PhaseWeave: Phase-Aware Execution on Heterogeneous Chiplet Architectures for Datacenters

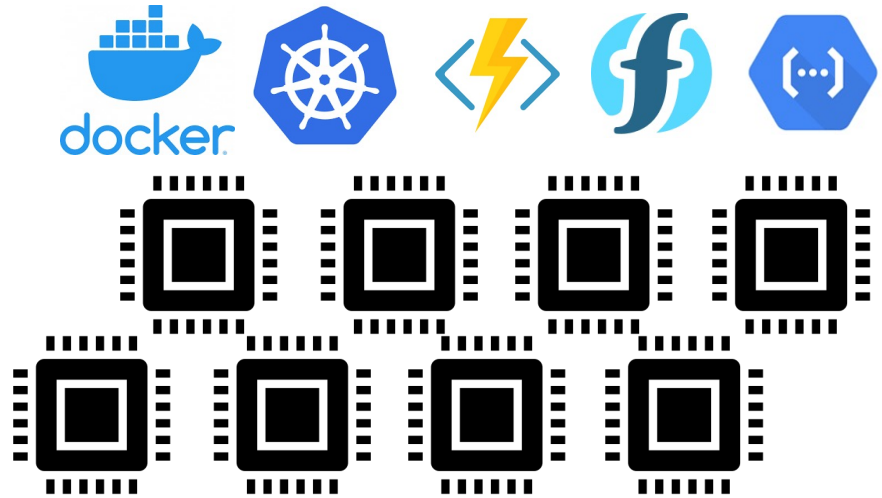
Joshua Kim[1], Chaojie Zhang[2], Íñigo Goiri[2], Christopher J. Rossbach[1,2], Jovan Stojkovic[1,3]

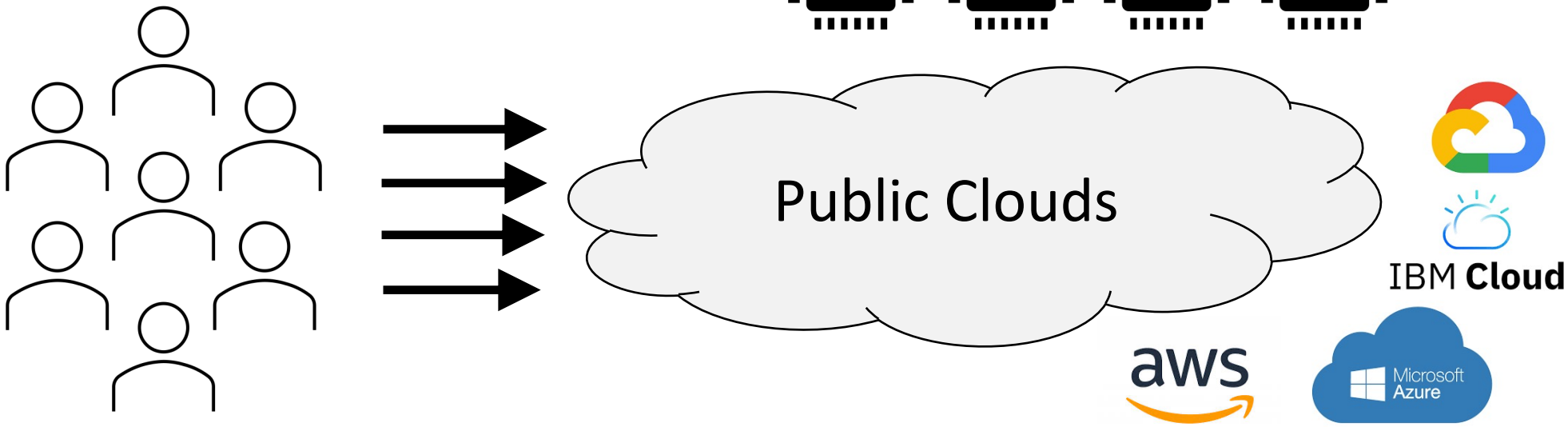
[1]The University of Texas at Austin, [2]Microsoft, [3]Meta

The Growth of Cloud Computing

Dominant Computing Paradigms:

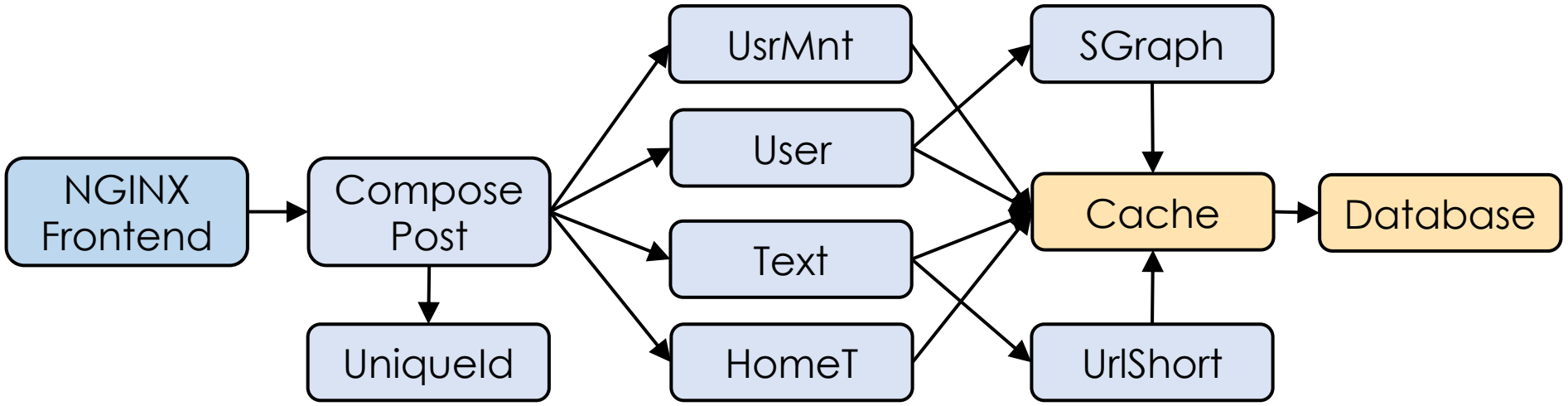
- ***Microservices***
- ***Serverless or Function-as-a-Service (FaaS)***





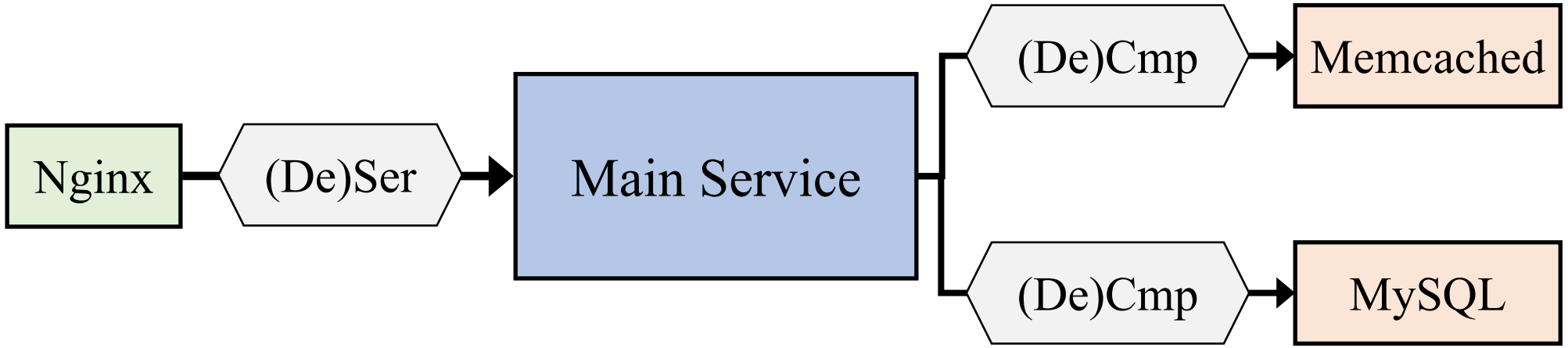
Microservices

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



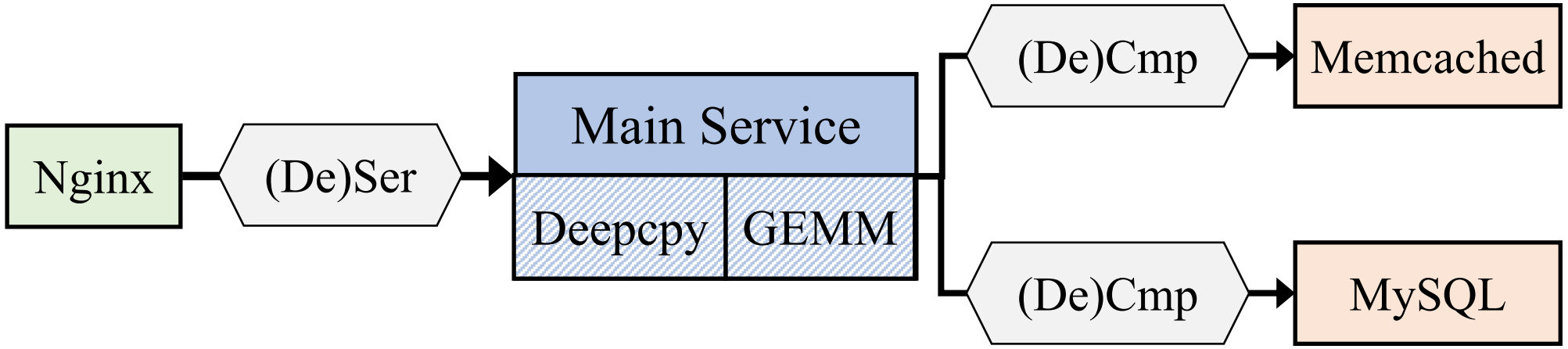
Understanding Datacenter Workload Structure

- We observe DCPerf benchmark applications that mimic Meta's production workloads



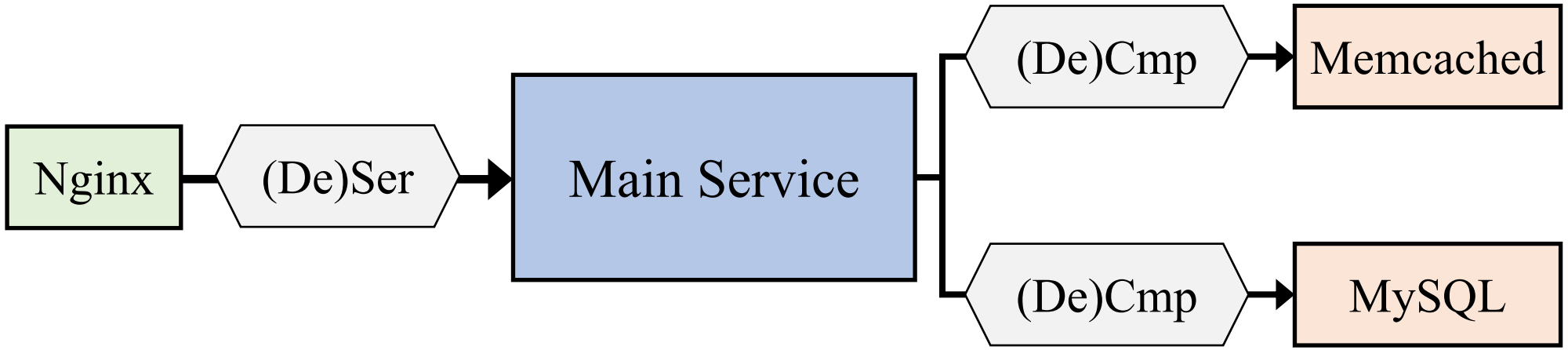
Understanding Datacenter Workload Structure

- We observe DCPerf benchmark applications that mimic Meta's production workloads



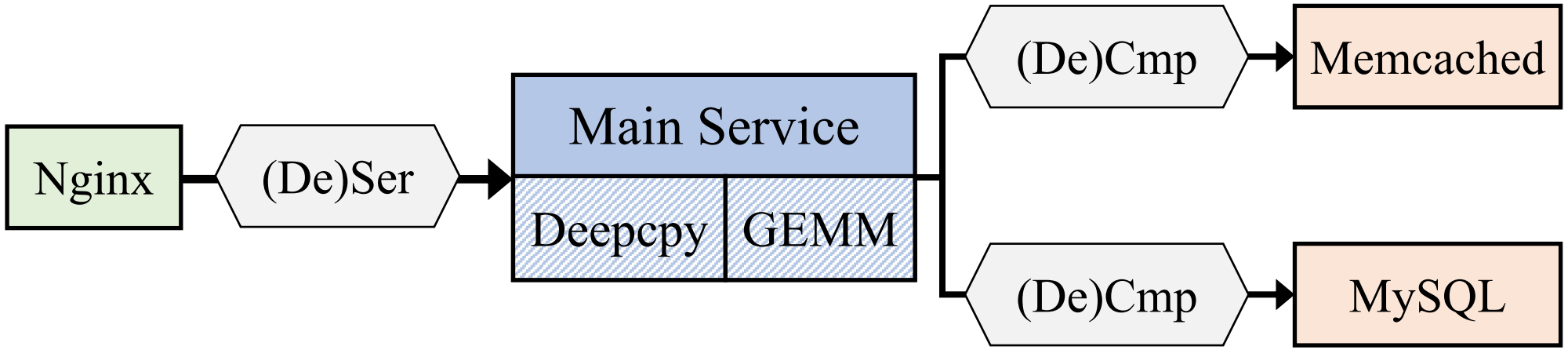
Understanding Datacenter Workload Structure

- We observe DCPerf benchmark applications that mimic Meta's production workloads



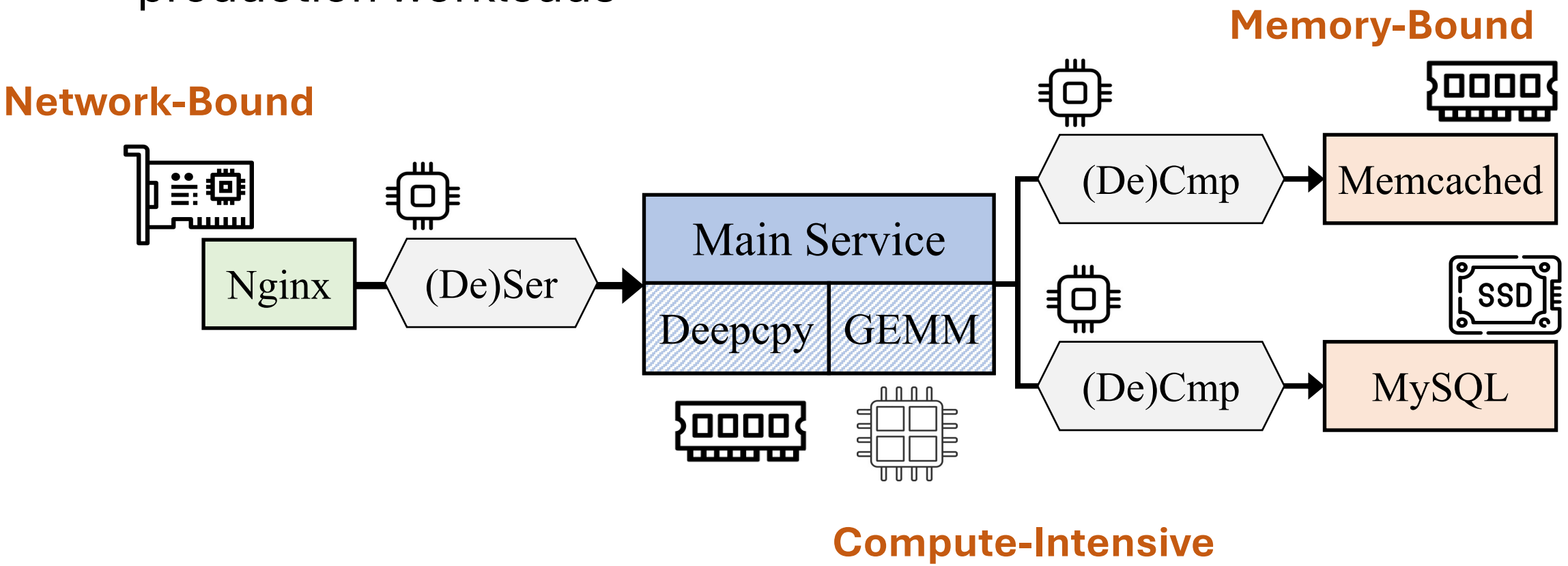
Understanding Datacenter Workload Structure

- We observe DCPerf benchmark applications that mimic Meta's production workloads



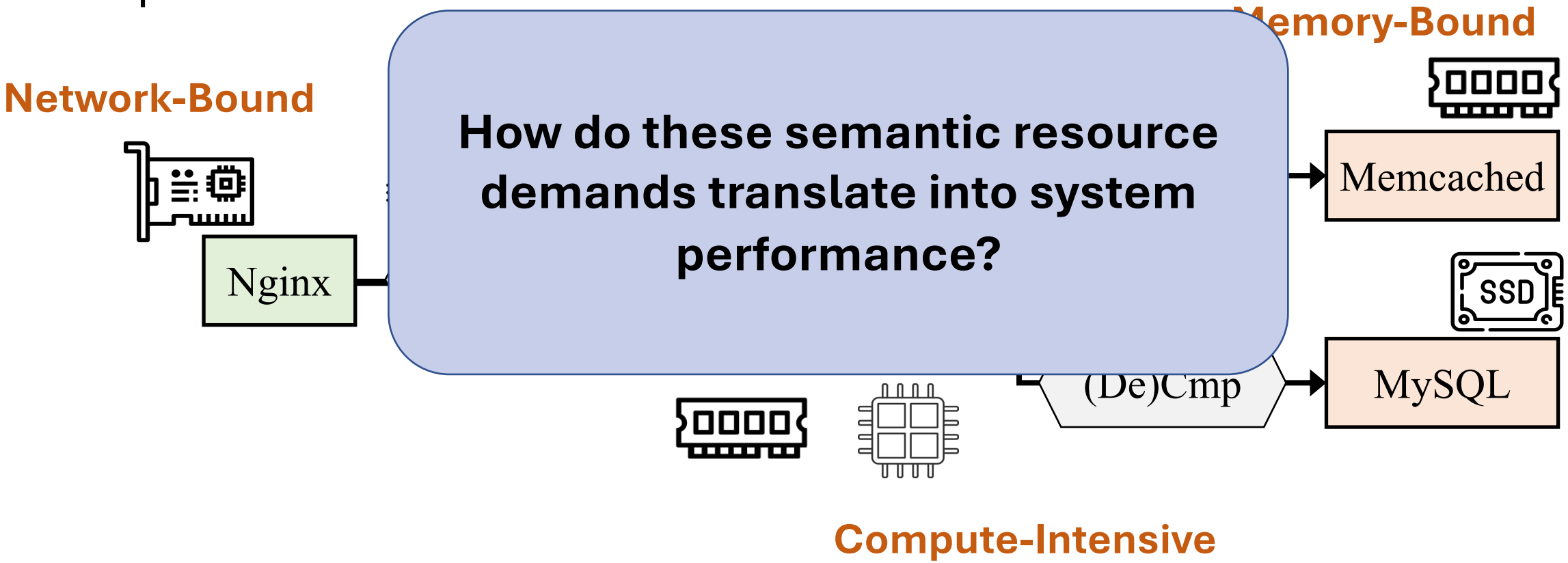
Understanding Datacenter Workload Structure

- We observe DCPerf benchmark applications that mimic Meta's production workloads



Understanding Datacenter Workload Structure

- We observe DCPerf benchmark applications that mimic Meta's production workloads



Characterizing Datacenter Workloads

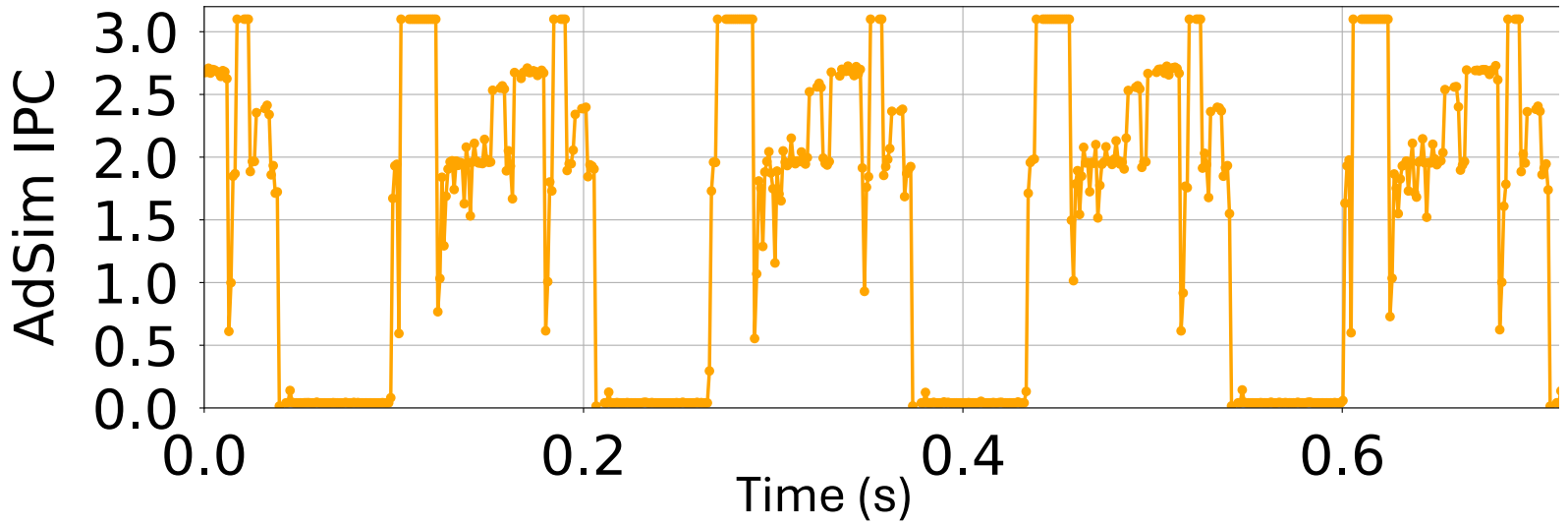
- Profile of DCPerf benchmarks running on an Emerald Rapids server

