ACM Symposium on Cloud Computing

Workload Consolidation in Alibaba Clusters

The Good, the Bad, and the Ugly

Yongkang Zhang[†*], Yinghao Yu[*], Wei Wang[†], Qiukai Chen[*], Jie Wu[*], Zuowei Zhang[*], Jiang Zhong[*], Tianchen Ding[*], Qizhen Weng[†*], Lingyun Yang[†*], Cheng Wang[†*], Jian He[*], Guodong Yang[*], Liping Zhang[*]

[†]Hong Kong University of Science and Technology [*]Alibaba Group

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Alibaba Group 阿里巴巴集团

Table of Contents

Background

Alibaba's E-commerce Businesses

- Alibaba is one of the largest IT giants in the world……
- Alibaba's businesses are developed on a wide range of technology stacks.

Images credit to Wikipedia: https://www.wikipedia.org/

Alibaba's Workload Management System

- The scale of Alibaba's clusters:
  - **Dozens of** large clusters.
  - **A few hundred ~ more than 10k** machines in each cluster.
  - **Hundreds of thousands of** machines in total.
  - **Tens of millions of** CPU cores and **tens of thousands** of GPUs.
  - **Millions of** service instances.
- Two types of workloads:
  - Long-running, latency-critical (**LC**) **services**.
  - Throughput-oriented **batch jobs**.

Design Principles

- Objectives
  - Reduce the **resource provisioning cost** without violating the **Service-level Objectives (SLOs)** of applications.

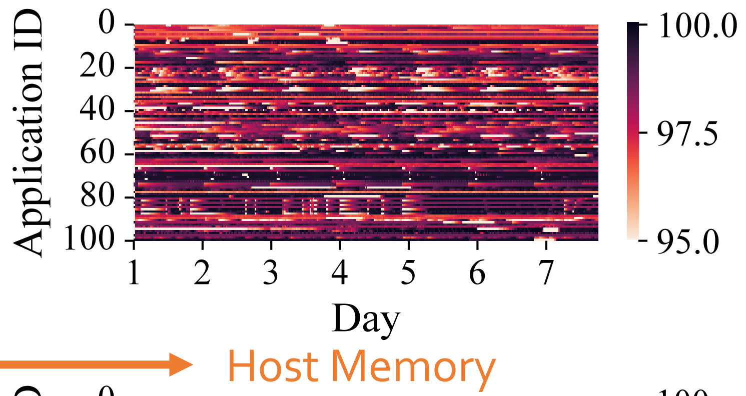
- **Transparent** to applications.

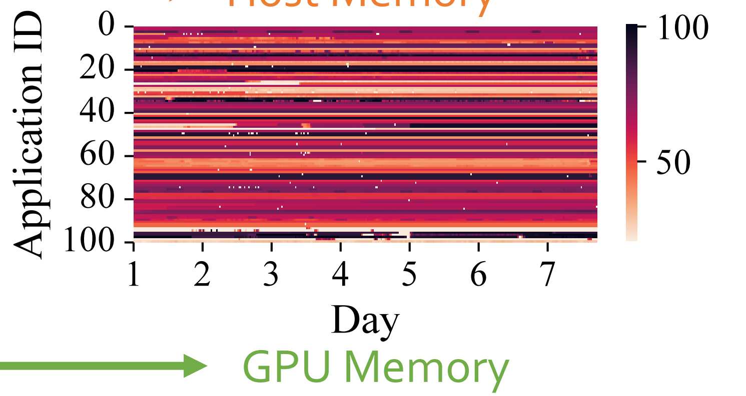
- **Generally applicable** to a range of services and frameworks.

Cluster-wide Macro-Management

The Problem of Overcommitment

- Diurnally changing LC services.
  - The diurnal pattern of CPU & GPU utilization creates opportunities for overcommitment.
  - **Host** and **GPU memory** limit the overcommitment at night.
- The memory bottleneck.
  - Unlike CPUs and GPUs, the host and GPU memory footprints of LC services **stay relatively stable**.
  - Batch jobs request more memory……
    - Aggravates this problem.

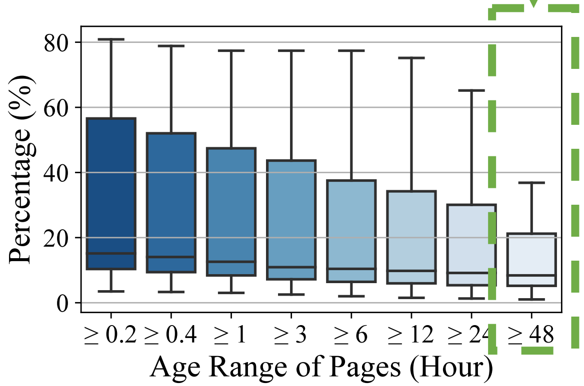




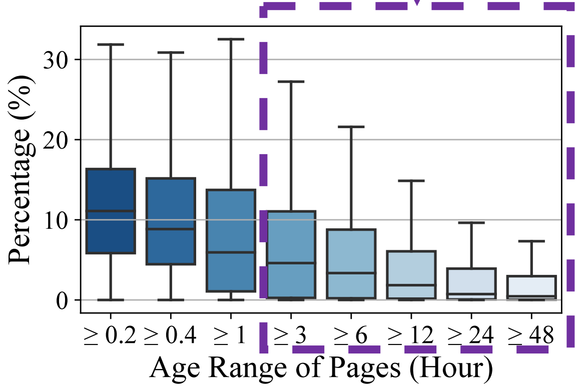
The memory utilization of 100 LC services (usage / request, %).

