AccelFlow: Orchestrating an On-Package Ensemble of Fine-Grained Accelerators for Microservices

HPCA 2026, Sydney, Australia

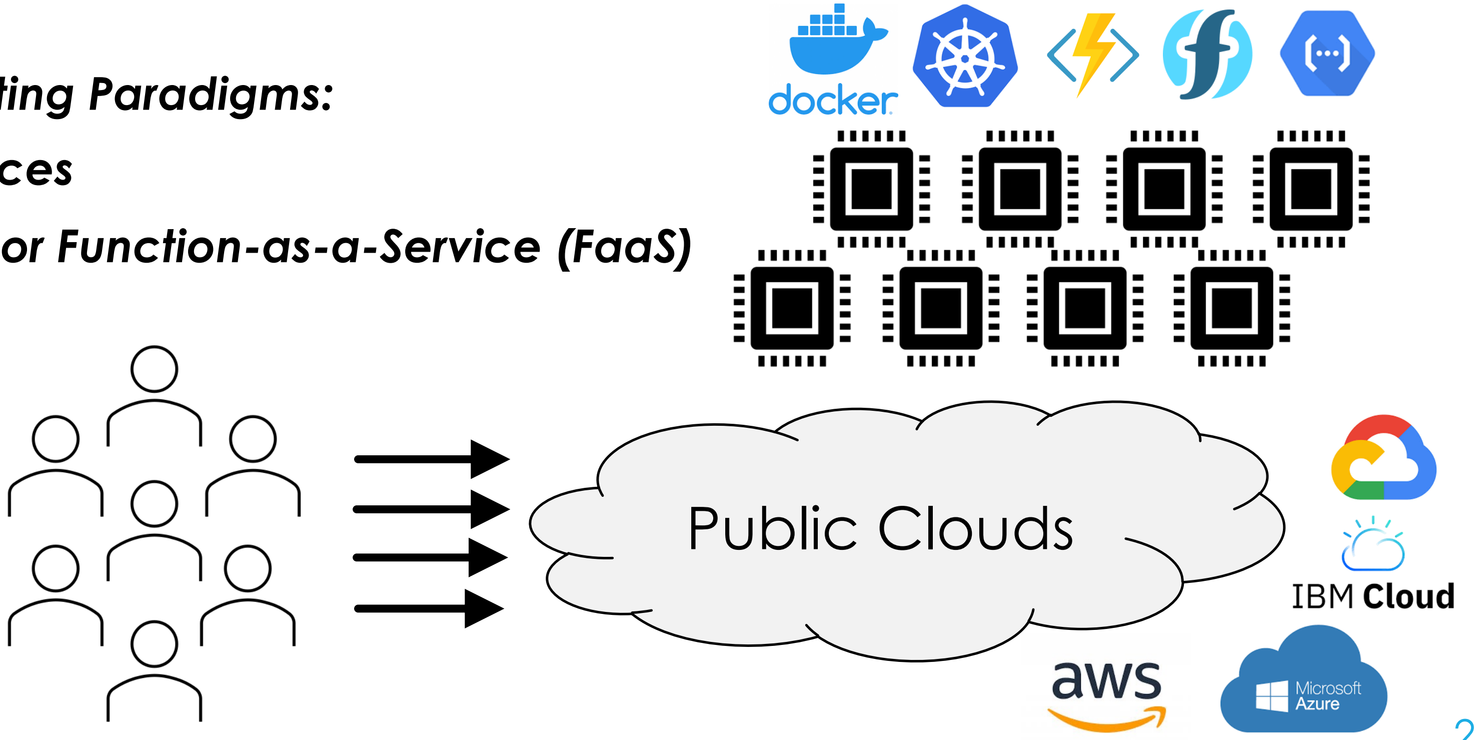
Jovan Stojkovic*, Abraham Farrell, Zhangxiaowen Gong[†], Christopher J. Hughes[†], Josep Torrellas

University of Illinois at Urbana-Champaign, [†]Intel, *Joining UT Austin in Fall 2026

The Growth of Cloud Computing

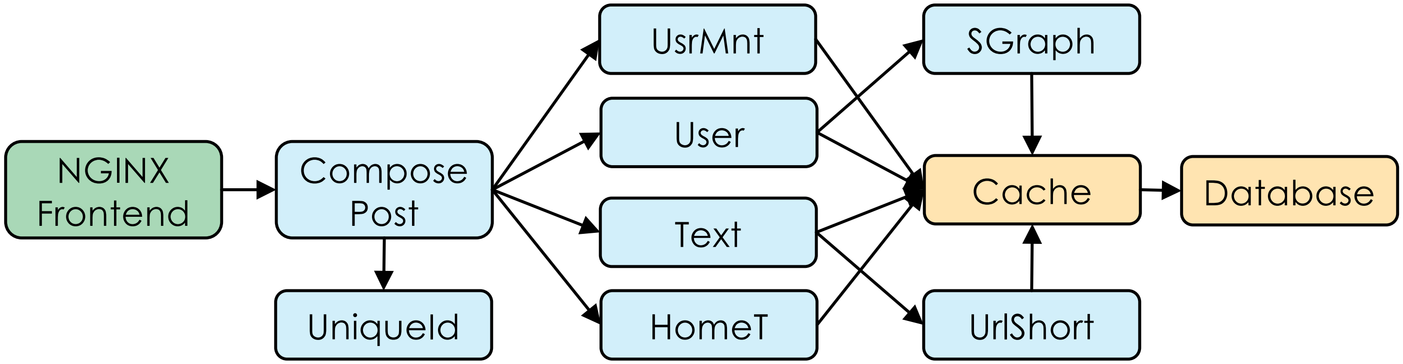
New Computing Paradigms:

- ***Microservices***
- ***Serverless or Function-as-a-Service (FaaS)***



Microservices

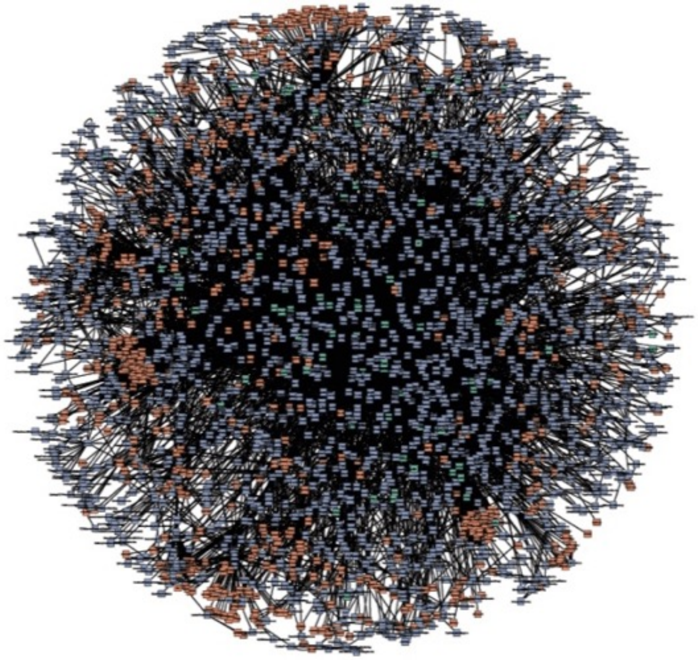
- Large monolithic applications decomposed into many small interdependent services
  - Each service implements separate functionality

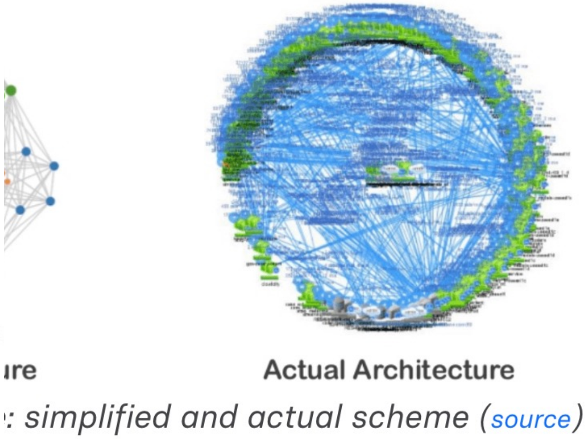


Benefits of Microservices

- Scalability ☑
- Design simplicity ☑
- HW management ☑

Microservices are Widely Used

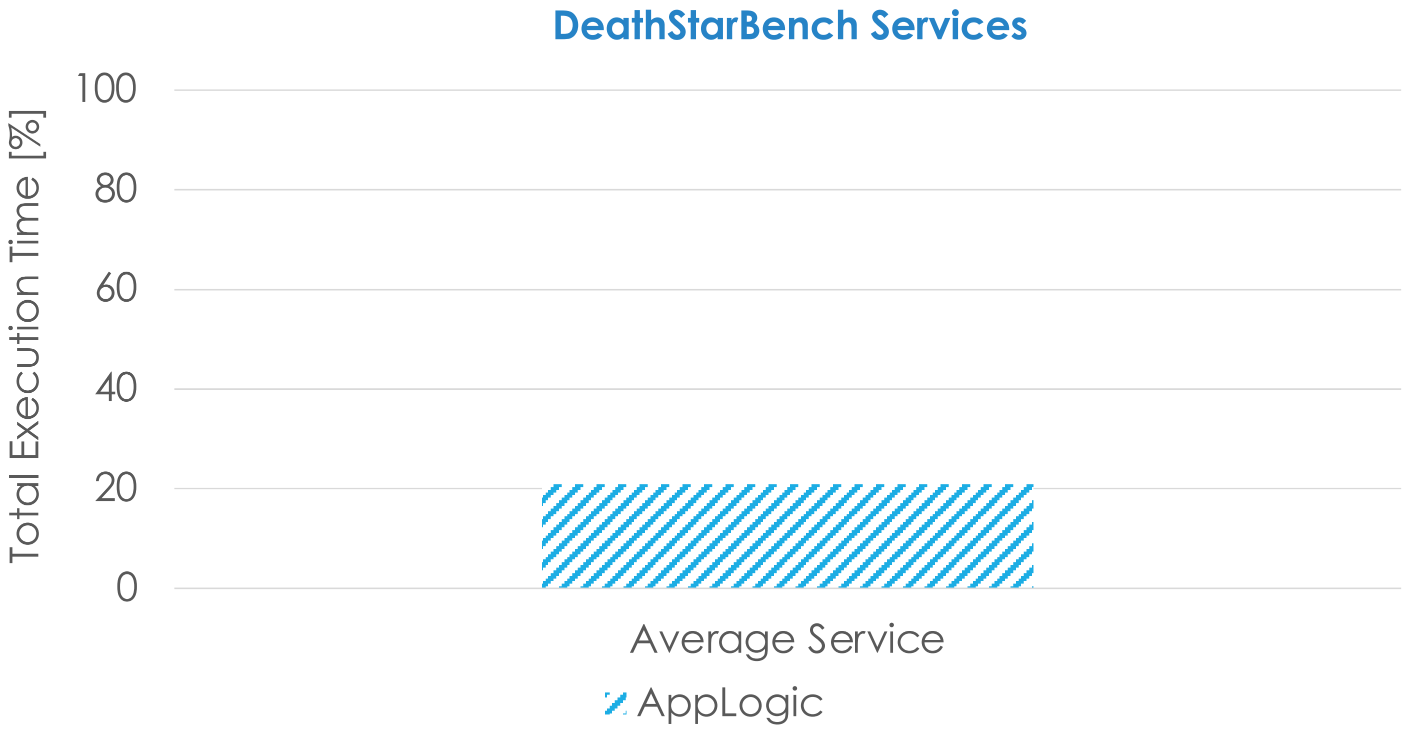




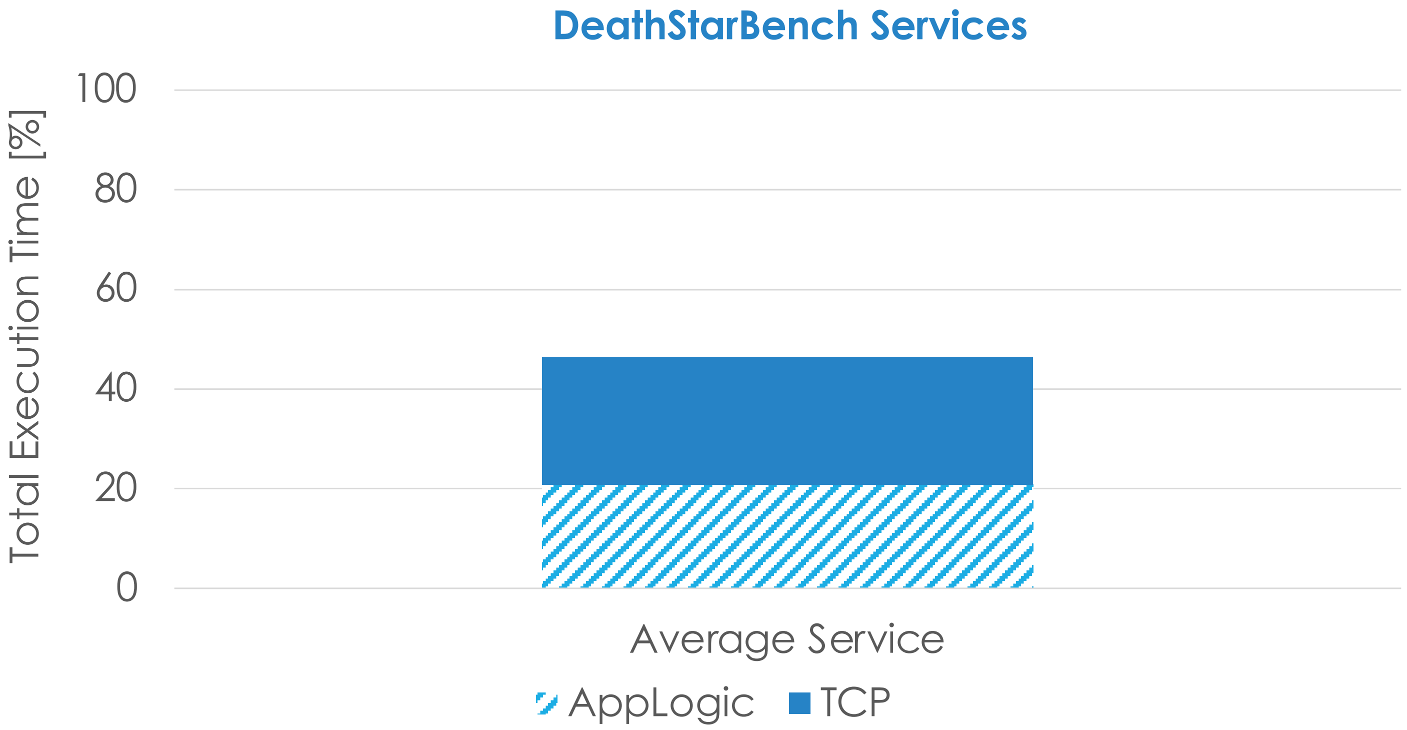
: simplified and actual scheme (source)

Structure of microservices at Amazon. Looks almost like a Death Star but is way more powerful.

