
KubeCon
CloudNativeCon
North America 2024

KubeCon | CloudNativeCon

North America 2024

Love thy (Noisy) Neighbor

Strategies for Mitigating Performance Interference in Cloud-Native Systems

Run 20%-50%
more transactions

Reduce tail latency
by 20-80%

2011 2013 2015 2016 2018 2019 2020 2021 2023

BubbleUp

Heracles

PerfIso

Caladan

Themis

iBench

Dirigent

PARTIES

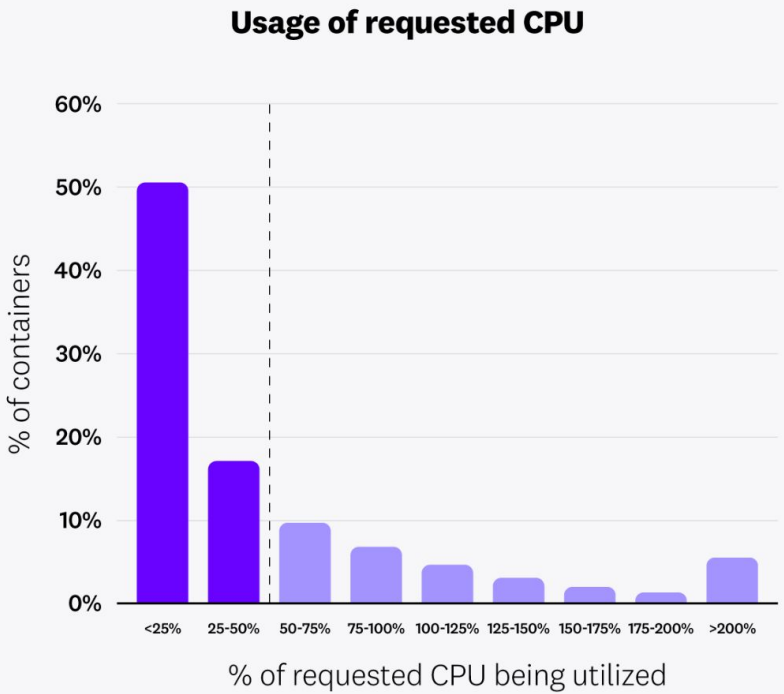
Libra

Can be made widely useful

CPI²

CAT at Scale

Alita



>65% OF CONTAINERS USE LESS THAN HALF OF REQUESTED CPU

Source: Datadog

source

Survey: cluster utilization

Raise hand if:

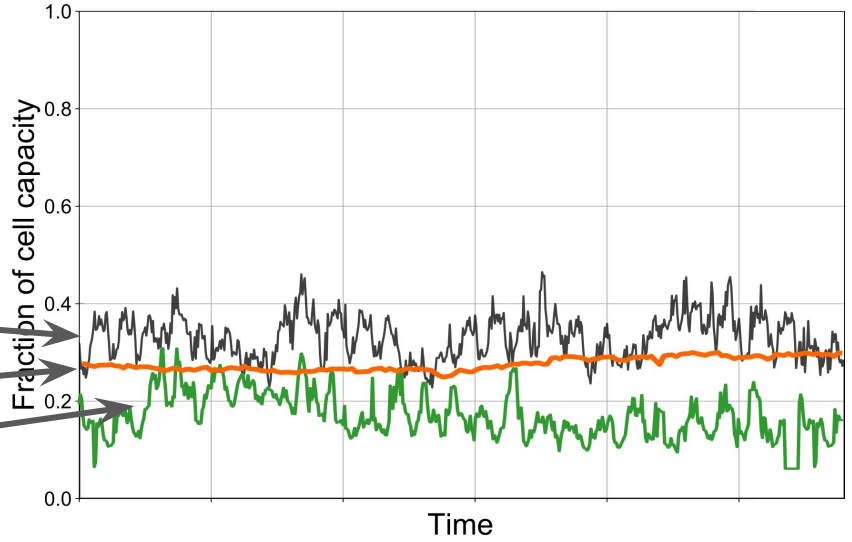
- Know prod cluster avg. CPU utilization
- Above 20%?
- Above 30%?
- Above 40%?
- Above 50%?