Memory Reclamation

- Tracking memory idleness.
  - Following Google's `kstaled` [1], we added `kidled` [2] into the Linux kernel to periodically mark the ***age*** of reclaimable pages.
  - **A large number of reclaimable idle pages** exist in LC services.
    - Around half of swappable anonymous pages have **an age ≥ 48 hrs**.
    - Around half of clean file pages have **an age ≥ 3 hrs**.



Swappable anonymous pages



Clean file pages

The distribution of (Reclaimable page / total memory usage) of LC services running on each machine in a cluster by the age (last access time) of memory pages.

[1] kstaled. https://lore.kernel.org/lkml/20110922161448.91a2e2b2.akpm@google.com/T/.
[2] kidled. https://github.com/alibaba/cloud-kernel/blob/linux-next/mm/kidled.c.

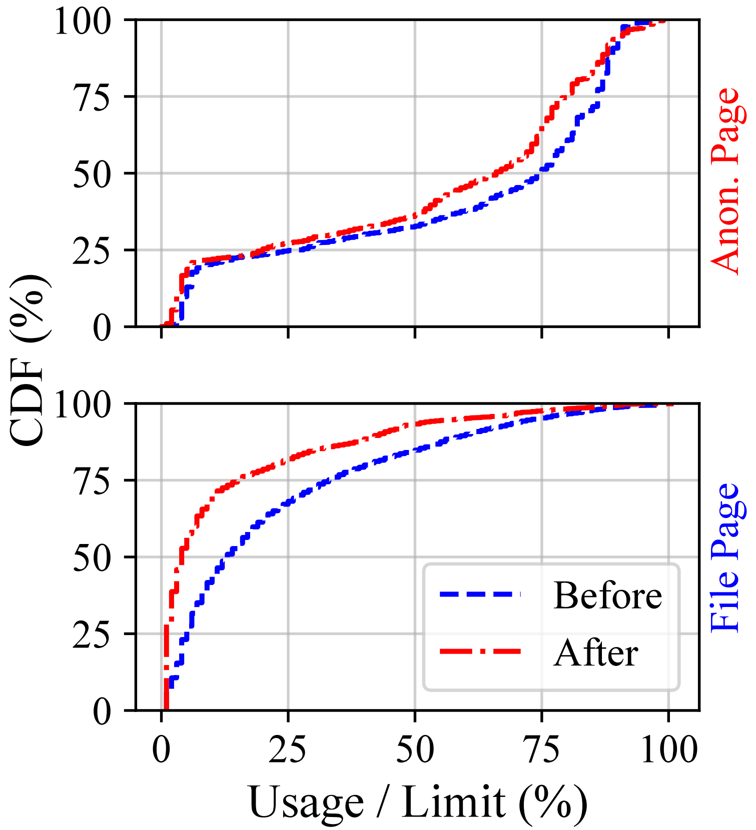
Memory Reclamation

- Proactive memory reclamation.
  - Reclaim pages with an ***age*** longer than a threshold tuned by small-scale experiments on representative LC services.
  - Use **memory pressure stall information (PSI)** [1] to detect memory pressure and evict batch jobs.
  - Please refer to our paper for more details.

[1] Tracking pressure-stall information. https://lwn.net/Articles/759781/.

Memory Reclamation

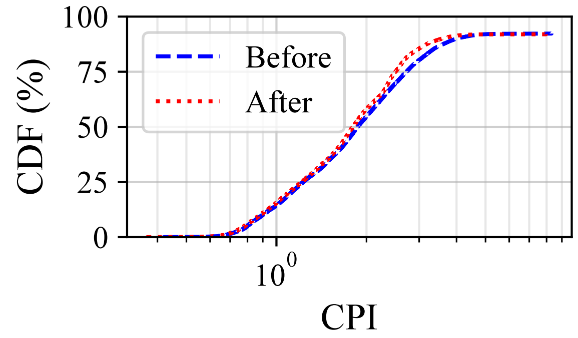
- Proactive memory reclamation.
  - Deployment results (*median utilization, %*):
    - **Anonymous pages**: **74%** -> **67%**
    - **File pages**: **13%** -> **4%**



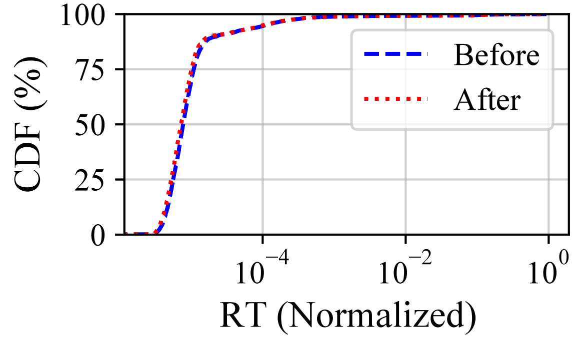
File / anonymous page utilization before and after memory reclamation.

