Stony Brook University

# Scavenger: A Black-Box Batch Workload Resource Manager for Improving Utilization in Cloud Environments

Seyyed **Ahmad** Javadi, Amogha Suresh, Muhammad Wajahat, Anshul Gandhi

PACE
Stony Brook University

November 22, 2019

# Cloud Computing

## Cloud providers

Operate cloud infrastructures

Great budget expenditure for:

- Data center equipment
- Power provisioning


## Tenants

Rent Virtual Machines (VMs)

# Cloud Computing

## Cloud providers

Operate cloud infrastructures

Great budget expenditure for:

- Data center equipment
- Power provisioning

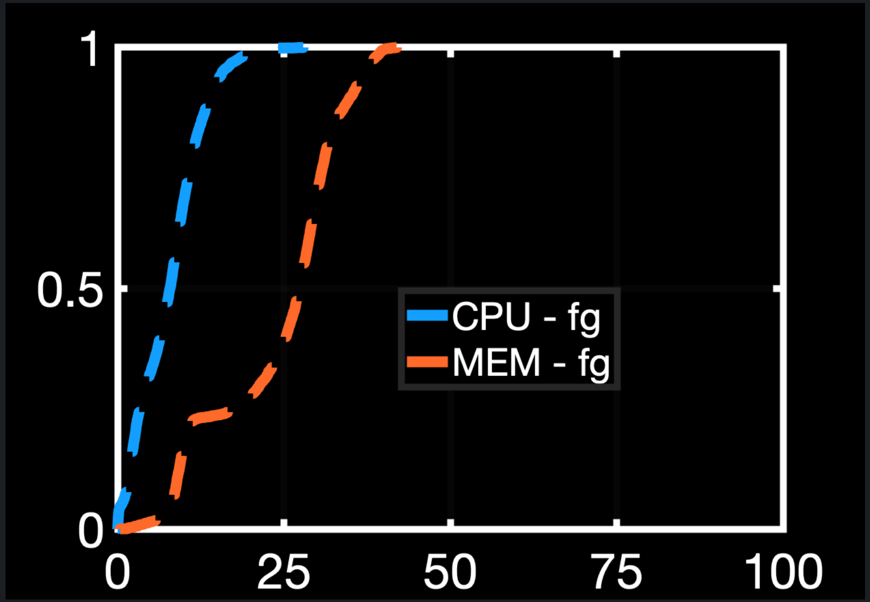

## Tenants

Rent Virtual Machines (VMs)

- ➢ Virtual resources might be provisioned (via tenants) for peak load
- ➢ Tenants' VM placement (via providers) is challenging

# Low Resource Utilization in Cloud Environments

Cumulative probability, F(x)



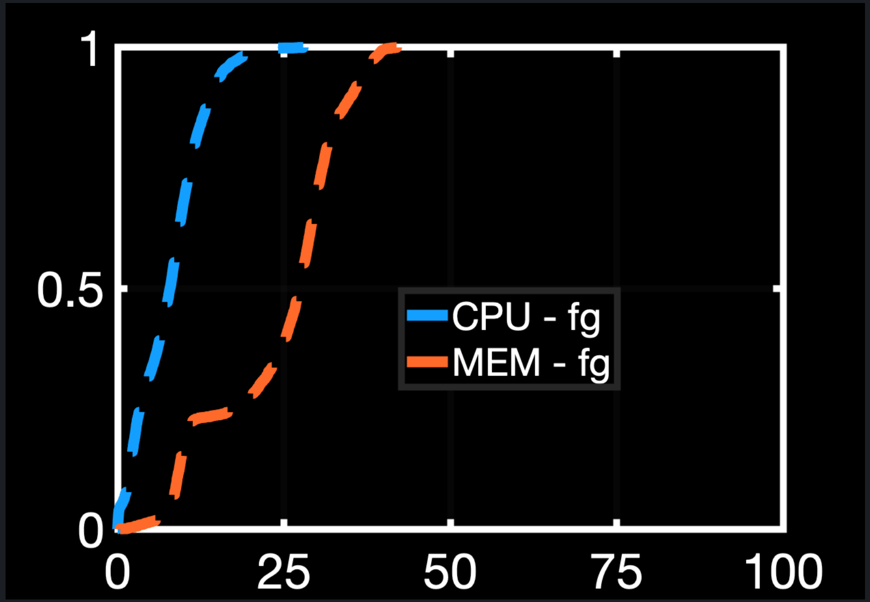

X = Average usage

CDF of average CPU and memory usage, Alibaba cluster trace (2018).

fg = foreground/online workload

# Low Resource Utilization in Cloud Environments

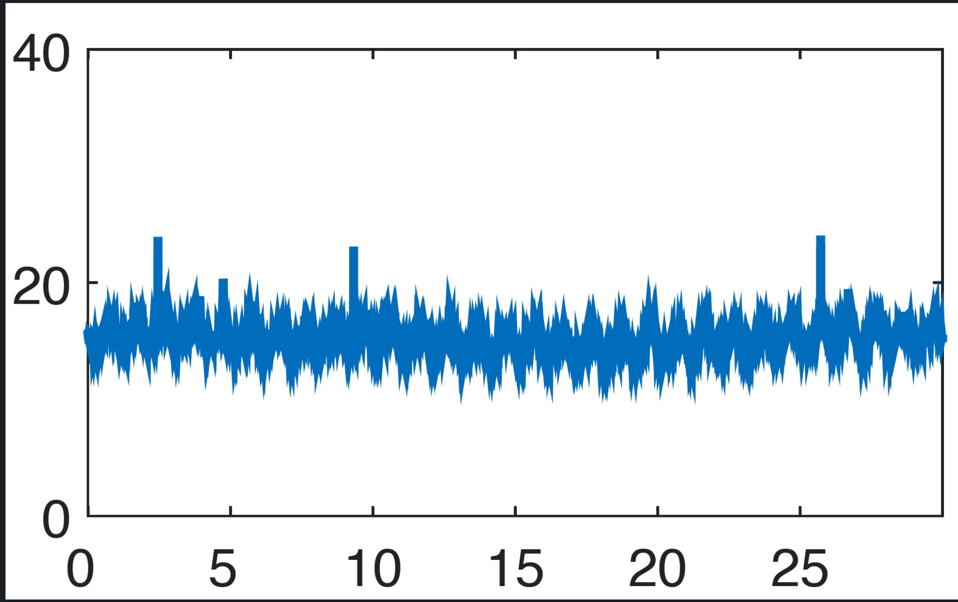
Cumulative probability, F(x)




X = Average usage

CDF of average CPU and memory usage, Alibaba cluster trace (2018).

CPU utilization (%)



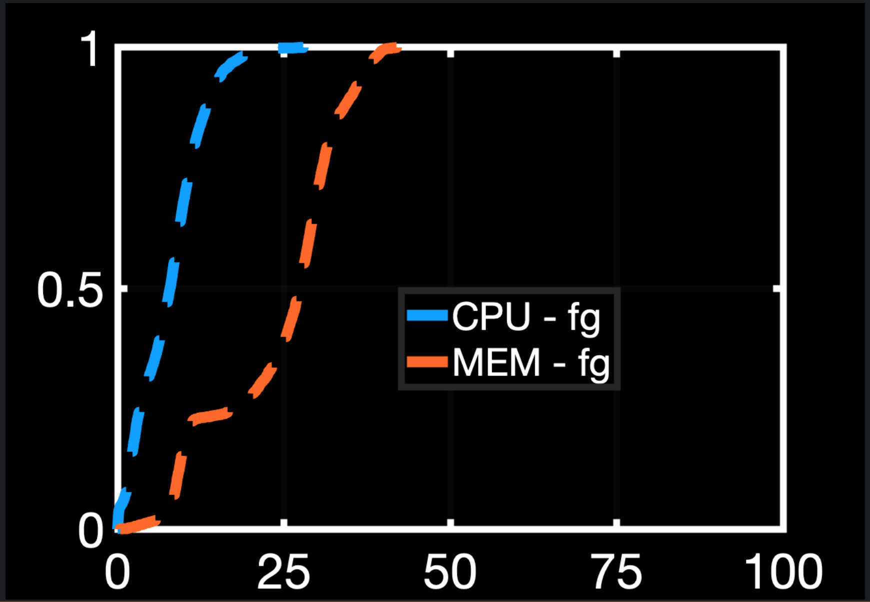

Time (days)

VM-level CPU usage for the Azure trace (2017).