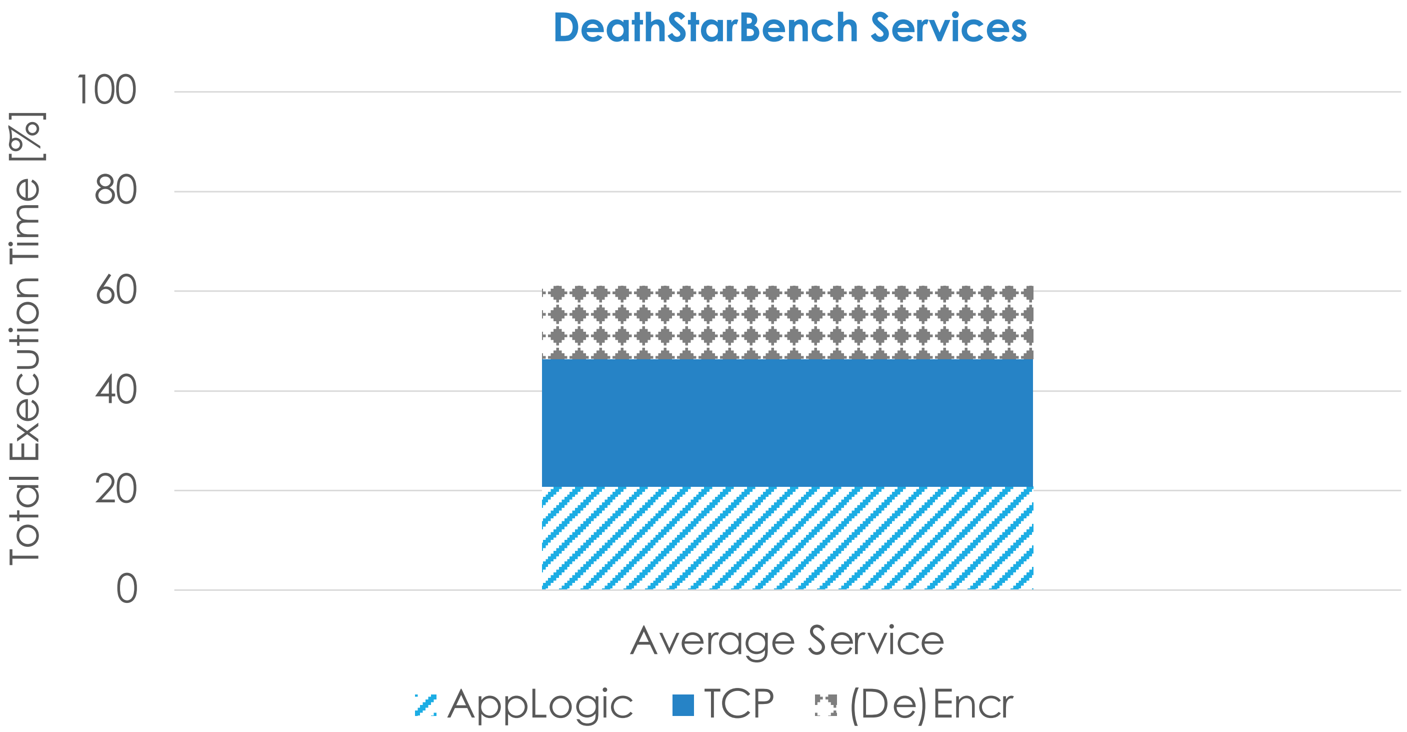
Datacenter Tax Dominates Execution



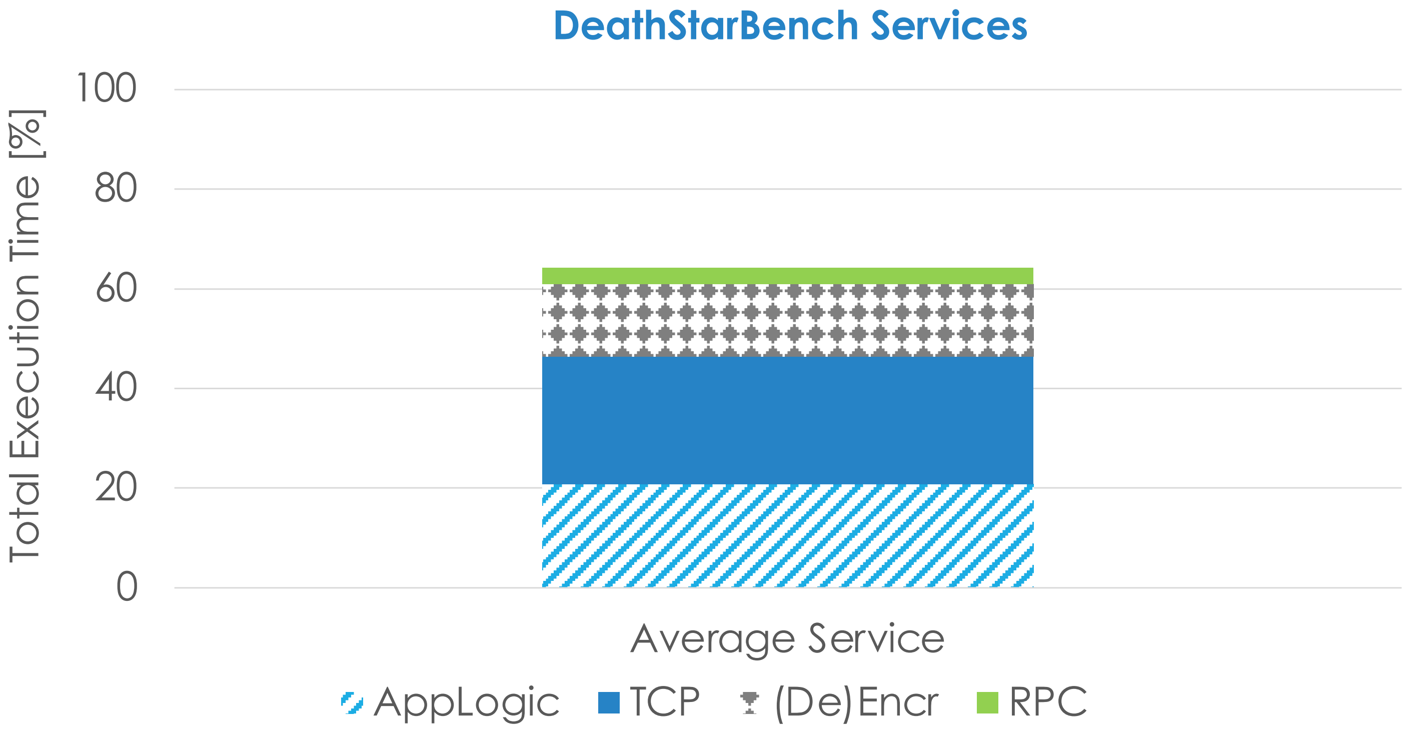
Datacenter Tax Dominates Execution



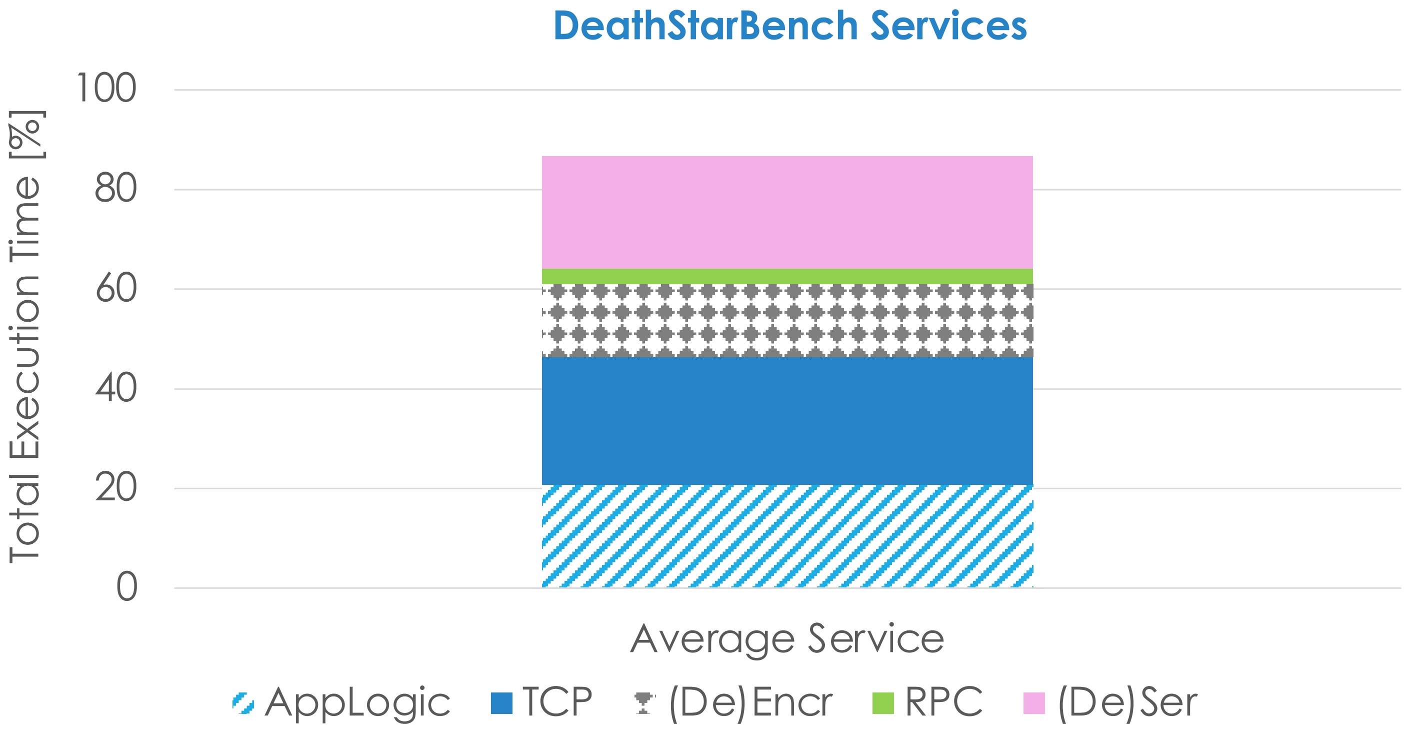
Datacenter Tax Dominates Execution



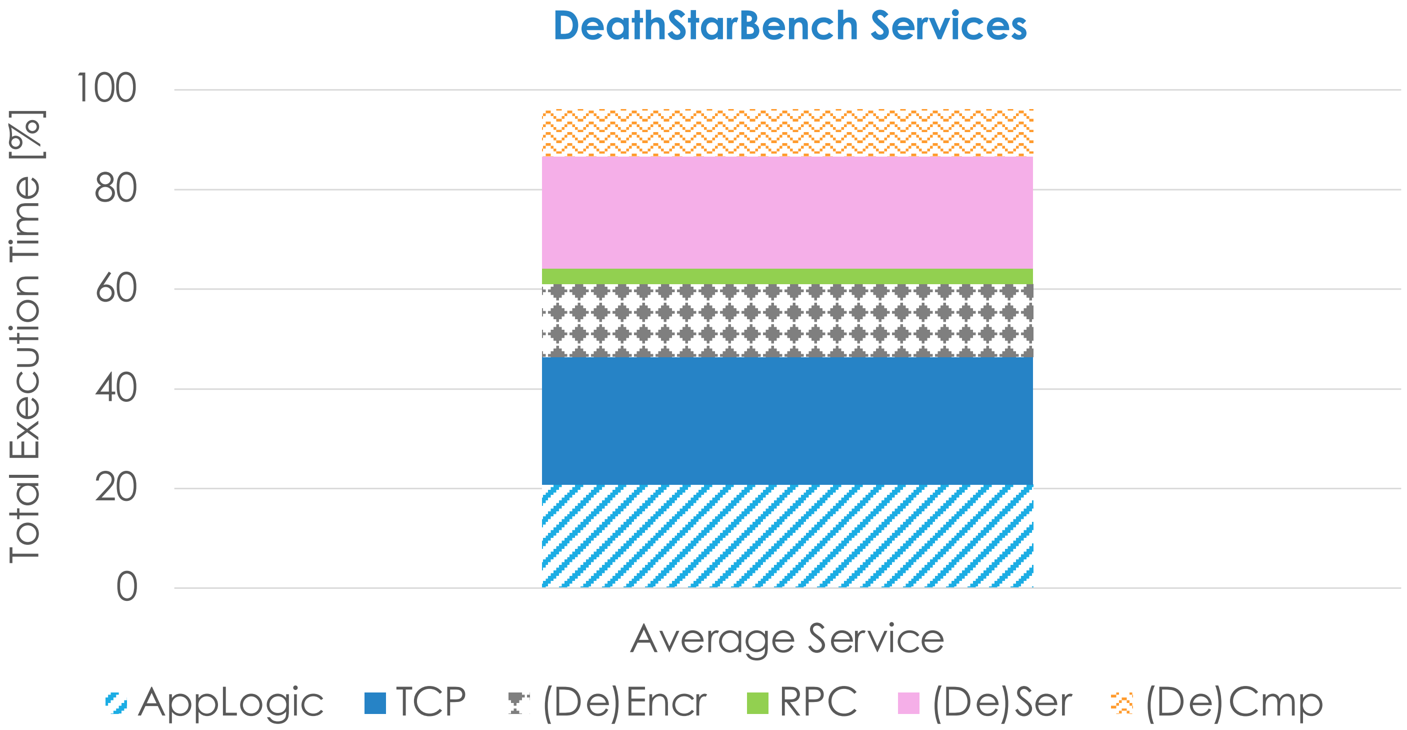
Datacenter Tax Dominates Execution



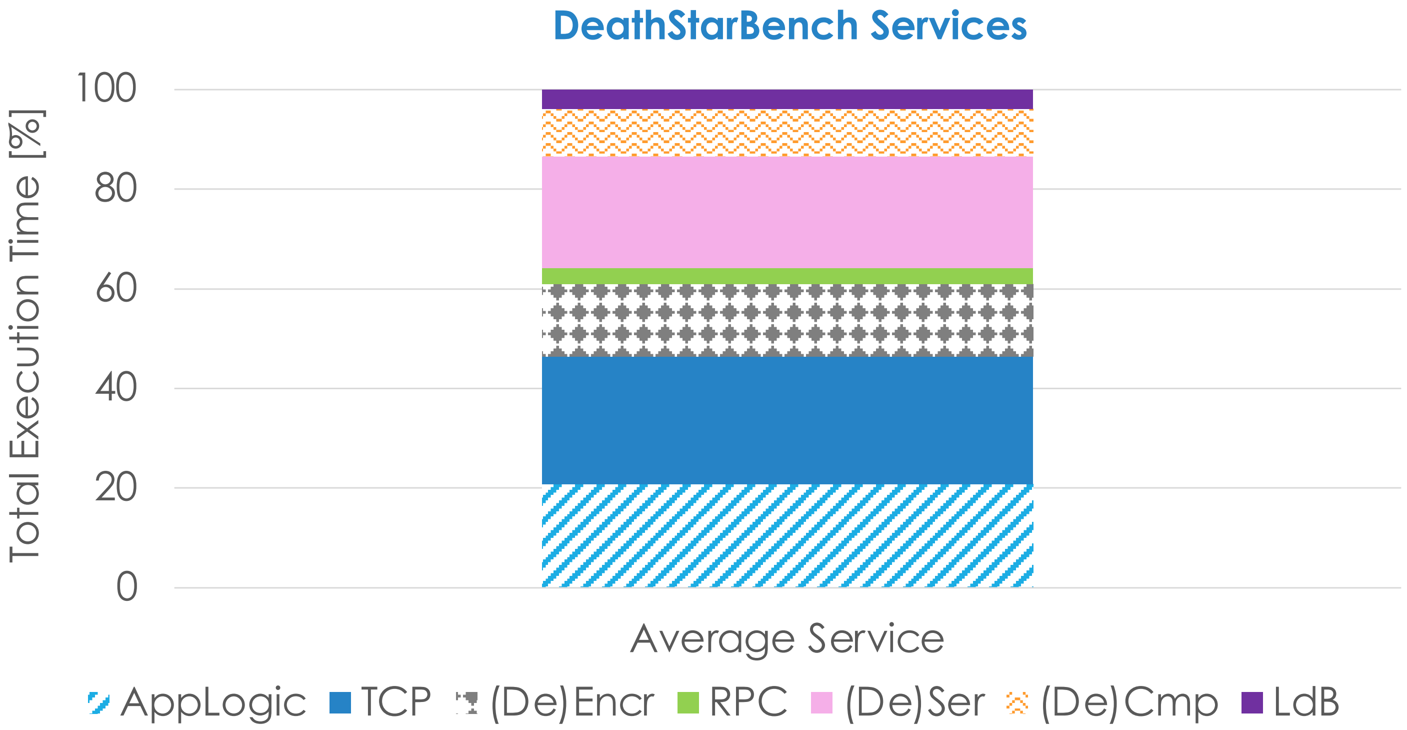
Datacenter Tax Dominates Execution



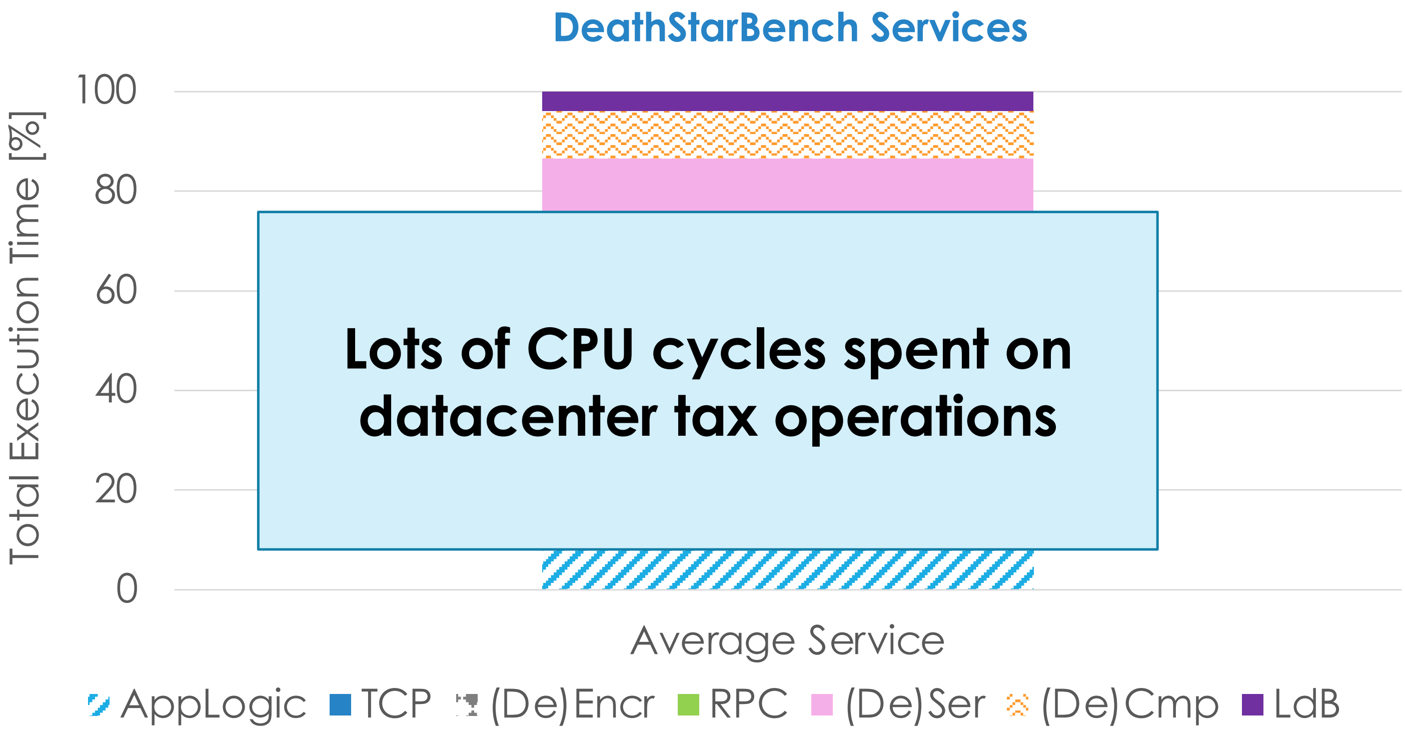
Datacenter Tax Dominates Execution



Datacenter Tax Dominates Execution



Datacenter Tax Dominates Execution



Lots of CPU cycles spent on datacenter tax operations

Datacenter Tax Reported by Major Hyperscalers

Google 2015

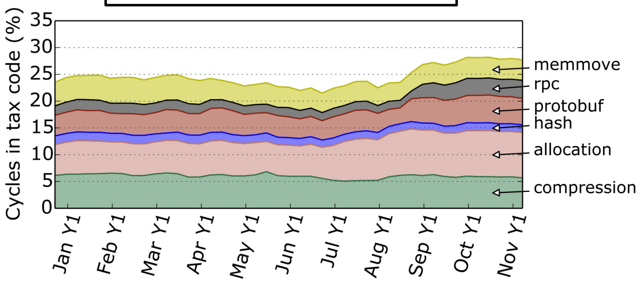


Figure 4: 22-27% of WSC cycles are spent in different components of "datacenter tax".

Facebook 2020

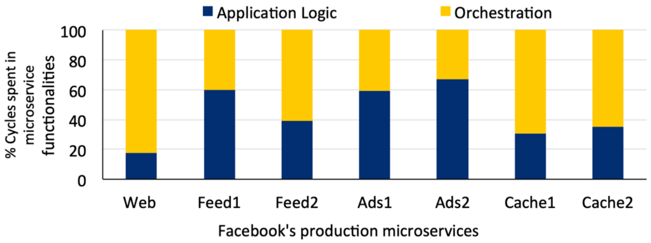


Figure 1. Breakdown of cycles spent in core application logic vs. orchestration work: orchestration overheads can significantly dominate.

Google 2023

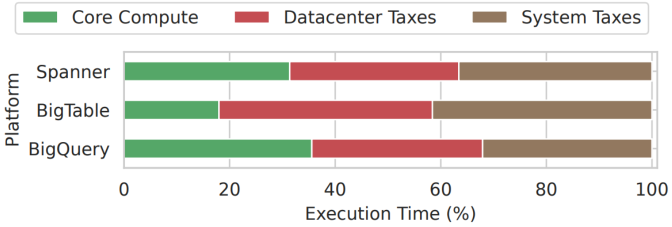


Figure 3: High-Level Application-Level Cycle Breakdown

Many Proposals for Individual Accelerators

Intel's SoC

CDPU: Co-designing Compression and