Memory Reclamation

- Proactive memory reclamation.
  - **No significant impacts** on LC services in terms of:
    - CPI (cycles per instruction) [1]
    - Average service response time



CPI (cycles per instruction)



Average response time

Comparison of LC services' performance before and after memory reclamation.

[1] Zhang et al., CPI²: CPU performance isolation for shared compute clusters. In Proc. ACM EuroSys 2013.

Tidal Scaling

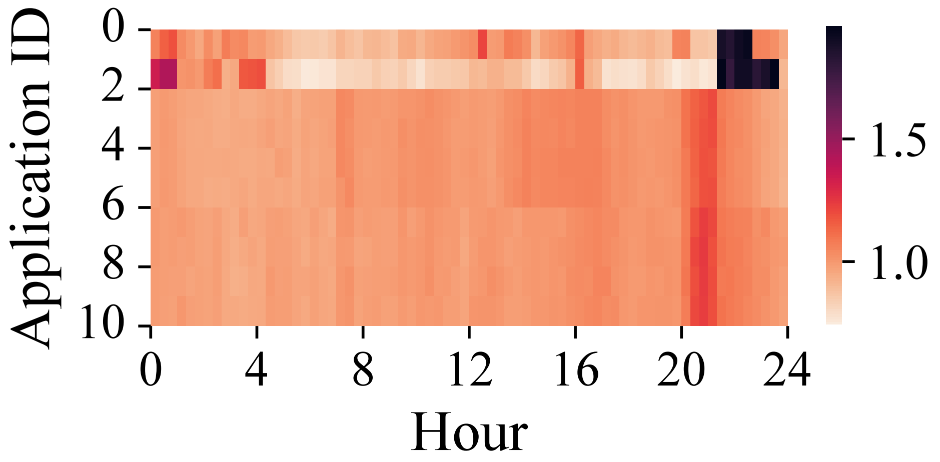
- Memory reclamation is insufficient.
  - **Cannot be applied to GPU memory**.
  - The resulting memory utilization still **has no clear diurnal pattern**.
- Why not use vertical scaling / horizontal scaling?
  - Fine-grained vertical scaling is **insufficient**.
  - Horizontal scaling **cannot be directly applied**.

Tidal Scaling

- Bimodal instance.
  - Applied to LC services with diurnal traffics.
  - Two states:
    - *Running* instances: actively serve user requests and consume resources.
    - *Dormant* instances: no running process and resource consumption.
  - A bimodal instance can rapidly change its state by:
    - **Starting** processes before the day's traffic picks up.
      - 🌛->🌞 : *Dormant* -> *Running*
    - **Terminating** processes before the night arrives.
      - 🌞->🌛 : *Running* -> *Dormant*

Tidal Scaling

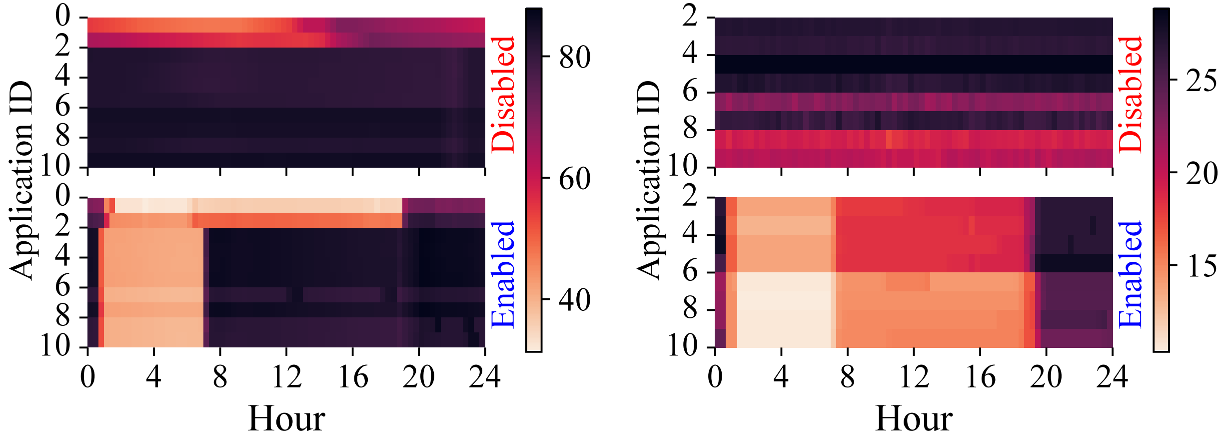
- Evaluation.
  - **#0, 1:** LC services that only consume **CPUs**.
  - **# 2-9:** LC services that consume both **CPUs** and **GPUs**.
  - No significant variations in **service response time (RT)**.