Hyperscaler CPU was low…

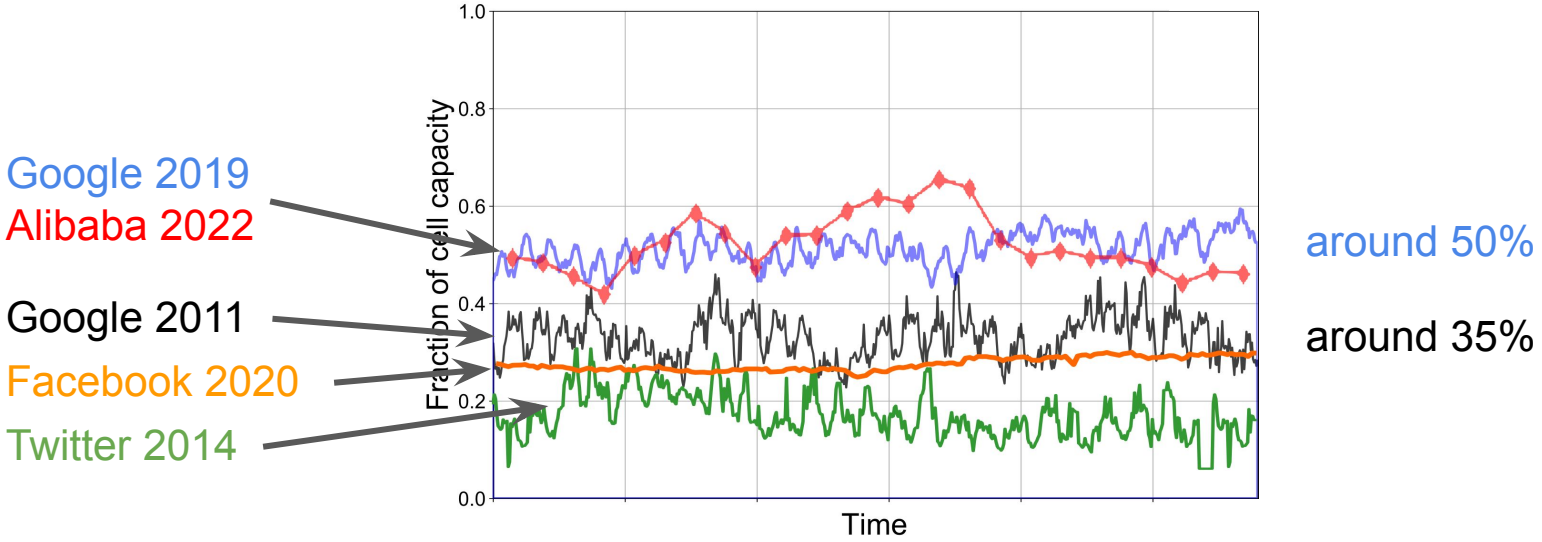
Google 2011

Facebook 2020

Twitter 2014



Then.. breakthrough?