Processing Units for Hyperscale

Sagar Karandikar
UC Berkeley, Google
Berkeley, CA, USA

Aniruddha N
Google
Mountain View,

Joonho Whangbo
UC Berkeley
Berkeley, CA, USA

Jerry Zh
UC Berkel
Berkeley, CA

F4T: A Fast and
TCP Ac

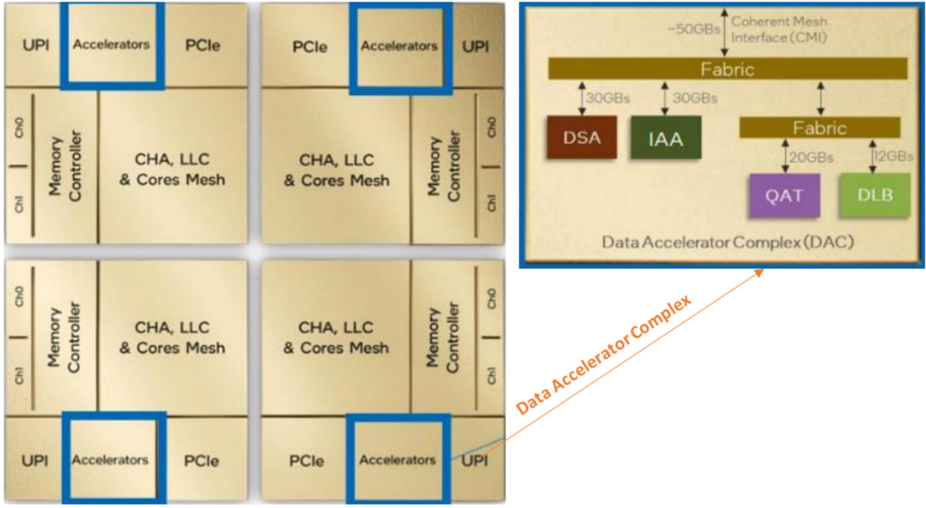
Be

Yak

Be

Junehyuk Boo
junehyuk@snu.ac.kr
Seoul National University
MangoBoost Inc.
Seoul, Republic of Korea

Seongmin Na
seongmin.na@snu.ac.kr
Seoul National University
Seoul, Republic of Korea



st RPCs in Cloud Microservices with
ory Reconfigurable NICs

Shaojie Xiang

Neil Adit
Cornell University
Ithaca, New York, USA
na469@cornell.edu

ervers

sein Kassir*
assir@zhinst.com
n Instruments

limitrou
versity
ork, USA
rnell.edu

aulo Drumond

for Protocol Buffers

ary

Chris Kennelly
Google
USA

rimi
ey

Borivoje Nikolić
UC Berkeley
USA

UC Berkeley
USA

Parthasarathy Ranganathan
Google
USA

Many Proposals for Individual Accelerators

CDPU: Co-designi
Processing

Sagar Karandikar
UC Berkeley, Google
Berkeley, CA, USA

Joonho Whangbo
UC Berkeley
Berkeley, CA, USA

F4T: A

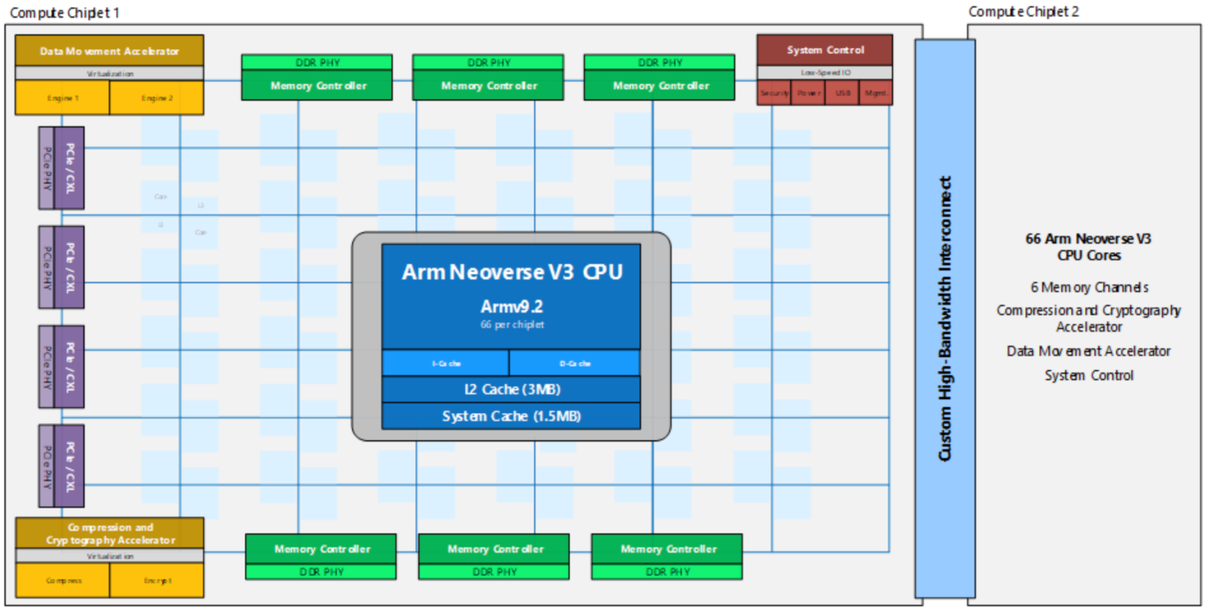
Be

Yak

Be

Junehyuk Boo
junehyuk@snu.ac.l
Seoul National Univer
MangoBoost Inc.
Seoul, Republic of Kc