Service RT (normalized by each application's daily average) when tidal scaling is enabled.

Tidal Scaling

- Evaluation.
  - **#0, 1:** LC services that only consume **CPUs**.
  - **# 2-9:** LC services that consume both **CPUs** and **GPUs**.
  - Create a diurnal pattern for **host** & **GPU** memory.

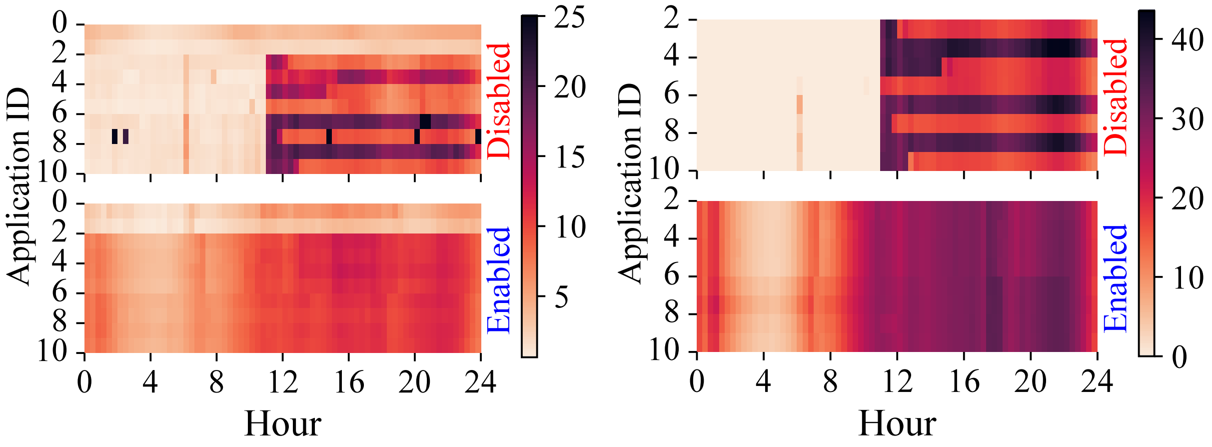


Host memory utilization (usage / request, %) of LC services

GPU memory utilization (usage / request, %) of LC services

Tidal Scaling

- Evaluation.
  - **#0, 1:** LC services that only consume CPUs.
  - **# 2-9:** LC services that consume both CPUs and GPUs.
  - Keep CPU & GPU utilization stable.



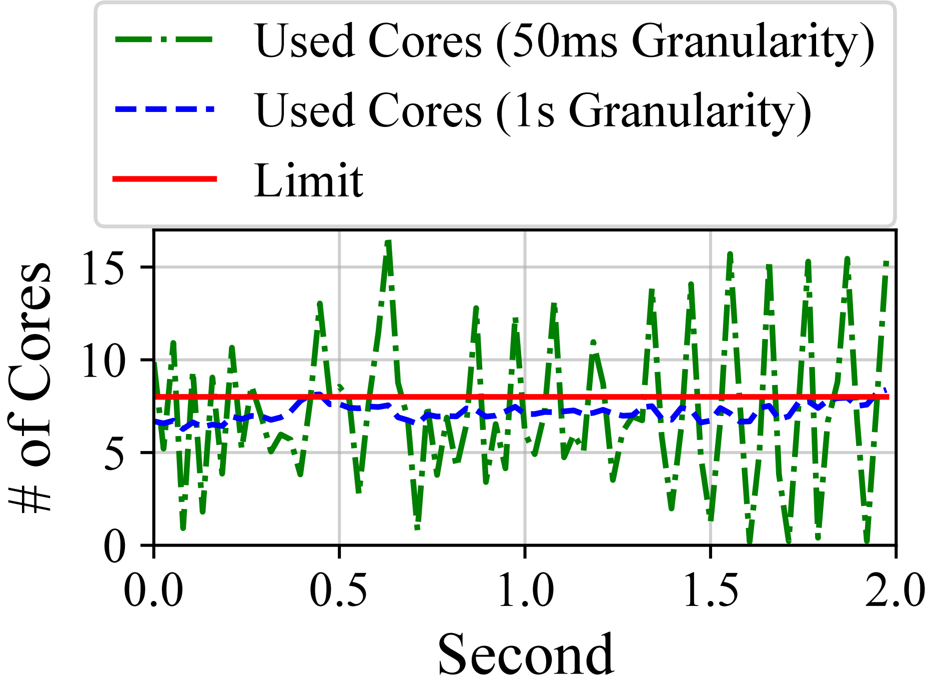
CPU utilization (usage / request, %) of LC services

