

Datacenter Computing: An Alibaba Case Study

DS 5110: Big Data Systems (Spring 2023)
Lecture 10

Yue Cheng



The sorry state of server utilization and the impending post-hypervisor era

Alex Benik, Battery Ventures Nov 30, 2013 - 10:30 AM CDT

- A McKinsey study in 2008 pegging data-center utilization at roughly 6 percent.
- A Gartner report from 2012 putting industry wide utilization rate at 12 percent.
- An [Accenture paper](#) sampling a small number on Amazon EC2 machines finding 7percent utilization over the course of a week.
- The [charts and quote below from Google](#), which show three-month average utilization rates for 20,000 server clusters. The typical cluster on the left spent most of its time running between 20-40 percent of capacity, and the highest utilization cluster on the right reaches such heights only because it's doing batch work.

The sorry state of server utilization and the impending post-hypervisor era

Alex Benik, Battery Ventures Nov 30, 2013 - 10:30 AM CDT

Me: Do you track server and CPU utilization?

Wall Street IT Guru: Yes

Me: So it's a metric you report on with other infrastructure KPIs?

Wall Street IT Guru: No way, we don't put it in reports. If people knew how low it really is, we'd all get fired.

- A McKinsey study in 2008 pegging data-center utilization at roughly 6 percent.
- A Gartner report from 2012 putting industry wide utilization rate at 12 percent.
- An [Accenture paper](#) sampling a small number on Amazon EC2 machines finding 7percent utilization over the course of a week.
- The [charts and quote below from Google](#), which show three-month average utilization rates for 20,000 server clusters. The typical cluster on the left spent most of its time running between 20-40 percent of capacity, and the highest utilization cluster on the right reaches such heights only because it's doing batch work.

Workload co-location

- Run all workloads on one datacenter
 - Latency-sensitive, long-running, **online** workloads (higher priority)
 - Short-lived, **offline**, batch job workloads
- Improved utilization and elasticity
 - Fill batch jobs into resource “**gaps**” that are not used by interactive workloads
 - Evict batch jobs if interactive workload demand **spikes**

Workload co-location

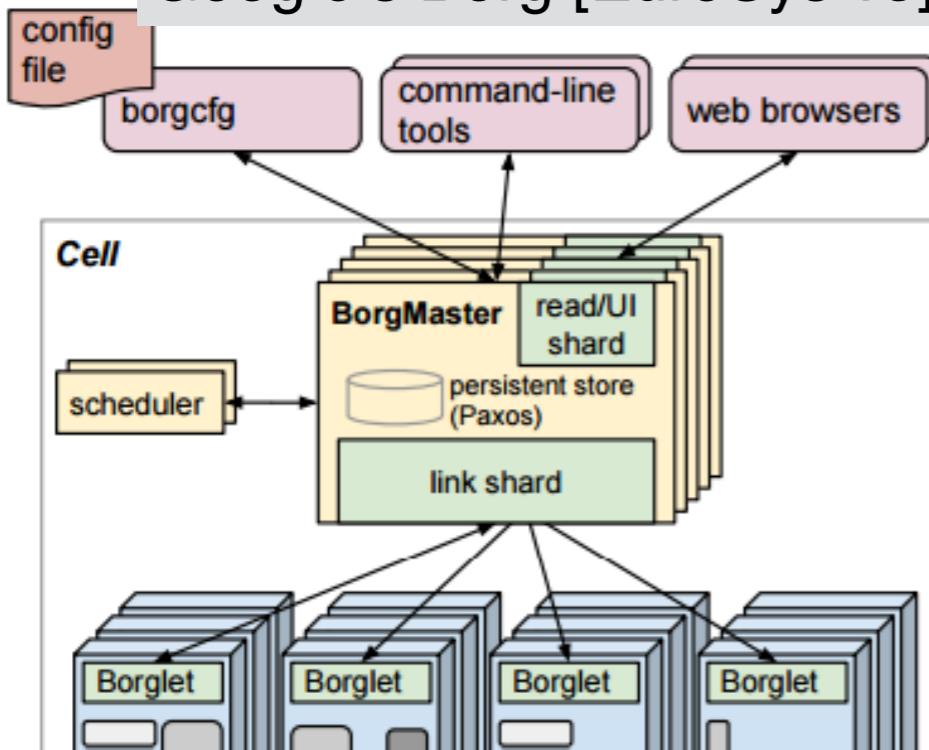
- Run all workloads on one datacenter
 - Latency-sensitive, long-running, **online** workloads (higher priority)
 - Short-lived, **offline**, batch job workloads
- Improved utilization and elasticity
 - Fill batch jobs into resource “**gaps**” that are not used by interactive workloads
 - Evict batch jobs if interactive workload demand **spikes**



Workload co-location

- Run all workloads on one datacenter
 - Latency-sensitive, long-running, **online** workloads (higher priority)
 - Short-lived, **offline**, batch job workloads
- Improved utilization and elasticity
 - Fill batch jobs into resource “**gaps**” that are not used by interactive workloads
 - Evict batch jobs if interactive workload demand **spikes**

Google's Borg [EuroSys'15]



Google trace analysis [SoCC'12]

Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis

Charles Reiss
University of California,
Berkeley
charles@eecs.berkeley.edu

Gregory R. Ganger
Carnegie Mellon University
ganger@ece.cmu.edu

Alexey Tumanov
Carnegie Mellon University
atumanov@cmu.edu

Randy H. Katz
University of California,
Berkeley
randy@eecs.berkeley.edu

Michael A. Kozuch
Intel Labs
michael.a.kozuch@intel.com

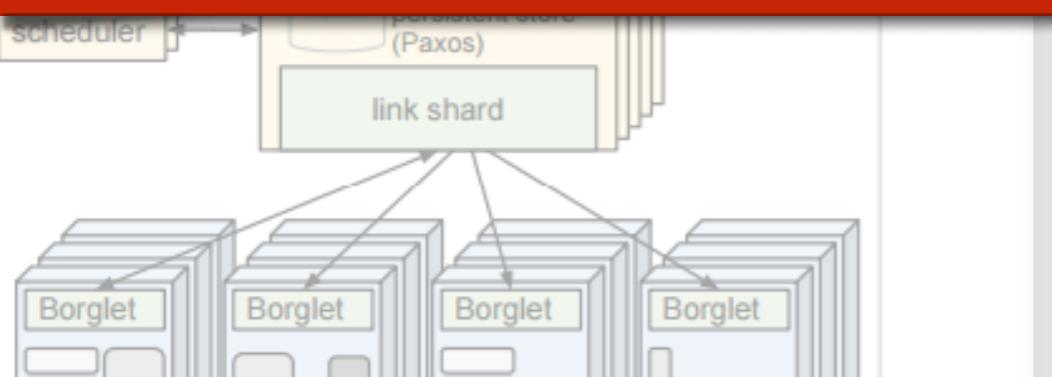
ABSTRACT

To better understand the challenges in developing effective cloud-based resource schedulers, we analyze the first publicly available trace data from a sizable multi-purpose cluster. The most notable workload characteristic is heterogeneity: in resource types (e.g., cores:R.A.M.

Workload co-location

- Run all workloads on one datacenter
 - Latency-sensitive, long-running, **online** workloads (higher priority)
 - Short-lived, **offline**, batch job workloads
- Improved utilization and elasticity
 - Fill batch jobs into resource “**gaps**” that are not used by interactive workloads
 - Evict batch jobs if interactive workload demand **spikes**

Co-located workload patterns
remain a mystery



Berkeley
charles@eecs.berkeley.edu
atumanov@cmu.edu

Gregory R. Ganger
Carnegie Mellon University
ganger@ece.cmu.edu

Randy H. Katz
University of California,
Berkeley
randy@eecs.berkeley.edu

Michael A. Kozuch
Intel Labs
michael.a.kozuch@intel.com

ABSTRACT

To better understand the challenges in developing effective cloud-based resource schedulers, we analyze the first publicly available trace data from a sizable multi-purpose cluster. The most notable workload characteristic is heterogeneity: in resource types (e.g., cores:RAM

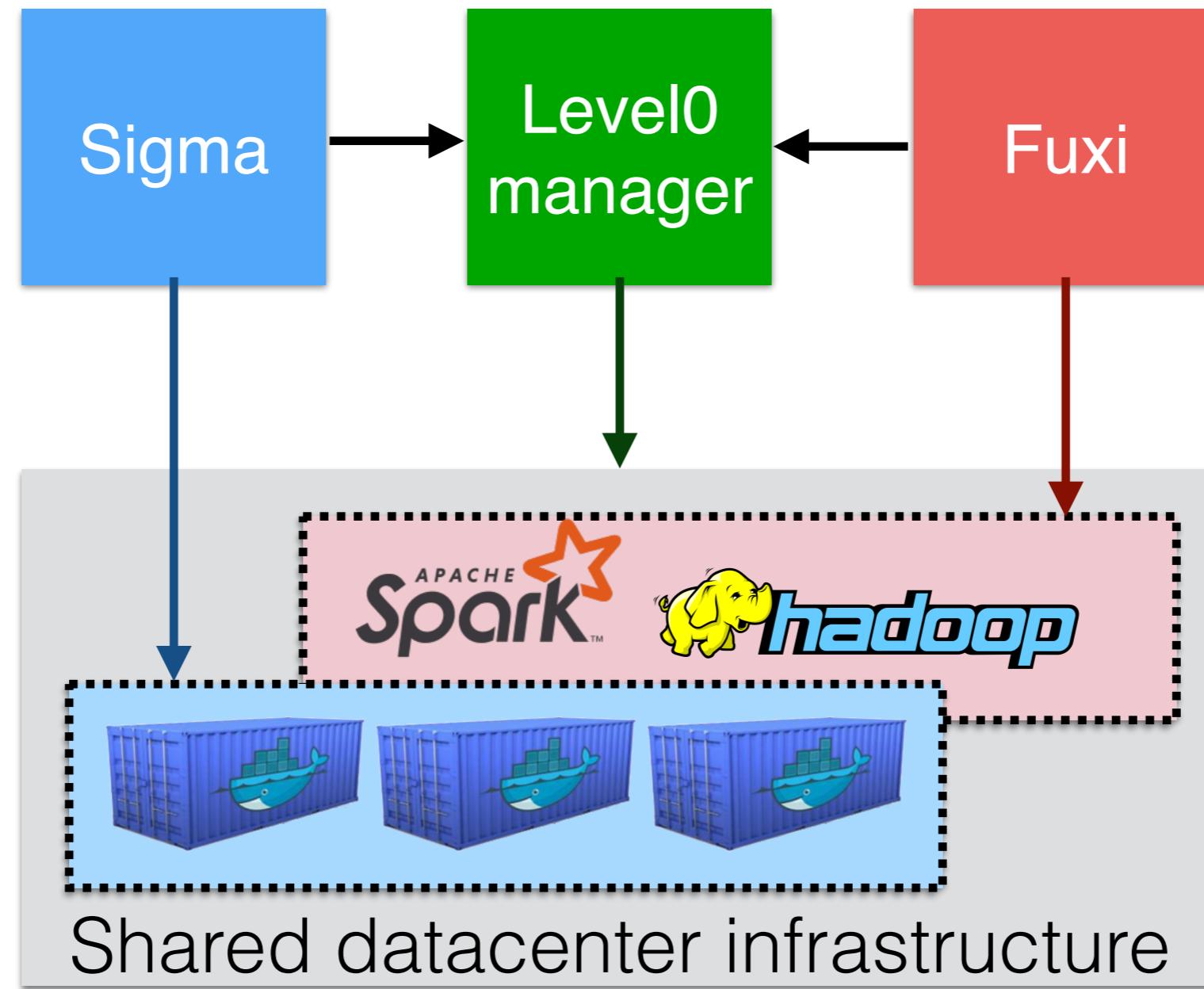
The Alibaba trace

Released Aug 2017

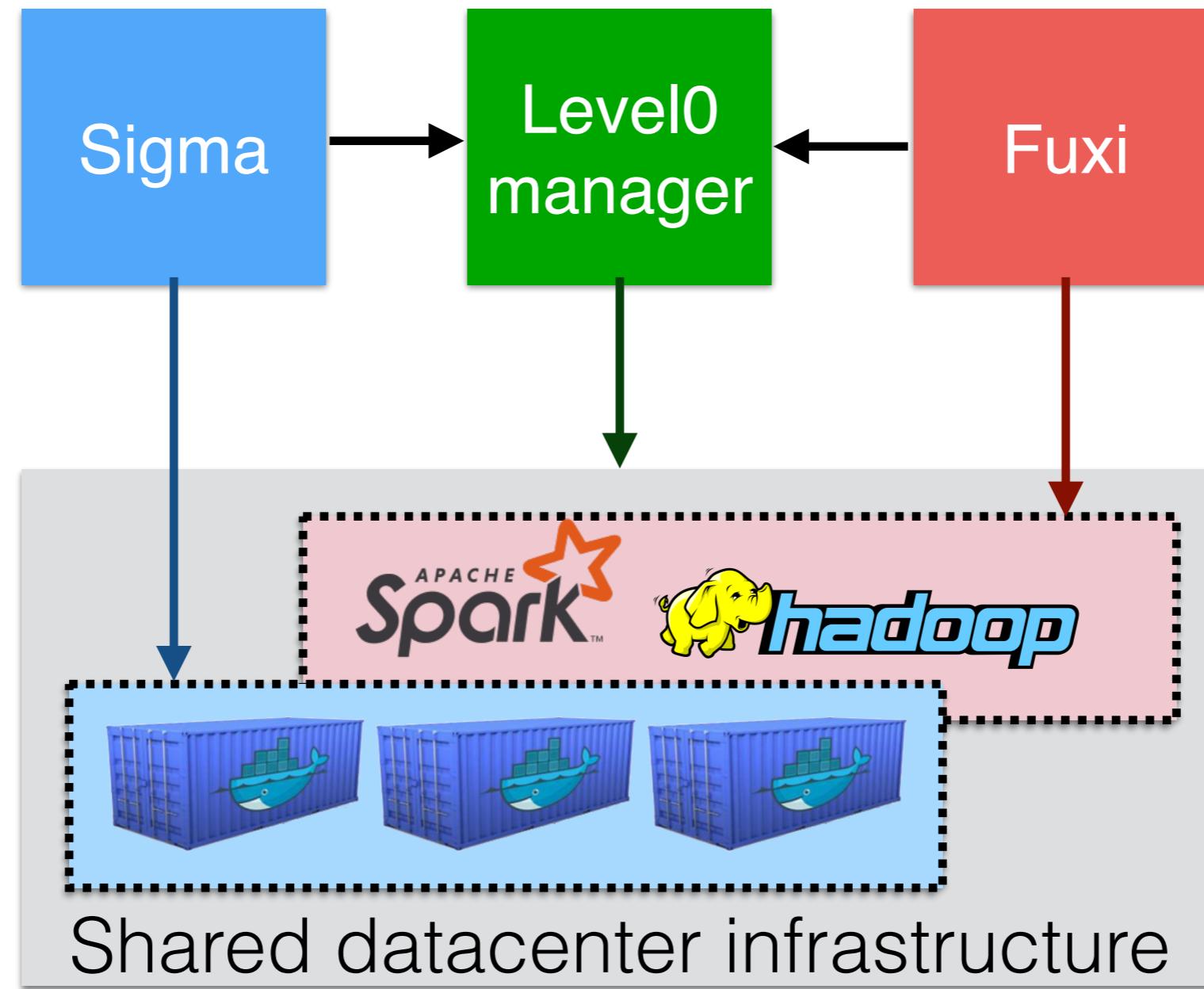
“**The data is provided to address the challenges Alibaba face in IDCS where online services and batch jobs are co-allocated ...**

- Two general types of workloads sharing a production cluster of **1.3k machines** for **24 hours**
 - **Containerized** interactive services (e.g., Email, DBs)
 - **Batch** jobs (**DAG** of tasks, e.g., MapReduce/Spark)
 - Ran on separate clusters before **2015**

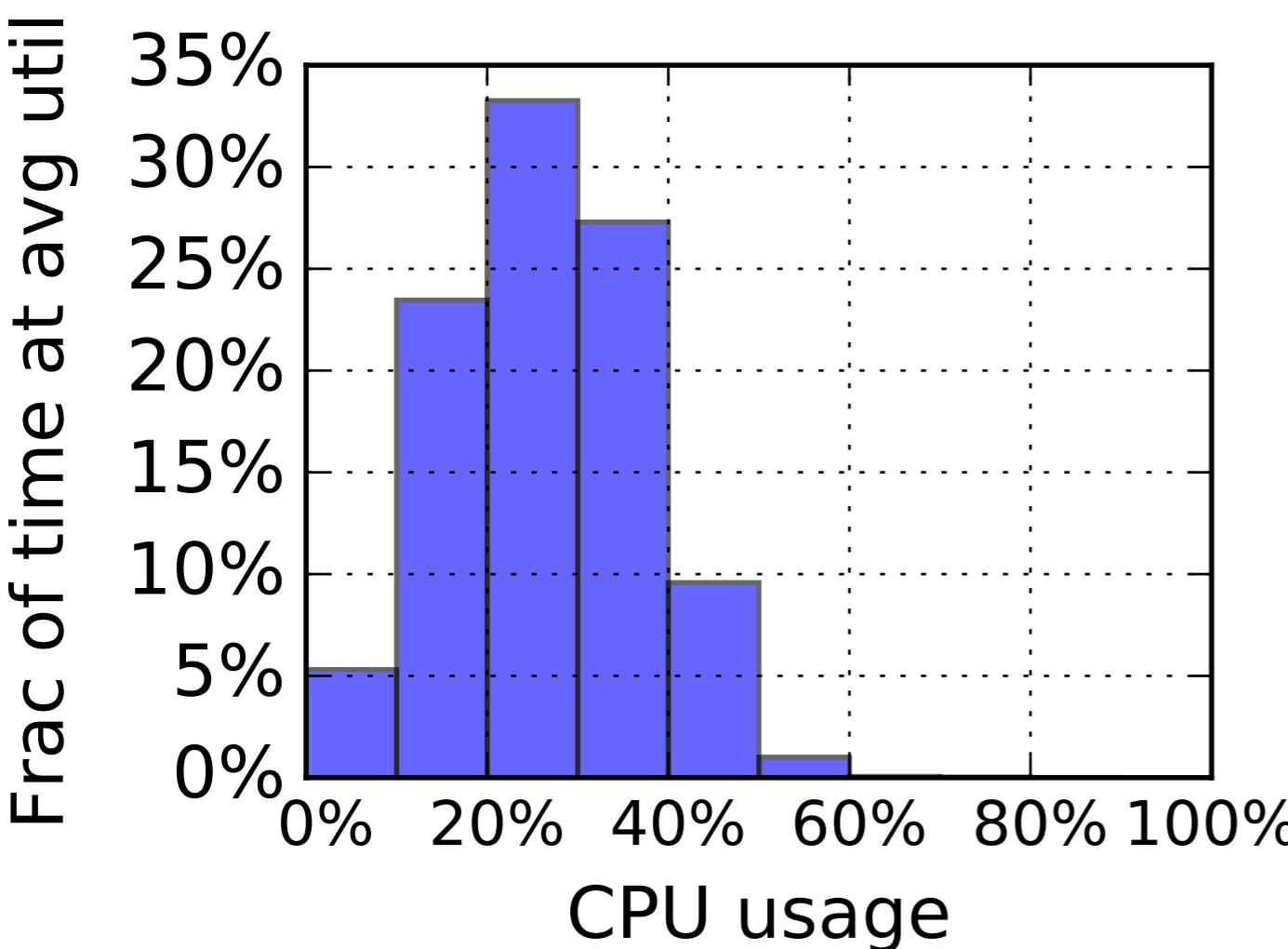
Alibaba's cluster management systems



Overall resource utilization

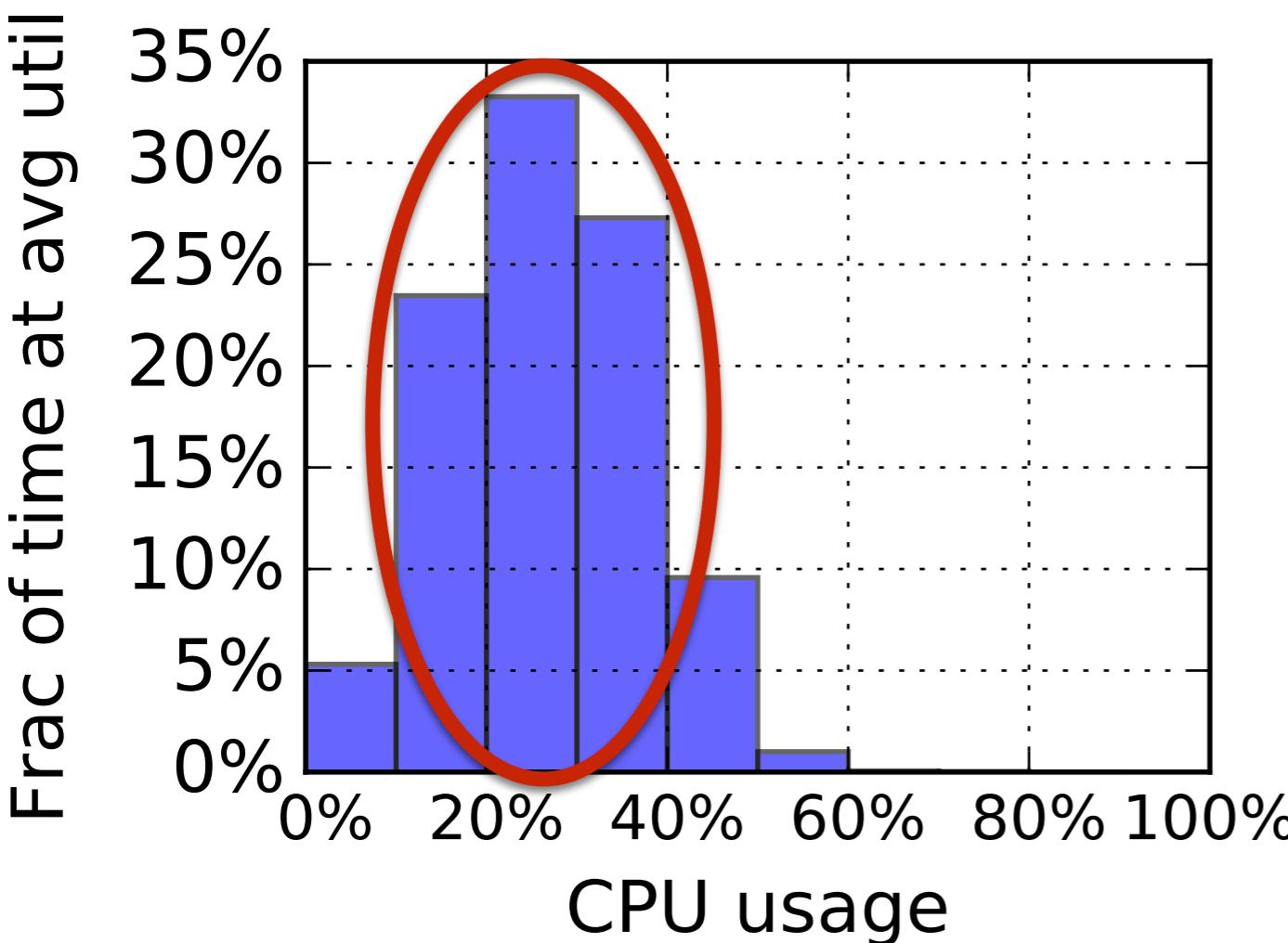


Average machine utilization



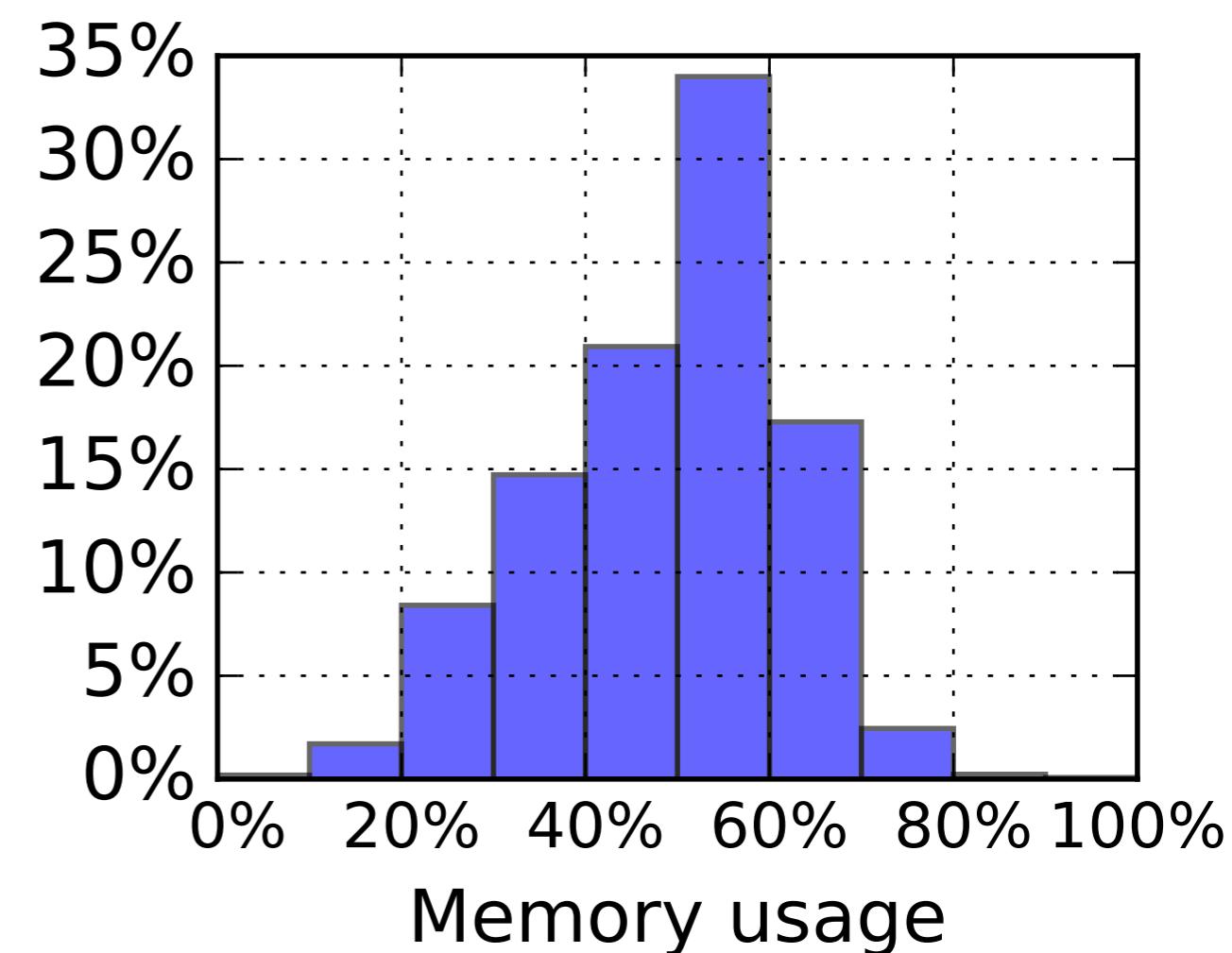
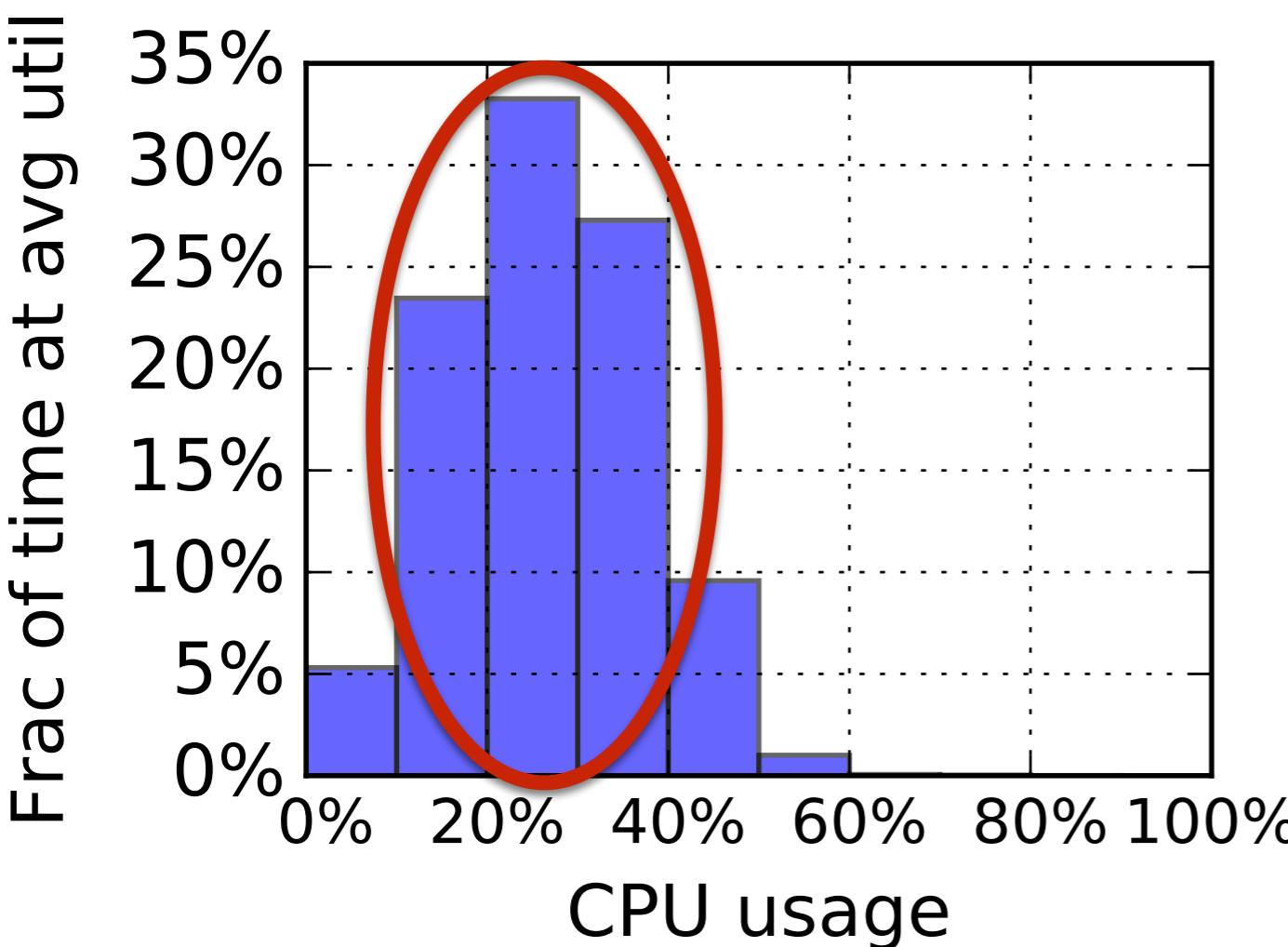
Average machine utilization