Seongmin Na
seongmin.na@snu.ac
Seoul National Univer
Seoul, Republic of Kc



Microservices with
le NICs

ocol Buffers

Neil Adit
Cornell University
Ithaca, New York, USA
na469@cornell.edu

limitrou
versity
ork, USA
rnell.edu

Chris Kennelly
Google
USA

Borivoje Nikolić
UC Berkeley
USA

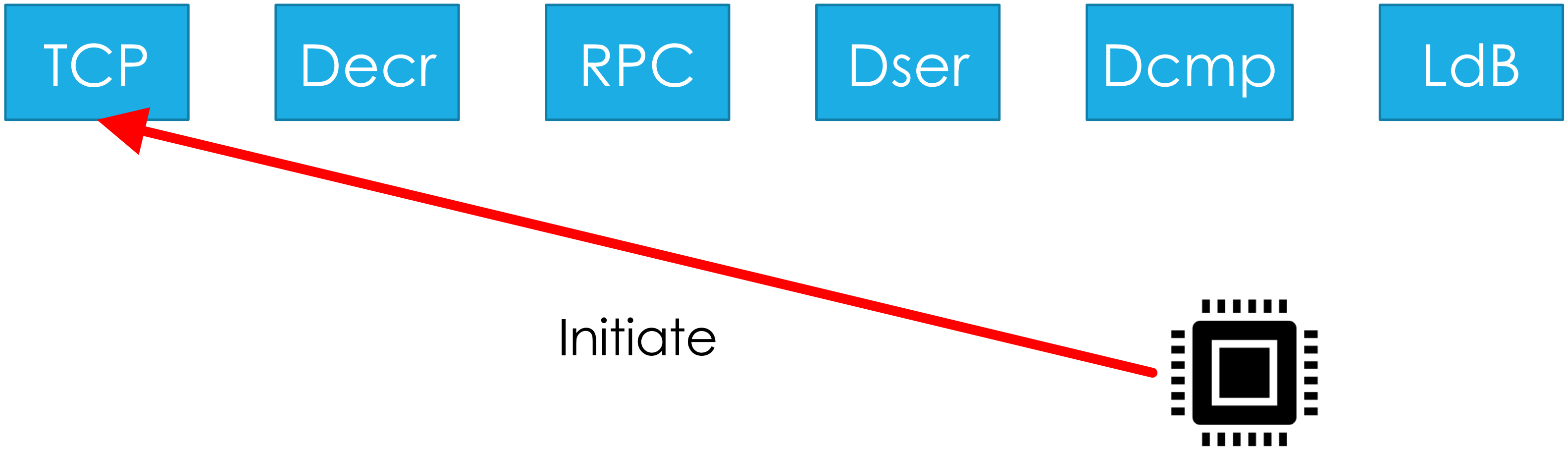
UC Berkeley
USA

Parthasarathy Ranganathan
Google
USA

How to Orchestrate Many Accelerators?

- Many individual accelerators proposed – how to manage them?

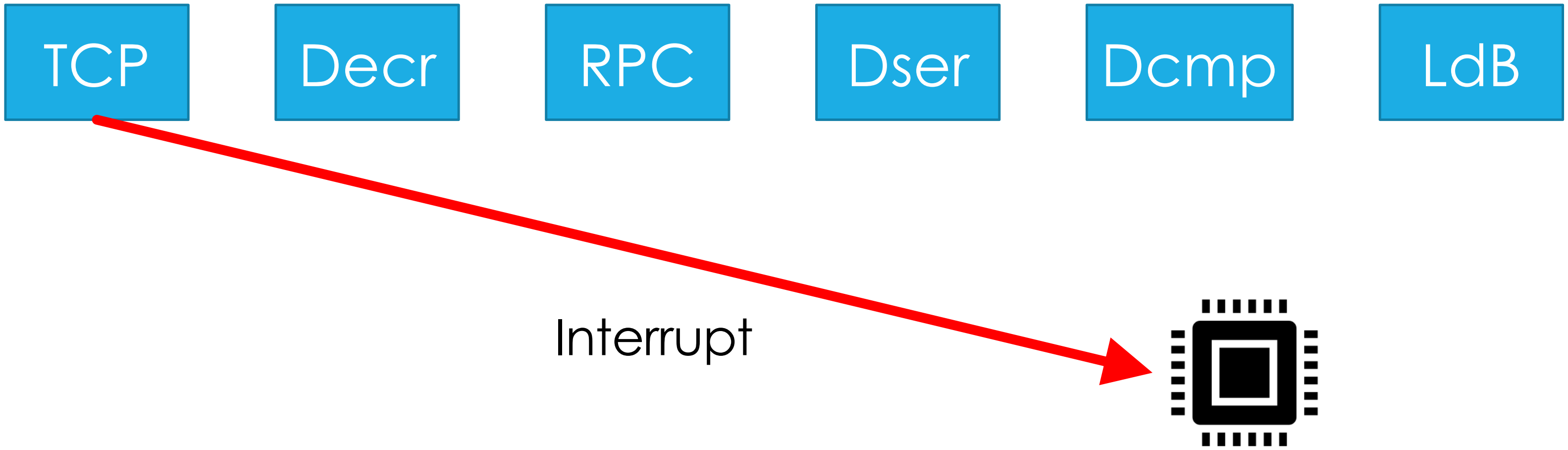
Orchestrate Many Accelerators: CPU-Centric



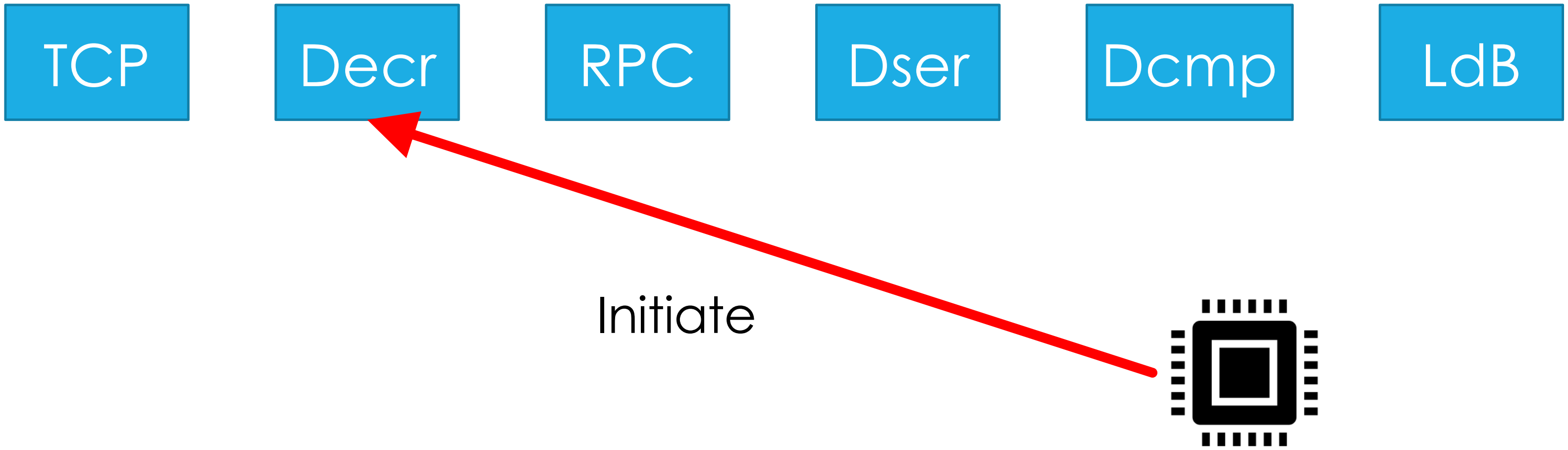
TCP Decr RPC Dser Dcmp LdB

Initiate

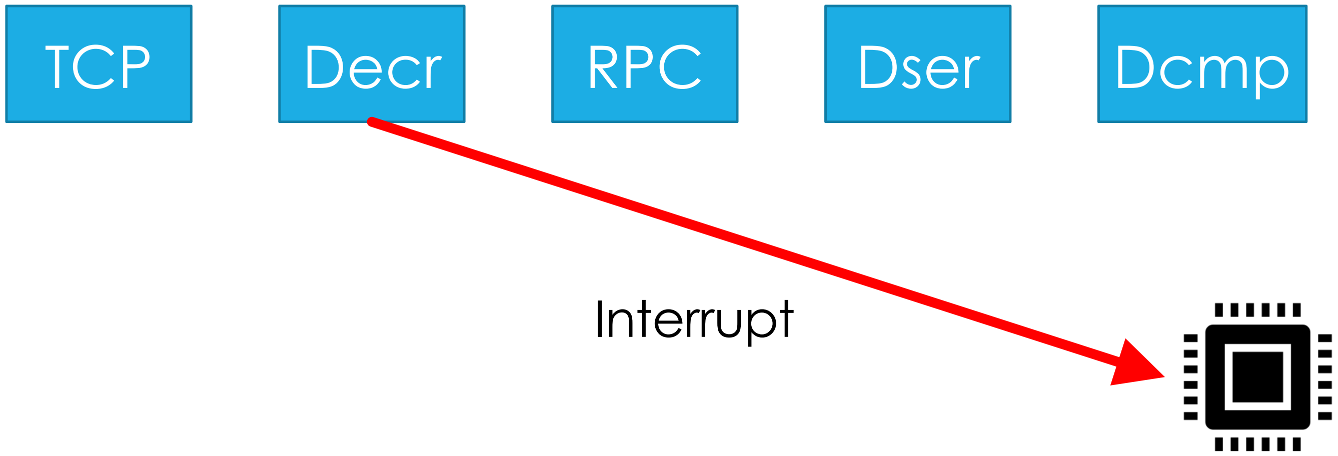
Orchestrate Many Accelerators: CPU-Centric