DCPerf benchmarks mimicking Meta's production workloads
An Emerald Rapids server
Put figure 2 => recurring theme of shifts in their IPC that indicate phases with distinct compute requirement
High IPC = compute, but also see medium and low IPC which indicate memory or network bound events
Label figure 2 with pointers that correlate IPC with different phases (focus on one, then zoom out to show generality)
Why? What are the sources?
First intuitive => quantify

Characterizing Datacenter Workloads

- Profile of DCPerf benchmark running on an Emerald Rapids server

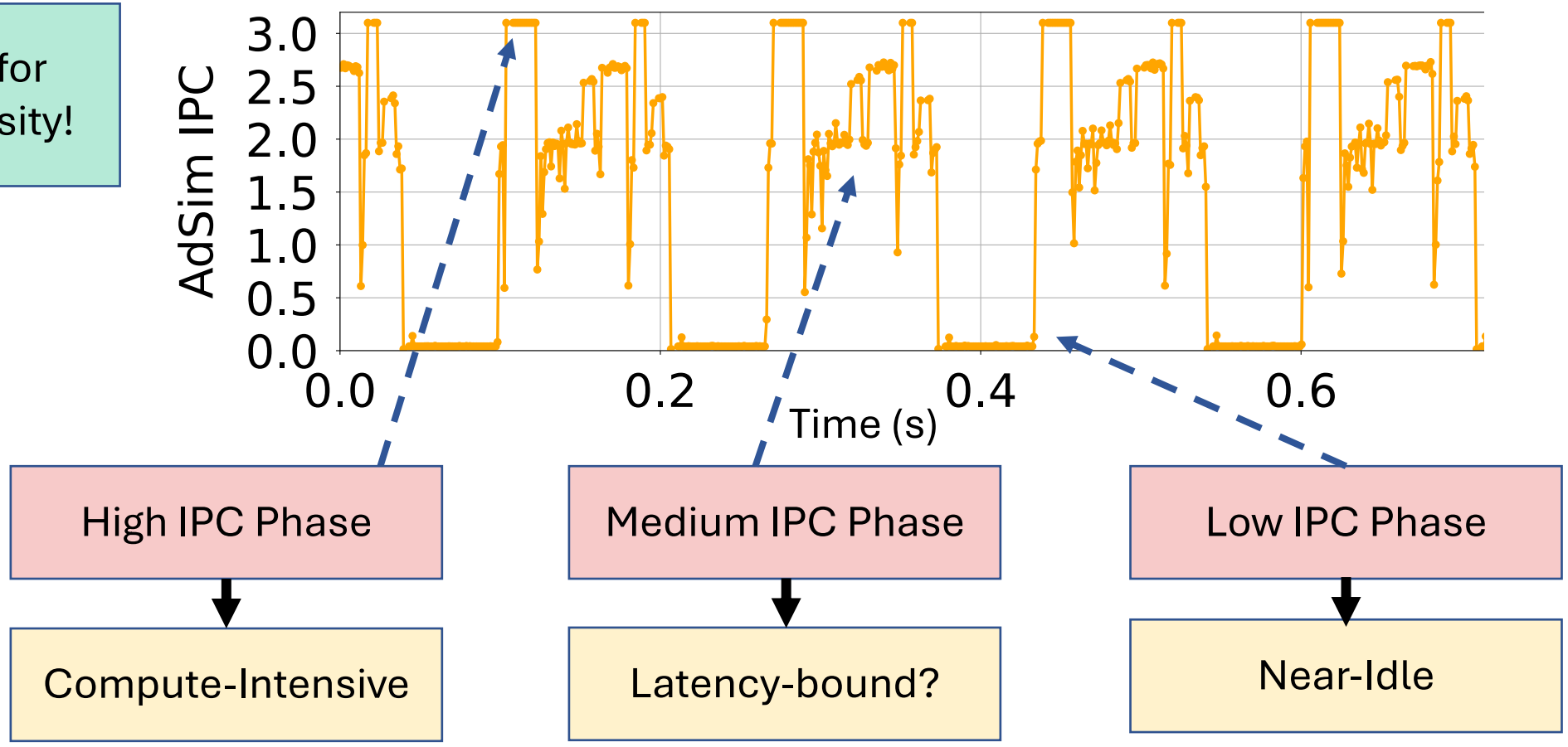
IPC indicator for compute-intensity!



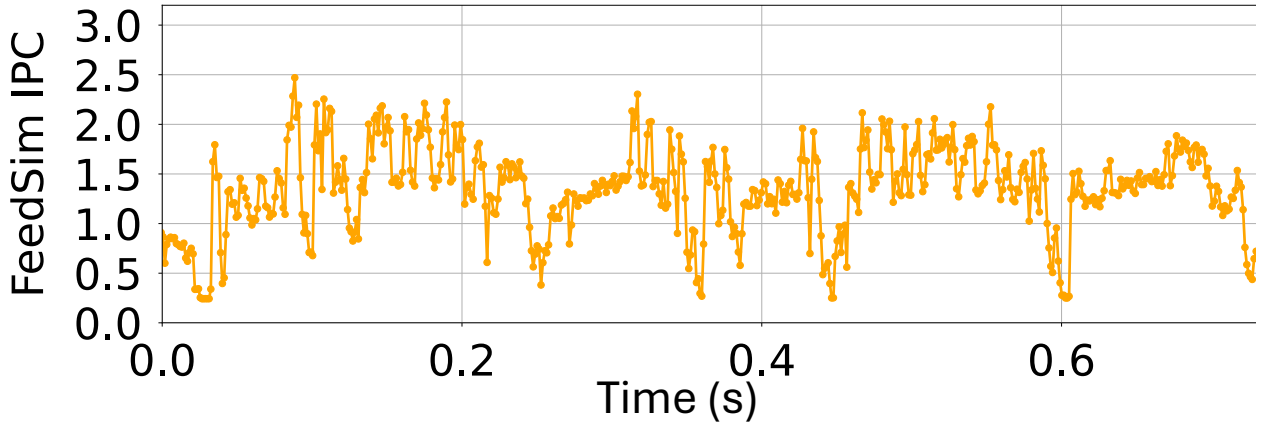
Characterizing Datacenter Workloads

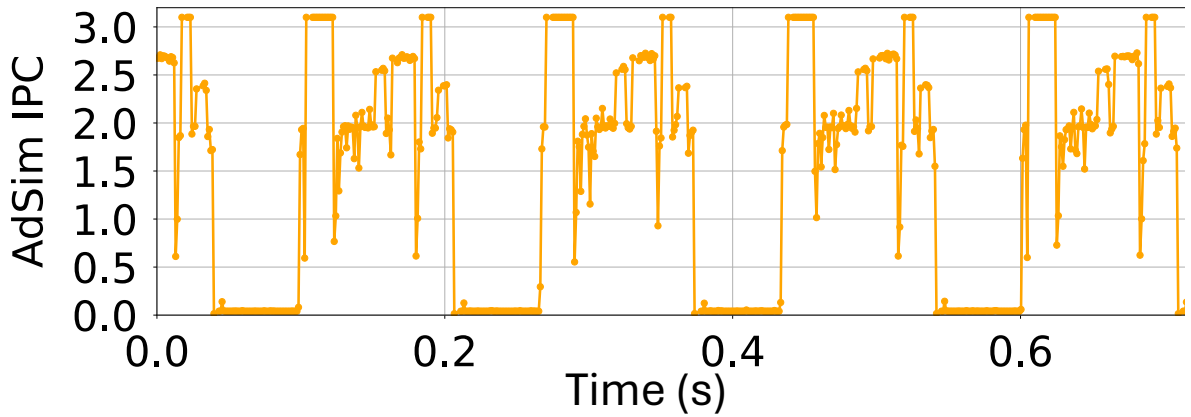
- Profile of DCPerf benchmark running on an Emerald Rapids server

IPC indicator for compute-intensity!

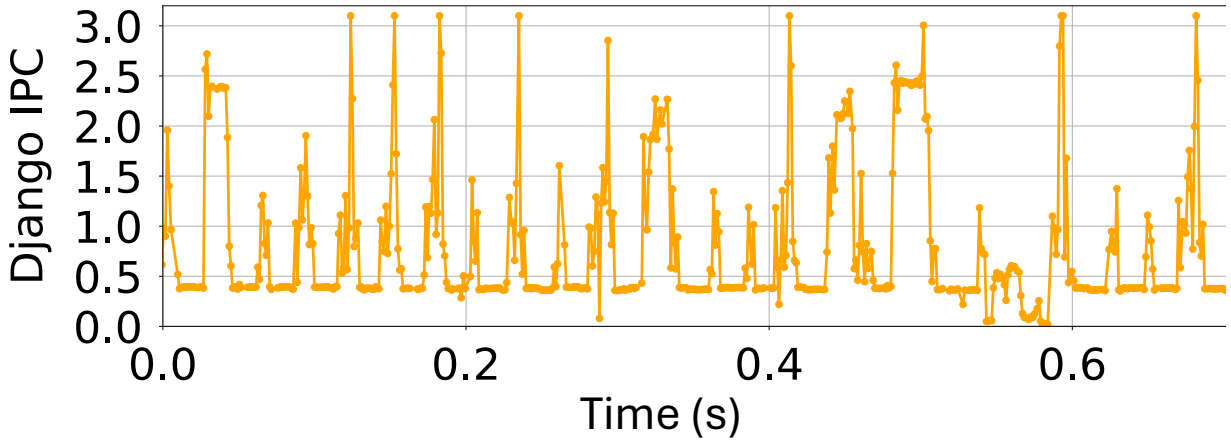


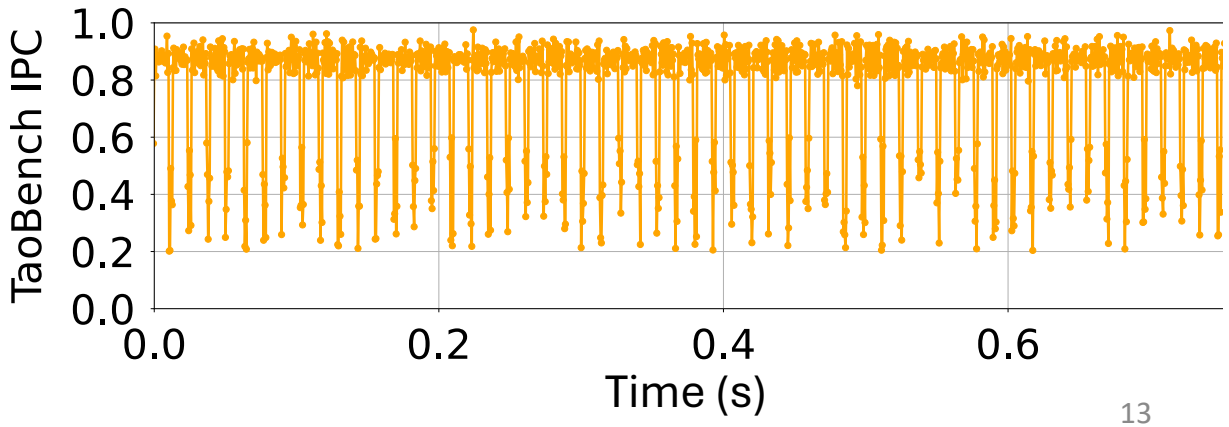
Characterizing Datacenter Workloads



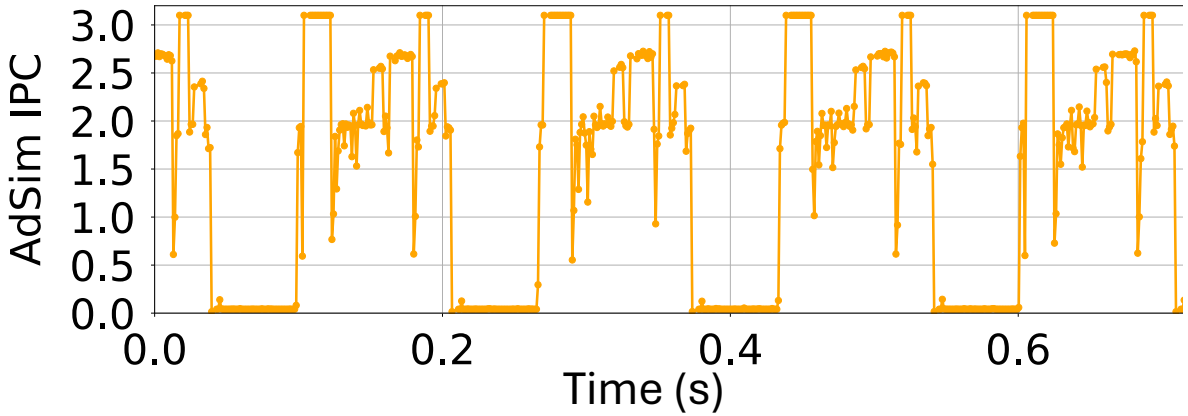


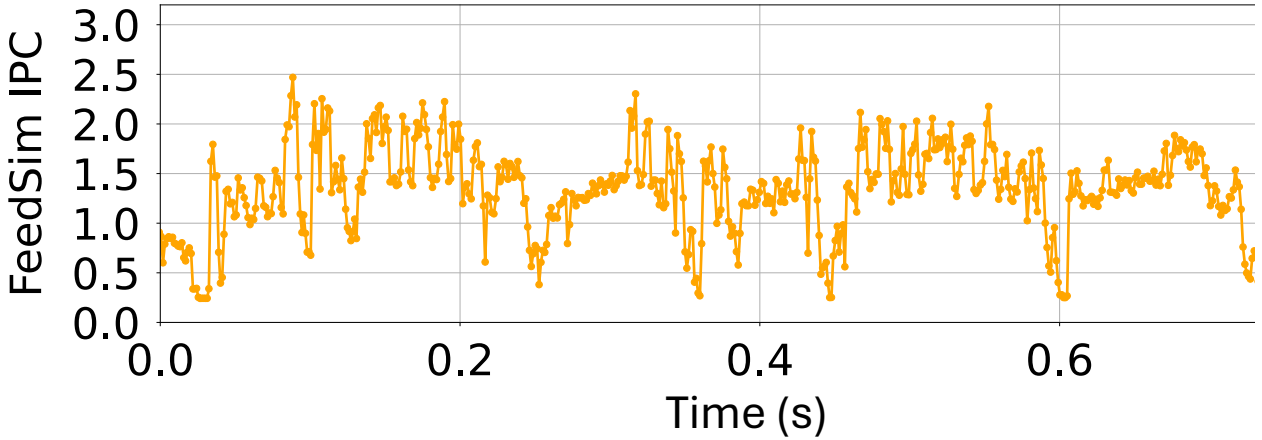
Different IPC-level phases: **What drives this?**



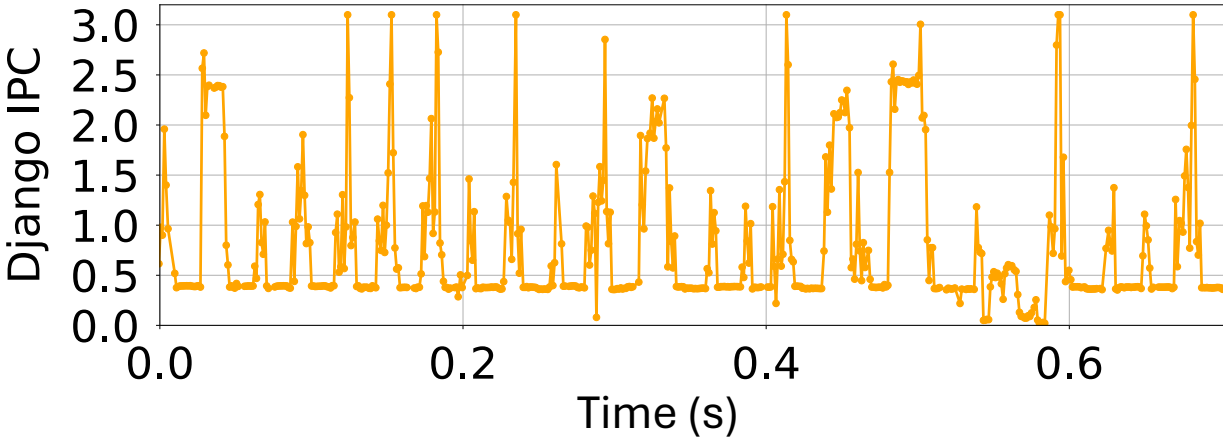


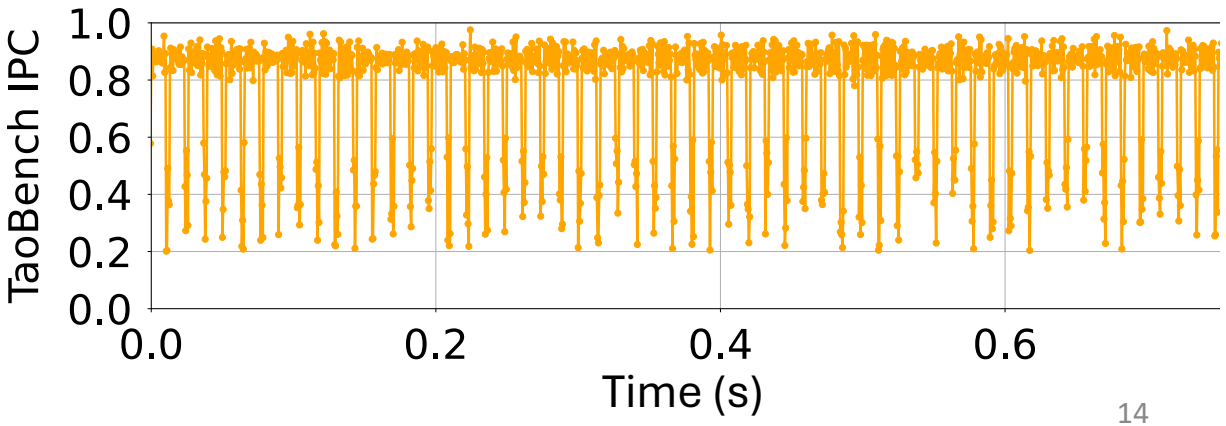
Characterizing Datacenter Workloads



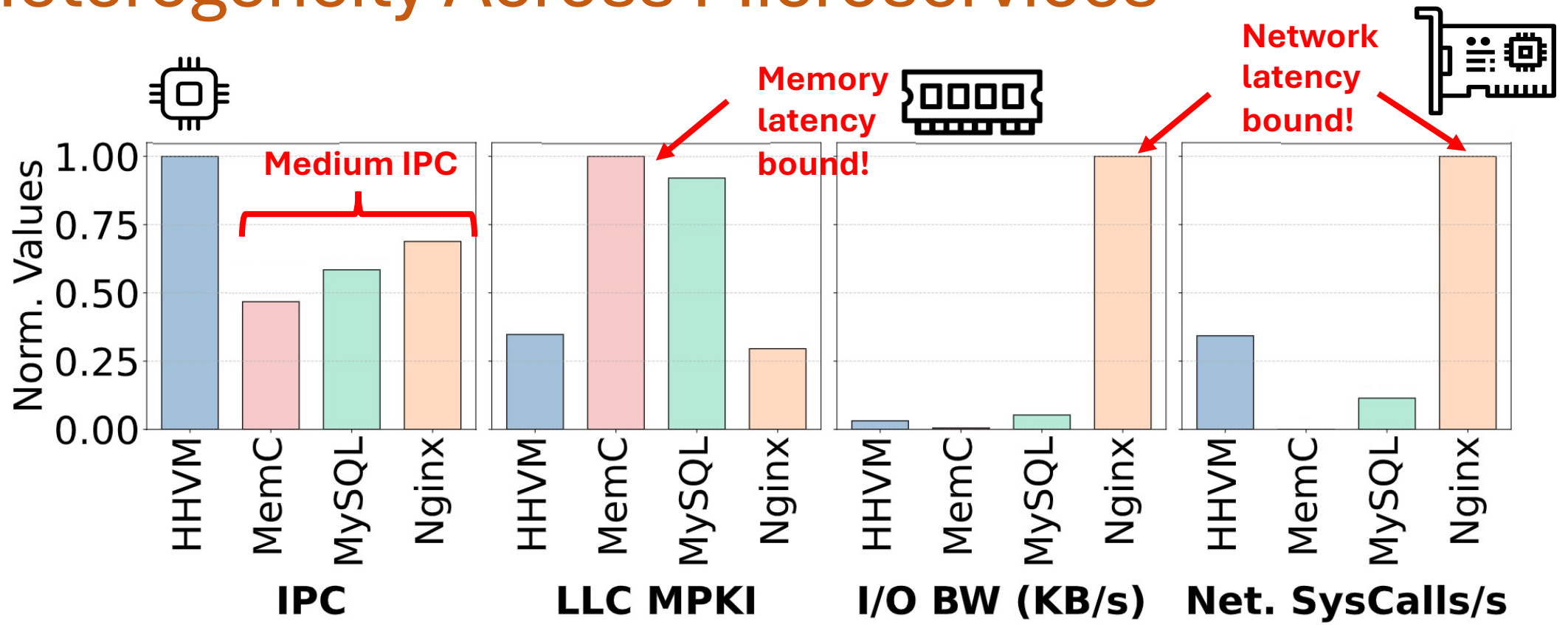


Different IPC-level phases: **What drives this?**





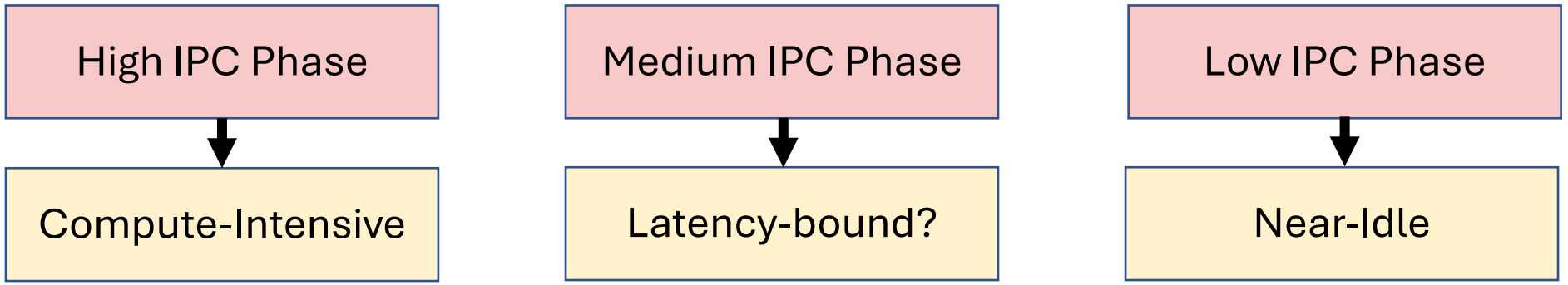
Heterogeneity Across Microservices



Normalized metrics for different microservices that appear in Mediawiki

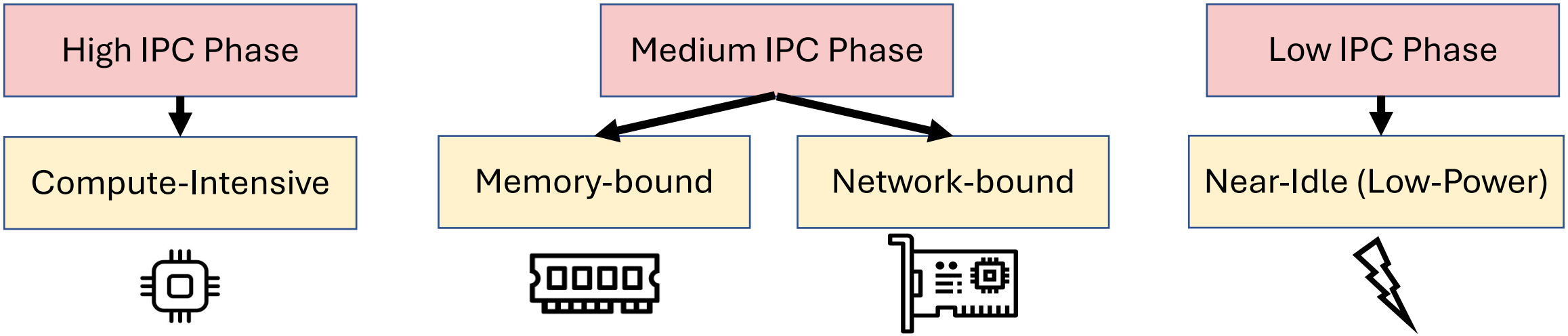
Heterogeneity Across Microservices

Categorizing phase classes:



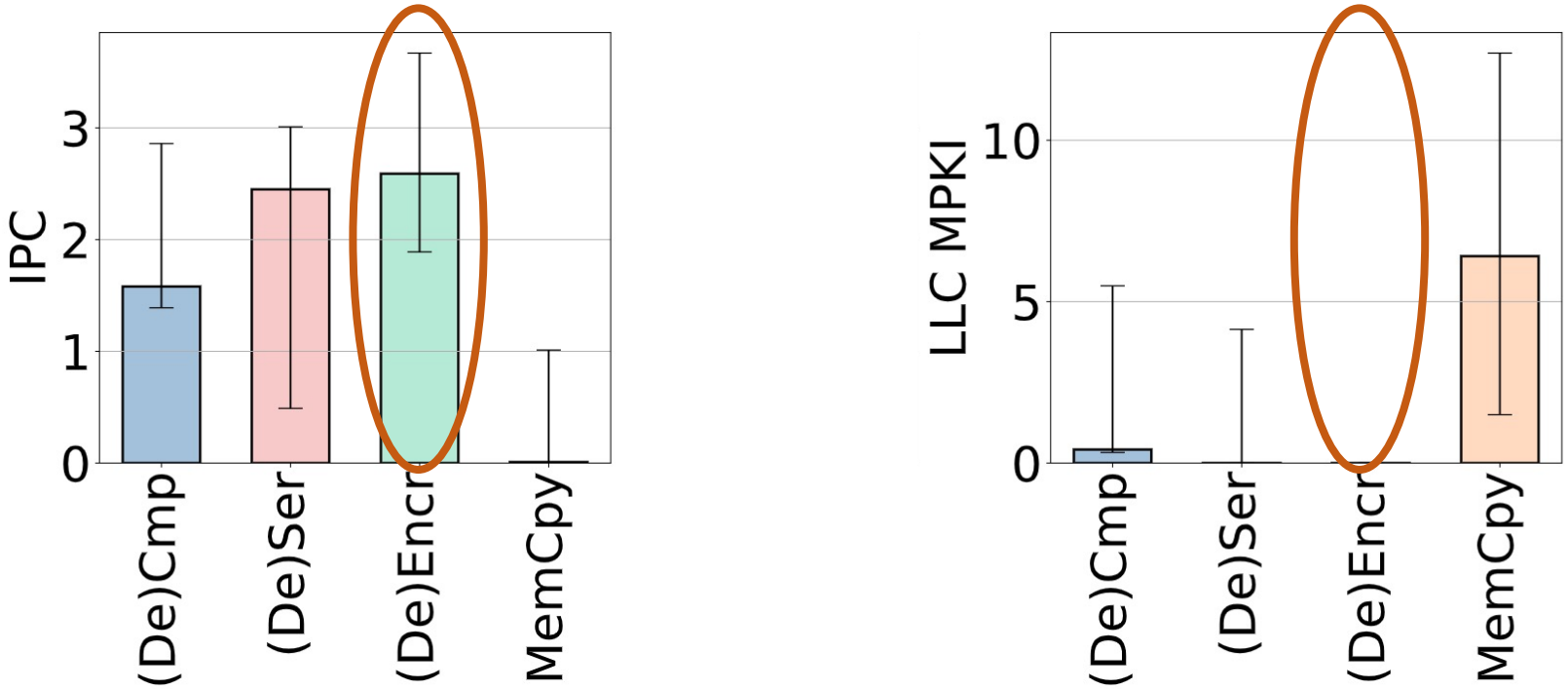
Heterogeneity Across Microservices

Categorizing phase classes:



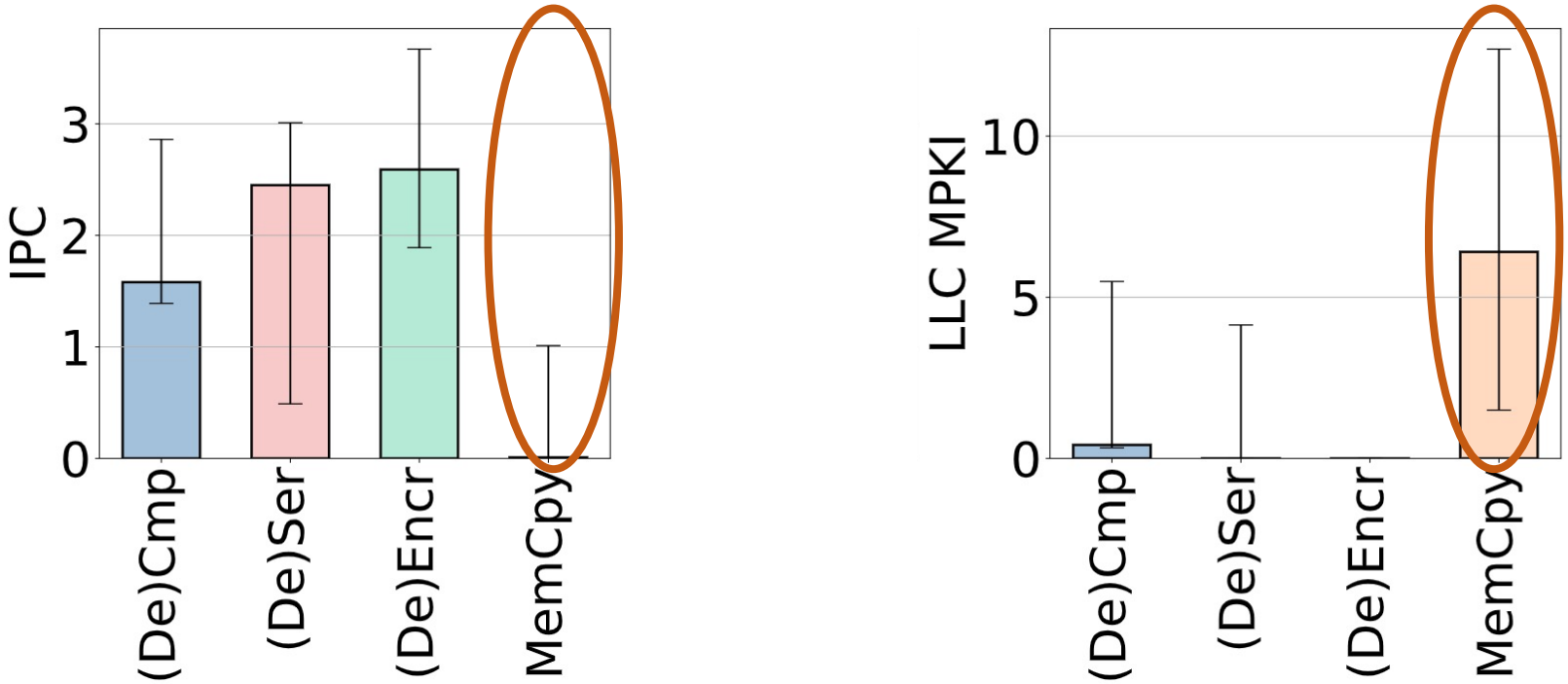
How do other workload components fit into these categories?

Heterogeneity Across Datacenter Tax



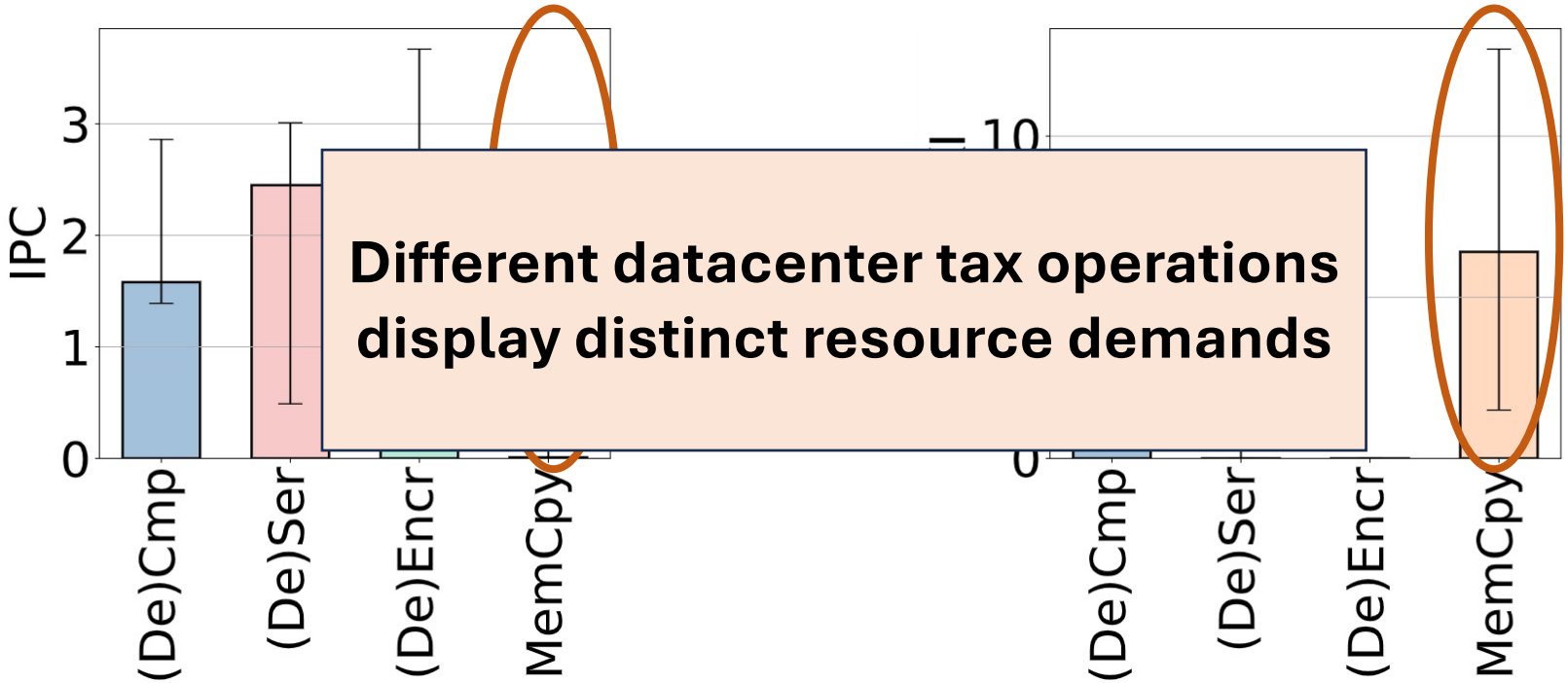
Distribution of metrics across datacenter taxes in *DCPerf* workloads

Heterogeneity Across Datacenter Tax



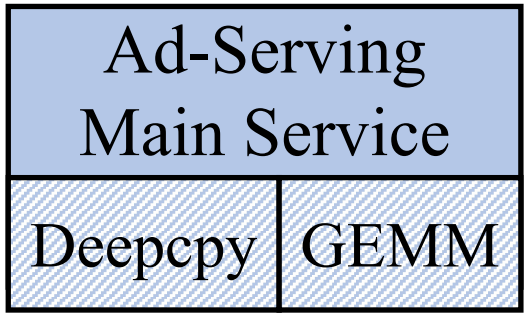
Distribution of metrics across datacenter taxes in *DCPerf* workloads

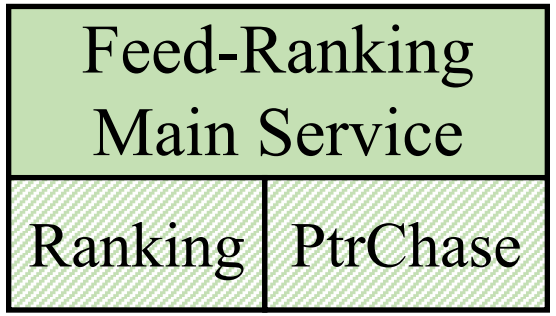
Heterogeneity Across Datacenter Tax



Distribution of metrics across datacenter taxes in *DCPerf* workloads

Heterogeneity in a Single Service

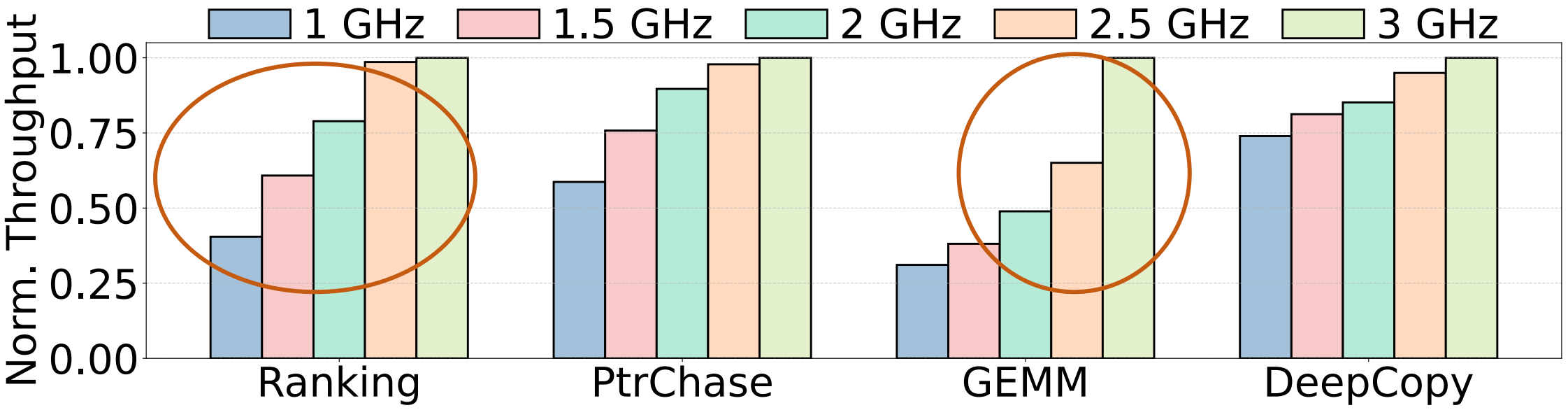




How does performance scale with:

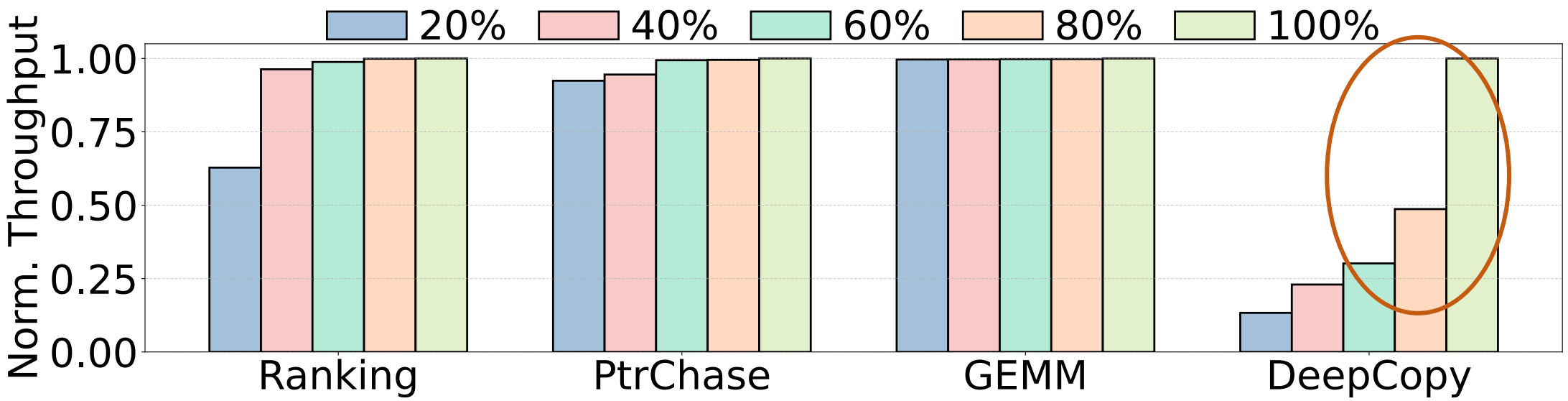
1. Core Frequency
2. Memory Bandwidth
3. L2 Cache Capacity

Heterogeneity in a Single Service



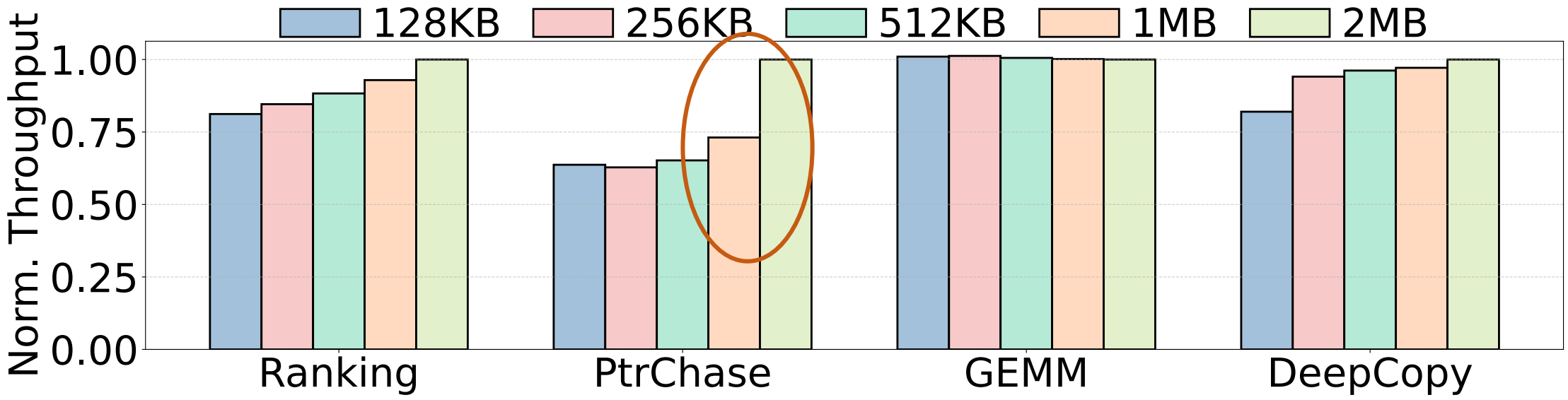
Scaling Core Frequency across Internal Services

Heterogeneity in a Single Service



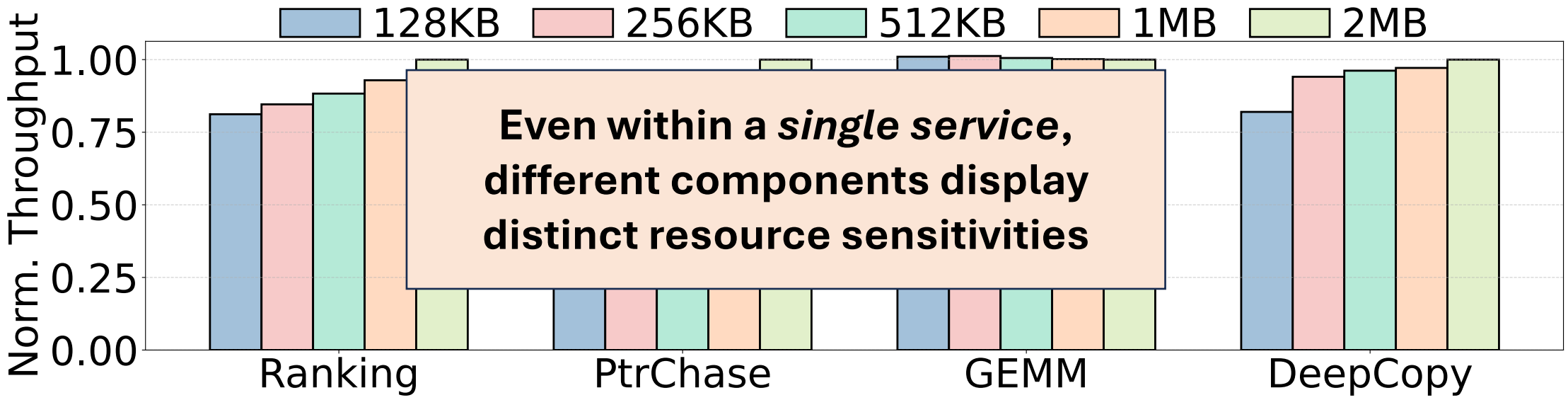
Scaling Memory Bandwidth across Internal Services

