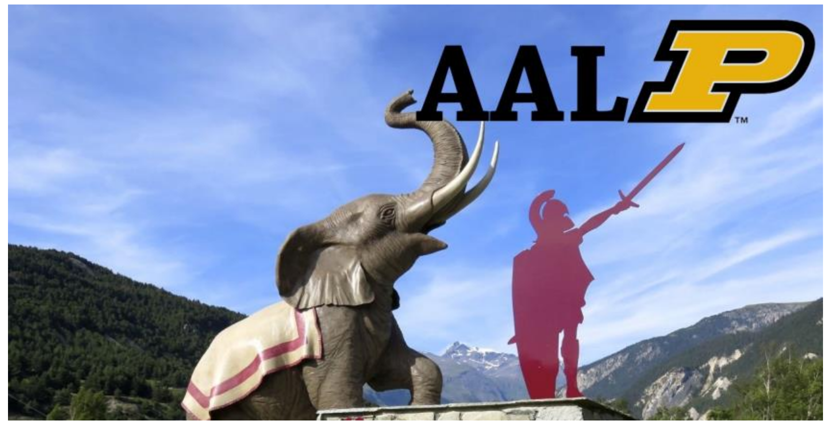
Conclusions

- SIMT-based accelerators, like GPUs and RPUs, are promising solutions to achieve significant *energy efficiency* in the data centers while still preserving *programmability*.

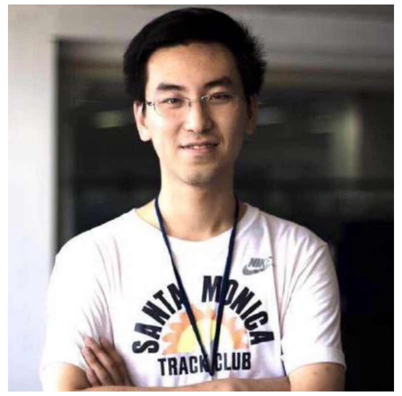
- Challenges:
  - (1) How to overcome the non-uniform memory access overhead for next-generation multi-chiplet GPUs in the era of ML-driven workloads?
  - (2) How to improve the energy efficiency of data center's CPUs in the light of microservices evolution?

- Moving forward, studying the feasibility of RPU architecture and prototyping is an important area of research.