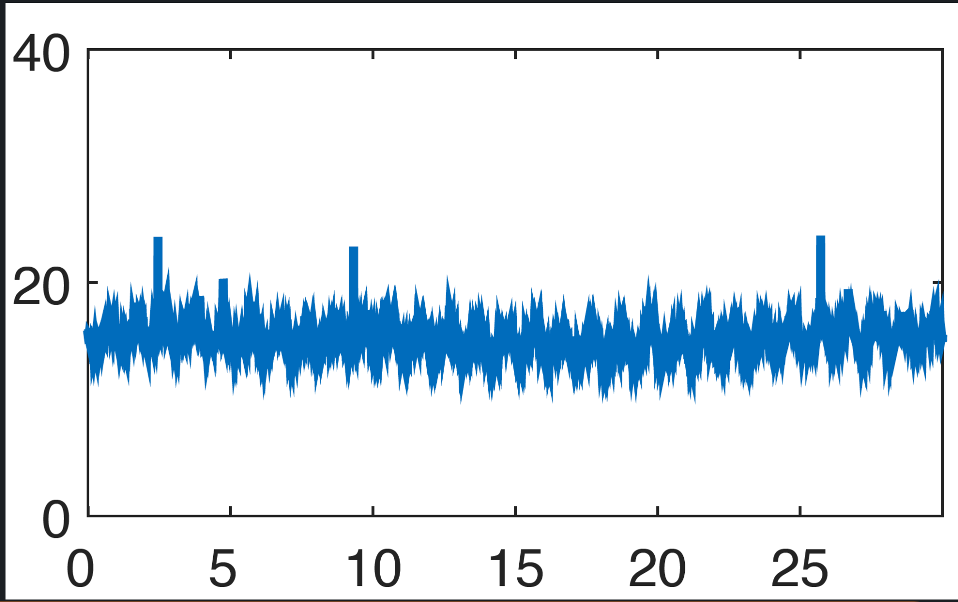
fg = foreground/online workload

# Low Resource Utilization in Cloud Environments

Cumulative probability, F(x)



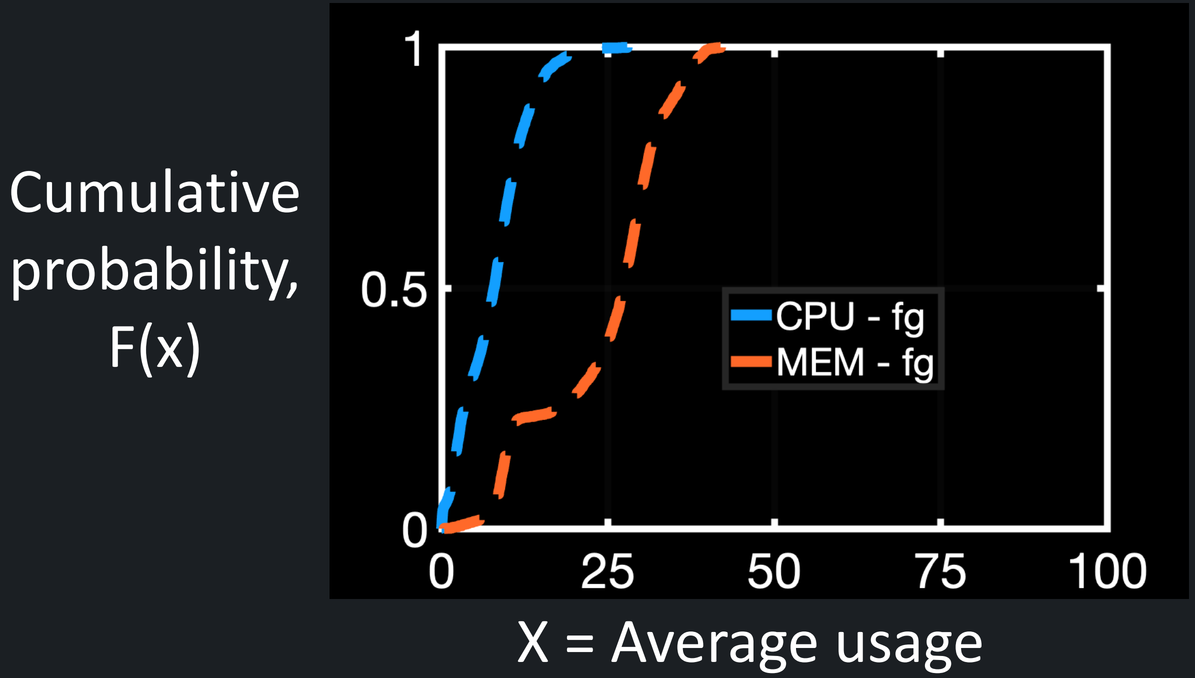

CPU utilization (%)




*Great opportunity to use cloud idle resources*

fg = foreground/online workload

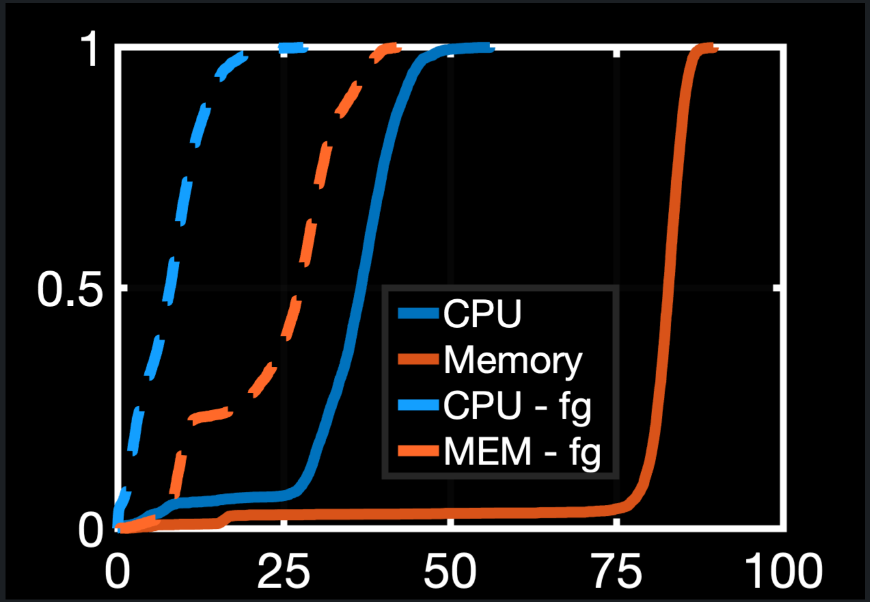
# Opportunity: Running Background Batch Workload




CDF of average CPU and memory usage, Alibaba cluster trace (2018).

# Opportunity: Running Background Batch Workload

Cumulative probability, F(x)




X = Average usage

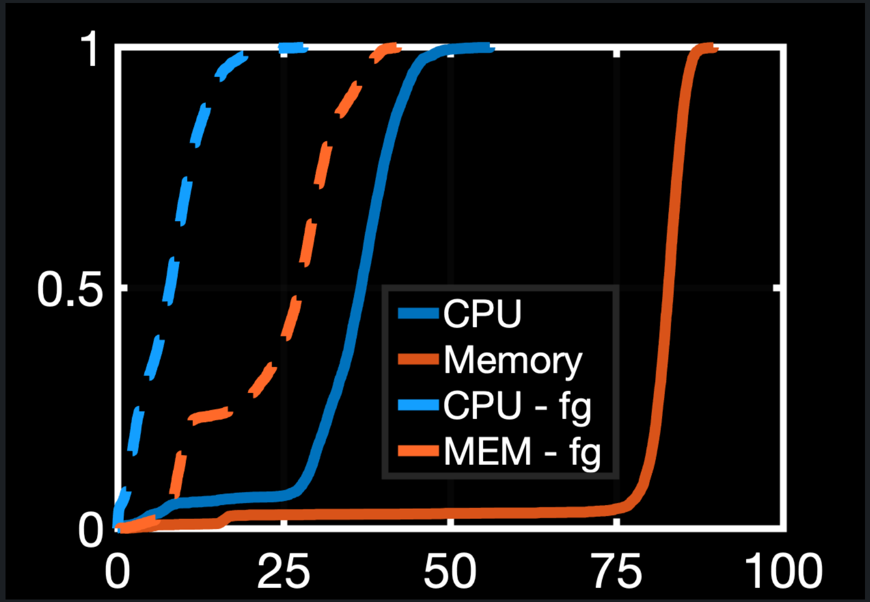
CDF of average CPU and memory usage, Alibaba cluster trace (2018).

bg = background/batch workload

- Key challenge: Resource contention
  - May violate SLOs of ***foreground dynamic workload***
  - Foreground workload is a ***black-box***, SLOs not known

# Opportunity: Running Background Batch Workload

Cumulative probability, F(x)




X = Average usage

- Key challenge: Resource contention
  - May violate SLOs of ***foreground dynamic workload***
  - Foreground workload is a ***black-box***, SLOs not known

*Problem statement: How to schedule background batch jobs to improve utilization without hurting black-box foreground performance?*

# Outline

- Prior approaches
- Our approach: Scavenger
  - High-level idea
  - Resource regulation algorithm
  - Evaluation methodology
  - Evaluation results
- Conclusion

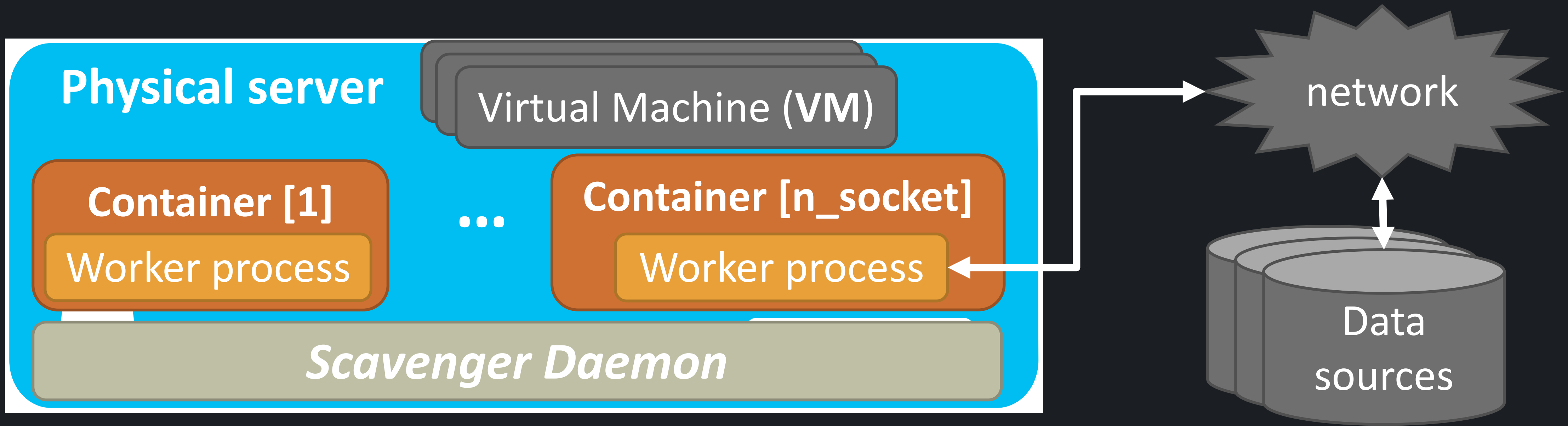
# Prior approaches

- ***Treat foregroud as white-box (assume SLO is known)***
  - Bistro (ATC’15, Facebook)
  - Heracles (ISCA’15, Google)
  - History-based harvesting (OSDI’16, Microsoft)
  - PARTIES (ASPLOS ‘19, SAIL group-Cornell Uni.)


