Sleeping with One Eye Open: Fast, Sustainable Storage with Sandman

Yanbo Zhou, Erci Xu†, Anisa Su*, Jim Harris*, Adam Manzanares*, Steven Swanson

UC San Diego

†Shanghai Jiao Tong University

*Samsung Semiconductor

SOSP 2025

NVSL

SAMSUNG

The Dilemma in Storage Evolution

Evolve towards high-performance and high-density all-flash servers

The Dilemma in Storage Evolution

Evolve towards high-performance and high-density all-flash servers

High performance

PCIe 5.0

NVMe SSDs

E.g., Samsung PM1743

Up to 2500K IOPS and 14GB/s BW

**Images and data from their official websites*

The Dilemma in Storage Evolution

Evolve towards high-performance and high-density all-flash servers

High performance

PCIe 5.0

NVMe SSDs

E.g., Samsung PM1743

Up to 2500K IOPS and 14GB/s BW

High density

Multi-layer (196)

TLC/QLC (3/4 bits / cell)

E.g., Solidigm P5336

Up to 122TB capacity per single drive

**Images and data from their official websites*

The Dilemma in Storage Evolution

Evolve towards high-performance and high-density all-flash servers

High performance

PCIe 5.0

NVMe SSDs

E.g., Samsung PM1743

Up to 2500K IOPS and 14GB/s BW

High density

Multi-layer (196)

TLC/QLC (3/4 bits / cell)

E.g., Solidigm P5336

Up to 122TB capacity per single drive

Disaggregated

High performance

RDMA NICs

E.g., NVIDIA CX-7

Up to 400Gbps and four network ports

**Images and data from their official websites*

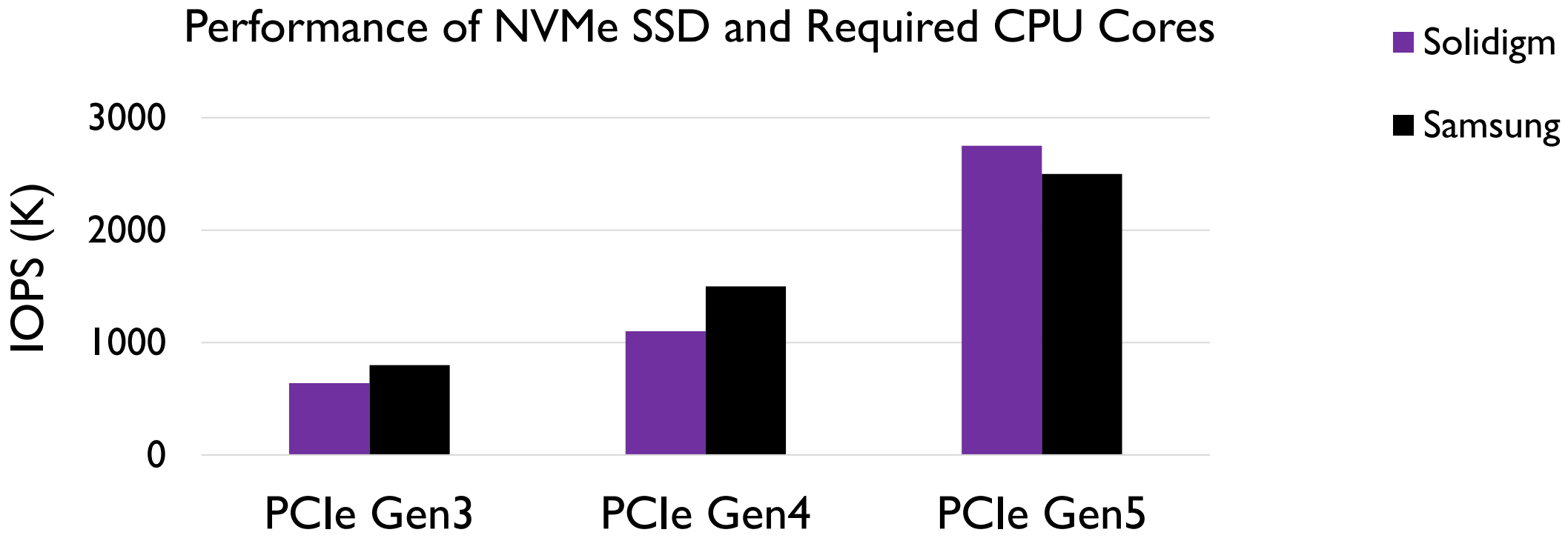
The Dilemma in Storage Evolution

Reason 1: high demand for processing power

The Dilemma in Storage Evolution

Reason 1: high demand for processing power

- SSD performance is almost **doubled** across PCIe generations

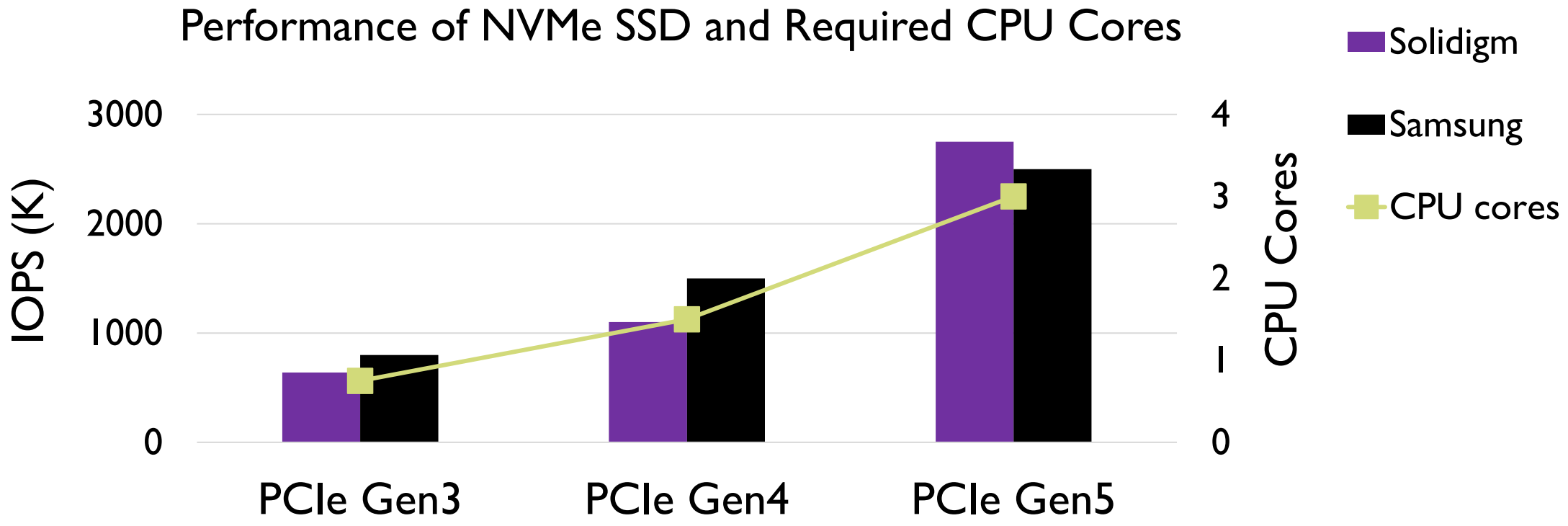


**Performance data are from official specifications from vendors and evaluated on SPDK*

The Dilemma in Storage Evolution

Reason 1: high demand for processing power

- SSD performance is almost **doubled** across PCIe generations
- Demand **more CPU cores** in host server to support such high performance



**Performance data are from official specifications from vendors and evaluated on SPDK*

The Dilemma in Storage Evolution

Reason 2: high power consumption from processing I/O

- Cloud infrastructure today uses polling-based software stack for low latency
- Reserve CPU cores to operate in busy-polling state (100% util.) regardless of load

**SSD power is captured through upcoming NVMe TP4199 feature*

The Dilemma in Storage Evolution

Reason 2: high power consumption from processing I/O

- Cloud infrastructure today uses polling-based software stack for low latency
- Reserve CPU cores to operate in busy-polling state (100% util.) regardless of load

	System Idle		Light Load		Peak Load	
	Power	Pct.	Power	Pct.	Power	Pct.
CPU	134 W	33%	244 W	34%	245 W	26%
SSD	74 W	17%	74 W	10%	296 W	32%
Others	204 W	50%	398 W	56%	399 W	42%
System	412 W	100%	716 W	100%	940 W	100%

**SSD power is captured through upcoming NVMe TP4199 feature*

The Dilemma in Storage Evolution

Reason 2: high power consumption from processing I/O

- Cloud infrastructure today uses polling-based software stack for low latency
- Reserve CPU cores to operate in busy-polling state (100% util.) regardless of load

1. SSD can adaptively change power based on workloads

	System Idle		Light Load		Peak Load	
	Power	Pct.	Power	Pct.	Power	Pct.
CPU	134 W	33%	244 W	34%	245 W	26%
SSD	74 W	17%	74 W	10%	296 W	32%
Others	204 W	50%	398 W	56%	399 W	42%
System	412 W	100%	716 W	100%	940 W	100%

SSD power (5W-25W) goes up to 25W only when it comes to 100% performance utilization and GC is running to erase data.

**SSD power is captured through upcoming NVMe TP4199 feature*

The Dilemma in Storage Evolution

Reason 2: high power consumption from processing I/O

- Cloud infrastructure today uses polling-based software stack for low latency
- Reserve CPU cores to operate in busy-polling state (100% util.) regardless of load

1. SSD can adaptively change power based on workloads

	System Idle		Light Load		Peak Load	
	Power	Pct.	Power	Pct.	Power	Pct.
CPU	134 W	33%	244 W	34%	245 W	26%
SSD	74 W	17%	74 W	10%	296 W	32%
Others	204 W	50%	398 W	56%	399 W	42%
System	412 W	100%	716 W	100%	940 W	100%

2. CPU consumes high power consumption ***regardless of load.***

SSD power (5W-25W) goes up to 25W only when it comes to 100% performance utilization and GC is running to erase data;

**SSD power is captured through upcoming NVMe TP4199 feature*

The Dilemma in Storage Evolution

Reason 2: high power consumption from processing I/O

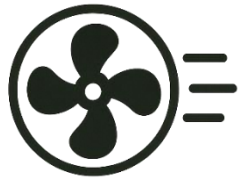
- Cloud infrastructure today uses polling-based software stack for low latency
- Reserve CPU cores to operate in busy-polling state (100% util.) regardless of load

1. SSD can adaptively change power based on workloads

	System Idle		Light Load		Peak Load	
	Power	Pct.	Power	Pct.	Power	Pct.
CPU	134 W	33%	244 W	34%	245 W	26%
SSD	74 W	17%	74 W	10%	296 W	32%
Others	204 W	50%	398 W	56%	399 W	42%
System	412 W	100%	716 W	100%	940 W	100%

*2. CPU consumes high power consumption **regardless of load.***

*3. Lead to high **collateral power consumption** from other components*



SSD power (5W-25W) goes up to 25W only when it comes to 100% performance utilization and GC is running to erase data;

**SSD power is captured through upcoming NVMe TP4199 feature*

The Dilemma in Storage Evolution

Reason 2: high power consumption from processing I/O

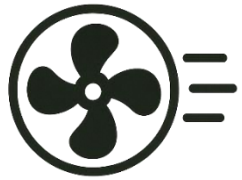
- Cloud infrastructure today uses polling-based software stack for low latency
- Reserve CPU cores to operate in busy-polling state (100% util.) regardless of load

1. SSD can adaptively change power based on workloads

	System Idle		Light Load		Peak Load	
	Power	Pct.	Power	Pct.	Power	Pct.
CPU	134 W	33%	244 W	34%	245 W	26%
SSD	74 W	17%	74 W	10%	296 W	32%
Others	204 W	50%	398 W	56%	399 W	42%
System	412 W	100%	716 W	100%	940 W	100%

*2. CPU consumes high power consumption **regardless of load.***

*3. Lead to high **collateral power consumption** from other components*



SSD power (5W-25W) goes up to 25W only when it comes to 100% performance utilization and GC is running to erase data;

**SSD power is captured through upcoming NVMe TP4199 feature*

The Dilemma in Storage Evolution

Reason 3: frequent variances and bursts in the field

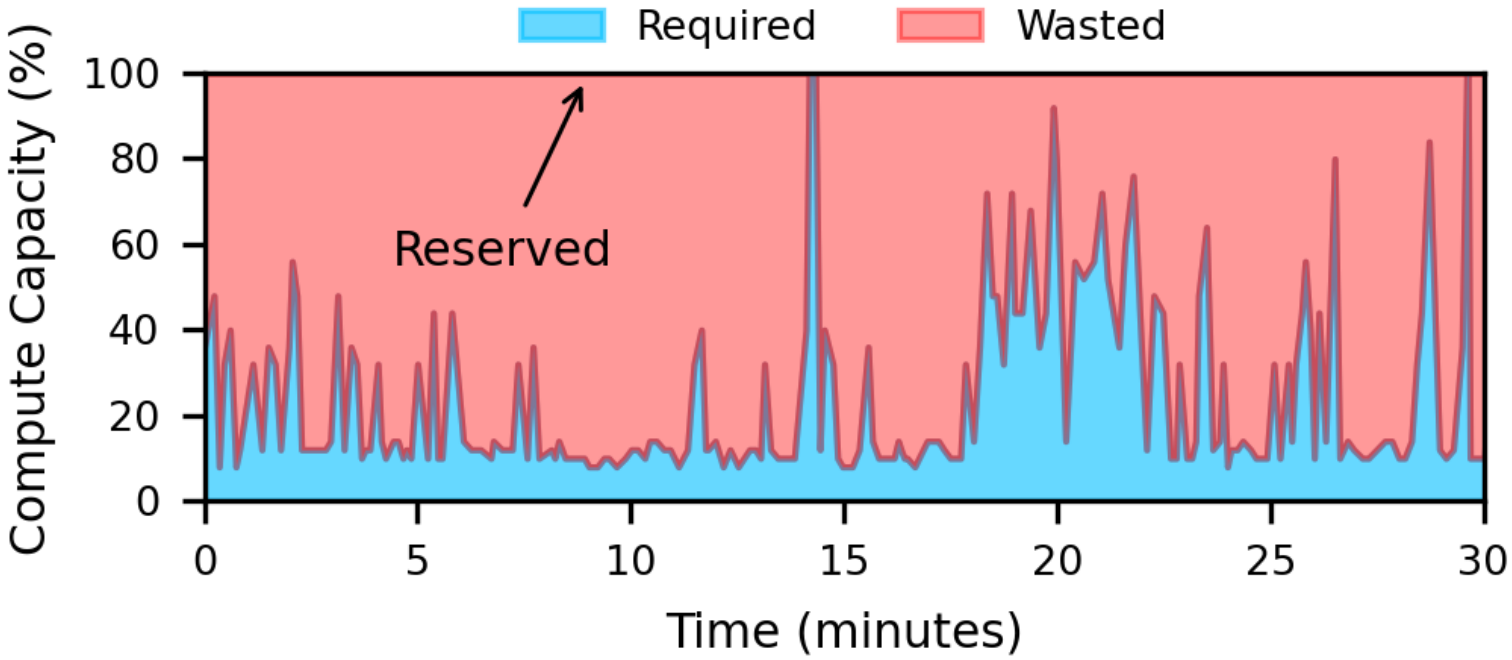
- High power consumption is acceptable if I/O pressure is consistently heavy
- However, hardware advancements have made bursts more severe and short-lived

**Data are from real cluster traces of Alibaba Cloud*

The Dilemma in Storage Evolution

Reason 3: frequent variances and bursts in the field

- High power consumption is acceptable if I/O pressure is consistently heavy
- However, hardware advancements have made bursts more severe and short-lived



Reserved, required, and wasted compute capacity under real cloud workloads.

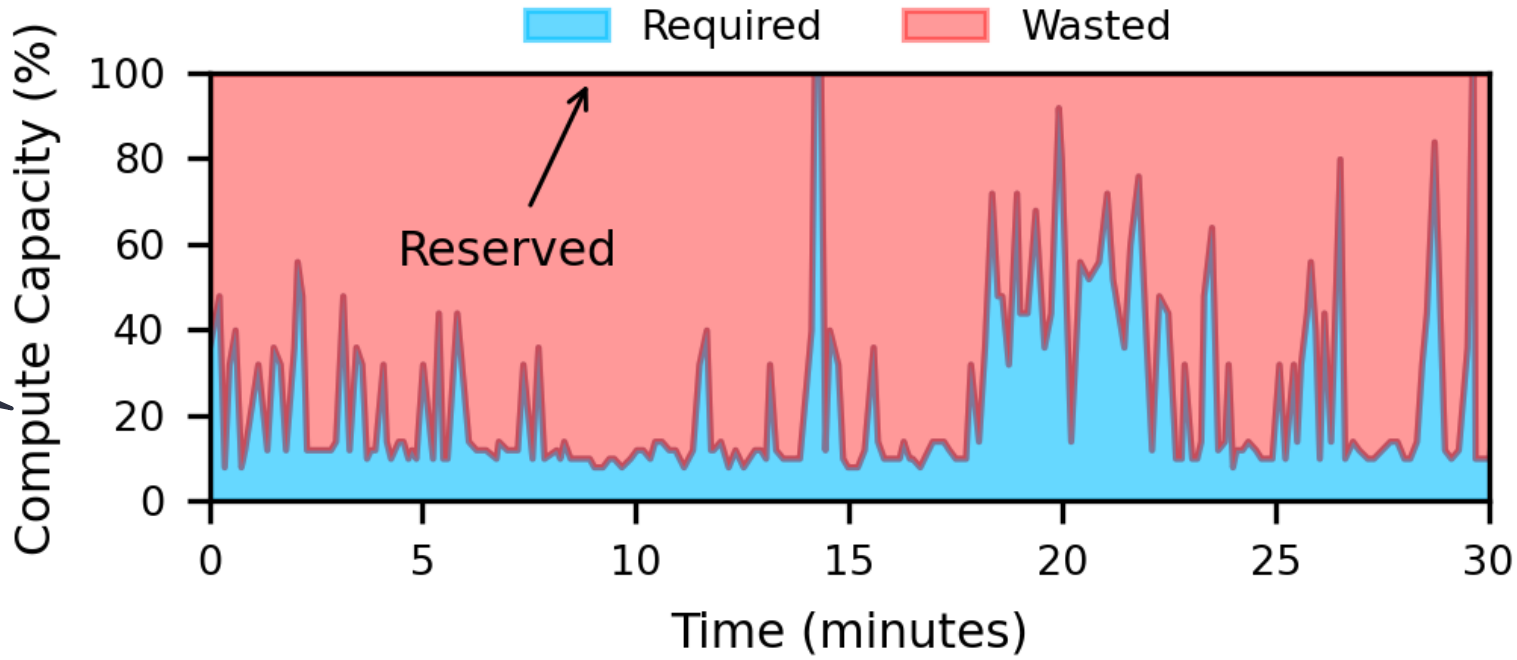
**Data are from real cluster traces of Alibaba Cloud*

The Dilemma in Storage Evolution

Reason 3: frequent variances and bursts in the field

- High power consumption is acceptable if I/O pressure is consistently heavy
- However, hardware advancements have made bursts more severe and short-lived

Equivalent to energy consumption caused by CPU activities.



Reserved, required, and wasted compute capacity under real cloud workloads.

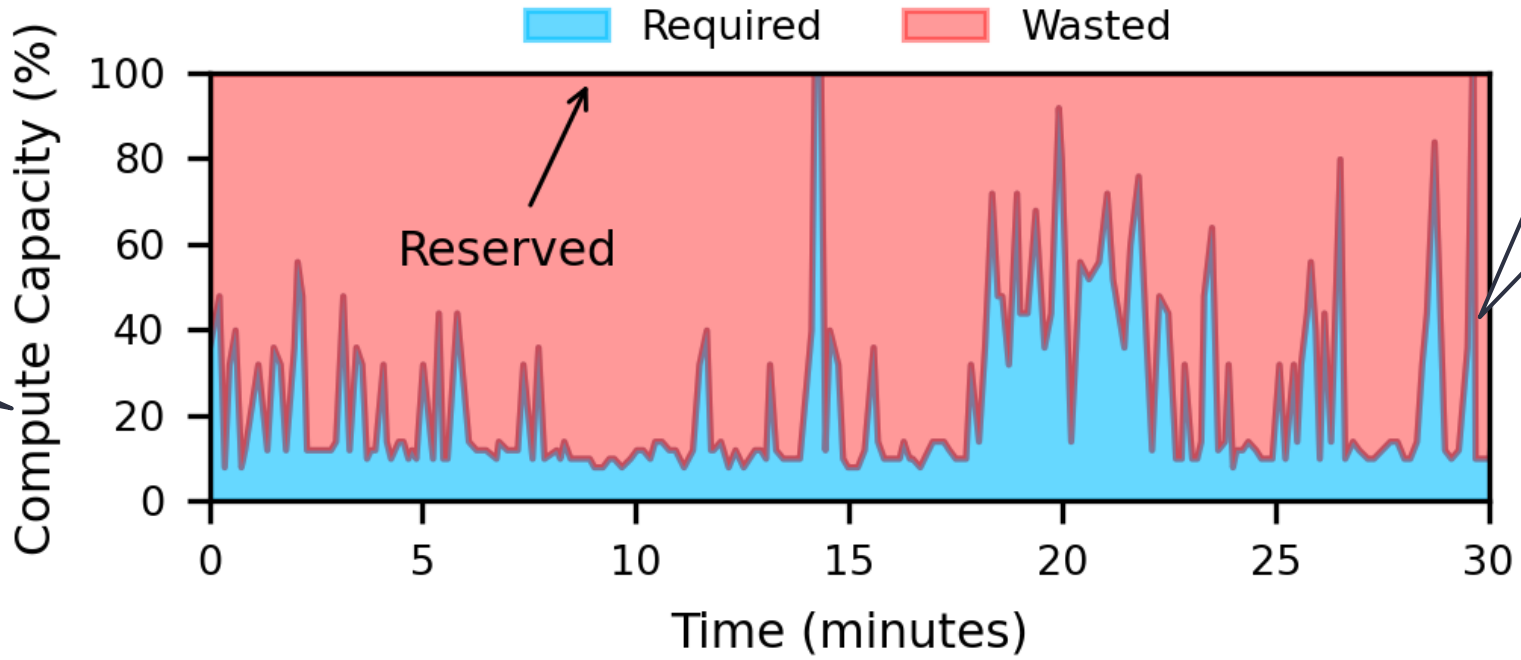
**Data are from real cluster traces of Alibaba Cloud*

The Dilemma in Storage Evolution

Reason 3: frequent variances and bursts in the field

- High power consumption is acceptable if I/O pressure is consistently heavy
- However, hardware advancements have made bursts more severe and short-lived

Equivalent to energy consumption caused by CPU activities.