Vertical autoscaler

Google's Autopilot (EuroSys'20, not GKE)
StormForge, PerfectScale, FairWinds, …

Less wasted resources → Better packing

Handle noisy neighbor

- Reduce cycles per request
- Improve tail latency

→ Less hardware at better SLA

This talk

Talk Outline

The Problem

Current Support

Mitigation Systems

Next Steps

~~deep dive to 1-2 papers~~ broad overview

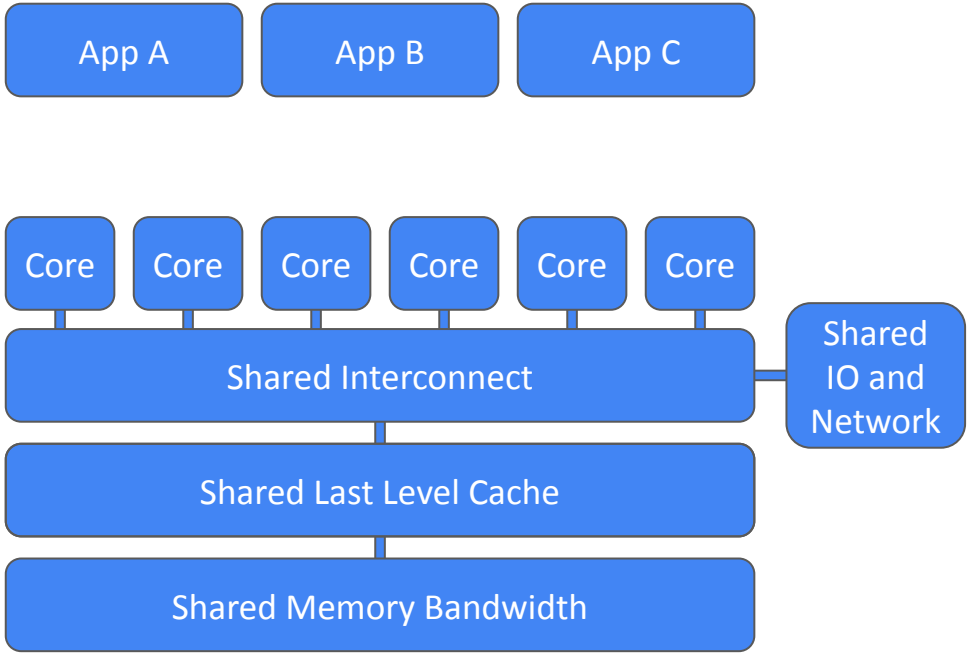
Problem

- Memory noisy neighbor
- How does this affect my Pods?
- Do I have it in my cluster?

Noisy neighbor: tragedy of the commons

Apps access physical resources

Shared resources are constrained



Noisy neighbor: tragedy of the commons

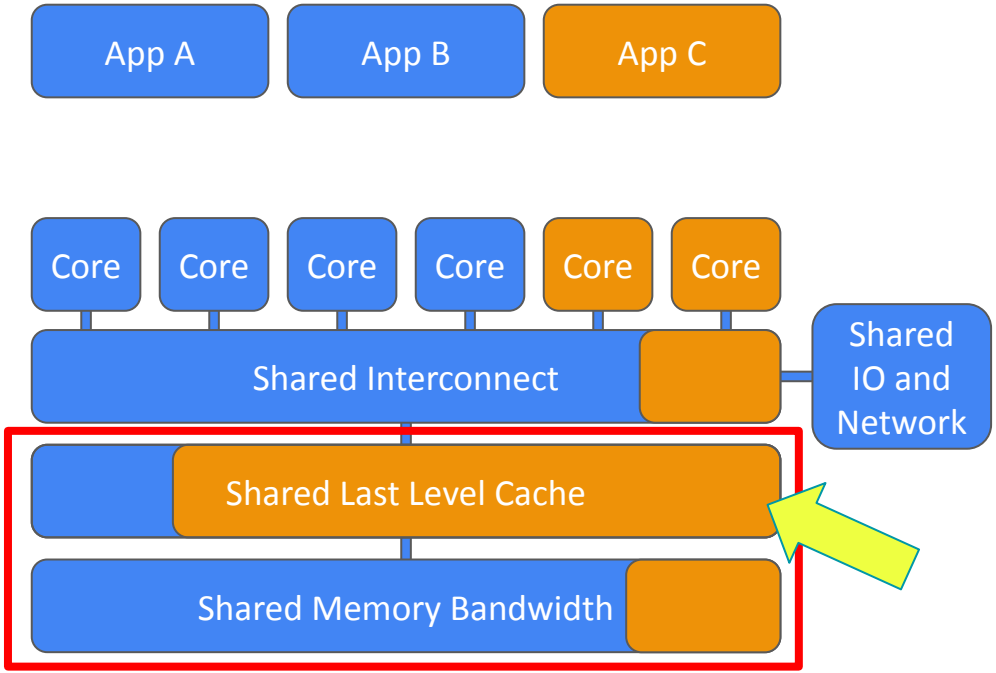
Apps access physical resources

Shared resources are constrained

One App can use more than its fair share, degrading others

This talk:

- Cache
- Memory bandwidth

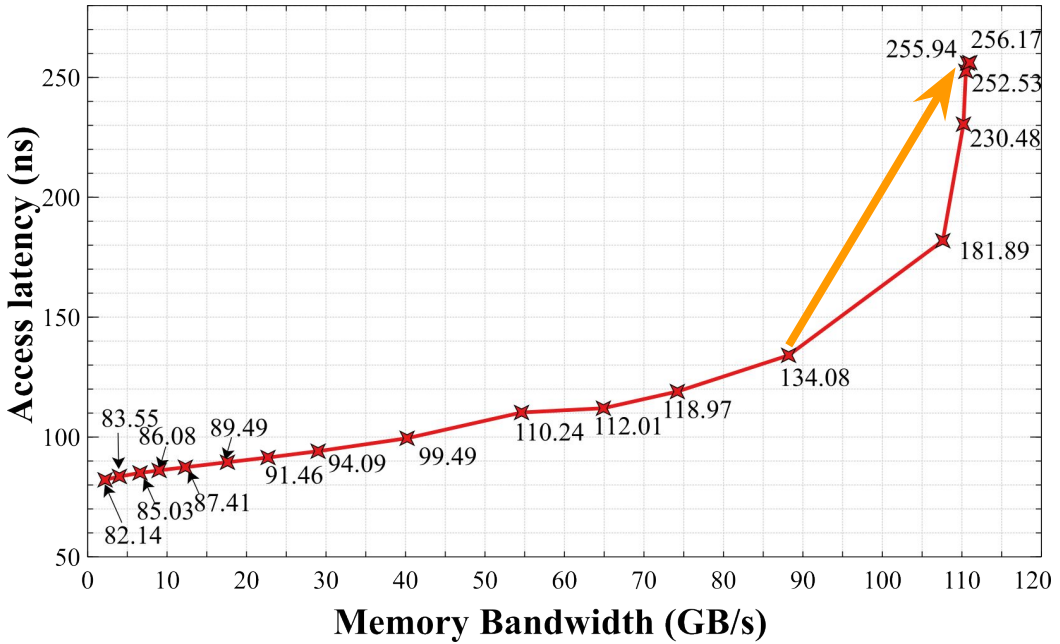


Latency increases with memory bandwidth

Change memory bandwidth,
Measure latency

Knee-point around 80%

80% → 100% bandwidth
latency doubles!



W. Tang et al, “Themis: Fair Memory Subsystem Resource Sharing with Differentiated QoS in Public Clouds,” ICPP ’22. doi: 10.1145/3545008.3545064.

CPUs try to protect us from memory latency:

- Prefetchers
- Reorder buffer (ROB)
- Caches

Are we protected?

…or not?