- ***Typically focus only on one resource (need some critical profiling)***
  - dCat (EuroSys’18, IBM)
  - PerfIso (ATC’18, Microsoft)
    - Reprofiles often if workload changes

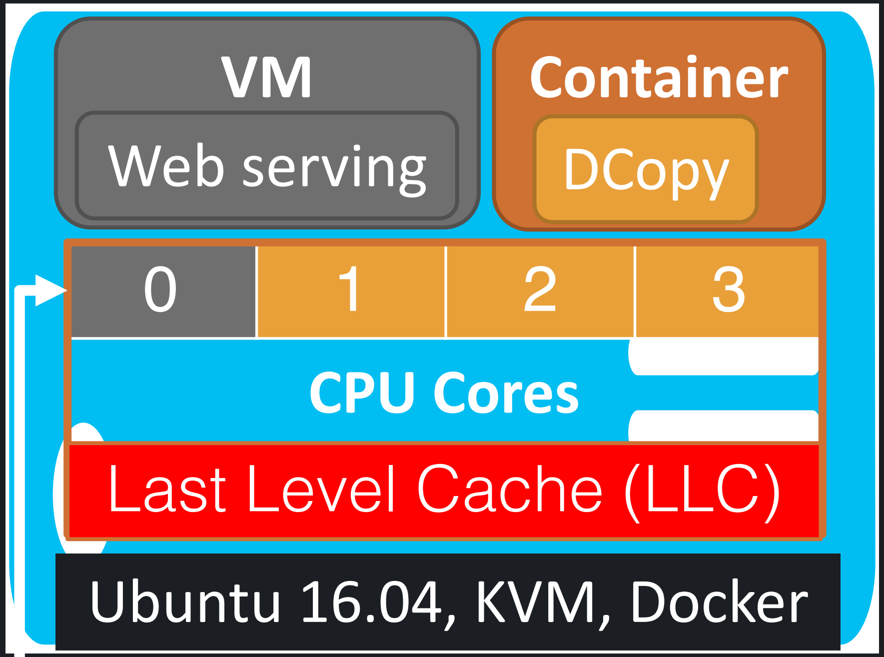
# Our approach: Scavenger

- Considers foreground workloads as a ***black-box***
- Takes ***multiple resources*** (processor, memory, nw) into account
- Is a dynamic and tunable solution
- Uses container as the ***agile execution environment*** for batch jobs



# Scavenger Daemon

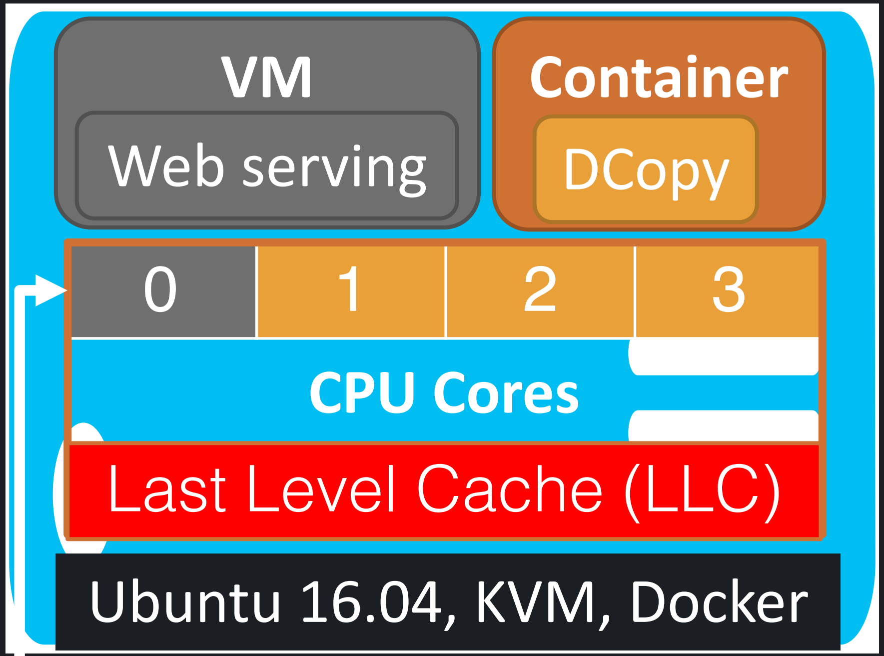
- Background resource regulation is the main design decision
  - Dealing with resource contention is challenging



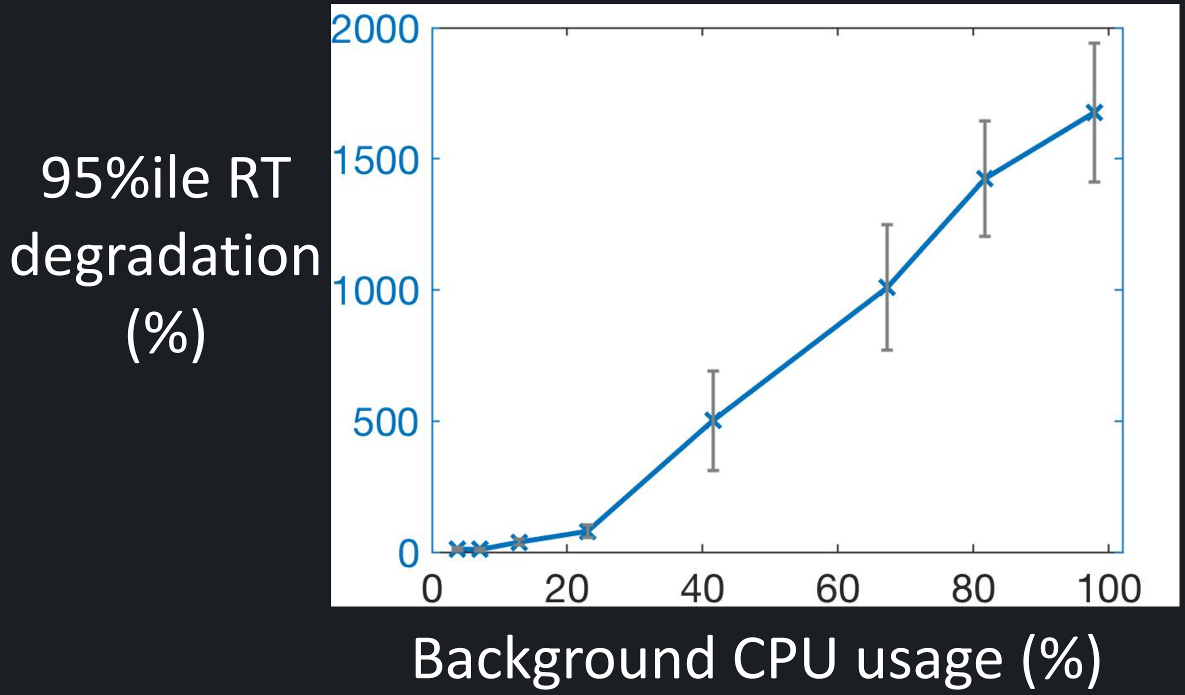

Using Linux's cpuset cgroups

# Scavenger Daemon

- Background resource regulation is the main design decision
  - Dealing with resource contention is challenging