Polling cores improve performance under bursts but waste lots of energy

Reserved, required, and wasted compute capacity under real cloud workloads.

**Data are from real cluster traces of Alibaba Cloud*

Overview

- **Investigation**:
  - power consumption of all-flash servers

- **Measurement study**:
  - Measure four existing potential power saving solutions of storage stacks
  - Identify four major challenges to achieve high performance and energy efficiency

- **Our solution: Sandman**
  - A scheduling framework for modern storage
  - Achieve both high performance and energy efficiency

- **Evaluation**
  - Application study
  - Real cloud workloads

Measurement Study

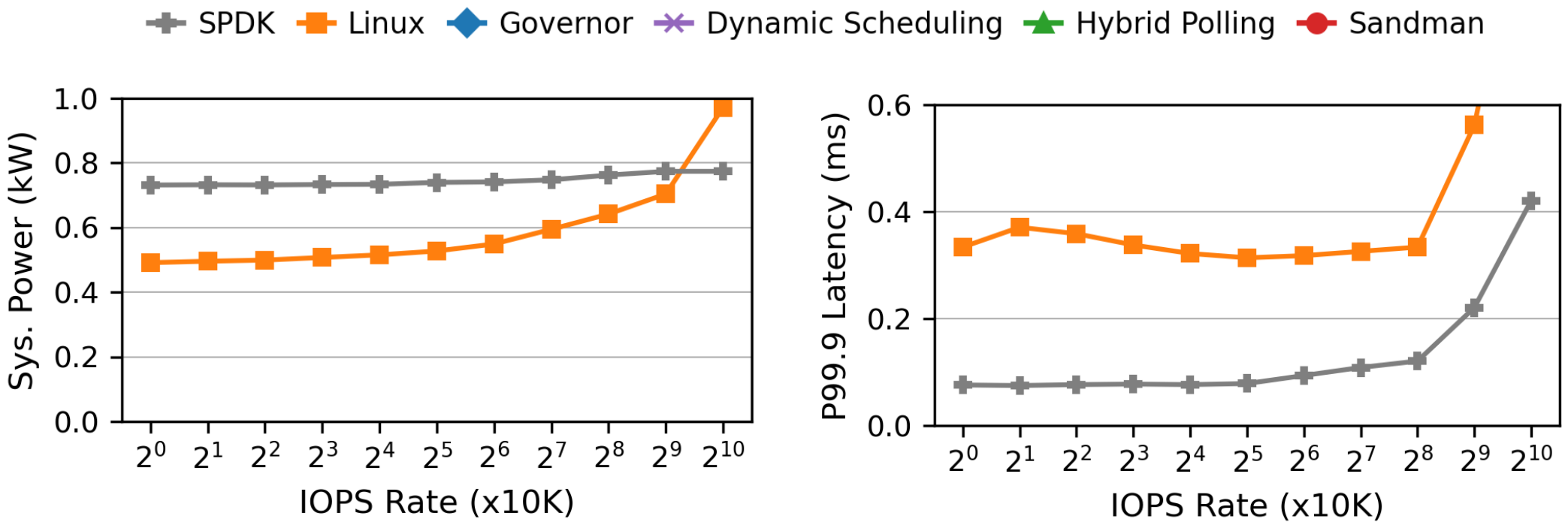
Ideal: comparable performance to busy polling with lower power consumption.

1. **SPDK**: busy-polling – best performance case
2. **Linux**: classic interrupts
3. **Governor**: adapting frequency of polling cores to workloads
4. **Dynamic Scheduling**: reallocating threads among polling cores
5. **Hybrid Polling**: alternating polling cores between sleep and active states

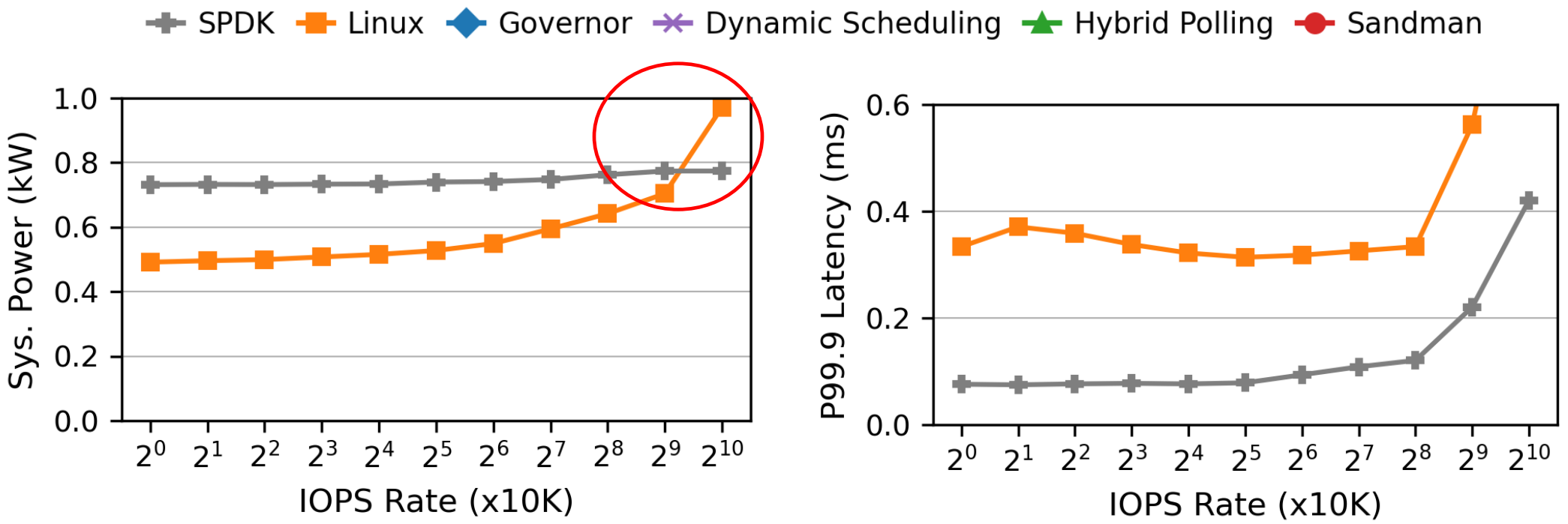
Workloads: 1) stable workloads at different IOPS rates and 2) bursty (1s) workloads

Setup: two nodes connected via NVMeoF (16x PM1743 SSDs and RDMA NICs with 2x 200Gbps ports)

Stable Workloads – Linux Interrupts

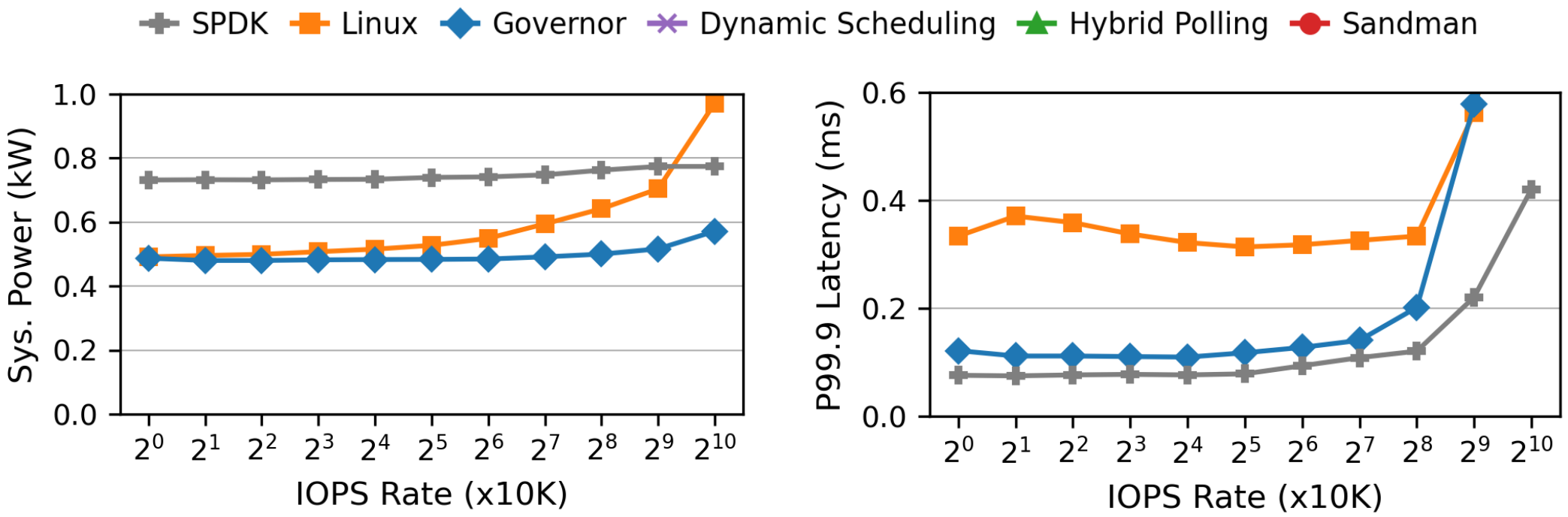


Stable Workloads – Linux Interrupts

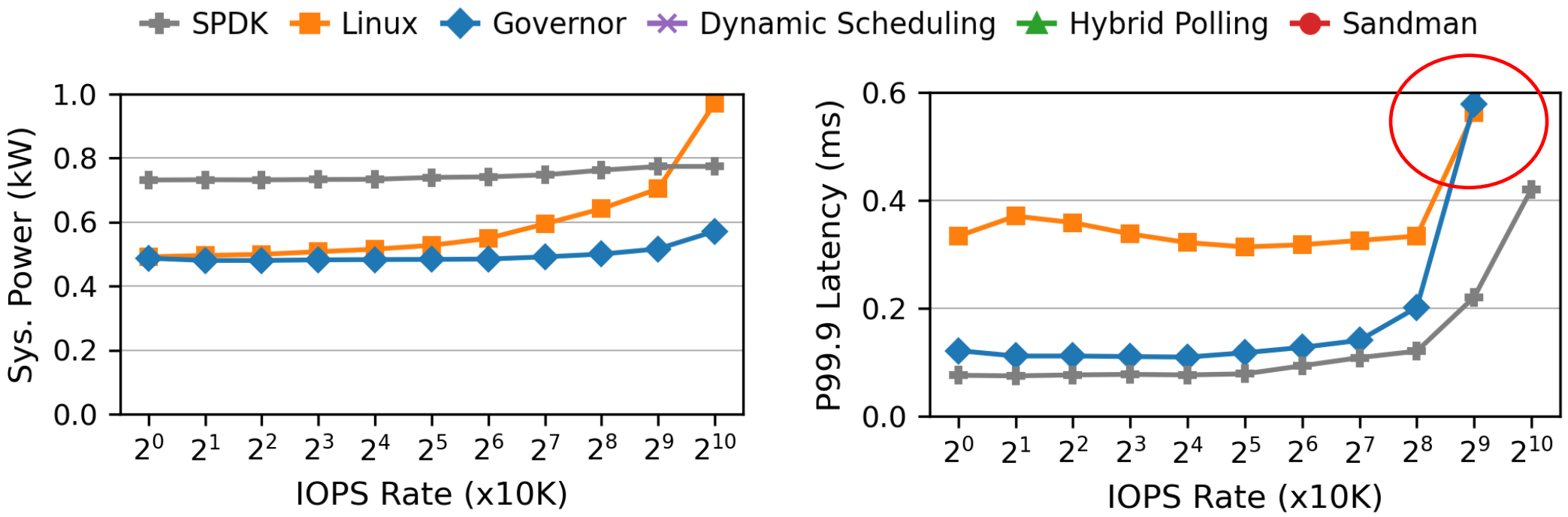


Observation: Interrupts can even consume ***higher-than-polling* power** and show lower performance due to context switches under heavy workloads

Stable Workloads – Governor Frequency Scaling

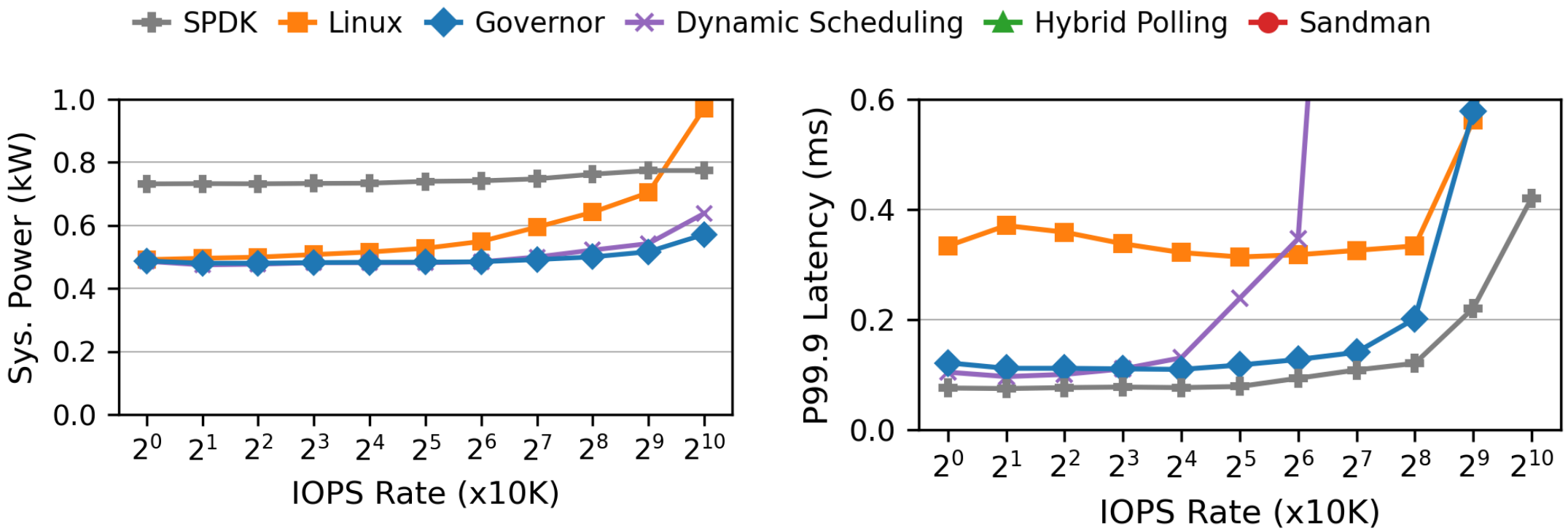


Stable Workloads – Governor Frequency Scaling

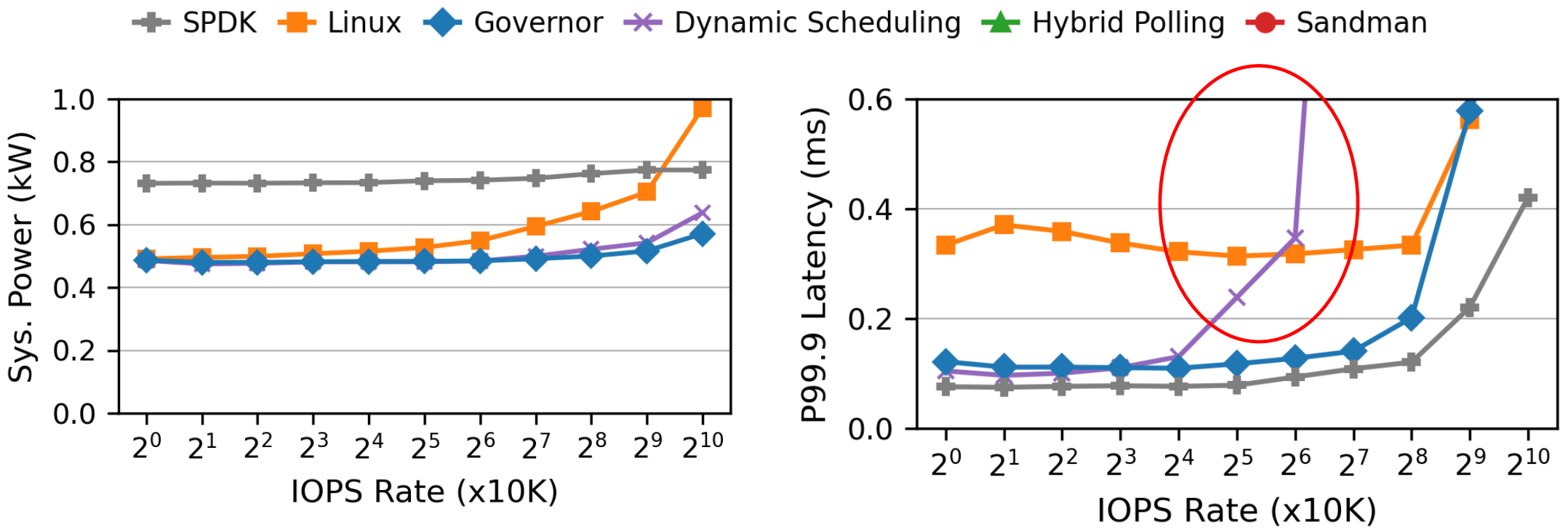


Observation: Governor reduces power consumption yet **increases tail latency** than busy-polling due to using low-frequency cores for processing

