UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN

Cloud-Native Workloads in Datacenters

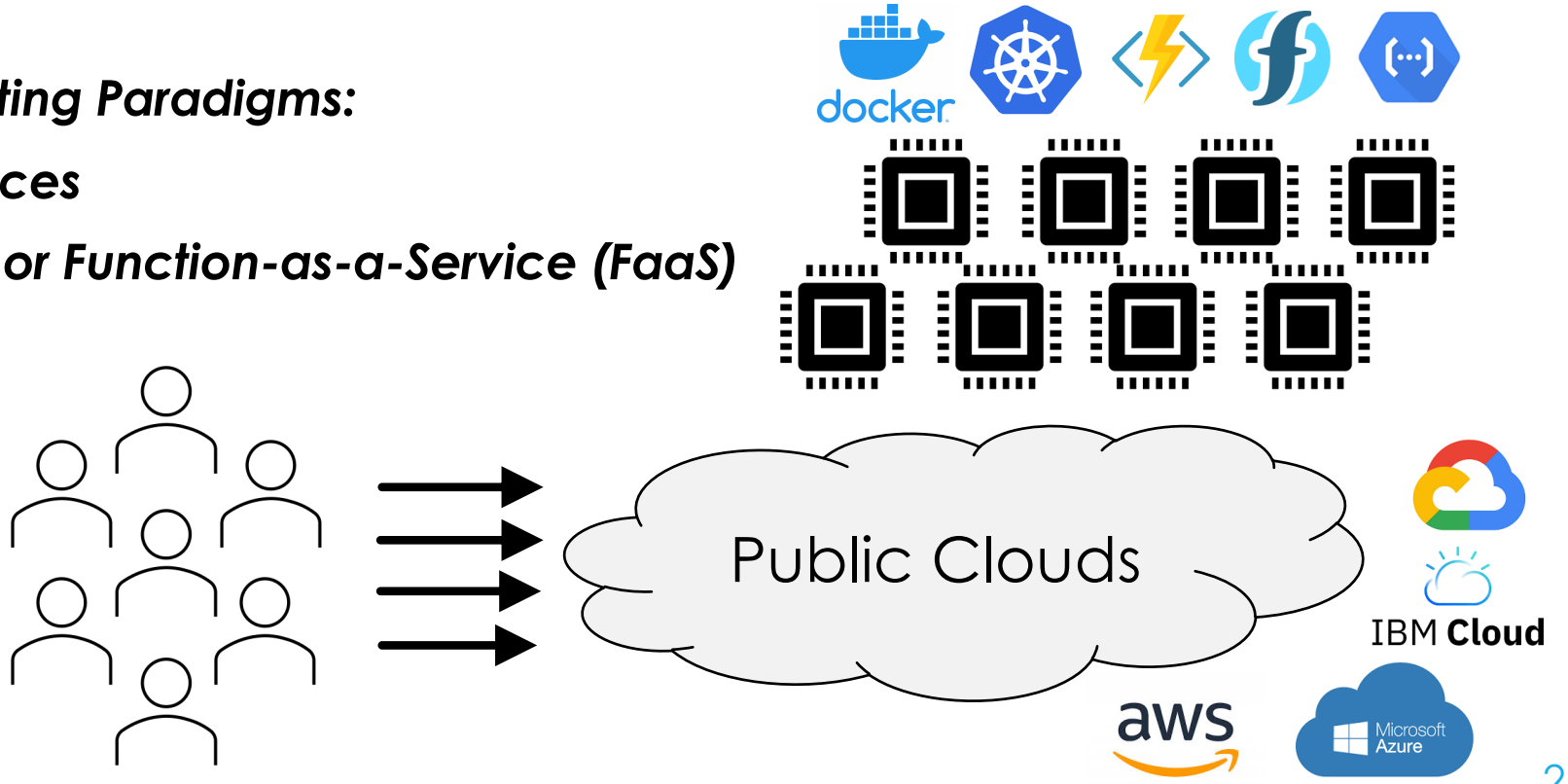
Jovan Stojkovic

CS 533

The Growth of Cloud Computing

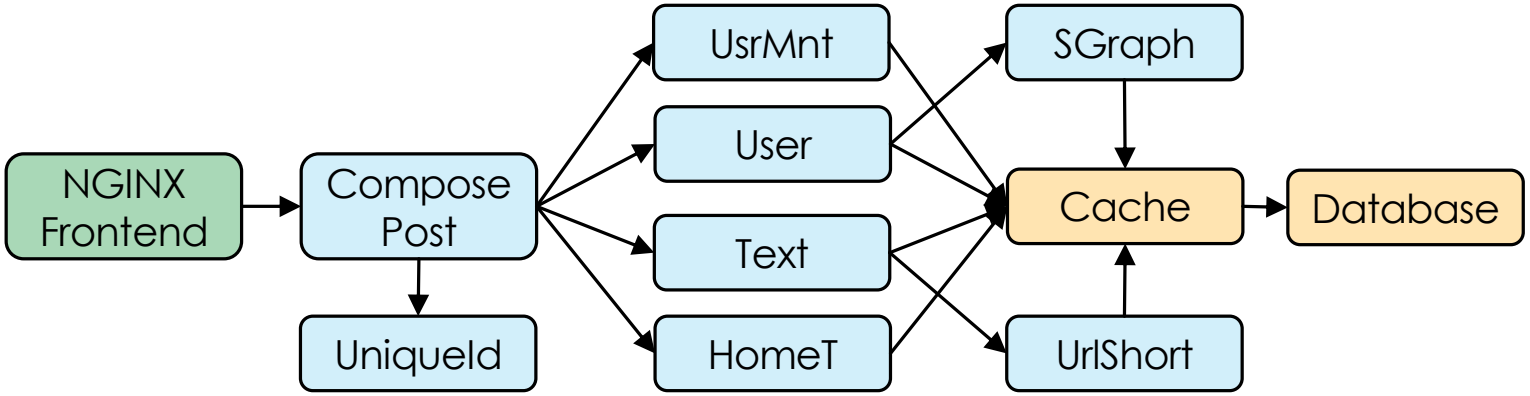
New Computing Paradigms:

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



Microservices

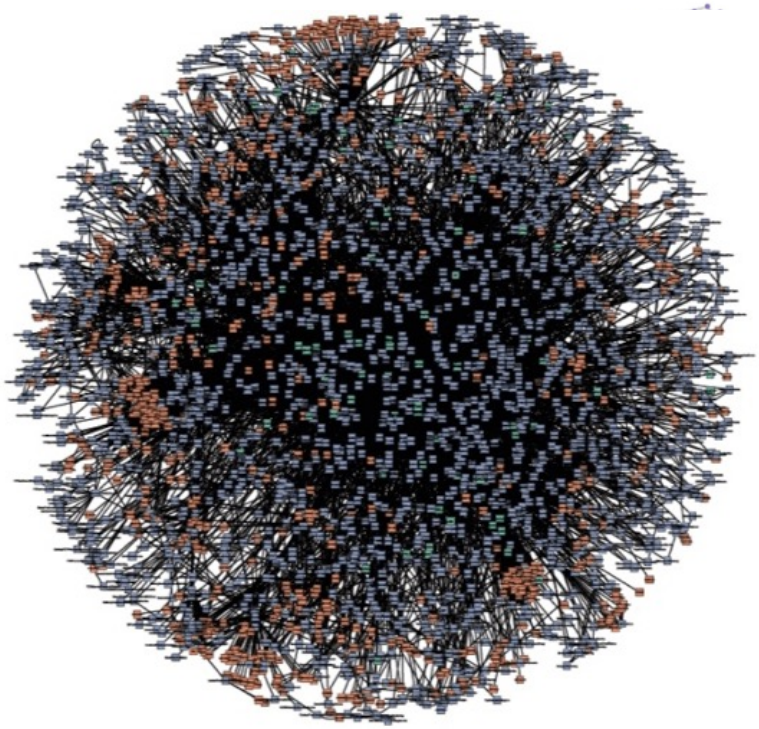
- Large monolithic applications decomposed into many small interdependent services
  - Each service implements separate functionality

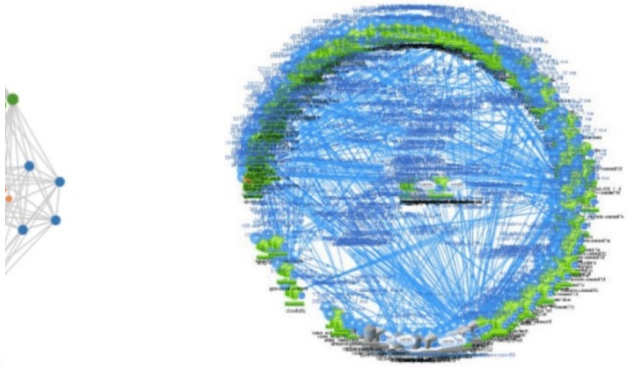


Benefits of Microservices

- Scalability 
- Design simplicity 
- HW management 

Microservices are Widely Used





ıre

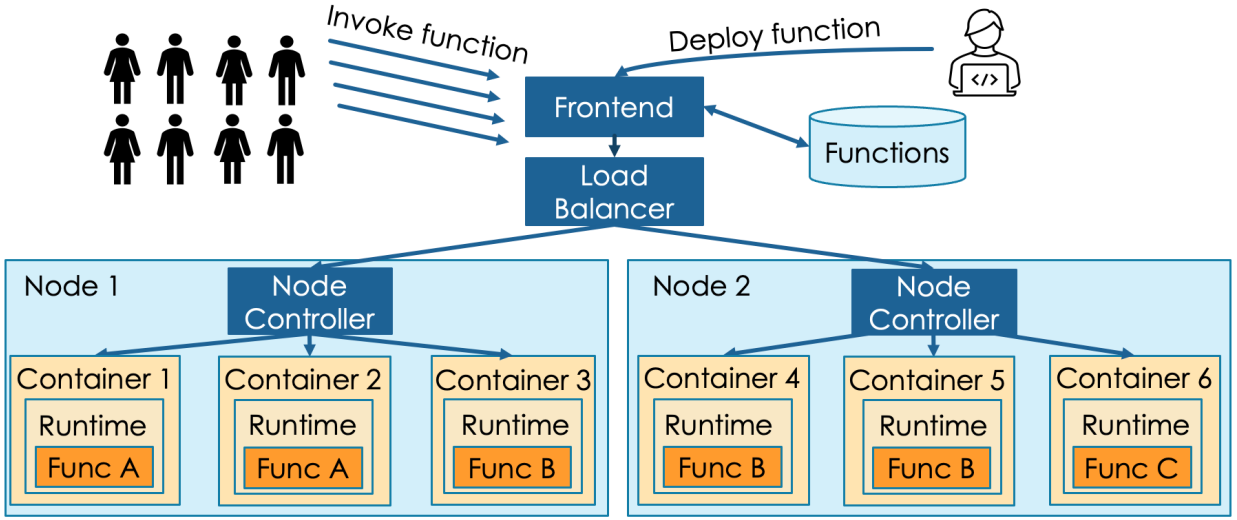
Actual Architecture

: simplified and actual scheme (source)