Heterogeneity in a Single Service



Scaling L2 Capacity across Internal Services

Heterogeneity in a Single Service

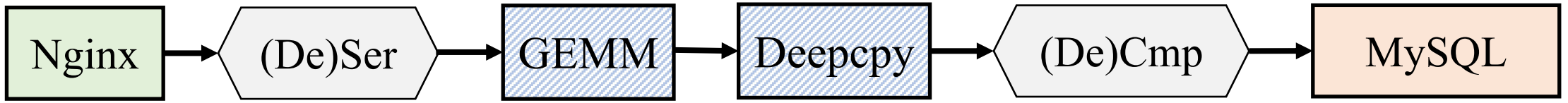


Scaling L2 Capacity across Internal Services

- Show branching of high med low IPC into compute, mem, net, low-power phases with some imaging that is consistent with the illustration from Figure 2

- Then next slide motivates ok why do we need specialized compute

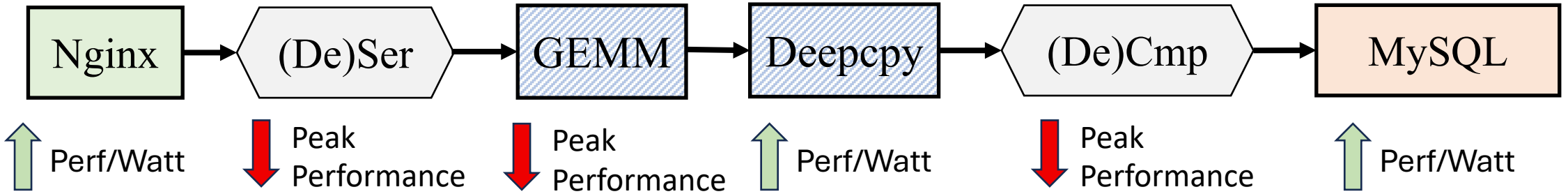
High Heterogeneity Yields Inefficiencies



Case 1: Under-Provisioning

Configure for efficiency (e.g., less cores or lower frequency)

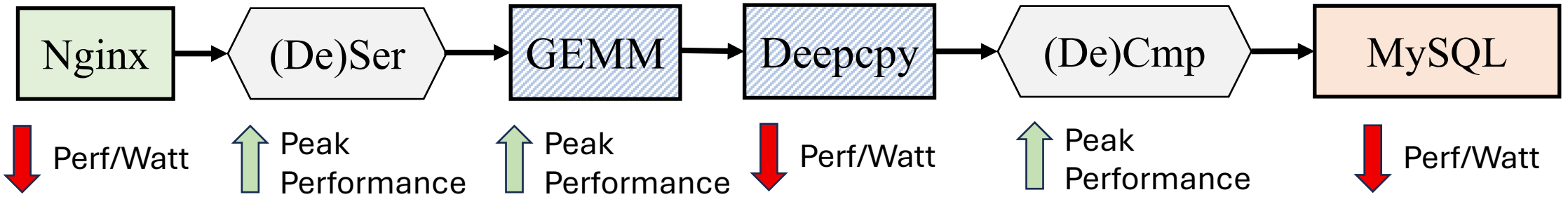
High Heterogeneity Yields Inefficiencies



Case 1: Under-Provisioning

Configure for efficiency (e.g., less cores or lower frequency)

High Heterogeneity Yields Inefficiencies



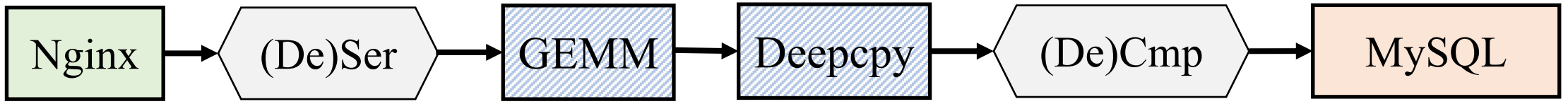
Case 1: Under-Provisioning

Configure for efficiency (e.g., less cores or lower frequency)

Case 2: Over-Provisioning

Match maximum resource requirement (e.g., compute)

High Heterogeneity Yields Inefficiencies



Insight: Heterogeneity of microservice applications require *specialized servers* for maximum efficiency

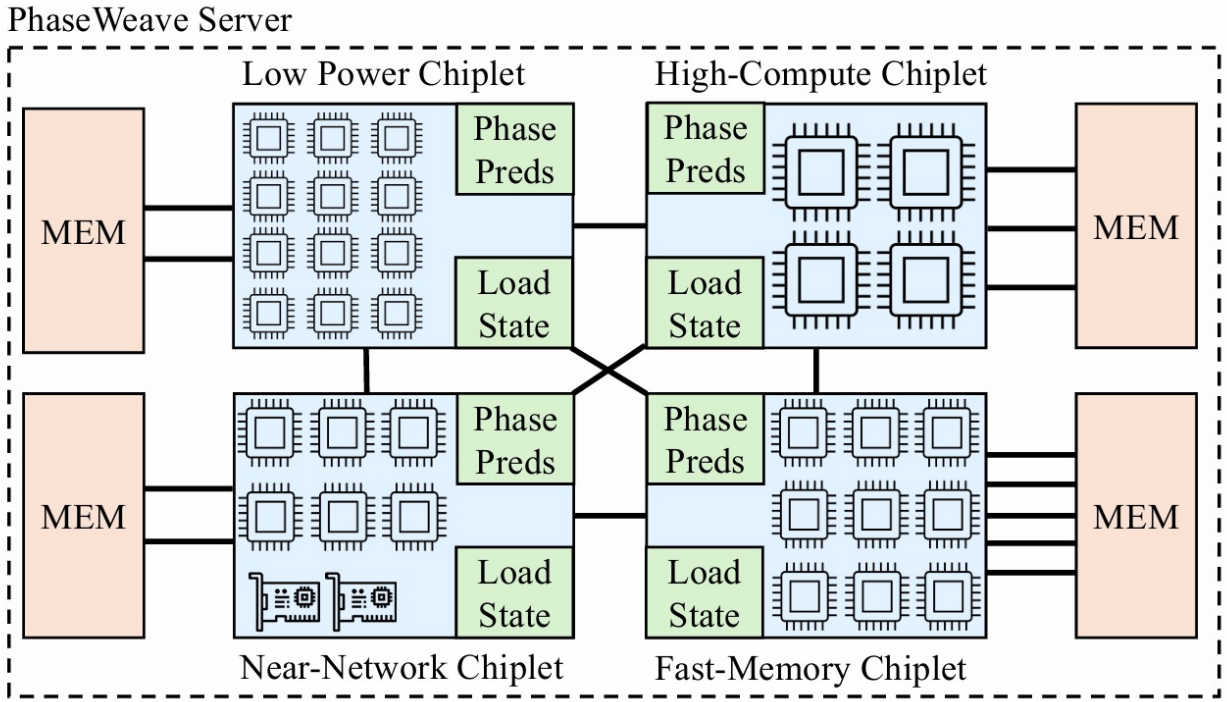
Case 1: Und

Case 2: Over-Provisioning

Match maximum resource requirement (e.g., compute)

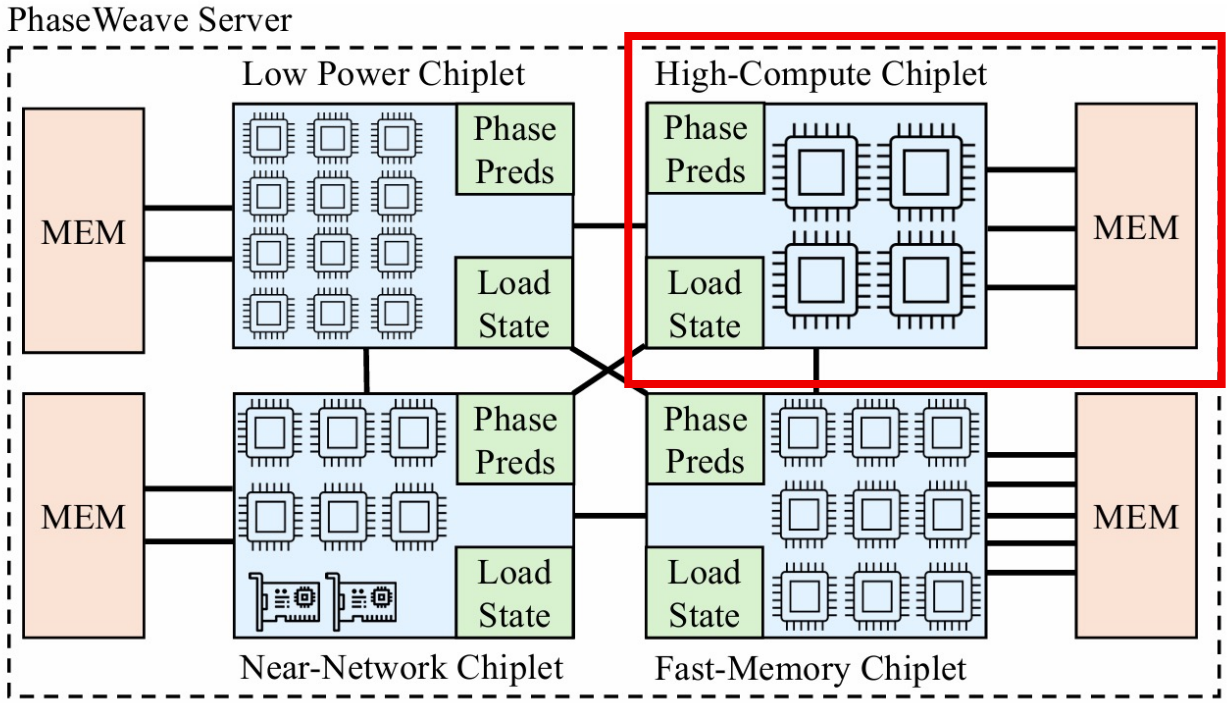
PhaseWeave: Heterogeneous Server Design

- Chiplets to introduce specialization



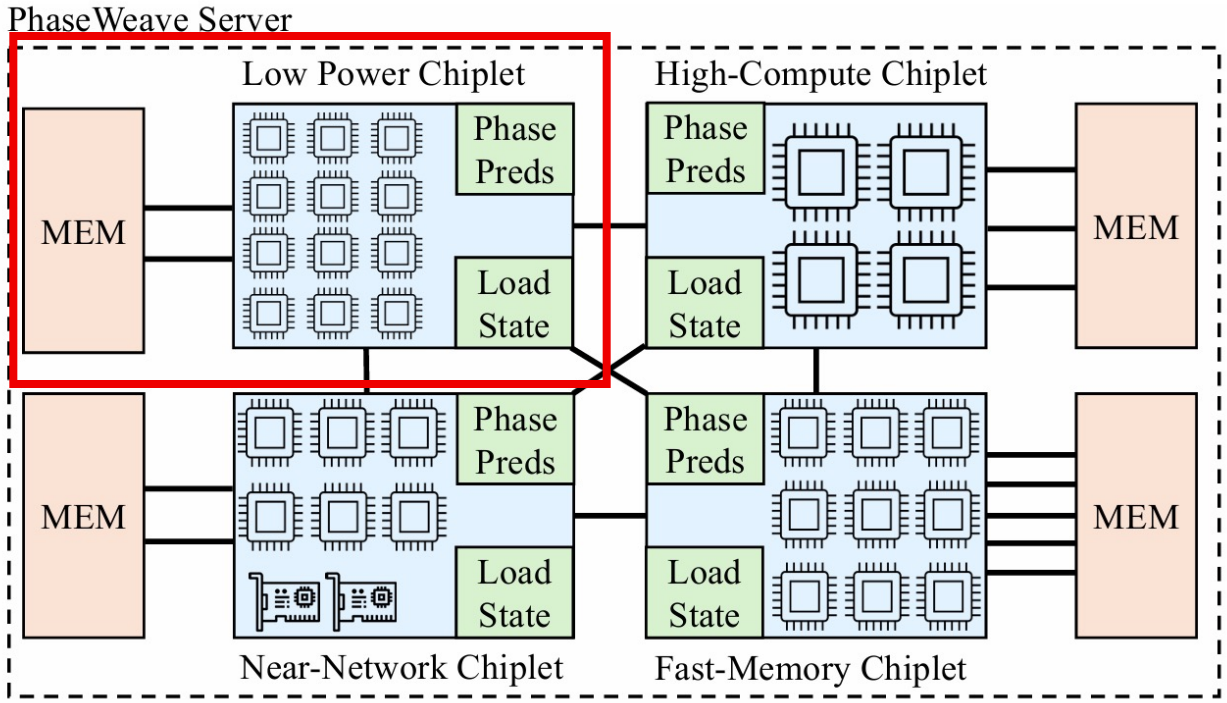
PhaseWeave: Heterogeneous Server Design

- Chiplets to introduce specialization
- High-compute chiplet
  - Large performance cores
  - Moderate LLC capacity & off chip memory bandwidth



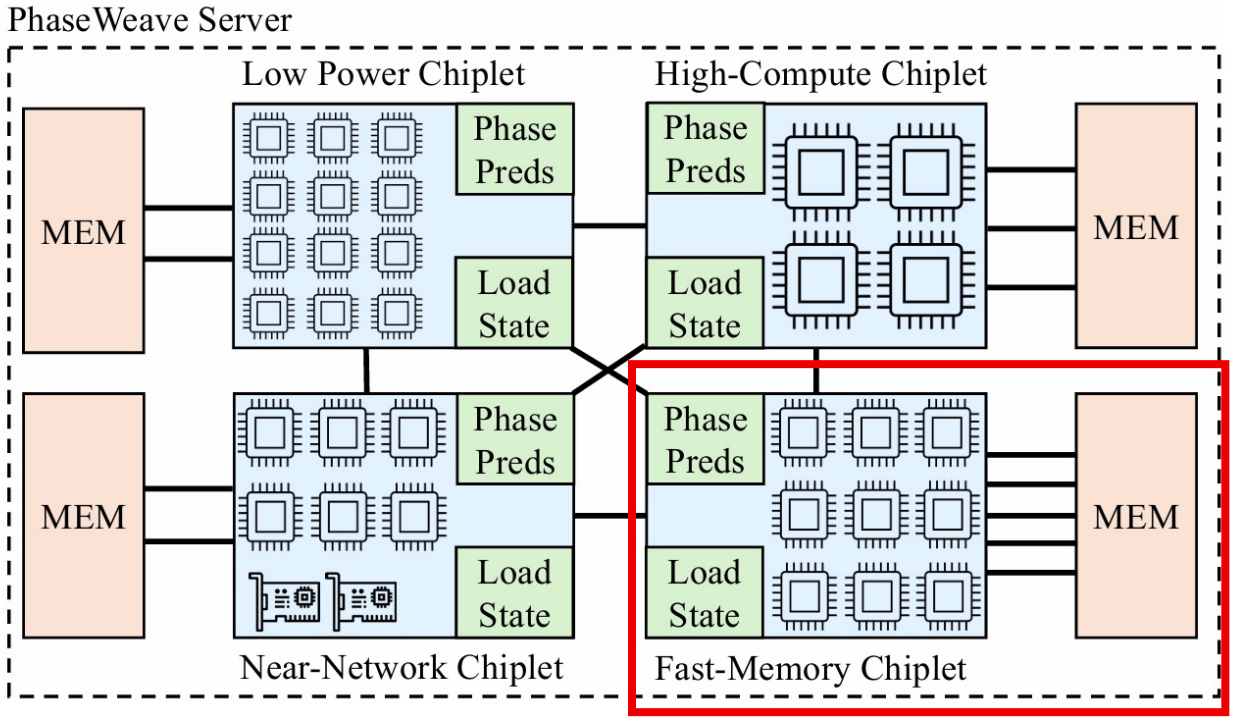
PhaseWeave: Heterogeneous Server Design

- Chiplets to introduce specialization
- High-compute chiplet
- Low-power chiplet
  - Small, efficiency cores
  - Smaller private caches
  - Slower DRAM channels



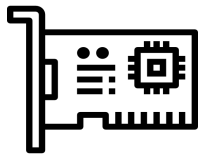
PhaseWeave: Heterogeneous Server Design

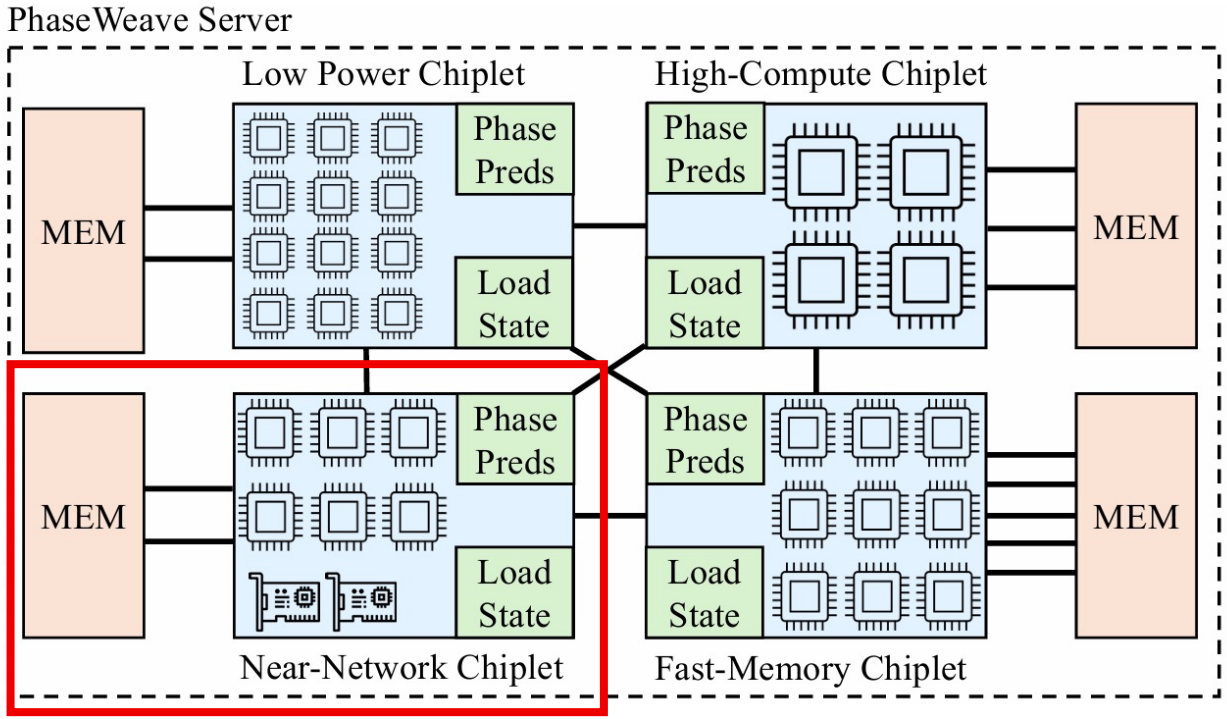
- Chiplets to introduce specialization
- High-compute chiplet
- Low-power chiplet
- Fast-memory chiplet
  - Medium-sized cores
  - Max cache allocation
  - More high bandwidth memory channels



PhaseWeave: Heterogeneous Server Design

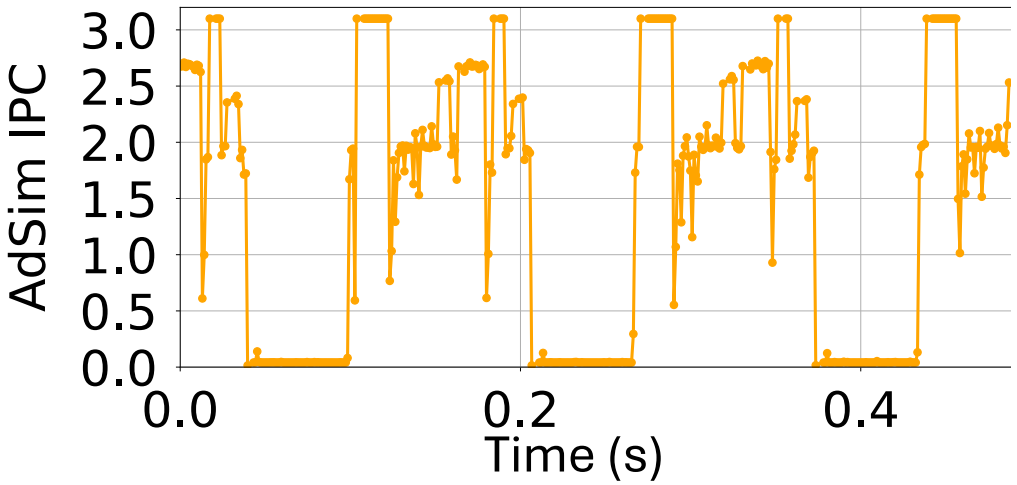
- Chiplets to introduce specialization
- High-compute chiplet
- Low-power chiplet
- Fast-memory chiplet
- Near-network chiplet
  - Medium-sized cores
  - Smaller cache & Moderate BW
  - Integrated NICs



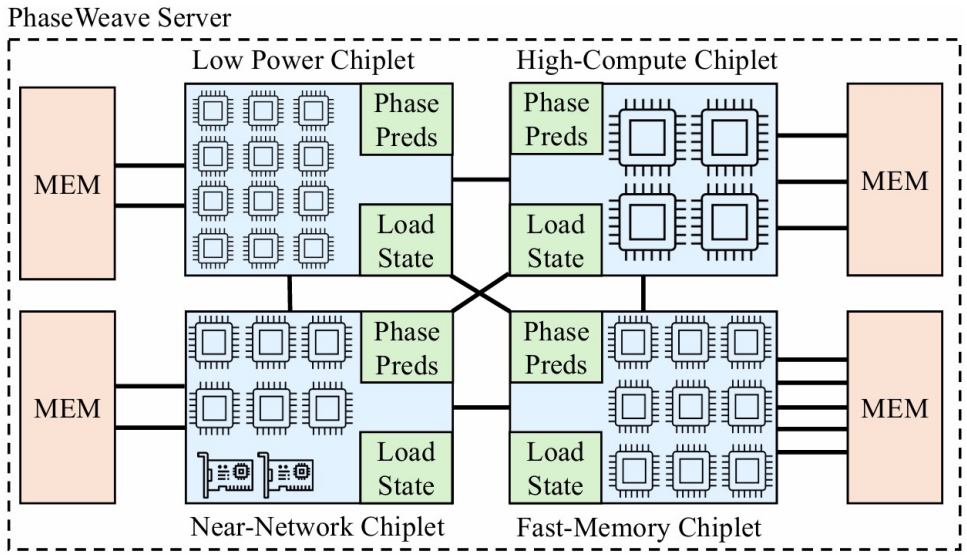


How to Match Workload and Hardware Heterogeneity?