GPU utilization (usage / request, %) of LC services

Node-level Micro-management

CPU Jitters

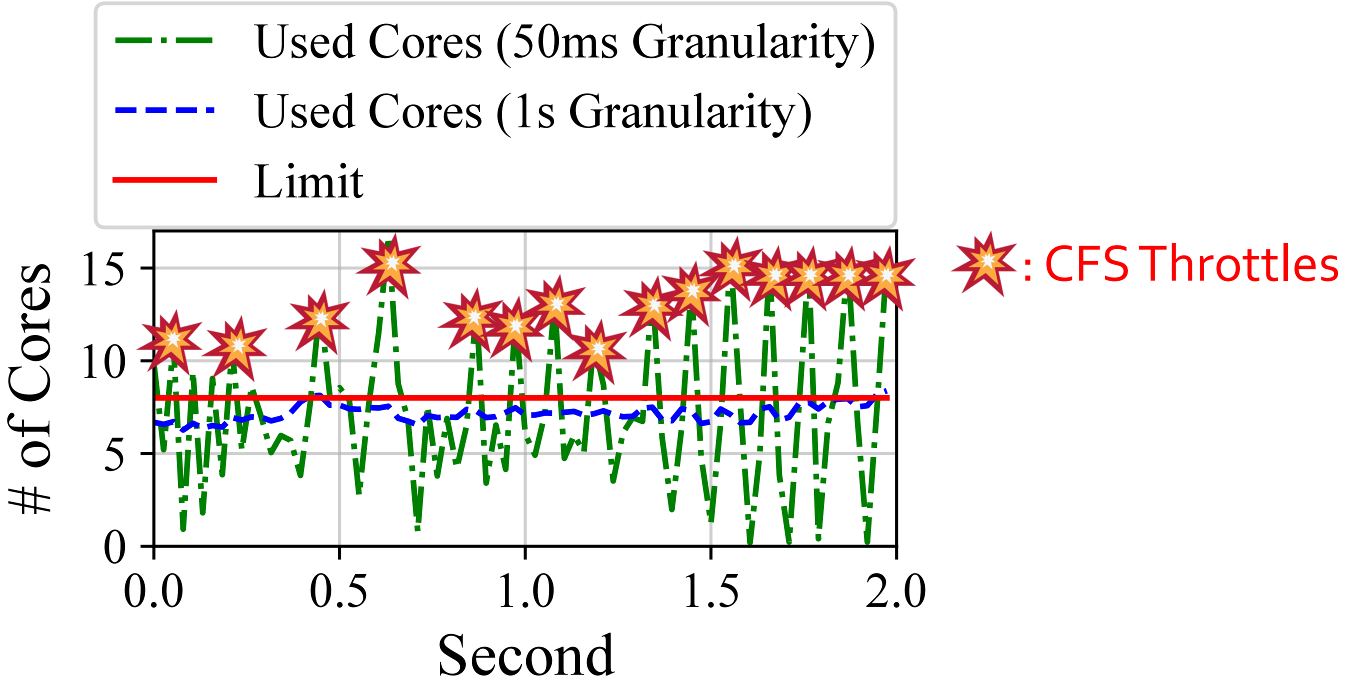
- Alibaba's applications have tiny CPU load spikes……



The CPU usage trace of a CPU-bursty LC service in different time scales.

CPU Jitters

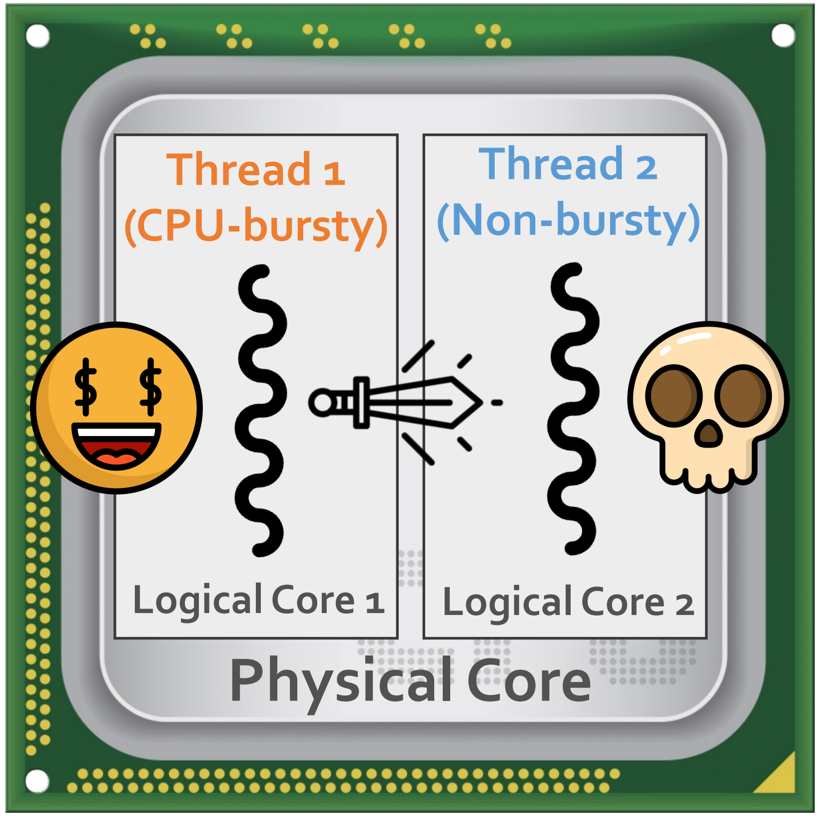
- CFS controller **throttles** the application's CPU usage when CPU jitters occur and exceed the CPU limit.