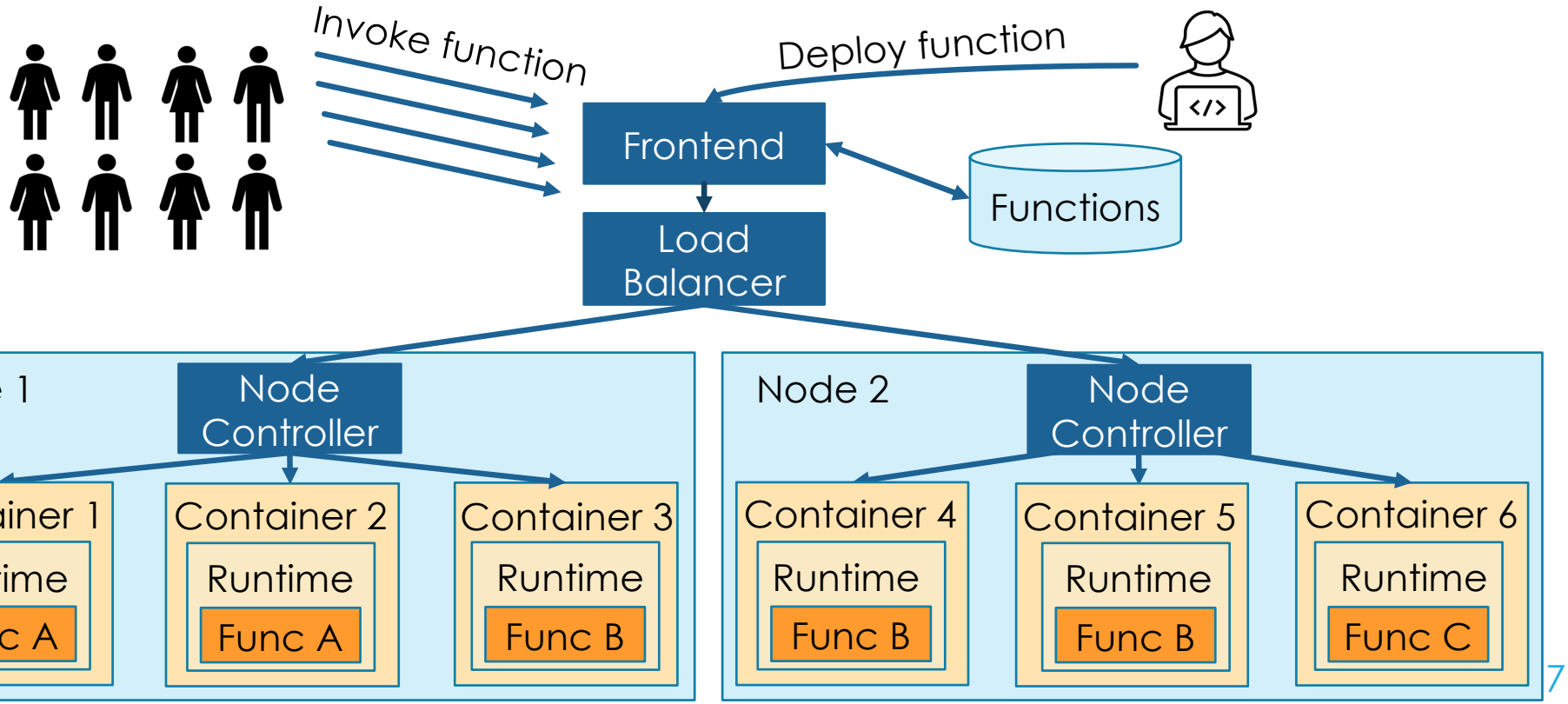
Structure of microservices at Amazon. Looks almost like a Death Star but is way more powerful.

Step Further: Serverless Computing

- Maintain microservice-based software architecture
- Remove the resource provisioning burden from the users



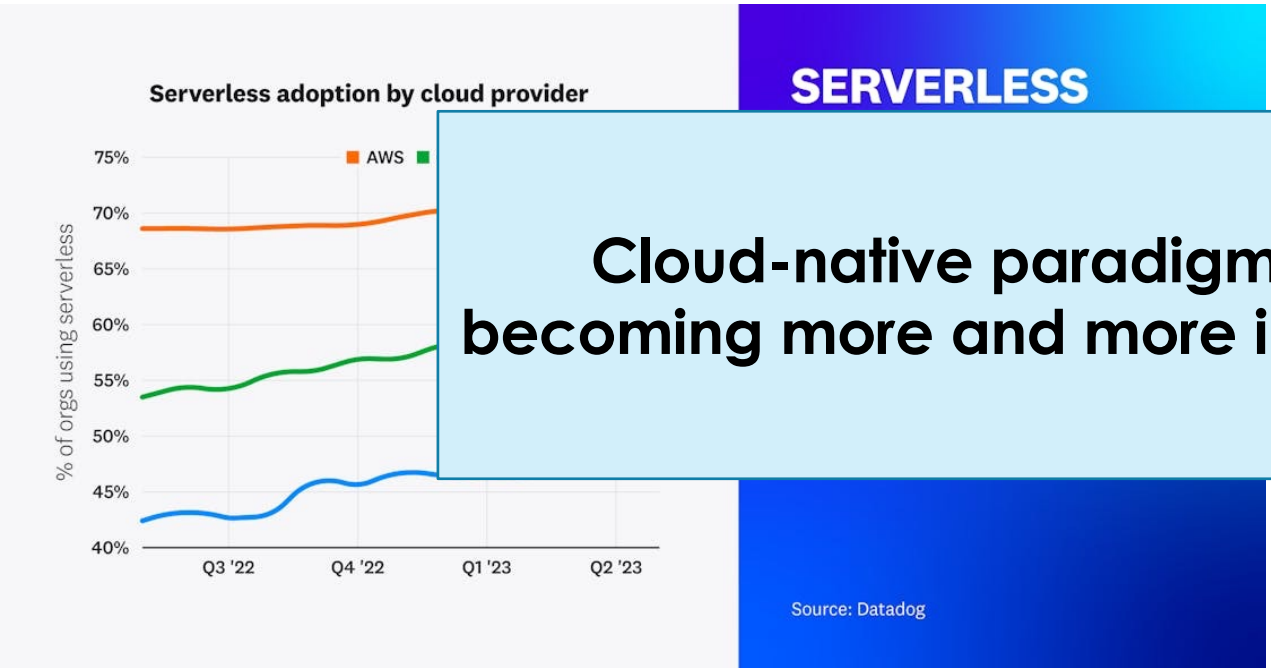
How Does Serverless Computing Work?



Benefits of Serverless Computing

- Simple and modular programming
- Automatic resource scaling
- Pay-as-you-go billing model

Serverless Computing is Also Widely Used



Cloud-native paradigms are becoming more and more important!

Mismatch: Current Processors vs. Cloud-Native Loads

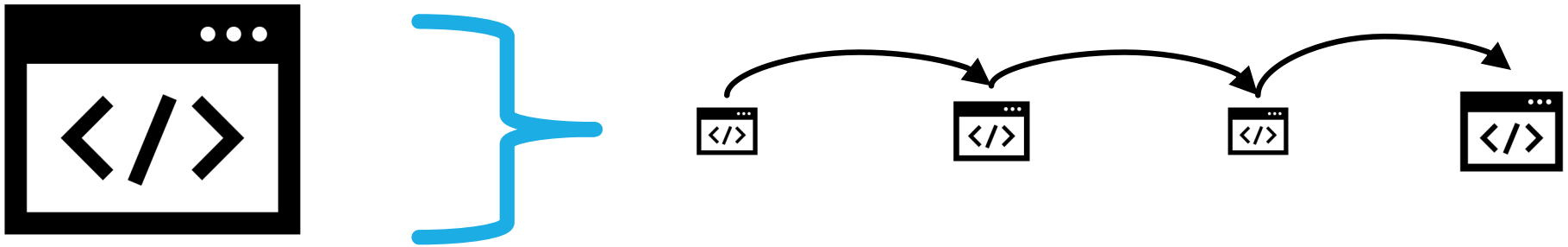
Current Processors	Cloud-Native Environments

Mismatch: Current Processors vs. Cloud-Native Loads

Current Processors	Cloud-Native Environments
Beefy processors	Small-sized services

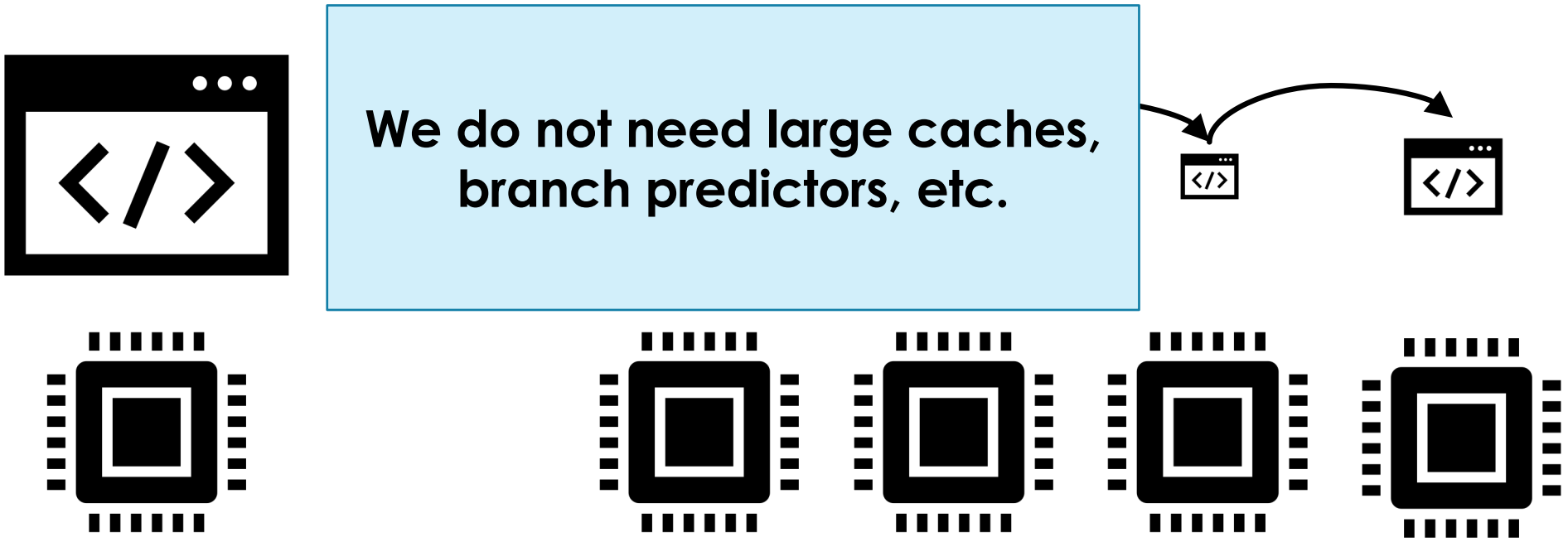
Cloud-Native Services Have Small Footprints

- Cloud-native services much smaller than monolithic counterparts



Cloud-Native Services Have Small Footprints

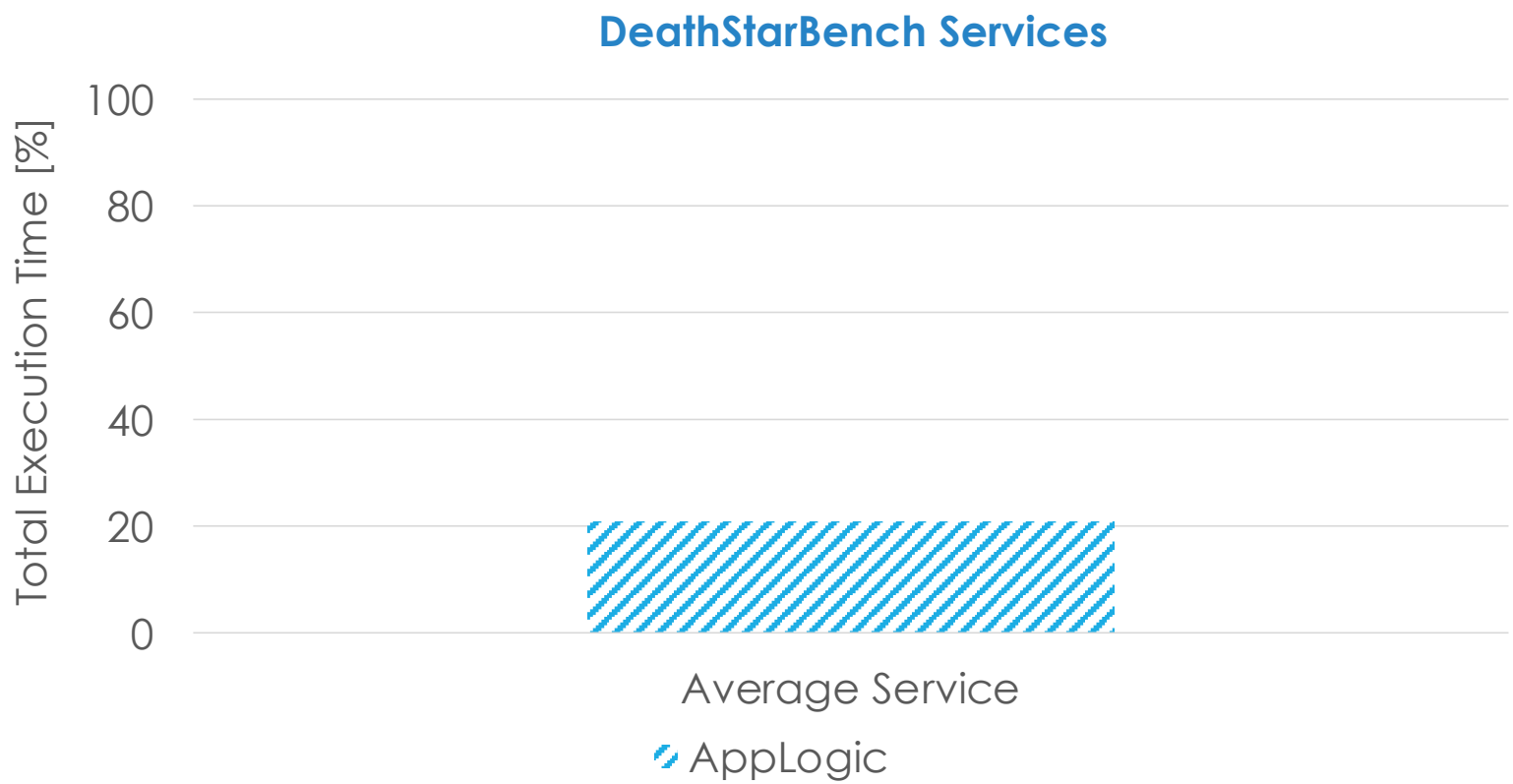
- Cloud-native services much smaller than monolithic counterparts