Stable Workloads – Dynamic Scheduling

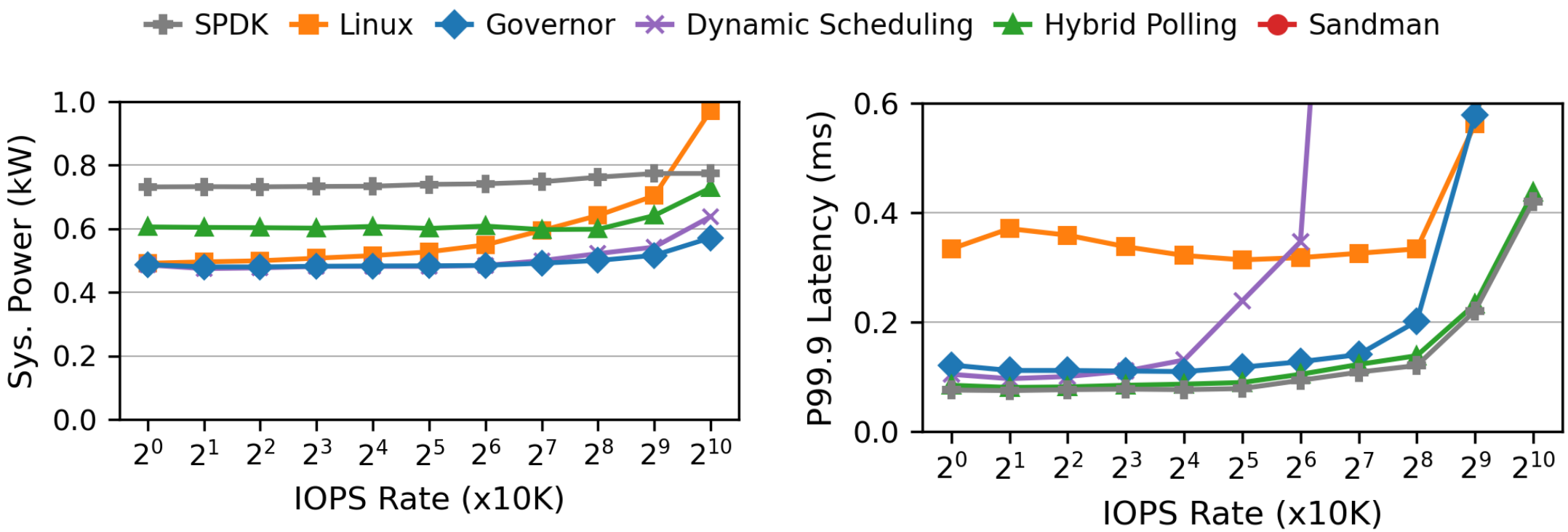


Stable Workloads – Dynamic Scheduling

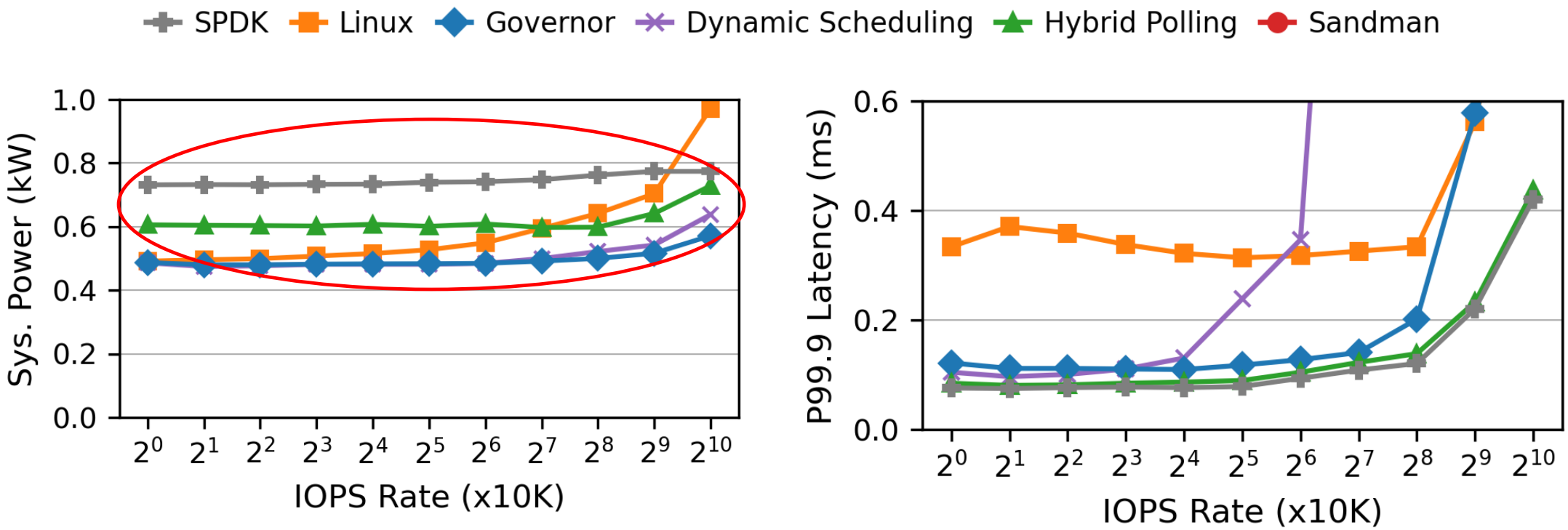


Observation: Adapting workload changes needs **microsecond-level detection**, incurring **frequent reallocation of tasks among cores.**

Stable Workloads – Hybrid Polling



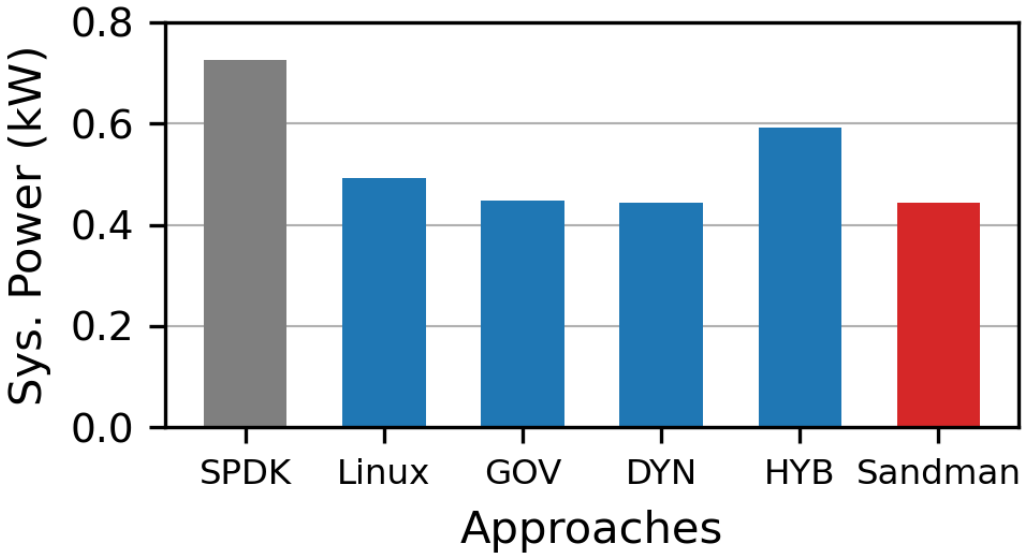
Stable Workloads – Hybrid Polling

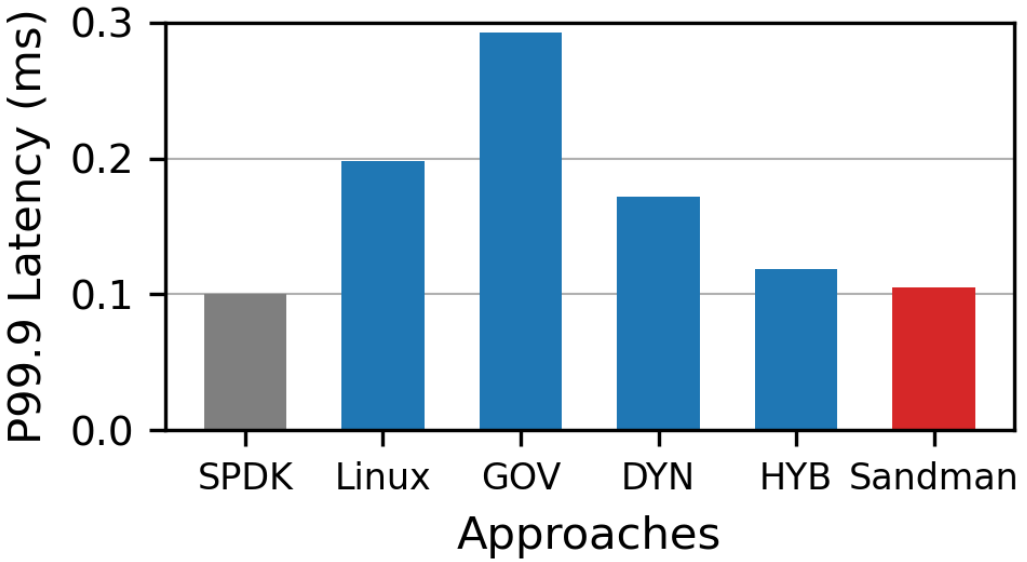


Observation: **Frequent transitions limit power saving**. Frequent transitions between low-power and high-performance states prevent CPU cores from staying in energy-saving mode for longer durations.

Bursty Workloads

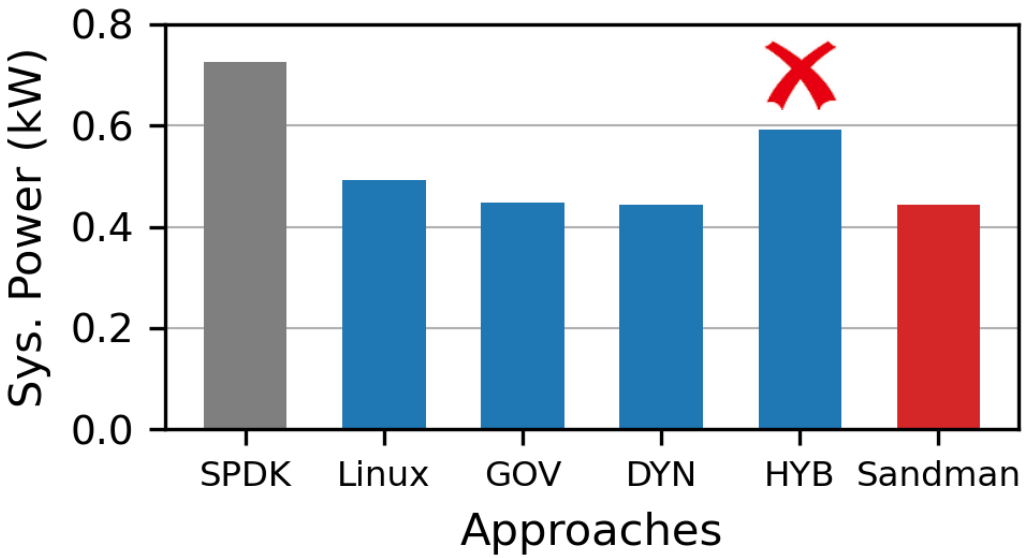
We expect system scales compute resource ASAP under short-term bursts

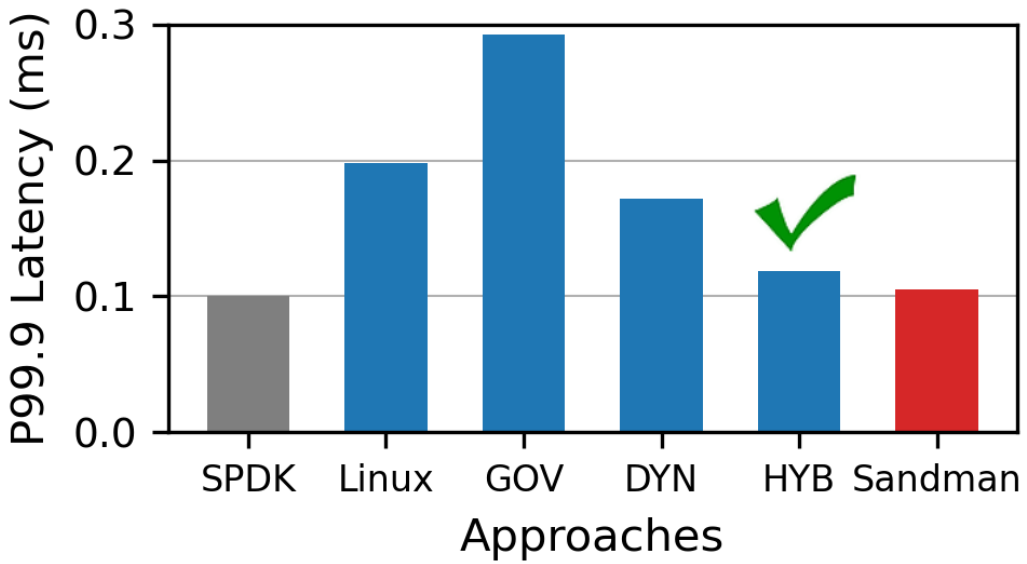




Bursty Workloads

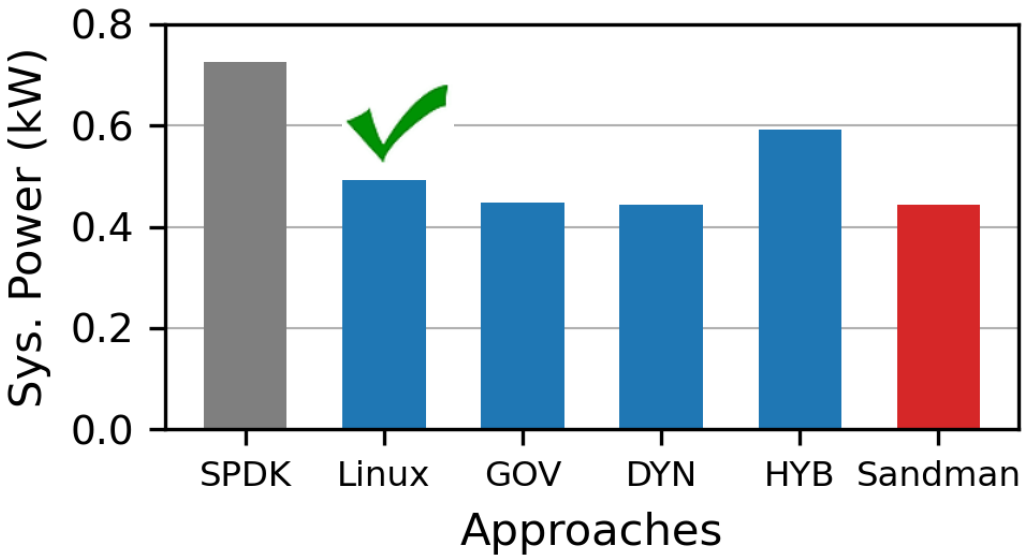
We expect system scales compute resource ASAP under short-term bursts

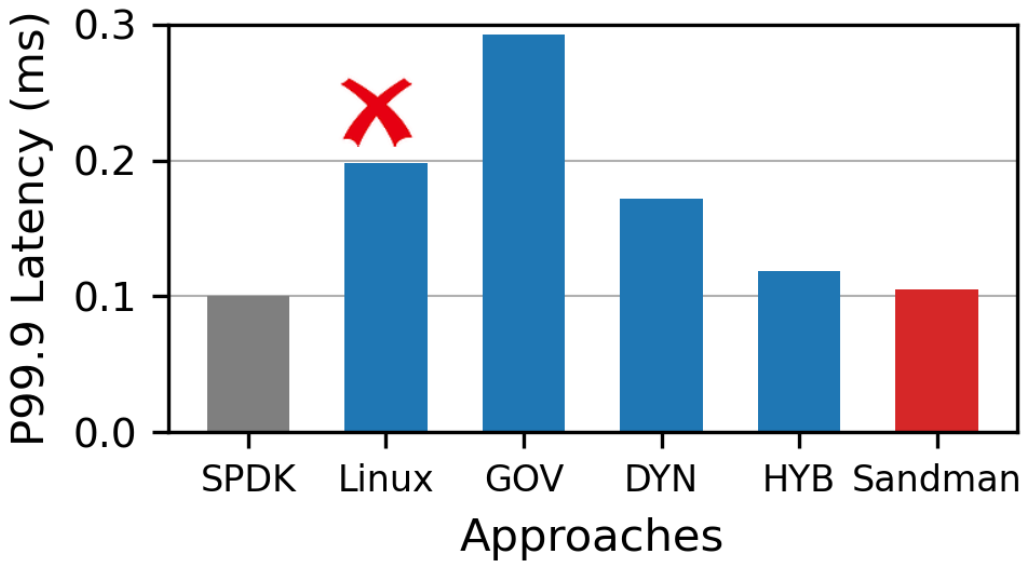




Bursty Workloads

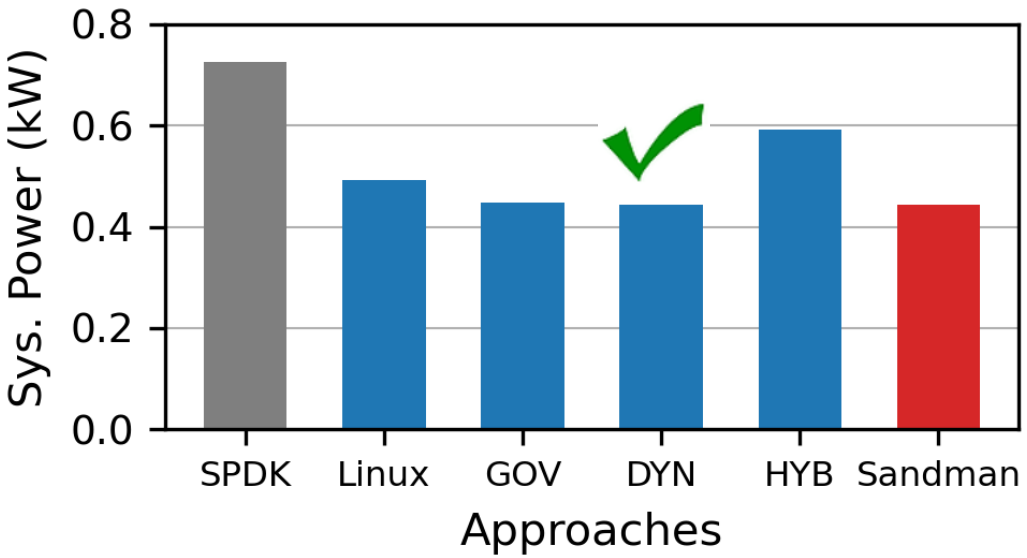
We expect system scales compute resource ASAP under short-term bursts

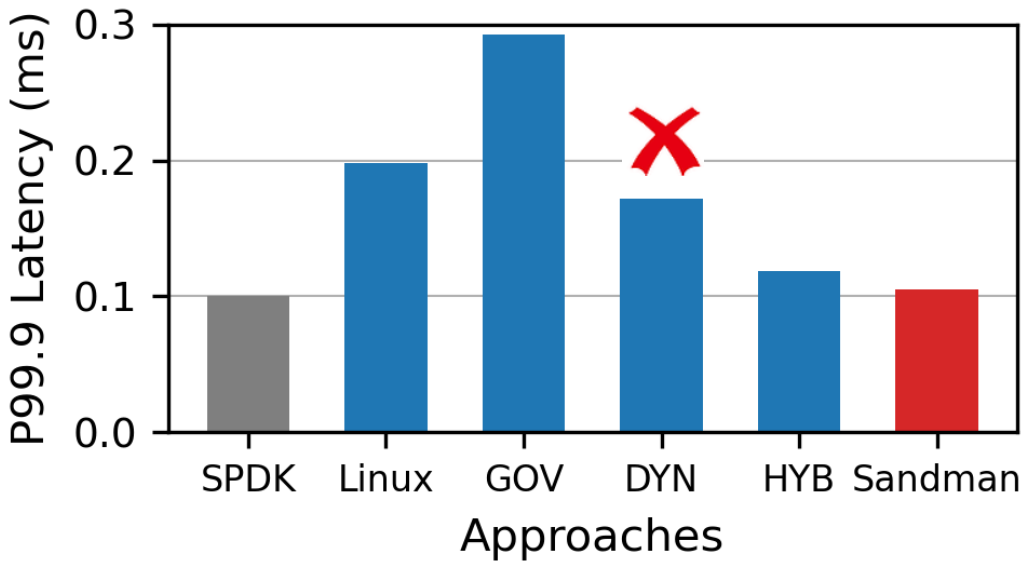




Bursty Workloads

We expect system scales compute resource ASAP under short-term bursts

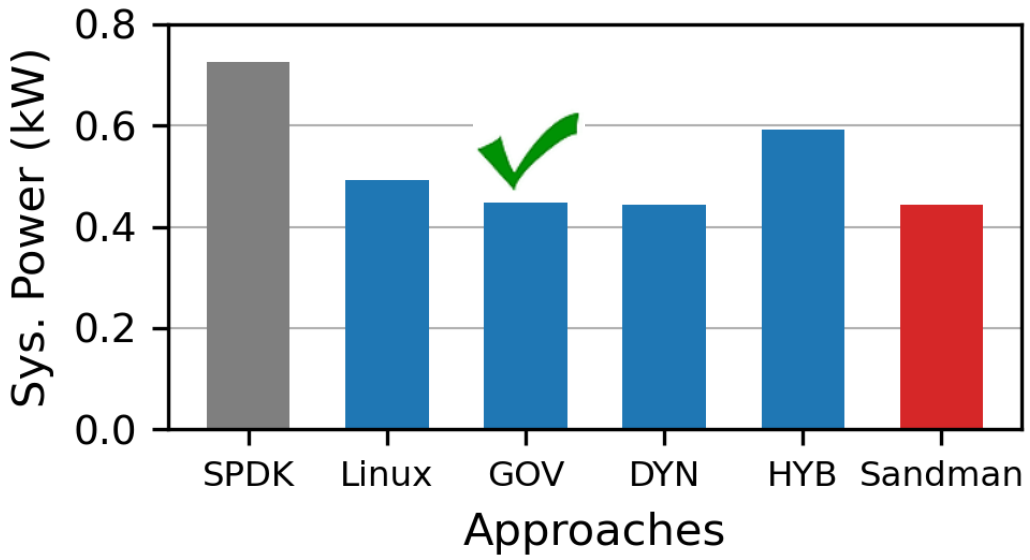


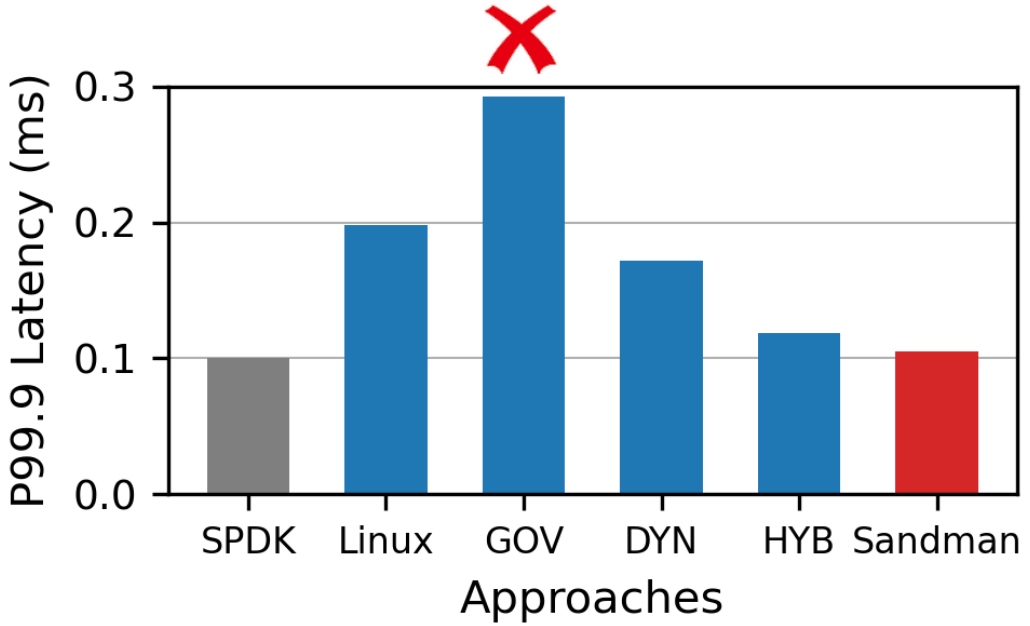


Observation: Dynamic scheduling wakes up sleeping cores through **software-based interrupts**, further leading to high tail latency