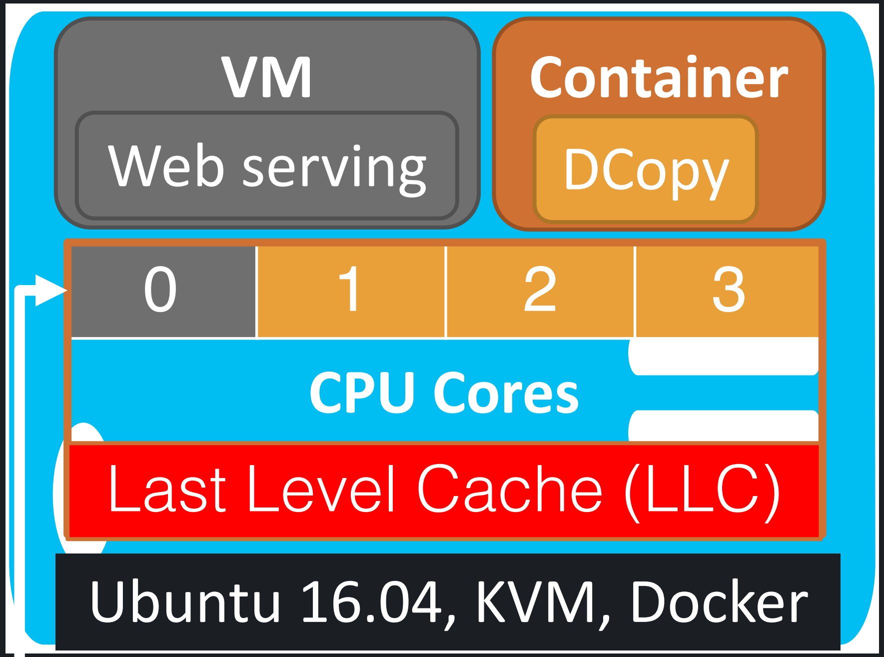



Using Linux's cpuset cgroups

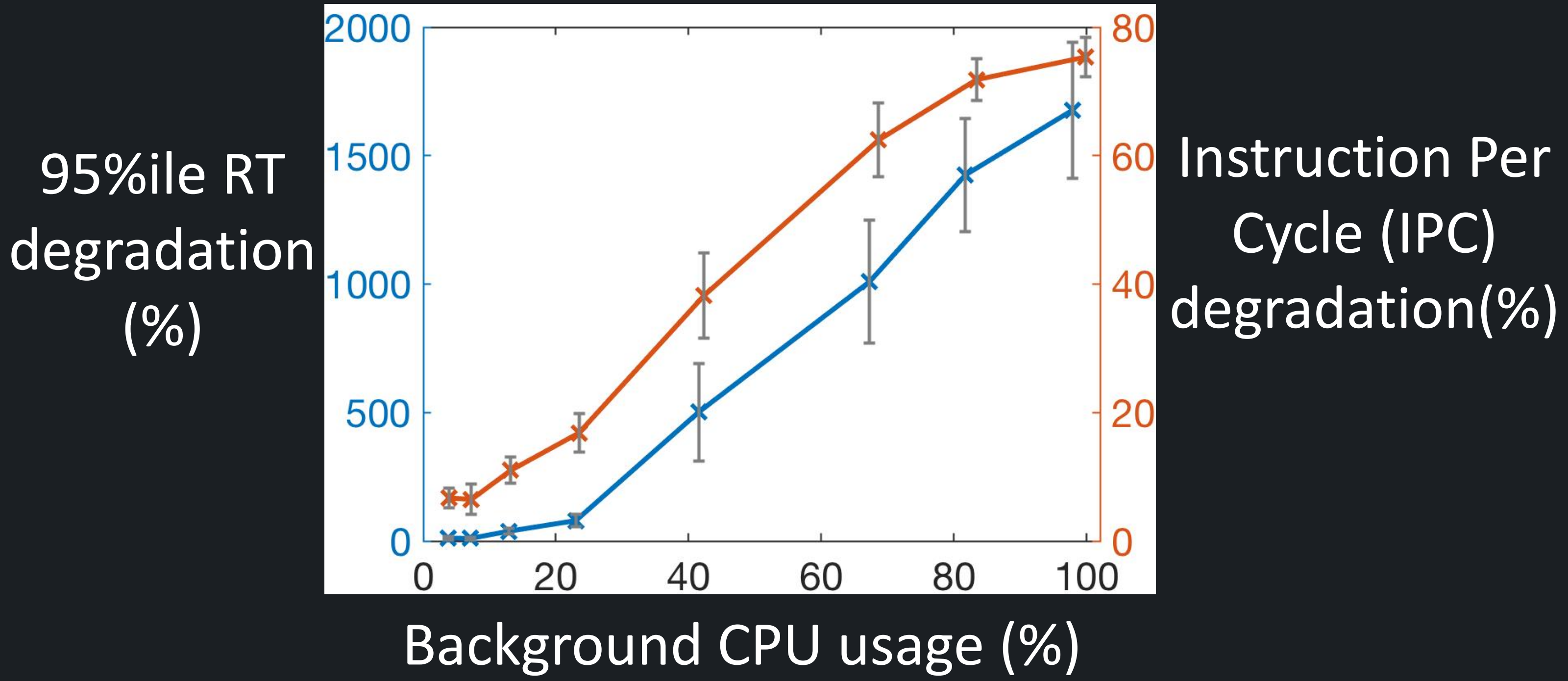


# Scavenger Daemon

- Background resource regulation is the main design decision
  - Dealing with resource contention is challenging

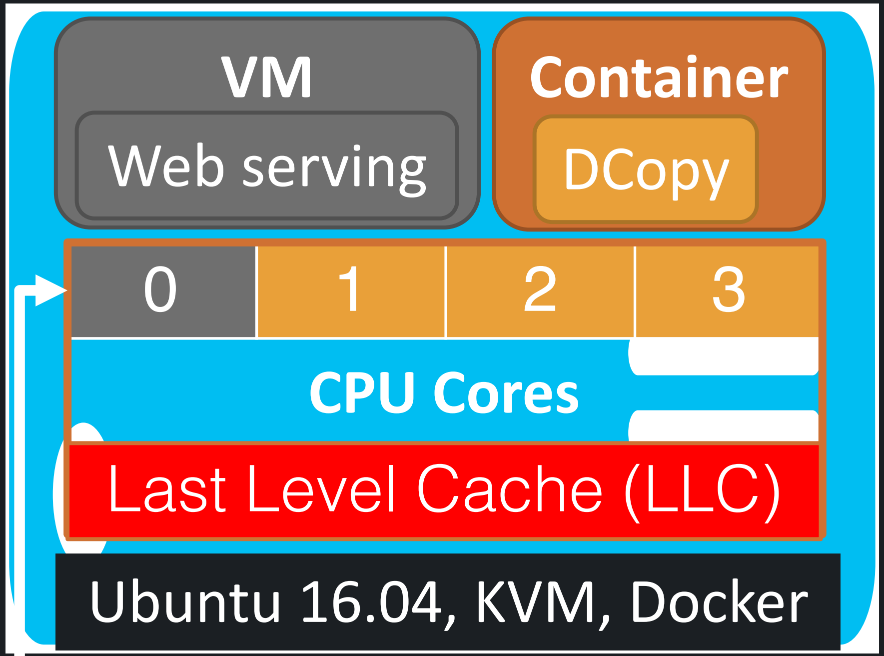



Using Linux's cpuset cgroups

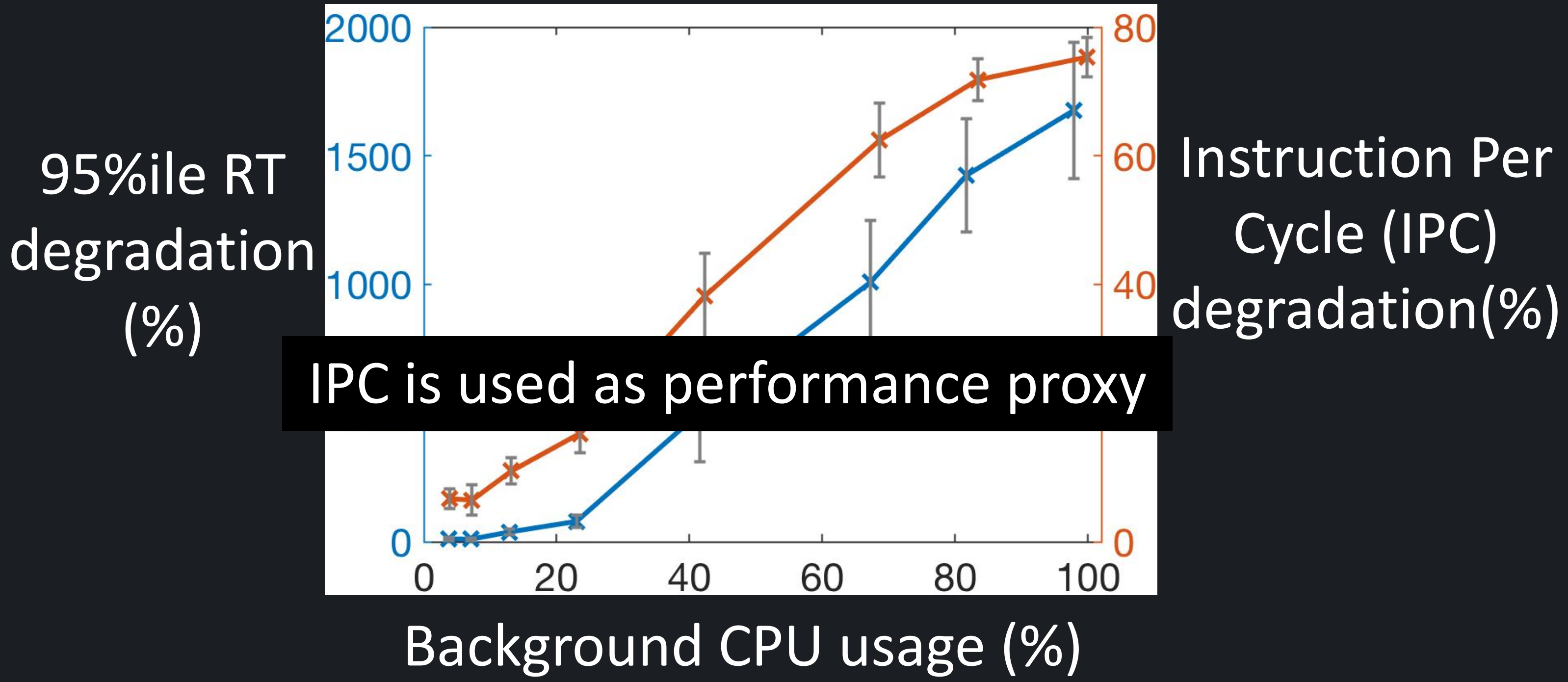


# Scavenger Daemon

- Background resource regulation is the main design decision
  - Dealing with resource contention is challenging