Orchestrate Many Accelerators: CPU-Centric



Orchestrate Many Accelerators: CPU-Centric



TCP

Decr

RPC

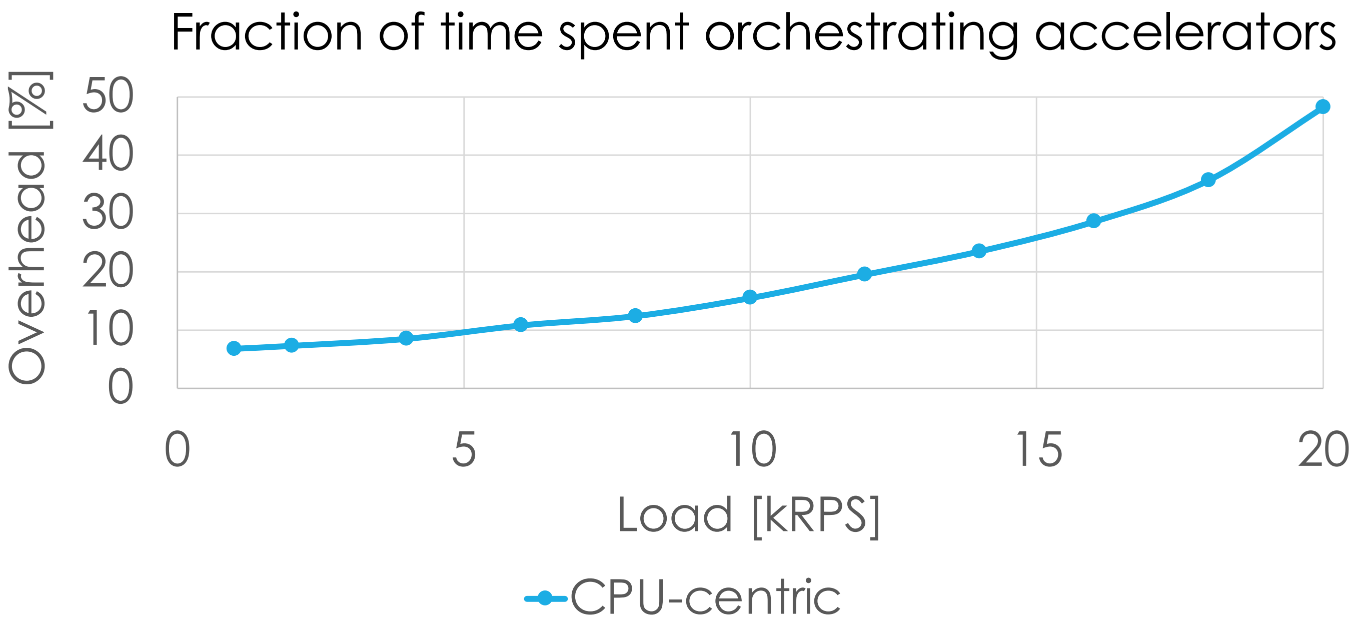
Dser

Dcmp

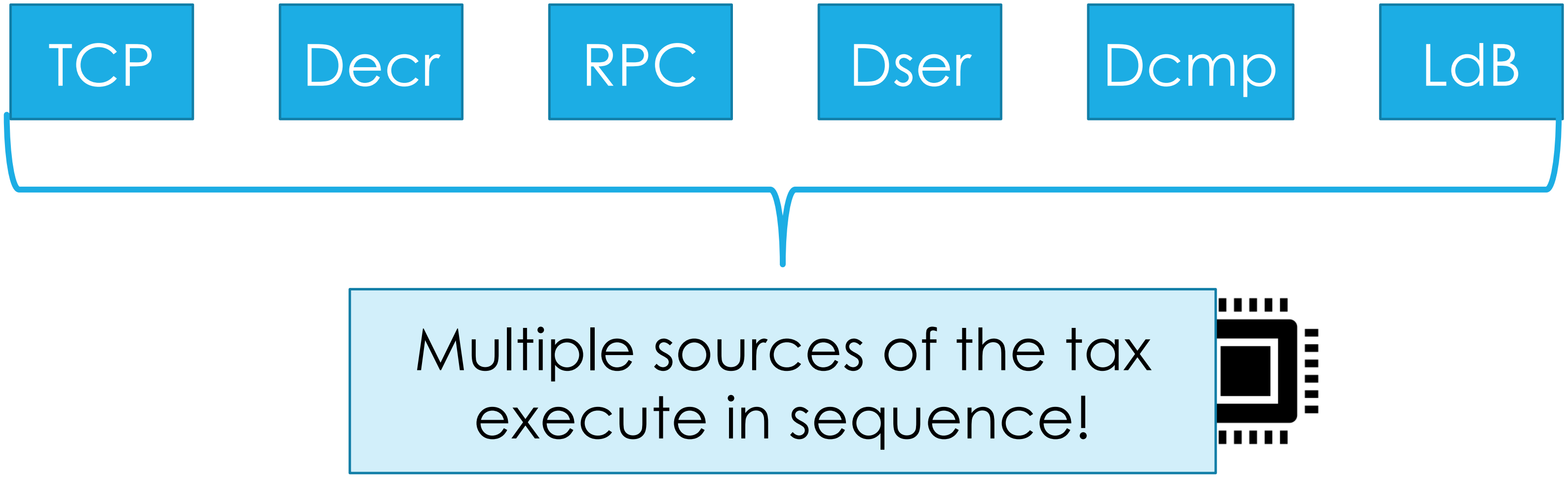
LdB

Interrupt

Repeated Interrupts → High Overhead

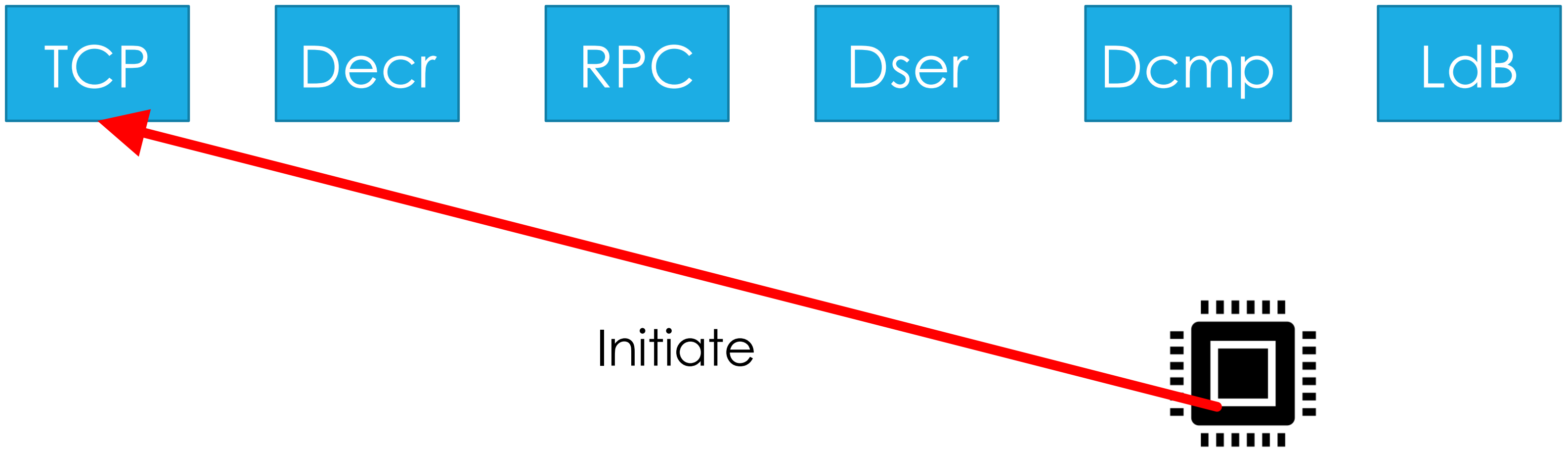


Orchestrate Many Accelerators: Direct Chain



Multiple sources of the tax execute in sequence!

Orchestrate Many Accelerators: Direct Chain



TCP

Decr

RPC

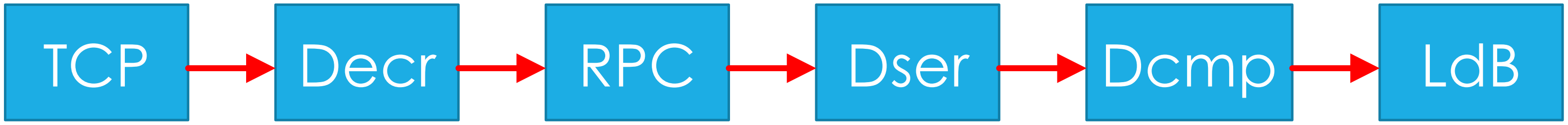
Dser

Dcmp

LdB

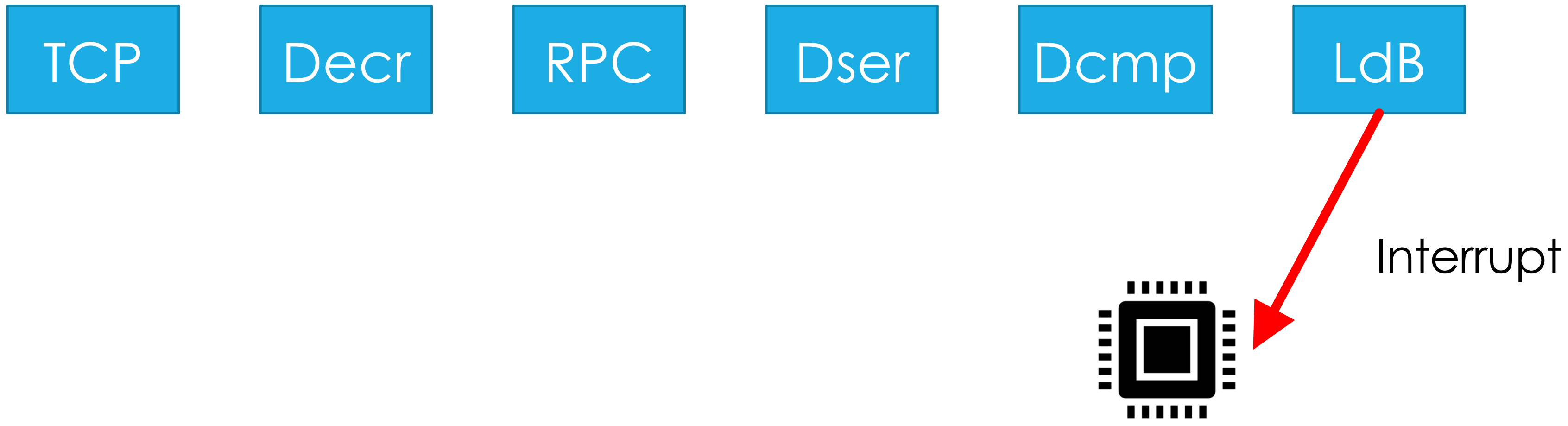
Initiate

Orchestrate Many Accelerators: Direct Chain



TCP → Decr → RPC → Dser → Dcmp → LdB

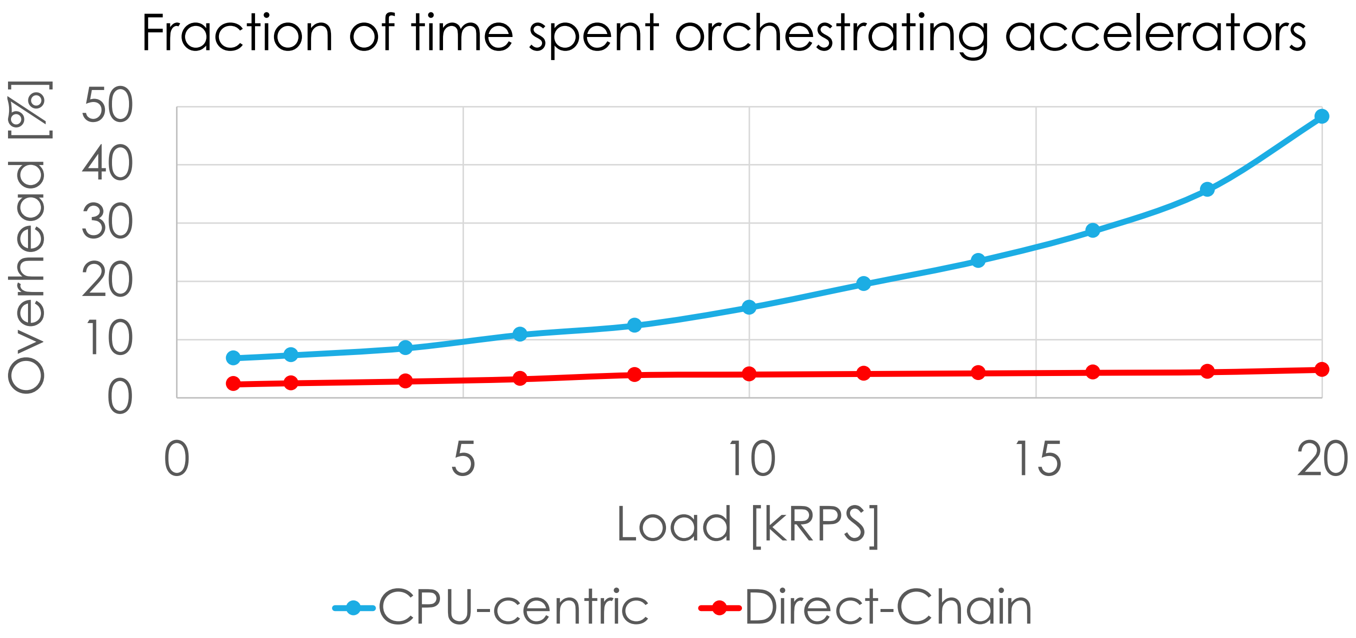
Orchestrate Many Accelerators: Direct Chain