Bursty Workloads

We expect system scales compute resource ASAP under short-term bursts

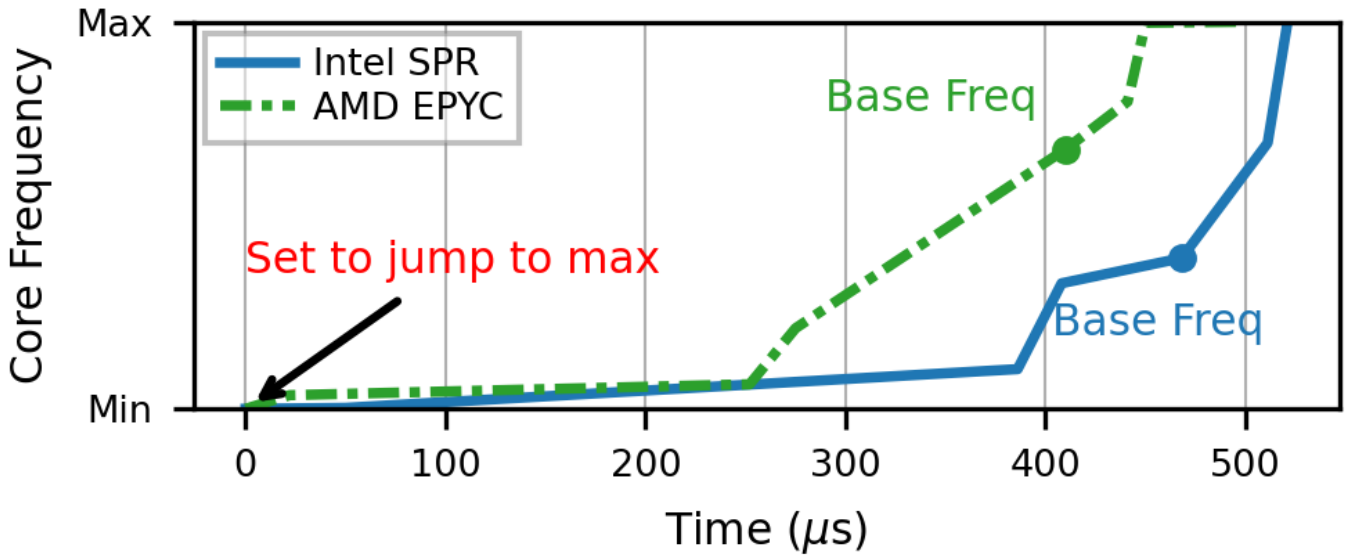




Observation: Governor increases core frequency to handle bursts, but **adjusting cores frequency takes time,** thus leading to high tail latency

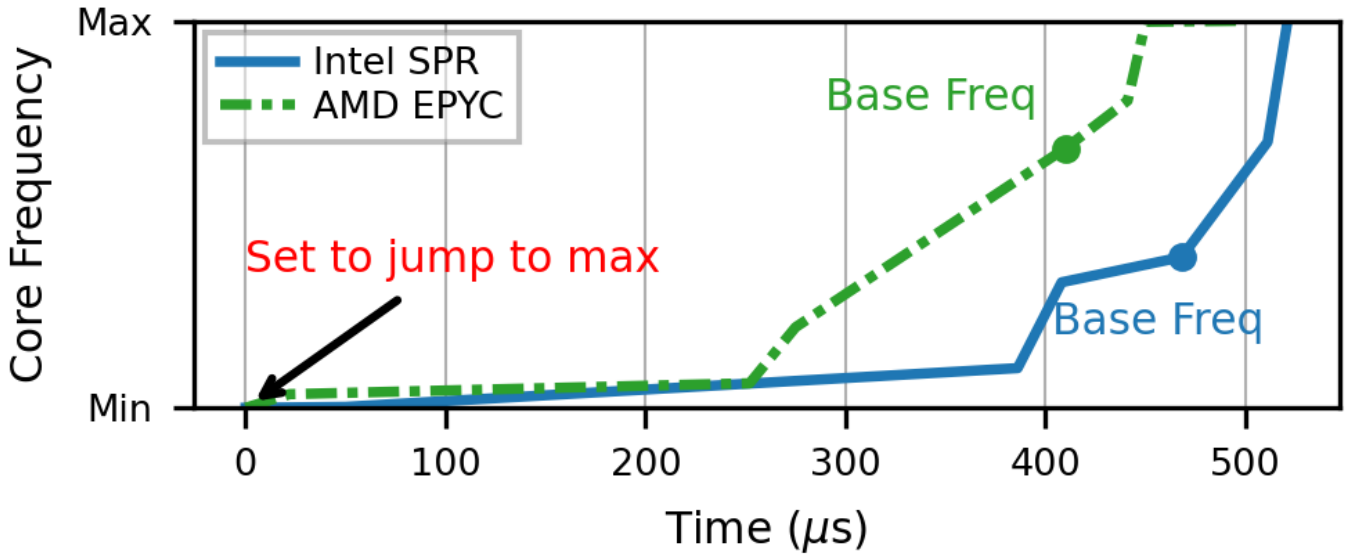
Dive Into Core Frequency Scaling Overhead

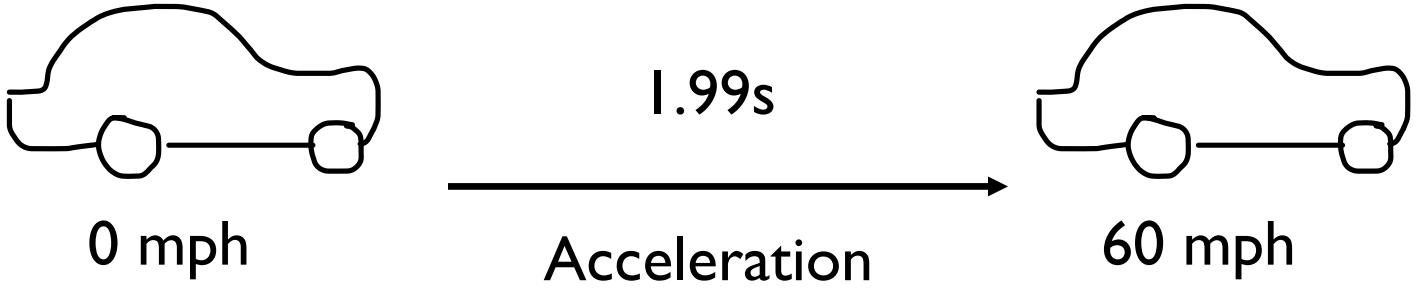
- Effective CPU frequency is invisible to users
- CPU takes time to increate to high frequency



Dive Into Core Frequency Scaling Overhead

- Effective CPU frequency is invisible to users
- CPU takes time to increate to high frequency





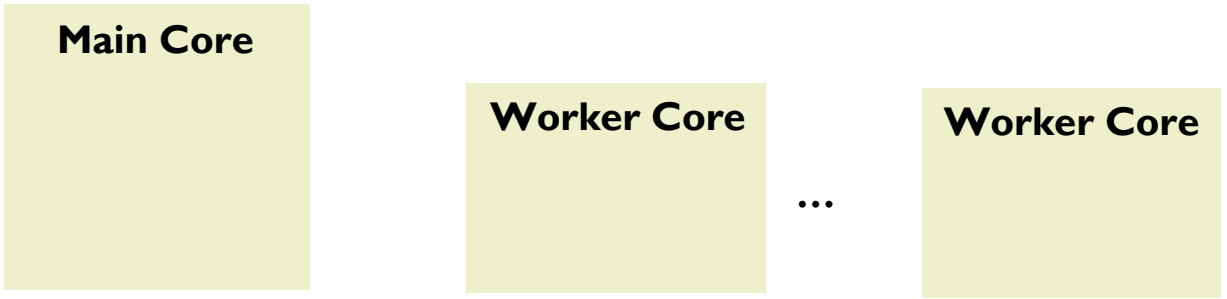
Challenges and Design Guidelines

Challenges	Guidelines
Changing core from low to high power states takes time	Shallow sleep cores NOT lower their frequency
Wake up sleeping cores via SW-based interrupts → high latency	Avoid system calls/interrupts for waking up cores
Frequent transitions from sleep to polling limit power saving	Schedule sibling HT-cores together for longer sleep
Unprecise workload estimation leads frequent thread reallocation	Measure short-term bursts from network queues

Sandman: Scheduler for Fast, Sustainable Storage

System architecture

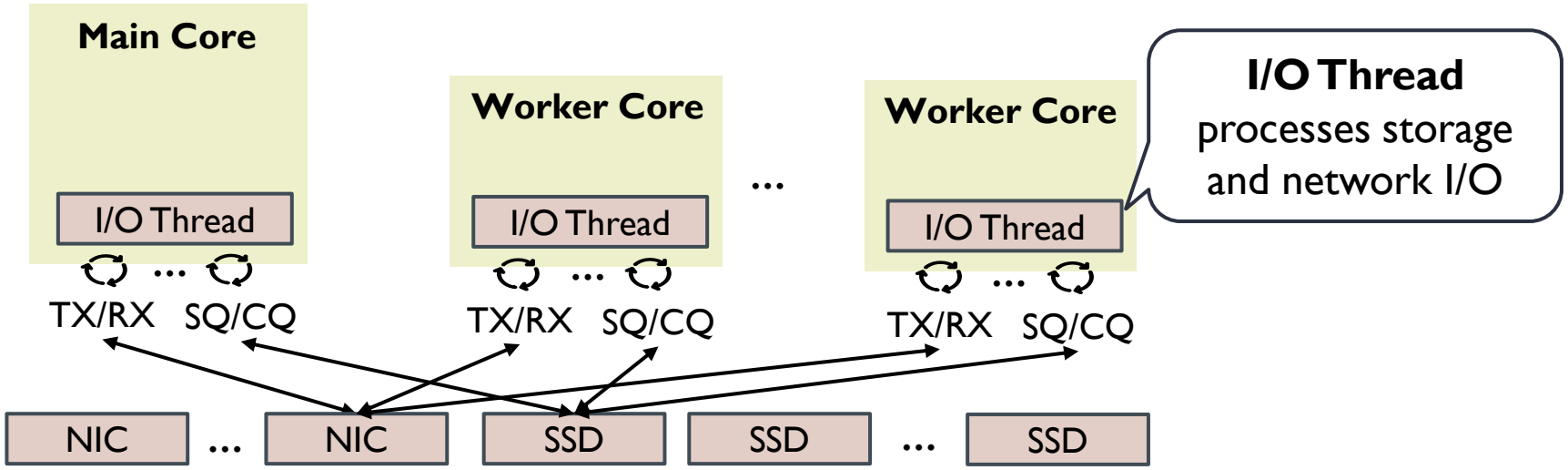
- Reserve CPU cores as ***worker cores*** and a ***main core*** running in polling mode



Sandman: Scheduler for Fast, Sustainable Storage

System architecture

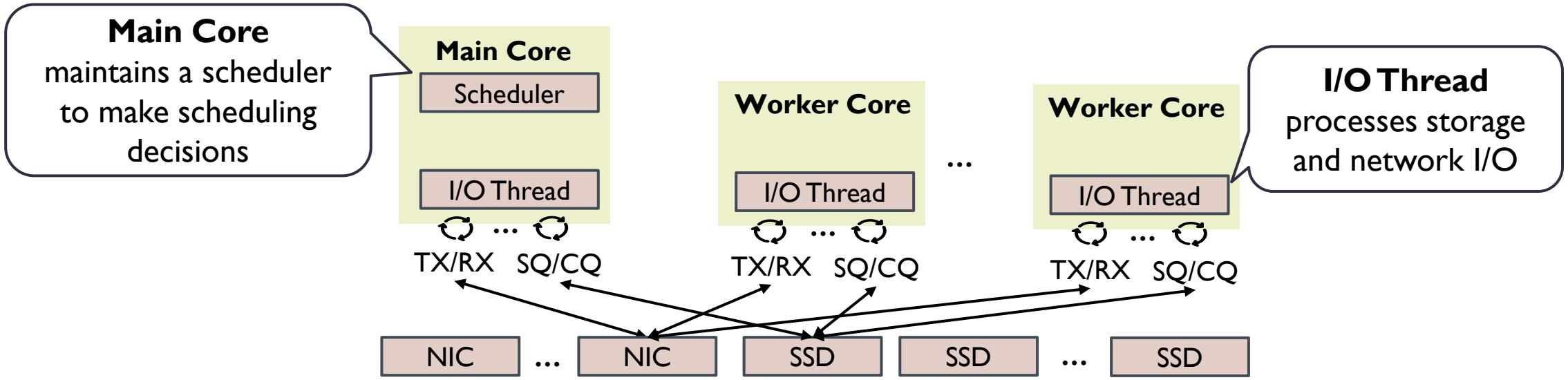
- Reserve CPU cores as ***worker cores*** and a ***main core*** running in polling mode



Sandman: Scheduler for Fast, Sustainable Storage

System architecture

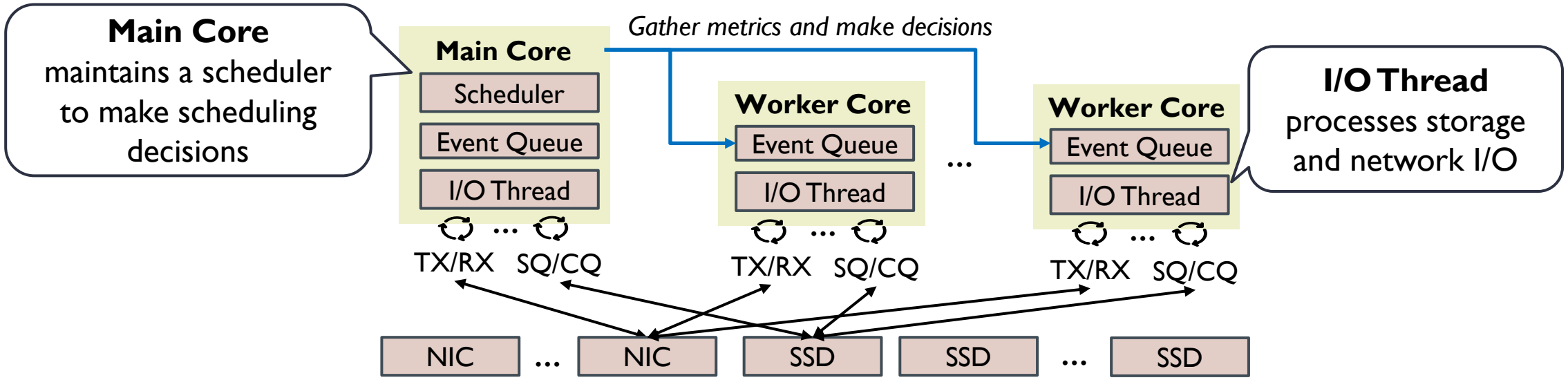
- Reserve CPU cores as ***worker cores*** and a ***main core*** running in polling mode



Sandman: Scheduler for Fast, Sustainable Storage

System architecture

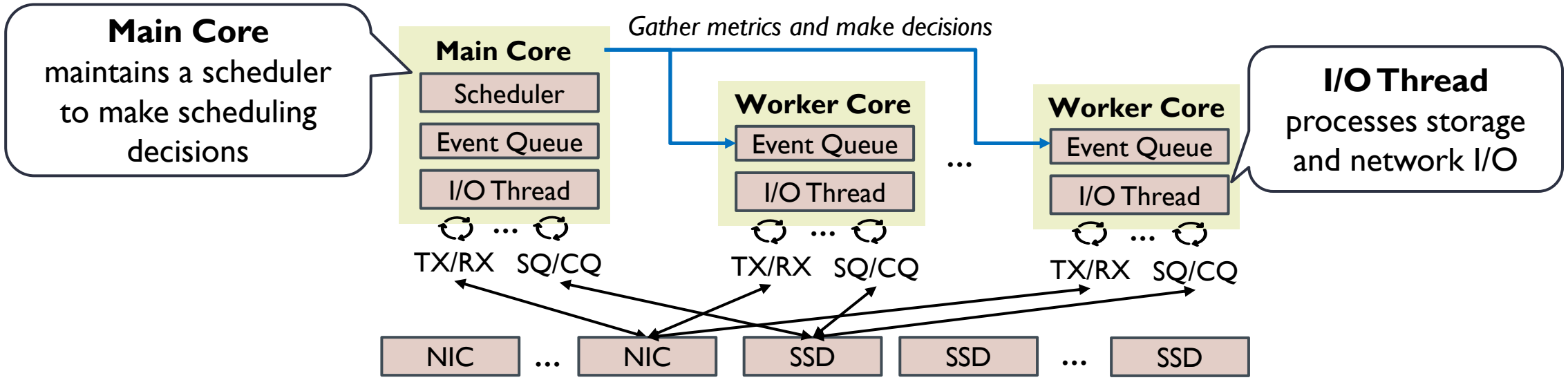
- Reserve CPU cores as ***worker cores*** and a ***main core*** running in polling mode
- Core-to-core communicate through ***event queues*** by sending event messages



Sandman: Scheduler for Fast, Sustainable Storage

System architecture

- Reserve CPU cores as ***worker cores*** and a ***main core*** running in polling mode
- Core-to-core communicate through ***event queues*** by sending event messages



Key design

- Fast Resource Scaling **Mechanism**
- Resource Monitoring and Burst Detection **Policies**

Fast Resource Scaling Mechanism

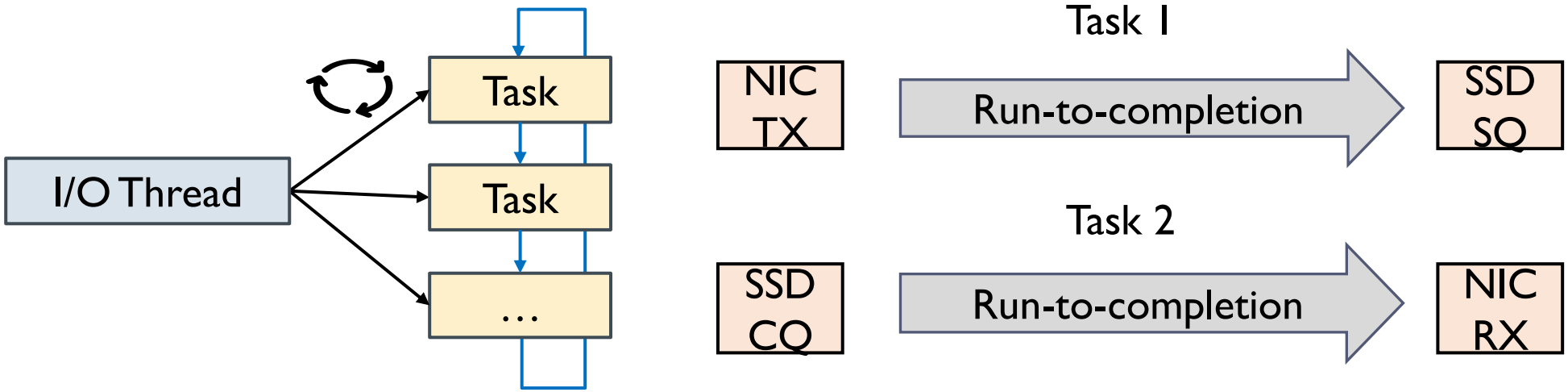
1. Lightweight I/O threads

- **Goal**: move resource fast without overhead

Fast Resource Scaling Mechanism

1. Lightweight I/O threads

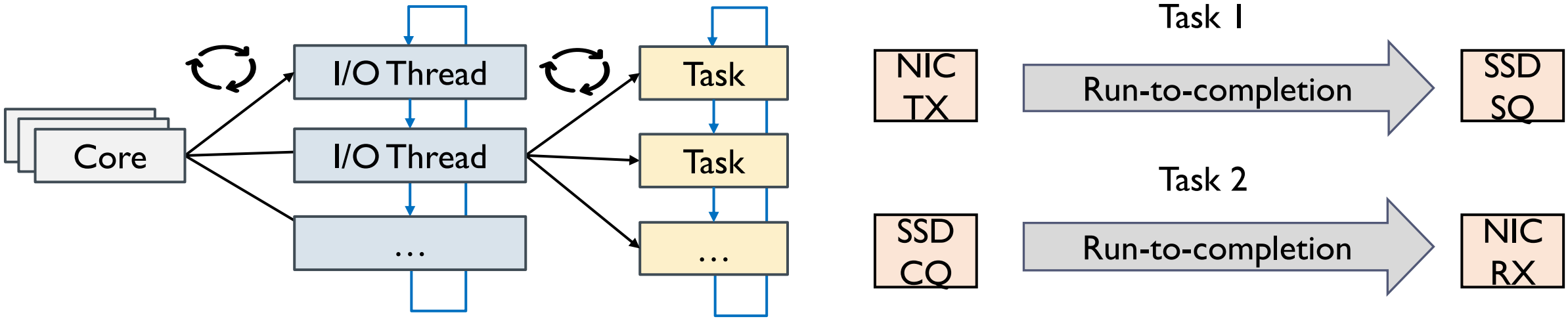
- **Goal**: move resource fast without overhead
- **Thread**: an abstraction of a set of tasks