Using Linux's cpuset cgroups




IPC is used as performance proxy

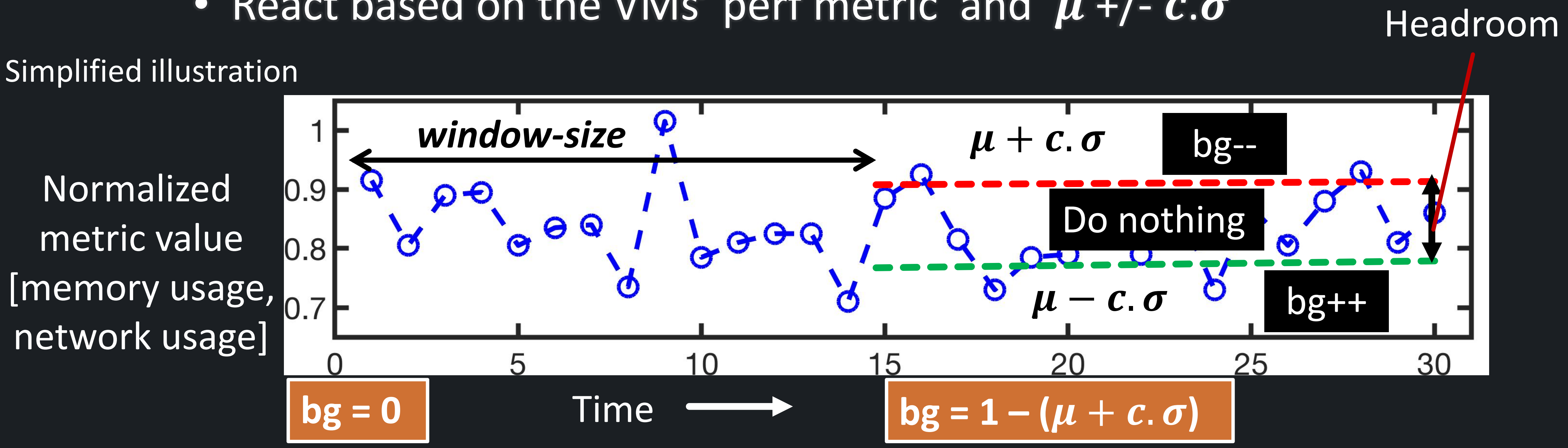
# Resource Regulation Algorithm

- Scavenger determines availability of resources for bg jobs
  - Background CPU load (cgroups)
    - CPU quota (maximum CPU cycles given to a process under the CFS)
  - Memory capacity (libvit)
  - Network bandwidth (TC)

# Resource Regulation Algorithm

- Our generic online algorithm
  - Monitor VMs' perf metric (e.g., memory usage) for window-size
  - Calculate mean, $\mu$, and standard deviation, $\sigma$
  - React based on the VMs' perf metric and $\boldsymbol{\mu}$ +/- $\boldsymbol{c.\sigma}$

Simplified illustration



# Evaluation Methodology

- Scavenger prototype implementation
  - Largely written in C++ and shell script  (~750 lines of code)

# Evaluation Methodology

- Scavenger prototype implementation
  - Largely written in C++ and shell script (~750 lines of code)

| | | | |
|---|---|---|---|
| Foreground | Training | CloudSuite | Widely used benchmark suite |
| | Testing | TailBench | Designed for latency-critical applications |
| Background (SparkBench) | KMeans | | A popular clustering algorithm |
| | SparkPi | | Computes Pi with very high precision |

# Evaluation Methodology

➢ Scavenger prototype implementation

- Largely written in C++ and shell script (~750 lines of code)

| | | | |
|---|---|---|---|
| Foreground | Training | CloudSuite | Widely used benchmark suite |
| | Testing | TailBench | Designed for latency-critical applications |
| Background (SparkBench) | KMeans | | A popular clustering algorithm |
| | SparkPi | | Computes Pi with very high precision |

Sensitivity analysis → Experimental evaluation

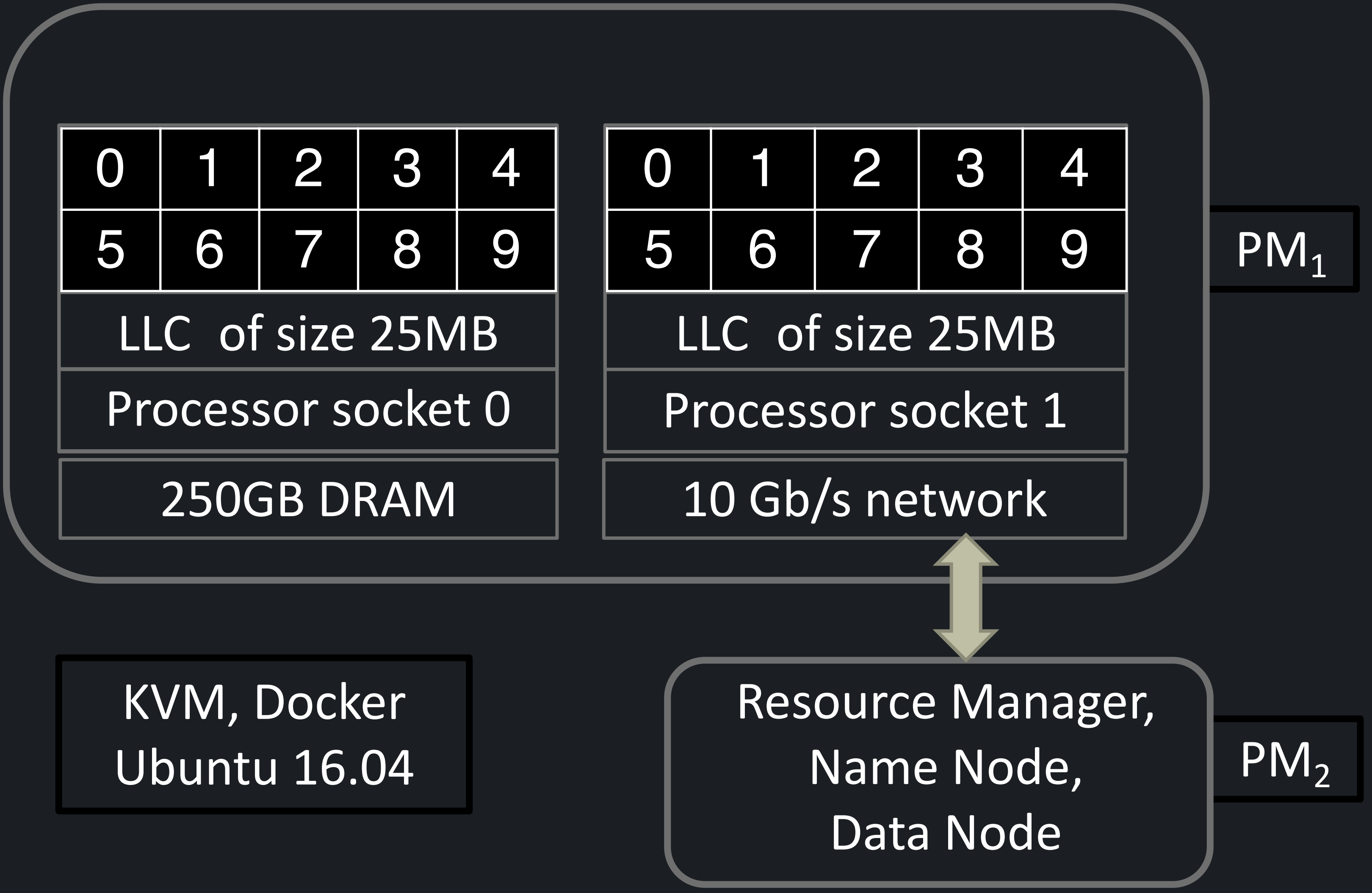
# TailBench

The load generators employed in TailBench are open-loop.

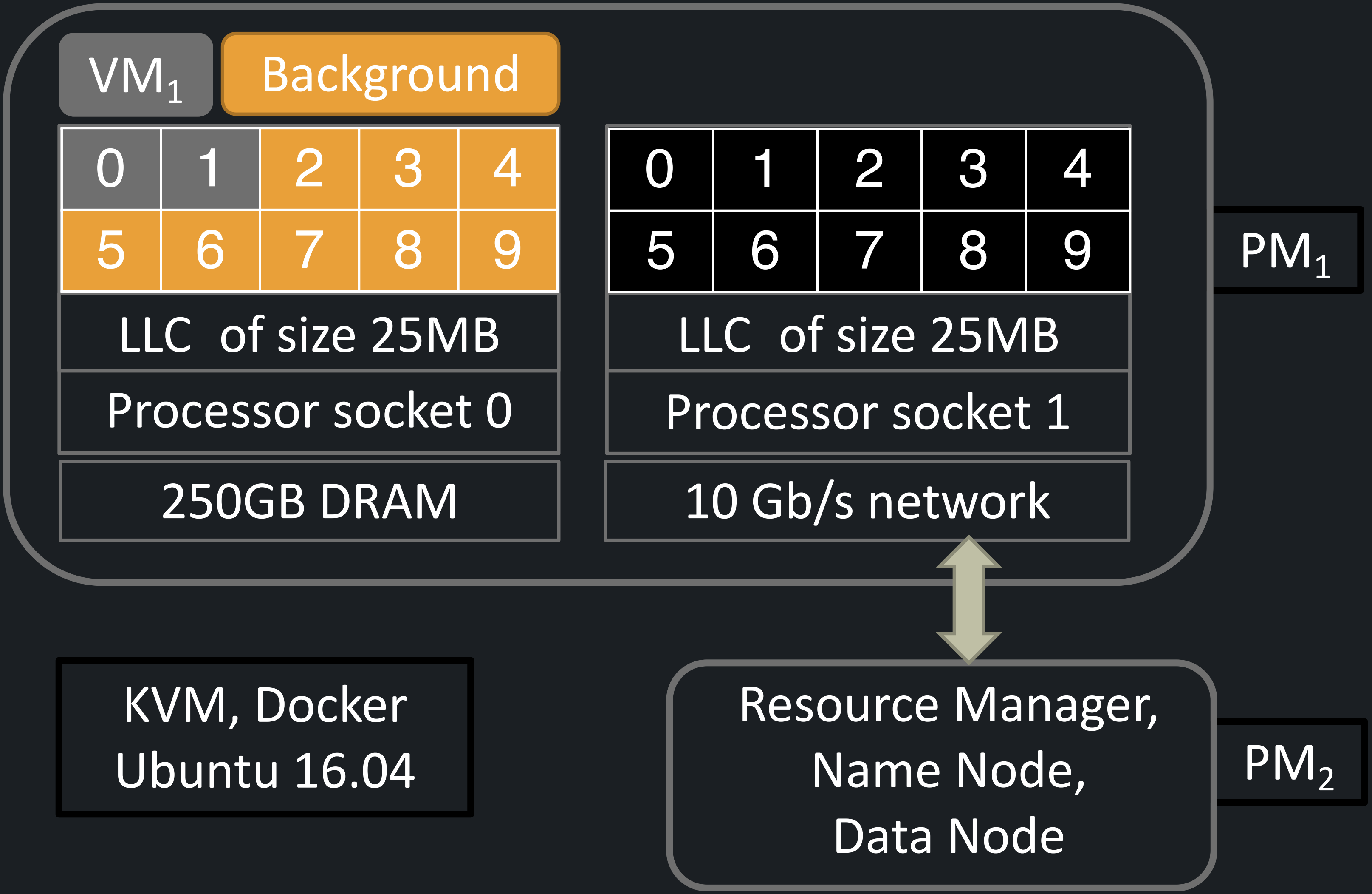
| Workload | Domain | Tail latency scale |
|---|---|---|
| Xapian | Online search | Milliseconds |
| Moses | Real-time translation | Milliseconds |
| Silo | In-memory database (OLTP) | Microseconds |
| Specjbb | Java middleware | Microseconds |
| Masstree | Key-value store | Microseconds |
| Shore | On-disk database (OLTP) | Milliseconds |
| Sphinx | Speech recognition | Seconds |
| Img-dnn | Image recognition | Milliseconds |