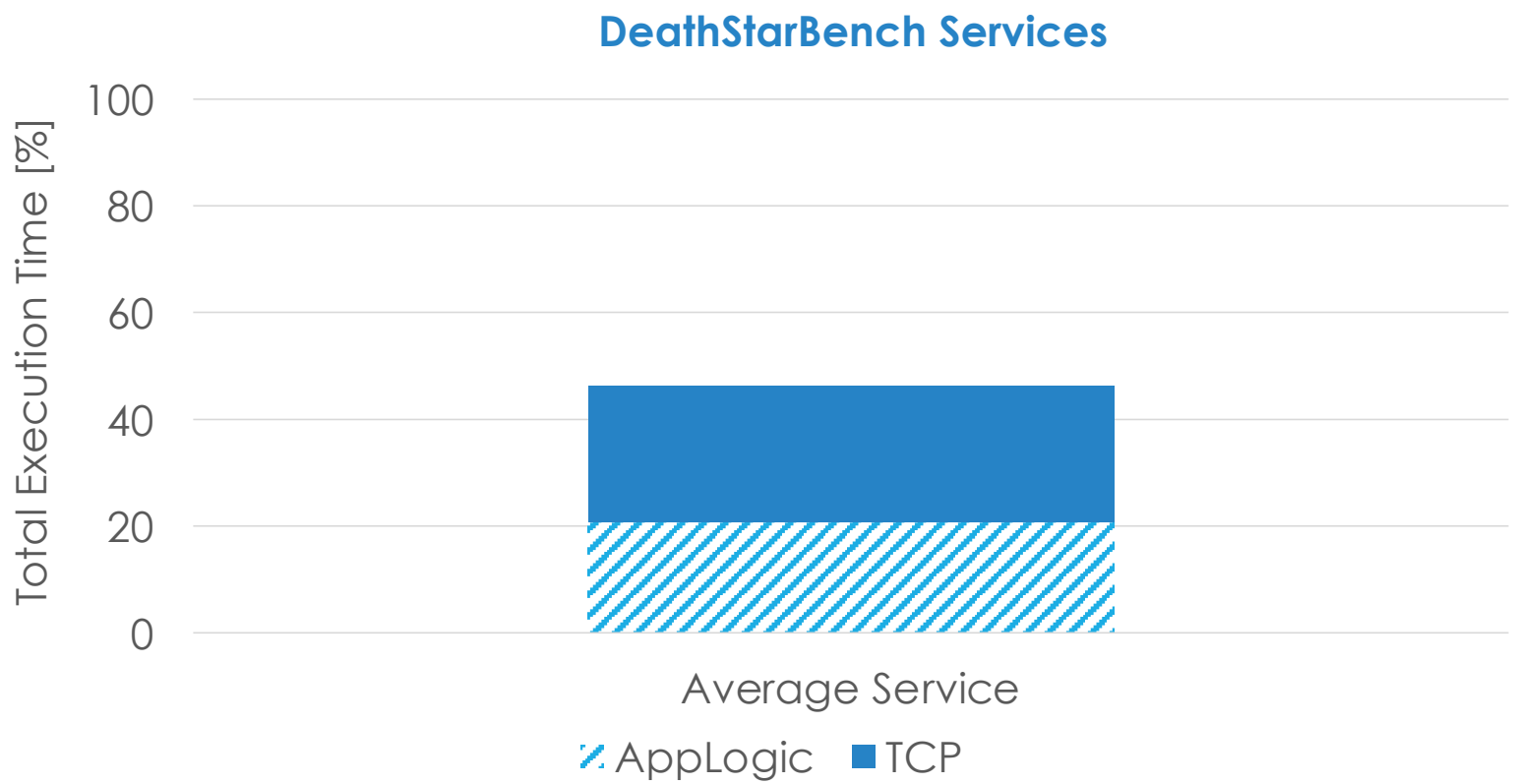
Mismatch: Current Processors vs. Cloud-Native Loads

Current Processors	Cloud-Native Environments
Beefy processors	Small-sized services; Low ILP

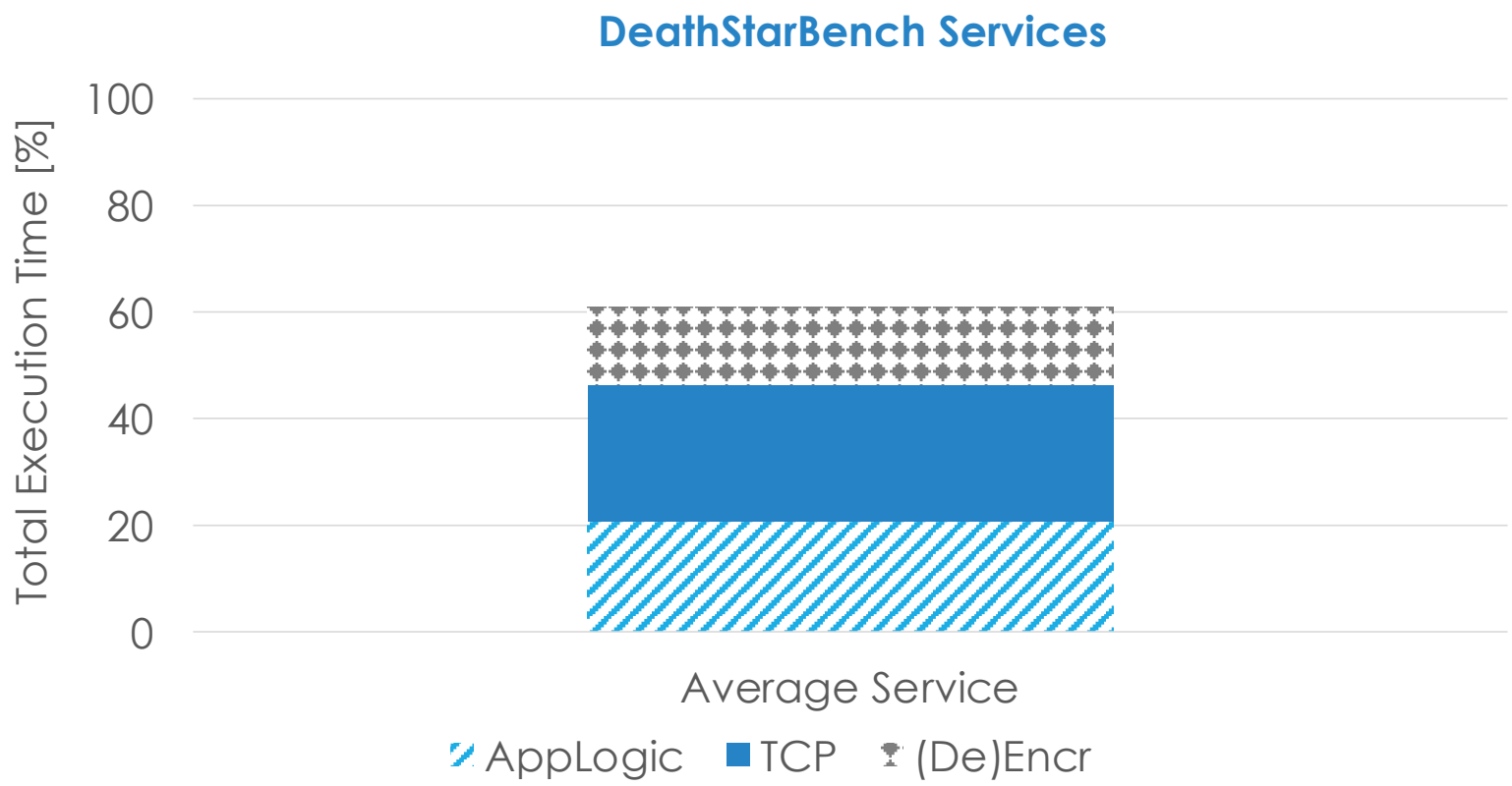
Low ILP: Datacenter Tax Dominates Execution



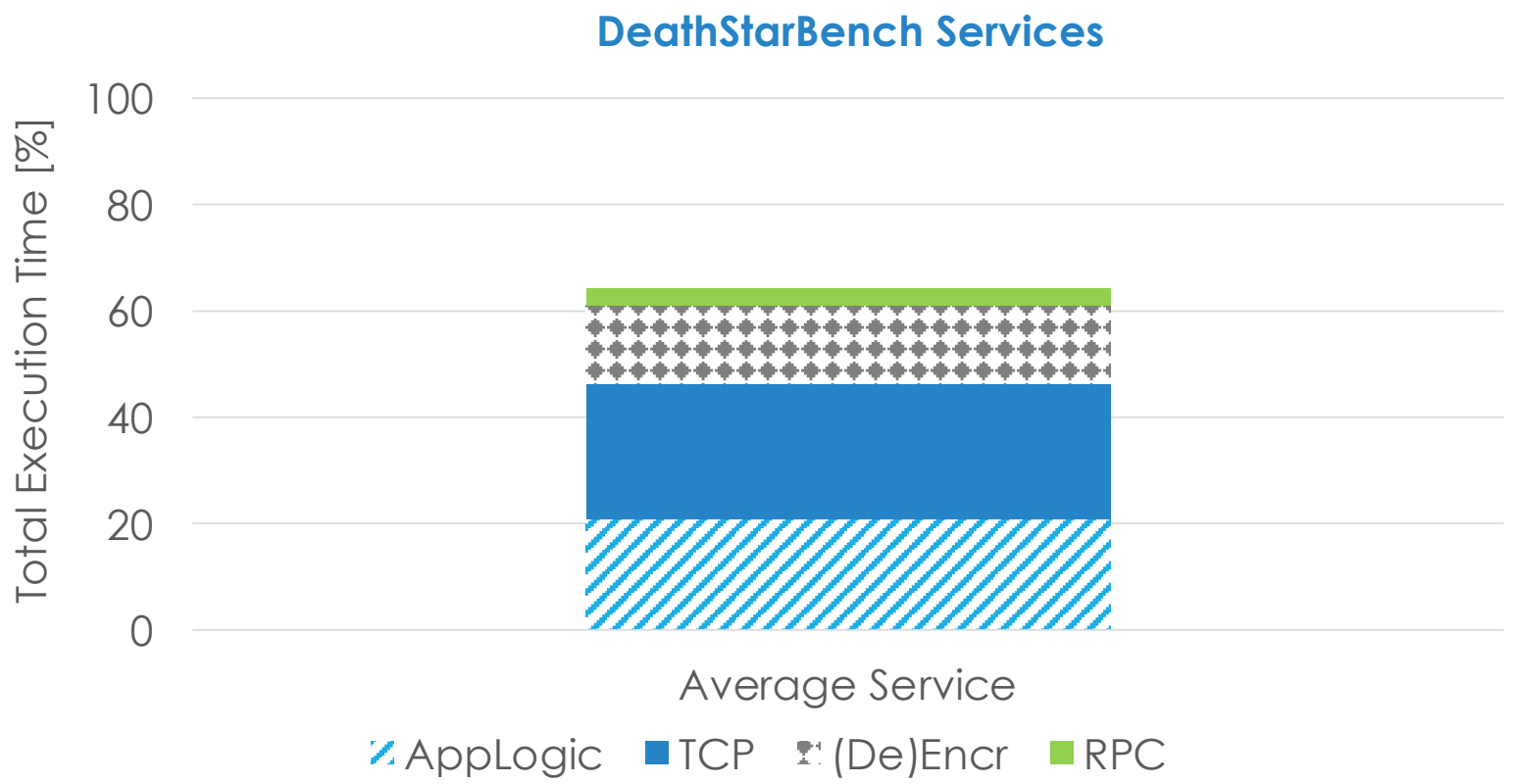
Low ILP: Datacenter Tax Dominates Execution