> 80% time running b/w
10-30% CPU usage



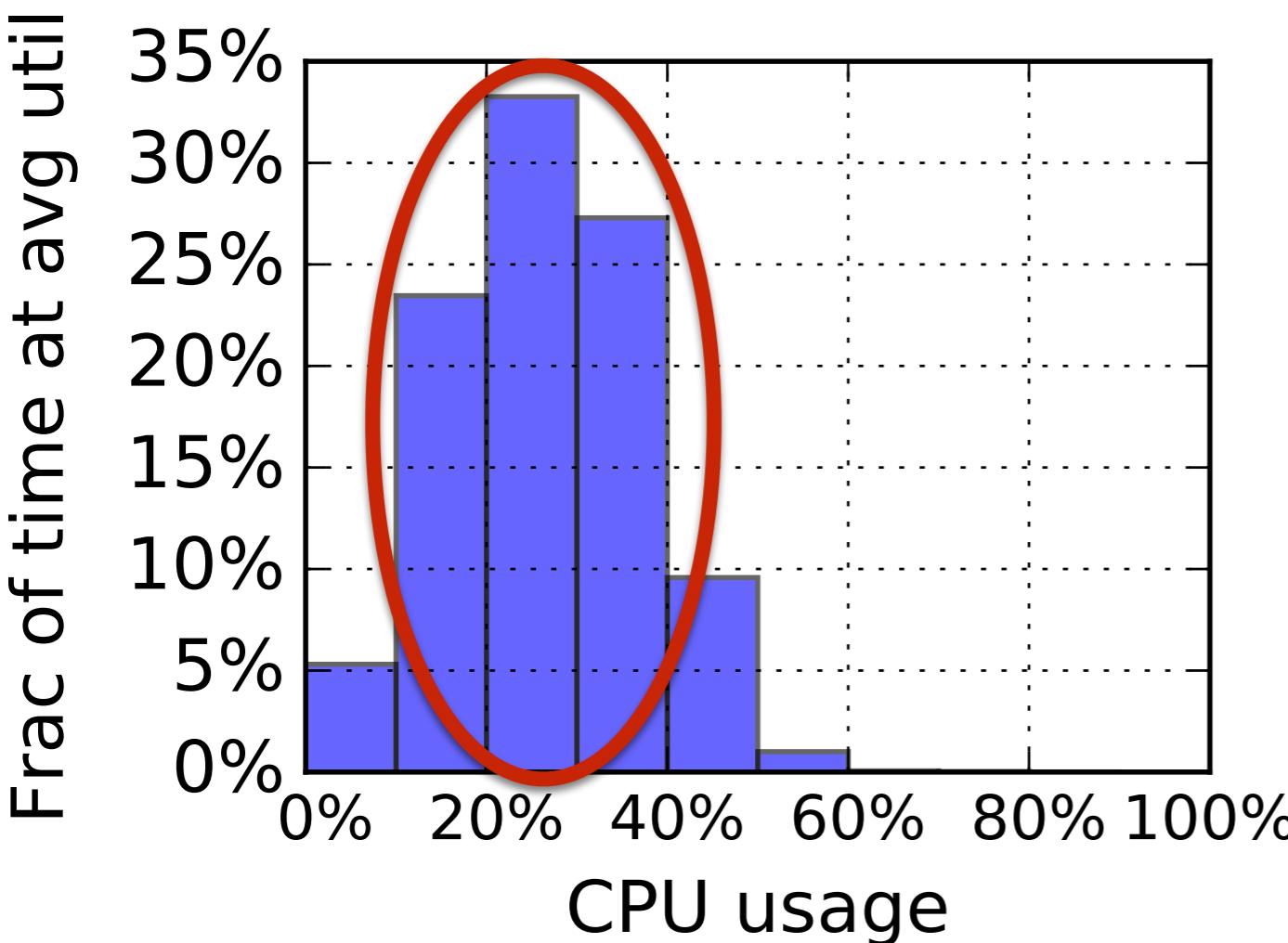
Average machine utilization

> 80% time running b/w
10-30% CPU usage

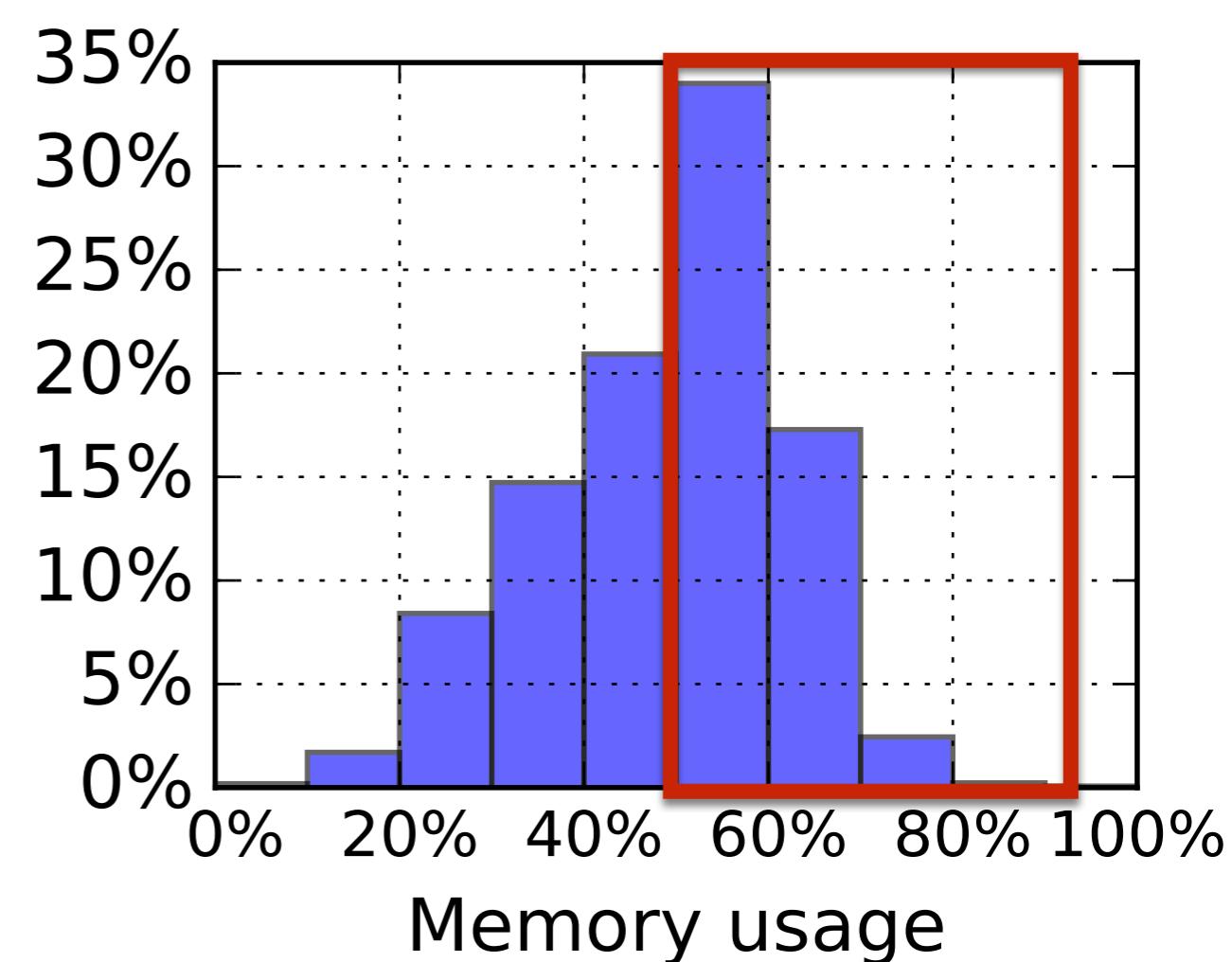


Average machine utilization

> 80% time running b/w
10-30% CPU usage

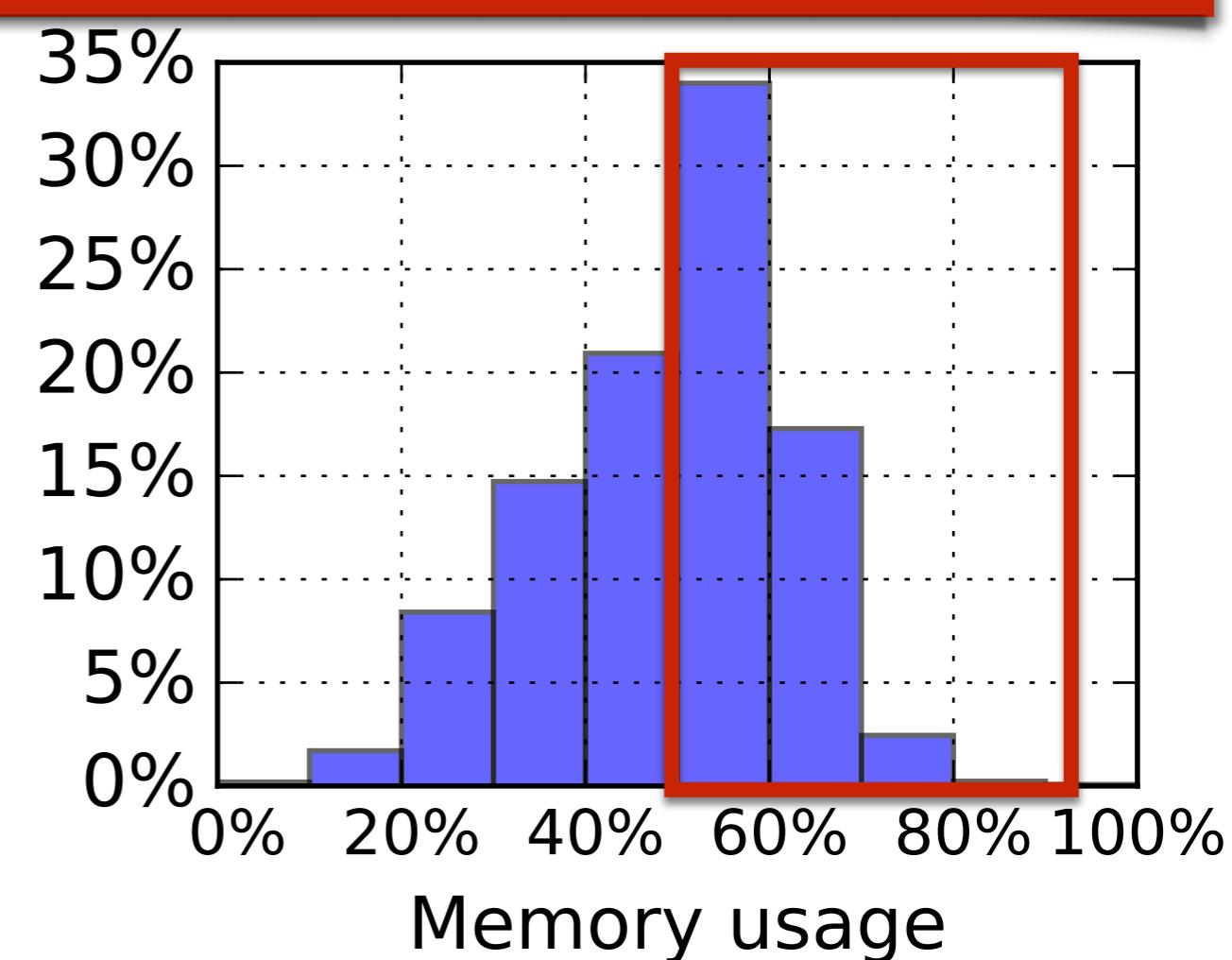
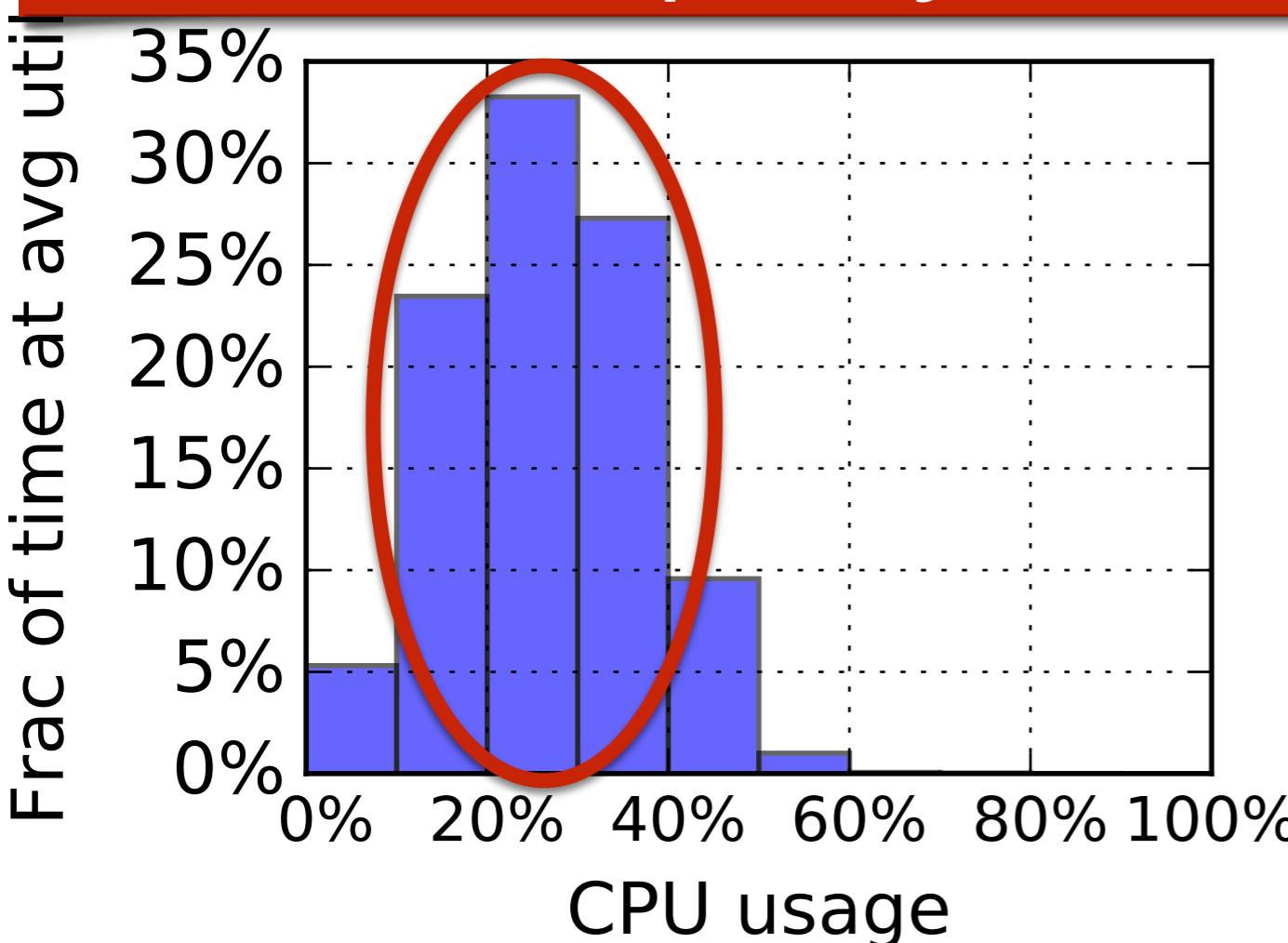


> 50% memory usage
for over 55% time

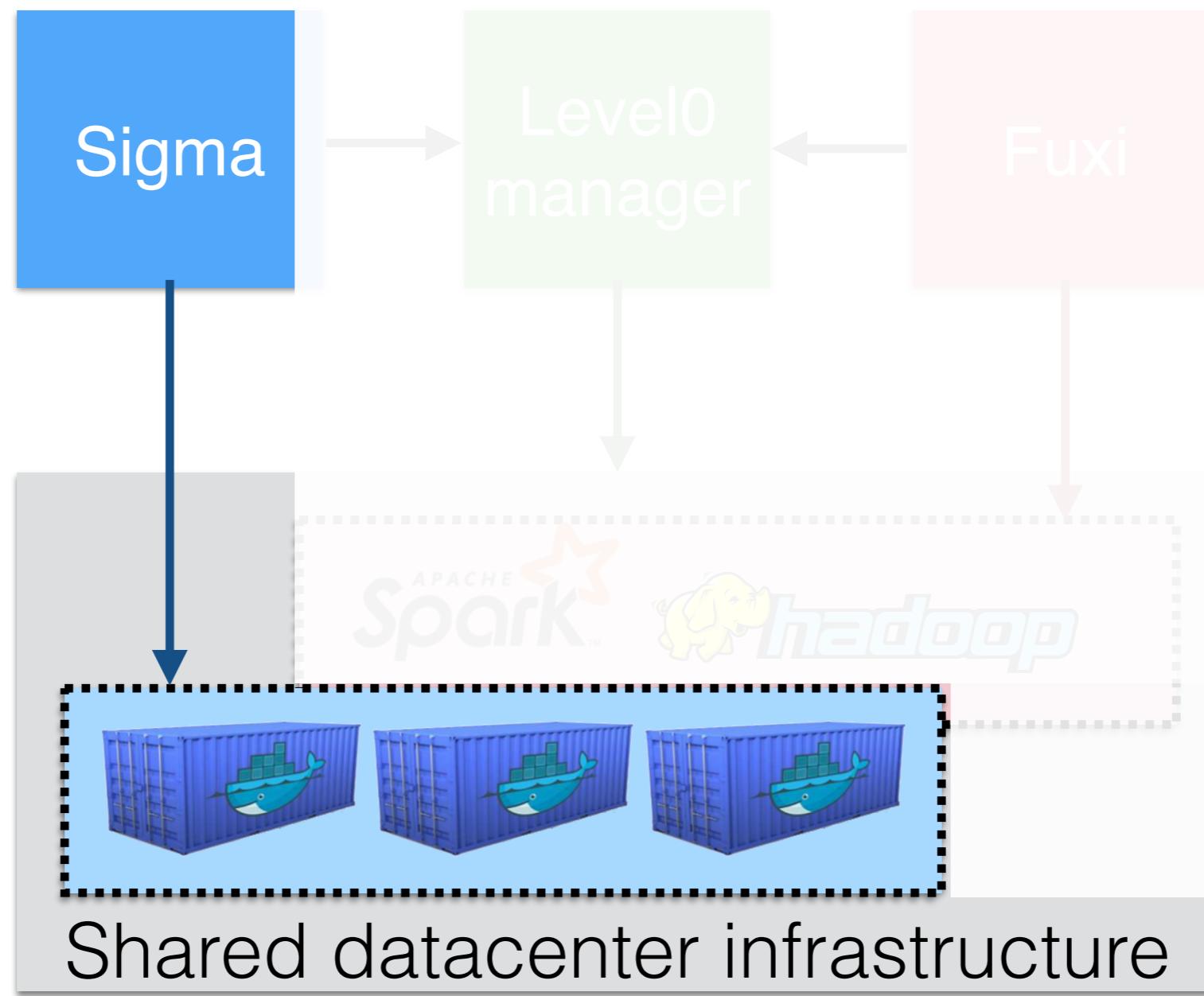


Average machine utilization

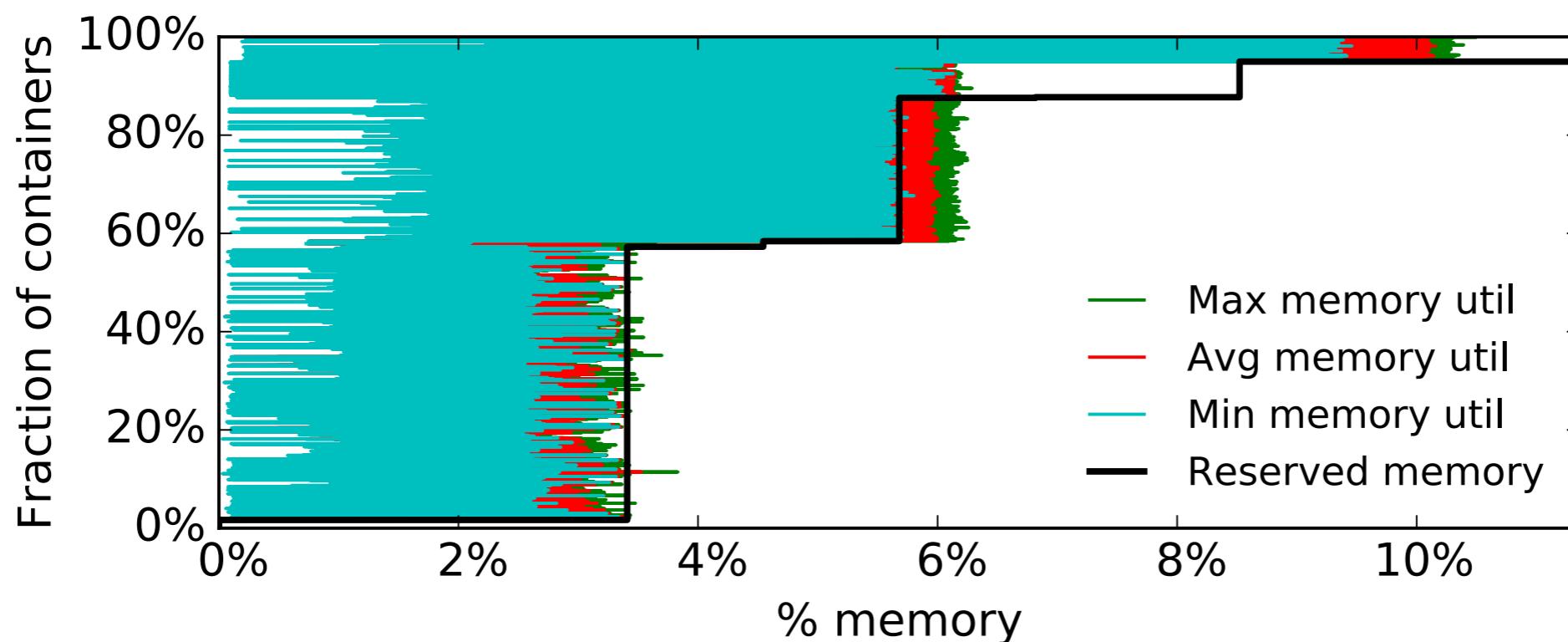
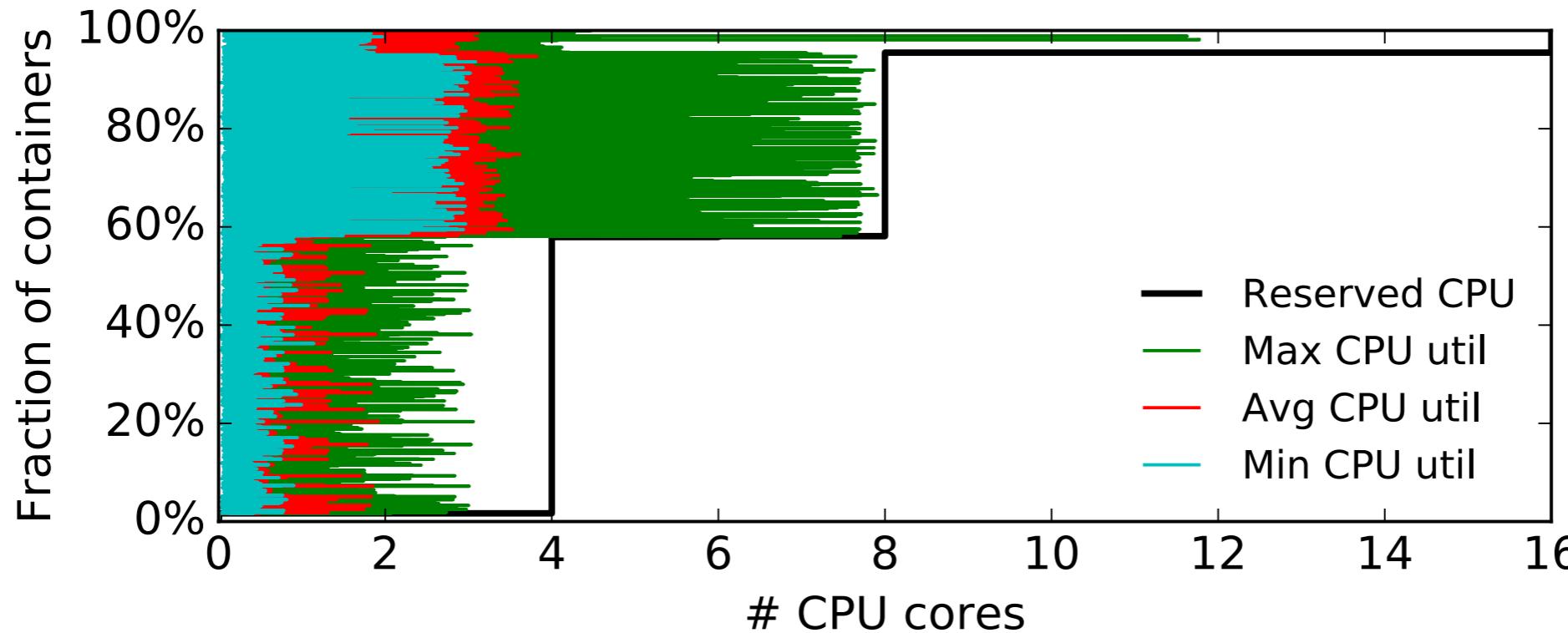
Memory tends to be of higher demands with over half capacity consumed over half the time