http://people.csail.mit.edu/sanchez/papers/2016.tailbench.iiswc.pdf

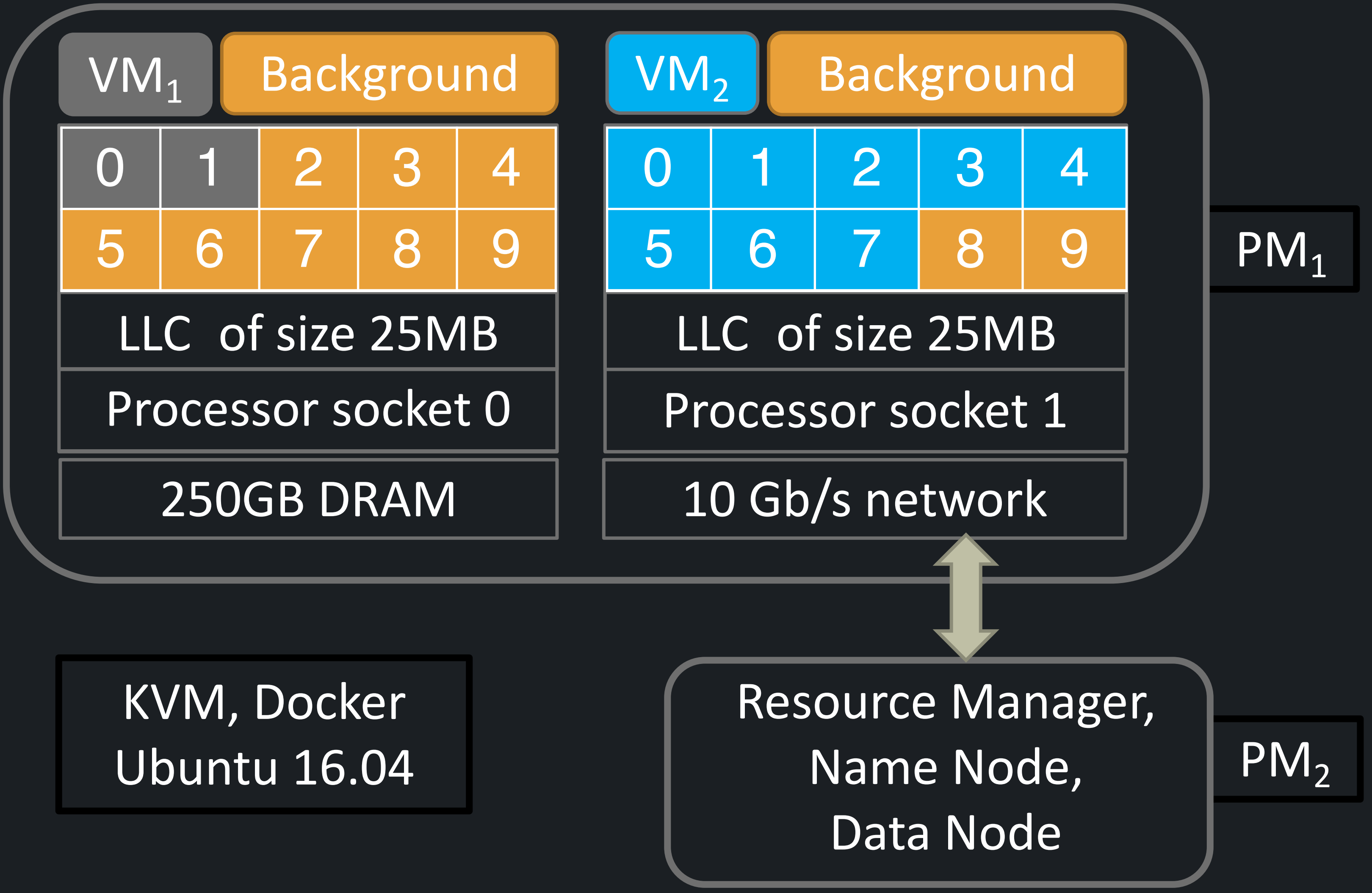
# Cloud Testbed



# Cloud Testbed



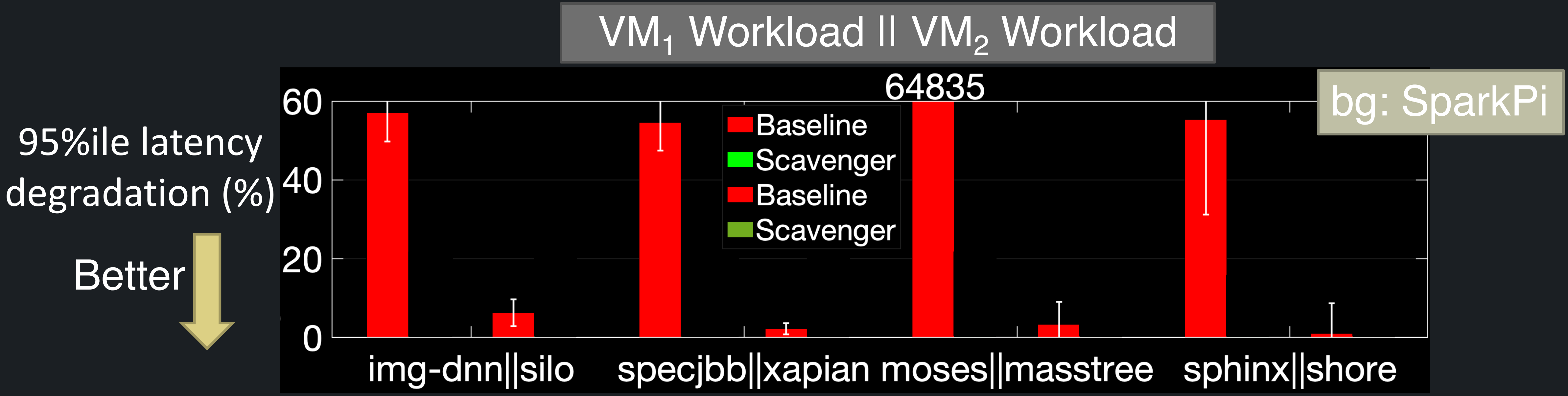
# Cloud Testbed



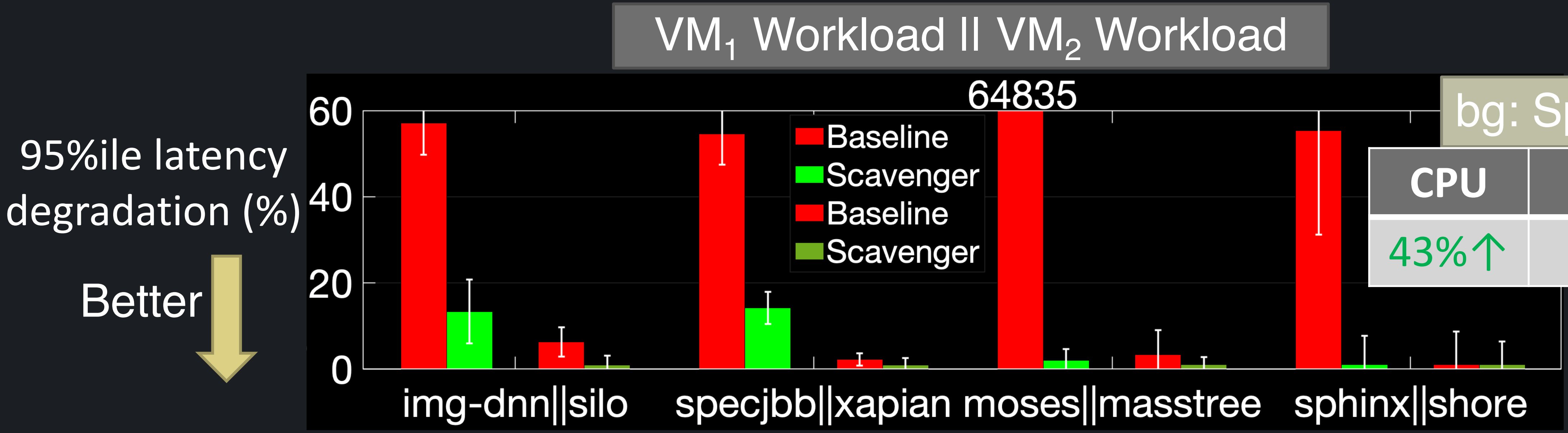
# Outline

- Prior approaches
- Our approach: Scavenger
  - High-level idea
  - Resource regulation algorithm
  - Evaluation methodology
  - **Evaluation results**
- Conclusion