TCP Decr RPC Dser Dcmp LdB

Interrupt

Direct Chaining Significantly Reduces Overheads

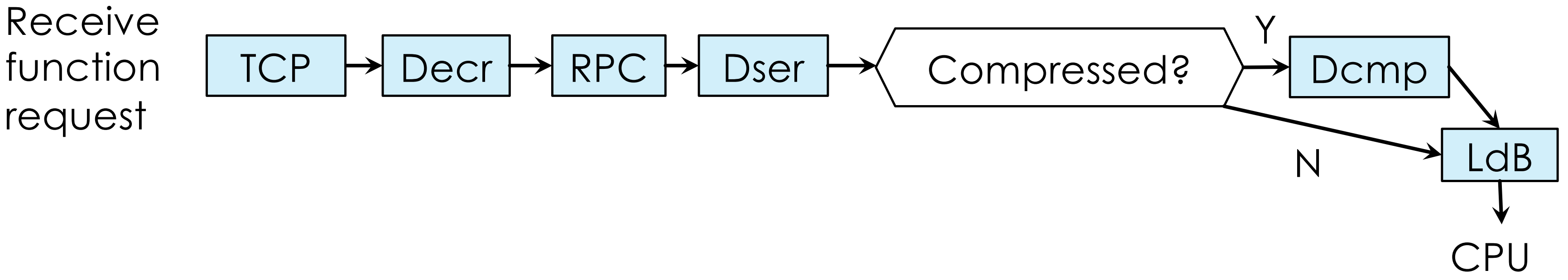


Challenges of Direct Chaining

- Control-flow divergences

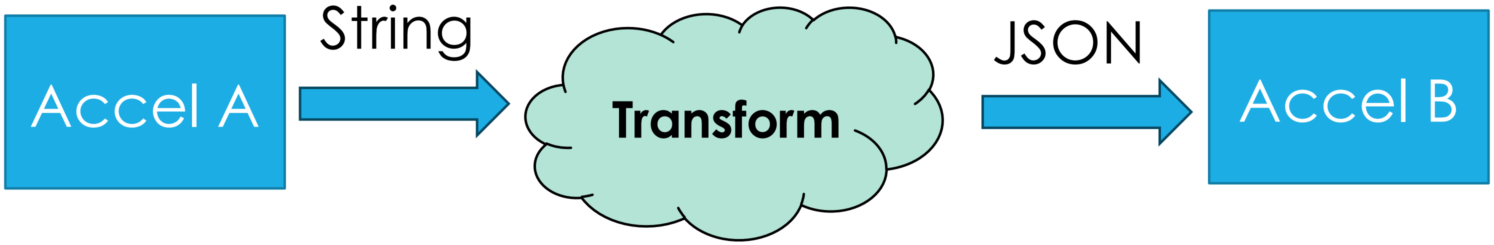
Challenges of Direct Chaining

- Control-flow divergences

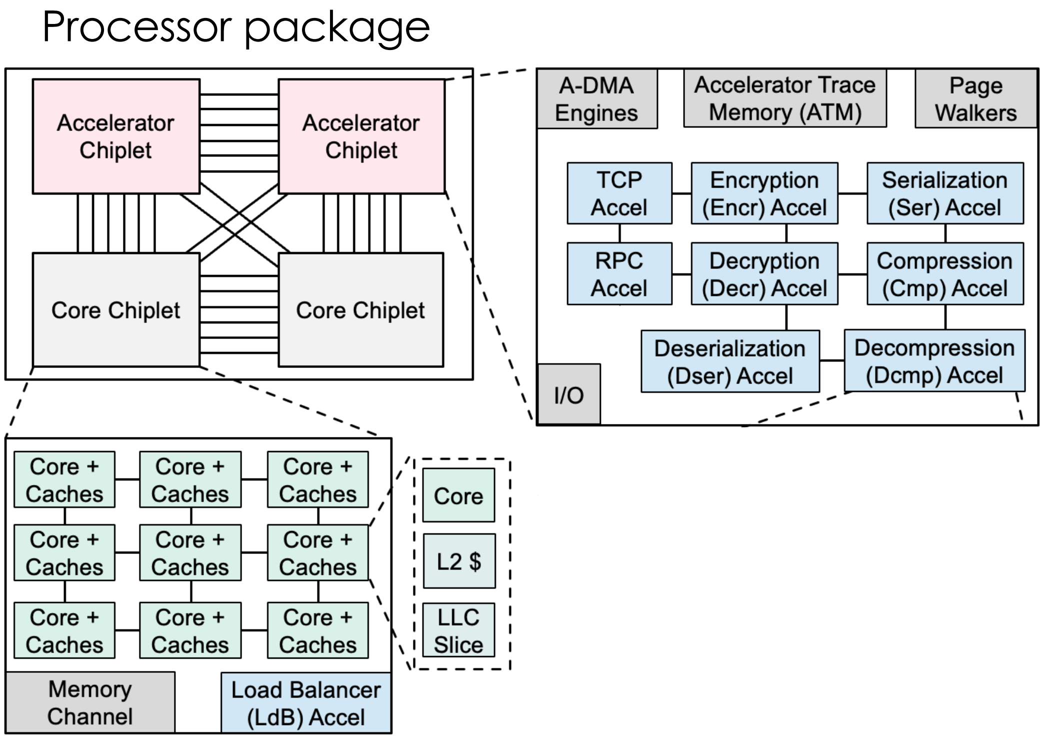


Challenges of Direct Chaining

- Control-flow divergences
- Data format transformations



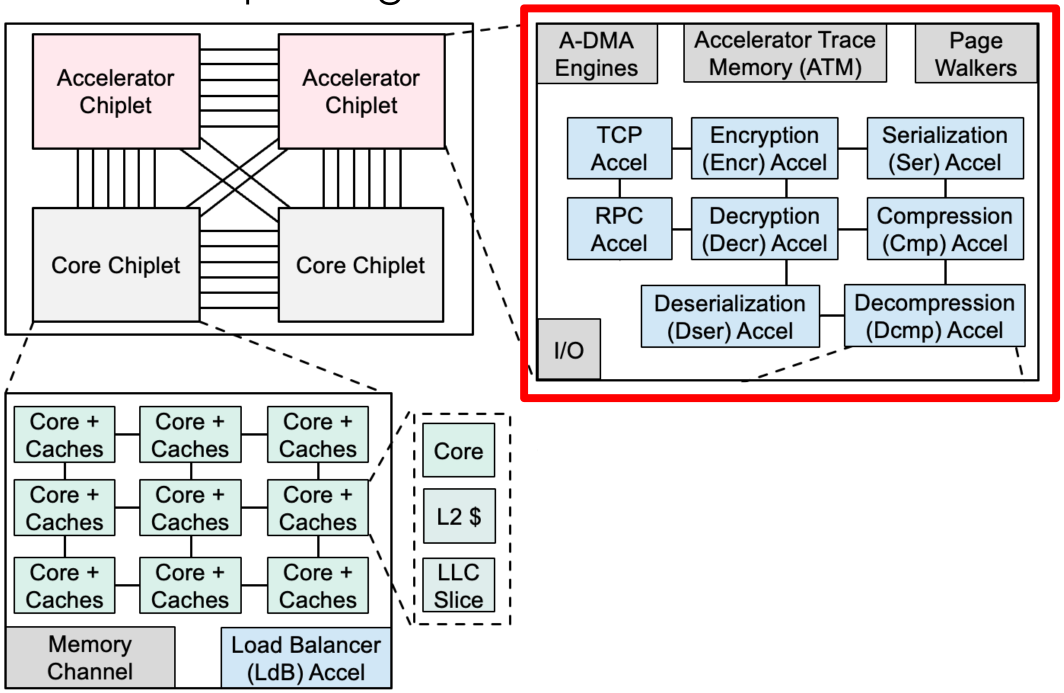
AccelFlow: Accelerator Orchestration Framework



AccelFlow: Accelerator Orchestration Framework

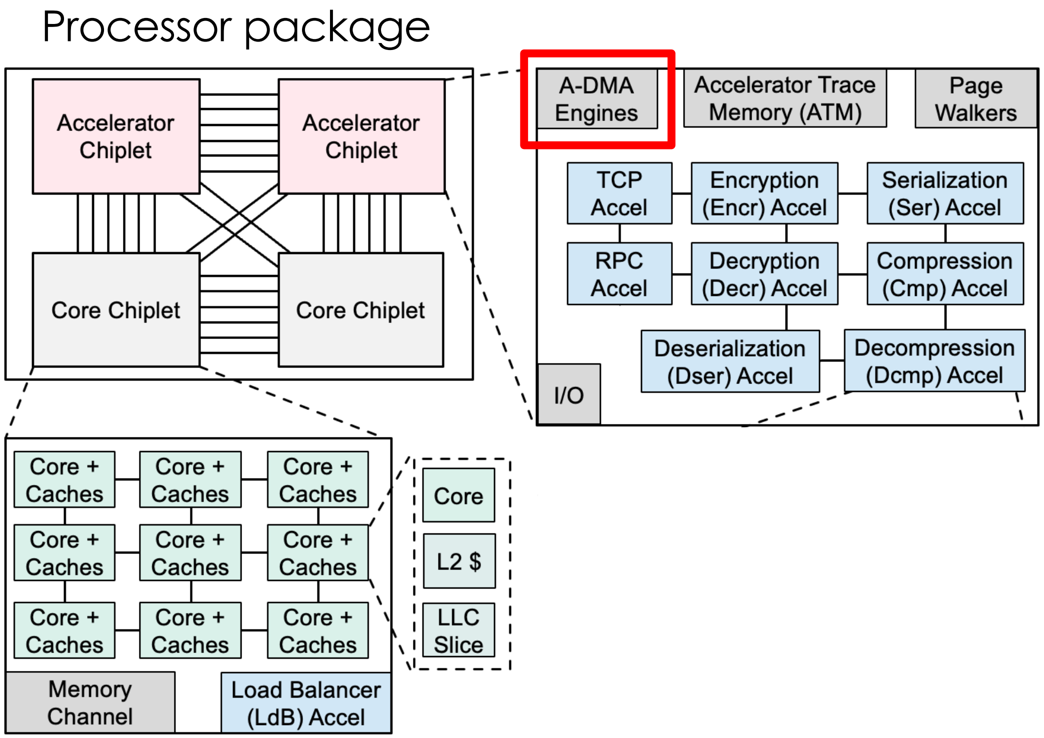
- Ensemble of accelerators

Processor package



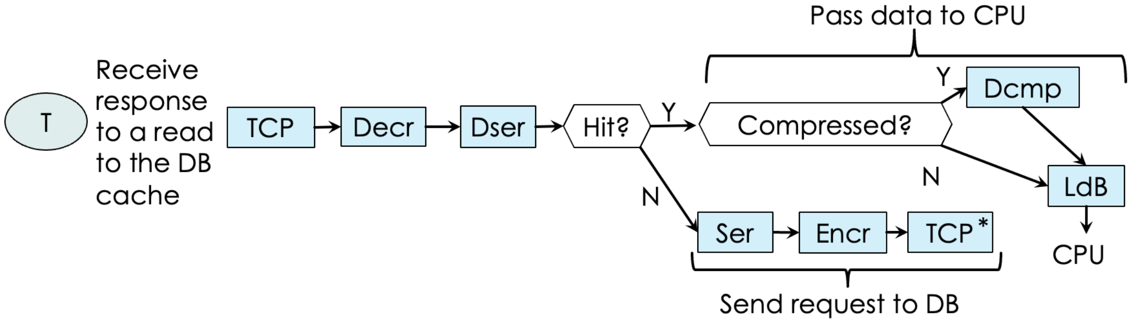
AccelFlow: Accelerator Orchestration Framework

- Ensemble of accelerators
- Direct inter-accelerator chaining



AccelFlow: Accelerator Orchestration Framework

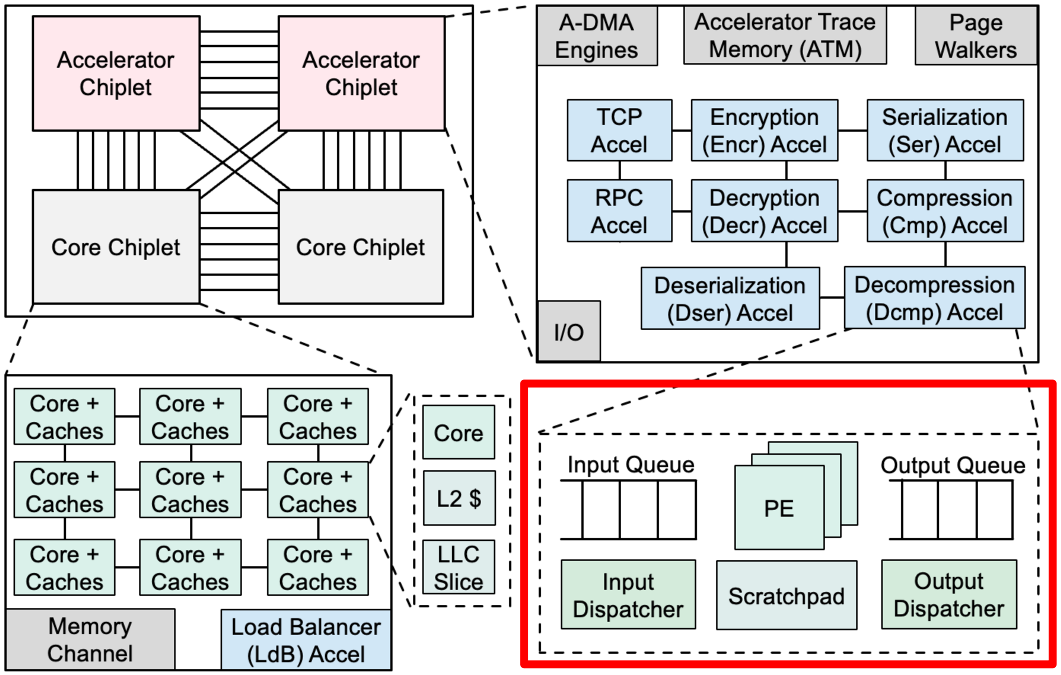
- Ensemble of accelerators
- Direct inter-accelerator chaining
- Sequence of accelerators stored in **Software "Traces"**



AccelFlow: Accelerator Orchestration Framework

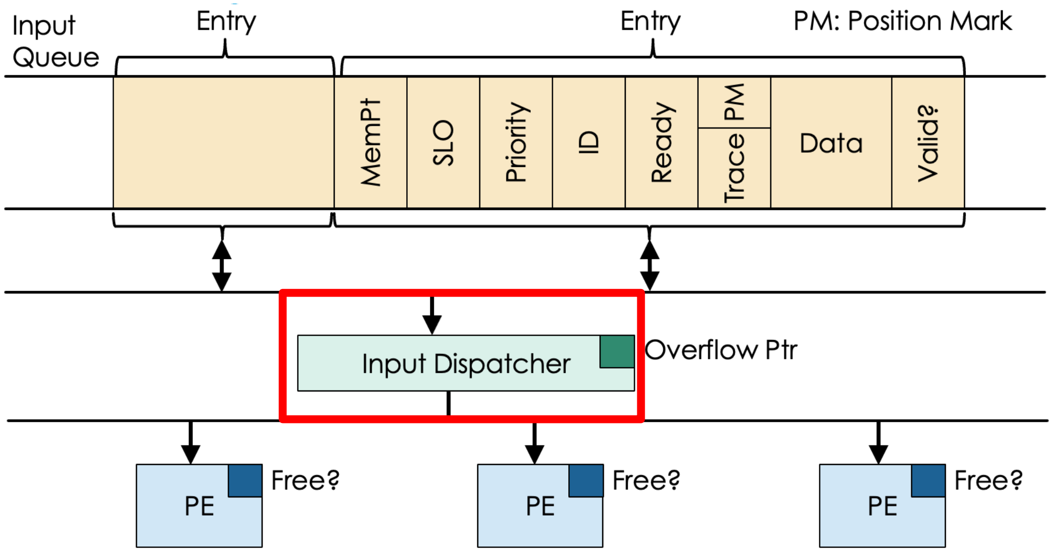
- Ensemble of accelerators
- Direct inter-accelerator chaining
- Sequence of accelerators stored in software “traces”
- Standard interface
  - Input and output queues and dispatchers

Processor package



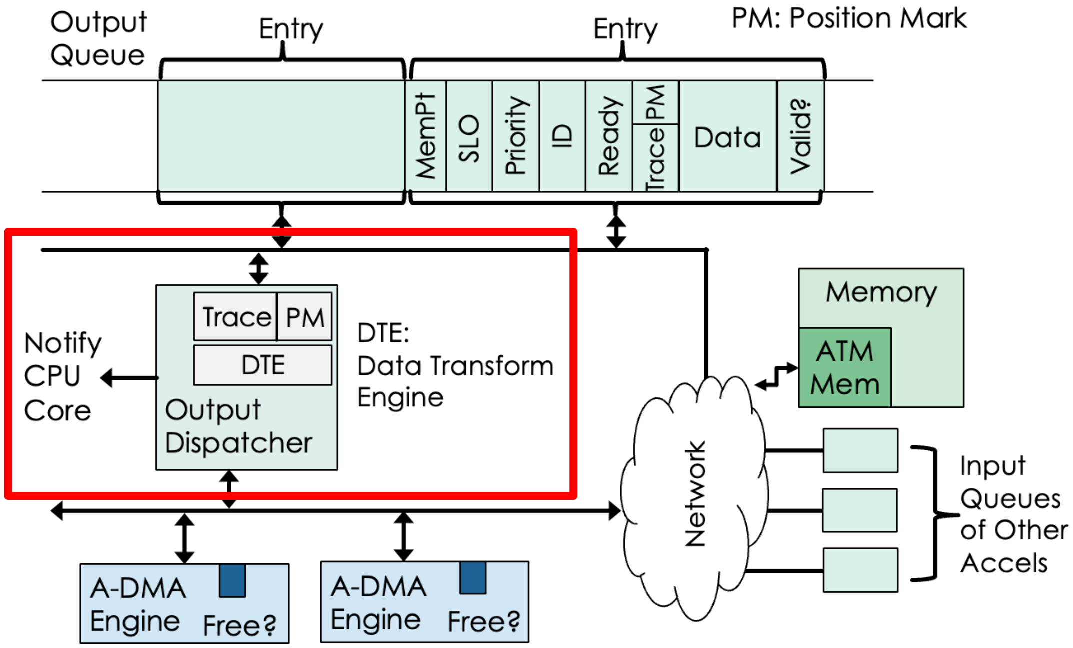
Input Dispatcher

- Schedules the requests from Input Queue to PEs
- Fetches large input payloads from memory
- Simple Finite State Machine