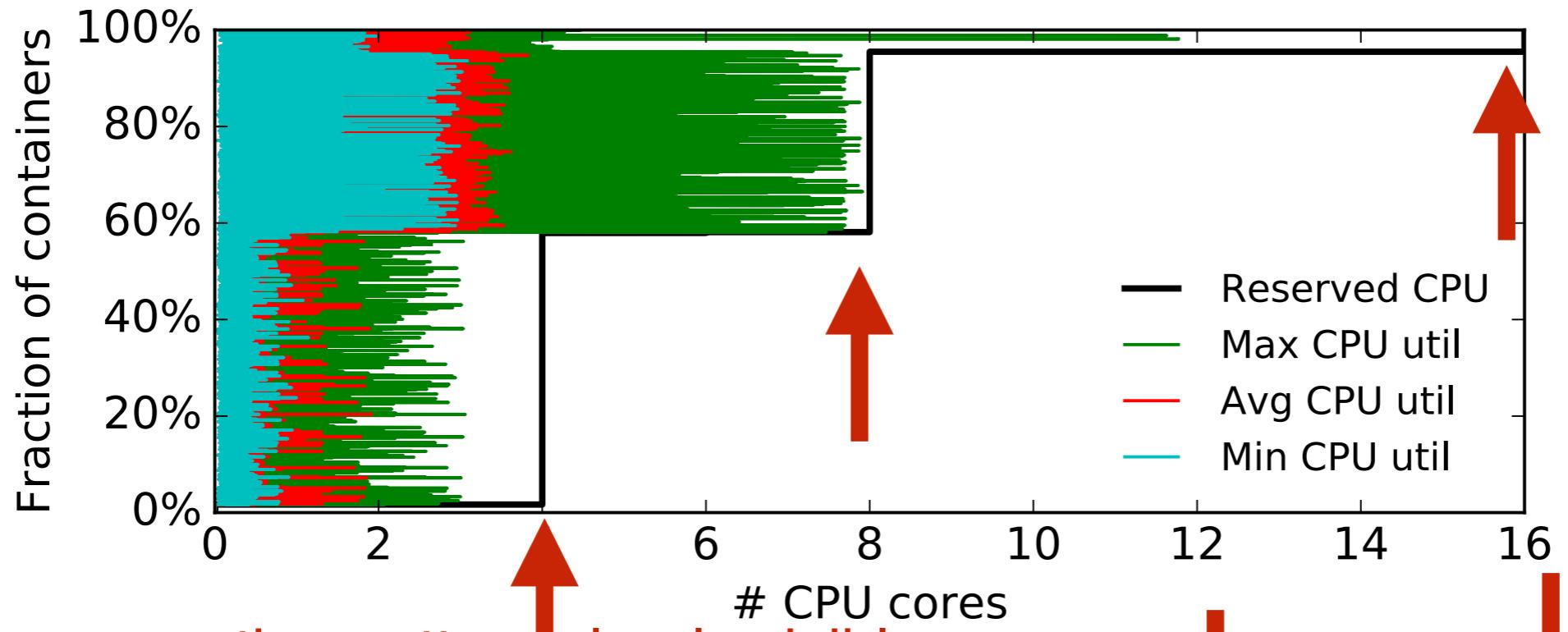
Long-running, containerized, online workloads



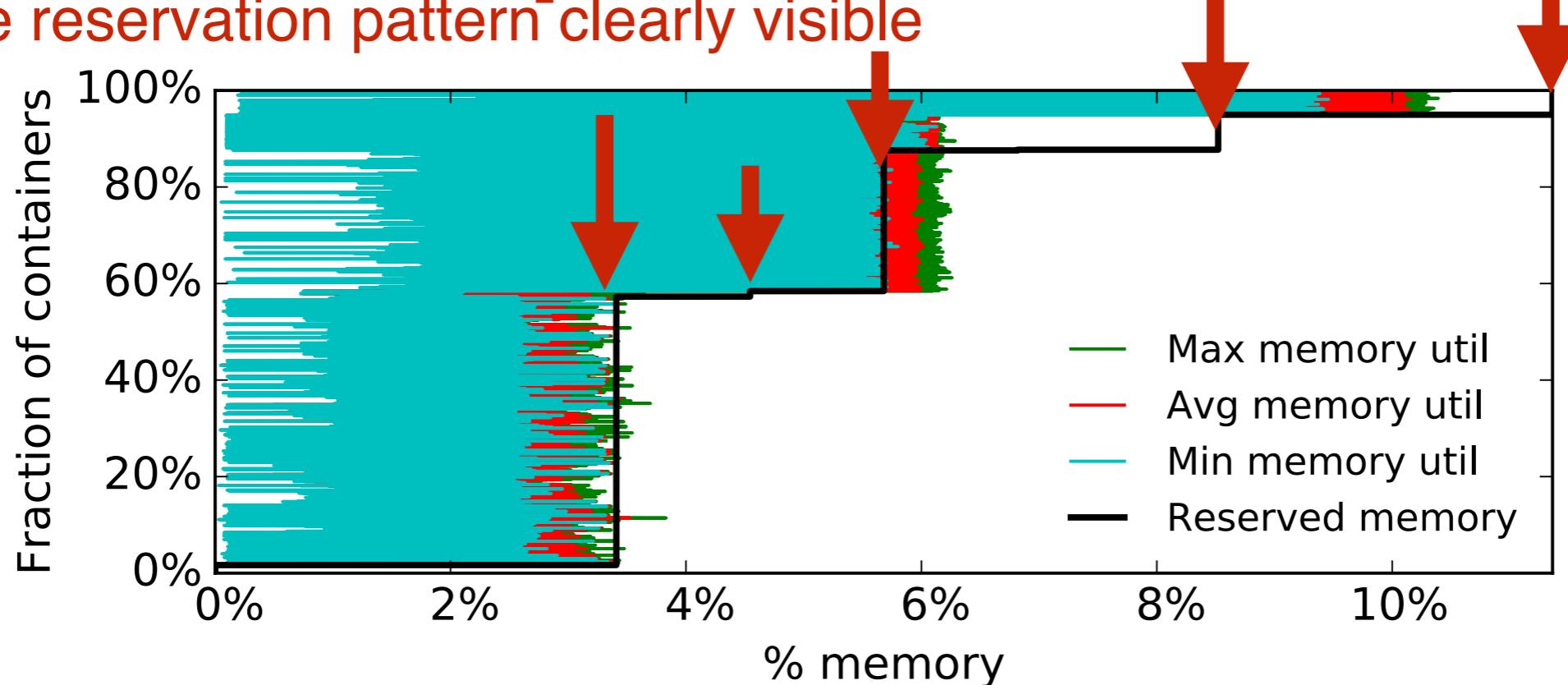
Long-running, online workload: Reserved resources vs. actual usage



Long-running, online workload: Reserved resources vs. actual usage

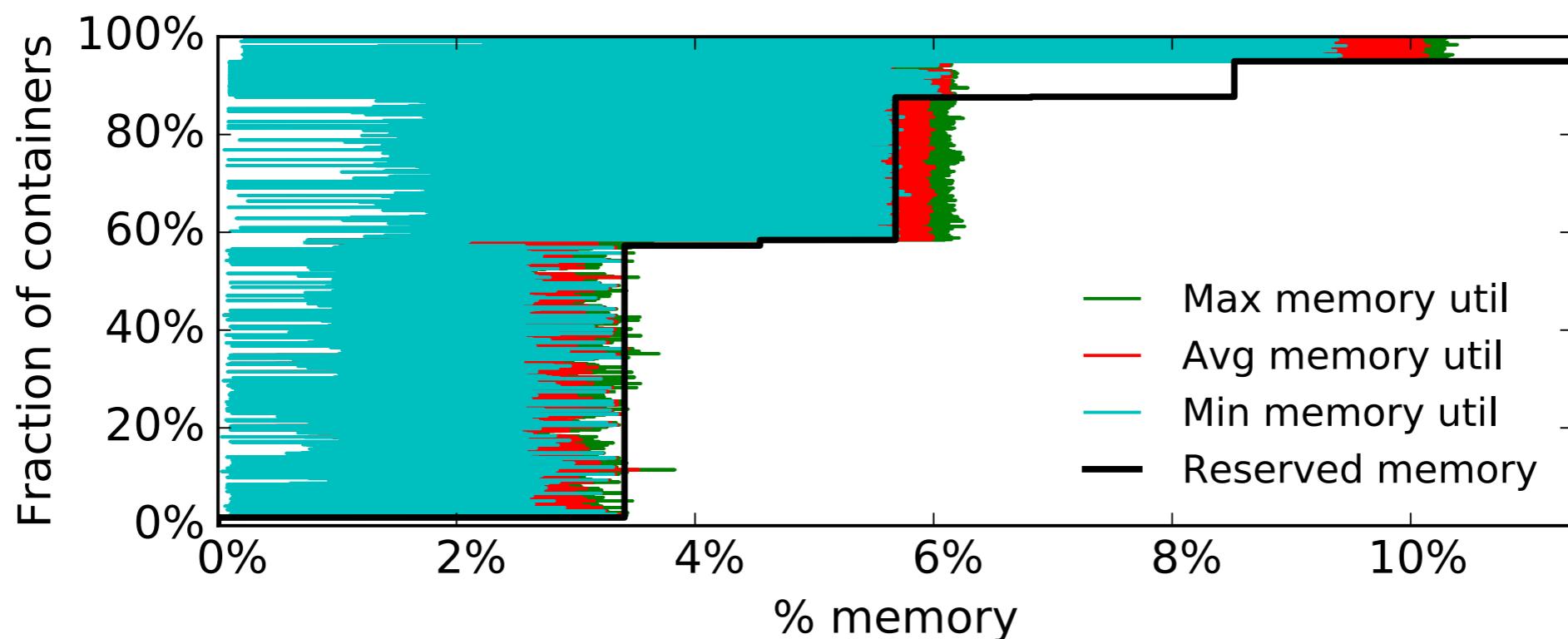
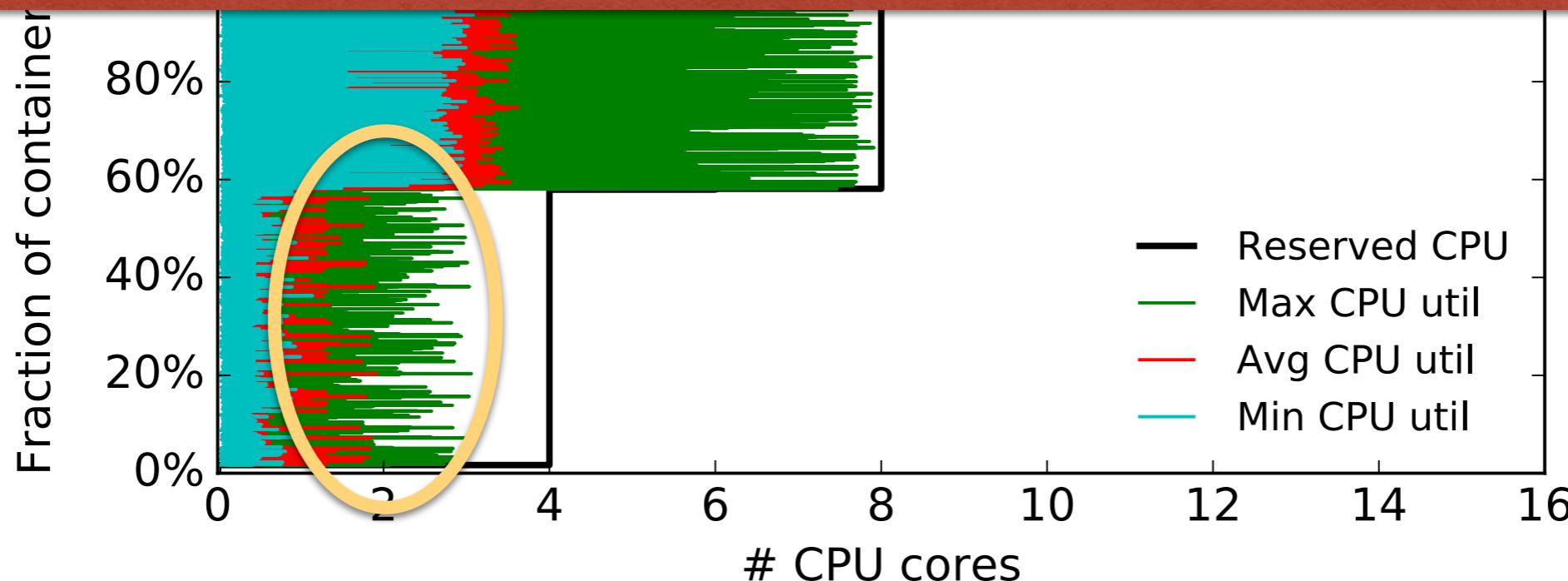


Resource reservation pattern clearly visible



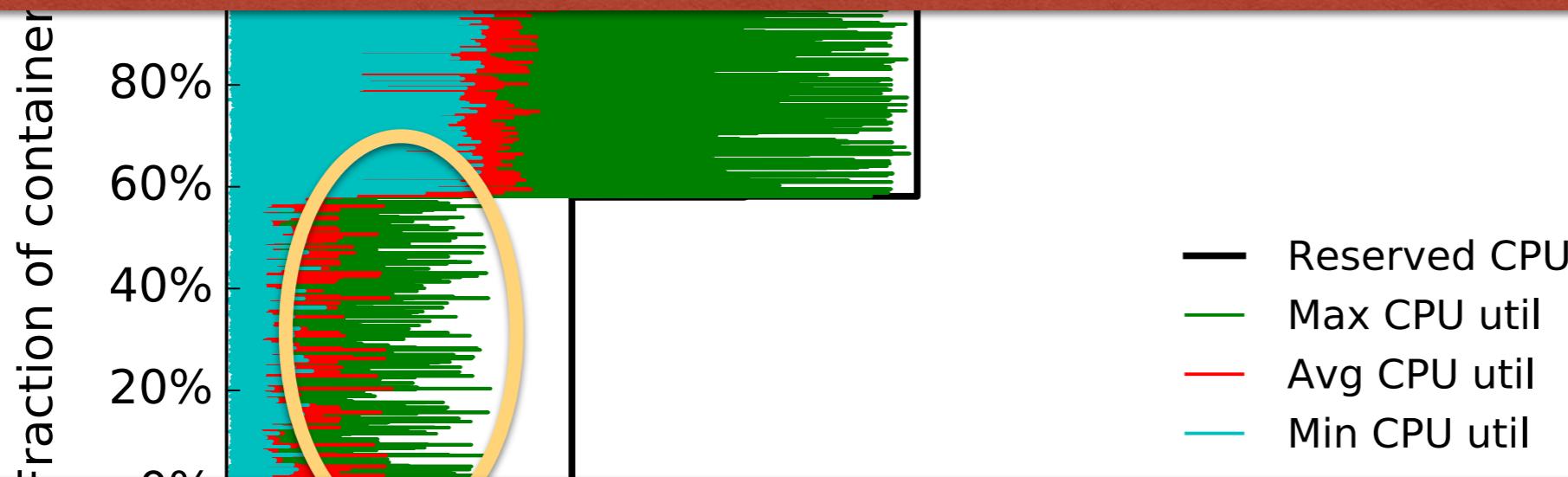
Long-running, online workload: Reserved resources vs. actual usage

Temporal dynamicity is not significant for half the containers

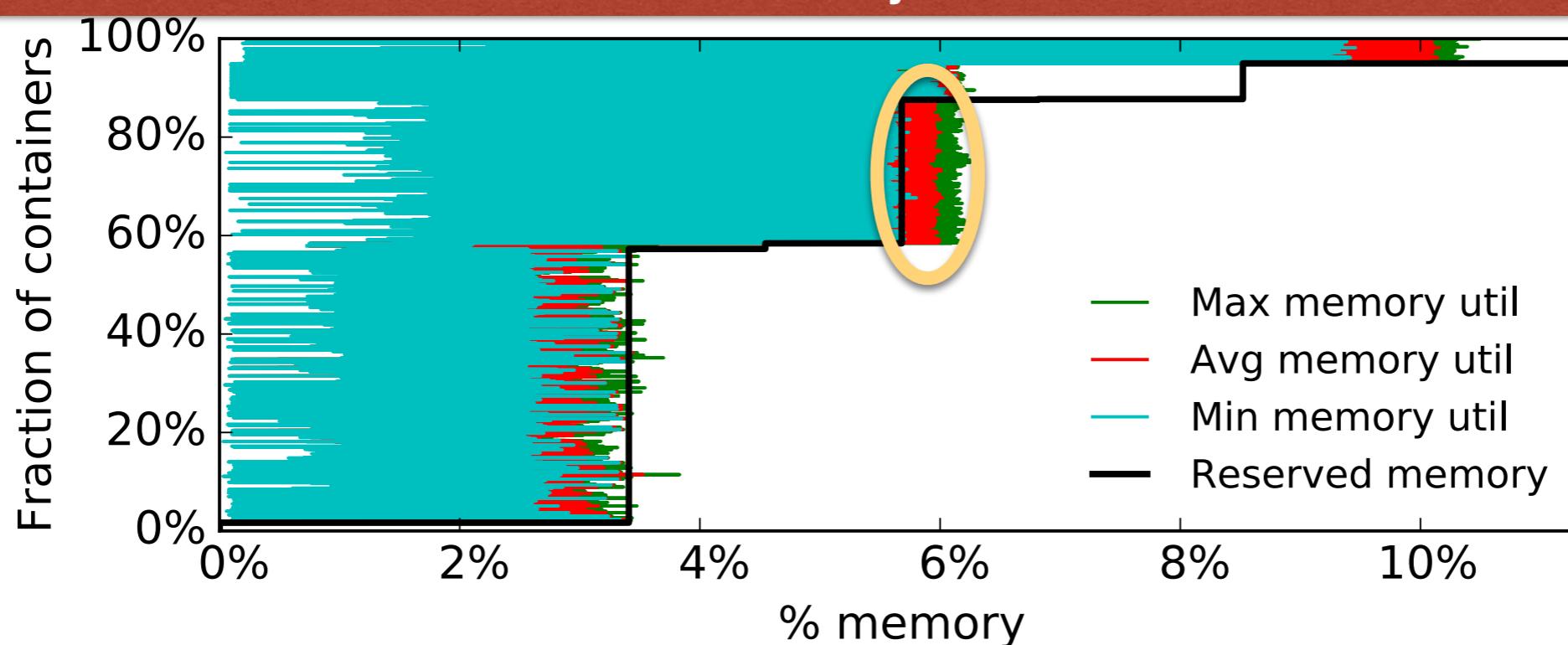


Long-running, online workload: Reserved resources vs. actual usage

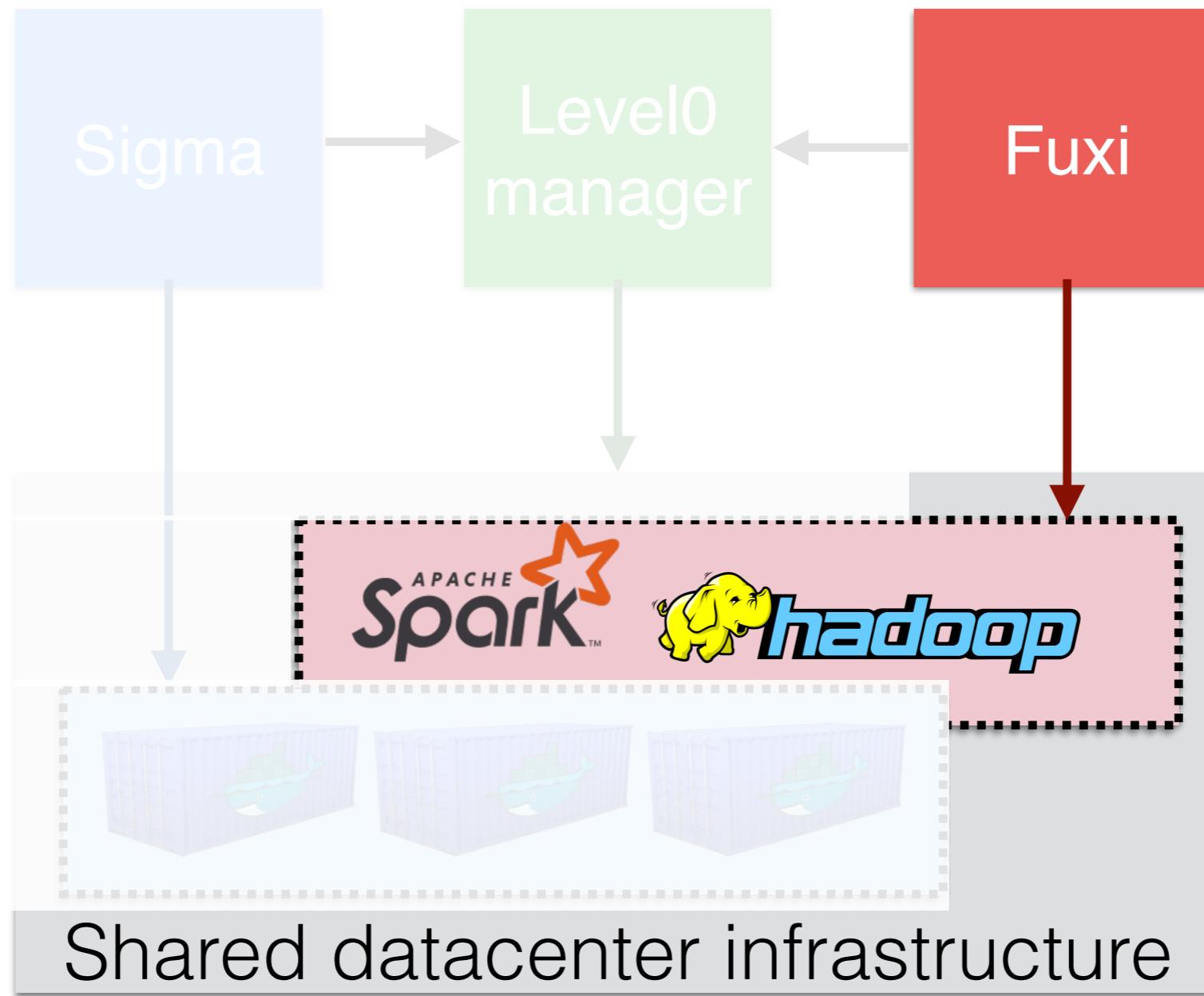
Temporal dynamicity is not significant for half the containers



Memory usage is more stable, and
a small fraction of containerized jobs overcommit memory