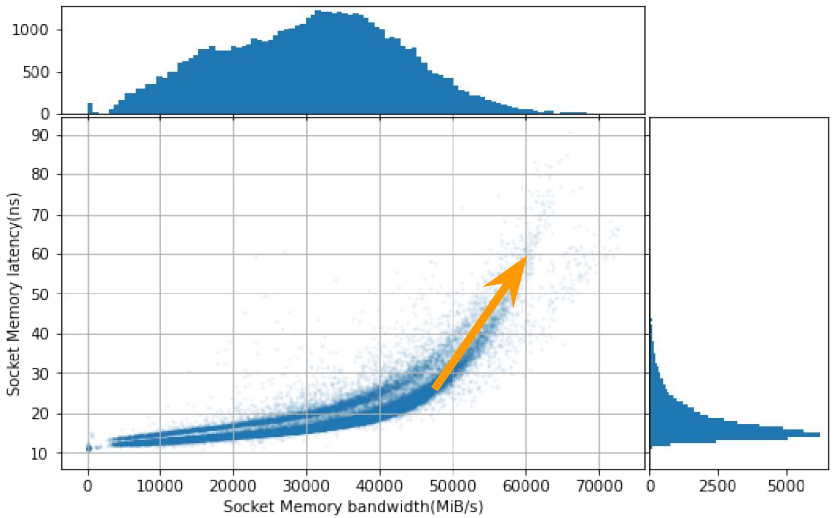
Measuring slowdown: CPI

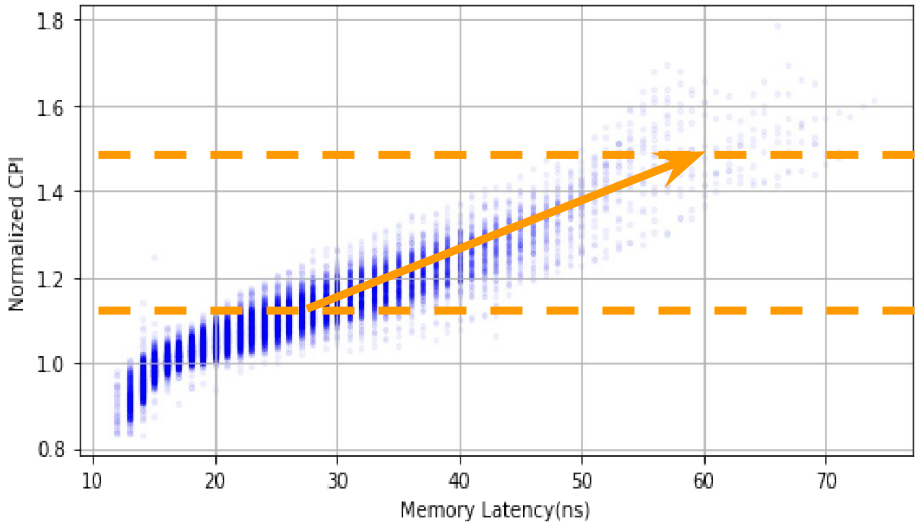
Does memory latency cause CPU memory stalls?

Cycles Per Instruction – **CPI**

CPU waits for memory → Stall cycles → High CPI

80% bandwidth cap → 25% more compute-efficient





Memory bandwidth only – has cache mitigation

K. Wang *et al.*, “Characterizing Job Microarchitectural Profiles at Scale: Dataset and Analysis,”, *ICPP*’22. doi: 10.1145/3545008.3545026.

Tail latency can explode with noisy neighbors!

websearch	10%	20%	30%	40%	50%	60%	70%	80%	90%
baseline(approx)	52%	57%	61%	60%	63%	62%	66%	77%	88%
LLC(small)	103%	96%	102%	96%	104%	100%	100%	103%	103%
LLC(med)	106%	99%	111%	103%	116%	[illegible]	110%	125%	111%
LLC(big)	>300%	>300%	>300%	>300%	>300%	[illegible]	[illegible]	264%	123%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	270%	122%

> 4.75x

ml_cluster	10%	20%	30%	40%	50%	60%	70%	80%	90%
baseline(approx)	5[illegible]	58%	57%	59%	56%	58%	58%	60%	68%
LLC(small)	88%	84%	110%	93%	216%	106%	105%	206%	202%
LLC(med)	88%	91%	115%	104%	>300%	212%	220%	212%	205%
LLC(big)	>300%	>300%	>300%	>300%	>300%	>300%	>300%	250%	214%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	287%	223%

> 5.35x

memkeyval	10%	20%	30%	40%	50%	60%	70%	80%	90%
baseline(approx)	20%	20%	21%	21%	22%	29%	35%	42%	34%
LLC(small)	88%	91%	101%	91%	101%	138%	140%	150%	78%
LLC(med)	148%	107%	119%	108%	138%	230%	181%	162%	100%
LLC(big)	>300%	>300%	>300%	>300%	>300%	>300%	280%	222%	79%
DRAM	>300%	>300%	>300%	>300%	>300%	>300%	>300%	234%	103%