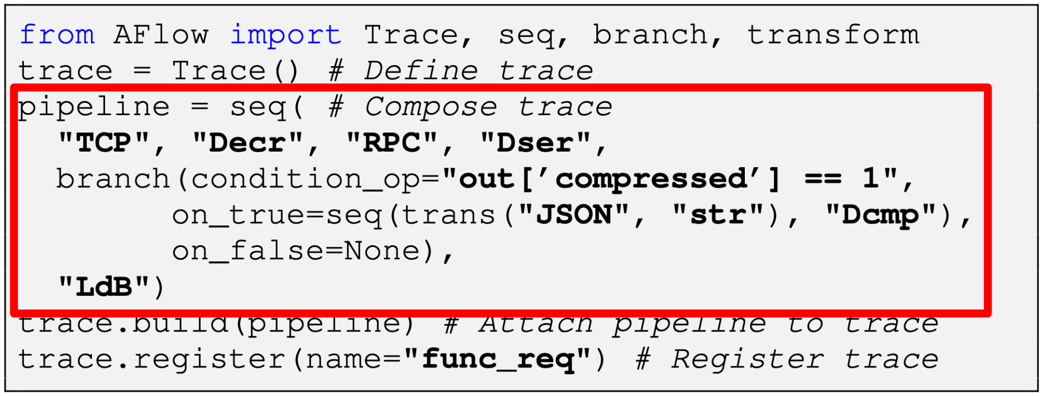
Output Dispatcher

- Forward the request + data to next accelerator or to the CPU
- Compute branch conditions
- Perform data transformations



Programming AccelFlow

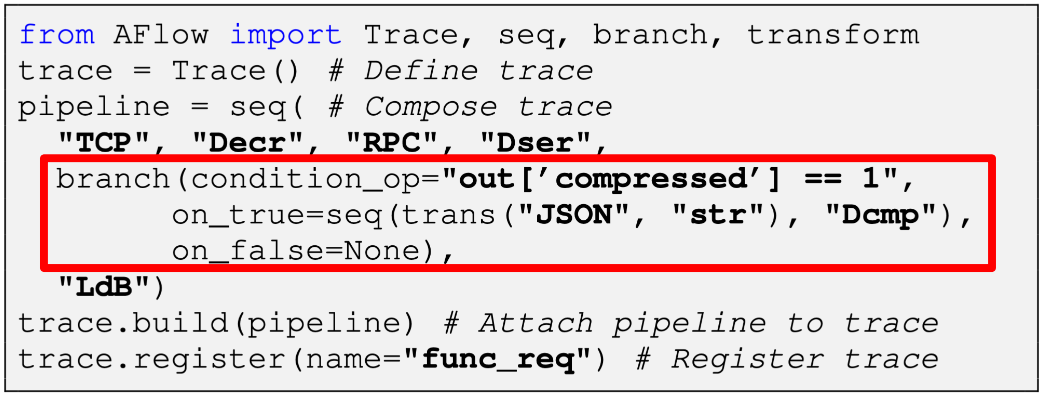
- AccelFlow API allows programmers to construct new traces:
  - Define a linear chain of accelerators



```
from AFlow import Trace, seq, branch, transform
trace = Trace() # Define trace
pipeline = seq( # Compose trace
  "TCP", "Decr", "RPC", "Dser",
  branch(condition_op="out['compressed'] == 1",
        on_true=seq(trans("JSON", "str"), "Dcmp"),
        on_false=None),
  "LdB")
trace.build(pipeline) # Attach pipeline to trace
trace.register(name="func_req") # Register trace
```

Programming AccelFlow

- AccelFlow API allows programmers to construct new traces:
    - Define a linear chain of accelerators
    - Add a conditional control flow



```
from AFlow import Trace, seq, branch, transform
trace = Trace() # Define trace
pipeline = seq( # Compose trace
  "TCP", "Decr", "RPC", "Dser",
  branch(condition_op="out[’compressed’] == 1",
        on_true=seq(trans("JSON", "str"), "Dcmp"),
        on_false=None),
  "LdB")
trace.build(pipeline) # Attach pipeline to trace
trace.register(name="func_req") # Register trace
```

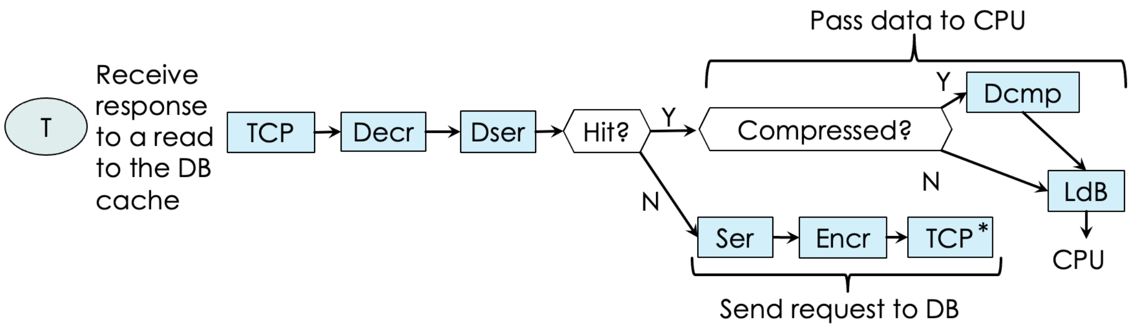
Programming AccelFlow

- AccelFlow API allows programmers to construct new traces:
  - Define a linear chain of accelerators
  - Add a conditional control flow
  - Transform the format of the data from one representation to another

```
from AFlow import Trace, seq, branch, transform
trace = Trace() # Define trace
pipeline = seq( # Compose trace
  "TCP", "Decr", "RPC", "Dser",
  branch(condition_op="out['compressed'] == 1",
        on_true=seq(trans("JSON", "str"), "Dcmp"),
        on_false=None),
  "LdB")
trace.build(pipeline) # Attach pipeline to trace
trace.register(name="func_req") # Register trace
```

AccelFlow Summary

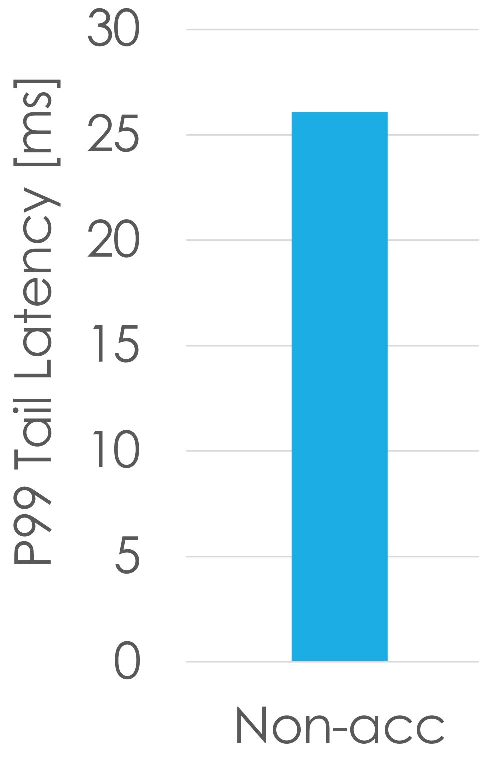
- Many on-chip accelerators to reduce datacenter tax
- Accelerators communicate directly with each other
- Small hardware engines
  - Schedule requests onto accelerator PEs
  - Compute branch conditions
  - Perform simple data transformations



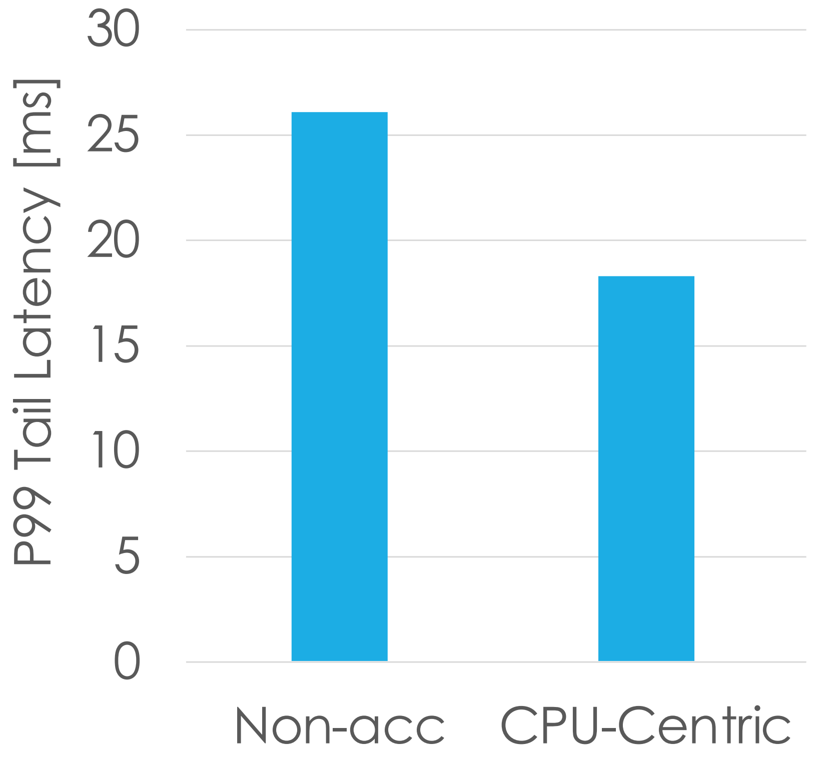
Evaluation Methodology

- Cycle-accurate full-system simulations: SST + QEMU
- DeathStarBench services with Alibaba's production invocation traces
- Systems evaluated
  - **CPU-centric:** accelerators orchestrated by CPU cores
  - **RELIEF (HPCA'24):** accelerators orchestrated by a dedicated and centralized hardware manager
  - **Cohort (ASPLOS'23):** links pairs of accelerators that frequently go together, but otherwise relies on the cores to orchestrate the accelerators
  - **AccelFlow:** our proposal

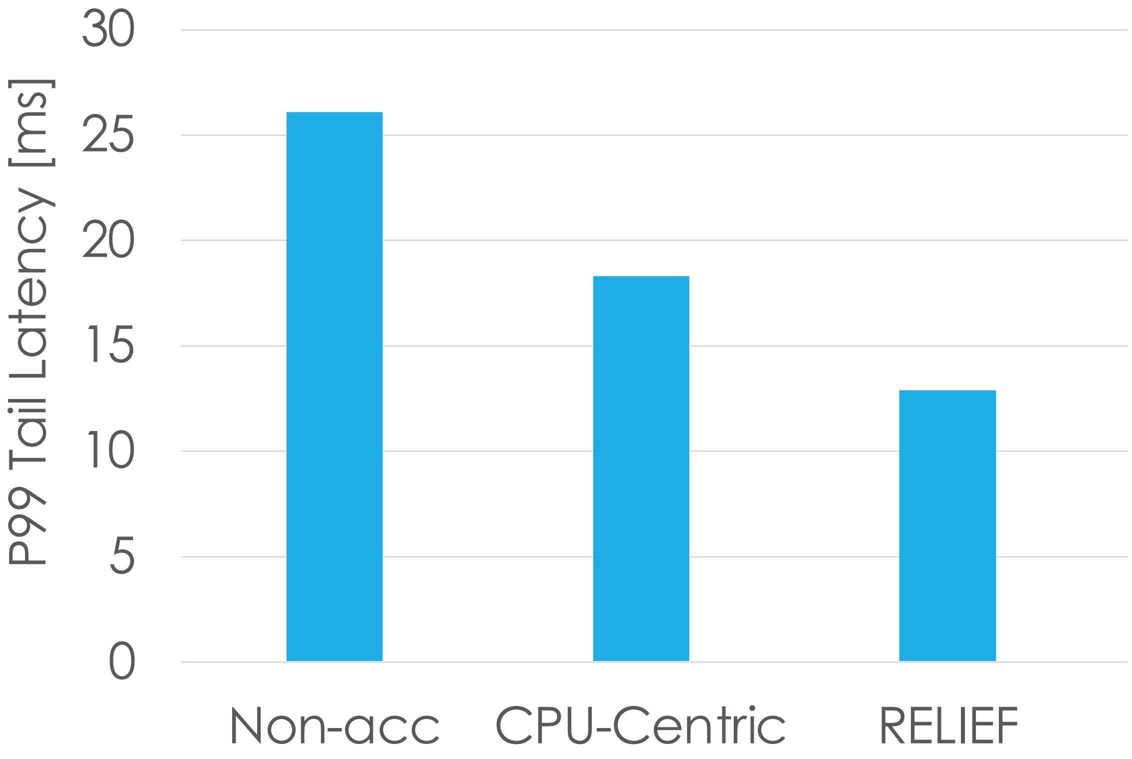
AccelFlow Significantly Reduces Tail Latency