Fast Resource Scaling Mechanism

1. Lightweight I/O threads

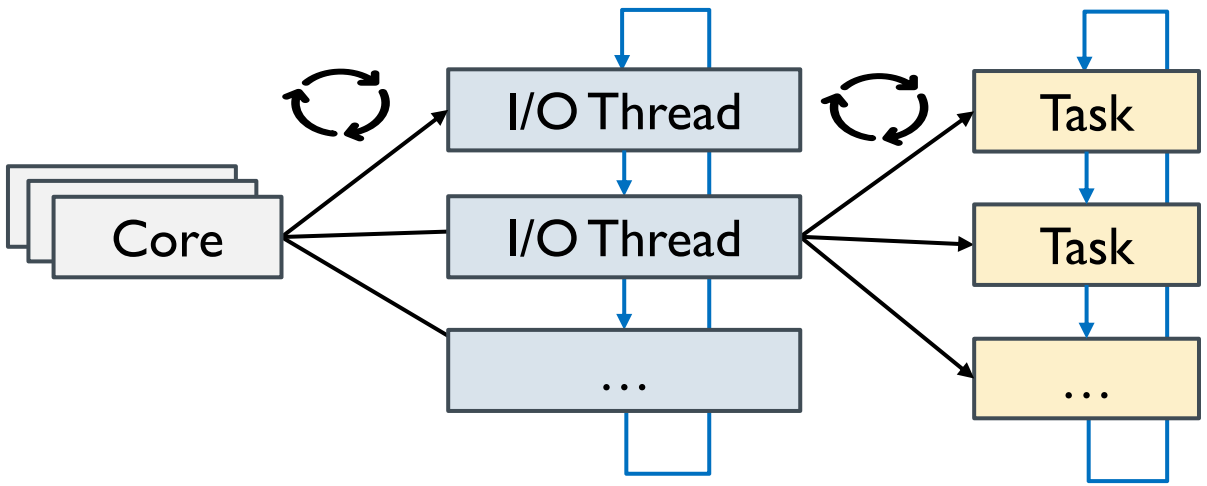
- **Goal**: move resource fast without overhead
- **Thread**: an abstraction of a set of tasks
- **CPU Core**: one core maintains multiple threads through linked list

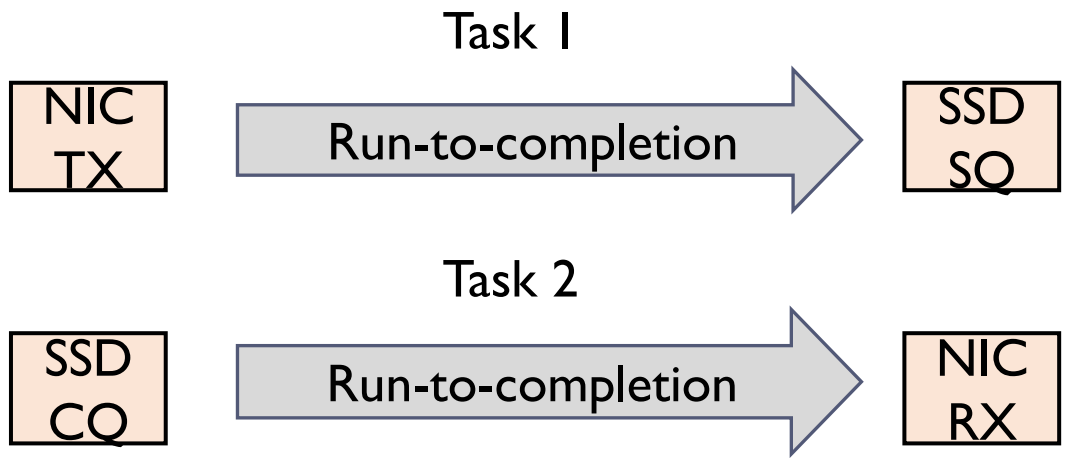


Fast Resource Scaling Mechanism

1. Lightweight I/O threads

- **Goal**: move resource fast without overhead
- **Thread**: an abstraction of a set of tasks
- **CPU Core**: one core maintains multiple threads through linked list
- **Migration**: removing a thread from the core's list and insert to another core's list

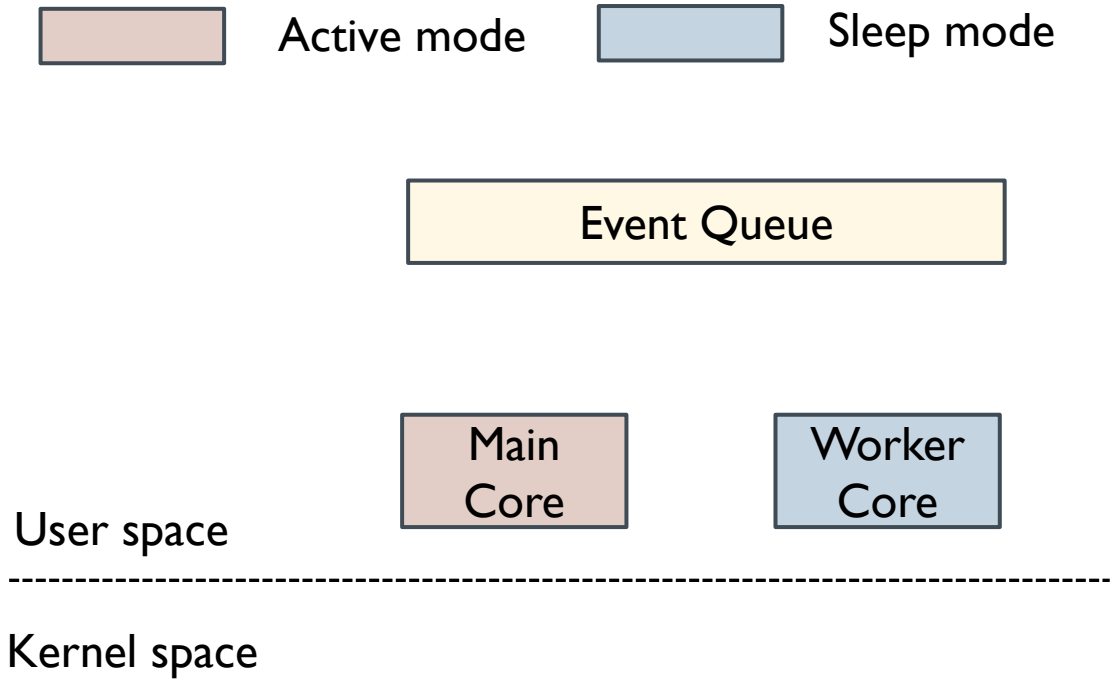




Fast Resource Scaling Mechanism

2. Shallow sleep with instant transition

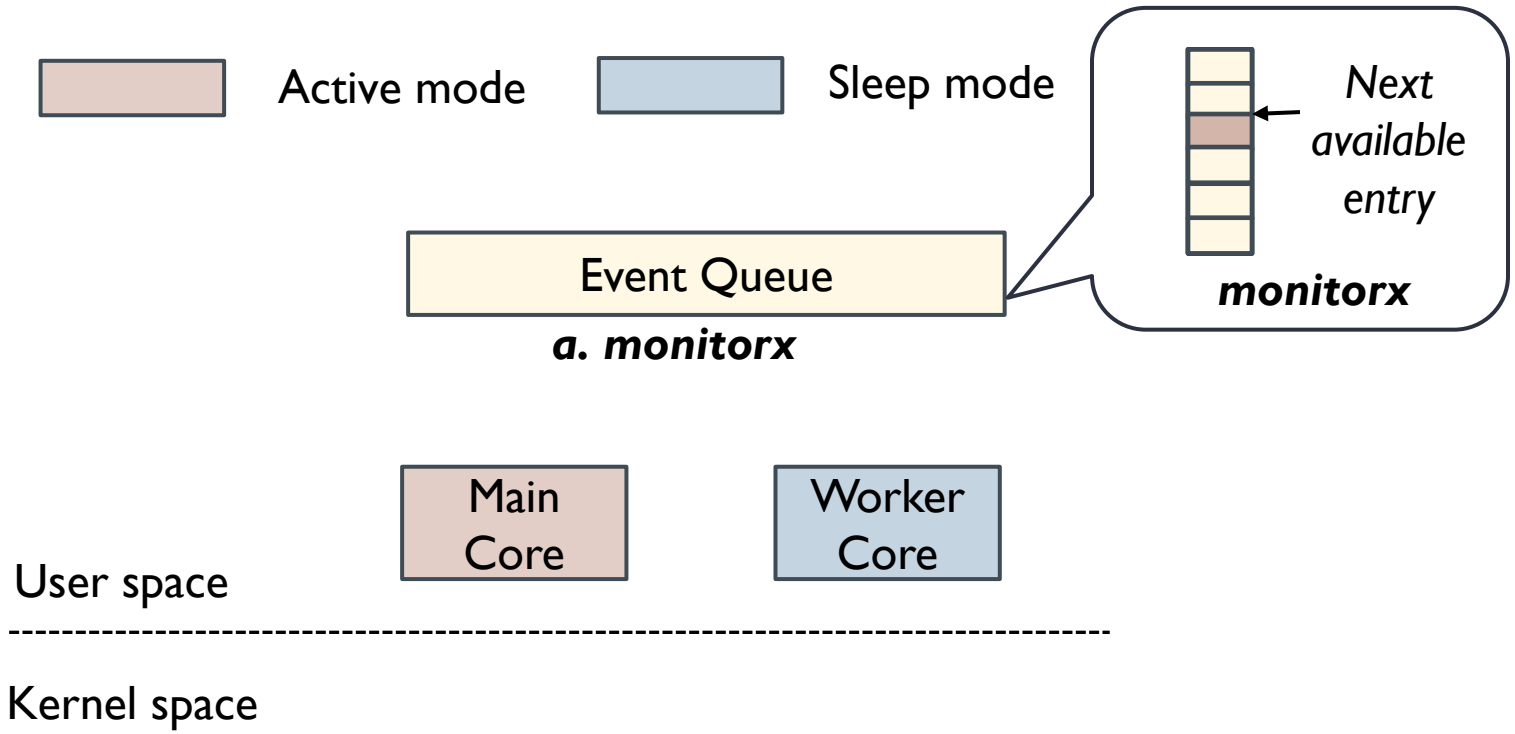
- **Goal**: wake up CPU core rapidly without overhead
- Wake up core through **HW cache-coherency** instead of **SW interrupts**



Fast Resource Scaling Mechanism

2. Shallow sleep with instant transition

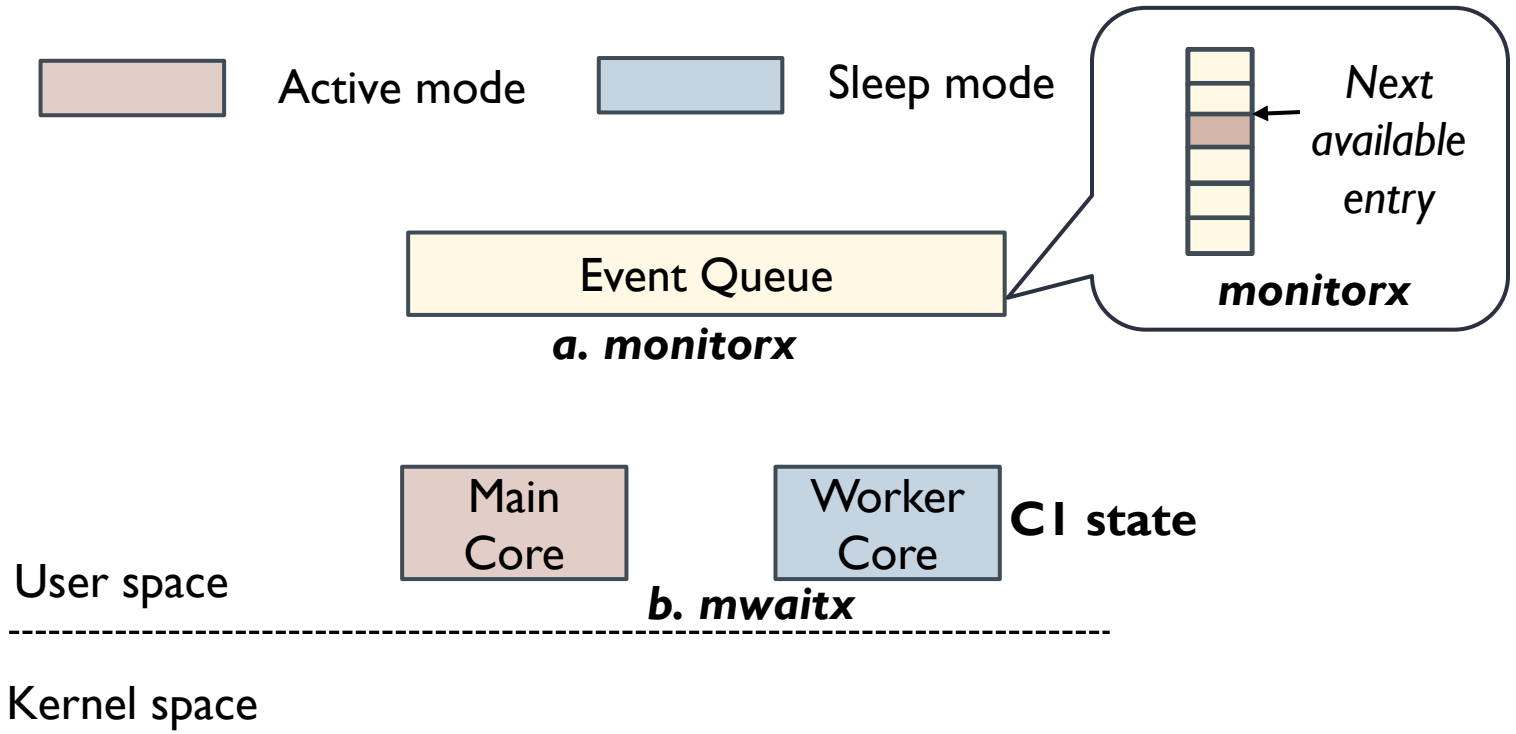
- **Goal**: wake up CPU core rapidly without overhead
- Wake up core through **HW cache-coherency** instead of **SW interrupts**



Fast Resource Scaling Mechanism

2. Shallow sleep with instant transition

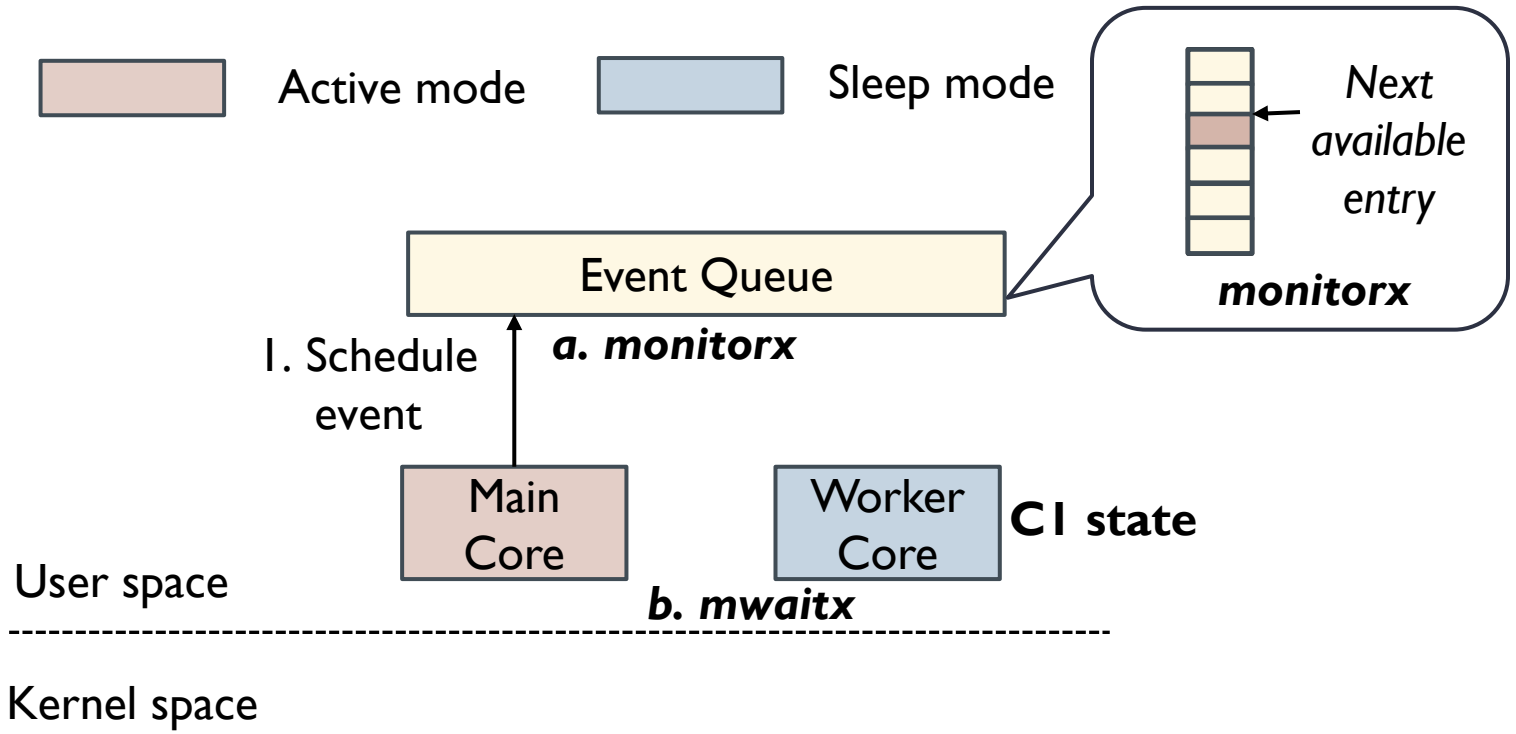
- **Goal**: wake up CPU core rapidly without overhead
- Wake up core through **HW cache-coherency** instead of **SW interrupts**



Fast Resource Scaling Mechanism

2. Shallow sleep with instant transition

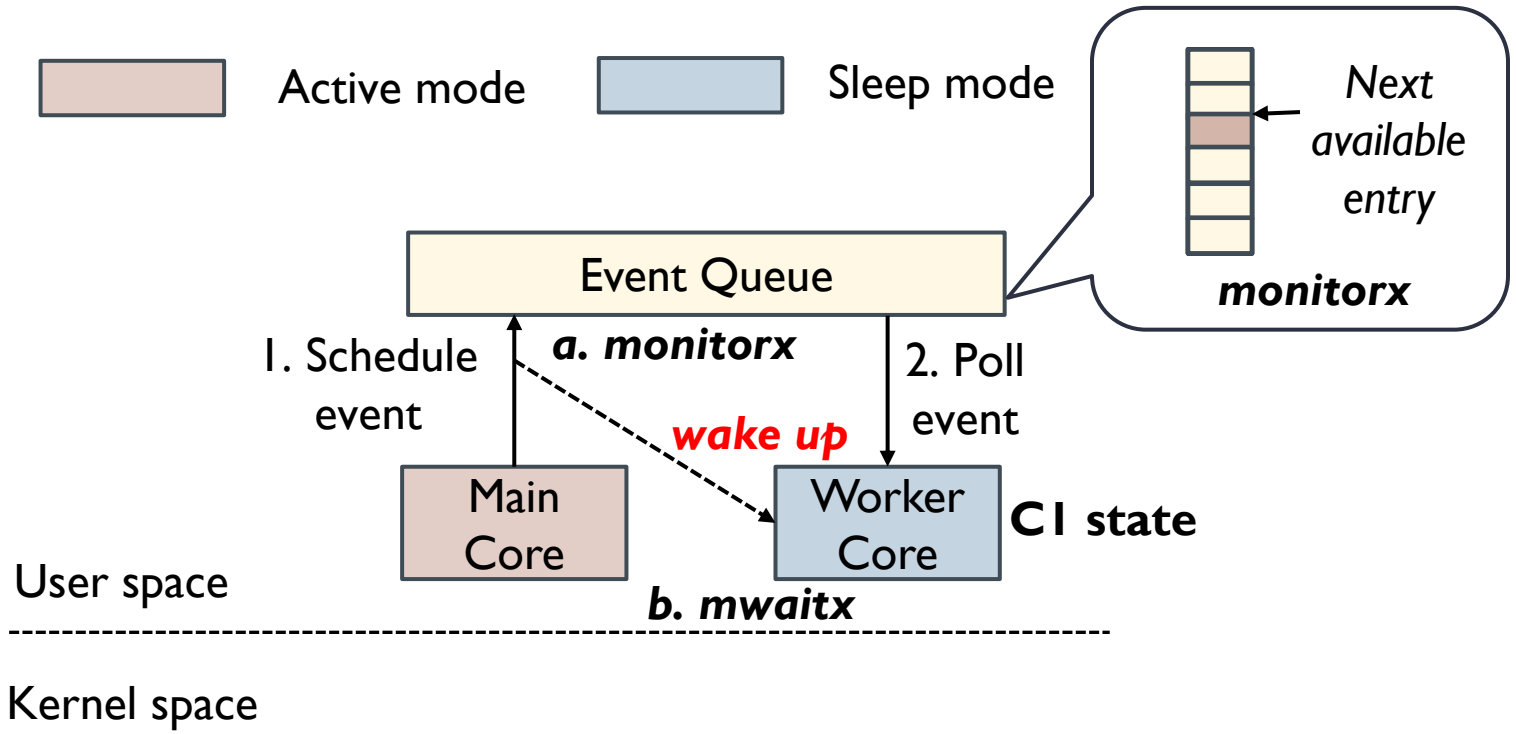
- **Goal**: wake up CPU core rapidly without overhead
- Wake up core through **HW cache-coherency** instead of **SW interrupts**