> 13.6x

Google, 2015

3 production services

get % of SLO target (99th / 95th percentile)

w/ synthetic noise generators

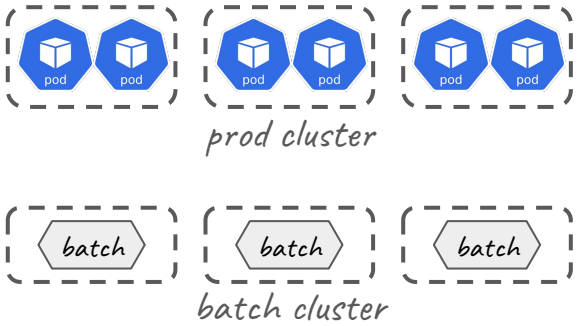
D. Lo et al. “Heracles: improving resource efficiency at scale,”, ISCA ’15. doi: 10.1145/2749469.2749475.

Survey: Node size

Please raise hand if:

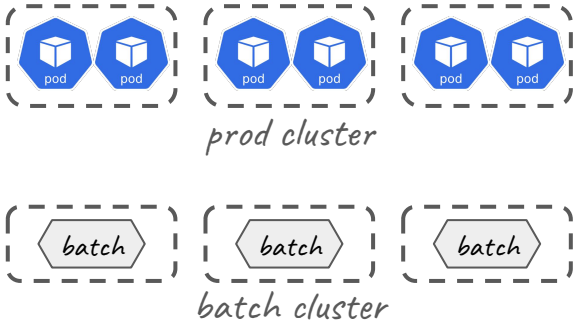
- Know what VMs or bare-metal are used in prod
- Never use fraction of physical CPU