Heterogeneous Workload Phases

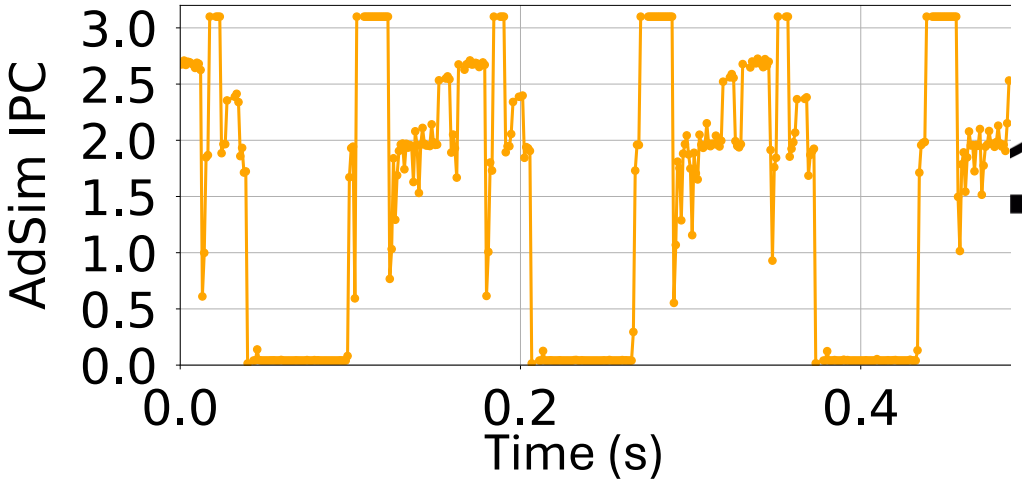


Heterogeneous Server Architecture

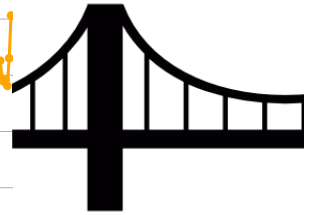


How to Match Workload and Hardware Heterogeneity?

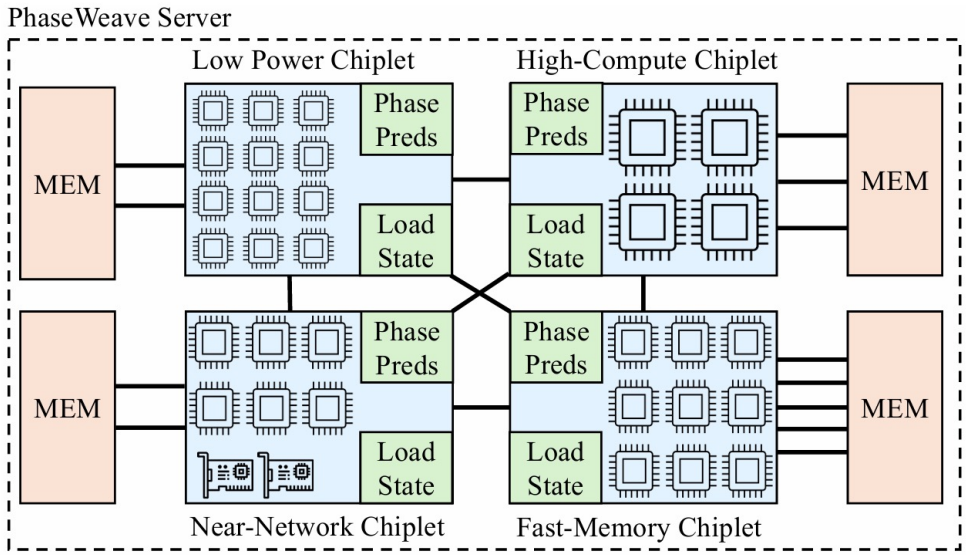
Heterogeneous Workload Phases



Phase Predictor



Heterogeneous Server Architecture



Phase Predictor: Leveraging Heterogeneous HW

- To leverage this specialized hardware, we must understand phase behavior

Phase Predictor: Leveraging Heterogeneous HW

- **Goal**: Classify program phases by their dominant resource needs to guide hardware selection.

Feature Vector

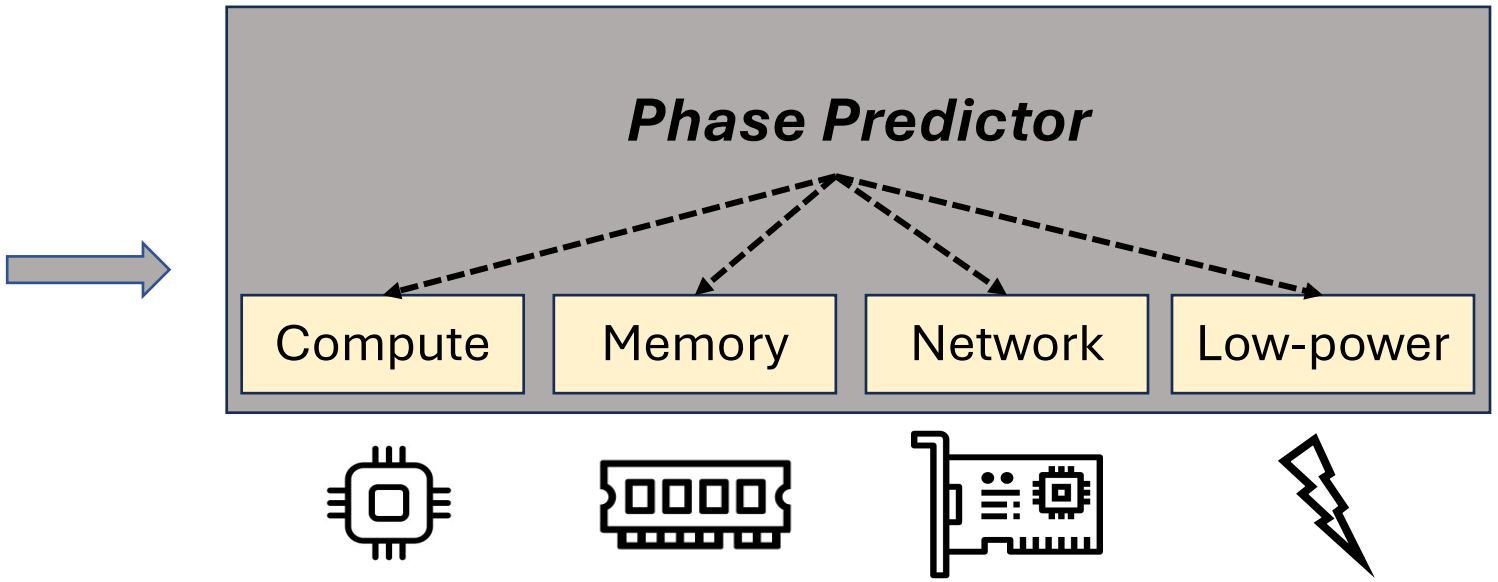
- IPC
- Cache MPKI
- TLB MPKI
- Branch MPKI
- I/O Bandwidth
- Frequency of Networking Syscalls
- …

Phase Predictor: Leveraging Heterogeneous HW

- **Goal**: Classify program phases by their dominant resource needs to guide hardware selection.

Feature Vector

- IPC
- Cache MPKI
- TLB MPKI
- Branch MPKI
- I/O Bandwidth
- Frequency of Networking Syscalls
- …

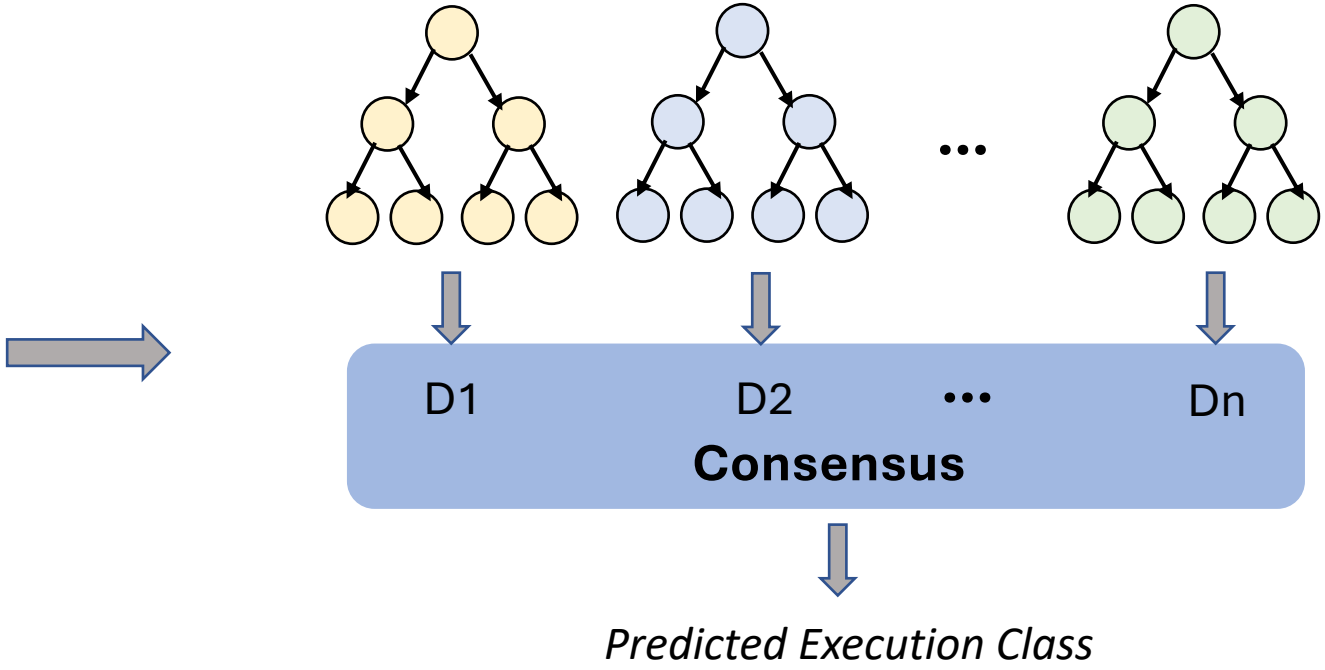


Phase Predictor: Leveraging Heterogeneous HW

- **Goal**: Classify program phases by their dominant resource needs to guide hardware selection.

Feature Vector

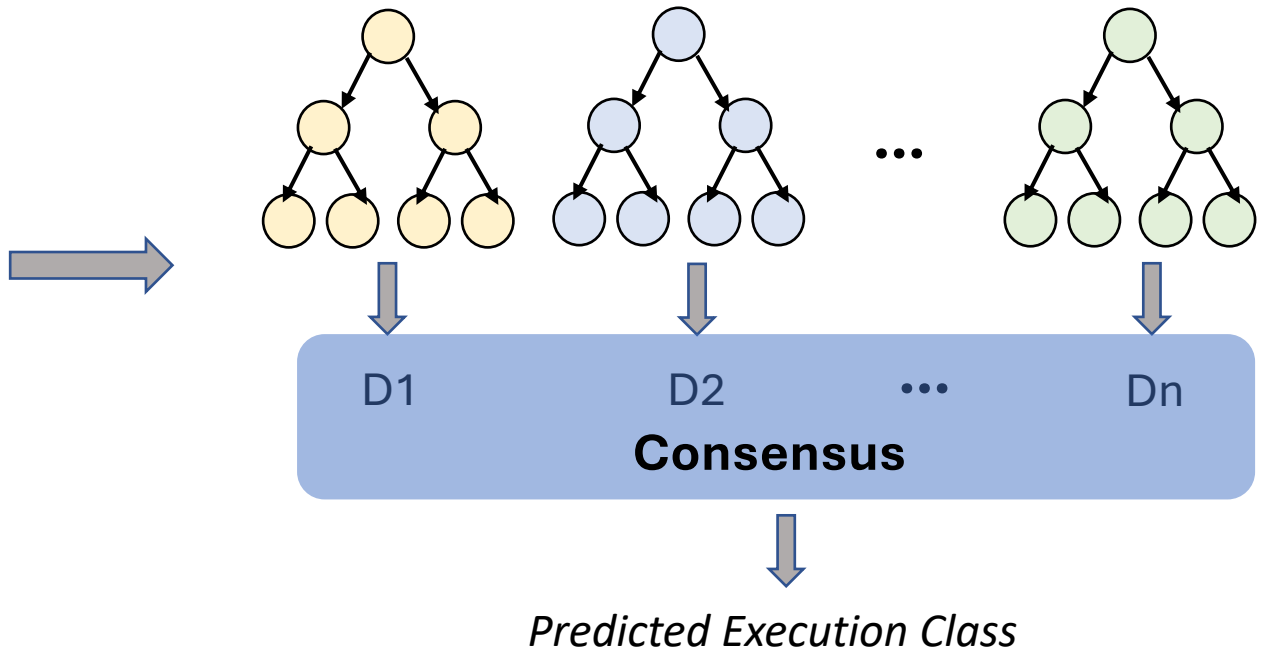
- IPC
- Cache MPKI
- TLB MPKI
- Branch MPKI
- I/O Bandwidth
- Frequency of Networking Syscalls
- …



Phase Predictor: Leveraging Heterogeneous HW

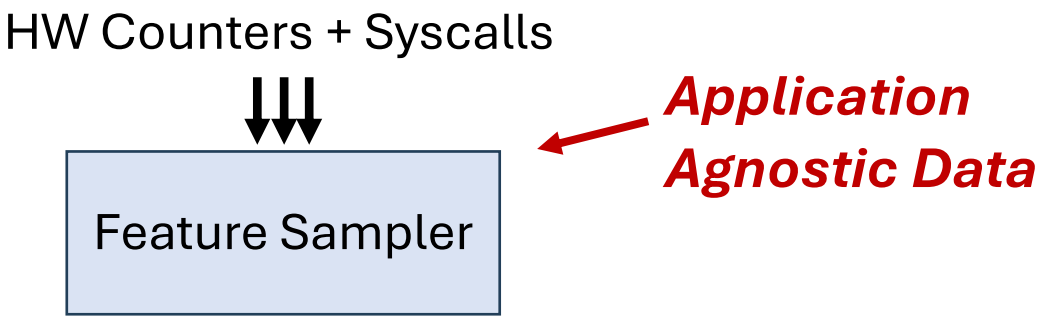
Feature Vector

- IPC
- Cache MPKI
- TLB MPKI
- Branch MPKI
- I/O Bandwidth
- Frequency of Networking Syscalls
- …



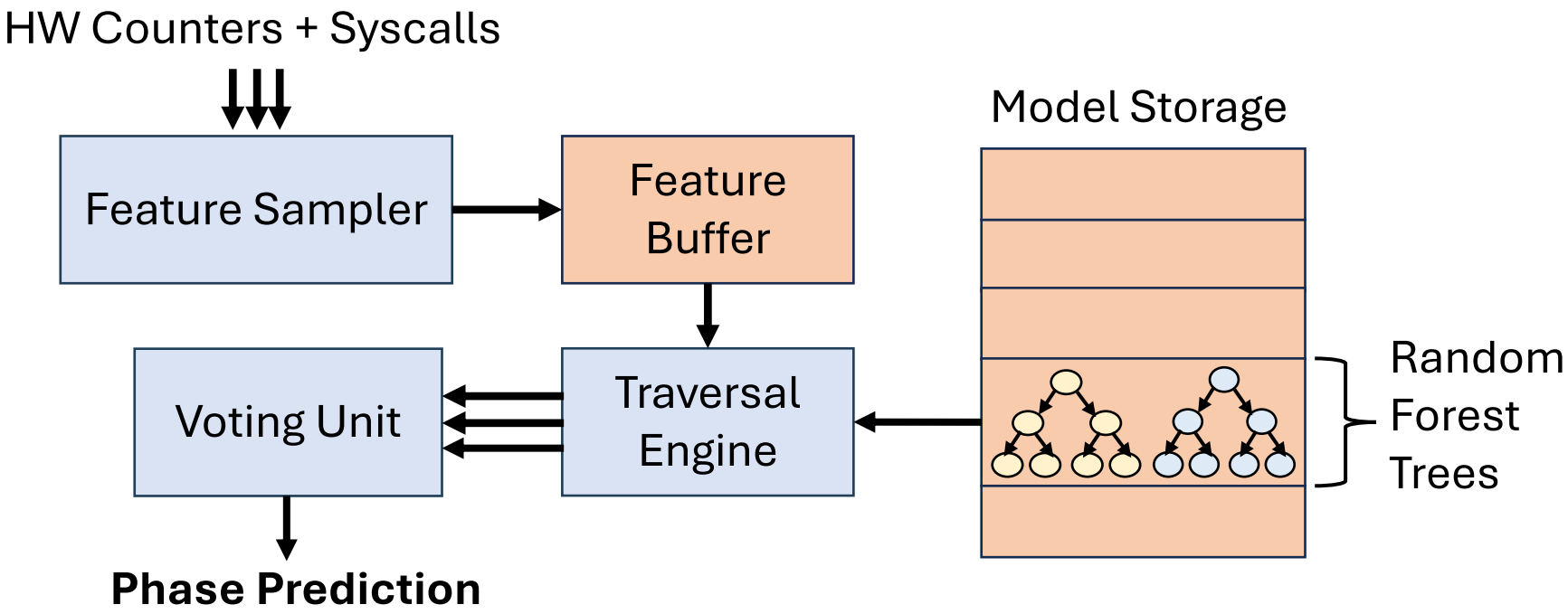
Phase Predictor: Hardware Support

- We implement in hardware to remove prediction from an application's critical path



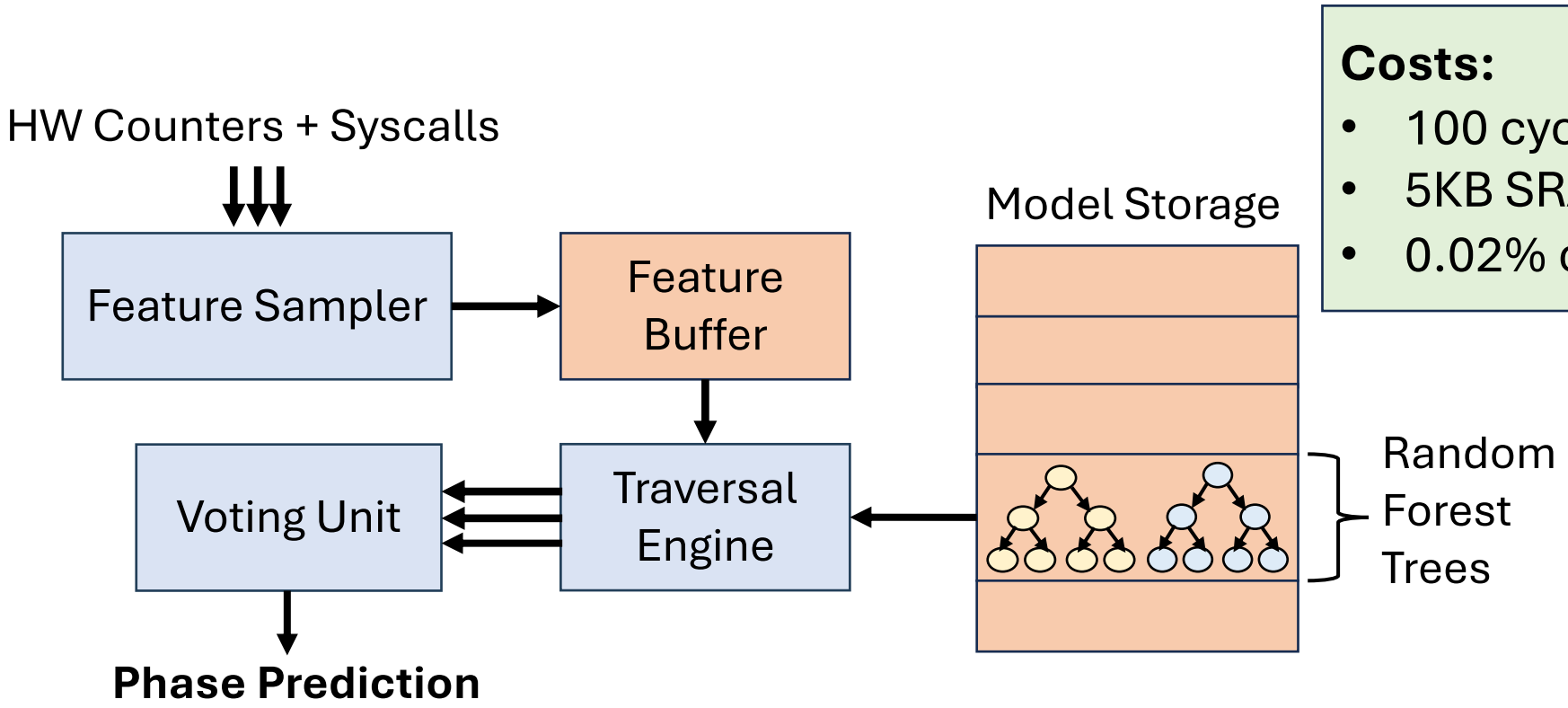
Phase Predictor: Hardware Support

- We implement in hardware to remove prediction from an application's critical path



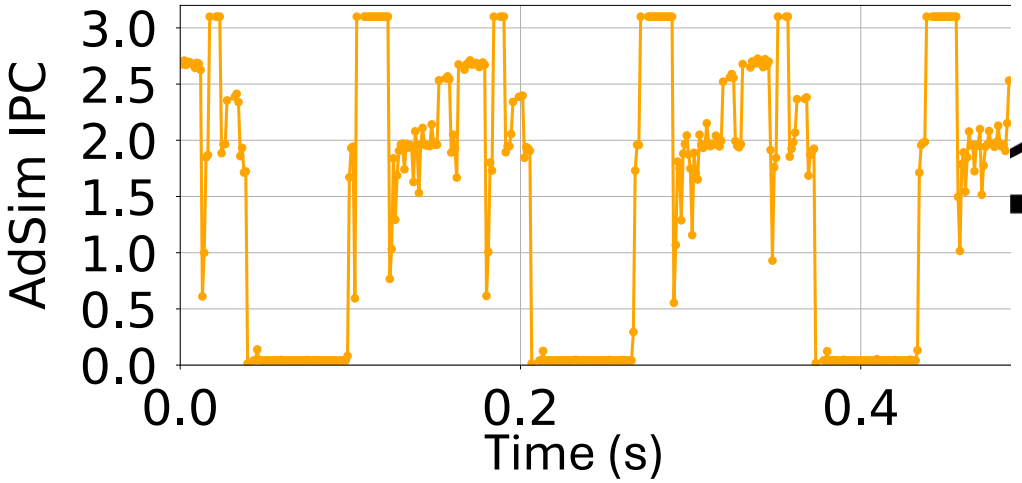
Phase Predictor: Hardware Support

- We implement in hardware to remove prediction from an application's critical path

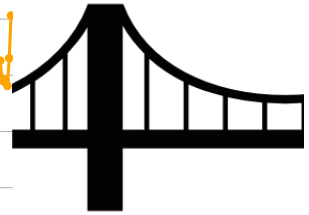


How to Match Workload and Hardware Heterogeneity?