Fast Resource Scaling Mechanism

2. Shallow sleep with instant transition

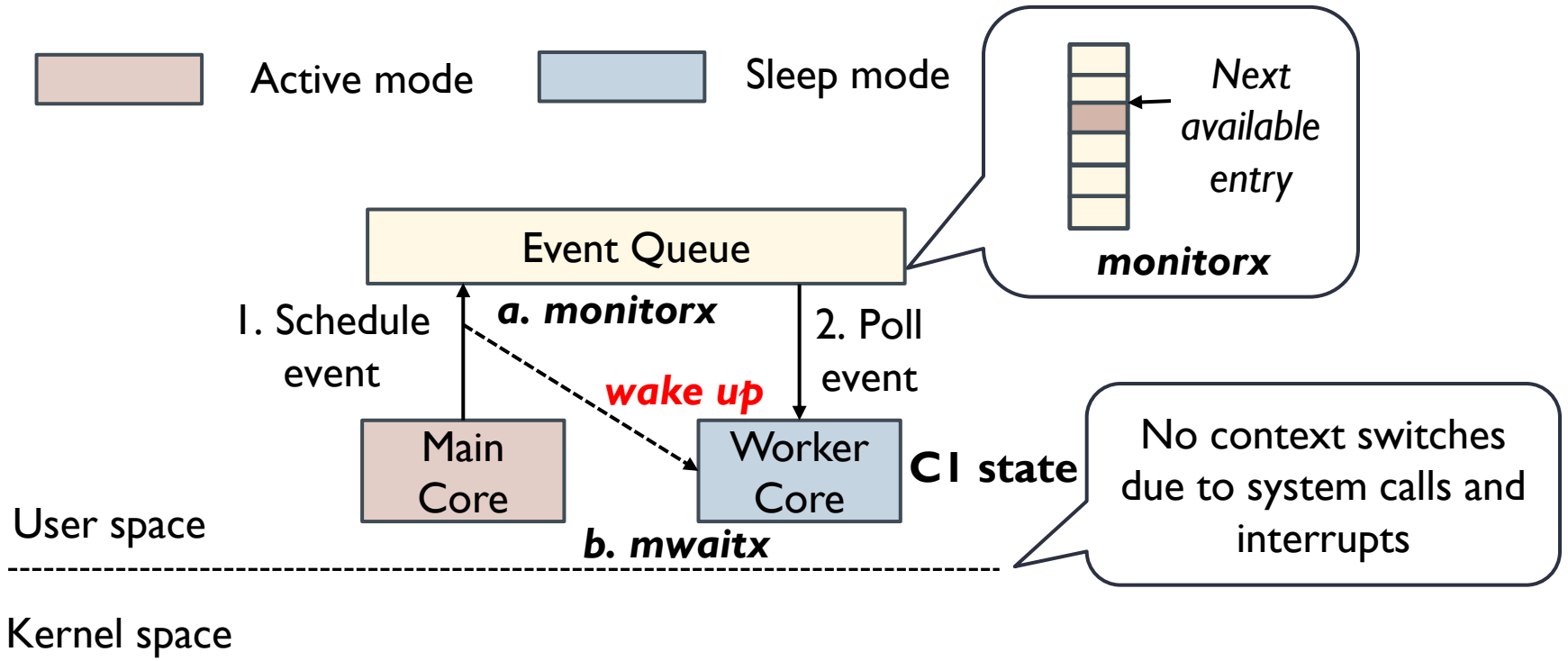
- **Goal**: wake up CPU core rapidly without overhead
- Wake up core through **HW cache-coherency** instead of **SW interrupts**



Fast Resource Scaling Mechanism

2. Shallow sleep with instant transition

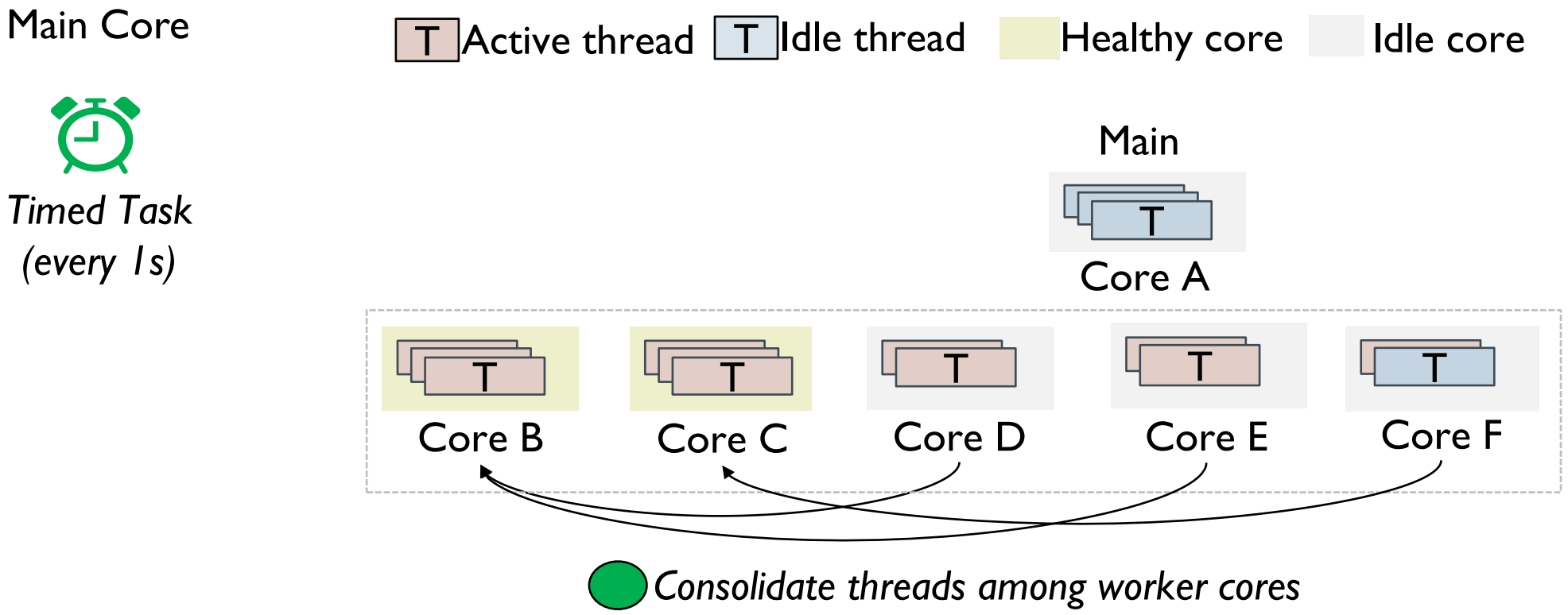
- **Goal**: wake up CPU core rapidly without overhead
- Wake up core through **HW cache-coherency** instead of **SW interrupts**



Scheduling Flow Overview

Two granularities of scheduling

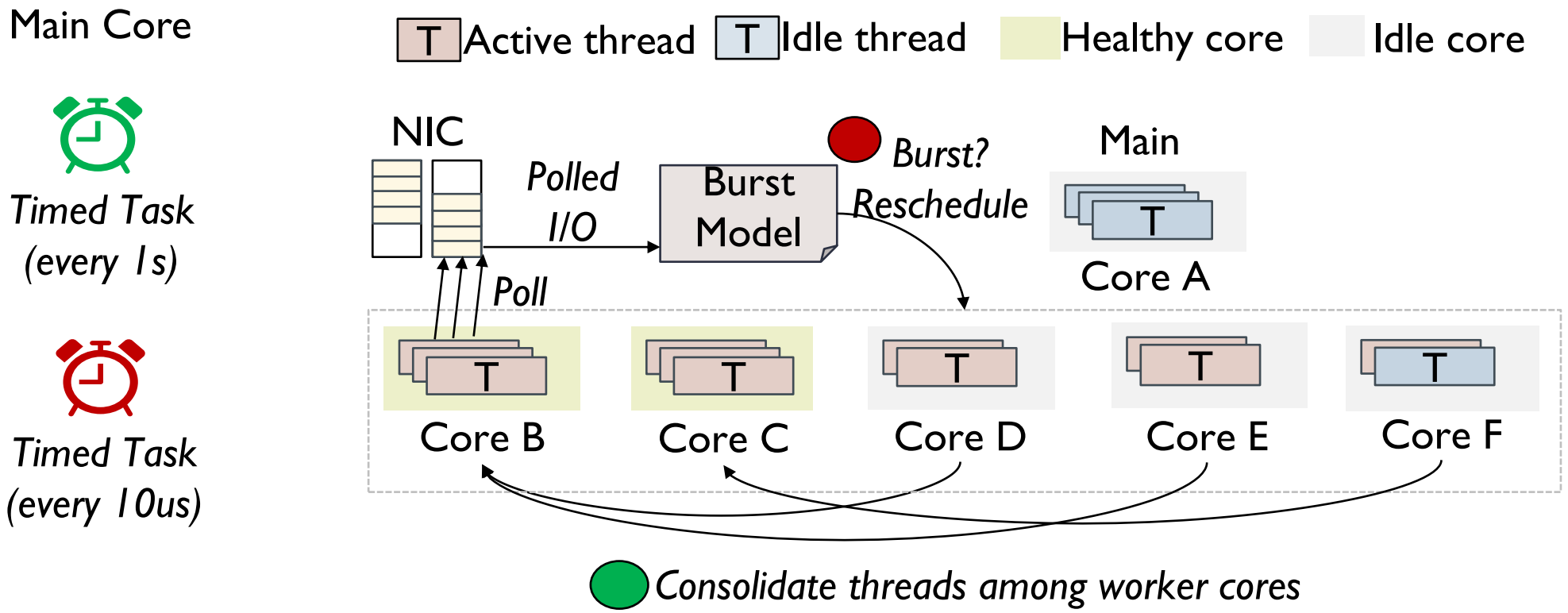
- Coarse-grained (1s): monitor resource wastage for power saving



Scheduling Flow Overview

Two granularities of scheduling

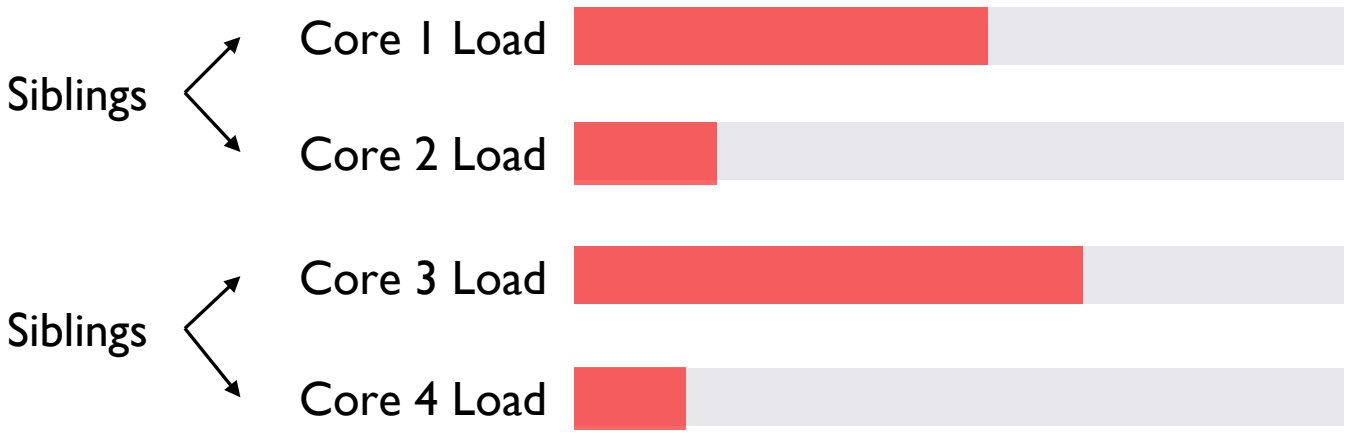
- Coarse-grained (1s): monitor resource wastage for power saving
- Fined-grained (10us): detect I/O bursts for performance



Monitoring Resource Wastage

Goal: let more (physical) cores to sleep and thus save power

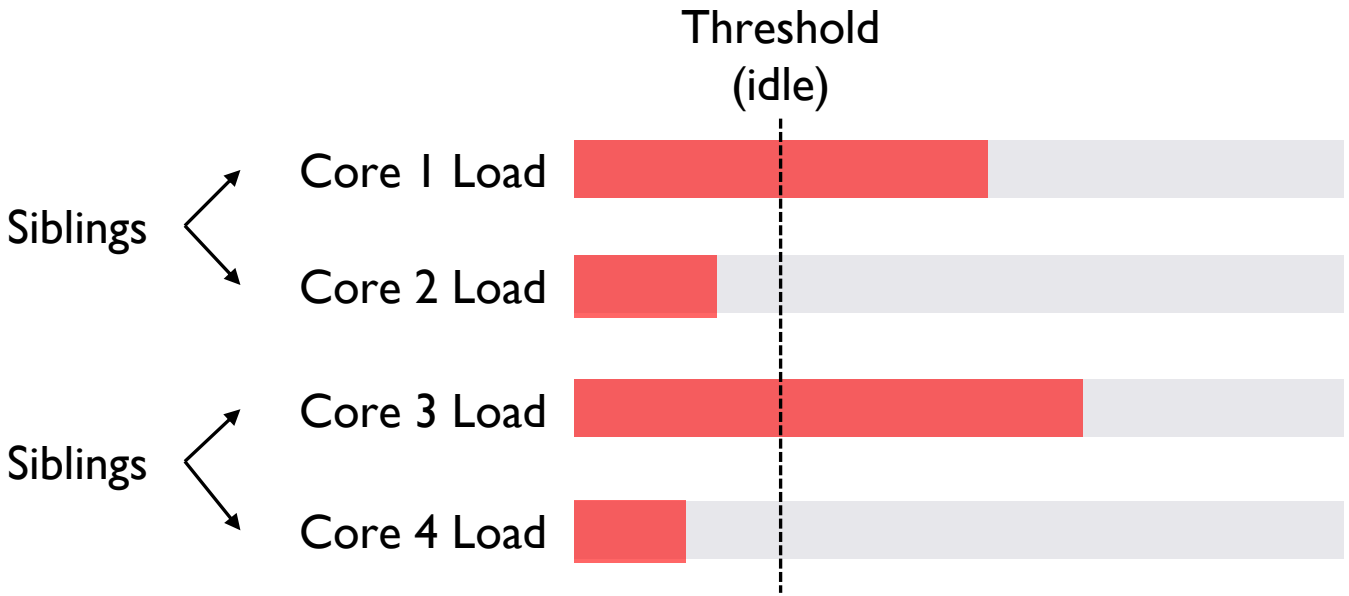
1. Consolidate threads of idle cores into healthy cores



Monitoring Resource Wastage

Goal: let more (physical) cores to sleep and thus save power

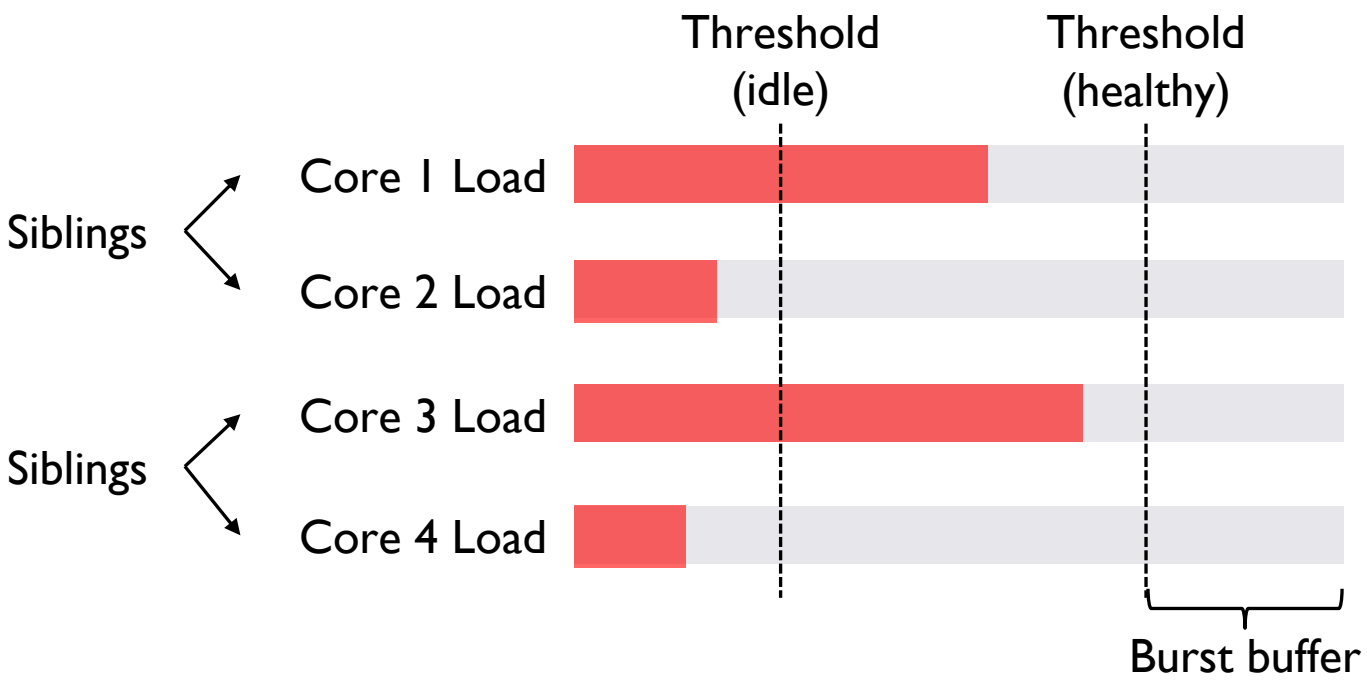
1. Consolidate threads of idle cores into healthy cores



Monitoring Resource Wastage

Goal: let more (physical) cores to sleep and thus save power

1. Consolidate threads of idle cores into healthy cores

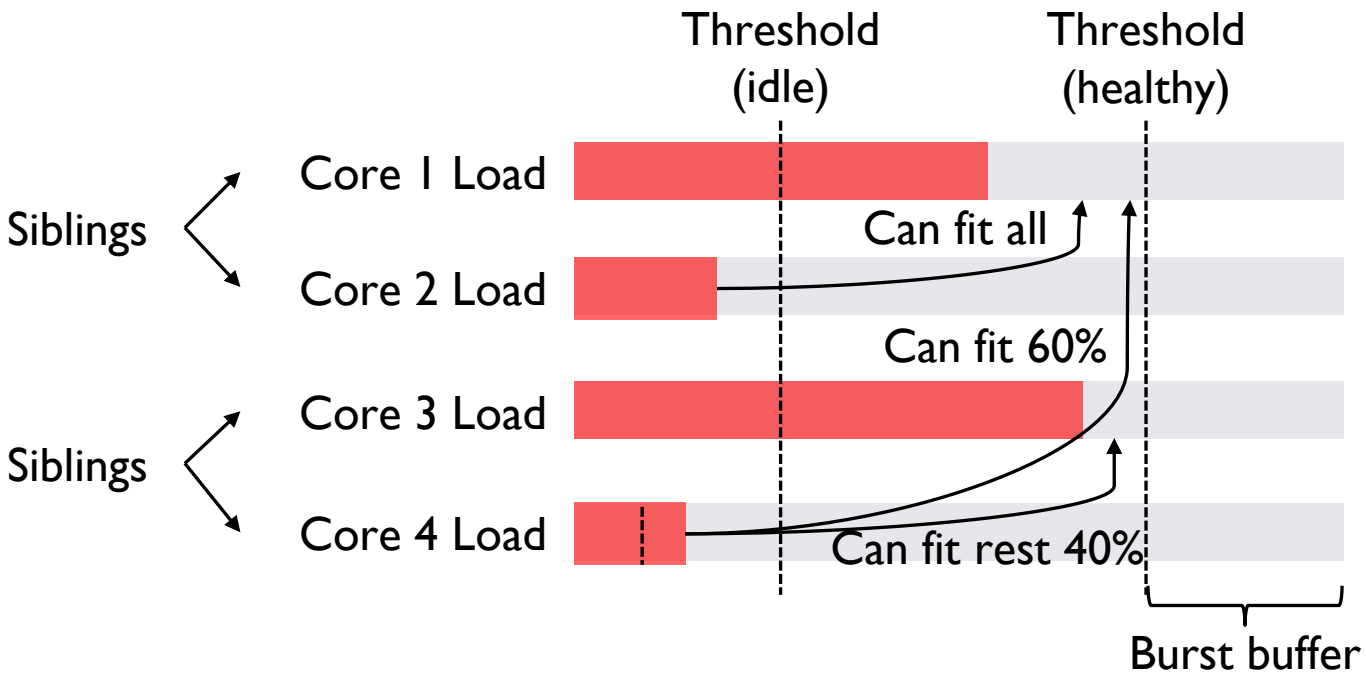


Reserve burst buffer (CPU cycles) shared by all threads on the same core for handling bursts