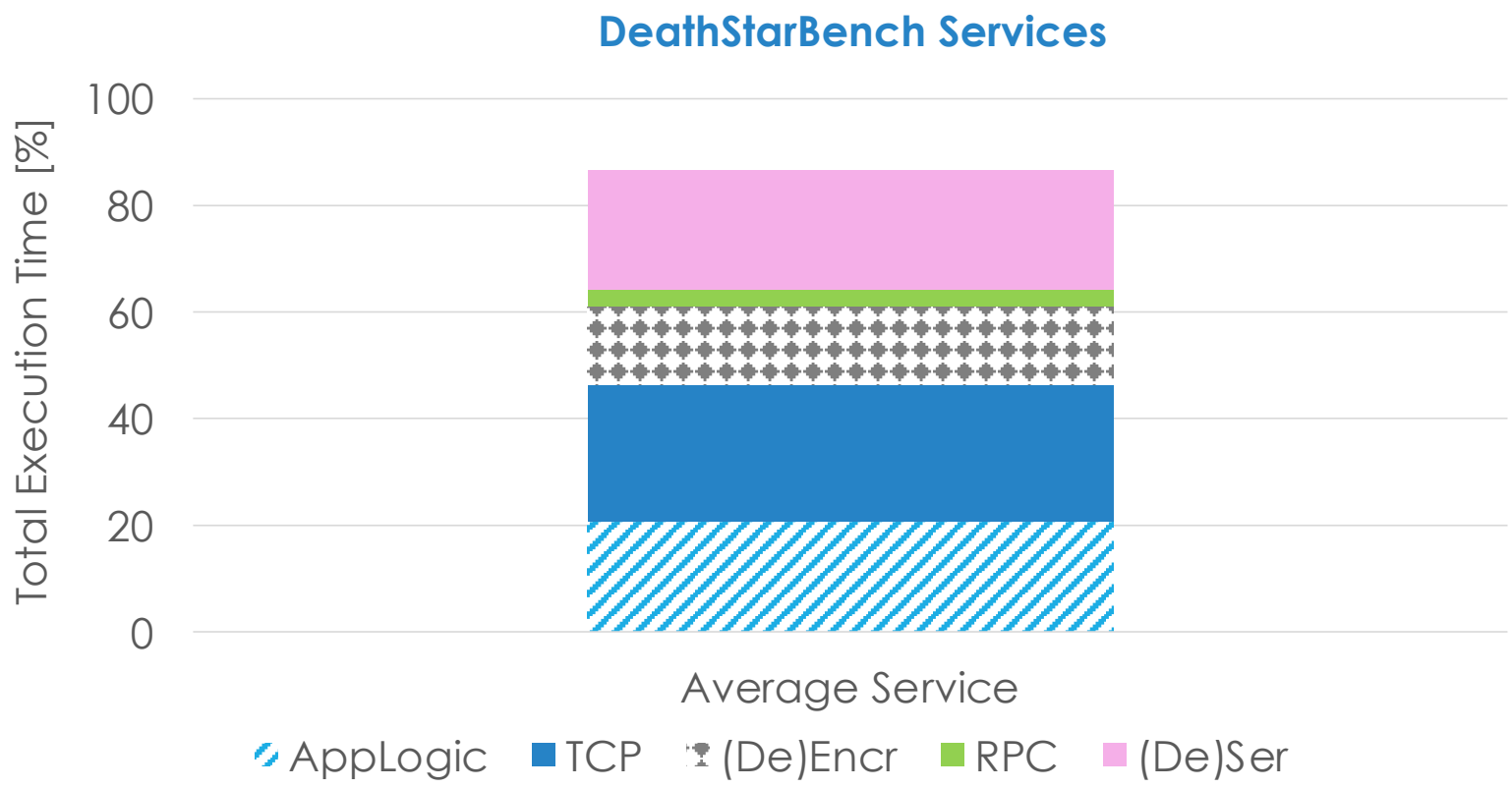
Low ILP: Datacenter Tax Dominates Execution



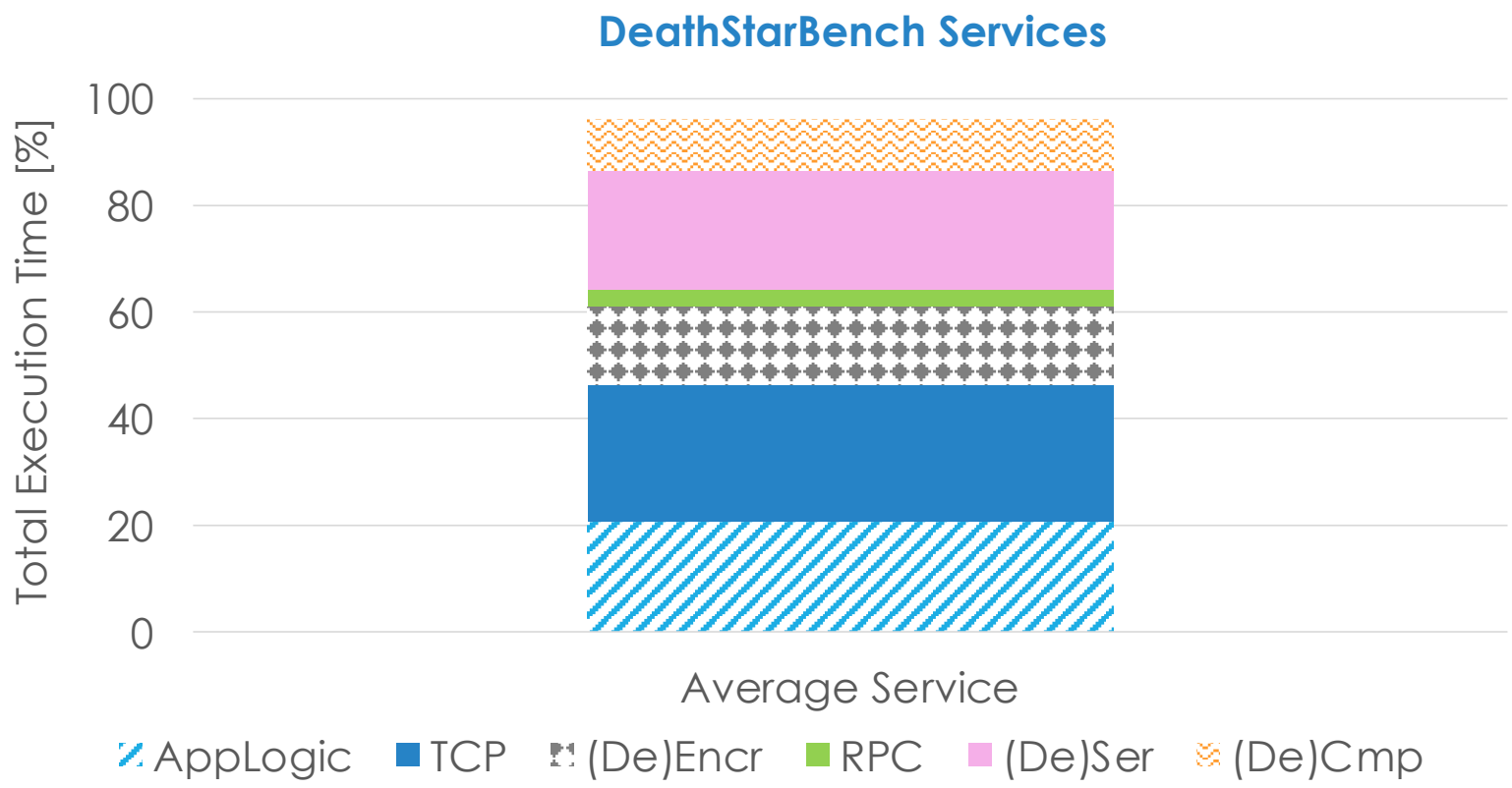
Low ILP: Datacenter Tax Dominates Execution



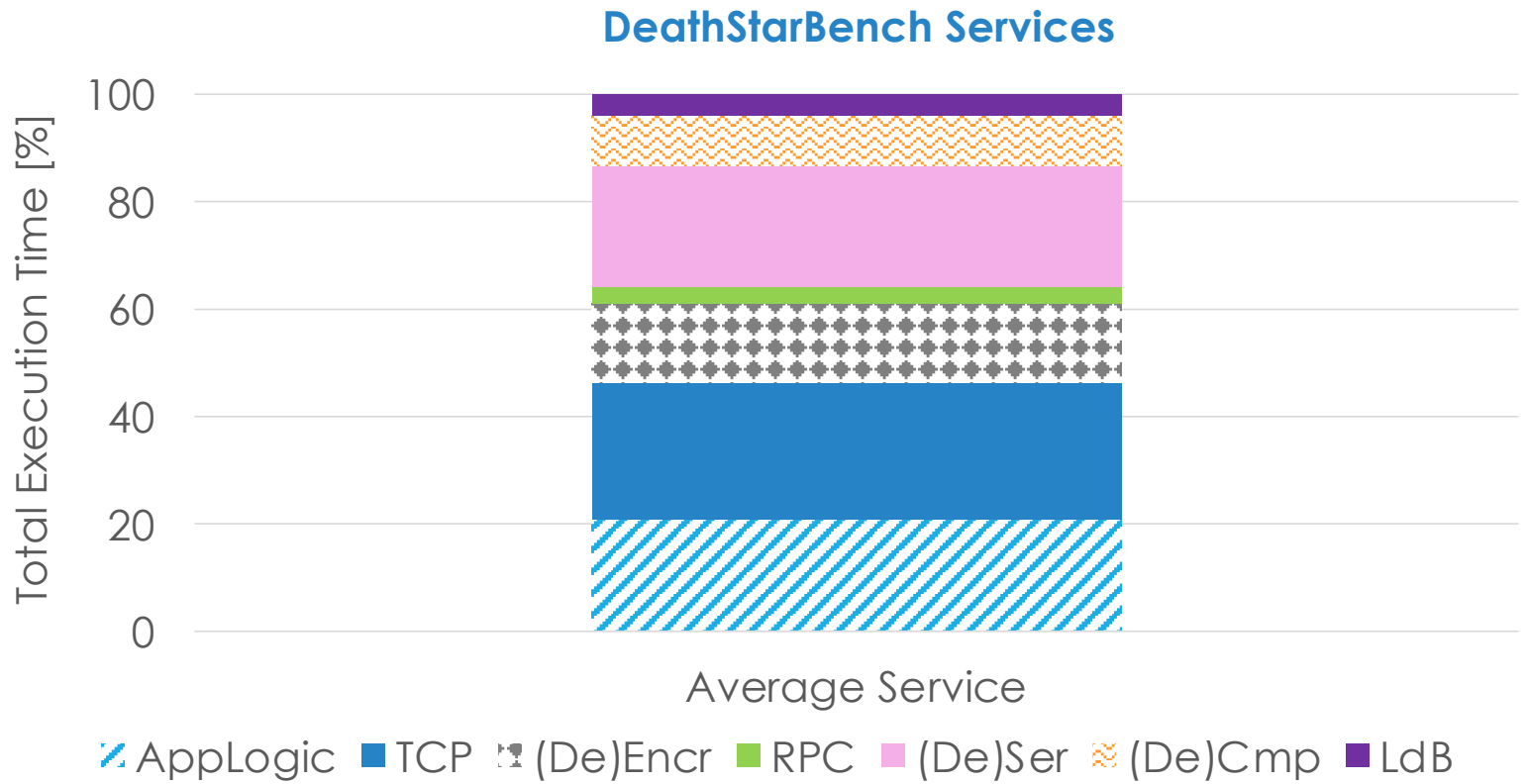
Low ILP: Datacenter Tax Dominates Execution