The CPU usage trace of a CPU-bursty LC service in different time scales.

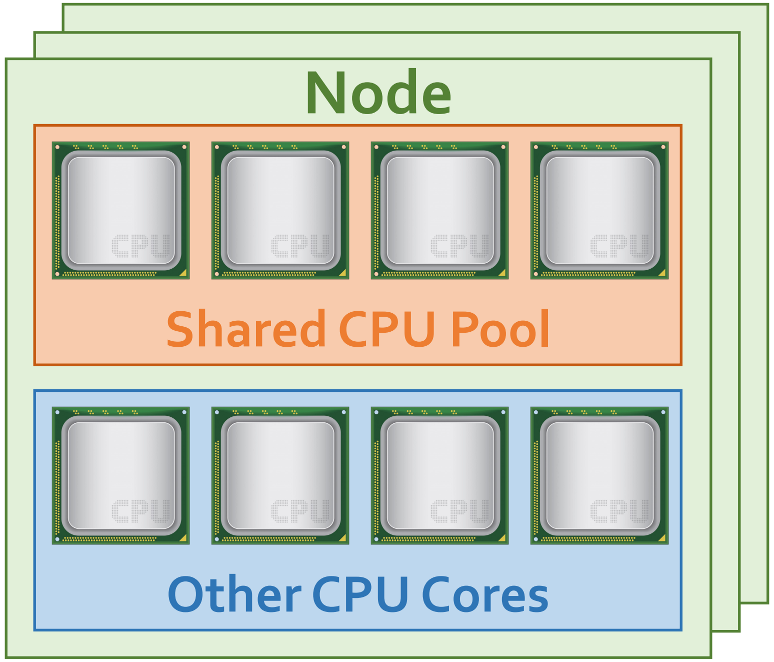
CPU Jitters

- Shared CPU pool for CPU-bursty applications.
  - **CPU-bursty hyper-threads** running on paired logical cores could contend for resources.



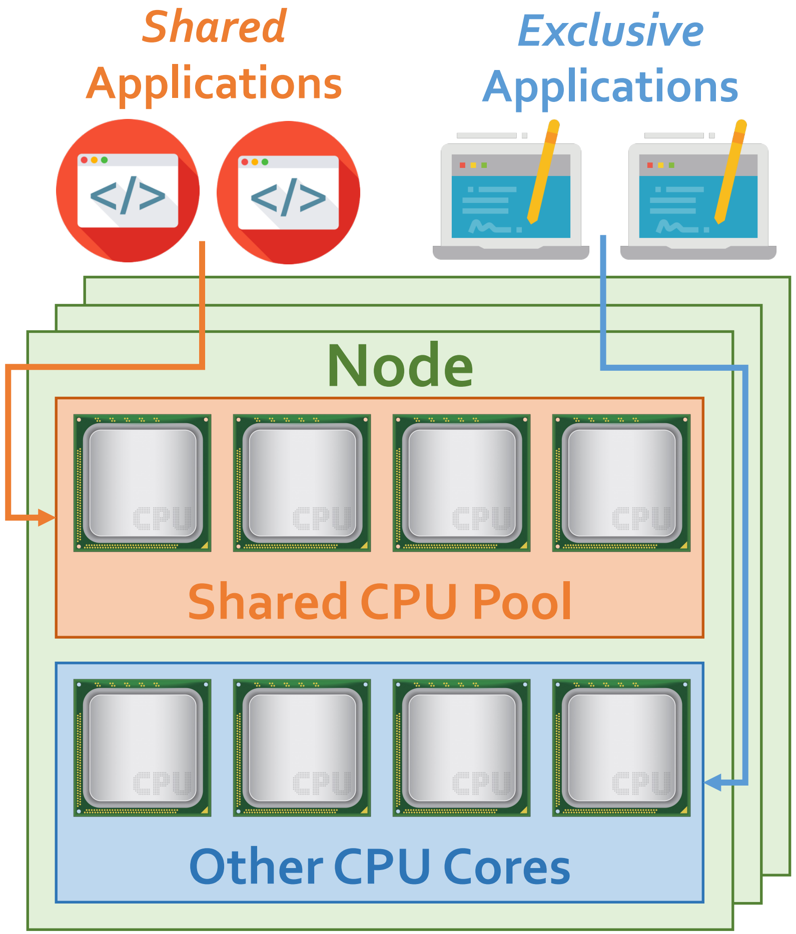
CPU Jitters

- Shared CPU pool for CPU-bursty applications.
  - Set up a **shared CPU pool** on each node for CPU-bursty applications.