Monitoring Resource Wastage

Goal: let more (physical) cores to sleep and thus save power

1. Consolidate threads of idle cores into healthy cores



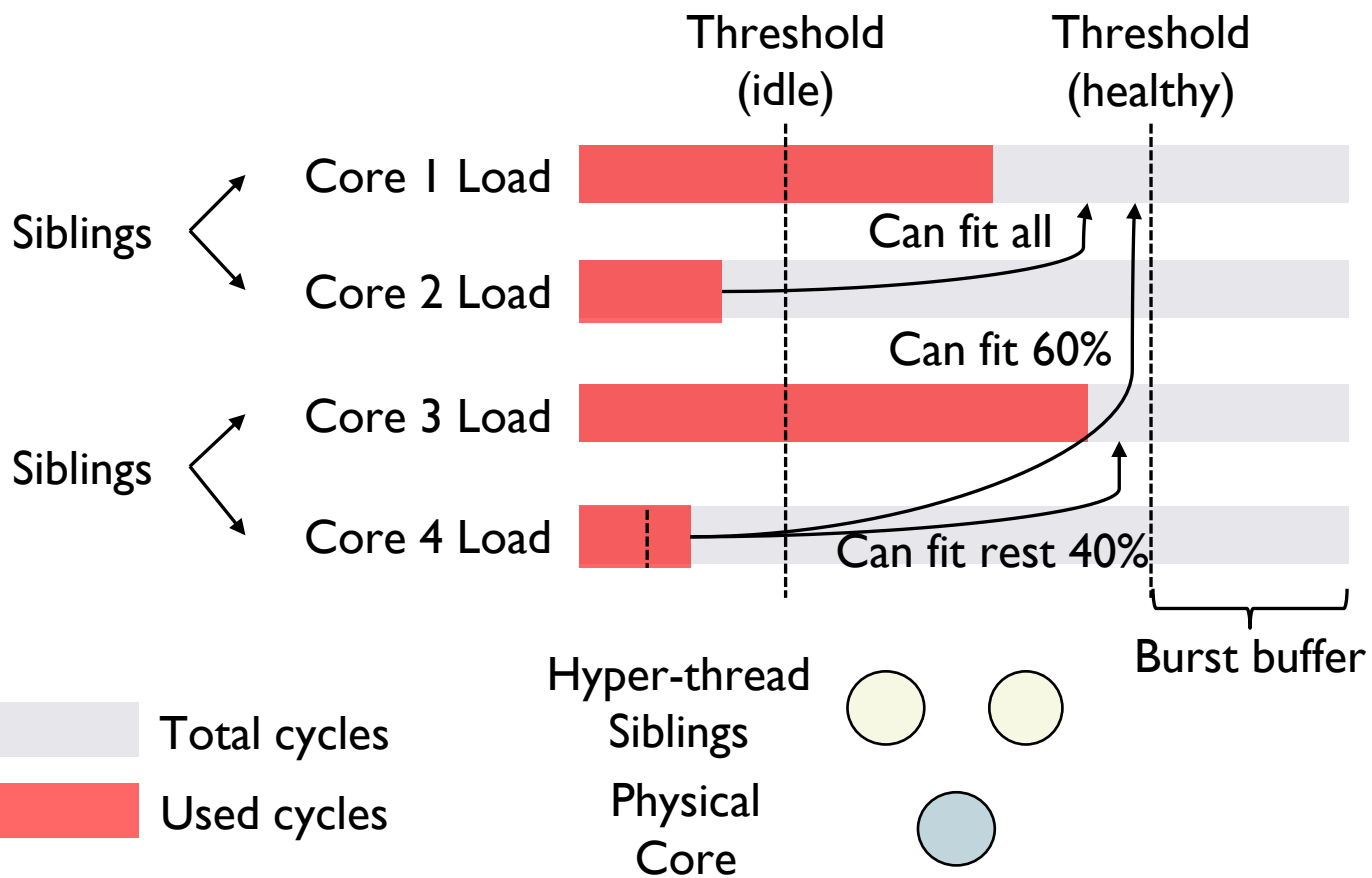
Reserve burst buffer (CPU cycles) shared by all threads on the same core for handling bursts

Check whether core can fit thread based on thread load and core load

Monitoring Resource Wastage

Goal: let more (physical) cores to sleep and thus save power

1. Consolidate threads of idle cores into healthy cores
2. **Schedule sibling cores to sleep together**



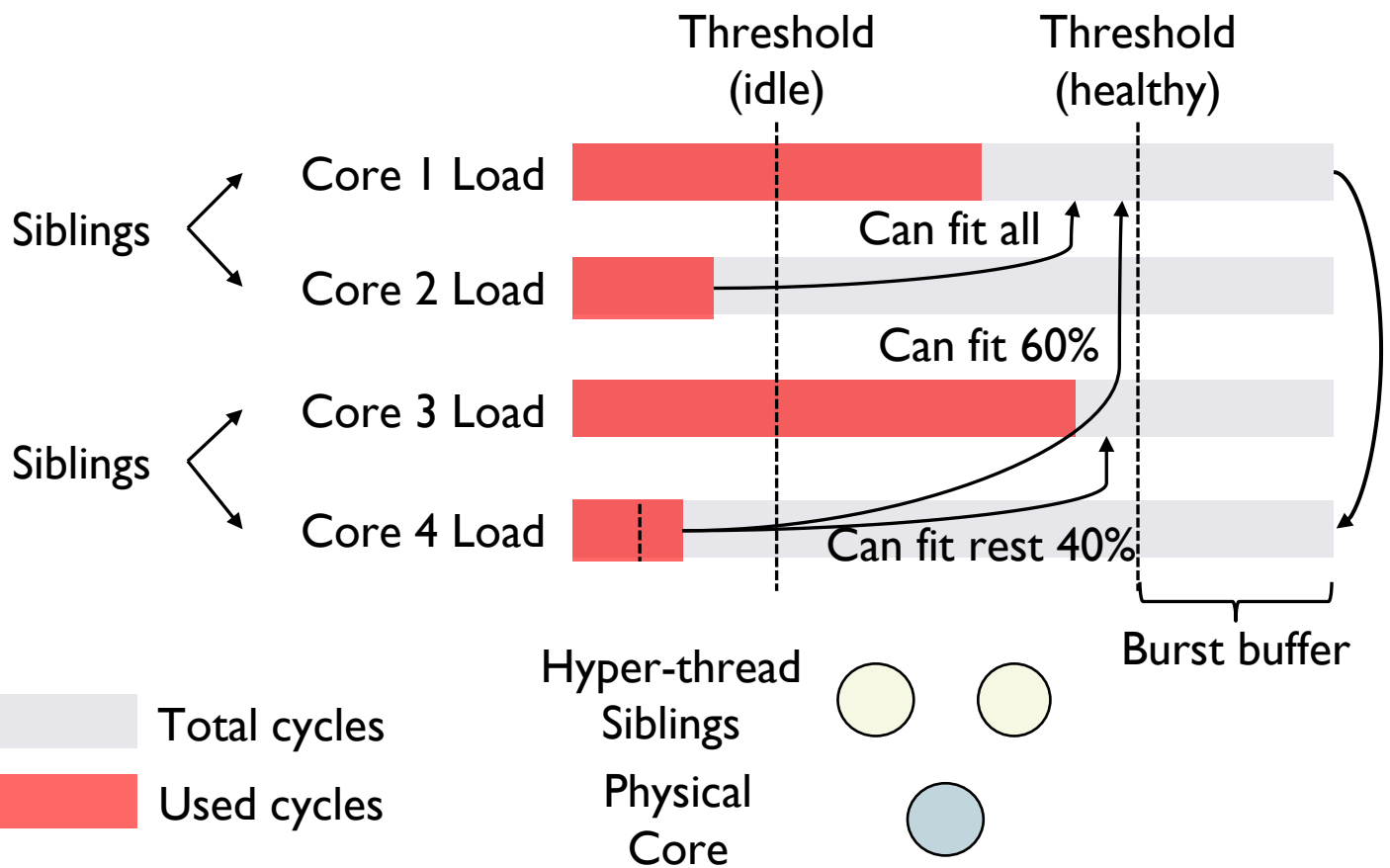
Reserve burst buffer (CPU cycles) shared by all threads on the same core for handling bursts

Check whether core can fit thread based on thread load and core load

Monitoring Resource Wastage

Goal: let more (physical) cores to sleep and thus save power

1. Consolidate threads of idle cores into healthy cores
2. **Schedule sibling cores to sleep together**



Reserve burst buffer (CPU cycles) shared by all threads on the same core for handling bursts

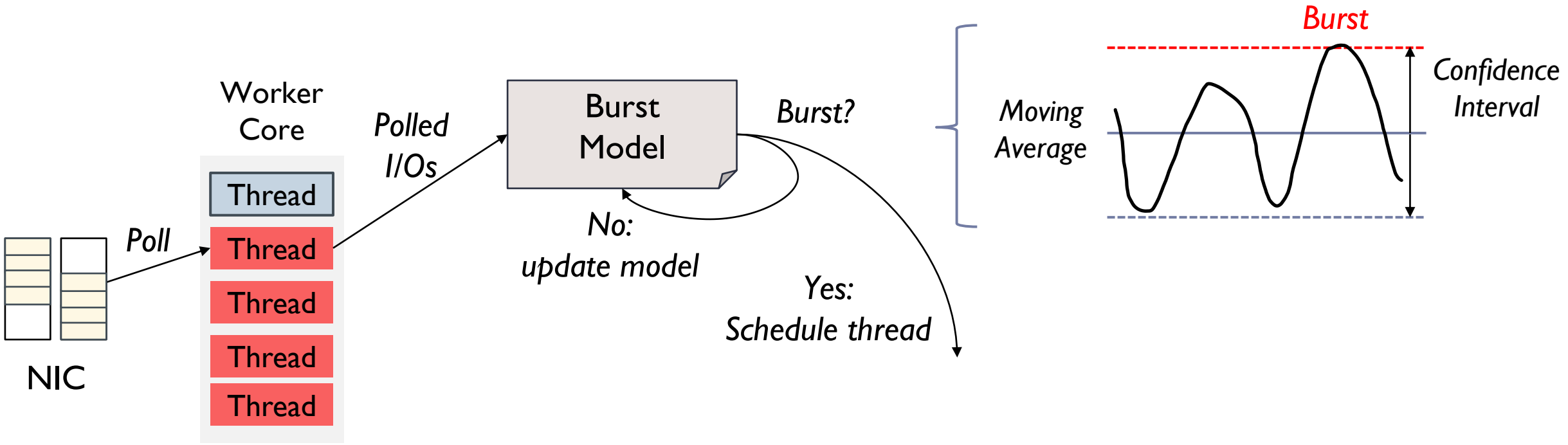
Check whether core can fit thread based on thread load and core load

Move threads of an active core to another free core if its sibling is idle and ready to sleep

Detecting I/O Bursts

Goal: scale compute resources timely for performance

1. Spread out threads that require more compute resource

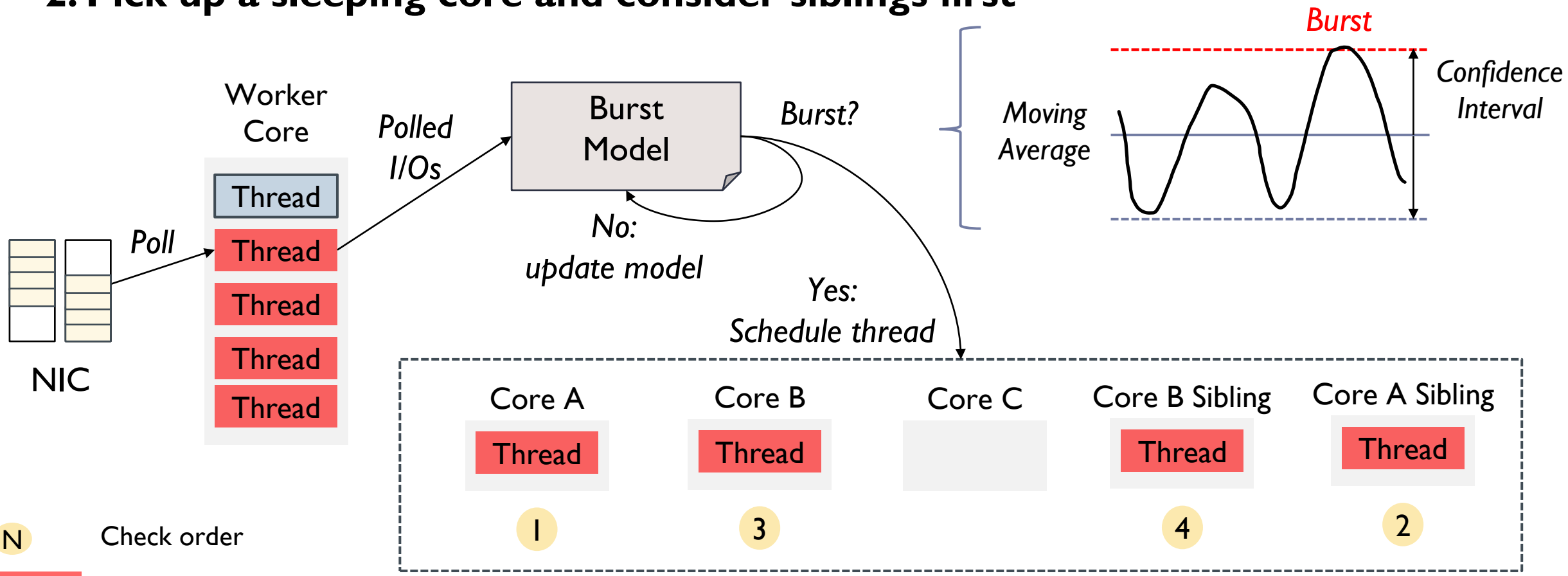


Thread Thread to be scheduled

Detecting I/O Bursts

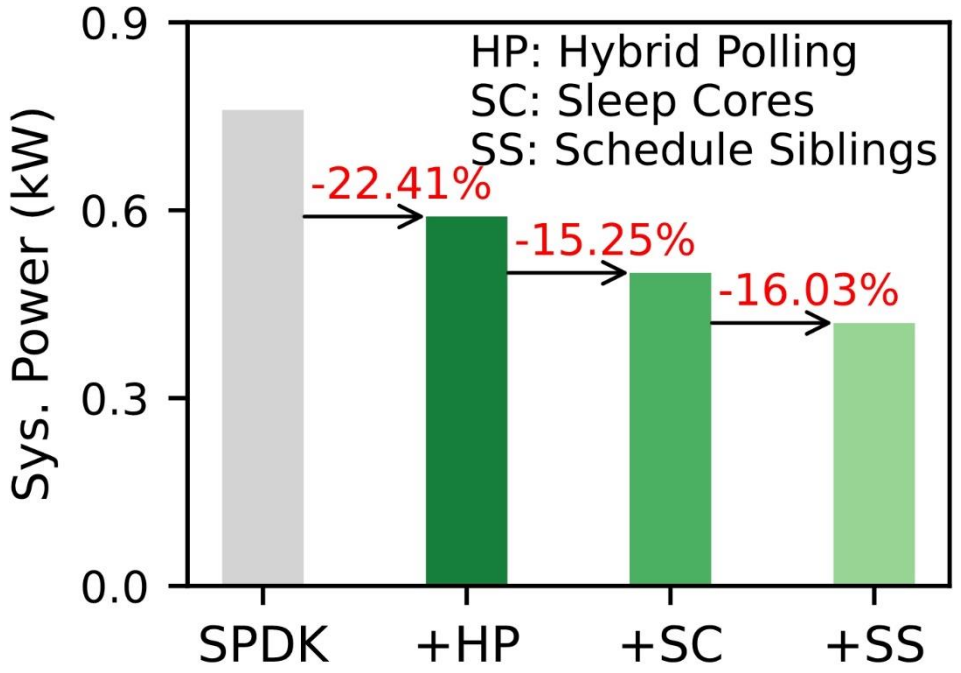
Goal: scale compute resources timely for performance

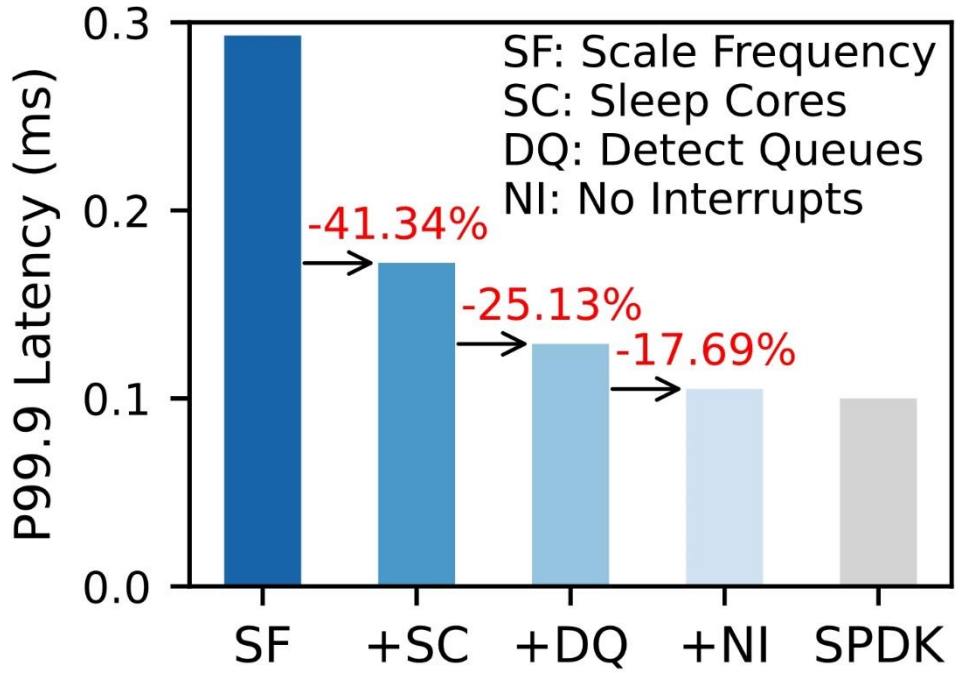
1. Spread out threads that require more compute resource
2. **Pick up a sleeping core and consider siblings first**