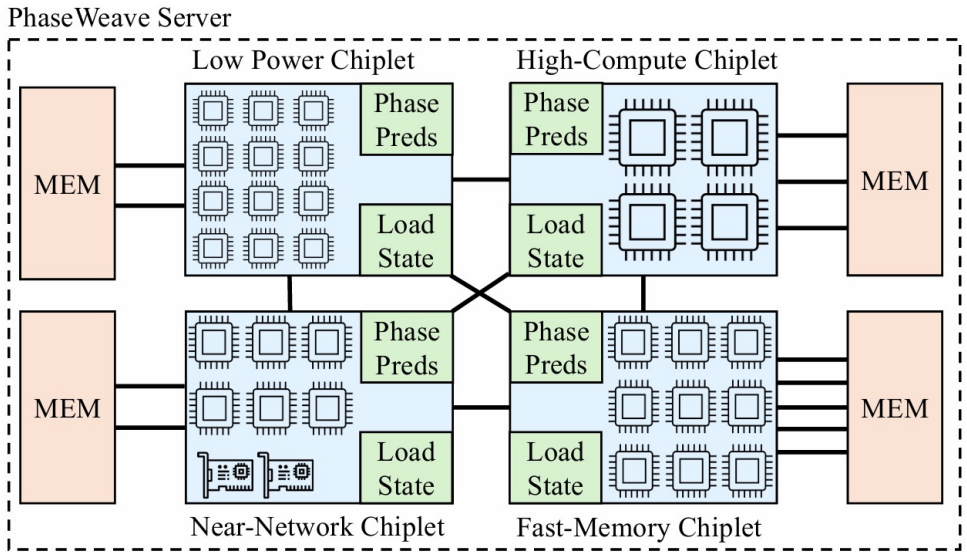
Heterogeneous Workload Phases



Phase Predictor

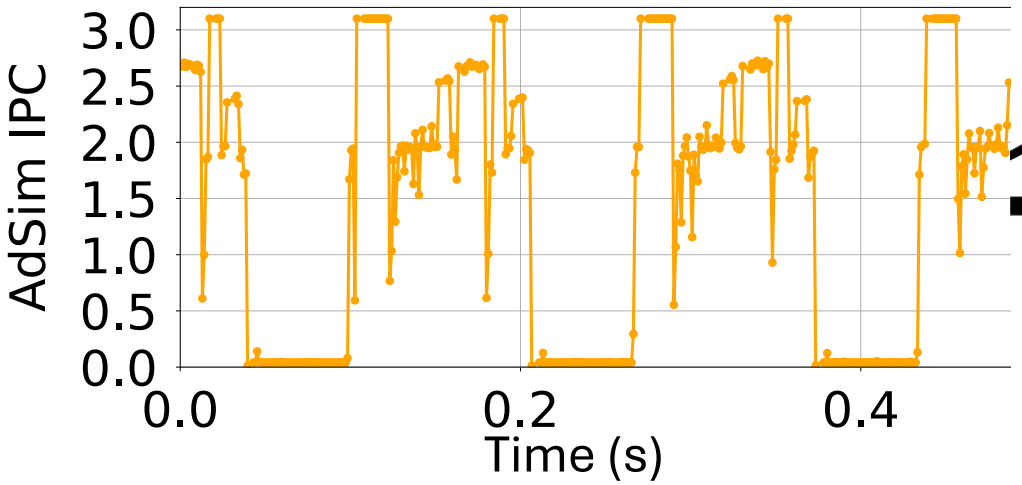


Heterogeneous Server Architecture

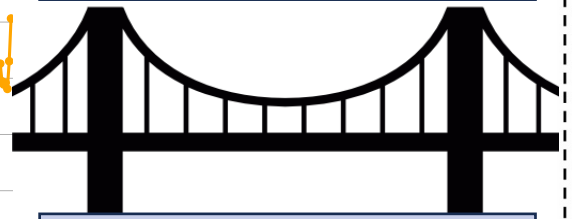


How to Match Workload and Hardware Heterogeneity?

Heterogeneous Workload Phases

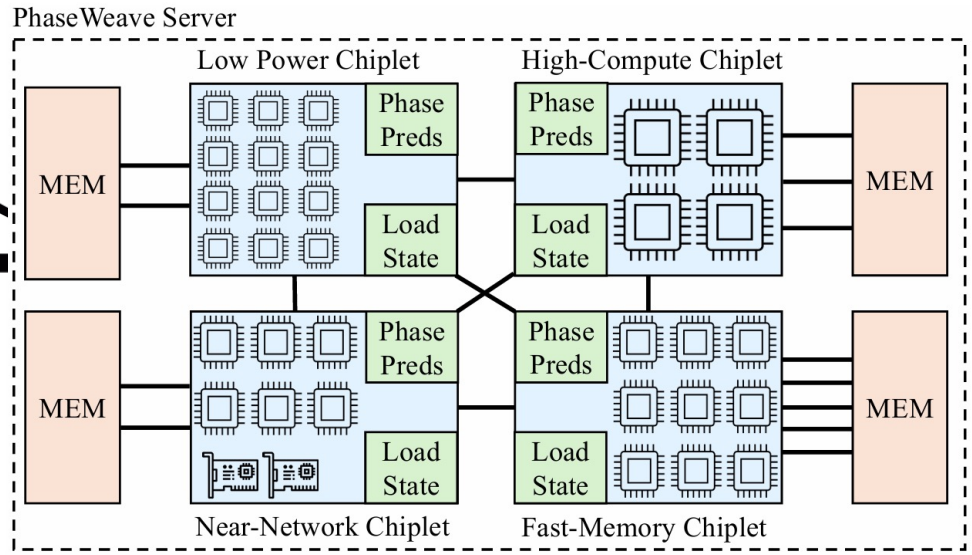


Phase Predictor



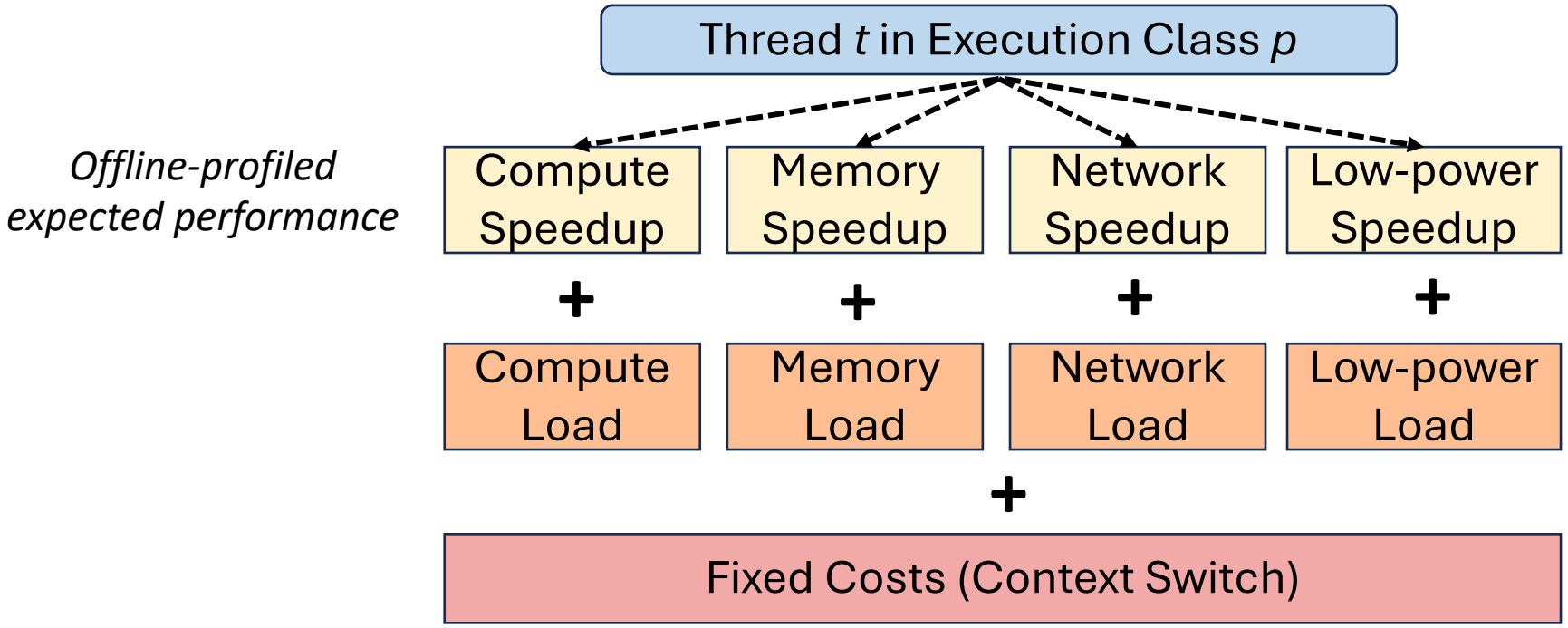
Migration Policy

Heterogeneous Server Architecture



Benefit-Aware Migration

- Migrating to optimal chiplet can be sub-optimal due to queuing



Evaluation Methodology

- Cycle-accurate full-system simulations: SST + QEMU
- DCPerf benchmark suite as evaluation workload with varying load levels (low, medium, high)
- Systems Evaluated:
    1. **Baseline**: Homogeneous 28-core Emerald Rapids Server

Evaluation Methodology

- Cycle-accurate full-system simulations: SST + QEMU
- DCPerf benchmark suite as evaluation workload with varying load levels (low, medium, high)
- Systems Evaluated:
  1. **Baseline**: Homogeneous 28-core Emerald Rapids Server
  2. **big.LITTLE**: ARM big.LITTLE configuration, iso-area to baseline
  3. **big.LITTLE-opt**: ARM-style with PhaseWeave compute & low-power cores

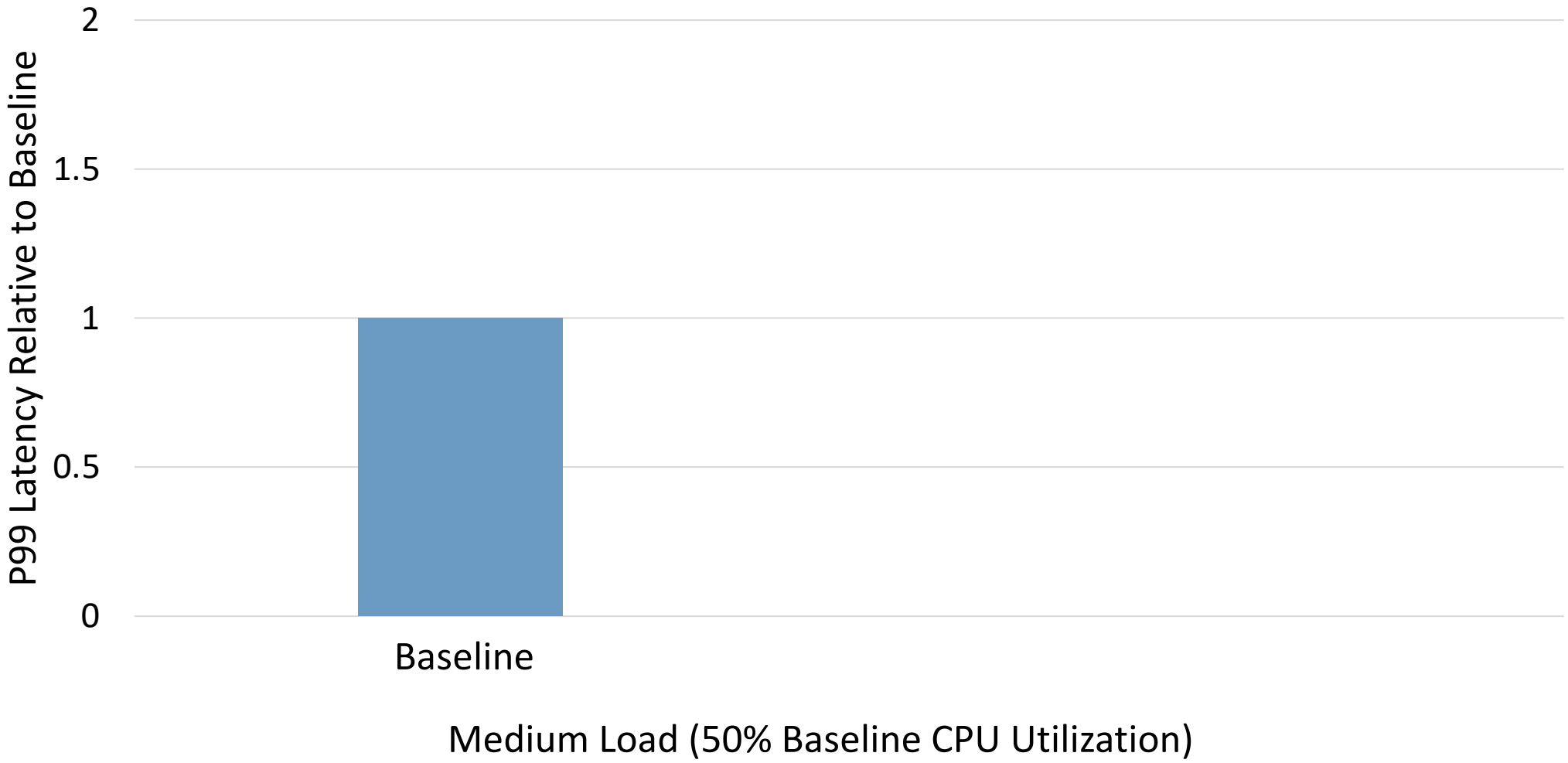
Coarse-Grain Phase Definitions

Hand optimized phases & PhaseWeave Migration Policy

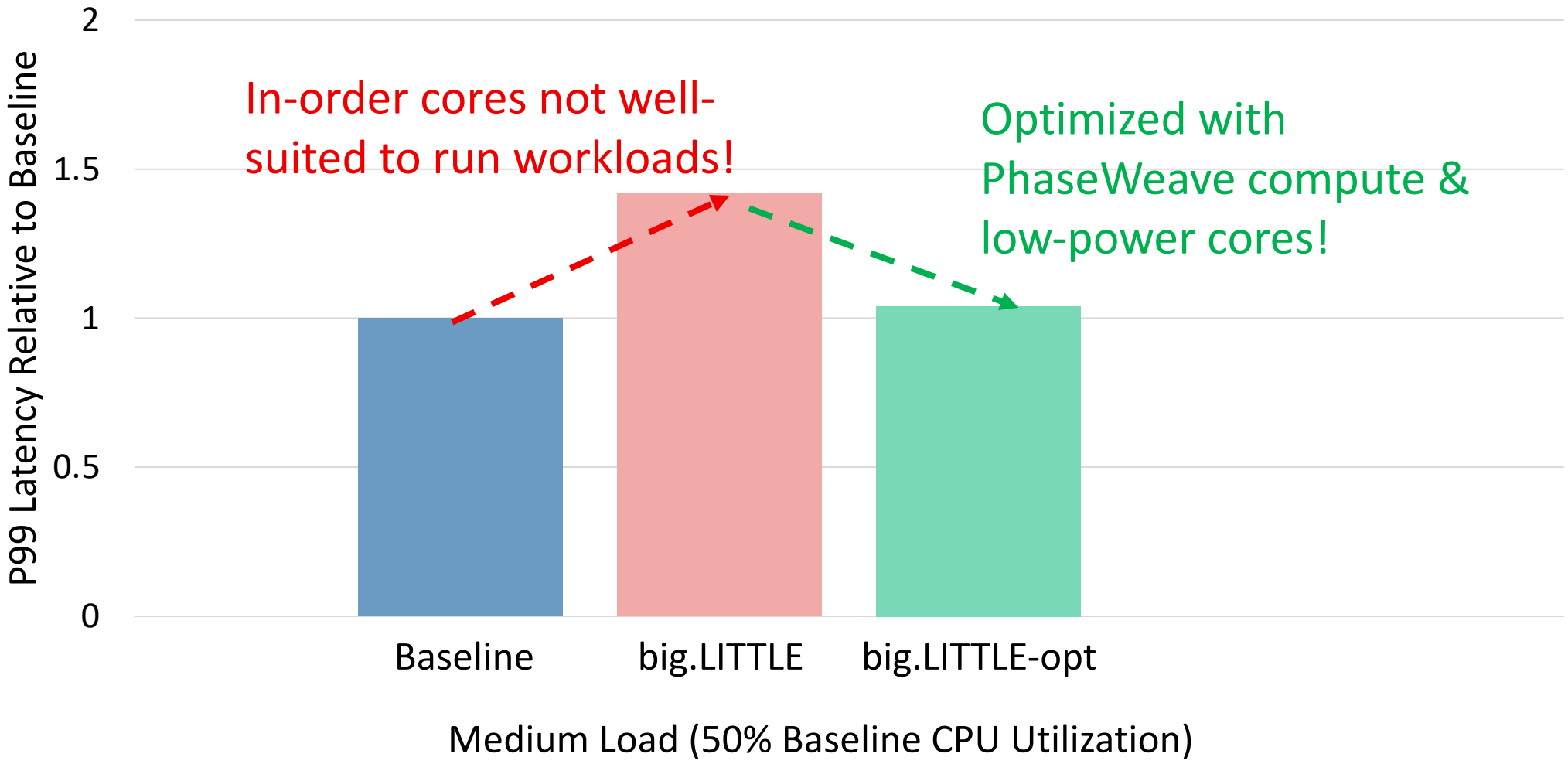
Evaluation Methodology

- Cycle-accurate full-system simulations: SST + QEMU
- DCPerf benchmark suite as evaluation workload with varying load levels (low, medium, high)
- Systems Evaluated:
  1. **Baseline**: Homogeneous 28-core Emerald Rapids Server
  2. **big.LITTLE**: ARM big.LITTLE configuration, iso-area to baseline
  3. **big.LITTLE-opt**: ARM-style with PhaseWeave compute & low-power cores
  4. **PhaseWeave:** Our proposal, iso-area to baseline

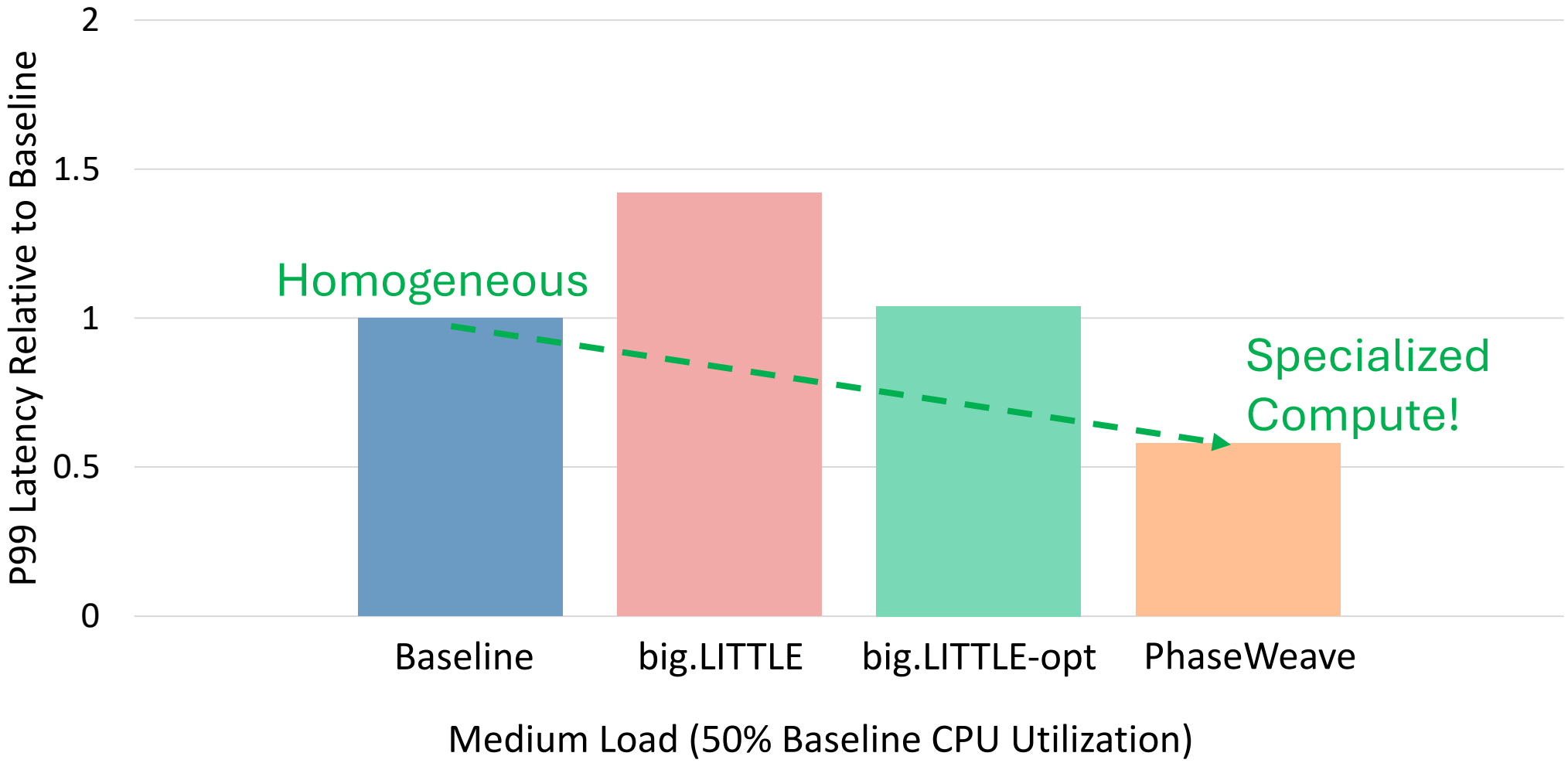
PhaseWeave Significantly Reduces Tail Latency