# Evaluation with Spark jobs as background



# Evaluation with Spark jobs as background

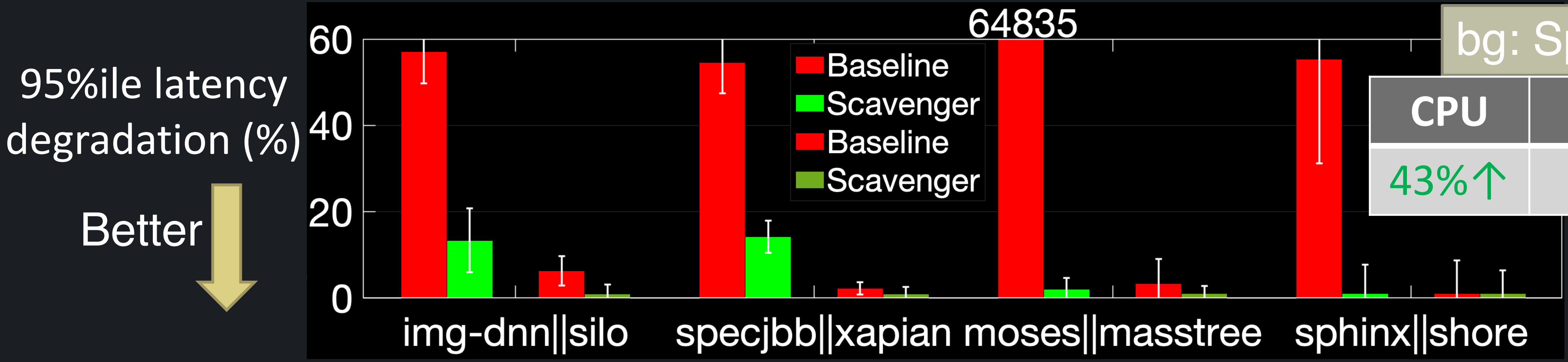



bg: SparkPi

| CPU | Memory |
|---|---|
| 43%↑ | 201%↑ |

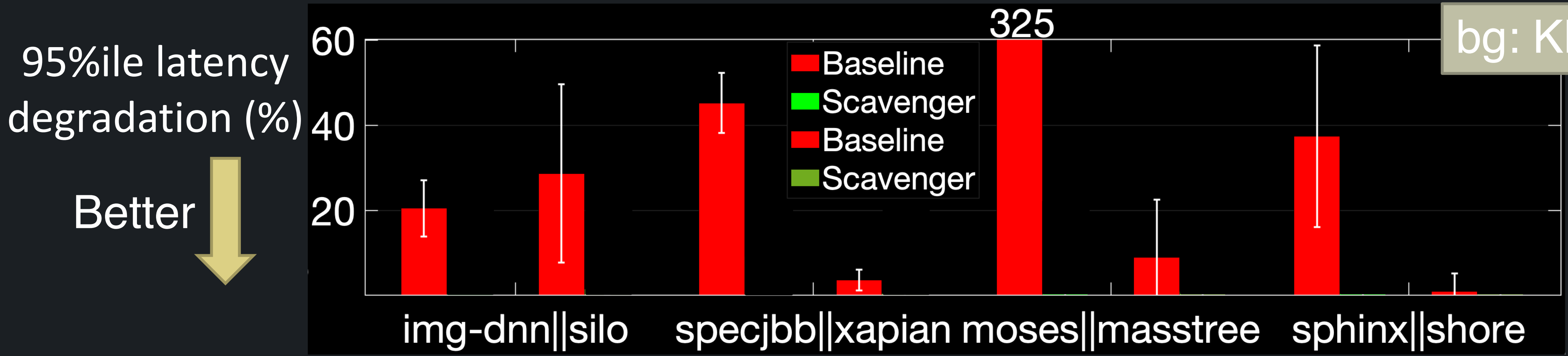
# Evaluation with Spark jobs as background

VM$_1$ Workload || VM$_2$ Workload




bg: SparkPi

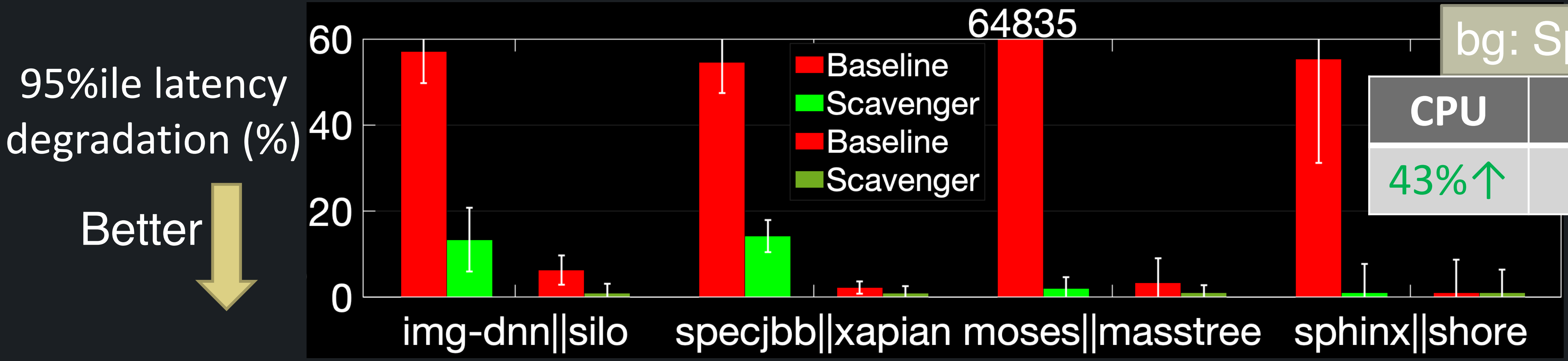
| CPU | Memory |
|---|---|
| 43%↑ | 201%↑ |




bg: KMeans

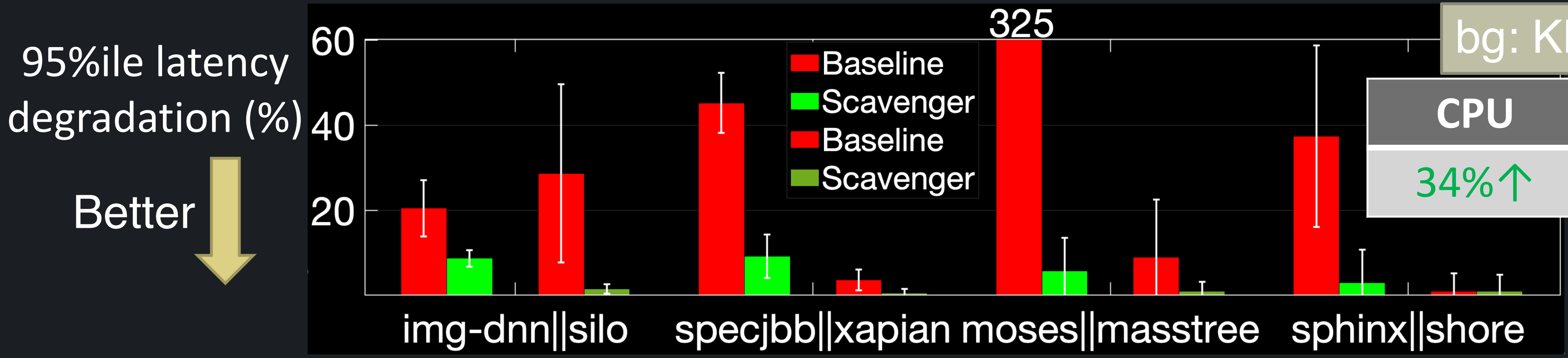
# Evaluation with Spark jobs as background

VM$_1$ Workload || VM$_2$ Workload




bg: SparkPi

| CPU | Memory |
|---|---|
| 43%↑ | 201%↑ |

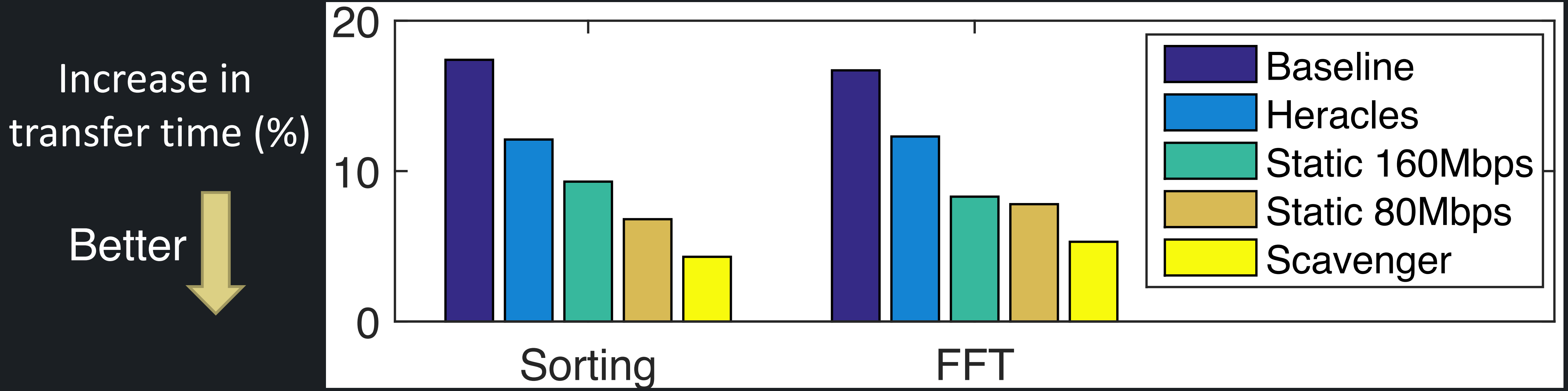



bg: KMeans

| CPU | Memory |
|---|---|
| 34%↑ | 321%↑ |

# Media-streaming as foreground

Lab testbed: 2-vCPU foreground VM, 2-core background container.