Tim Rogers



Accelerator Architecture Lab at Purdue



Mengchi Zhang

Aaron Barnes

Akshay Jain

Tsung Tai

Yechen Liu

Roland Green

Christin Bose

Ahmad Alawneh

Cesar Avalos

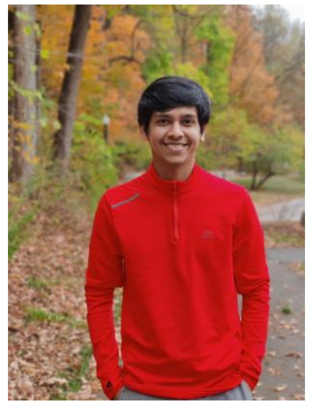
Ni Kang

Junrui Pan

Fanjia Shen

Vadim Nikiforov

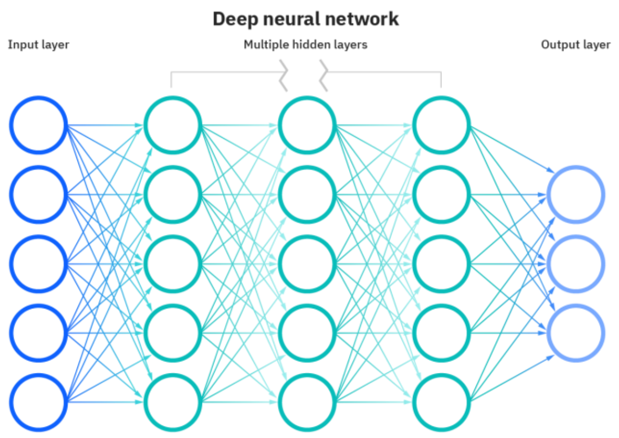
Zhesheng Shen



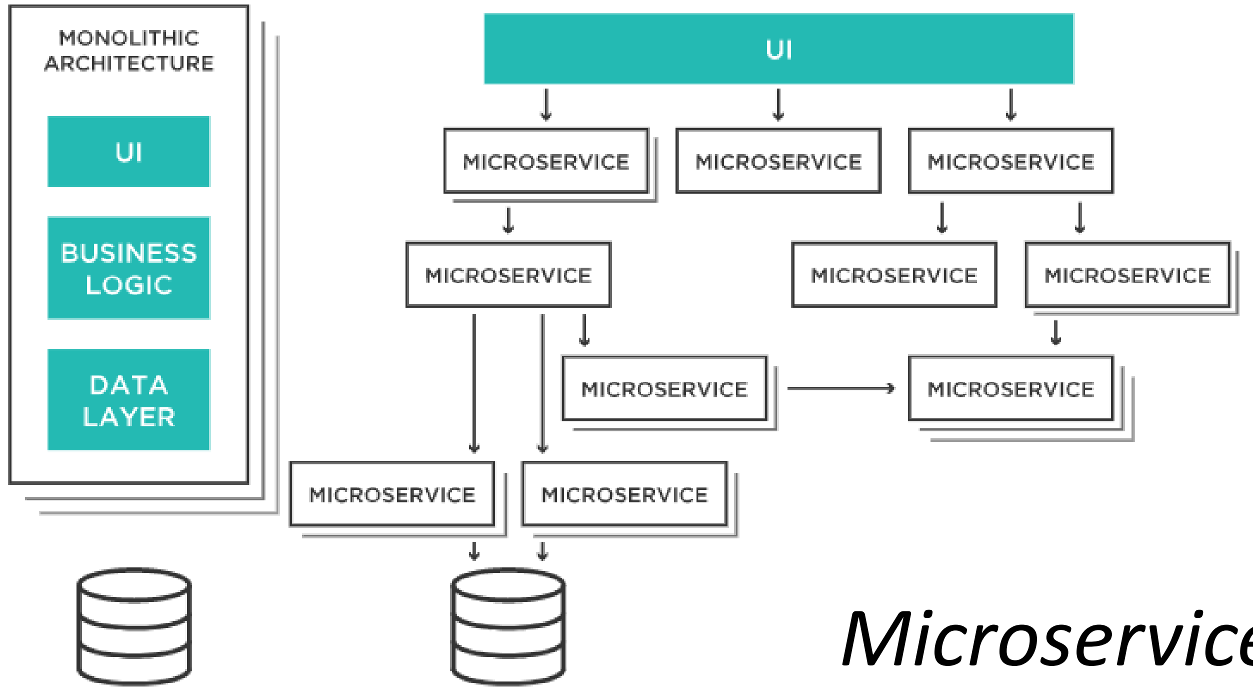
Abhishek Bhaumick

Thank You!
Q&A?

Software



Deep Learning



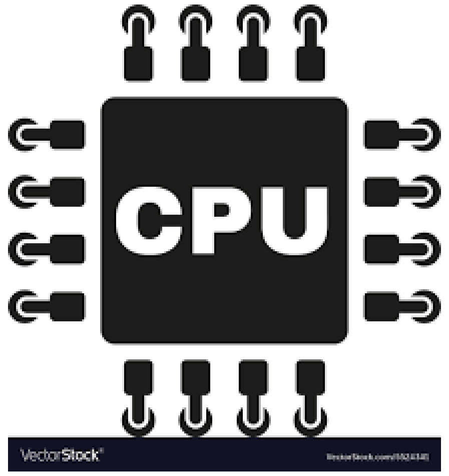
mongoDB

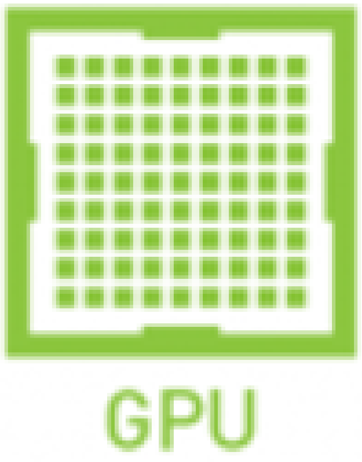
MEMCACHED

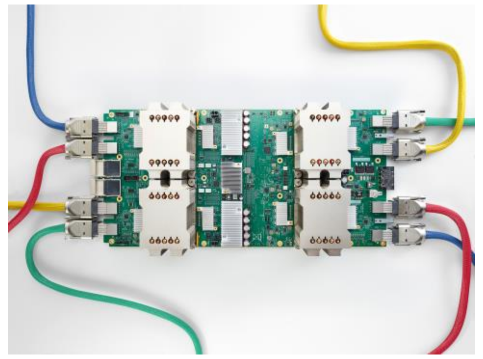
gRPC

Microservices

Hardware

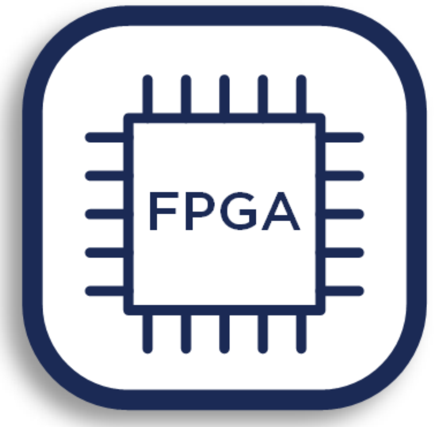


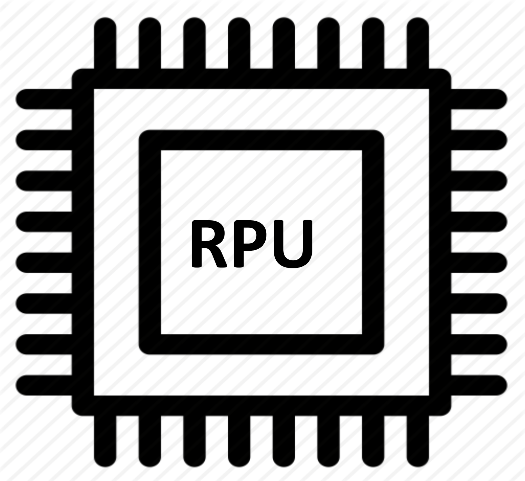




TPU

VCU





Accelerators