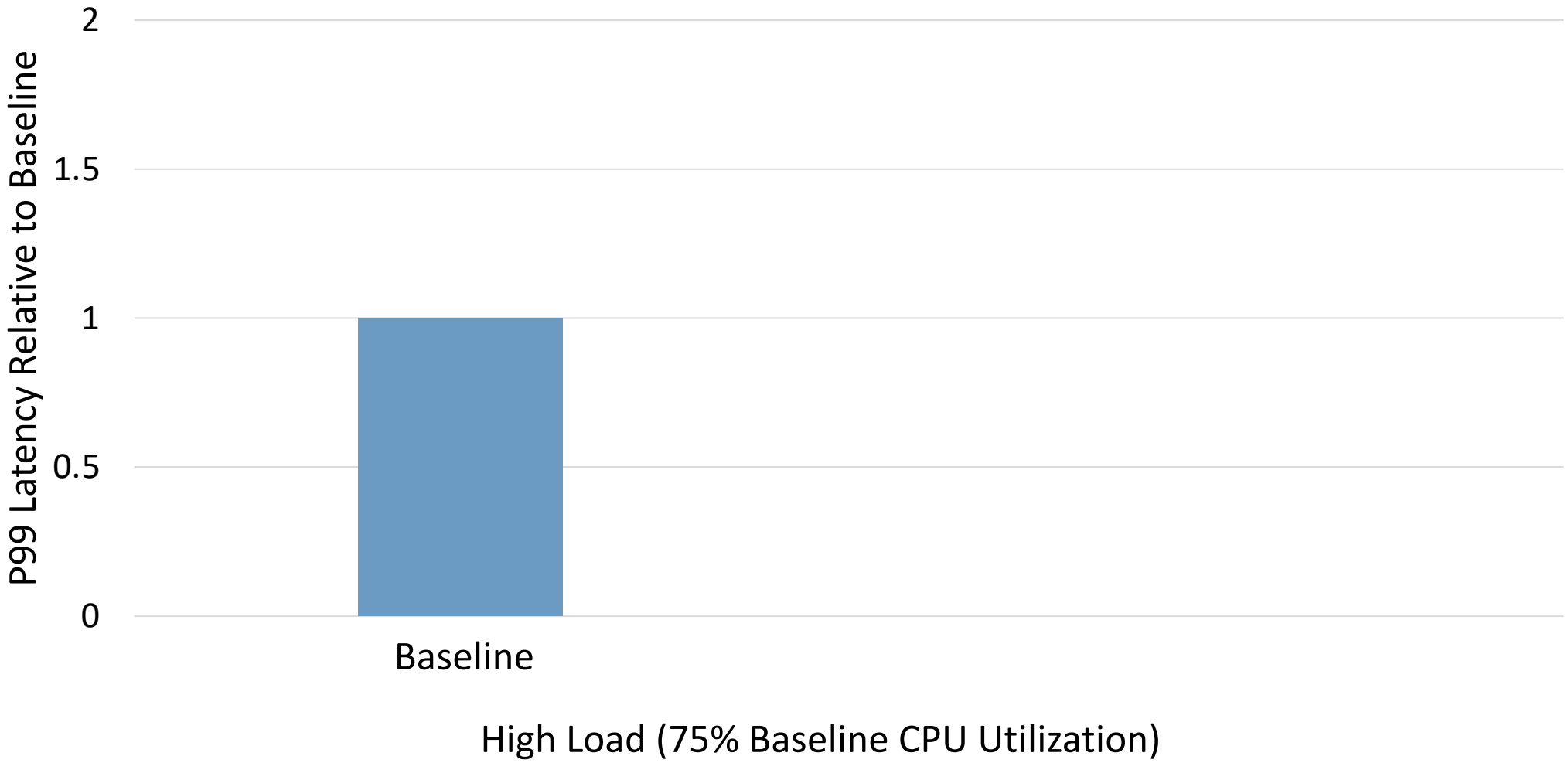
PhaseWeave Significantly Reduces Tail Latency



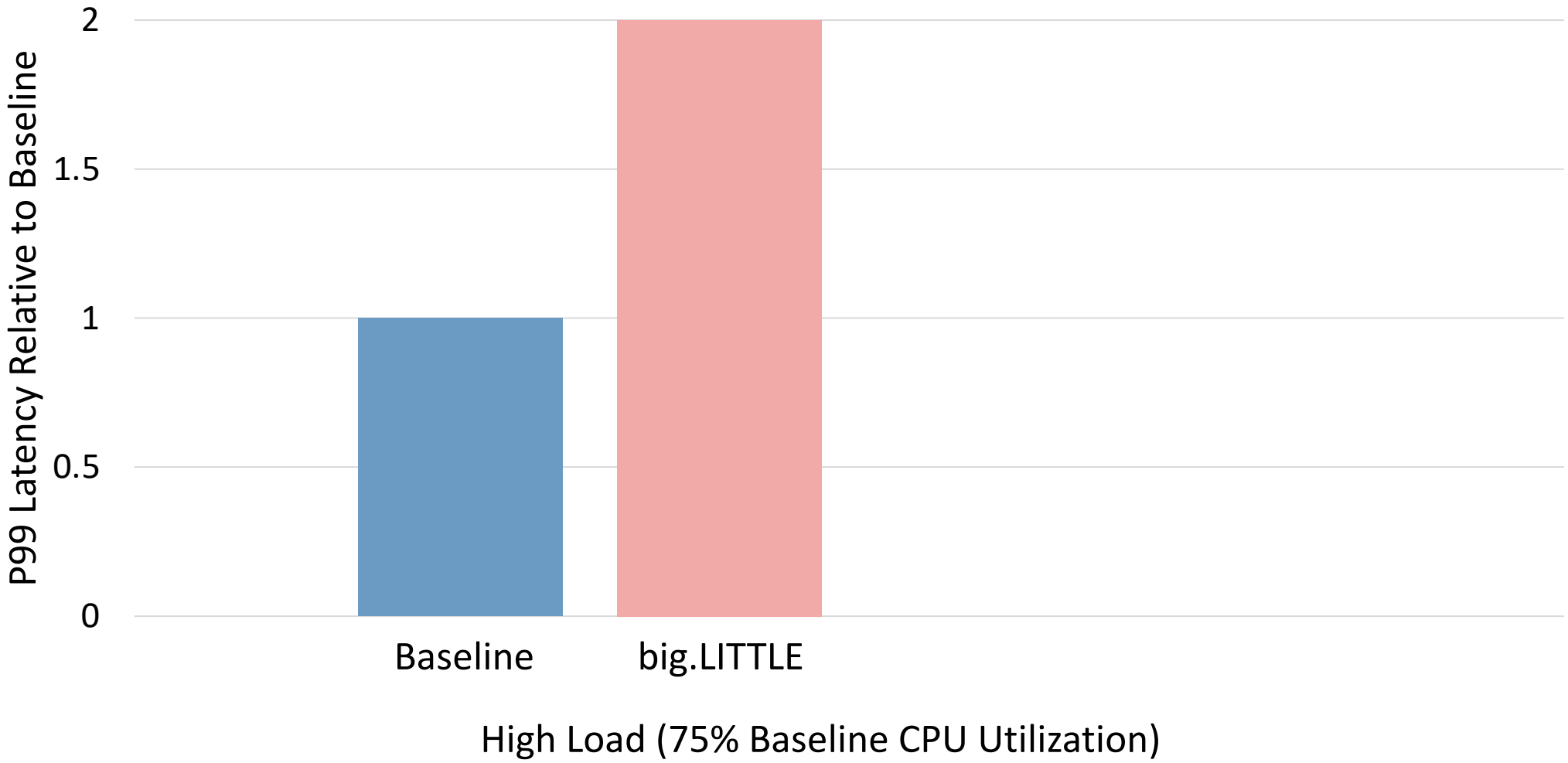
PhaseWeave Significantly Reduces Tail Latency



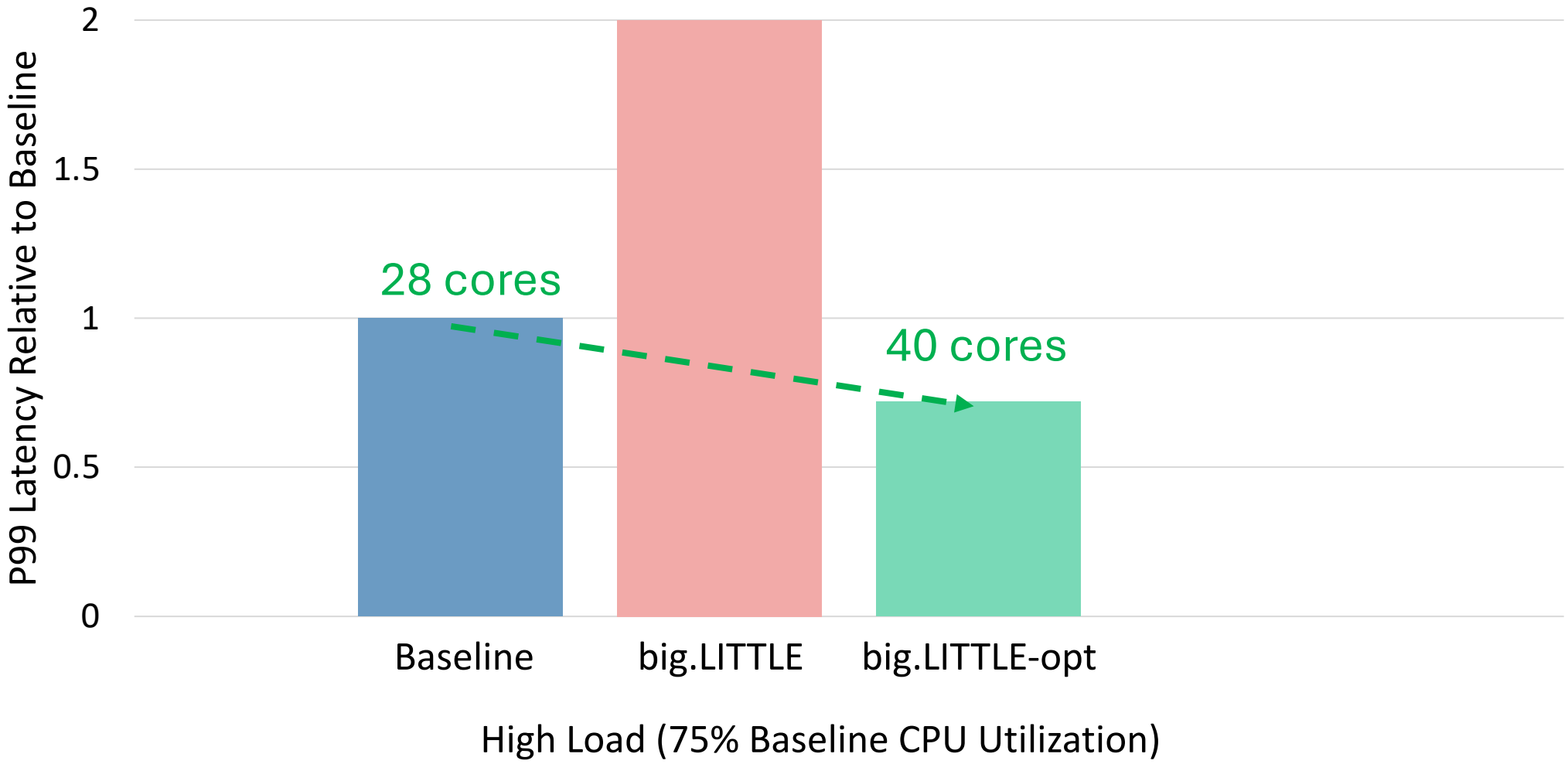
PhaseWeave Significantly Reduces Tail Latency



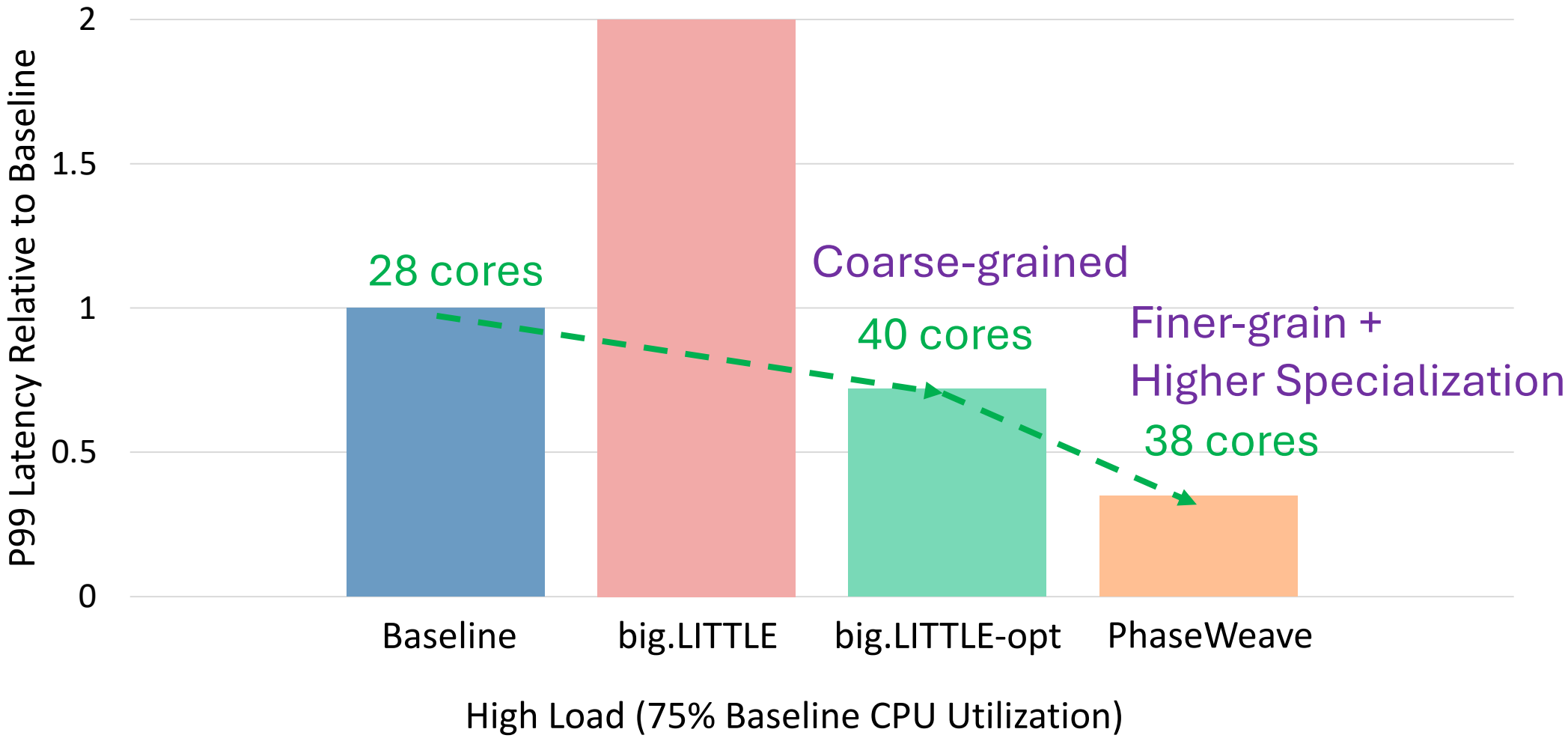
PhaseWeave Significantly Reduces Tail Latency



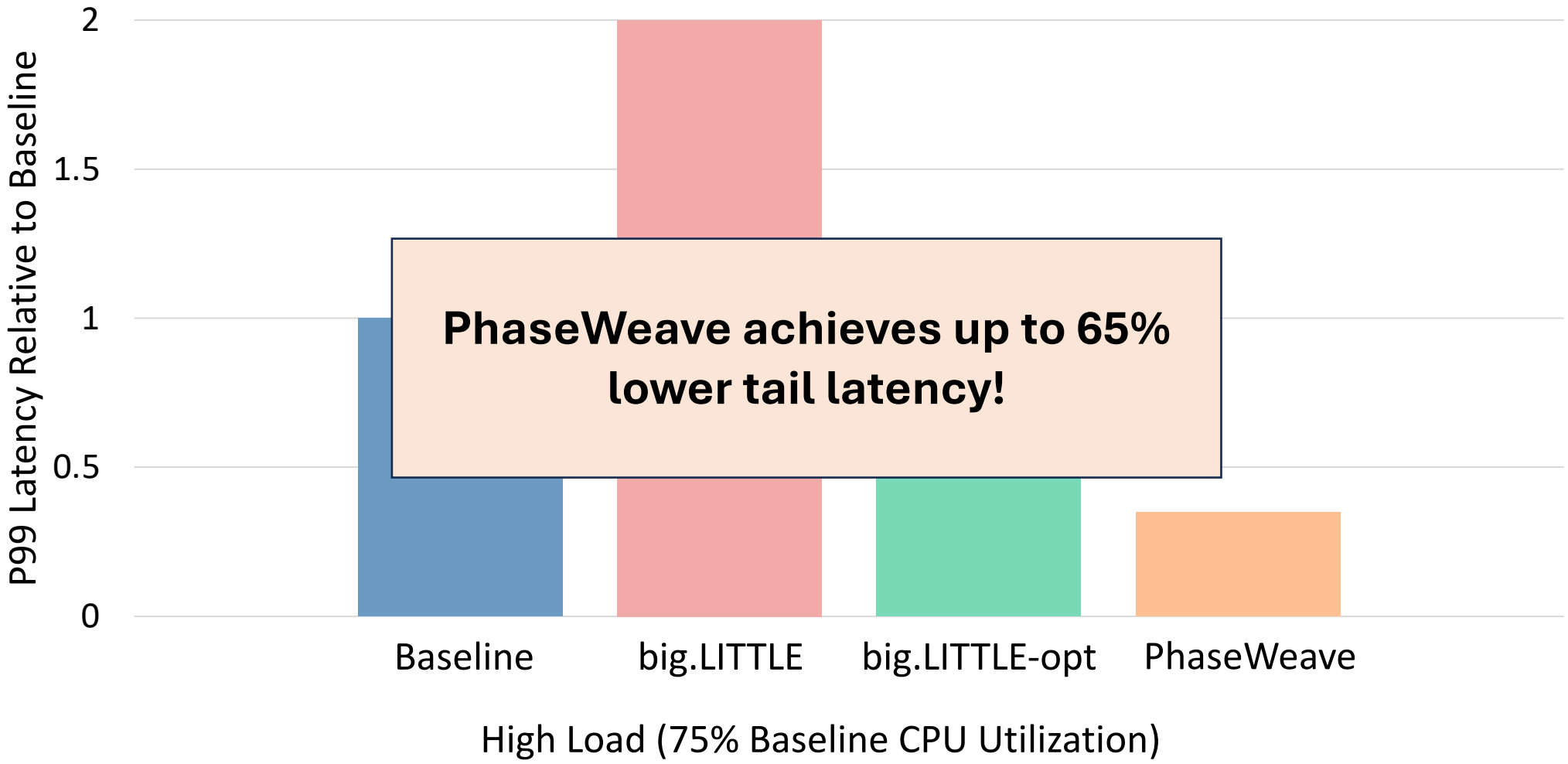
PhaseWeave Significantly Reduces Tail Latency



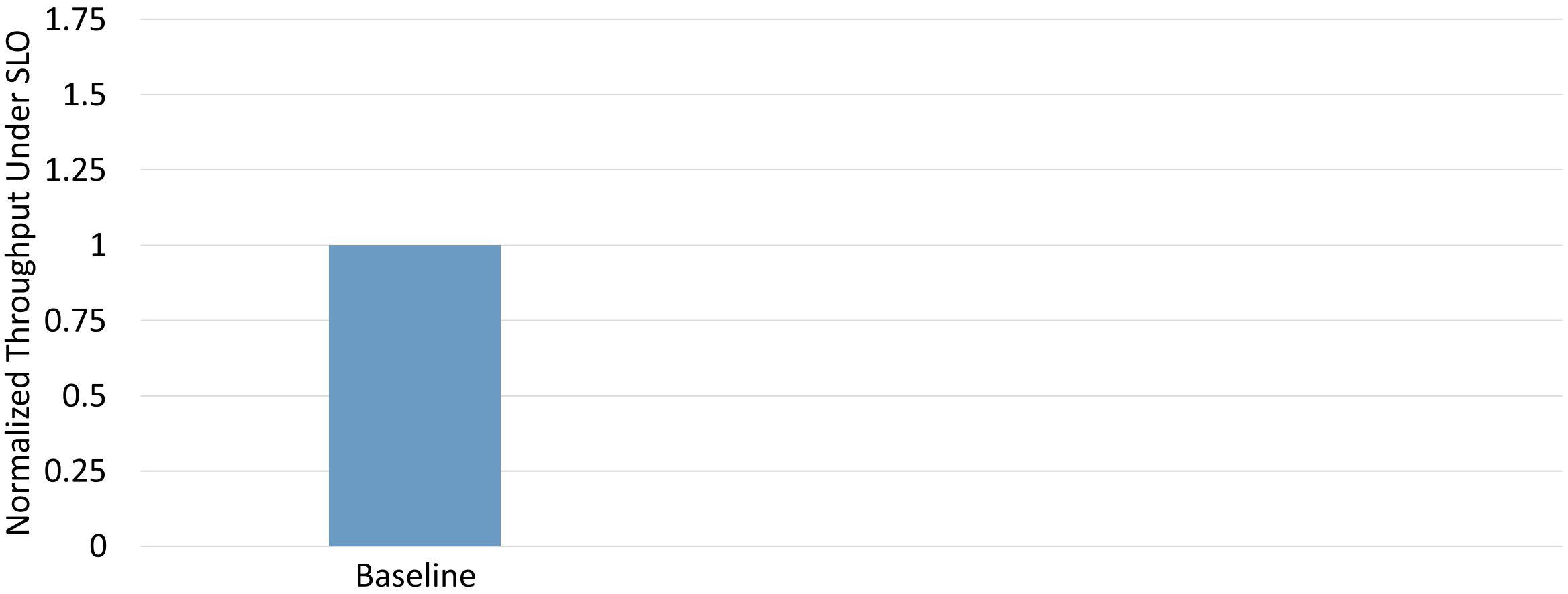
PhaseWeave Significantly Reduces Tail Latency