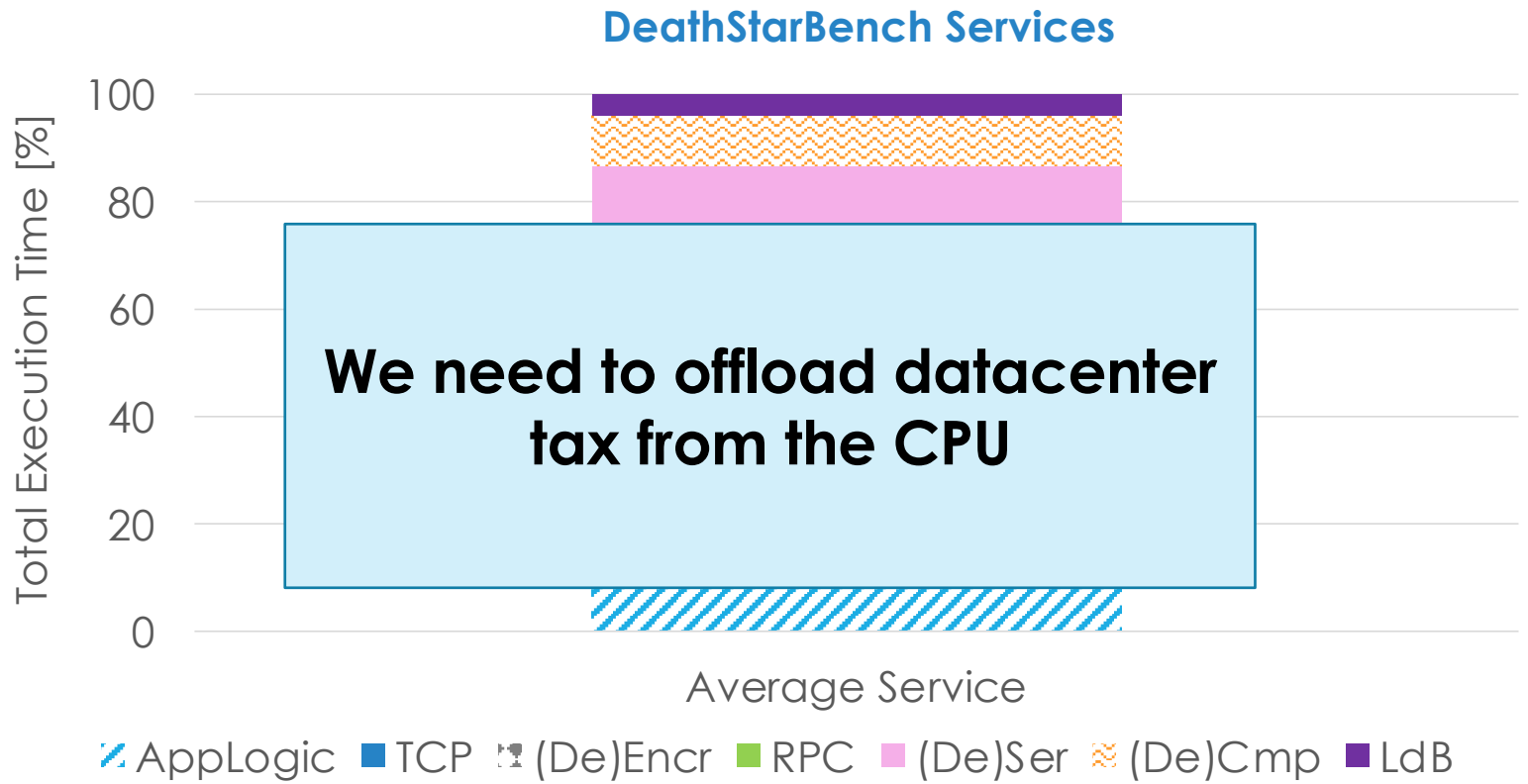
Low ILP: Datacenter Tax Dominates Execution



Low ILP: Datacenter Tax Dominates Execution



Low ILP: Datacenter Tax Dominates Execution

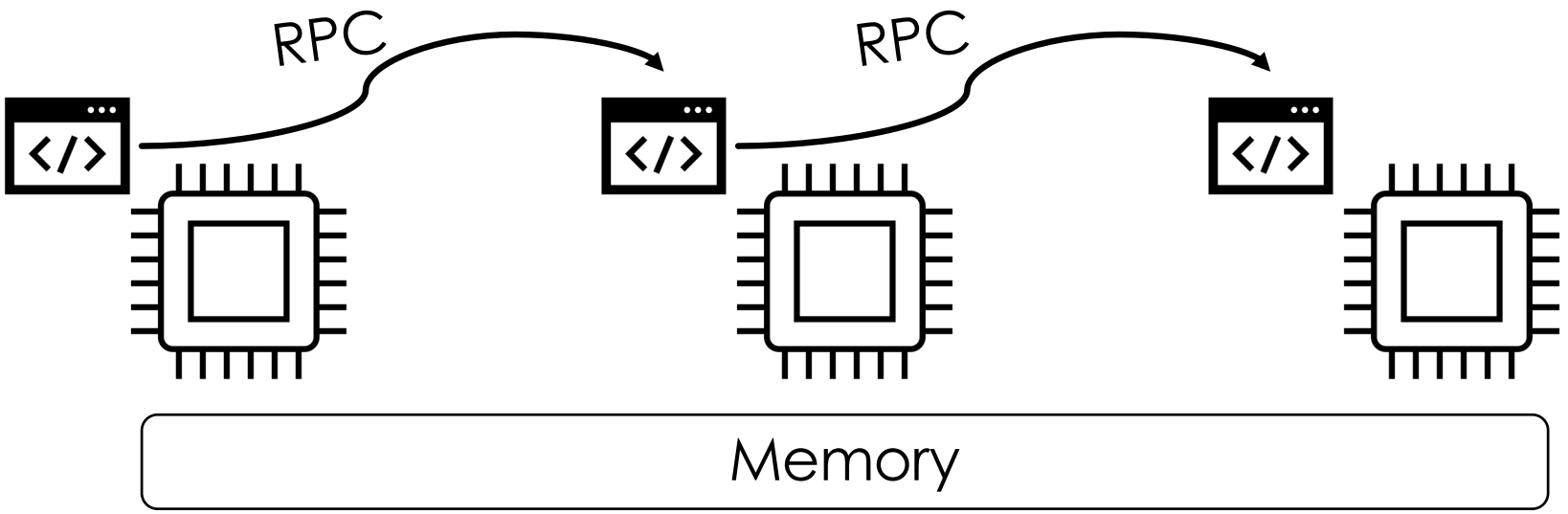


Mismatch: Current Processors vs. Cloud-Native Loads

Current Processors	Cloud-Native Environments
Beefy processors	Small-sized services; Low ILP
Monolithic cache coherence	No writable data sharing

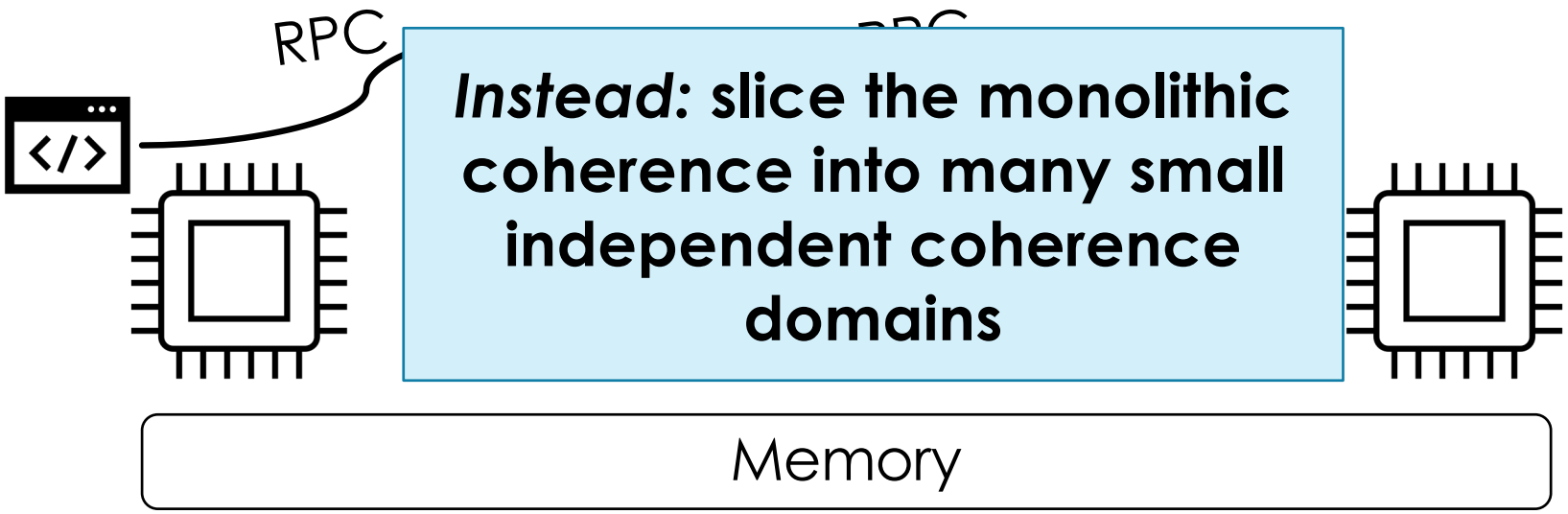
Is Chip-Wide Cache Coherence Needed?

- Services use RPCs for the communication, no shared memory



Is Chip-Wide Cache Coherence Needed?

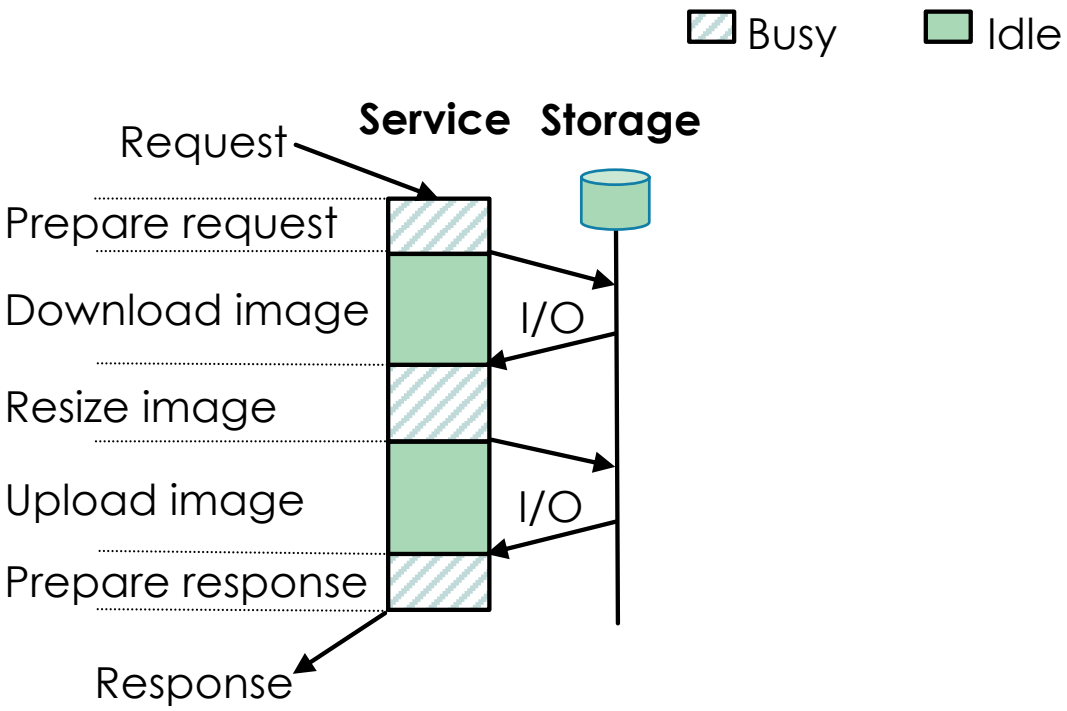
- Services use RPCs for the communication, no shared memory



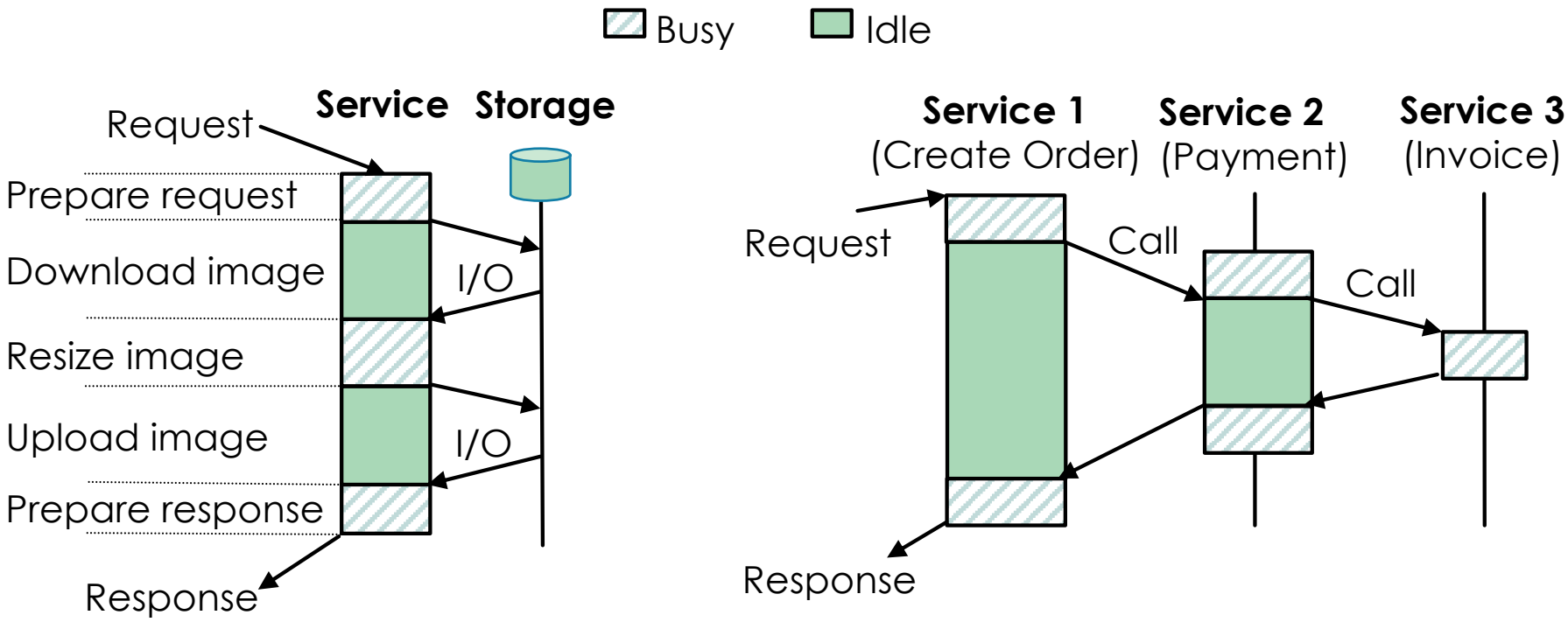
Mismatch: Current Processors vs. Cloud-Native Loads