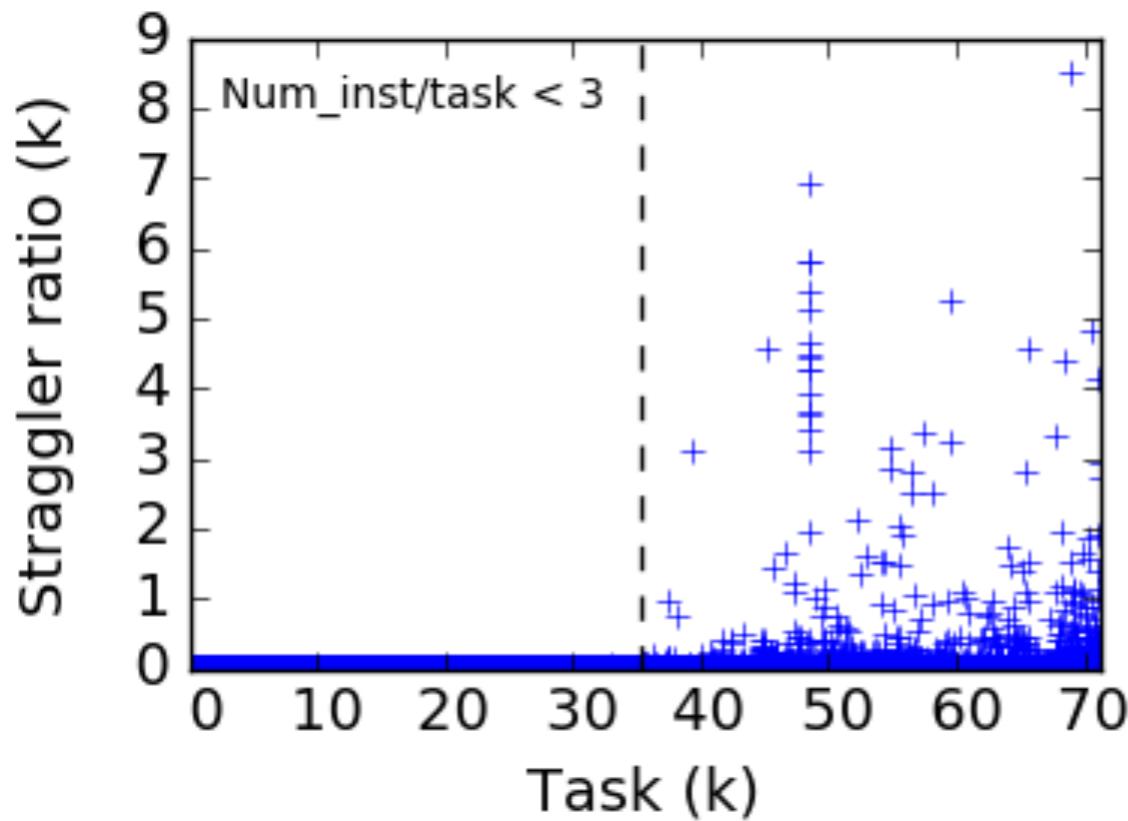


Transient, batch processing workloads



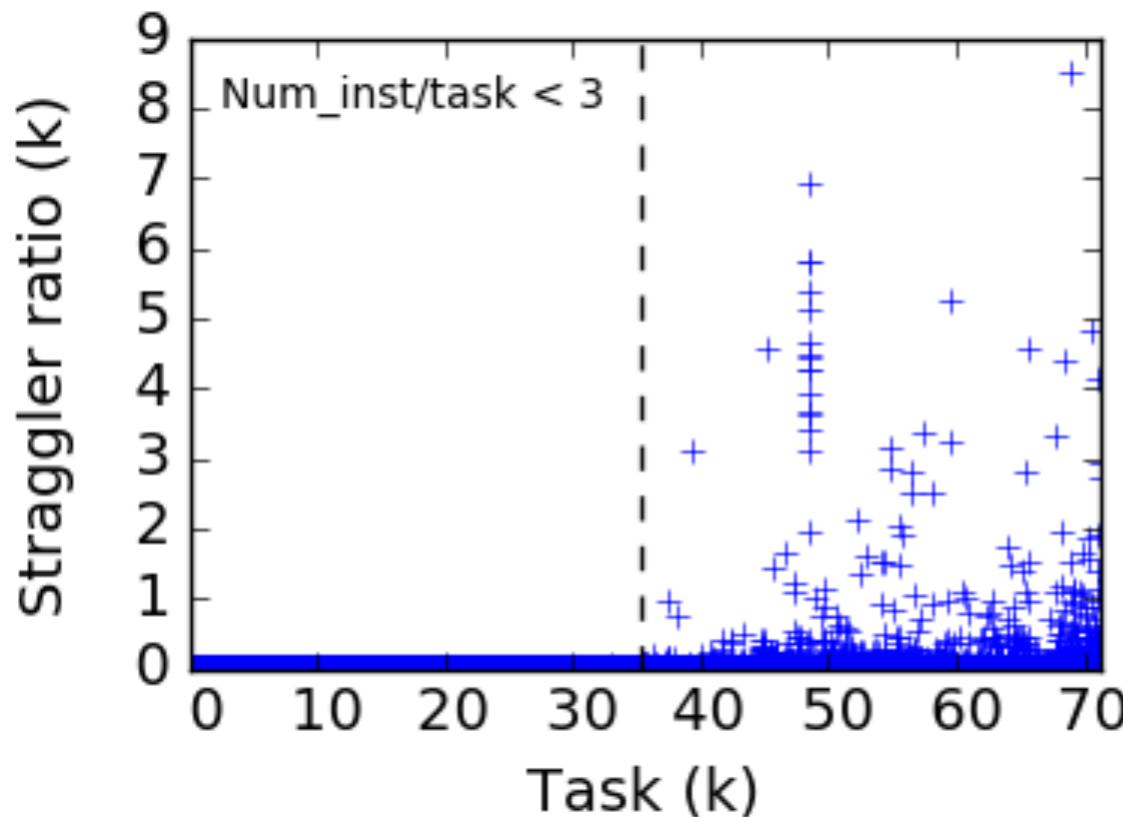
Straggler issues

Straggler issues still persist

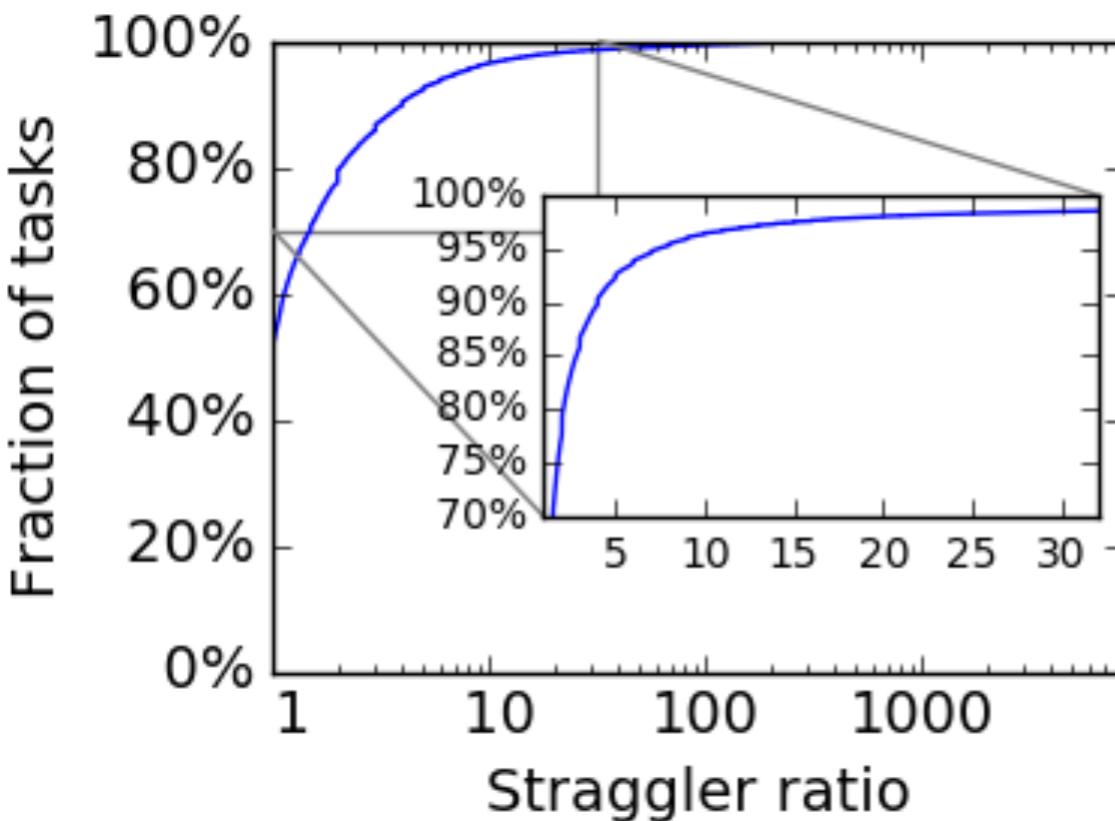


$\text{Straggler_ratio}_{\text{TaskN}} = \frac{\text{Maxmakespan}}{\text{Minmakespan}}$

Straggler issues still persist

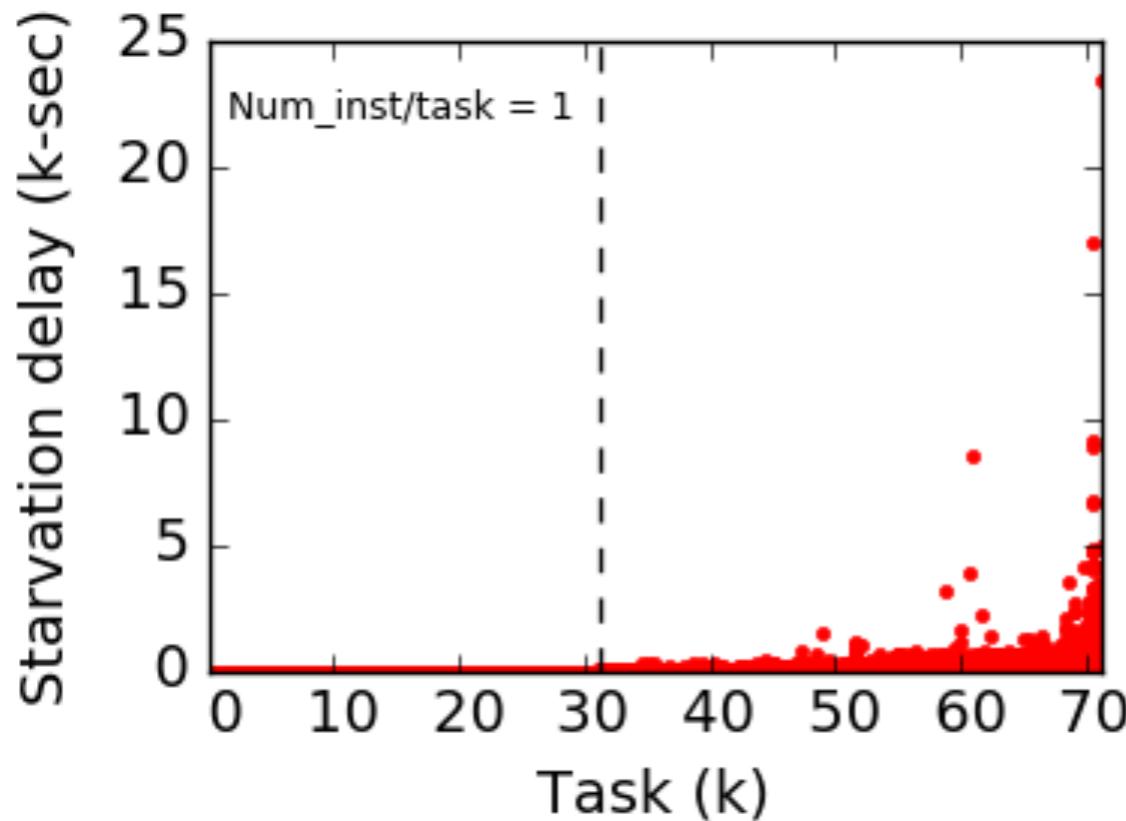


$\text{Straggler_ratio}_{\text{TaskN}} = \frac{\text{Maxmakespan}}{\text{Minmakespan}}$



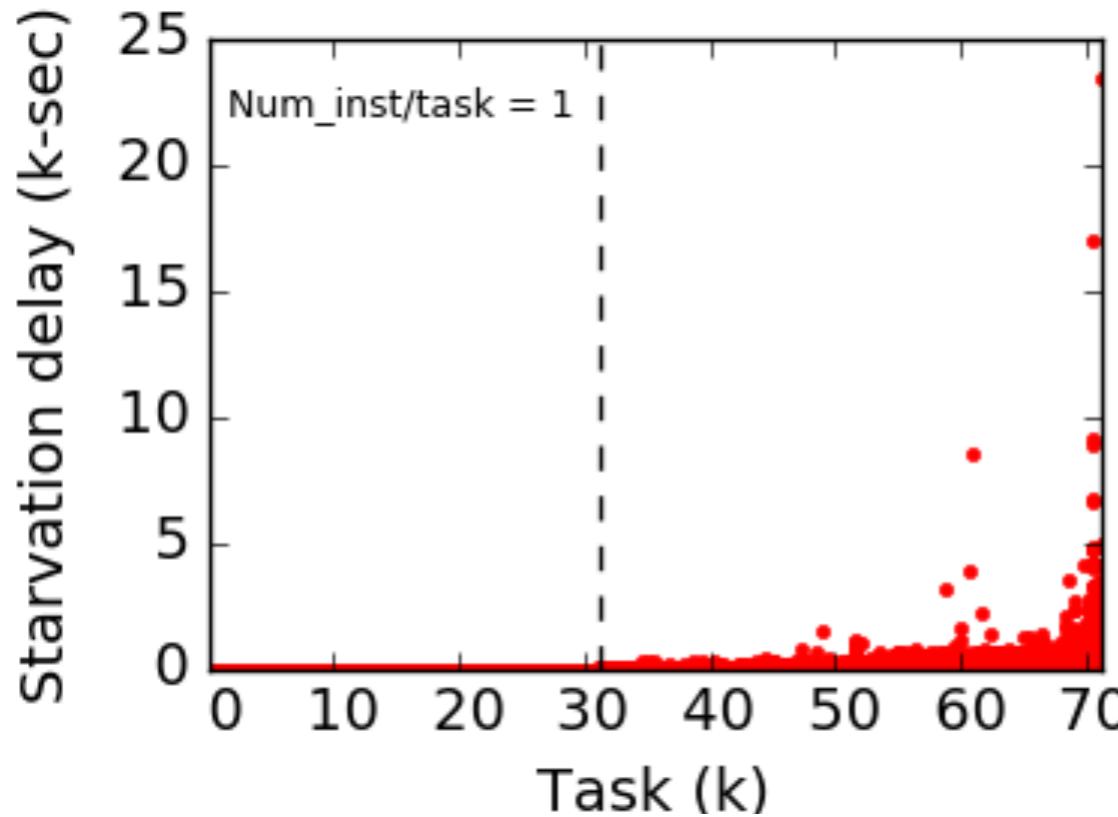
7% of tasks have a straggler ratio of >5X

Straggler issues still persist

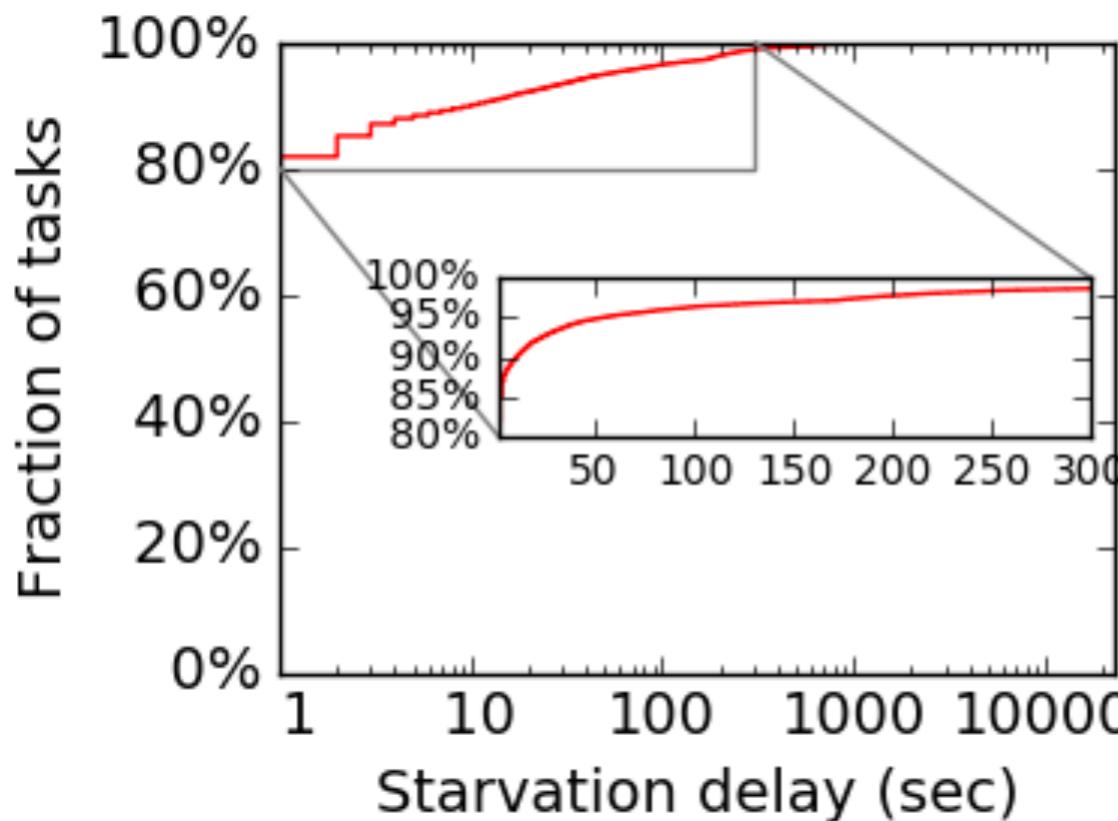


Starvation_delay_{TaskN} =
 $T_{\text{latest}} - T_{\text{earliest}}$

Straggler issues still persist

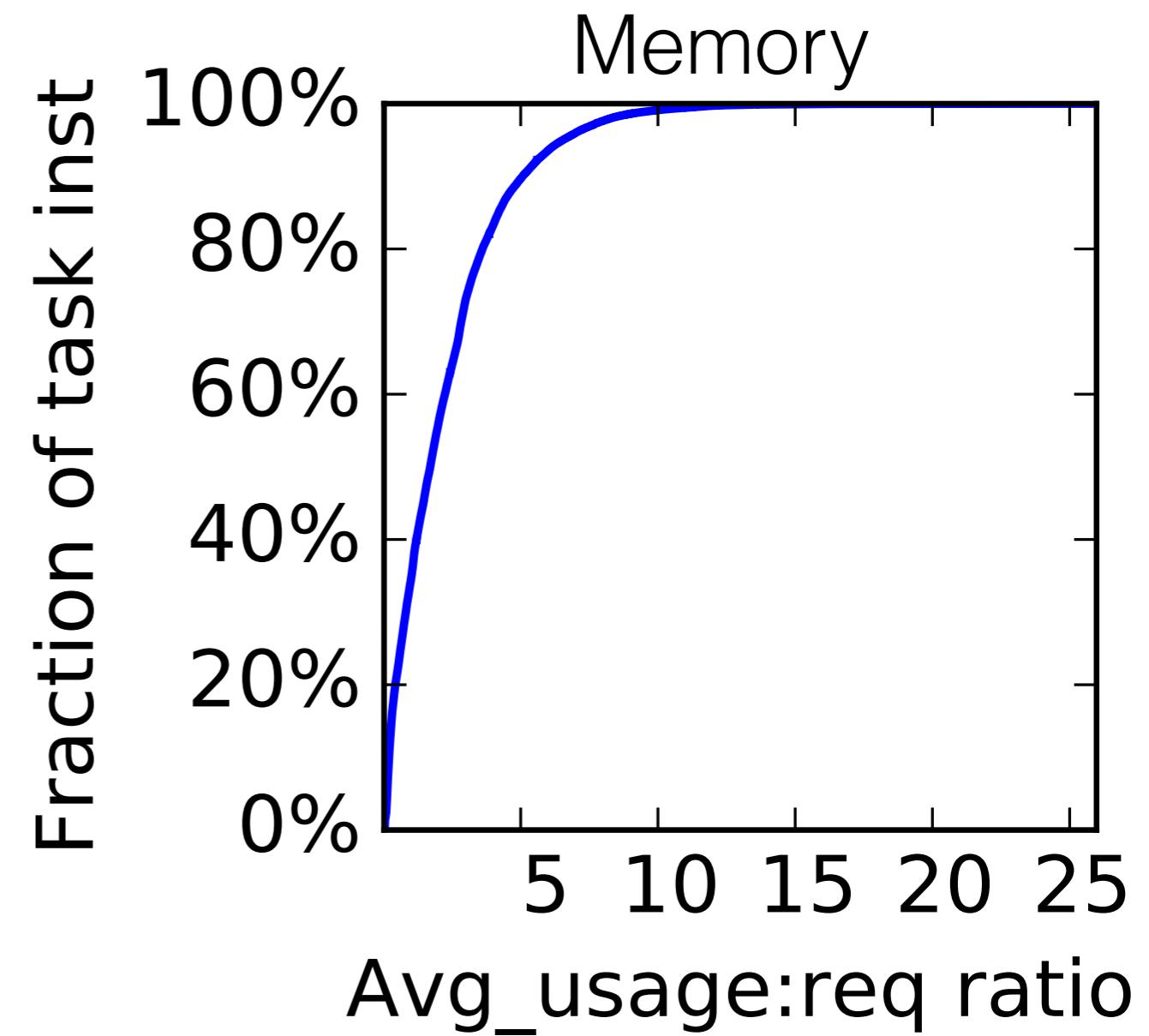
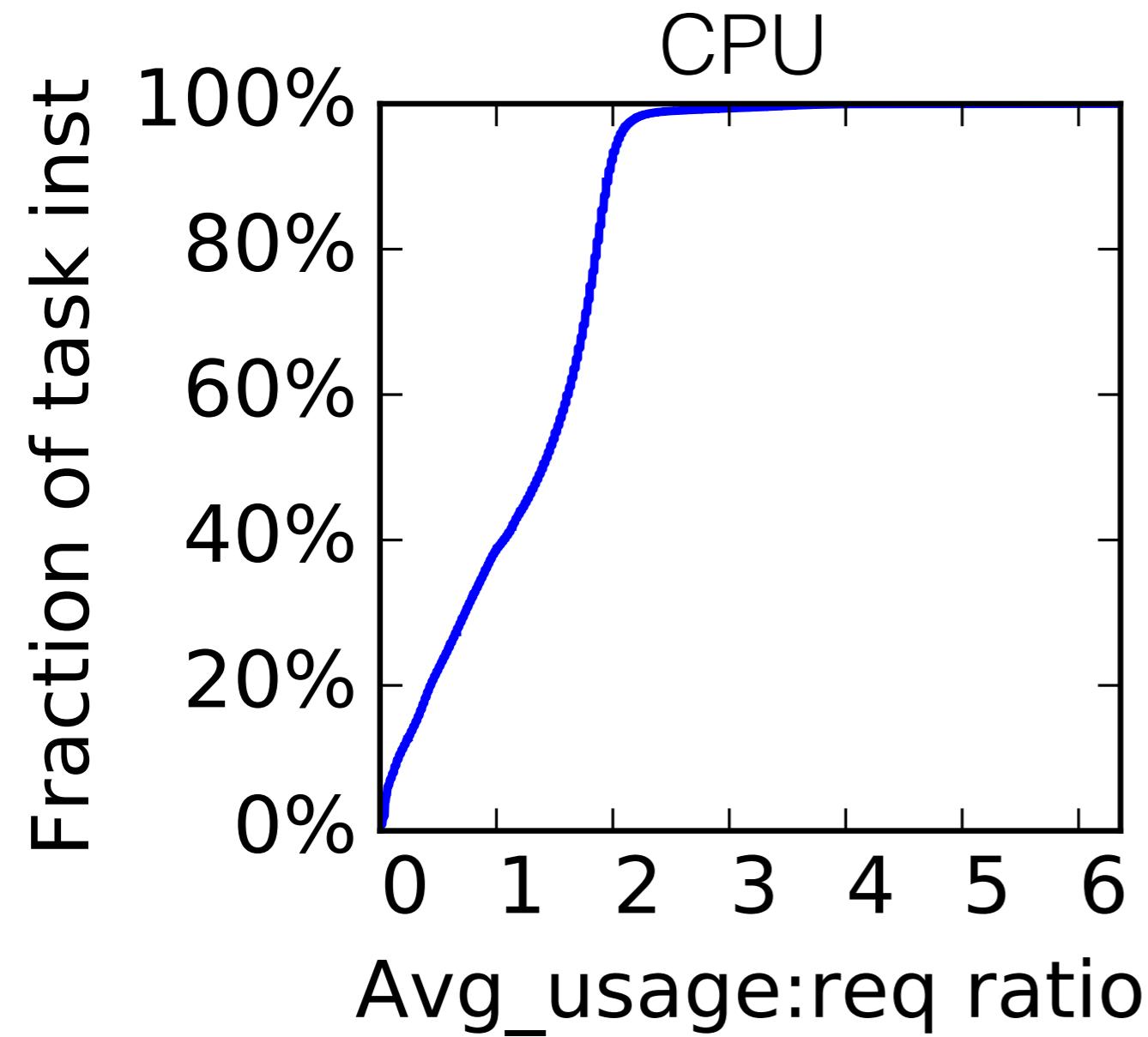