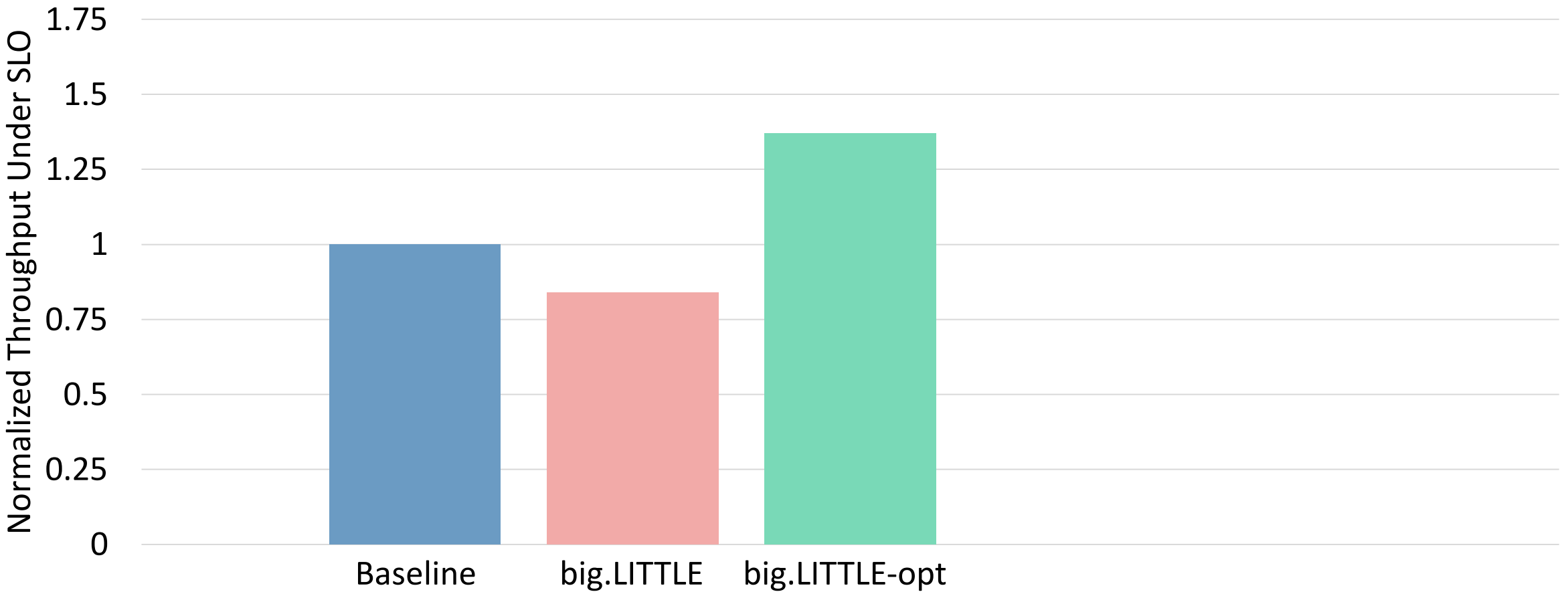
PhaseWeave Significantly Reduces Tail Latency



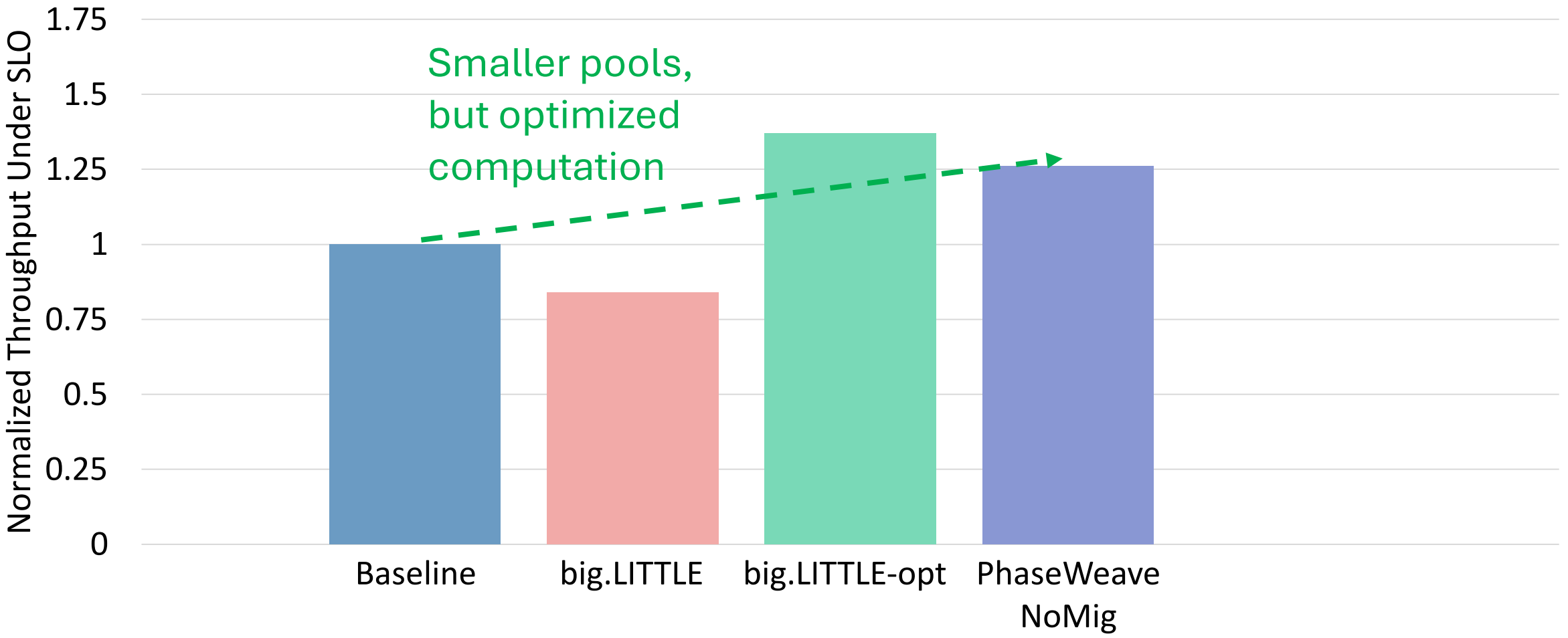
PhaseWeave Improves Throughput



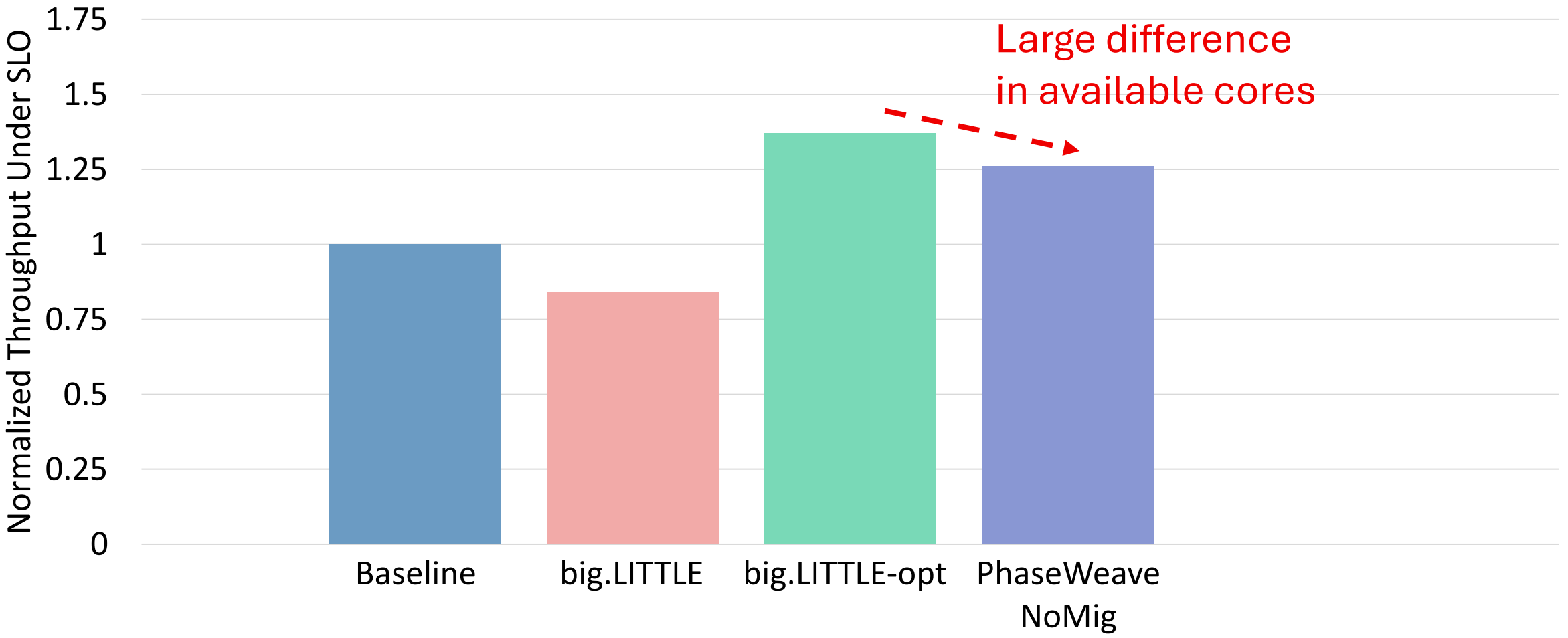
PhaseWeave Improves Throughput



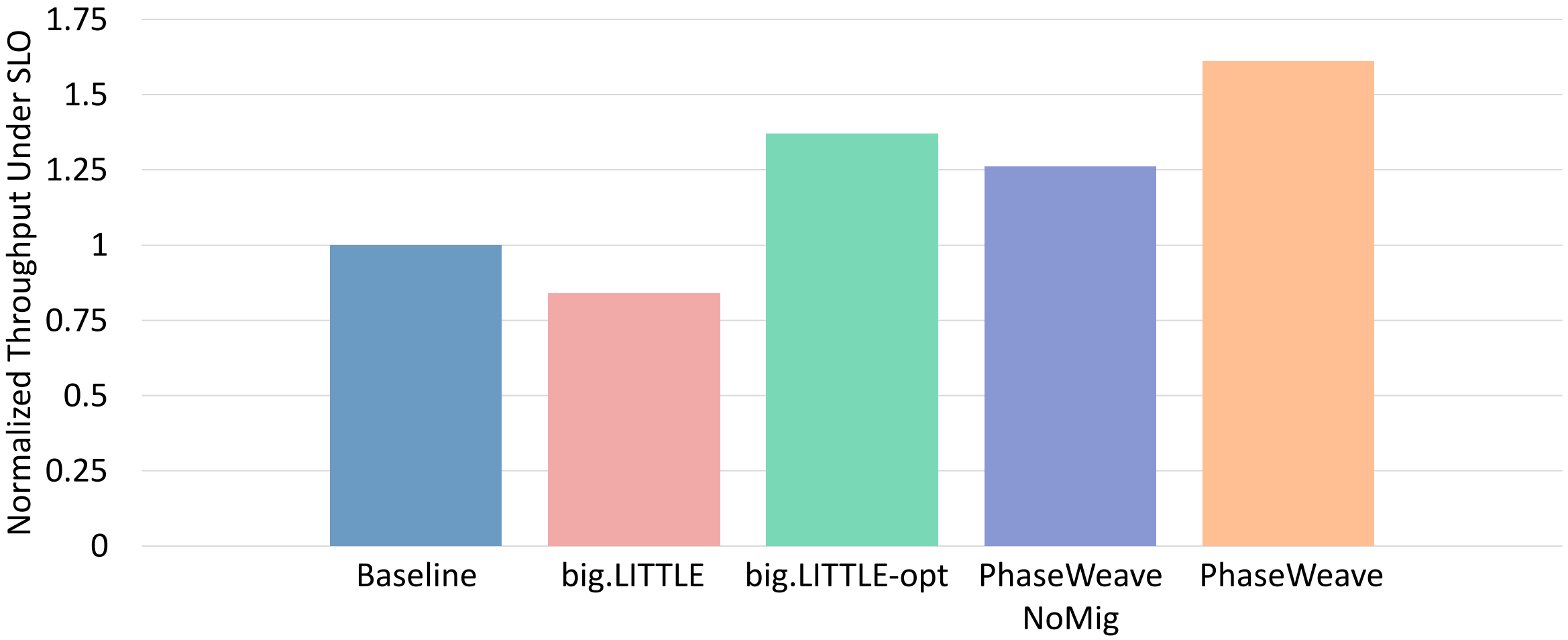
PhaseWeave Improves Throughput



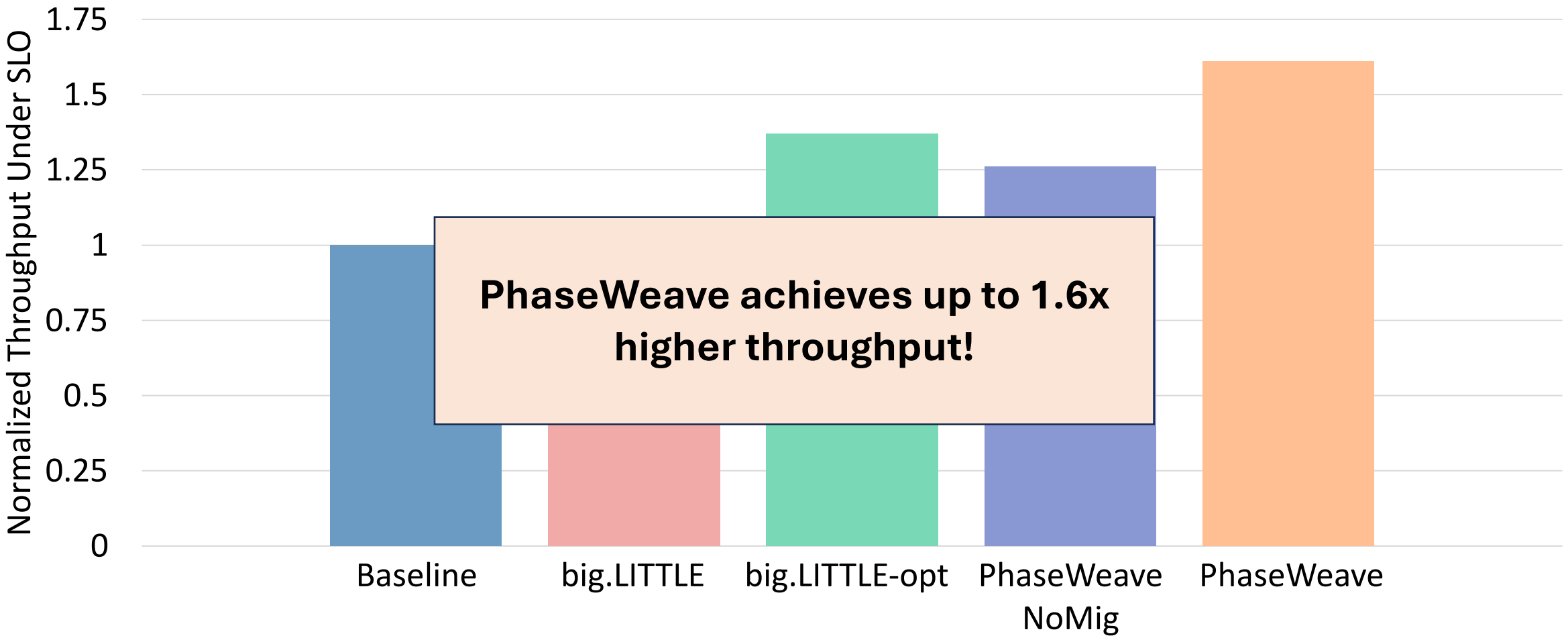
PhaseWeave Improves Throughput



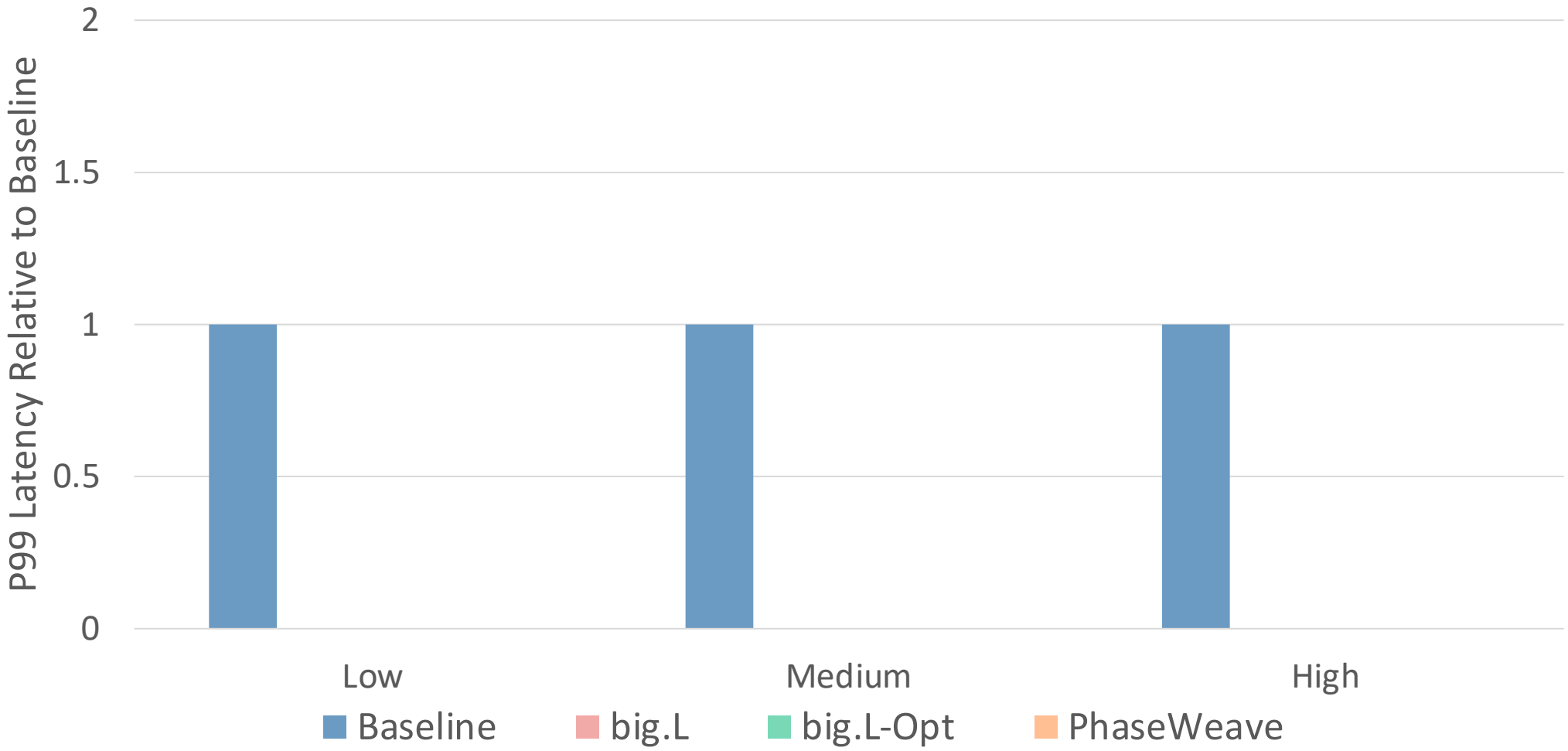
PhaseWeave Improves Throughput



PhaseWeave Improves Throughput



PhaseWeave Significantly Reduces Tail Latency



PhaseWeave Improves Cost-Efficiency

- Tuning each core type reduces per-core area and server power

Achieves 82% of Baseline Power

1.92x Performance/Watt Improvement!

Conclusion

- Fine-grained heterogeneous execution phases in datacenter workloads leaves large efficiency gaps in server design
- Matching the workload heterogeneity requires specialized hardware that can predict the upcoming execution class

Conclusion

- Fine-grained heterogeneous execution phases in datacenter workloads leaves large efficiency gaps in server design
- Matching the workload heterogeneity requires specialized hardware that can predict the upcoming execution class
- **PhaseWeave**: heterogeneous multiple-chiplet server architecture that is optimized for different workload phases
  - 65% lower tail latency
  - 1.6x higher throughput
  - 1.9x higher performance/watt

PhaseWeave: Phase-Aware Execution on Heterogeneous Chiplet Architectures for Datacenters

Joshua Kim[1], Chaojie Zhang[2], Íñigo Goiri[2], Christopher J. Rossbach[1,2], Jovan Stojkovic[1,3]

[1]The University of Texas at Austin, [2]Microsoft, [3]Meta

TEXAS
The University of Texas at Austin

Meta

Microsoft

CASCADE Lab @ UTCS

Open-Source Artifact