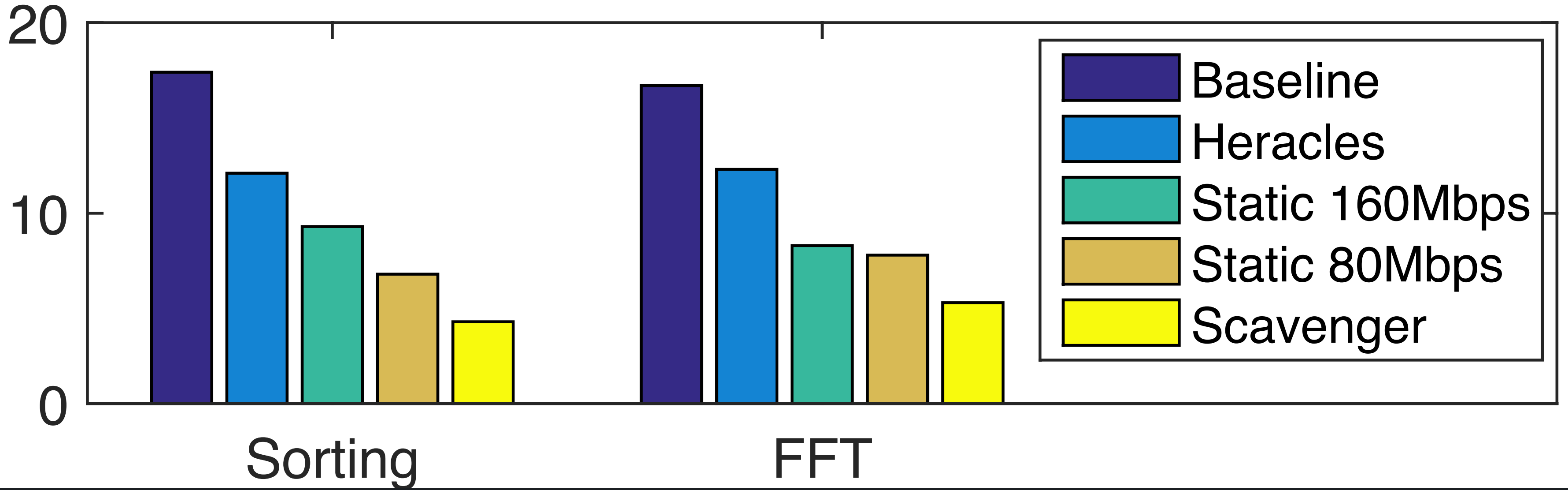


# Media-streaming as foreground

Lab testbed: 2-vCPU foreground VM, 2-core background container.

Increase in transfer time (%)

Better




| CPU | Network |
|---|---|
| 37%↑ | 180Mbps ↑ |

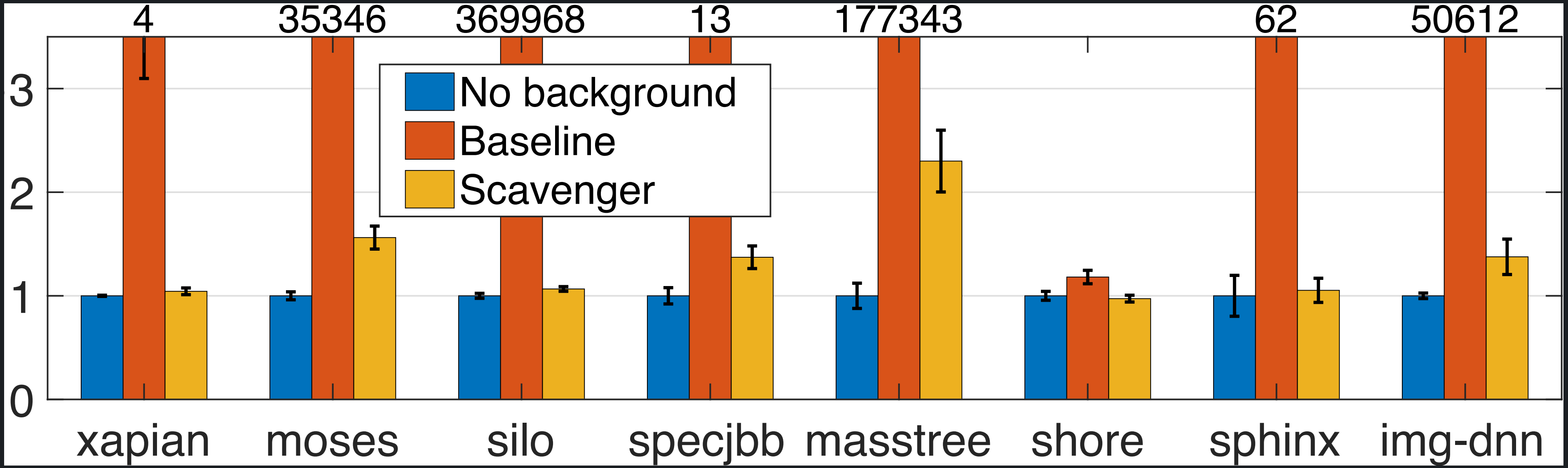
Scavenger outperforms static approaches while affording higher background usage.

# Limit Study With DCopy as the Background

Cloud testbed: 4-vCPU foreground VM, 6-core background DCopy container.

Normalized 95%ile latency

Better

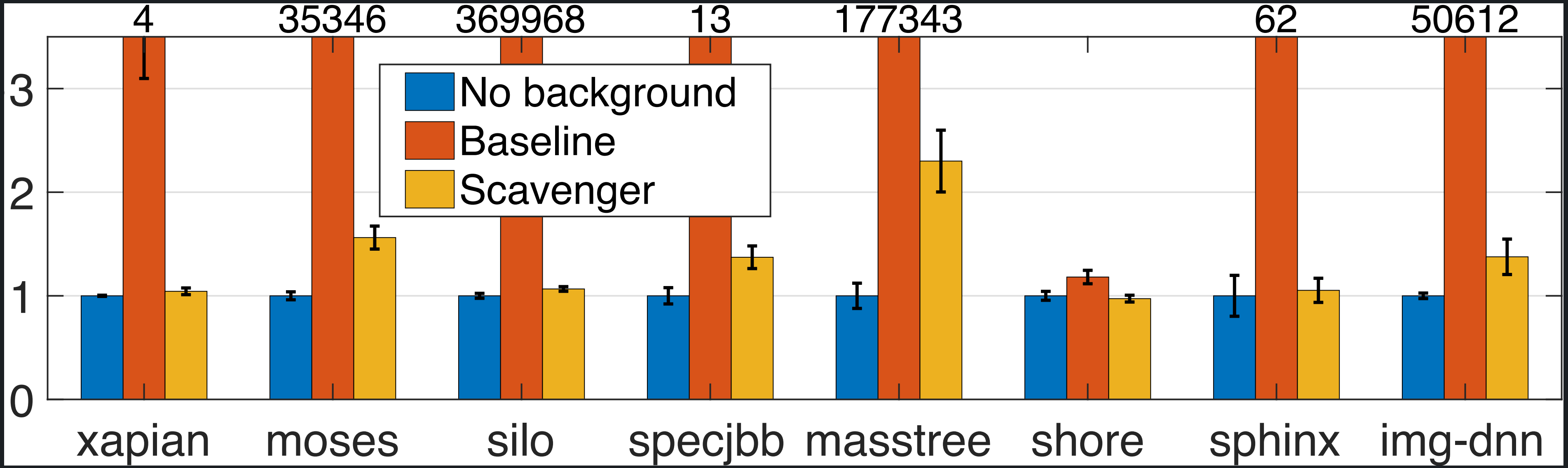


# Limit Study With DCopy as the Background

Cloud testbed: 4-vCPU foreground VM, 6-core background DCopy container.

Normalized 95%ile latency

Better




3-5% CPU ↑

Scavenger can successfully and aggressively regulate bg workload to mitigate its impact on fg performance.

# Conclusion

- ➢Significant opportunity to use cloud idle resource
- ➢Important features of cloud tenant’s VM workloads
  - Black-box, SLOs not known
  - Dynamic behavior
- ➢Scavenger: Dynamic, black-box multi-resource manager
  - Does not instrument or profile the tenant VMs offline.
  - Increases server utilization without compromising the resource demands of tenant VMs.

# Thank You

Scavenger: A Black-Box Batch Workload Resource Manager for Improving Utilization in Cloud Environments

# Q&A

Seyyed **Ahmad** Javadi

sjavadi@cs.stonybrook.edu

PACE Lab at Stony Brook University

ACM Symposium on Cloud Computing 2019