Starvation_delay_{TaskN} =
 $T_{\text{latest}} - T_{\text{earliest}}$

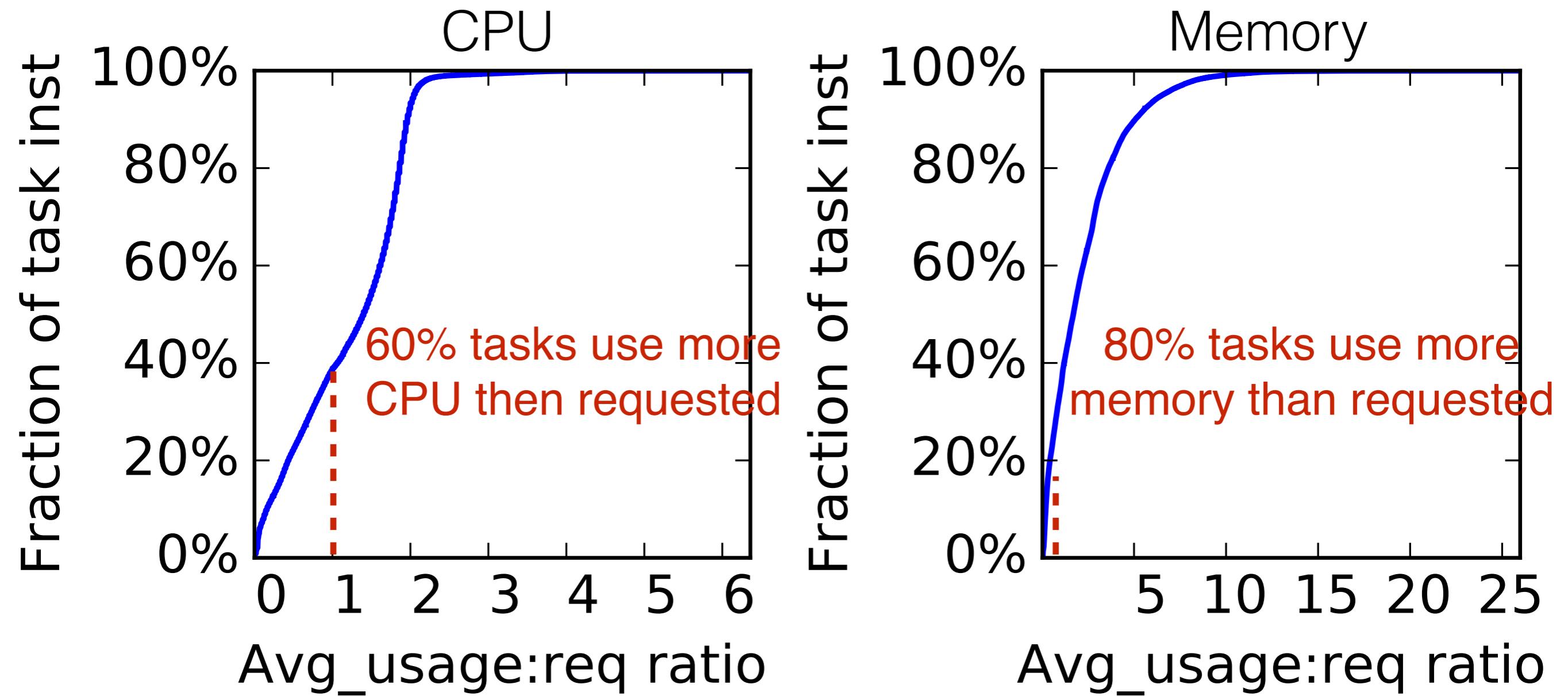


4% of tasks have a starvation delay of longer than 100 seconds

Transient, batch processing workloads: Requested resource vs. average usage

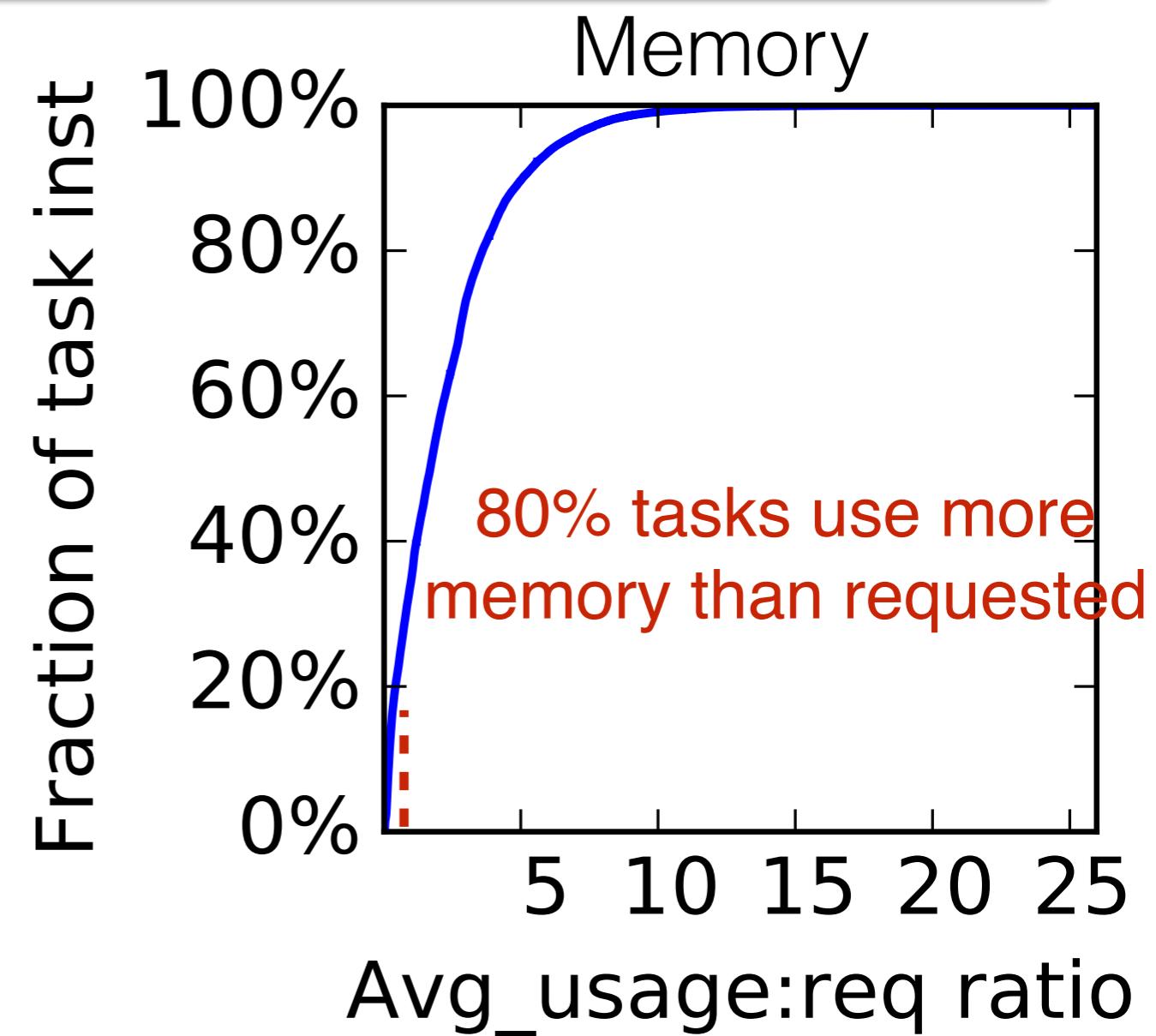
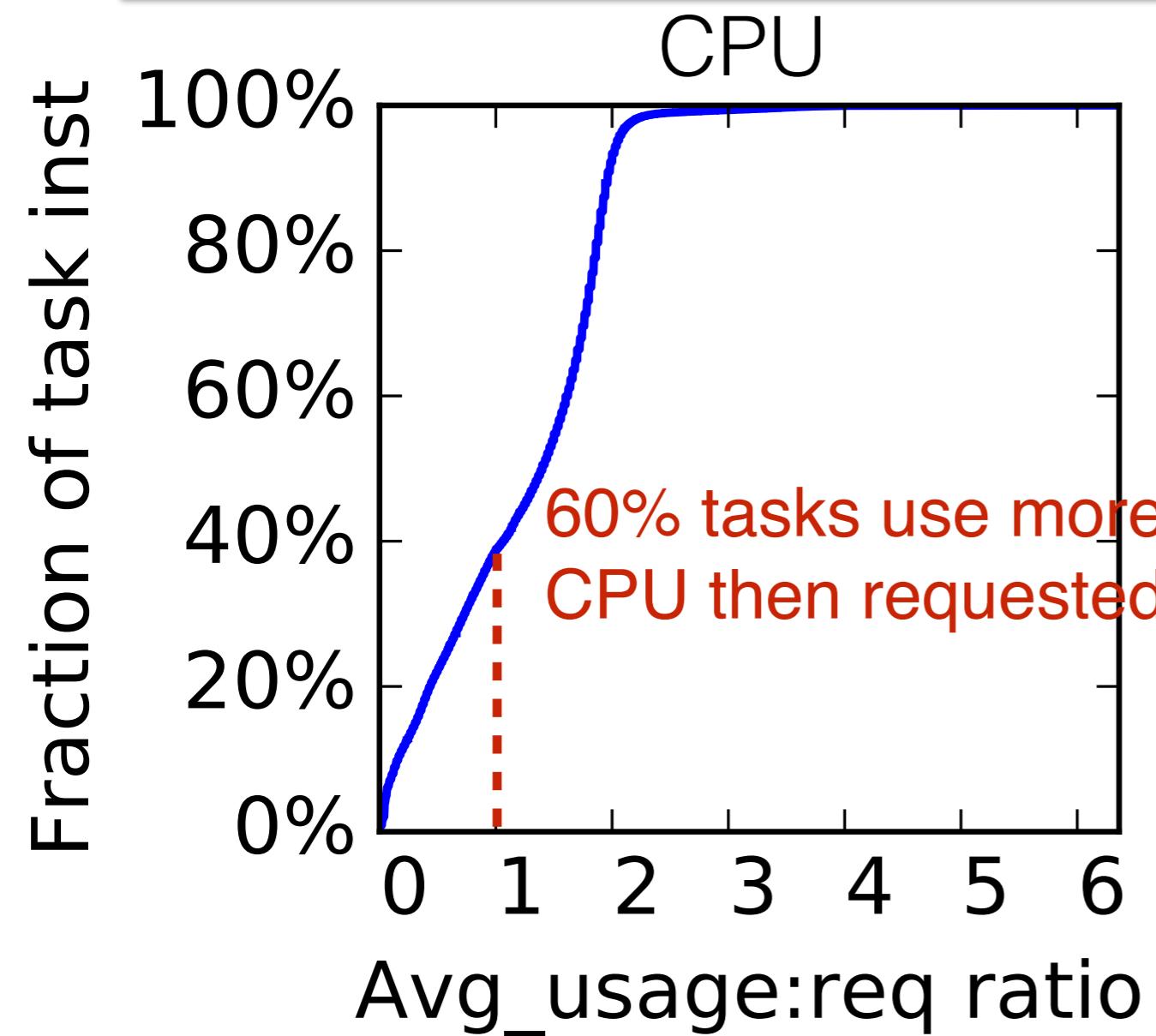


Transient, batch processing workloads: Requested resource vs. average usage



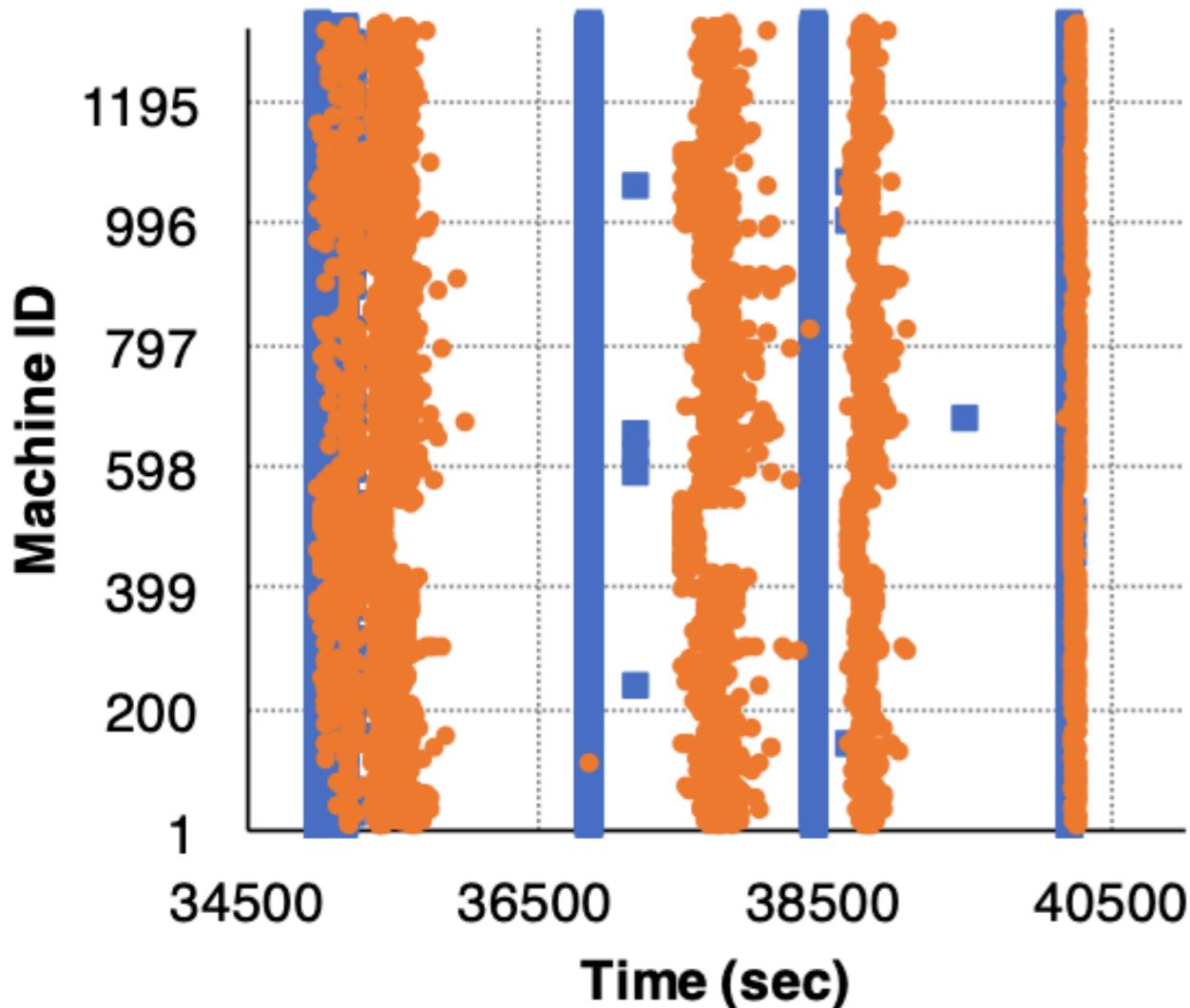
Transient, batch processing workloads: Requested resource vs. average usage

Batch jobs elastically overcommit resources reserved but
not yet consumed by containerized online services



Straggler examples

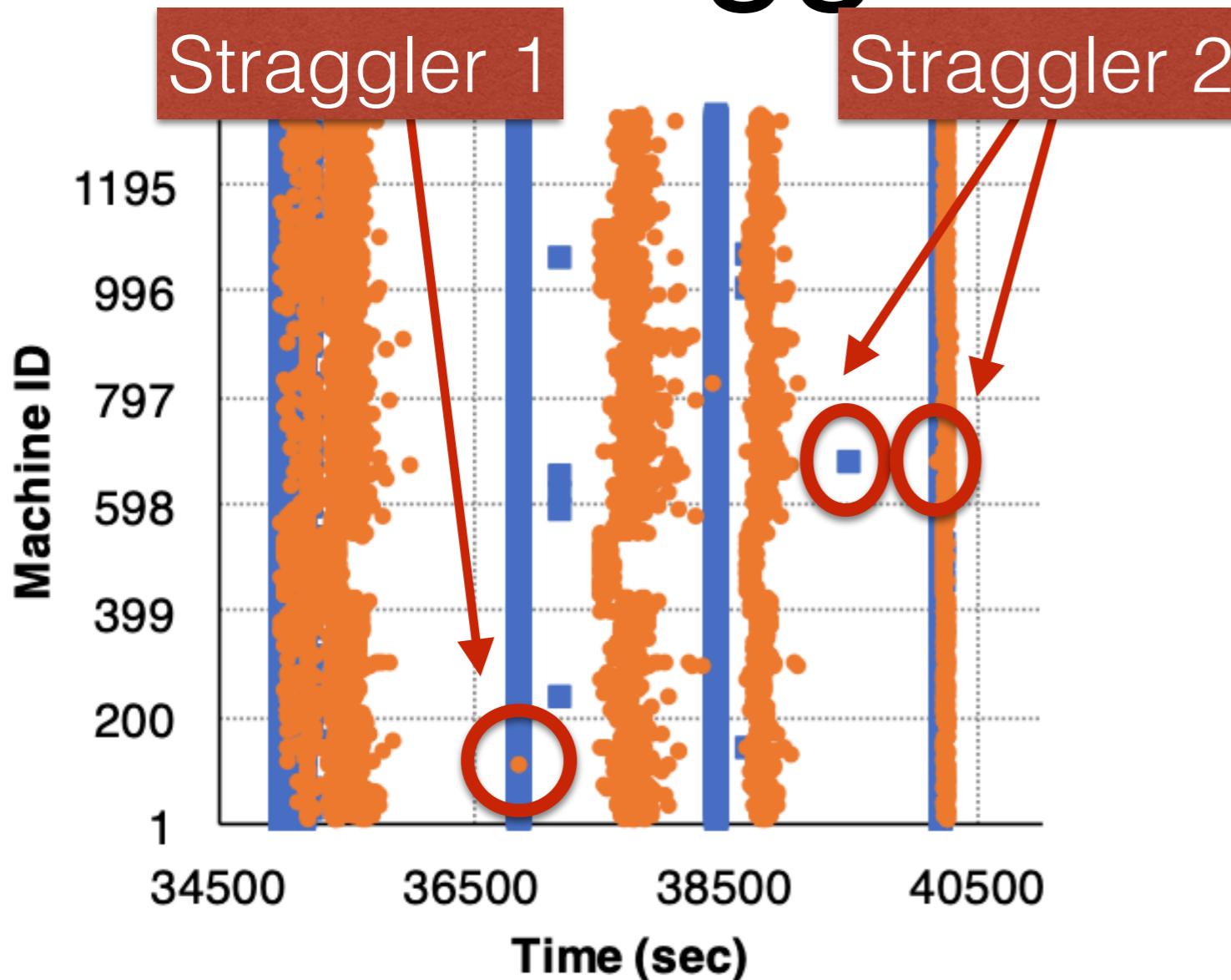
Straggler examples



■ The start of a task's makespan

● The end of a task's makespan

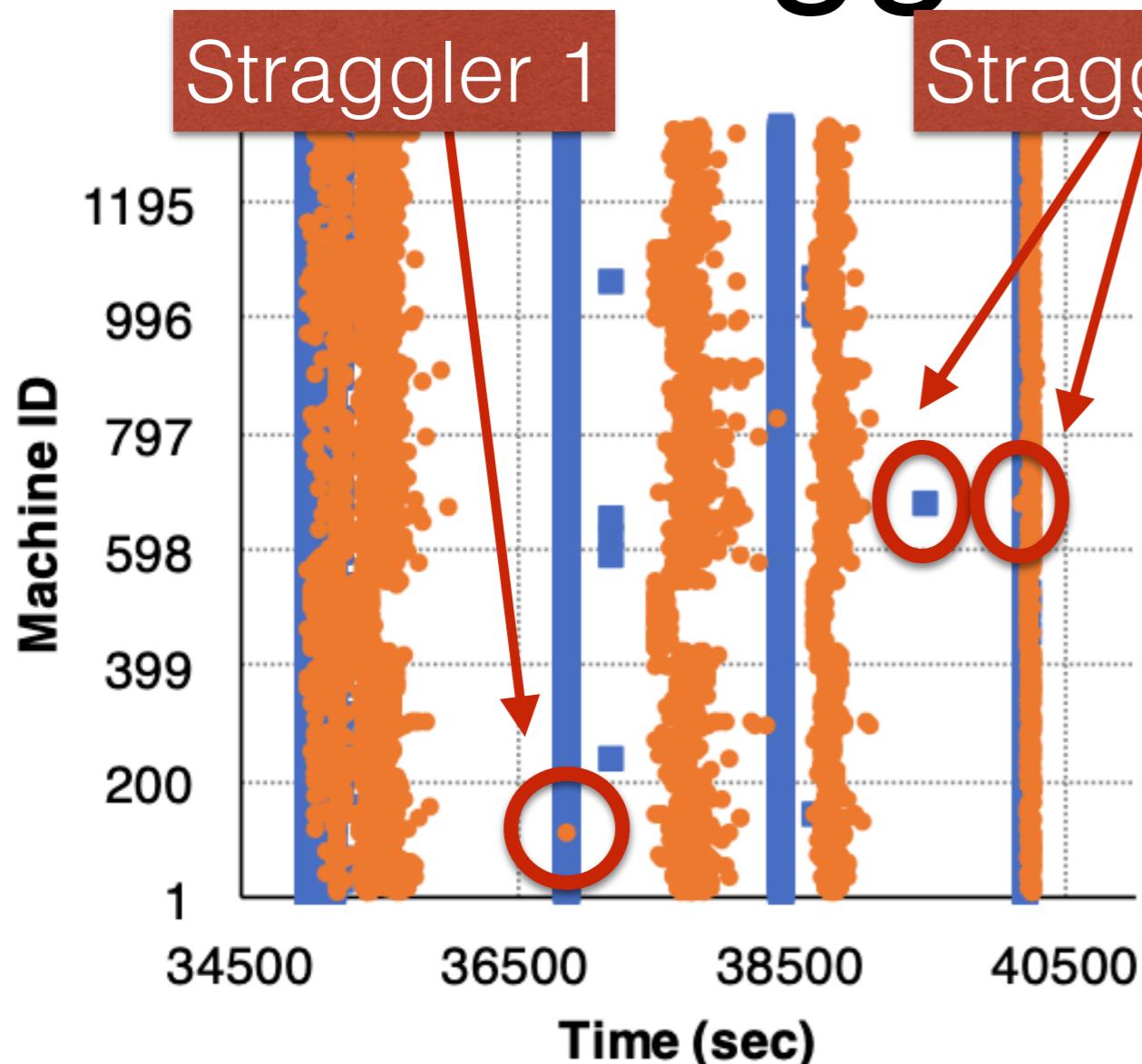
Straggler examples



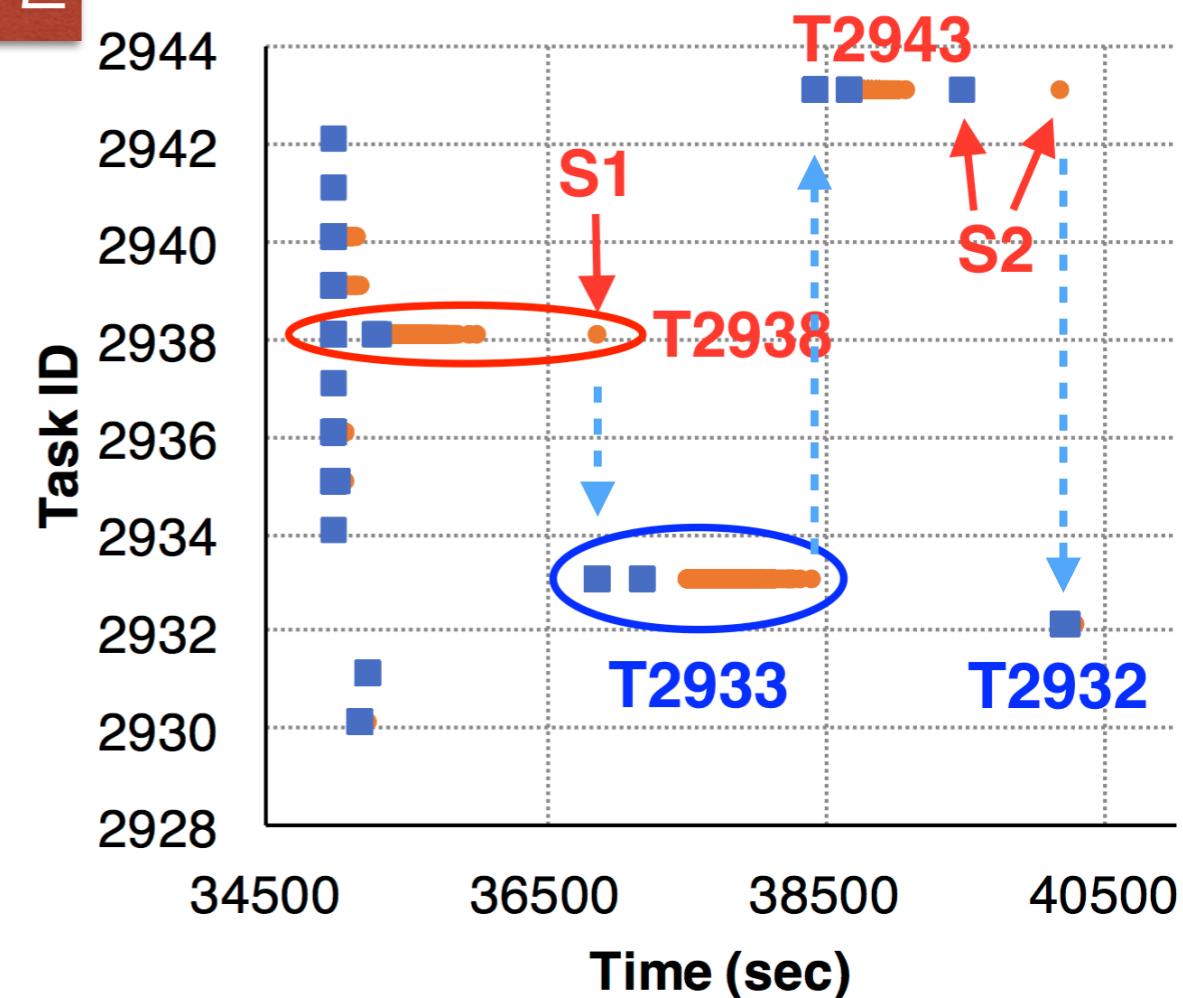
■ The start of a task's makespan

● The end of a task's makespan

Straggler examples



- The start of a task's makespan
- The end of a task's makespan



T2932 depends on T2943
T2943 depends on T2933
T2933 waits for all tasks in that waive to complete
T2938 marks the completion of the wave

Takeaways

Alibaba's co-located workloads tend to be more memory-demanding

- Cluster spends over **80% time w/ 10-30% CPU usage**

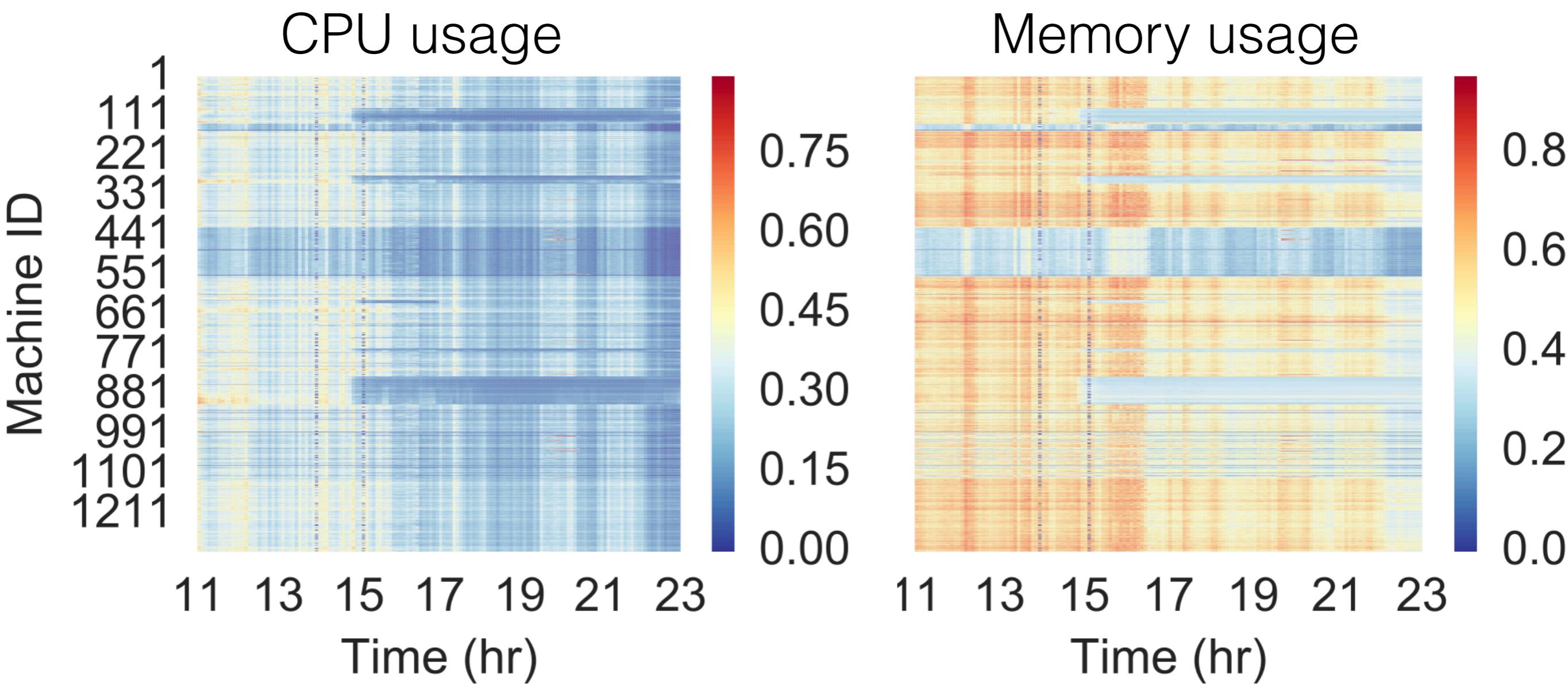
Long-running containerized jobs are mostly **idle**

Seemingly disjoint scheduling decision making regardless of the co-existence of two workloads