Separating batch clusters

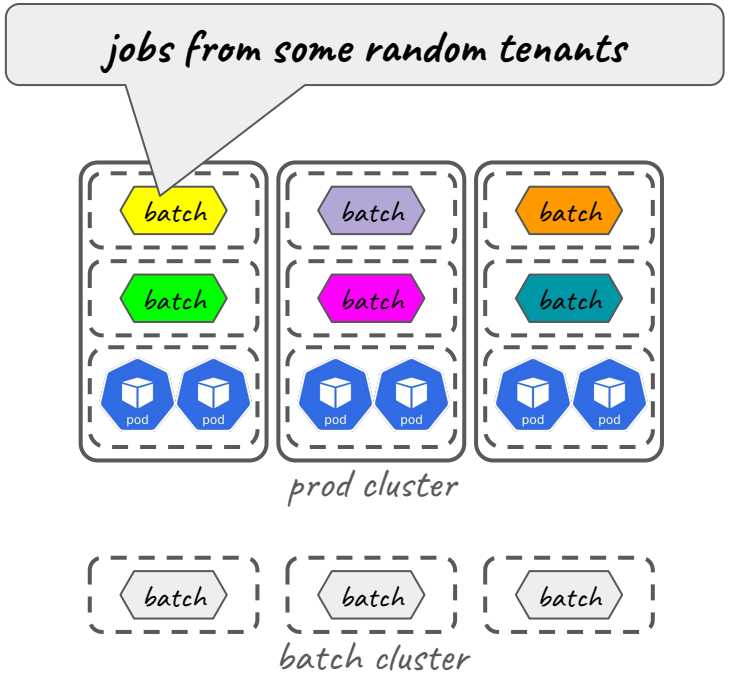


What I think I'm running

Separating batch clusters



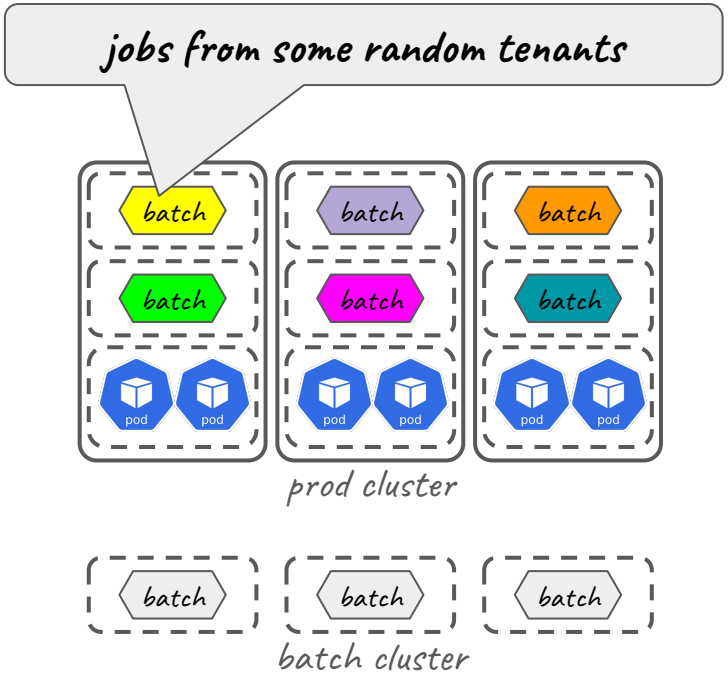
What I think I'm running



What I'm actually running

Separating batch clusters

Does your provider protect your VMs from other VMs on the same machine?



What I'm actually running

Engineers spend years
optimizing user experience

Does my cluster have noisy neighbors?

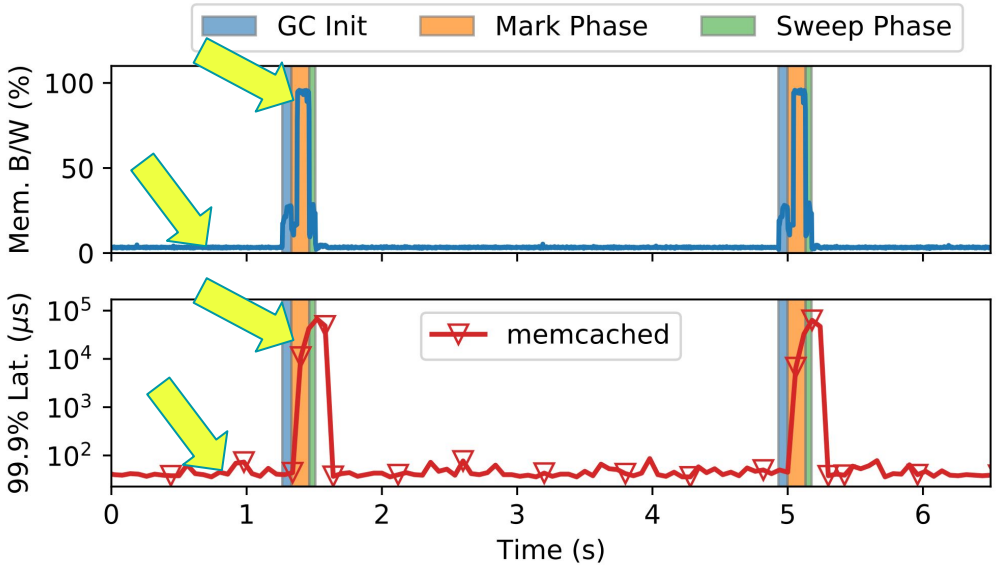
Run:

- Memcached
- garbage-collected workload

Mark Phase is memory-intensive, causes significant slowdown!

Not only “big-data” is noisy

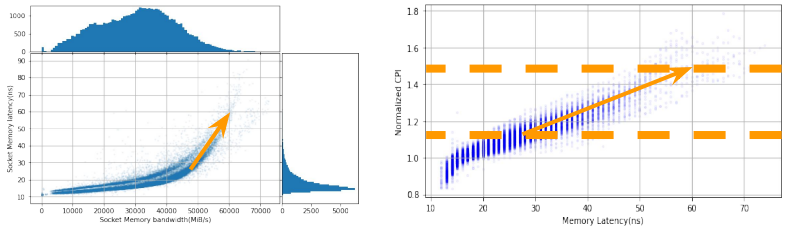
Also: security scanning, video streaming, transcoding…