Current Processors	Cloud-Native Environments
Beefy processors	Small-sized services; Low ILP
Monolithic cache coherence	No writable data sharing
Optimized for long-running, predictable apps	Short-running services; lots of idle time + context switching

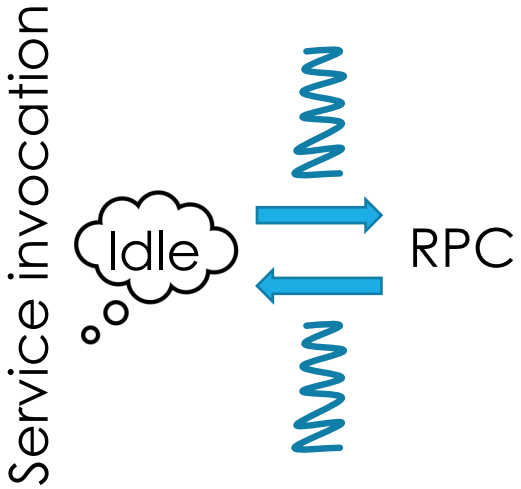
Idle Time Dominates Service Execution



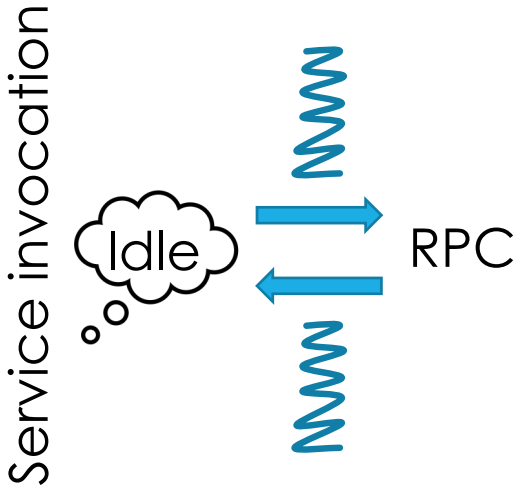
Idle Time Dominates Service Execution



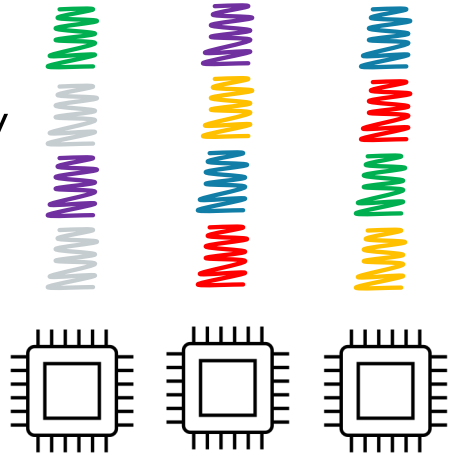
Idle Time Dominates Service Execution



As a Result: Frequent Context Switching

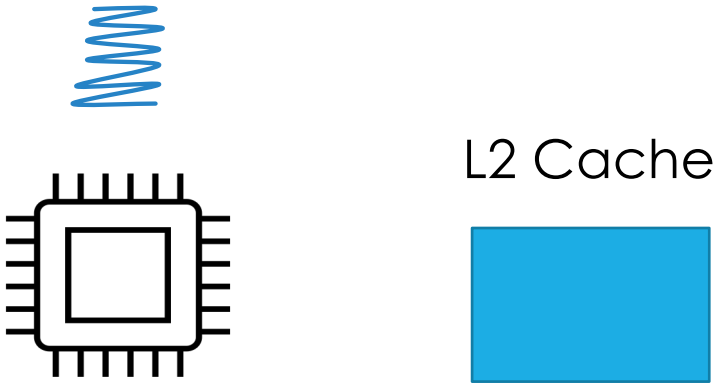


Need to frequently context switch!



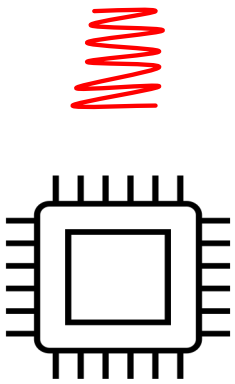
Pollution of Stateful Hardware Structures

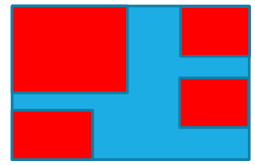
- Frequent context-switches interleave executions of different services on the same core
- Pollution of stateful structures: caches, TLB, branch predictors, etc,



Pollution of Stateful Hardware Structures

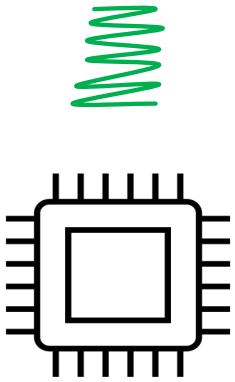
- Frequent context-switches interleave executions of different services on the same core
- Pollution of stateful structures: caches, TLB, branch predictors, etc,



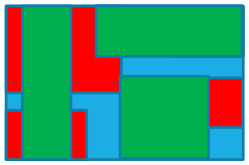


Pollution of Stateful Hardware Structures

- Frequent context-switches interleave executions of different services on the same core
- Pollution of stateful structures: caches, TLB, branch predictors, etc,

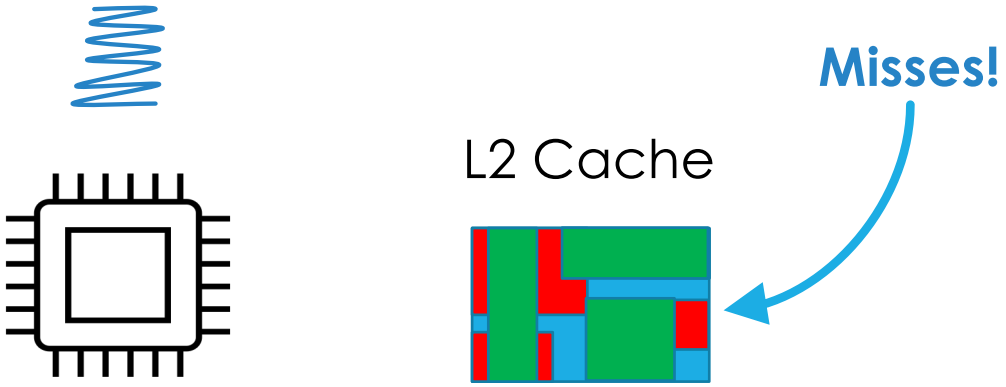


L2 Cache



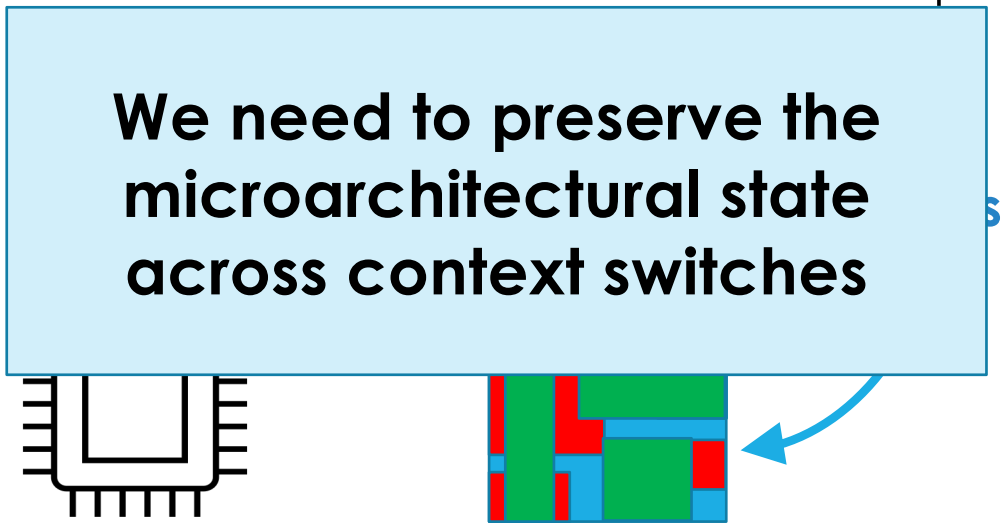
Pollution of Stateful Hardware Structures

- Frequent context-switches interleave executions of different services on the same core
- Pollution of stateful structures: caches, TLB, branch predictors, etc,



Pollution of Stateful Hardware Structures

- Frequent context-switches interleave executions of different services on the same core
- Pollution of stateful structures: caches, TLB, branch predictors, etc,



Mismatch: Current Processors vs. Cloud-Native Loads

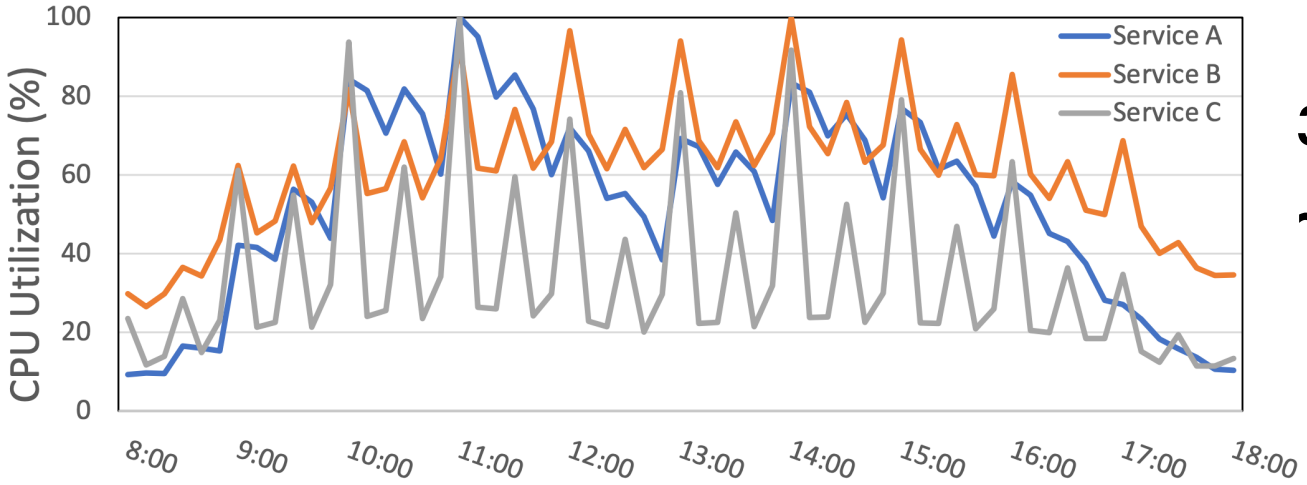
Current Processors	Cloud-Native Environments
Beefy processors	Small-sized services; Low ILP
Monolithic cache coherence	No writable data sharing
Optimized for long-running, predictable apps	Short-running services; lots of idle time + context switching
Maximize average performance	Strict tail latency requirements

What Causes High Tail Latency?