- Need for a more integrated Level-0 controller

Backup slides

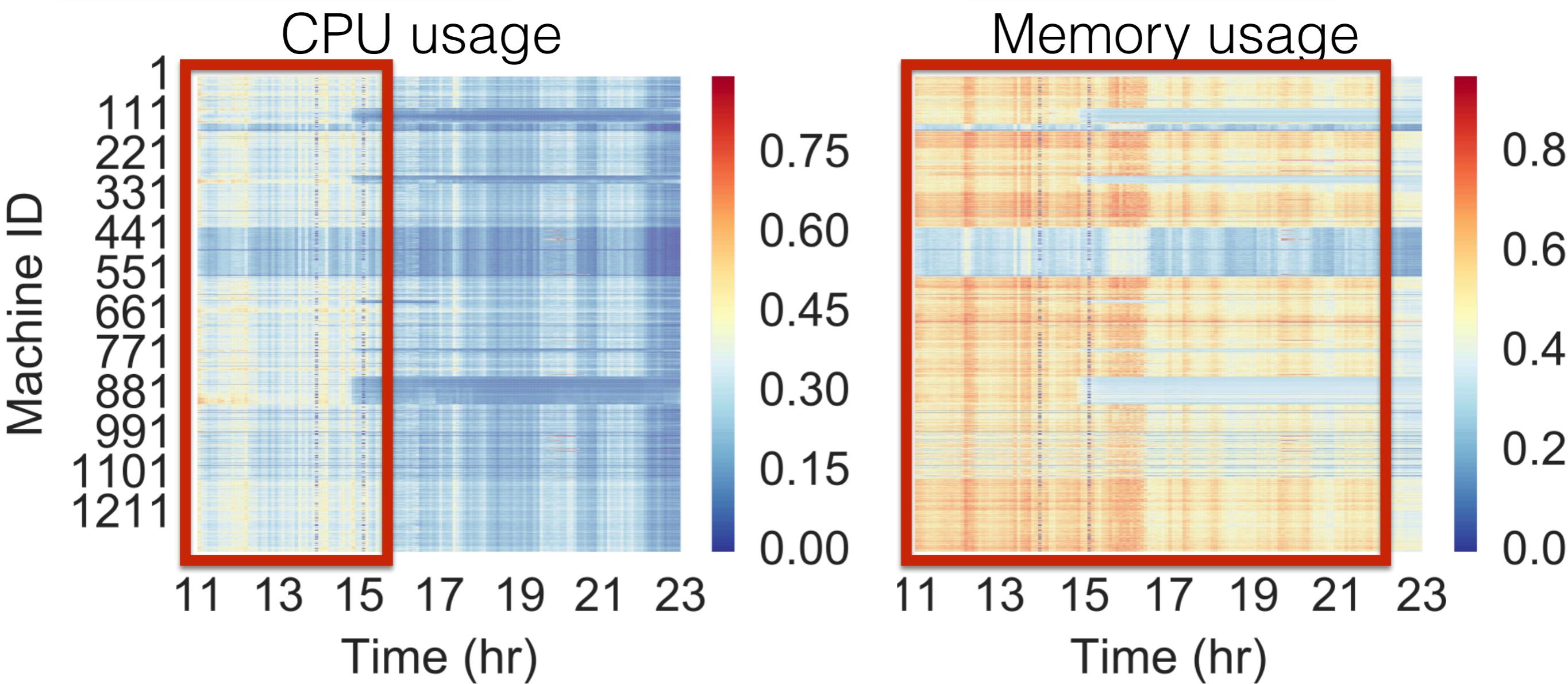
Overall cluster usage heatmap



Overall cluster usage heatmap

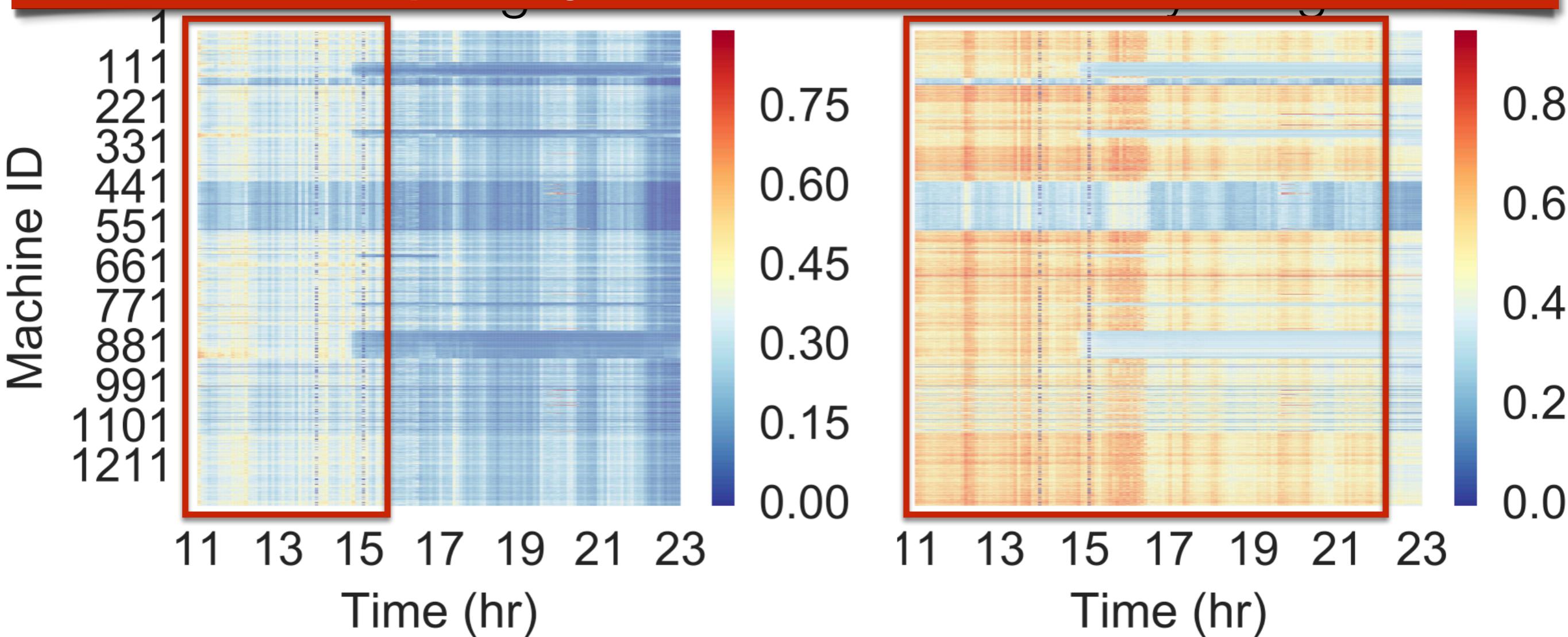
Medium usage for
the 1st 4 hours

> 50% for
majority of time

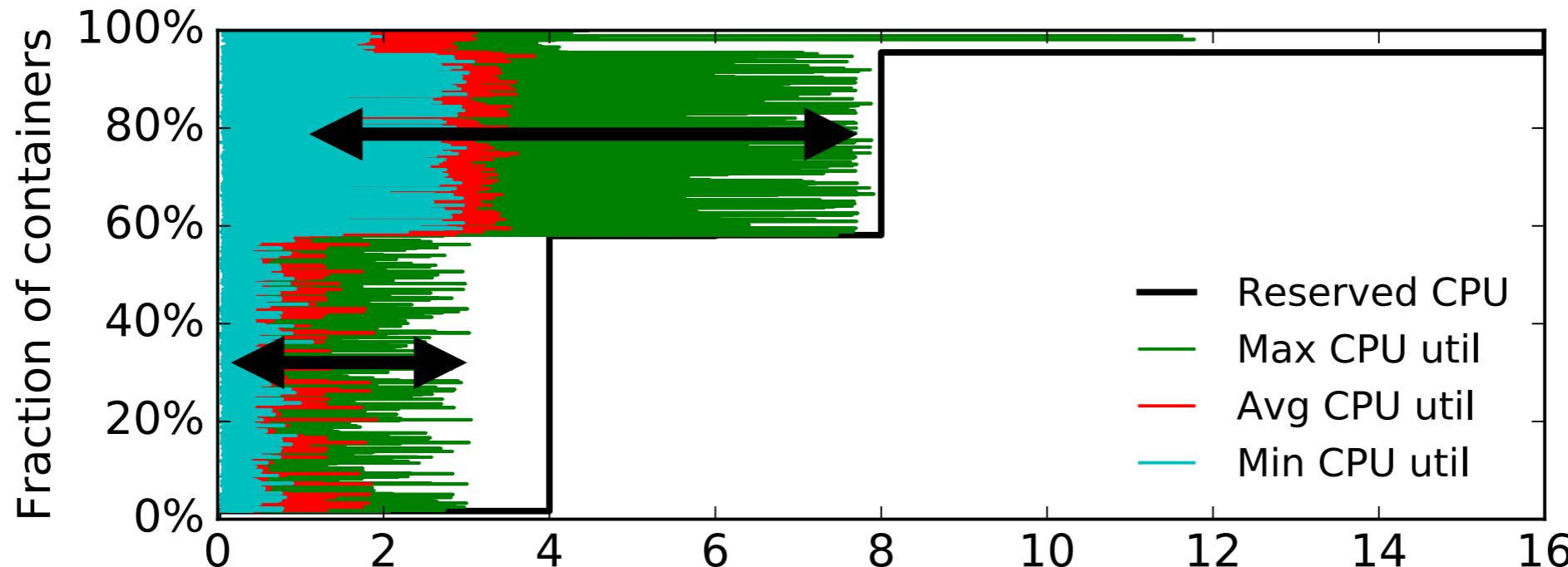


Overall cluster usage heatmap

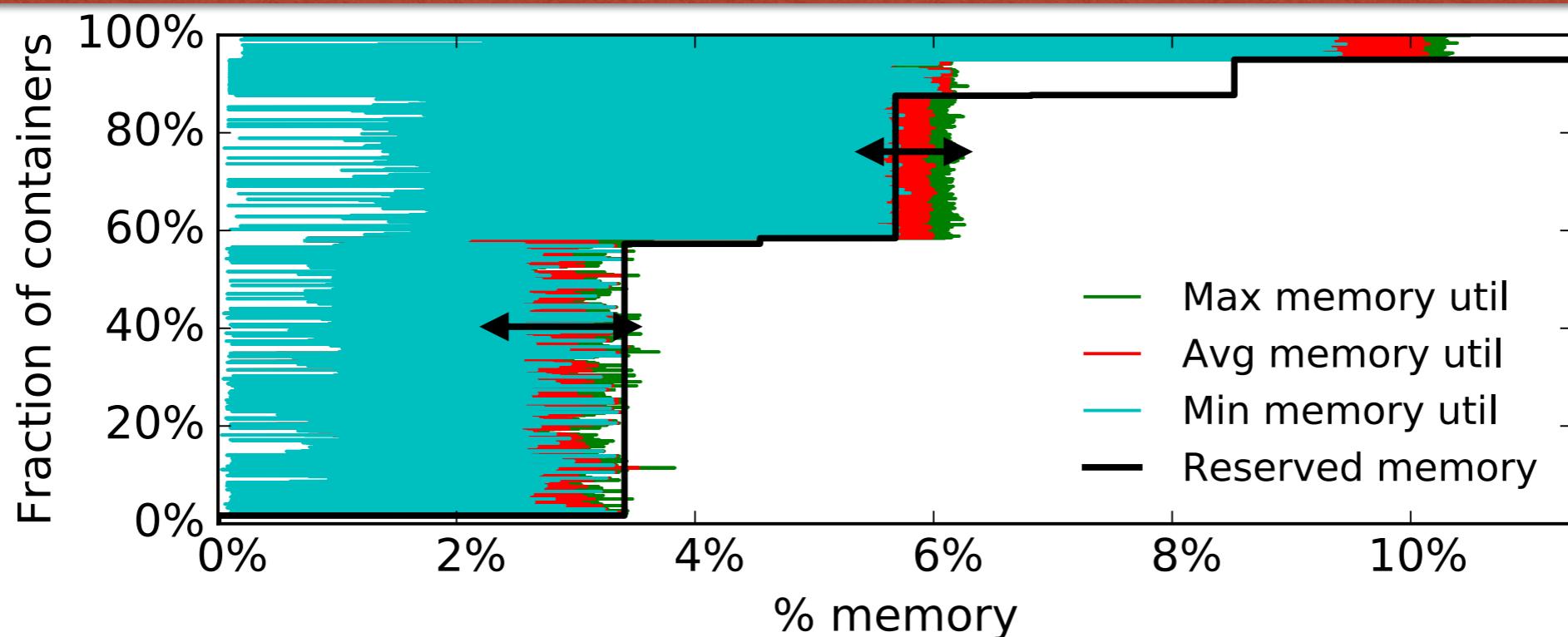
Memory tends to be of higher demands with over half capacity consumed over half the time



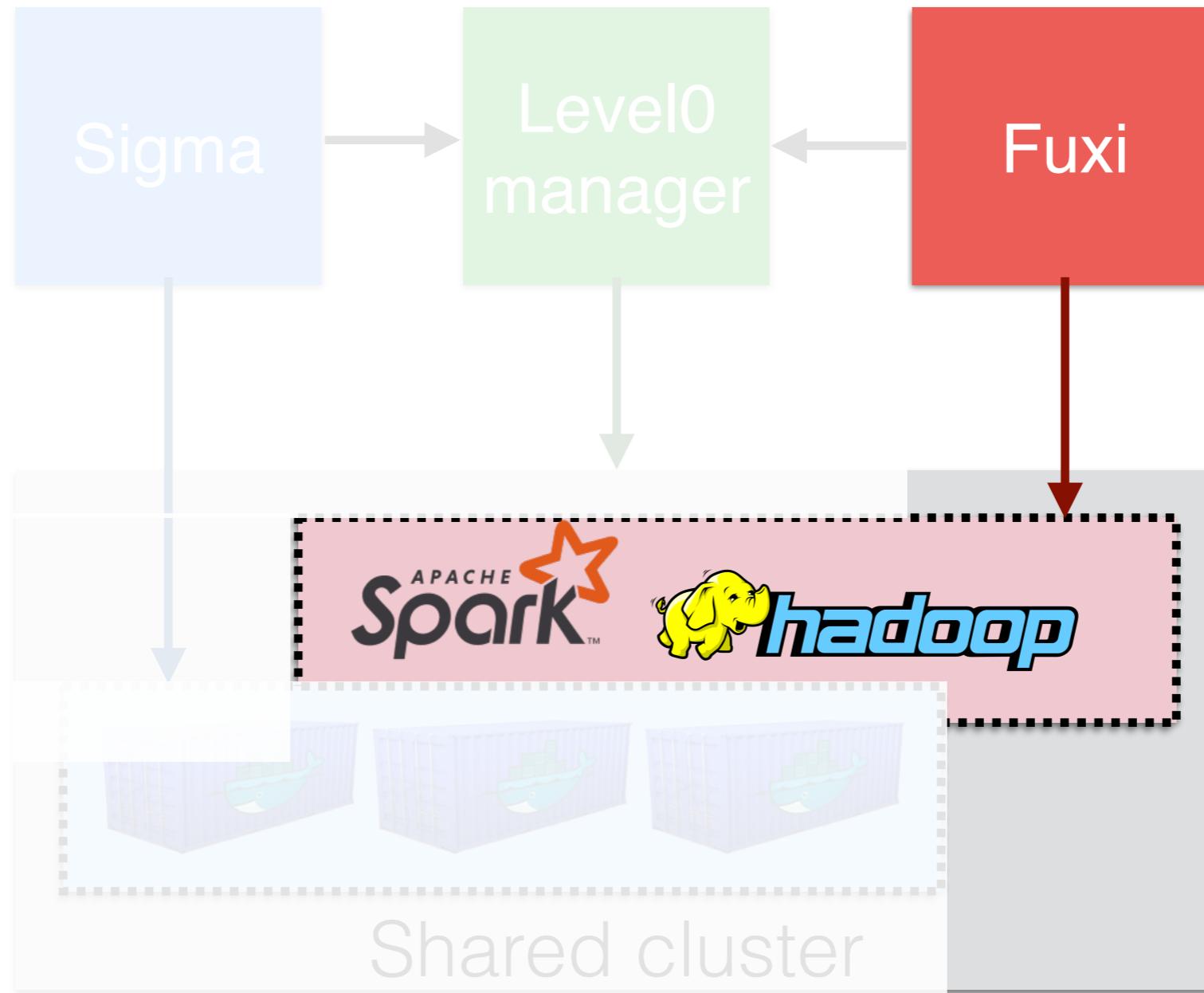
Long-running, containerized workload: Reserved resources vs. actual usage



Most long-running containerized jobs are not resource dynamic

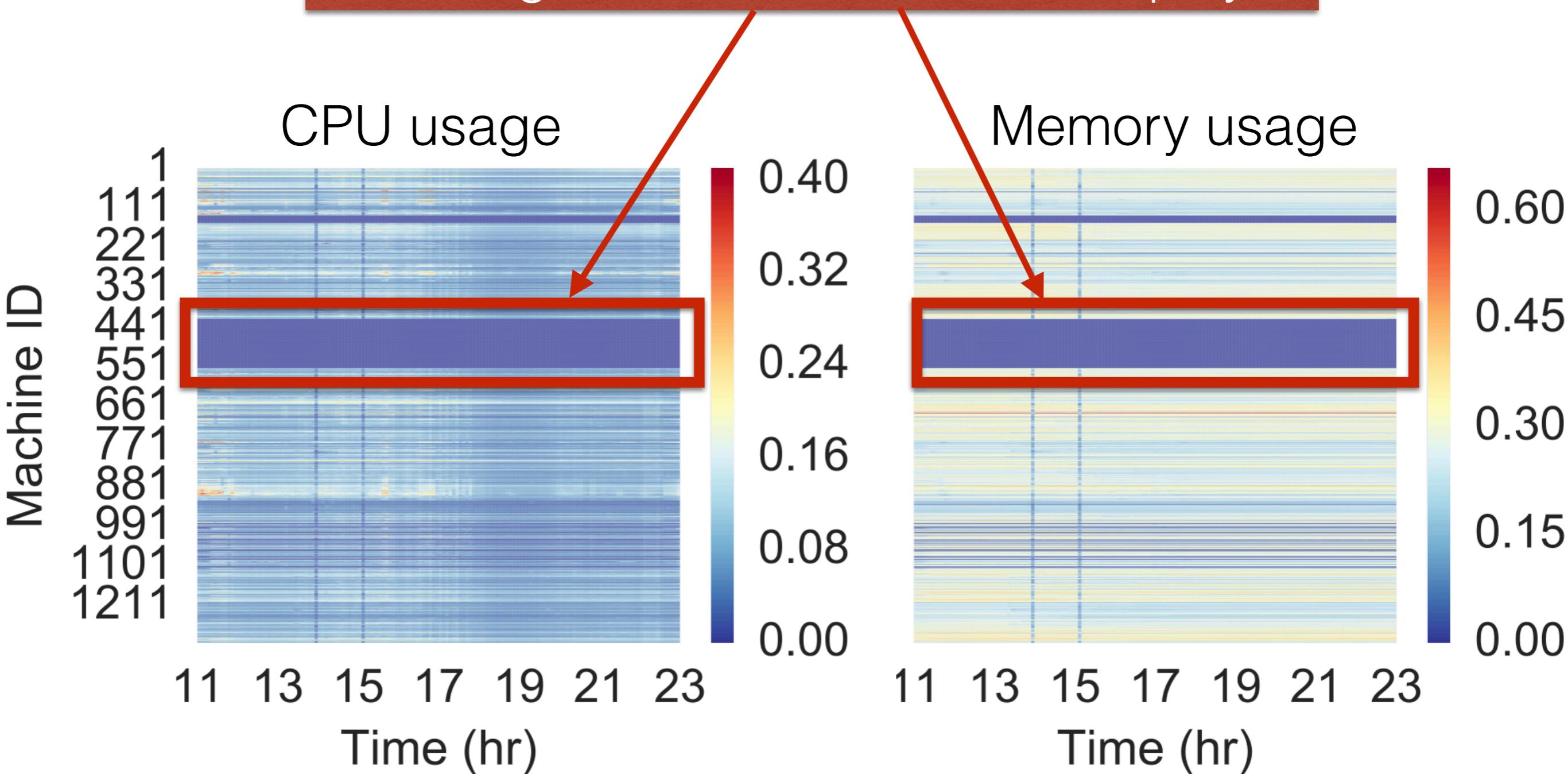


Transient, batch processing workloads

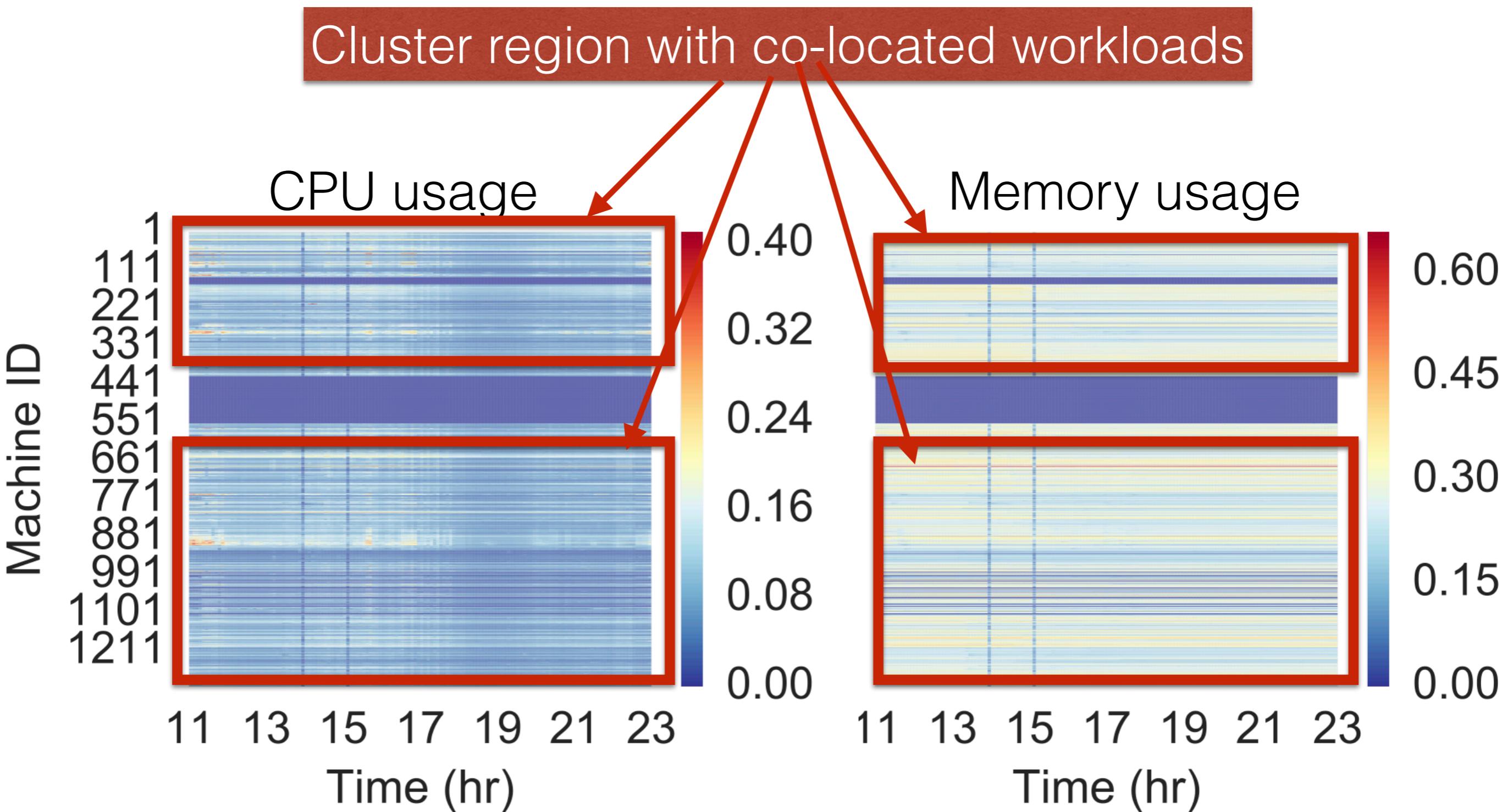


Cluster regions

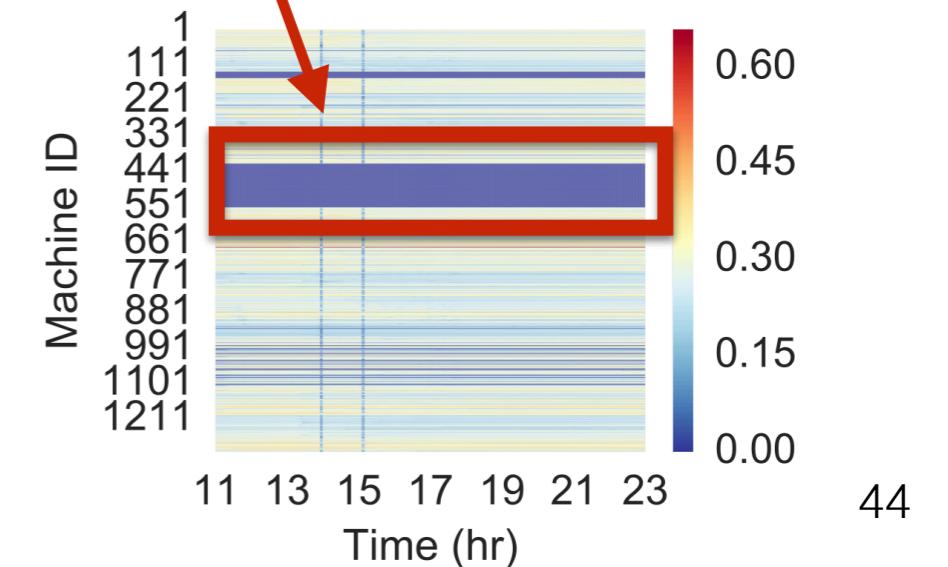
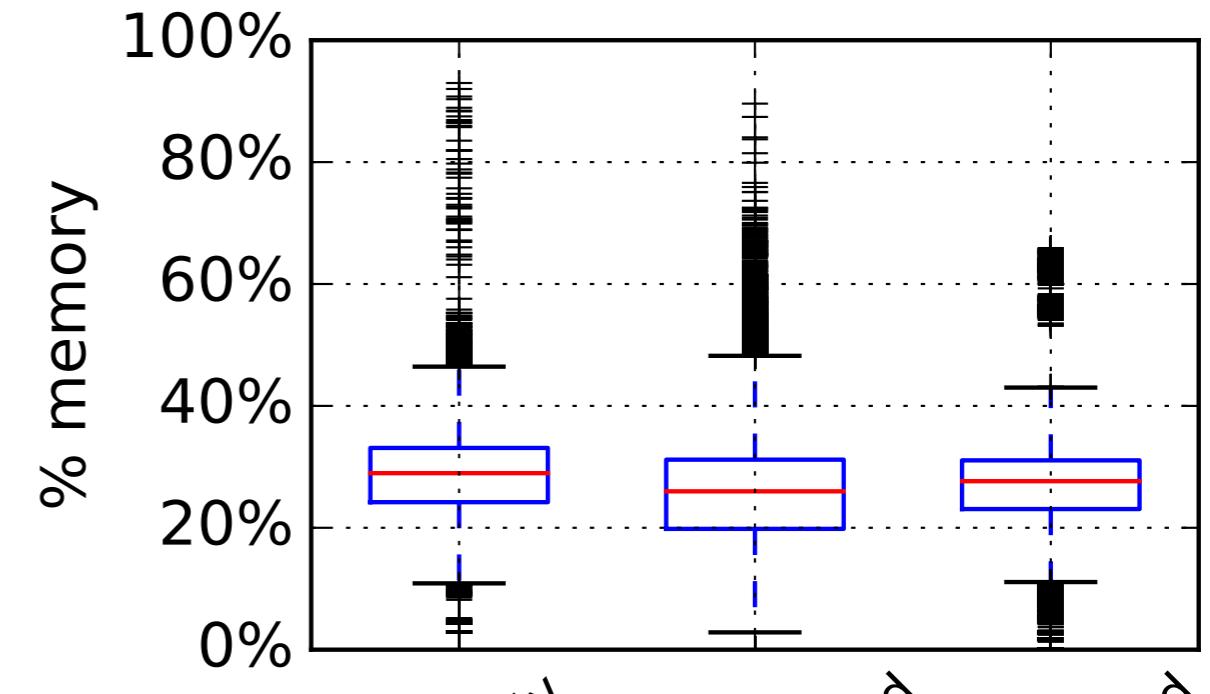
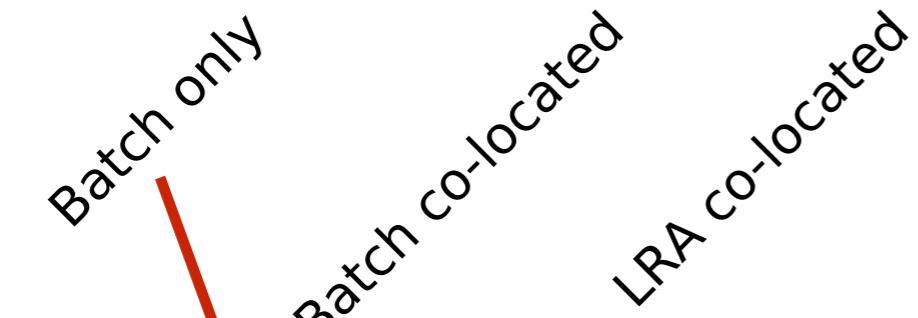
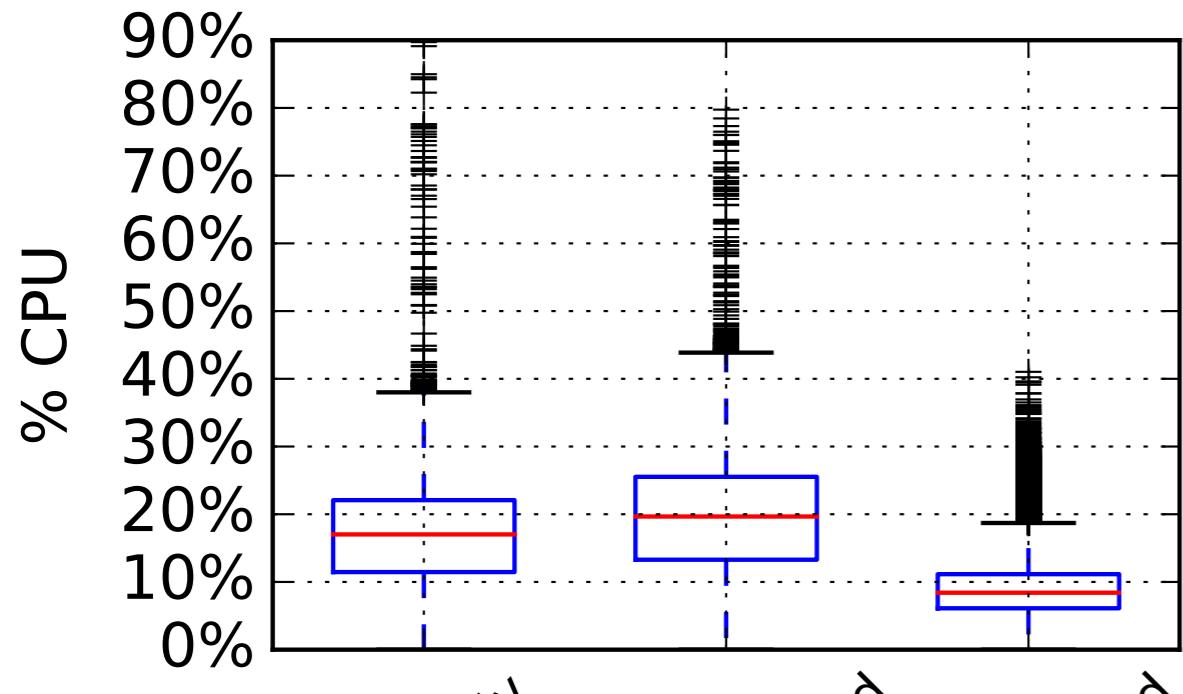
Buffer region with no containers deployed