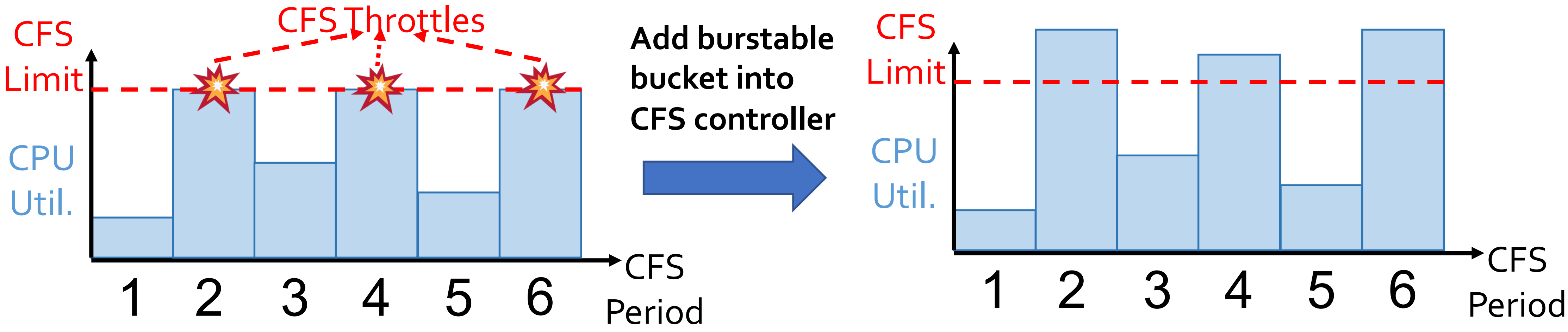
CPU Jitters

- Shared CPU pool for CPU-bursty applications.
  - Set up a **shared CPU pool** on each node for CPU-bursty applications.
  - Divide LC applications into two categories: ***exclusive*** and ***shared***.



CPU Jitters

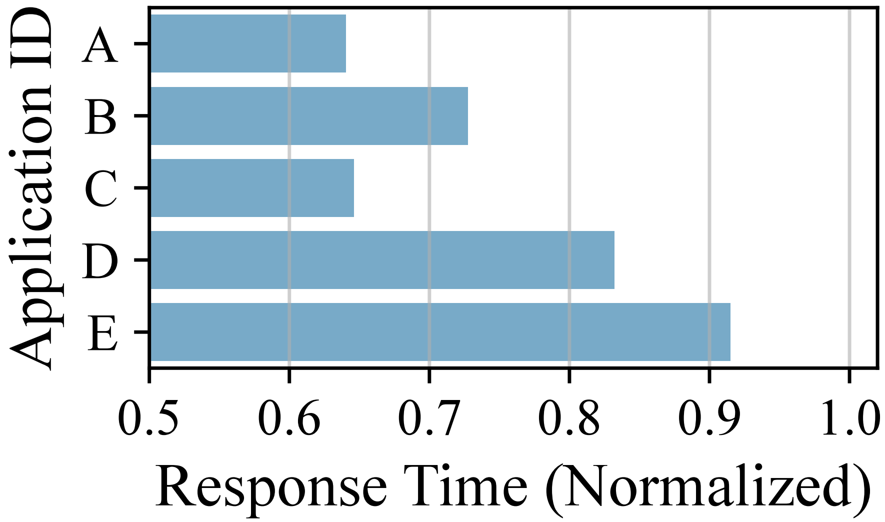
- Burstable CFS (*Completely Fair Scheduler*) Controller [1].
  - Use token bucket to carry over some unused quotas to future CFS periods.



[1] Burstable CFS bandwidth controller. [https://lwn.net/ml/linux- kernel/20210202114038.64870-1-changhuaixin@linux.alibaba.com/](https://lwn.net/ml/linux-kernel/20210202114038.64870-1-changhuaixin@linux.alibaba.com/).

CPU Jitters

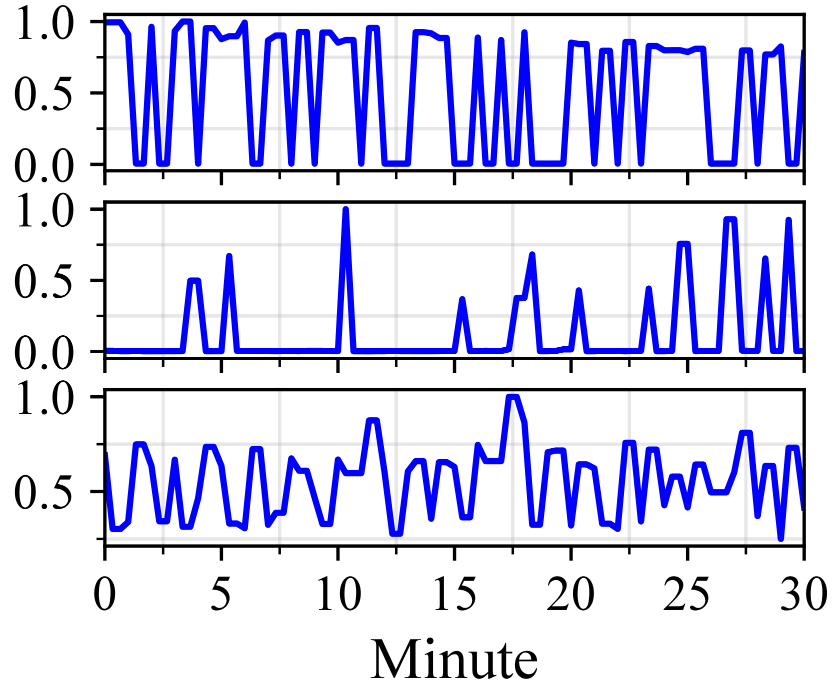
- Production Deployment (160k LC instances).
  - *Shared* LC instances being throttled during peak time: **73.4%** -> **0.12%**.
  - **10 – 35%** reduction in the average RT enabled by our approach.