- Contention → high tail latency
  - Queueing and scheduling of service requests
  - Context switching
  - On-package network

Service Requests Come in Bursts → Queueing

- Design of the queueing system can impact tail latency

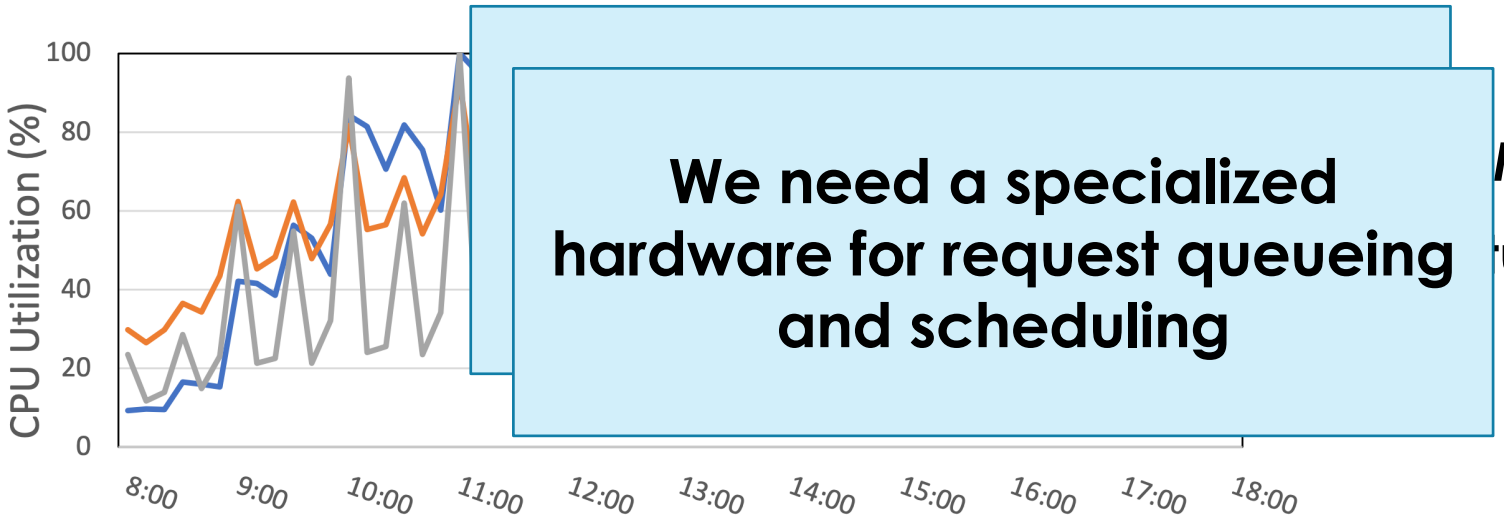


3 large Microsoft services

~1M virtual cores

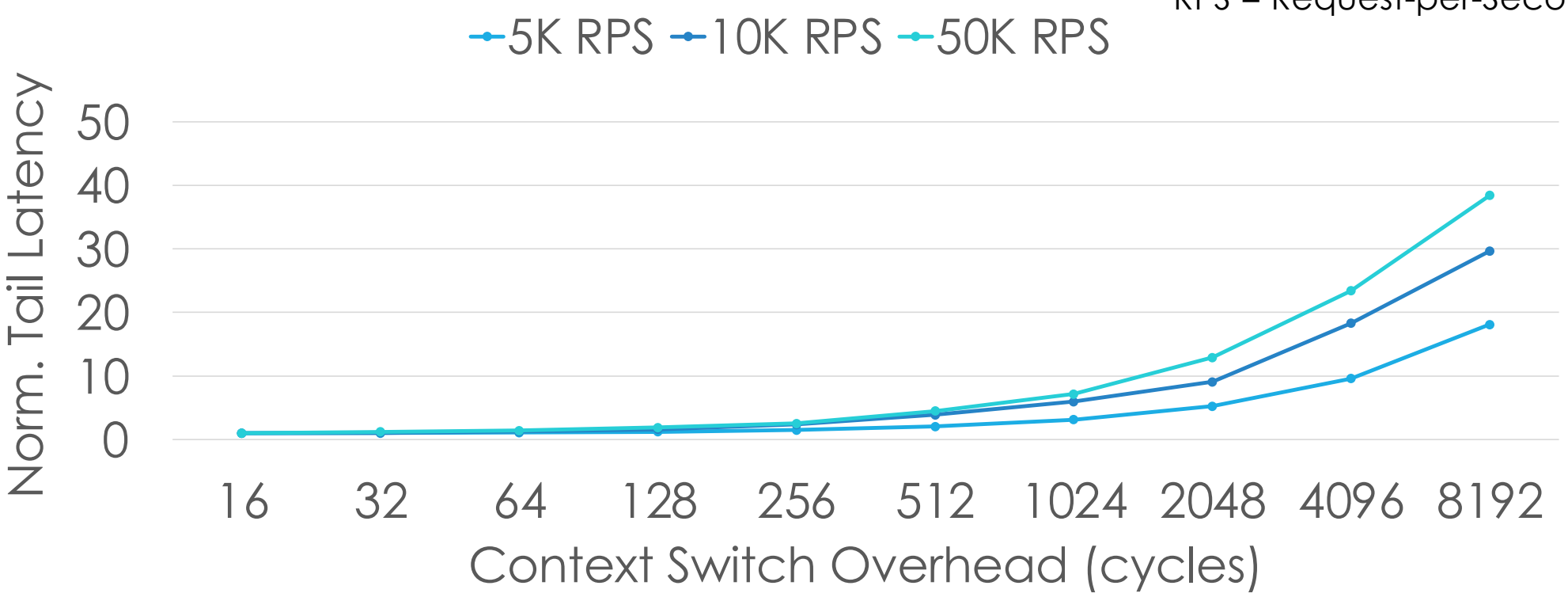
Service Requests Come in Bursts → Queueing