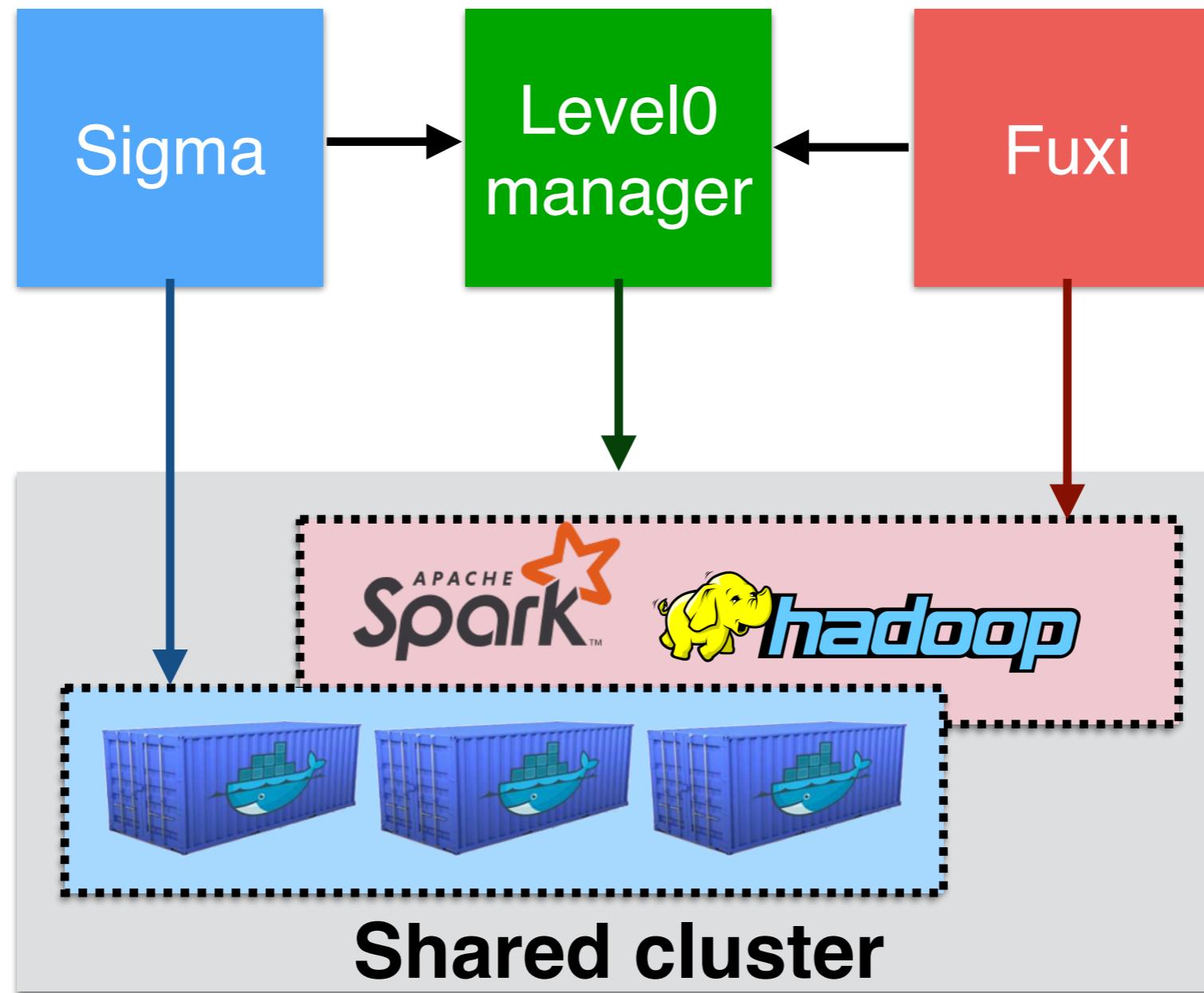
Cluster regions



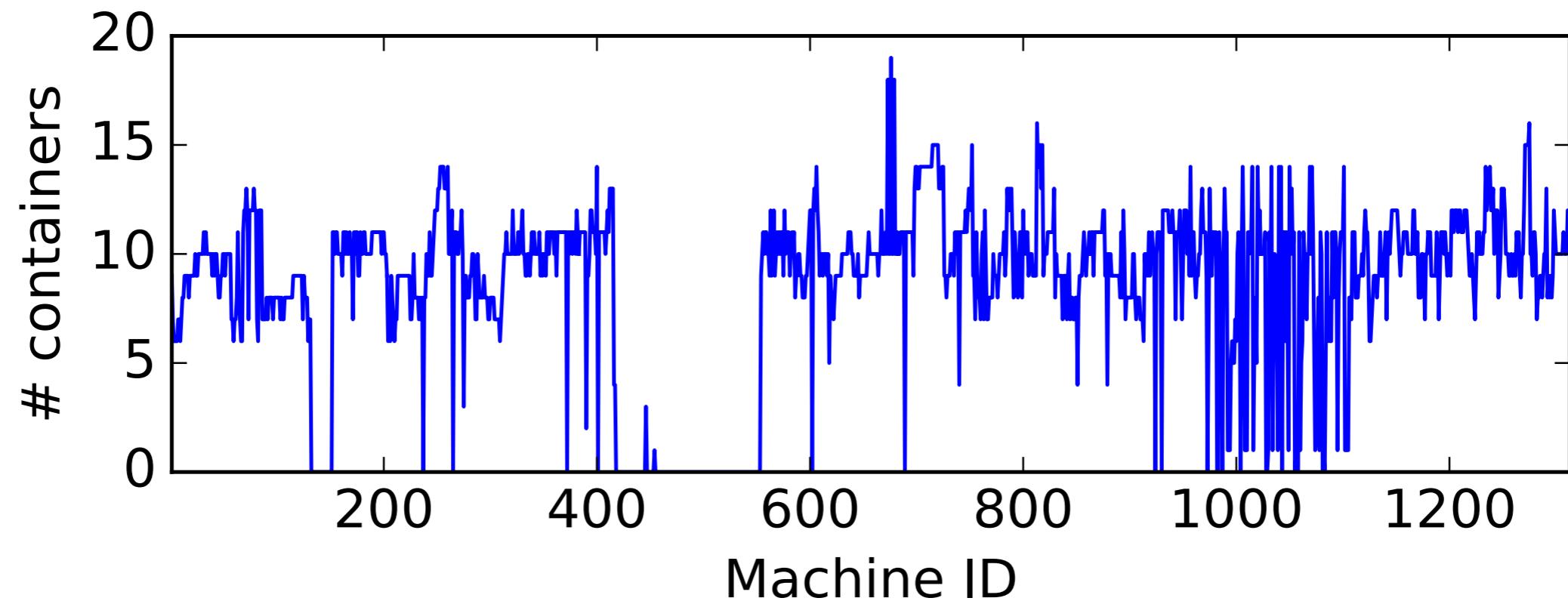
Resource usage at different cluster regions



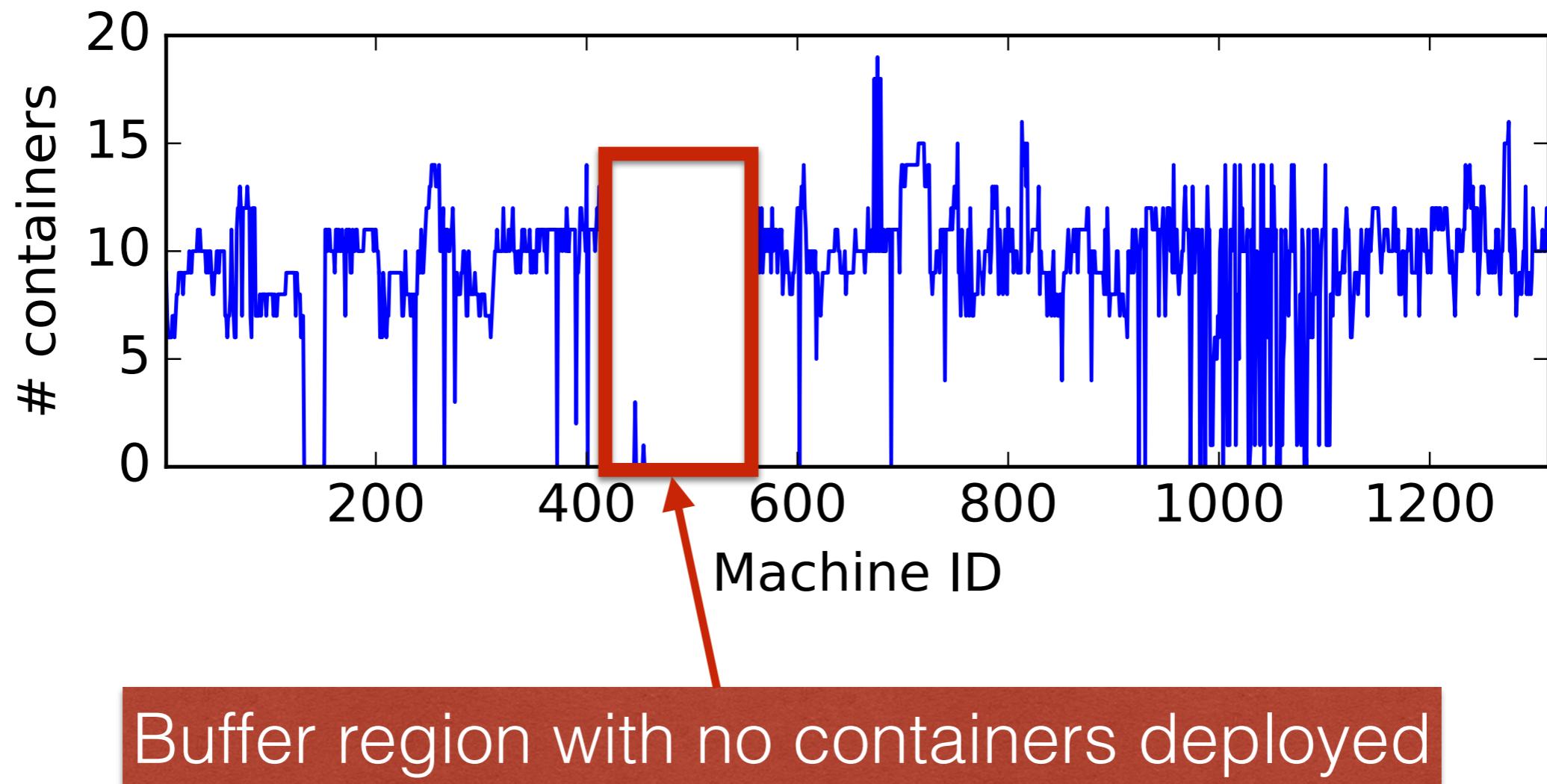
Workload co-location



Distribution of containers across the cluster

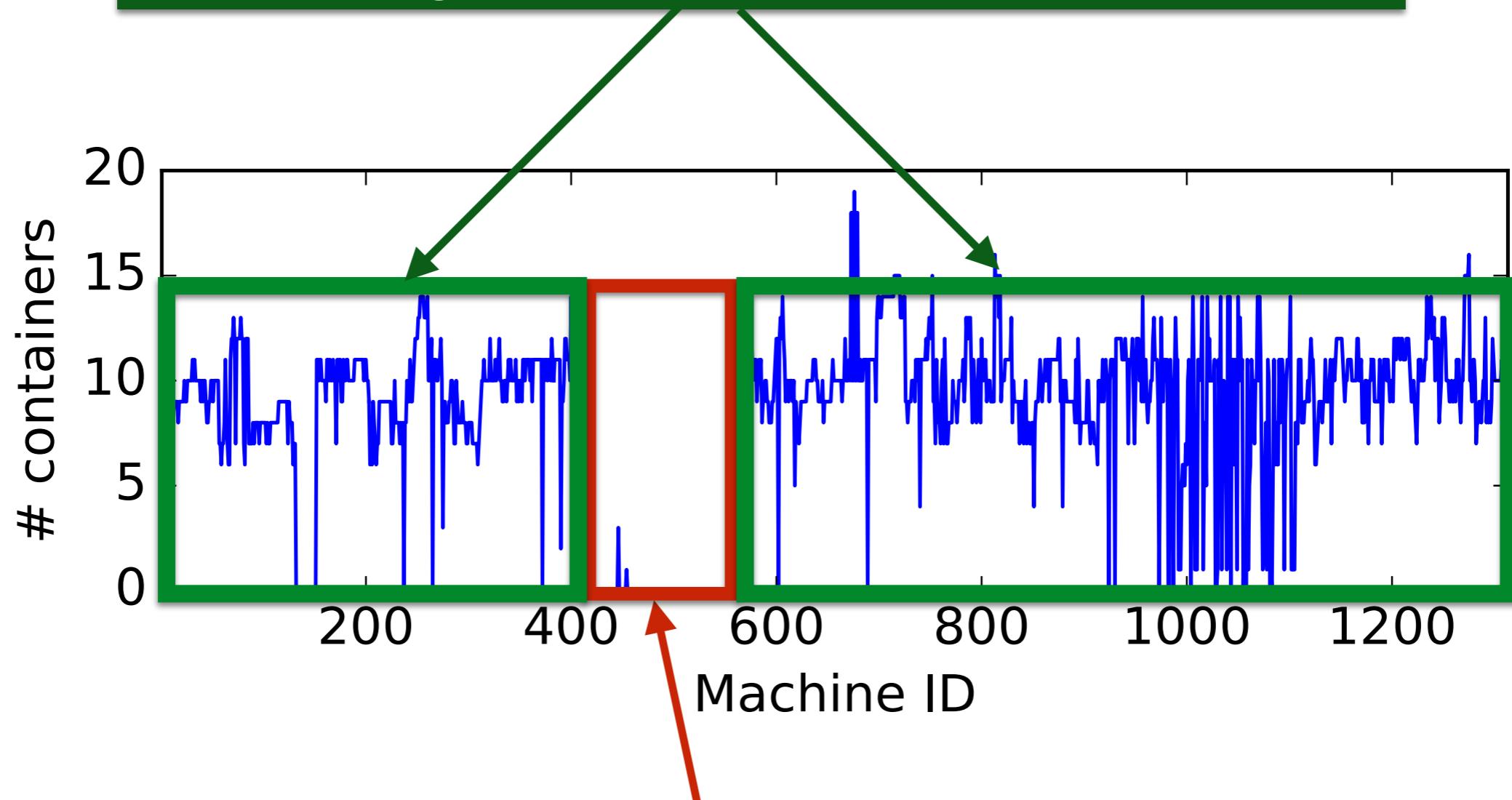


Cluster regions



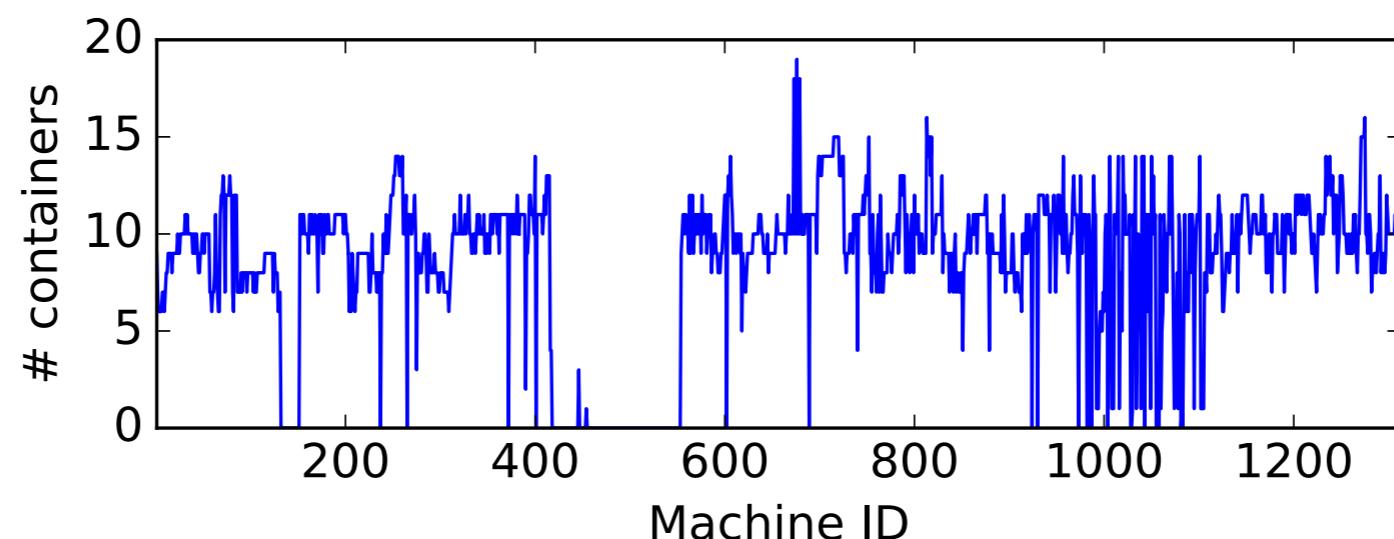
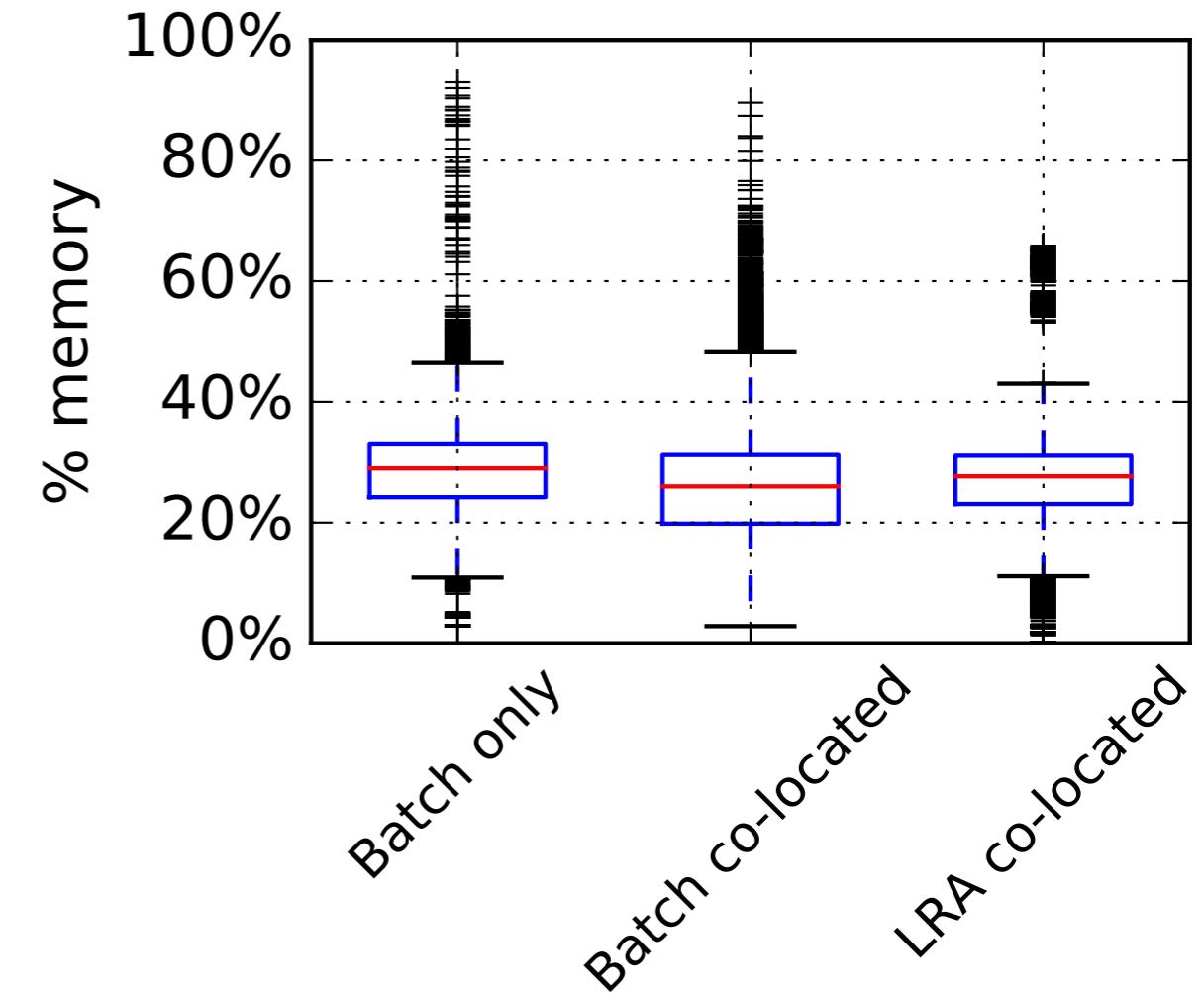
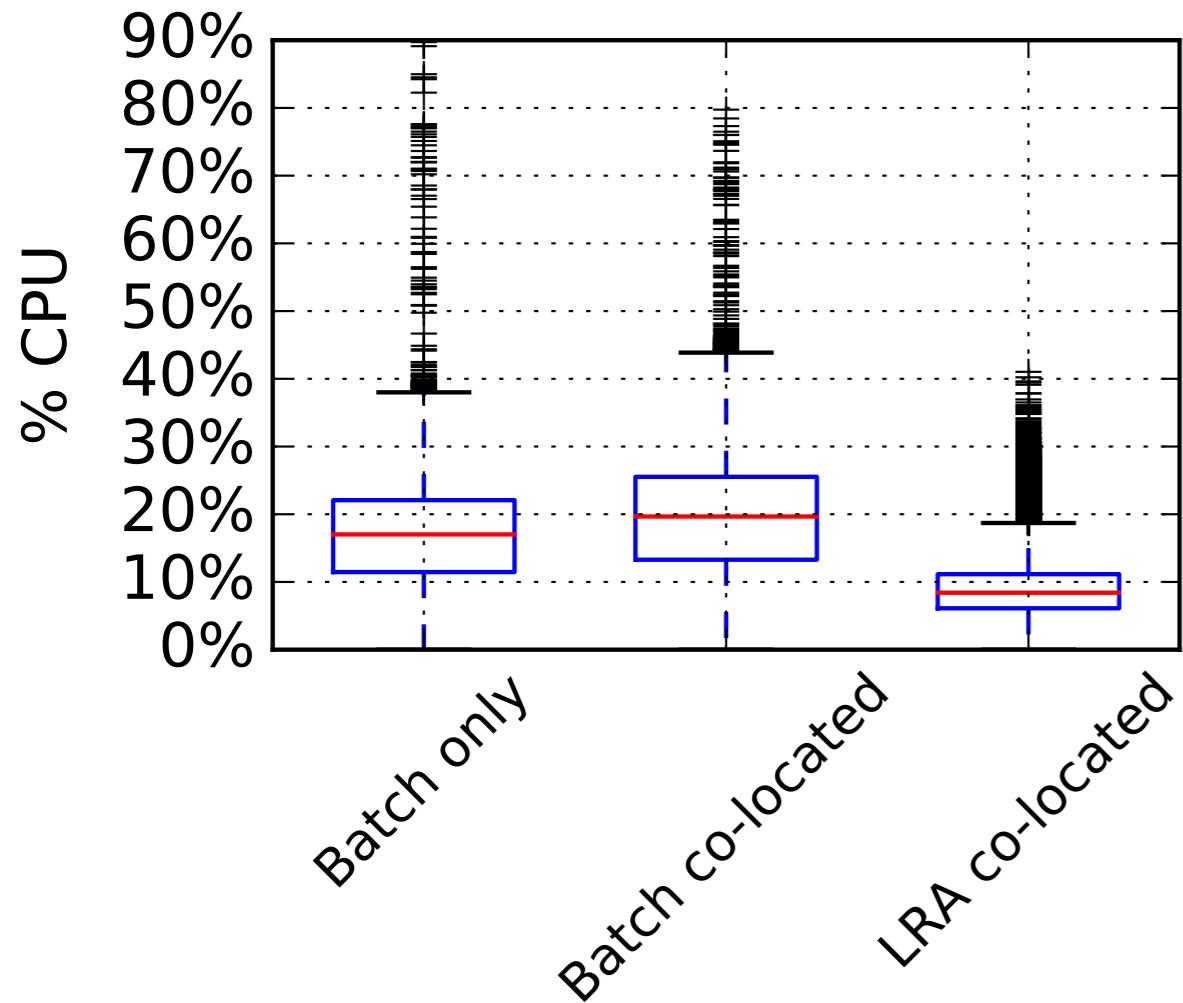
Cluster regions