Average response time of 5 representative CPU-bursty LC services after enabling our solution (normalized by the daily average before the deployment)

Variations on Memory Bandwidth

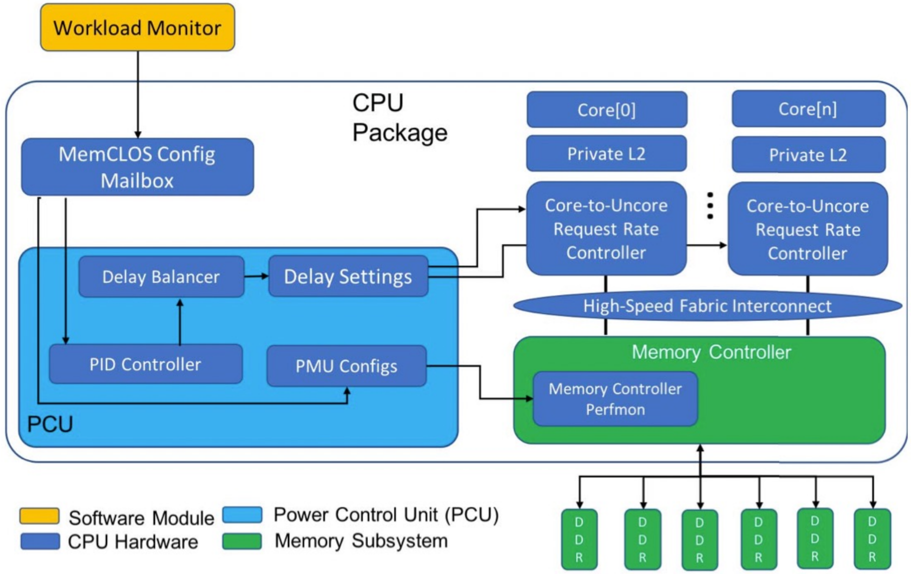
- Variations in memory bandwidth are prevalent.
  - Especially in batch jobs with different computing phases.
  - Around **12%** of the machines have memory access latency **1.5 - 8x** longer than the average due to **high memory bandwidth utilization**.
  - Excessive memory bandwidth utilization **undermines LC services' QoS**.



The memory bandwidth consumption of 3 batch job instances (estimated by # L3 cache misses per second and normalized by the maximum)

Variations on Memory Bandwidth

- Memory bandwidth control using Intel's Dynamic Resource Control (DRC) [1].
  - LC services' CPI: no noticeable changes.
  - Median memory memory access latency: ~100 ns -> ~140 ns.
  - Median memory bandwidth utilization: ~15% -> ~30%.
  - The throughput of batch jobs also sees an order-of-magnitude improvement.



Architectural overview of Dynamic Resource Control [1]

[1] Zhang et al., LIBRA: Clearing the Cloud Through Dynamic Memory Bandwidth Management. In Proc. IEEE HPCA 2021

Handling Seasonal Shopping Festivals

A Case Study

Handling Seasonal Shopping Festivals: A Case Study

- Alibaba's e-commerce platform hosts a number of ***Seasonal Shopping Festivals*** (***SSFs***) around the year, e.g., on Nov. 11.
- Please refer to our paper for more details.

Full-day sales on Tmall of an SSF held on Nov. 11, 2020:
498.2 billion RMB ($68.2 billion USD)

Image credit to Xinhuanet: http://www.xinhuanet.com/english/2020-11/12/c_139511564.htm

Conclusion

- Cluster-wide macro-management:
  - Host & GPU memory are the bottlenecks in resource overcommitment.
  - Proactive memory reclamation.
  - Tidal scaling.

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Alibaba Group 阿里巴巴集团

Conclusion

- Node-level micro-management:
  - **CPU tiny jitters** and **memory bandwidth contention** can undermine LC services' QoS.
  - **Shared CPU pool** and **burstable CFS controller** to reduce the impacts of tiny CPU spikes on applications' performance.
  - Introduced **Intel's Dynamic Resource Control (DRC)** to adaptively regulate memory bandwidth contentions among applications.

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Alibaba Group 阿里巴巴集团

Conclusion

- Handling **seasonal shopping festivals**:
  - We leveraged these techniques in our shopping festivals to handle exponentially surging user traffic at minimum resource cost.

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Alibaba Group
阿里巴巴集团

Acknowledgement

- We thank numerous colleagues at Alibaba who have implemented and maintained this system.

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Alibaba Group 阿里巴巴集团