- Design of the queueing system can impact tail latency



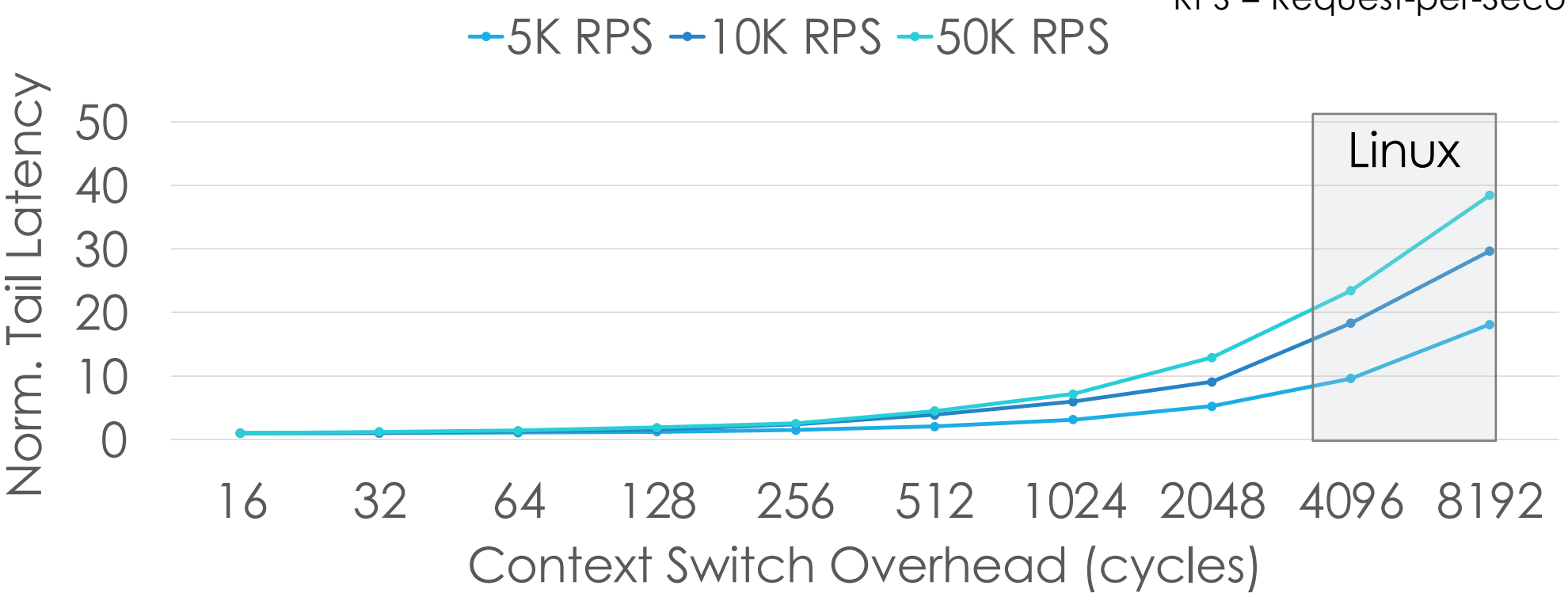
Overhead of Context Switching

RPS = Request-per-Second



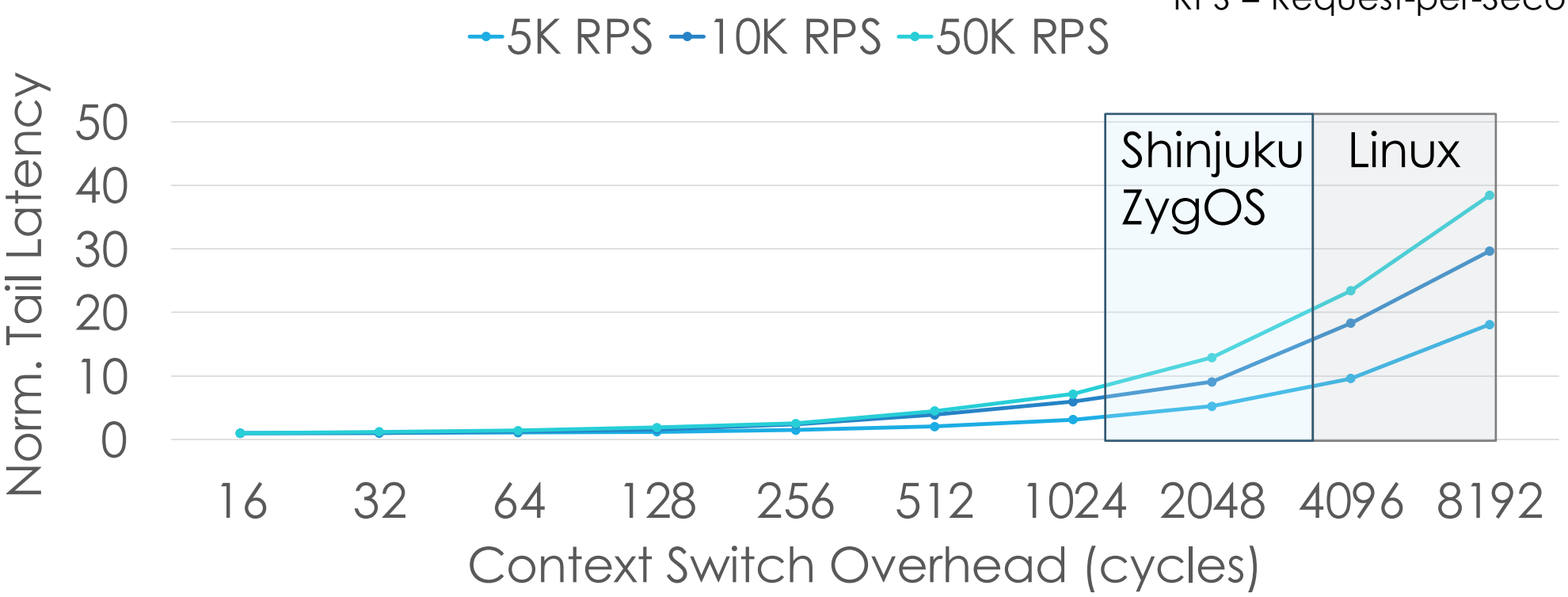
Overhead of Context Switching

RPS = Request-per-Second

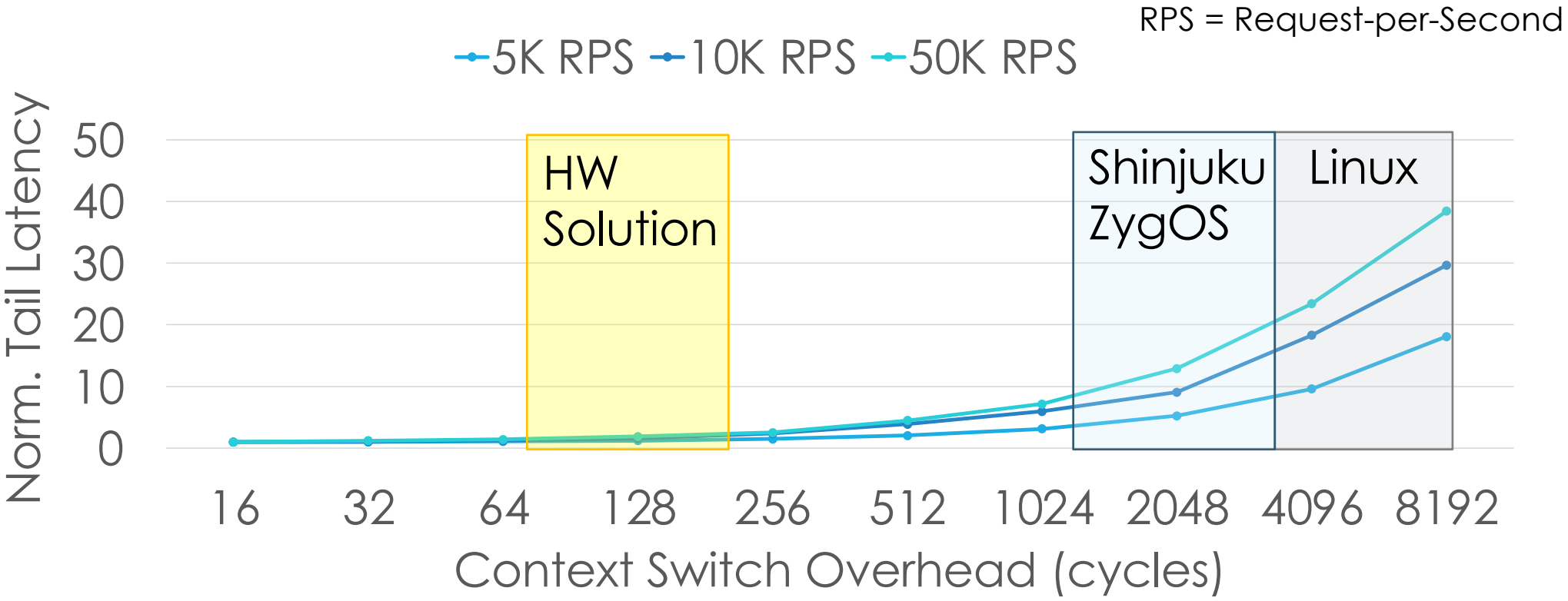


Overhead of Context Switching

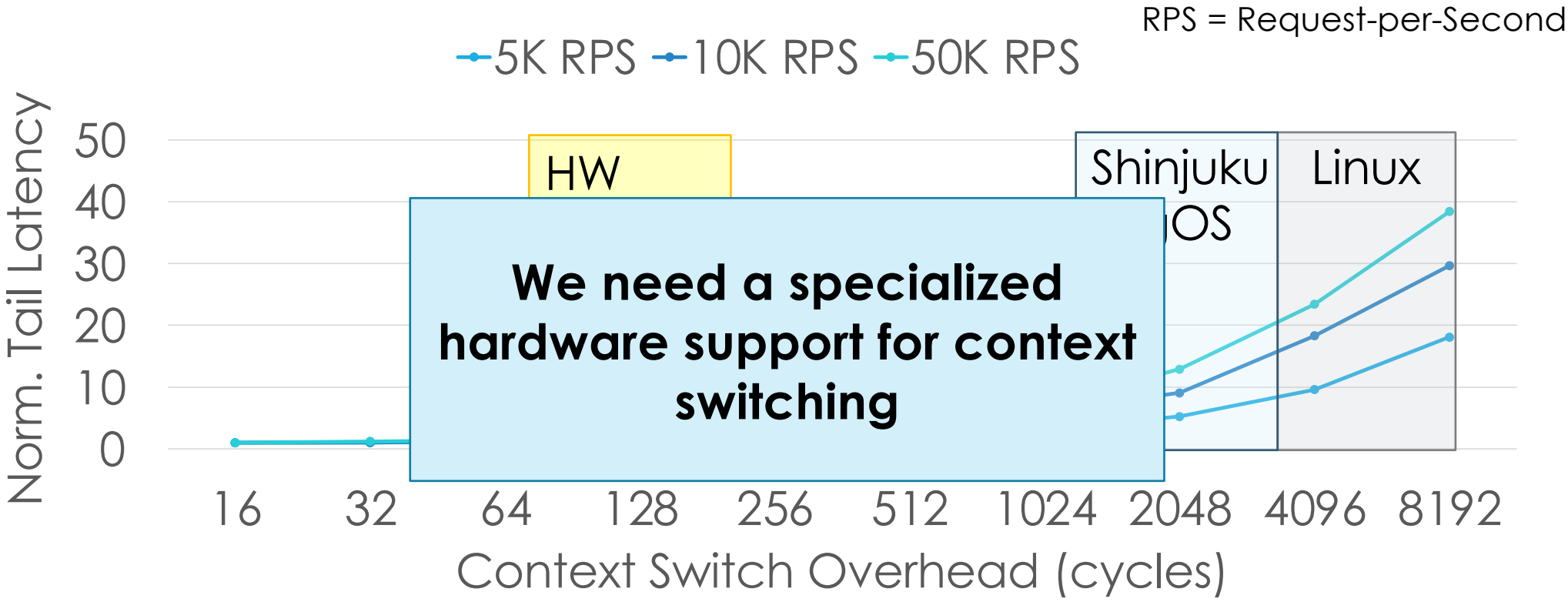
RPS = Request-per-Second



Overhead of Context Switching



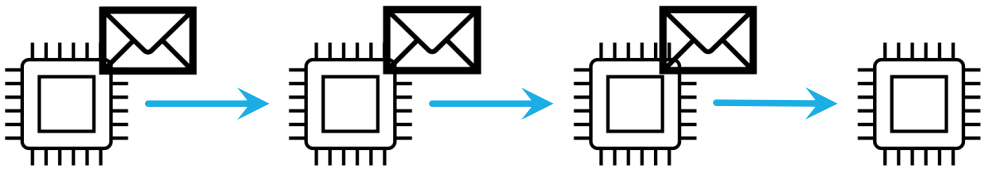
Overhead of Context Switching



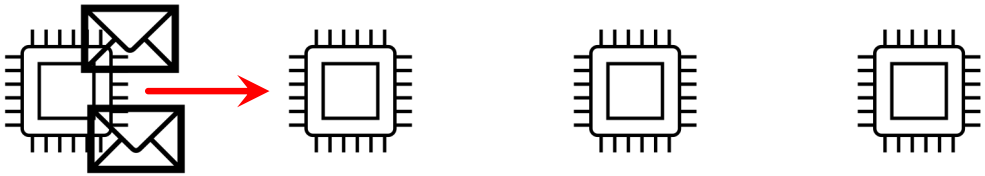
Contention in On-Package Network

- Inter-process communication due to RPCs and storage accesses
  - Lots of on-package messages

Multi-hop routes

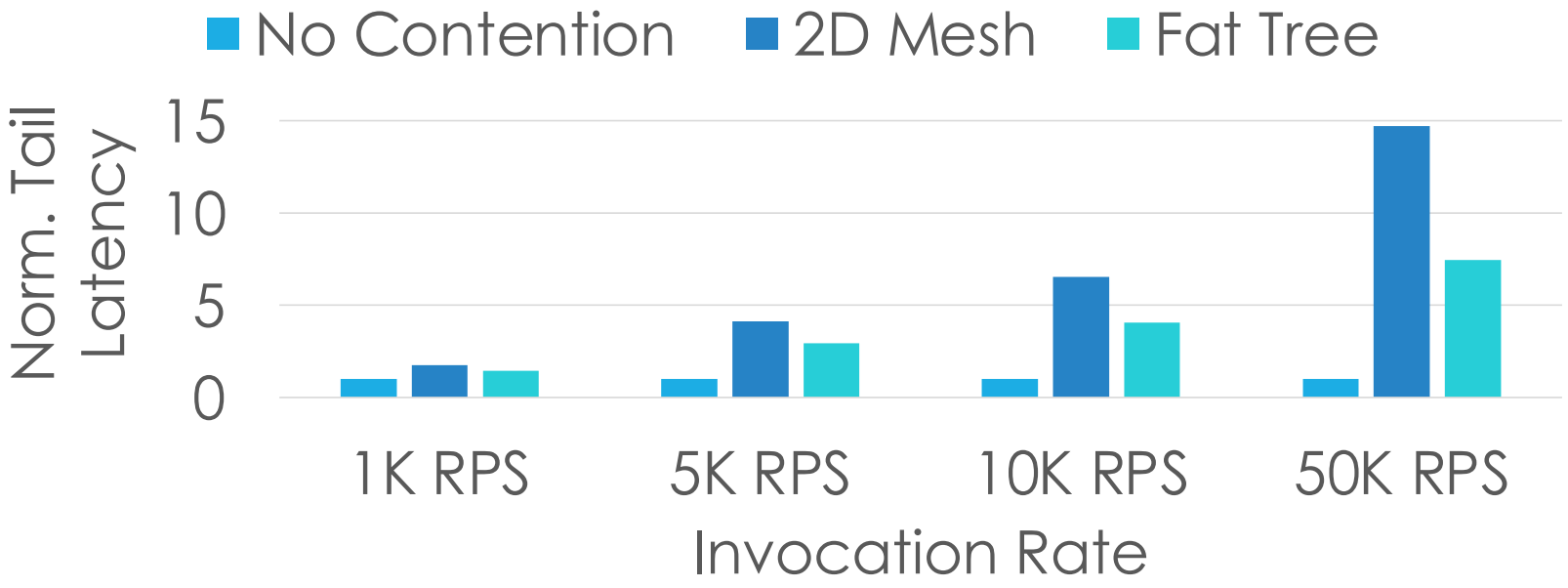


Contention at links



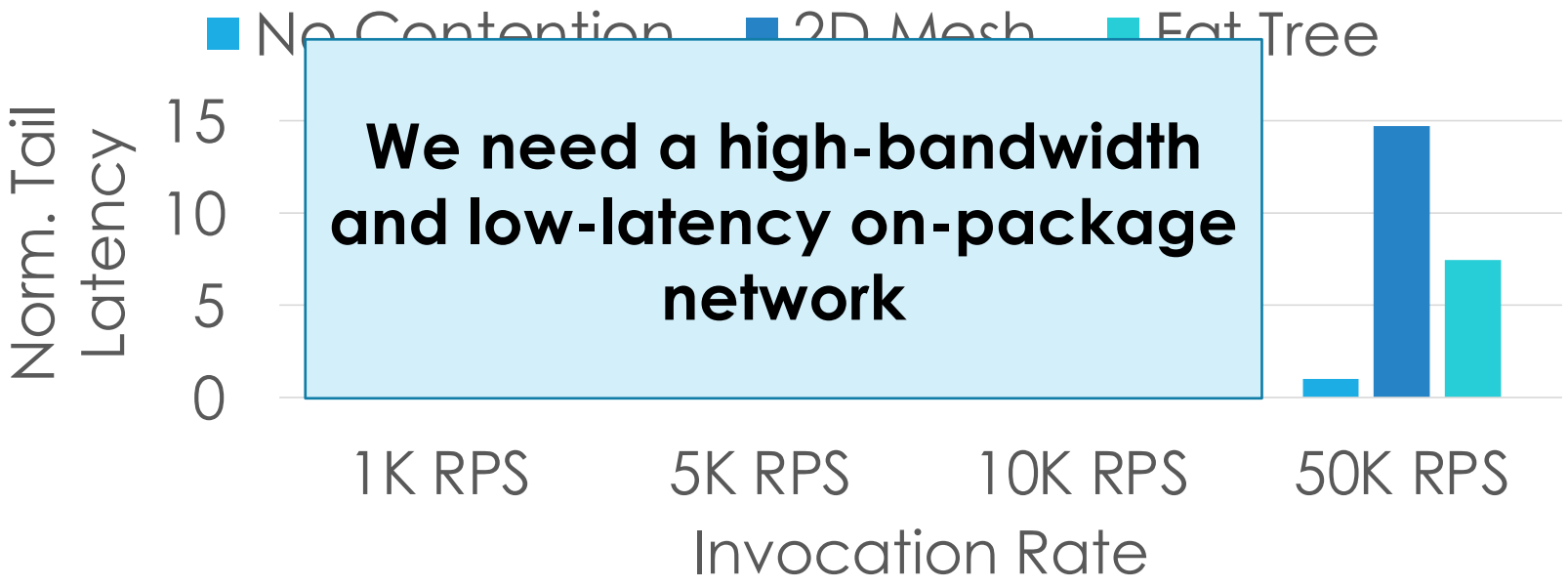
Contention in On-Package Network

- Contention at the on-package network hurts tail latency



Contention in On-Package Network

- Contention at the on-package network hurts tail latency



Summary

Cloud-native workloads on conventional HW and SW systems → performance, energy, and resource inefficiencies