Evaluation

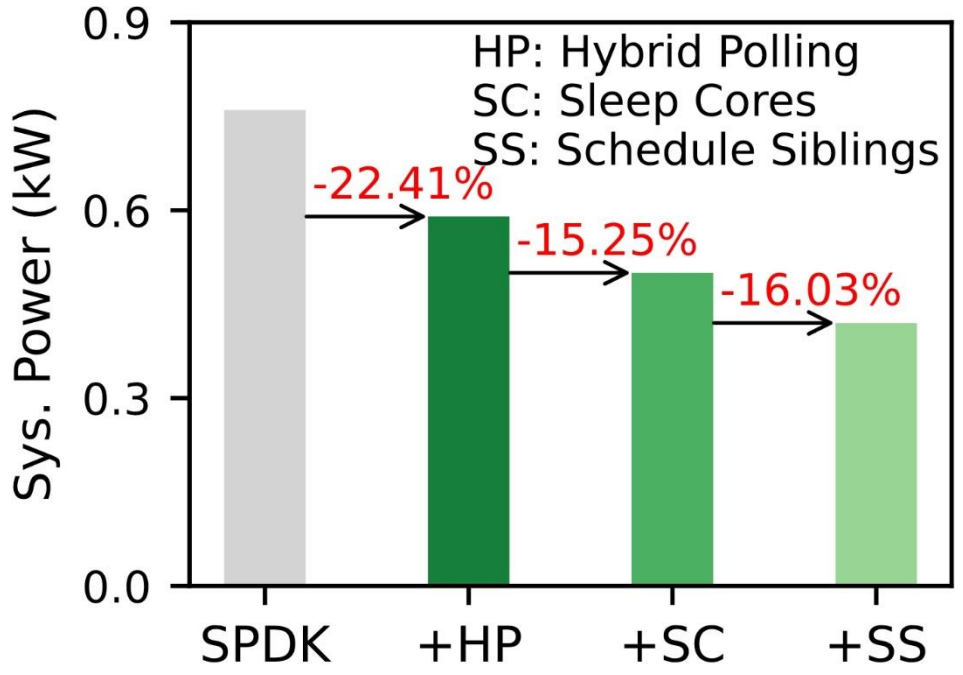
Ablation study – under bursty workloads

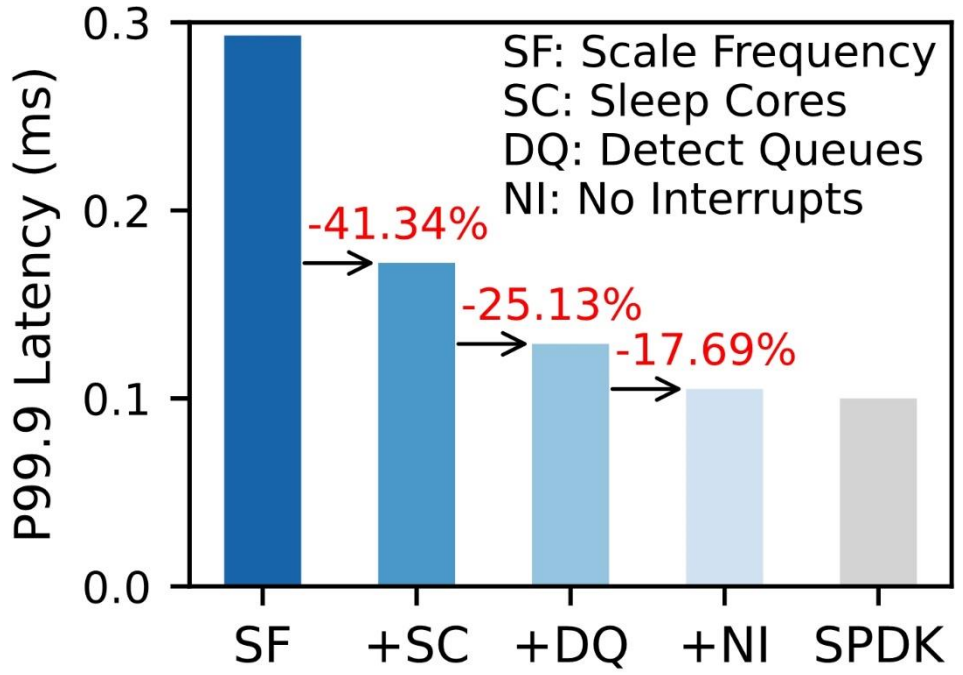




Evaluation

Ablation study – under bursty workloads

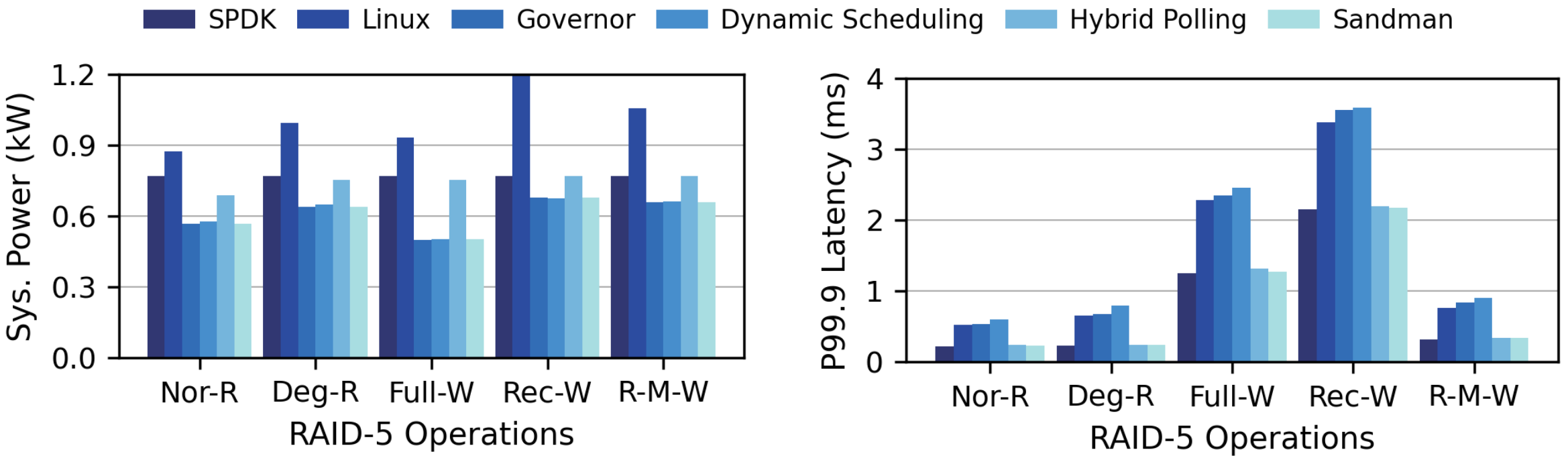




Putting everything together, Sandman achieves **latency comparable to the busy-polling stack**, while significantly **reducing power consumption**.

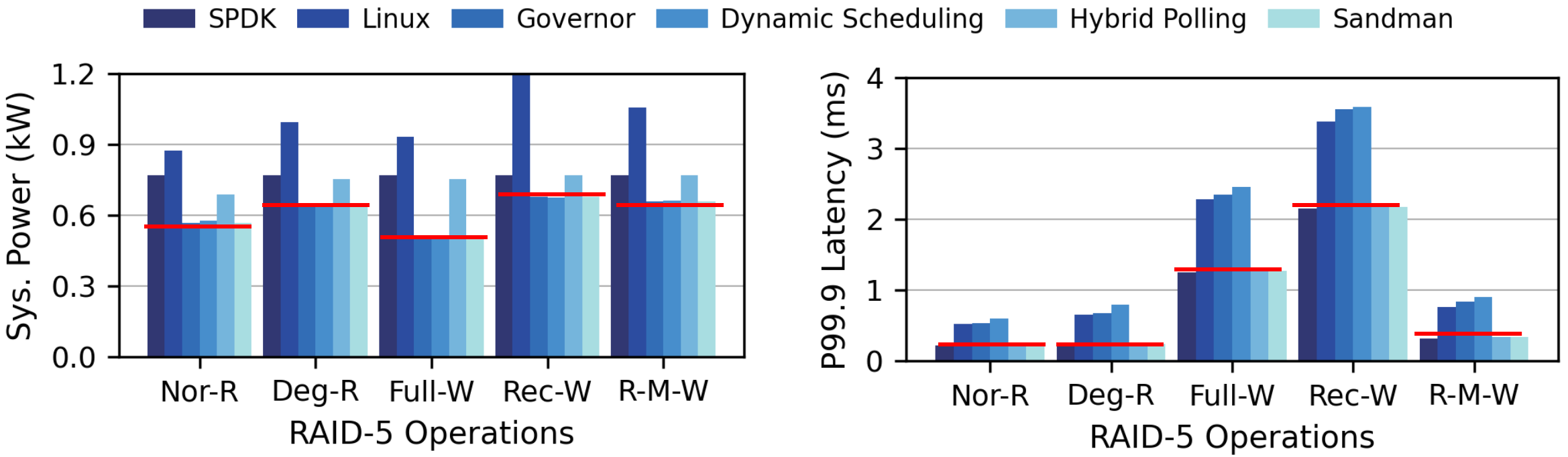
Application Benefits

Flash array configured with RAID-5



Application Benefits

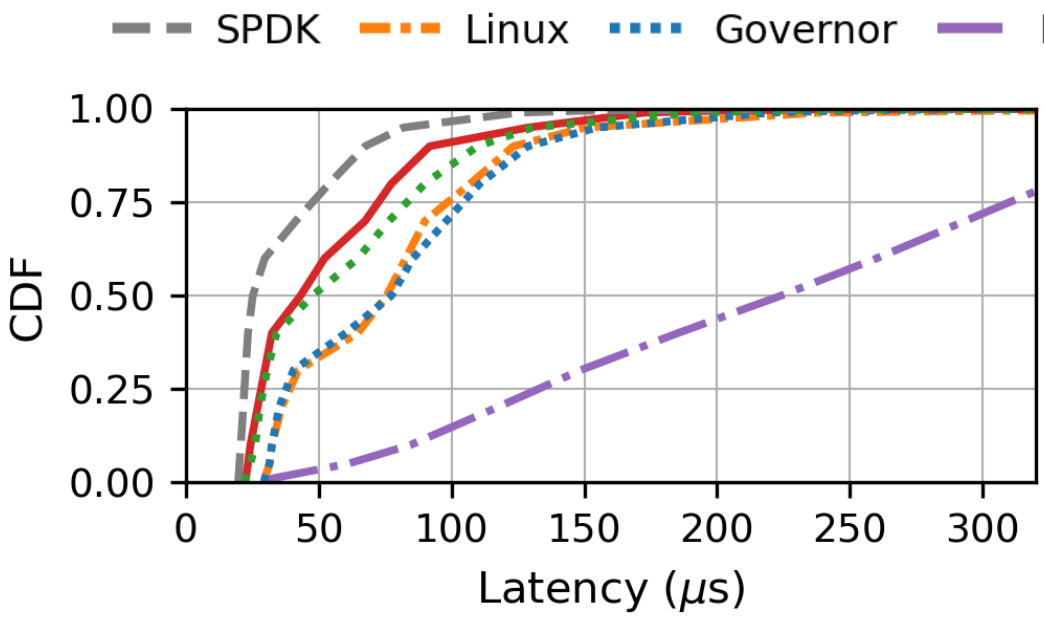
Flash array configured with RAID-5



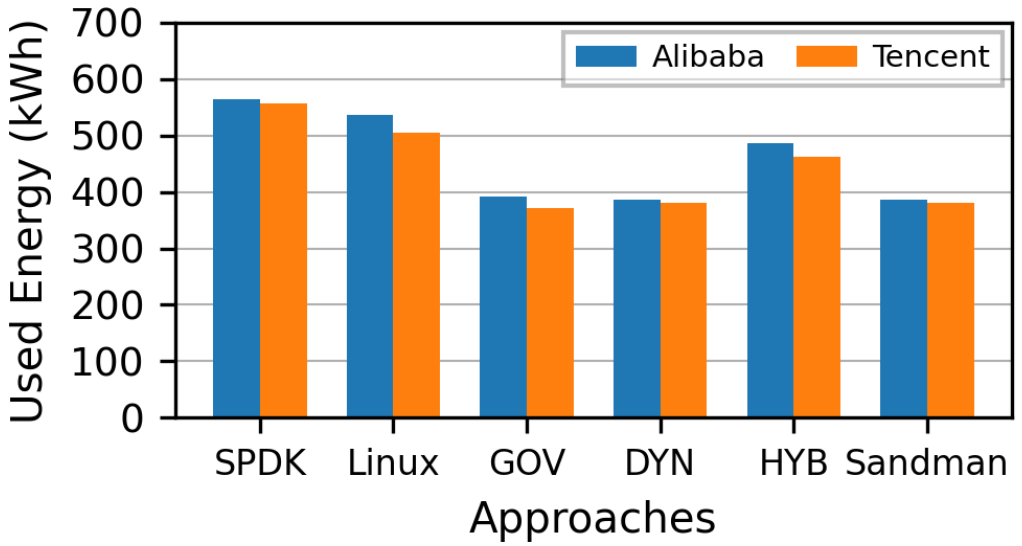
Sandman achieves comparable latency to SPDK but reduces power consumption **up to 39.38%** under flash array with RAID-5

Benefits under Cloud Workloads

Replaying block-level traces from two public cloud vendors



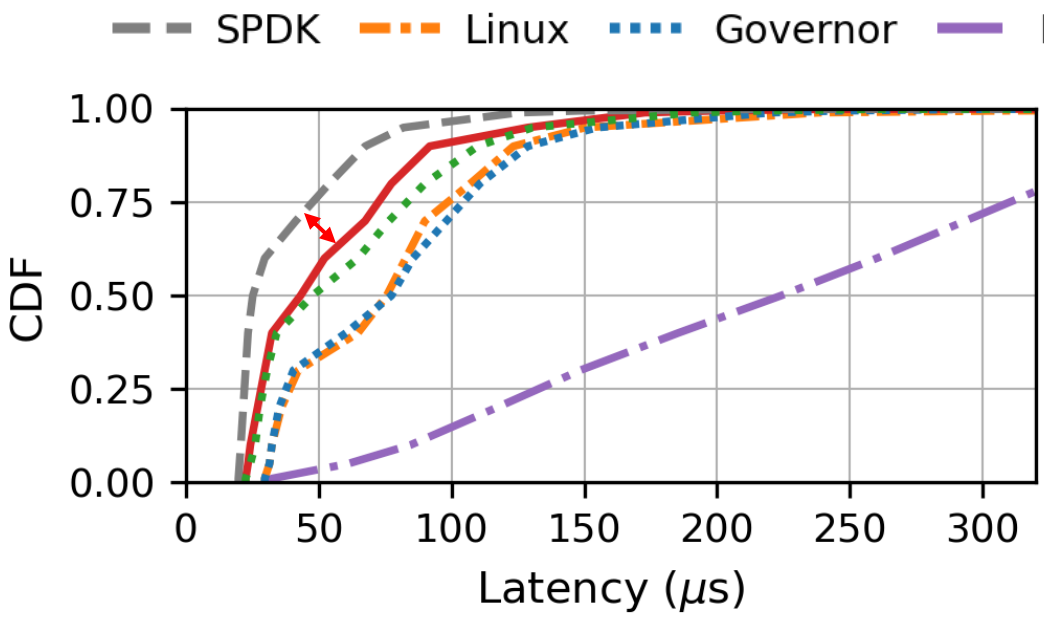
Latency CDF under Alibaba's Trace



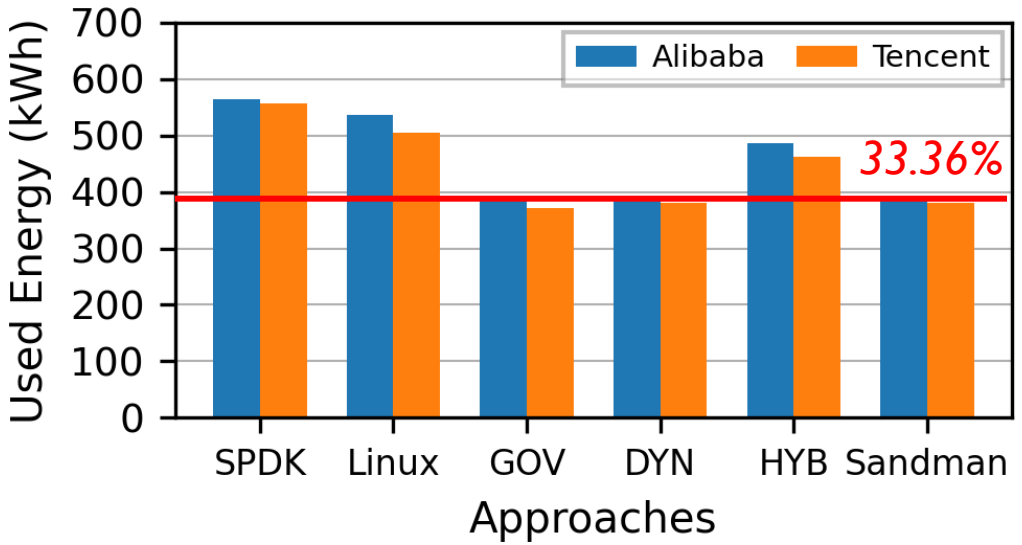
Monthly Energy Consumption

Benefits under Cloud Workloads

Replaying block-level traces from two public cloud vendors



Latency CDF under Alibaba's Trace



Monthly Energy Consumption

Sandman achieves latency close to SPDK while reducing energy consumption by **33.36% vs. SPDK** under cloud workloads

Takeaways

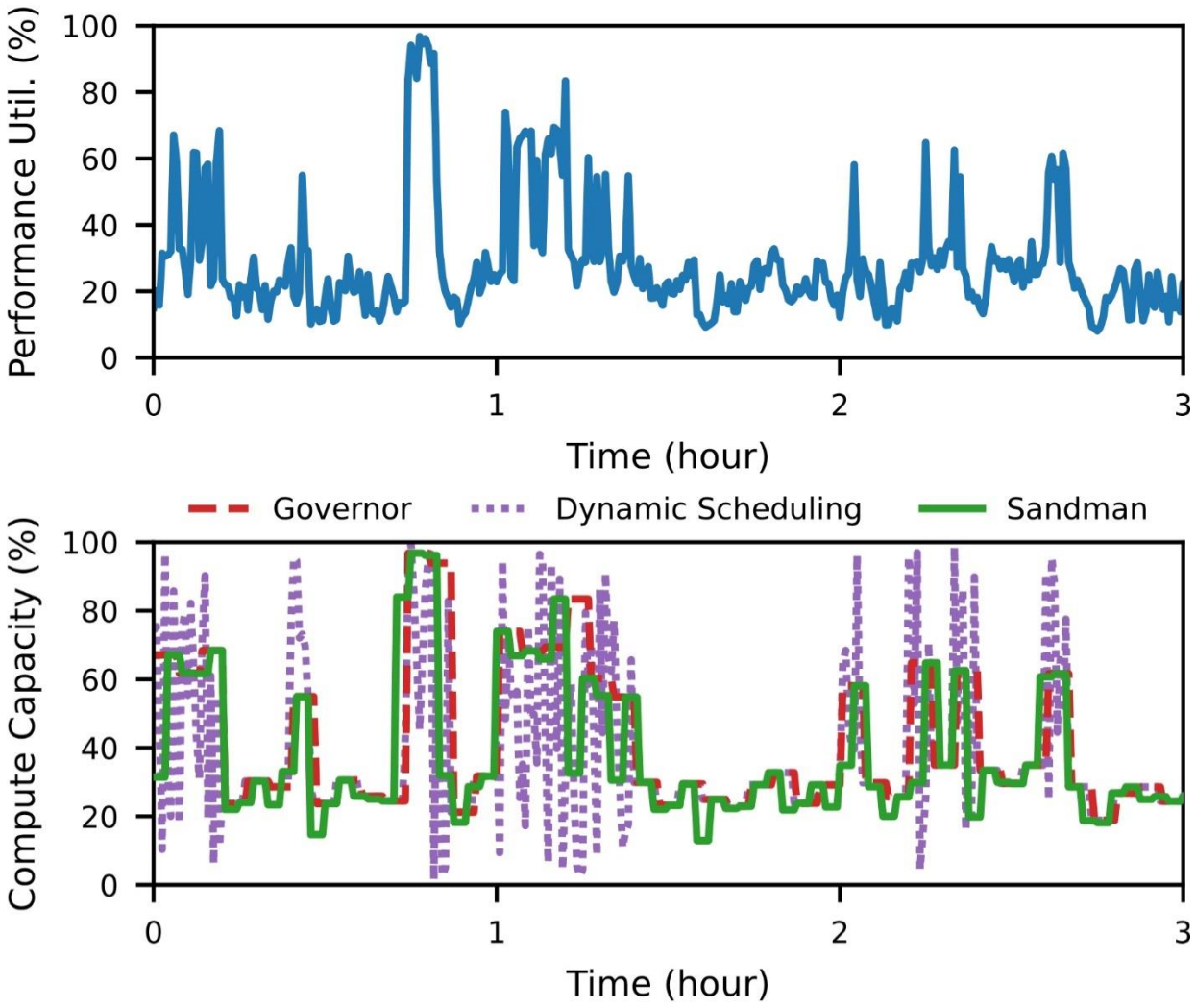
- Storage trends to consume more energy with evolving towards high performance and high density
- Current storage stacks either consume more power or sacrifice performance
- Storage stacks can be both high performance and energy efficiency with a carefully designed scheduler with hardware assistance.

Thank You!

Q & A

More details are in the paper

What about Runtime Behaviors?



Runtime resource allocation under cloud workloads

- Workload varies over time
- Sandman can (almost) follow the workload changes and allocate required resources timely
- The governor and dynamic schedulers shows delay and jitter

What about User Interrupts?

- **User-level interrupts from I/O devices (e.g., NICs and SSDs)**
  - Hardware cannot directly support
  - **Industry**
    - userspace NVMe SSDs -> VFIO -> Kernel eventfd forward to user applications
      - SPDK NVMe-over-TCP
  - **Academia**
    - Userspace NVMe SSDs -> MSI-X interrupts –> remapping to user applications
      - Aeolia (SOSP'25), SPDK+ (HotStorage'25)
    - Modify hardware
      - xUI (ASPLOS'25)
- **User-level Scheduler**
  - Used for inter-processor communication
    - Achieving low-overhead preemption scheduling in network (e.g., head-of-line blocking)
      - Aspen (NSDI'25), Skyloft (SOSP'25), uProcess (SOSP'25)