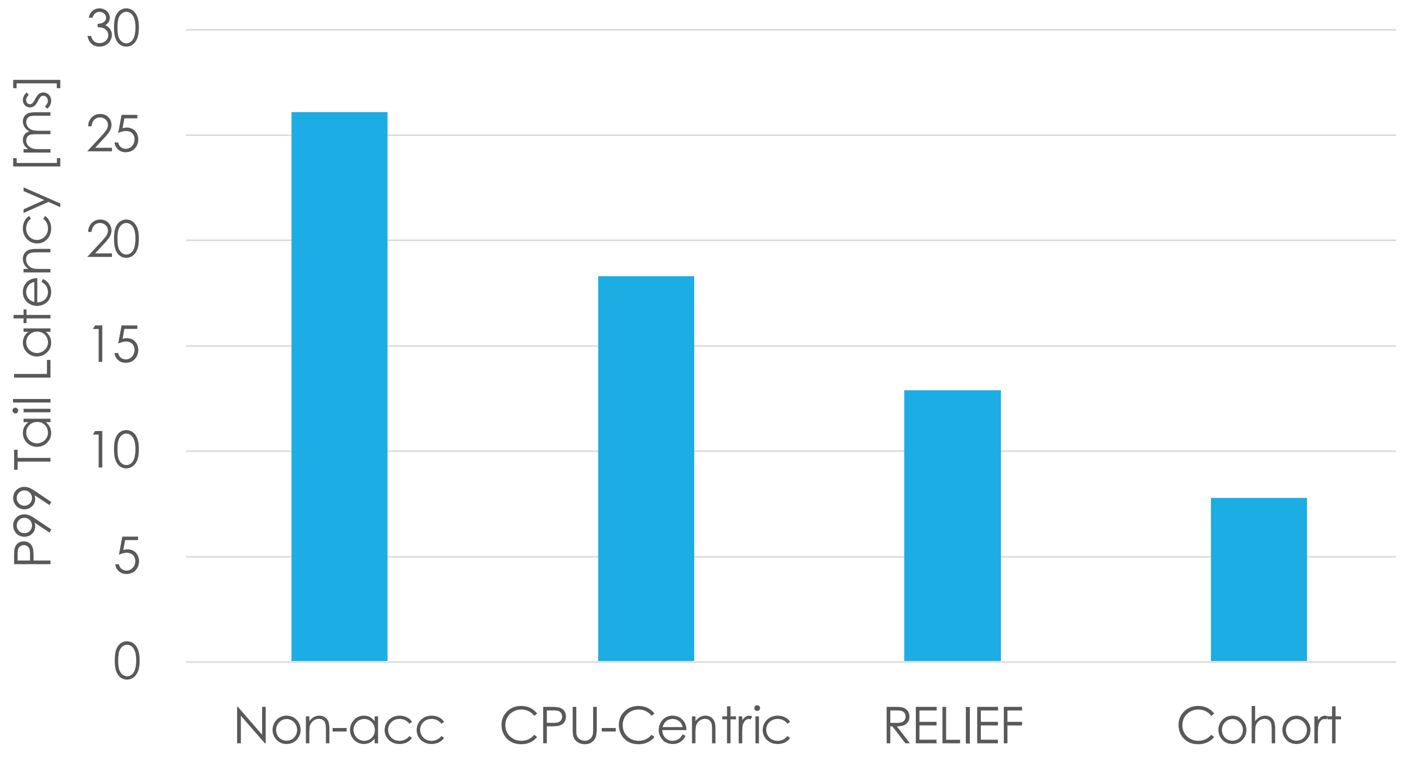
AccelFlow Significantly Reduces Tail Latency



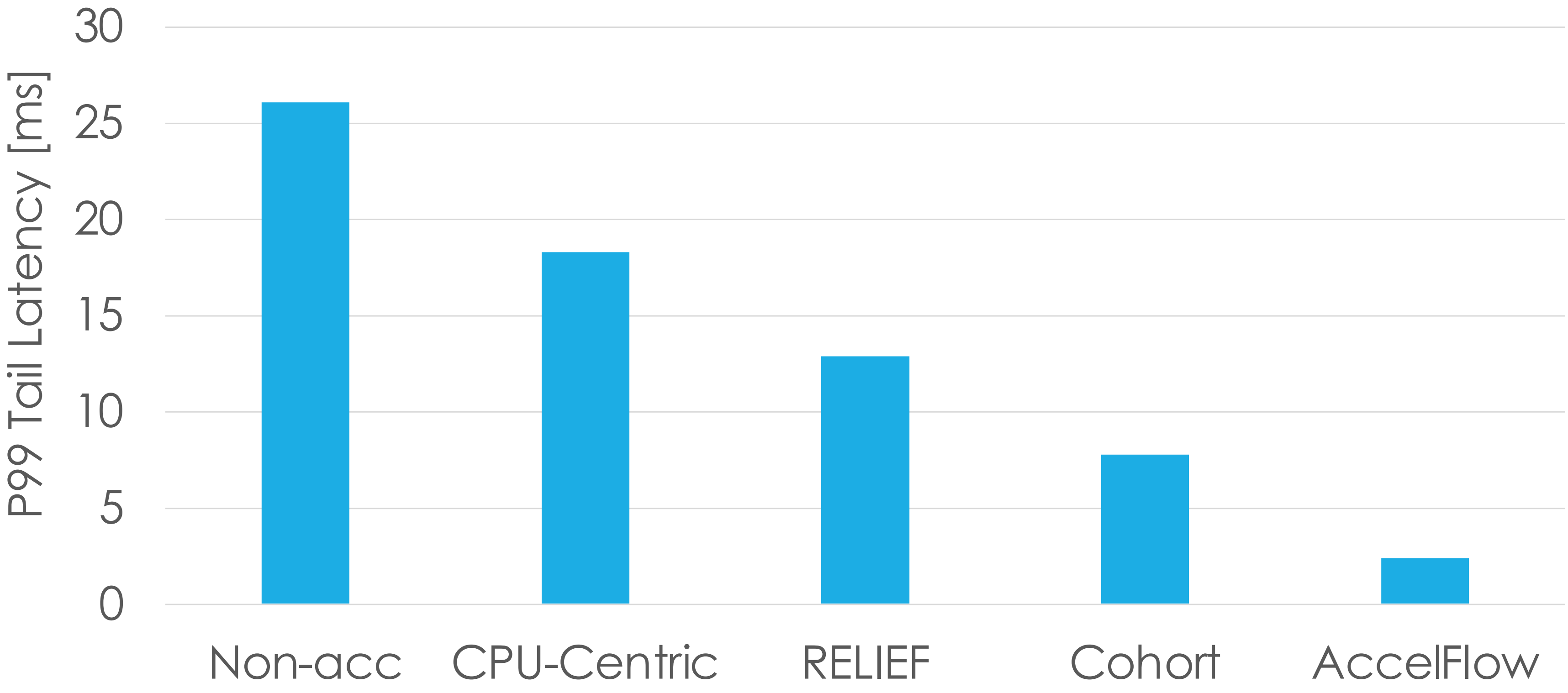
AccelFlow Significantly Reduces Tail Latency



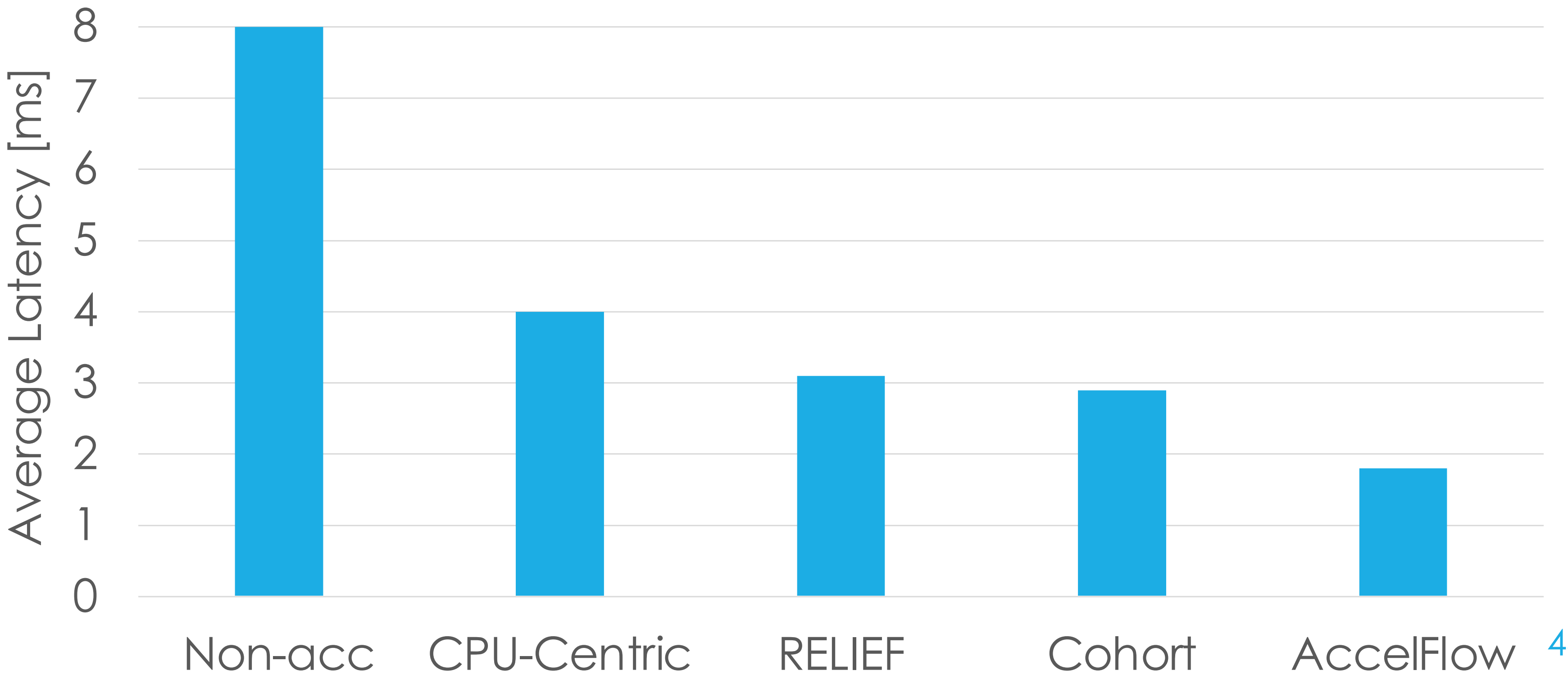
AccelFlow Significantly Reduces Tail Latency



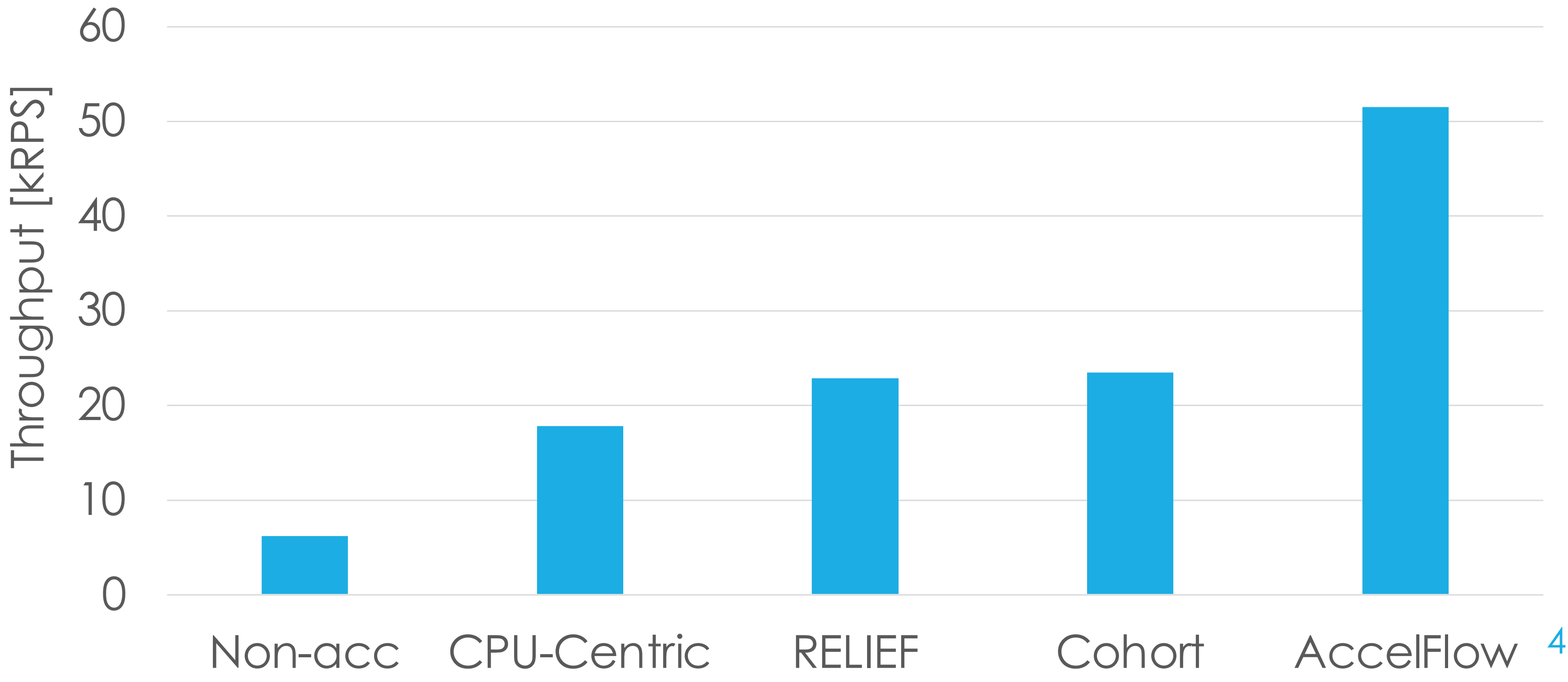
AccelFlow Significantly Reduces Tail Latency



AccelFlow Reduces Average Latency



AccelFlow Improves Throughput



Conclusion

- An ensemble of domain-specific accelerators for ”datacenter tax” has the potential to improve the efficiency of microservices
- Realizing these benefits requires an orchestration framework that can keep up with the fine-grained and dynamic microservices
- **AccelFlow:** the first accelerator-orchestration framework for on-chip accelerators targeting microservices
  - 70% lower tail latency
  - 38% lower average latency
  - 2.2x higher throughput

AccelFlow: Orchestrating an On-Package Ensemble of Fine-Grained Accelerators for Microservices

HPCA 2026, Sydney, Australia

Jovan Stojkovic[*], Abraham Farrell, Zhangxiaowen Gong[†], Christopher J. Hughes[†], Josep Torrellas

University of Illinois at Urbana-Champaign, [†]Intel, [*]Joining UT Austin in Fall 2026