Cluster regions with co-located workloads

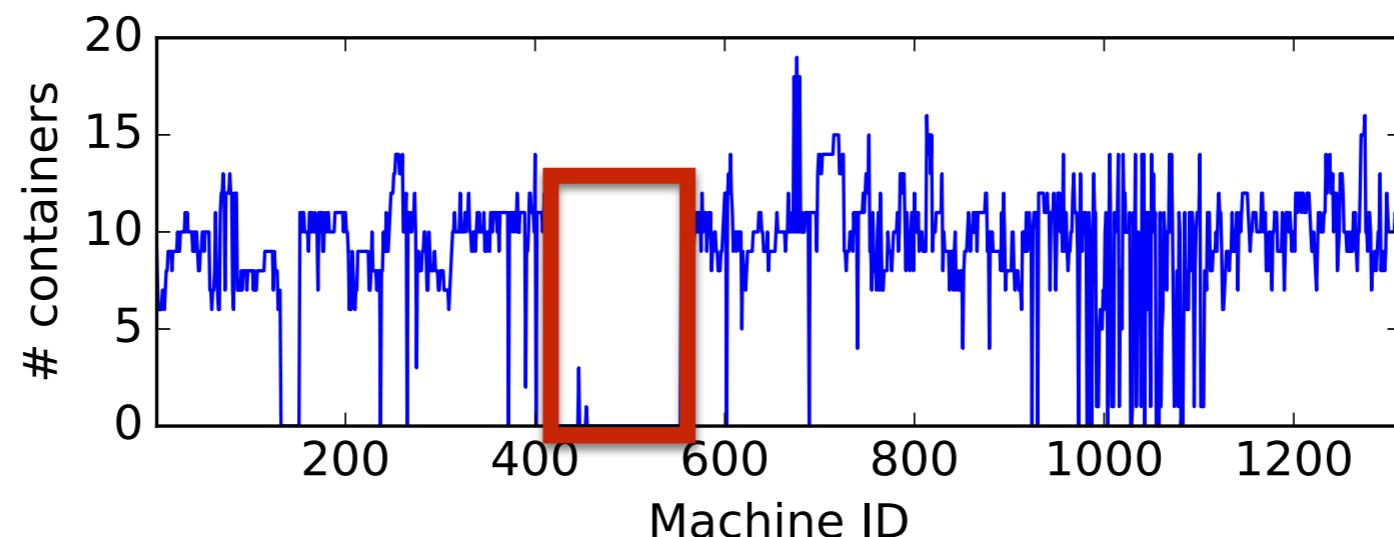
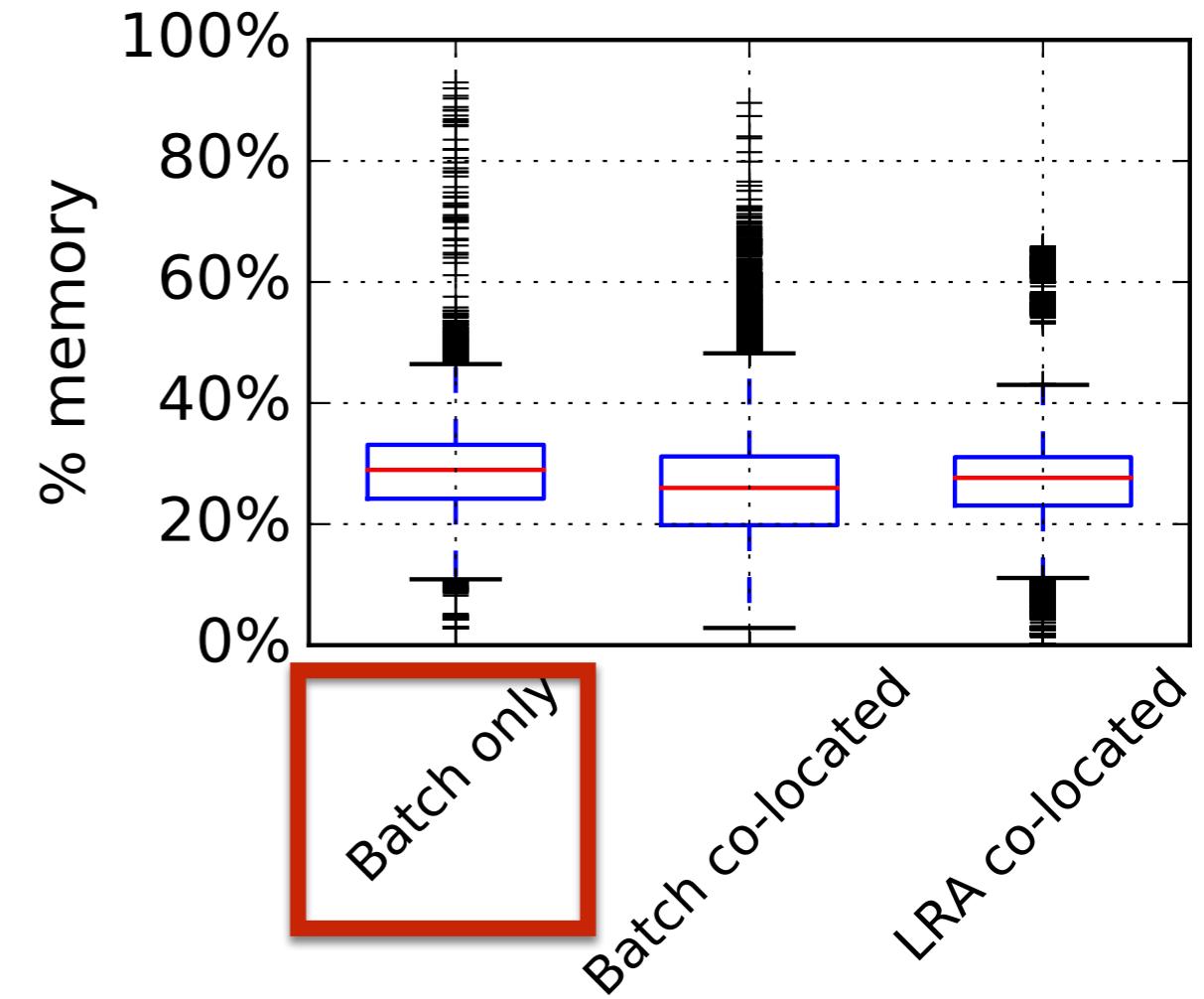
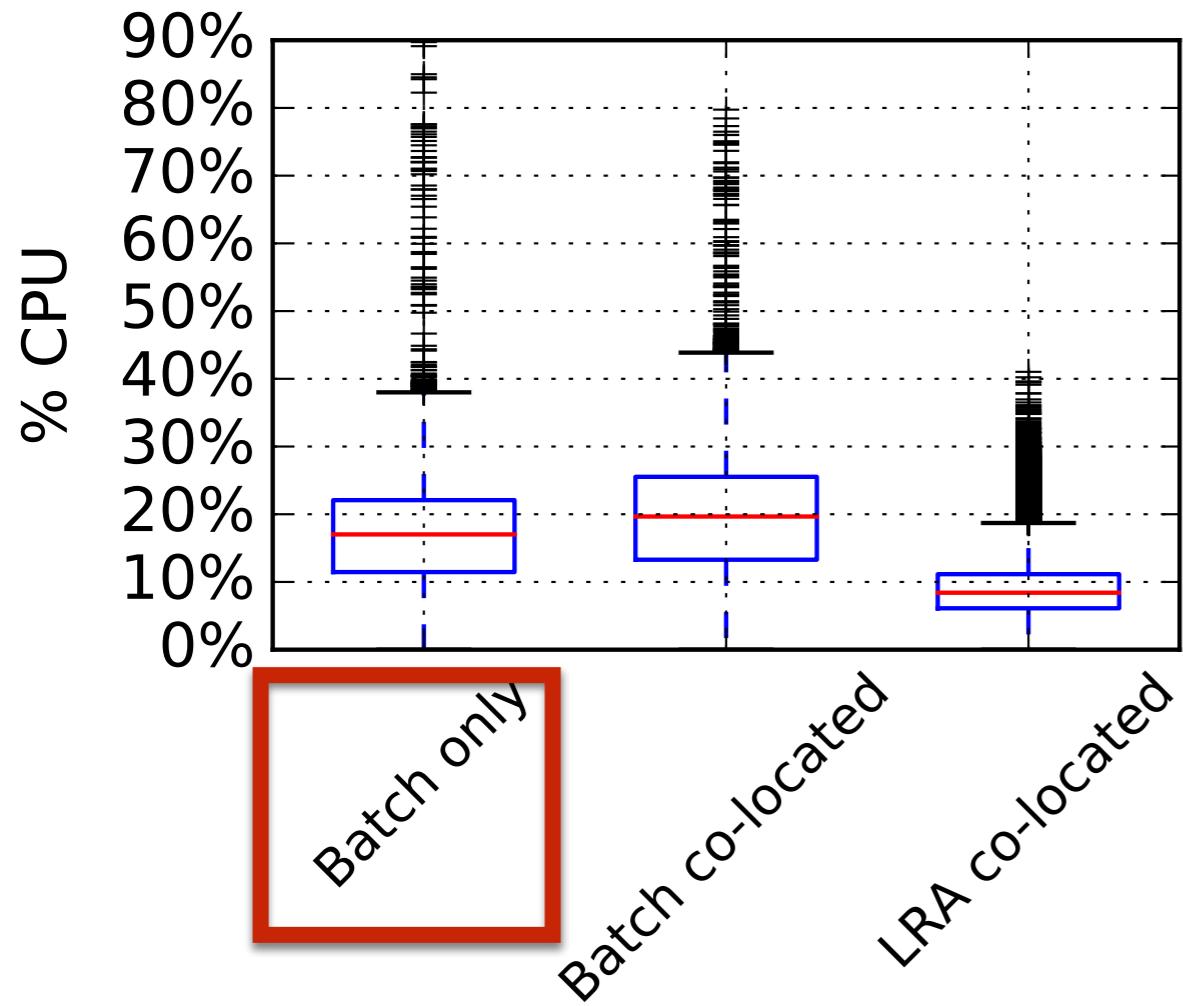


Buffer region with no containers deployed

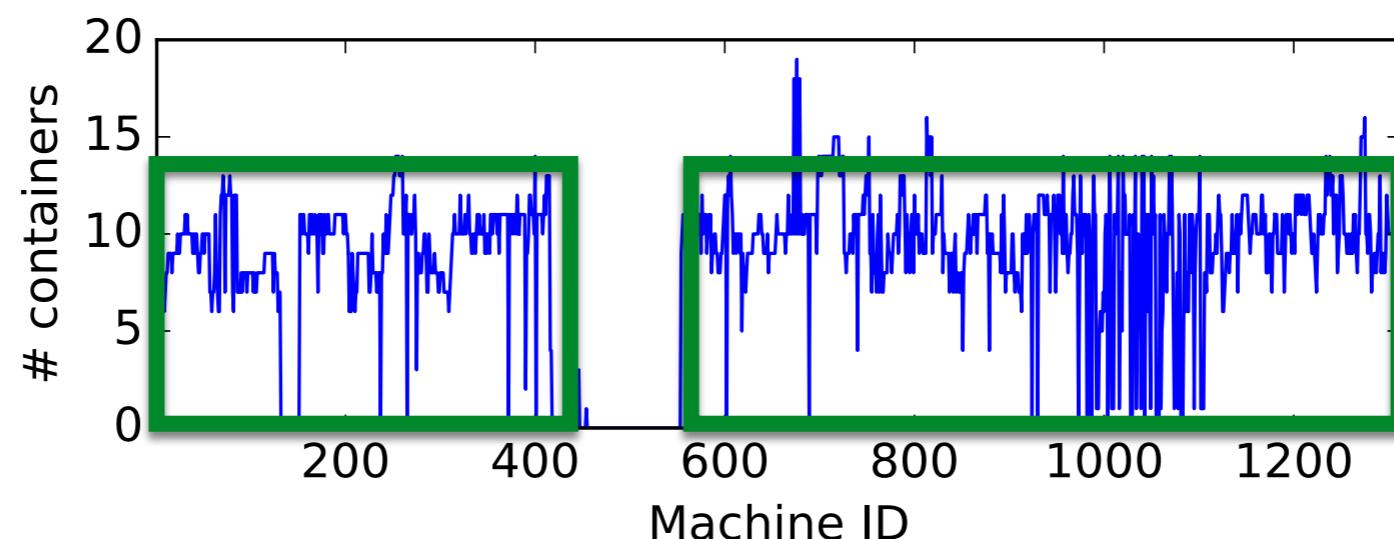
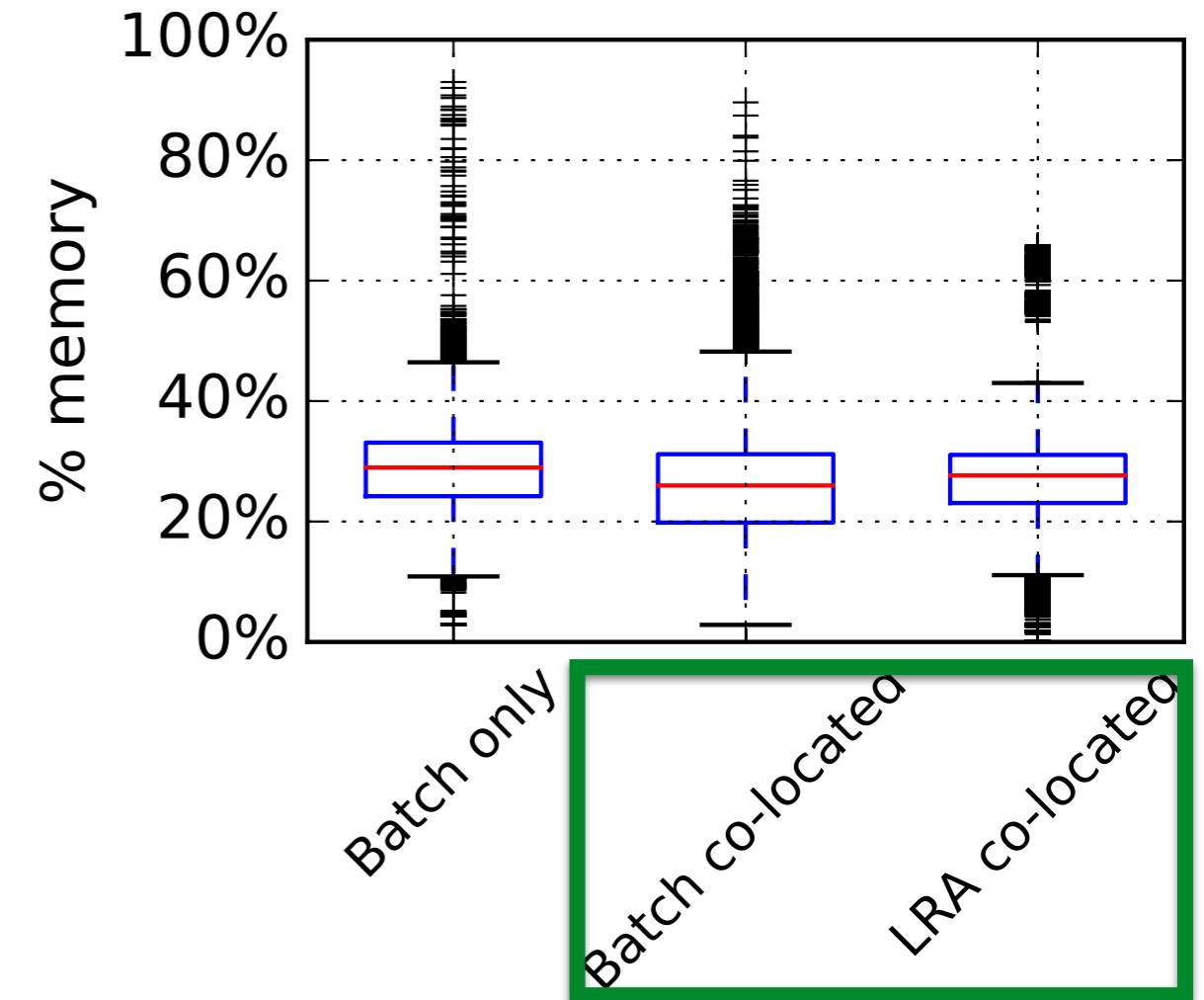
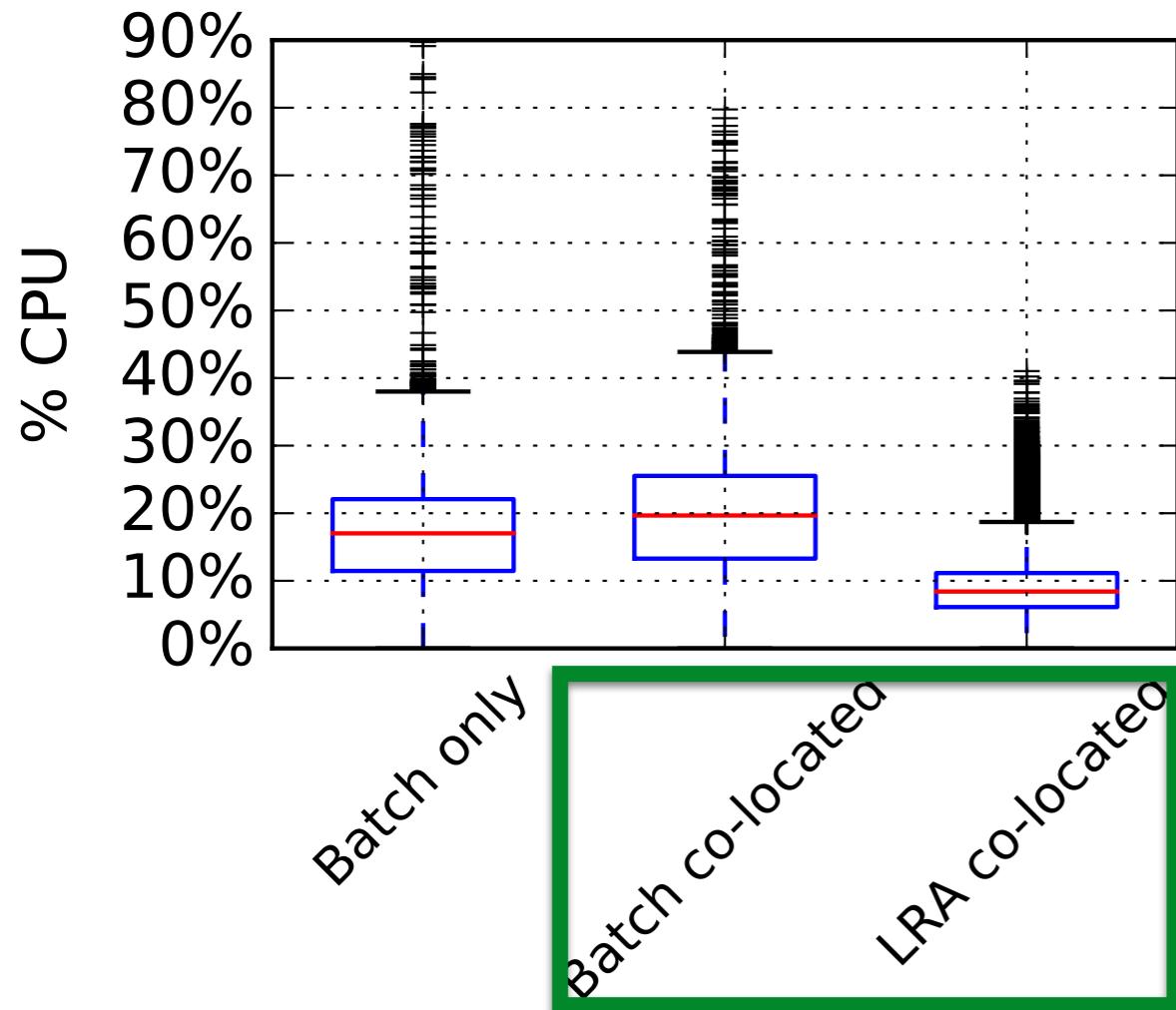
Resource usage at different regions



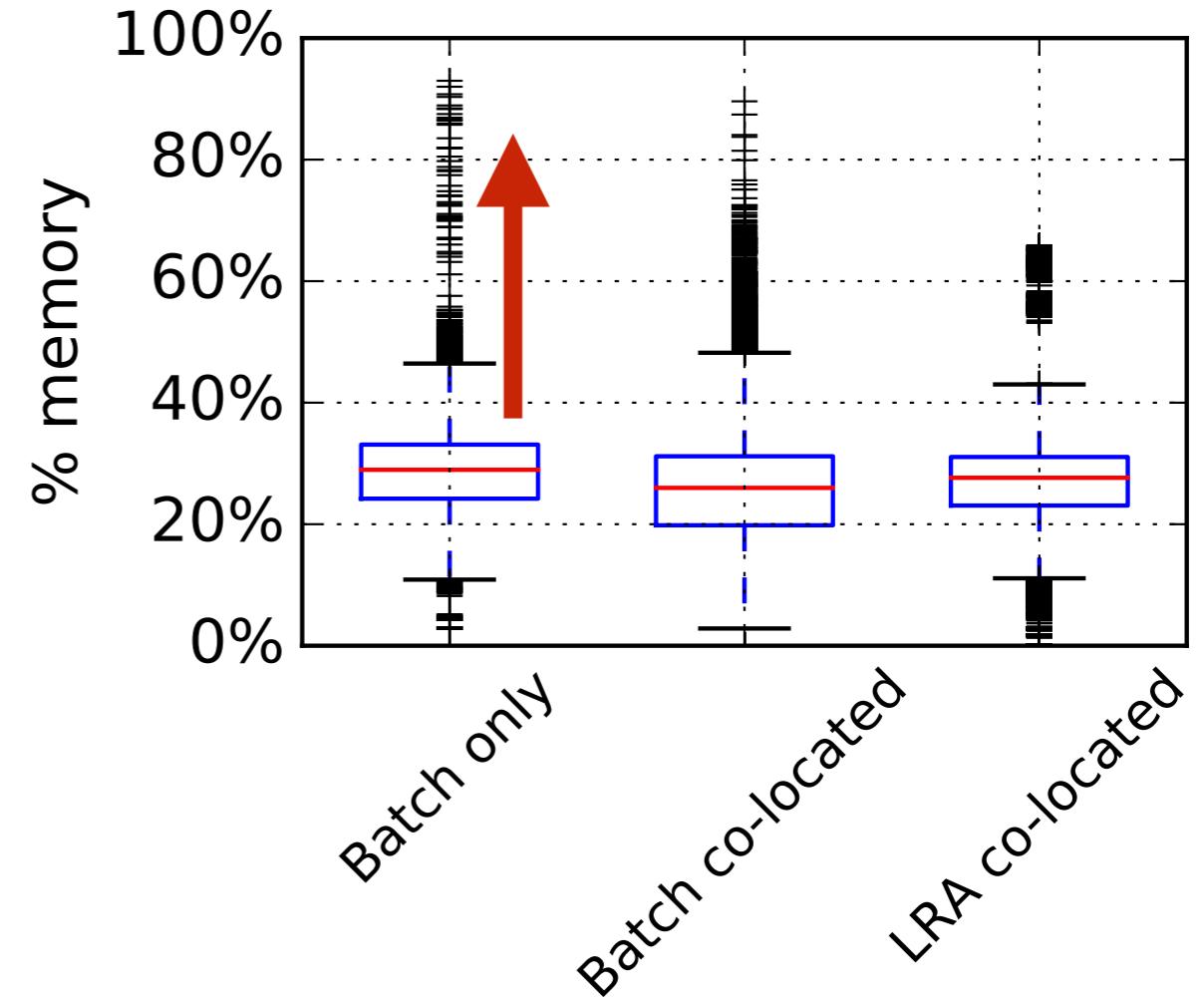
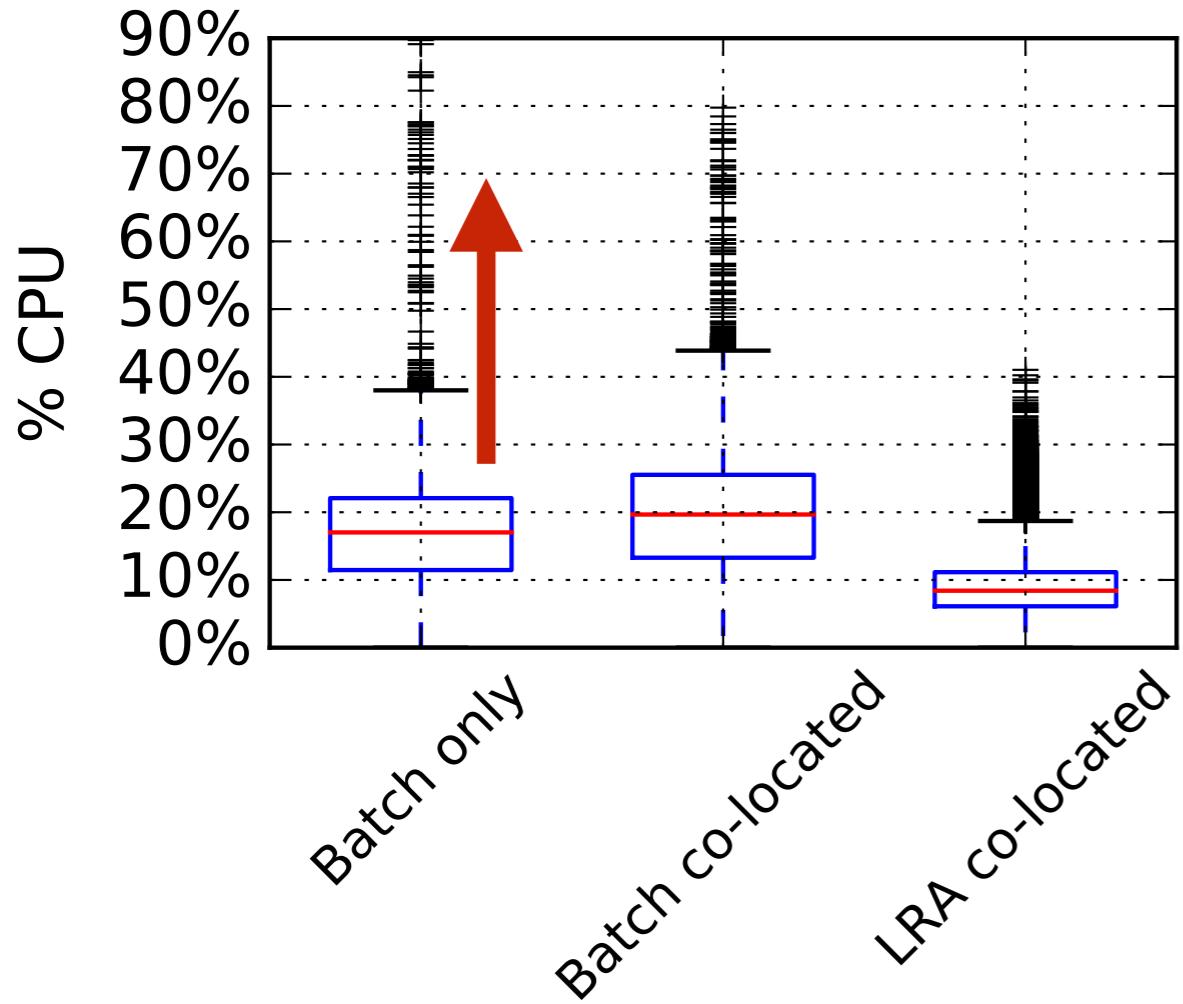
Resource usage at different regions



Resource usage at different regions



Resource usage at different regions



The batch only region has potential to improve its resource utilization by accommodating more batch jobs in there