J. Fried et al., “Caladan: Mitigating Interference at Microsecond Timescales,”. OSDI’20. on usenix.org

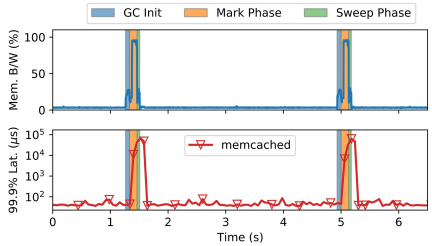
Recap: The Problem

High memory contention
→ High memory latency
→ High CPI (low efficiency)



Tail latency with noisy neighbor increases 4-13x

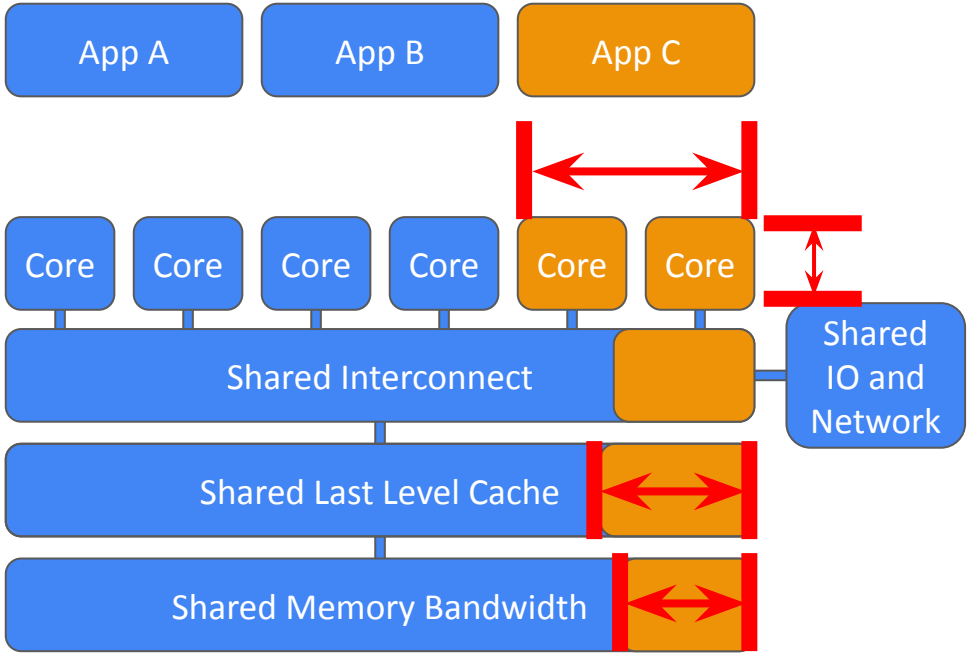
Many workloads can be noisy (e.g., garbage collection)



Current support

- Hardware and software mitigation
- Support in Linux

Memory system mitigation knobs



Reduce demand (#cycles):

- Core pinning
- Frequency scaling

Direct control:

- Cache allocation
- Memory bandwidth limits

Containers (not) to the rescue!

Different focus:

- cycles
- memory capacity (size)

Cgroup v2	Responsibility
CPU	Regulates distribution of CPU cycles
Memory	Controls distribution and accounting of memory usage
IO	Manages distribution of IO resources
PID	Limits the number of processes in a cgroup
Cpuset	Assigns specific CPU(s) and memory nodes
Device	Controls access to device files
RDMA	Regulates distribution and accounting of RDMA resources
HugeTLB	Limits the HugeTLB usage per control group
Perf_event	Allows perf events to be filtered by cgroup path
Misc	Provides resource limiting for scalar resources not covered by other controllers

Reading 1 2

Linux supports allocation

~~cgroup~~ resctrl allocates:

- Memory bandwidth
- Cache space

Evolved:

- Intel (2016)
- AMD (2018)
- ARM (in progress)

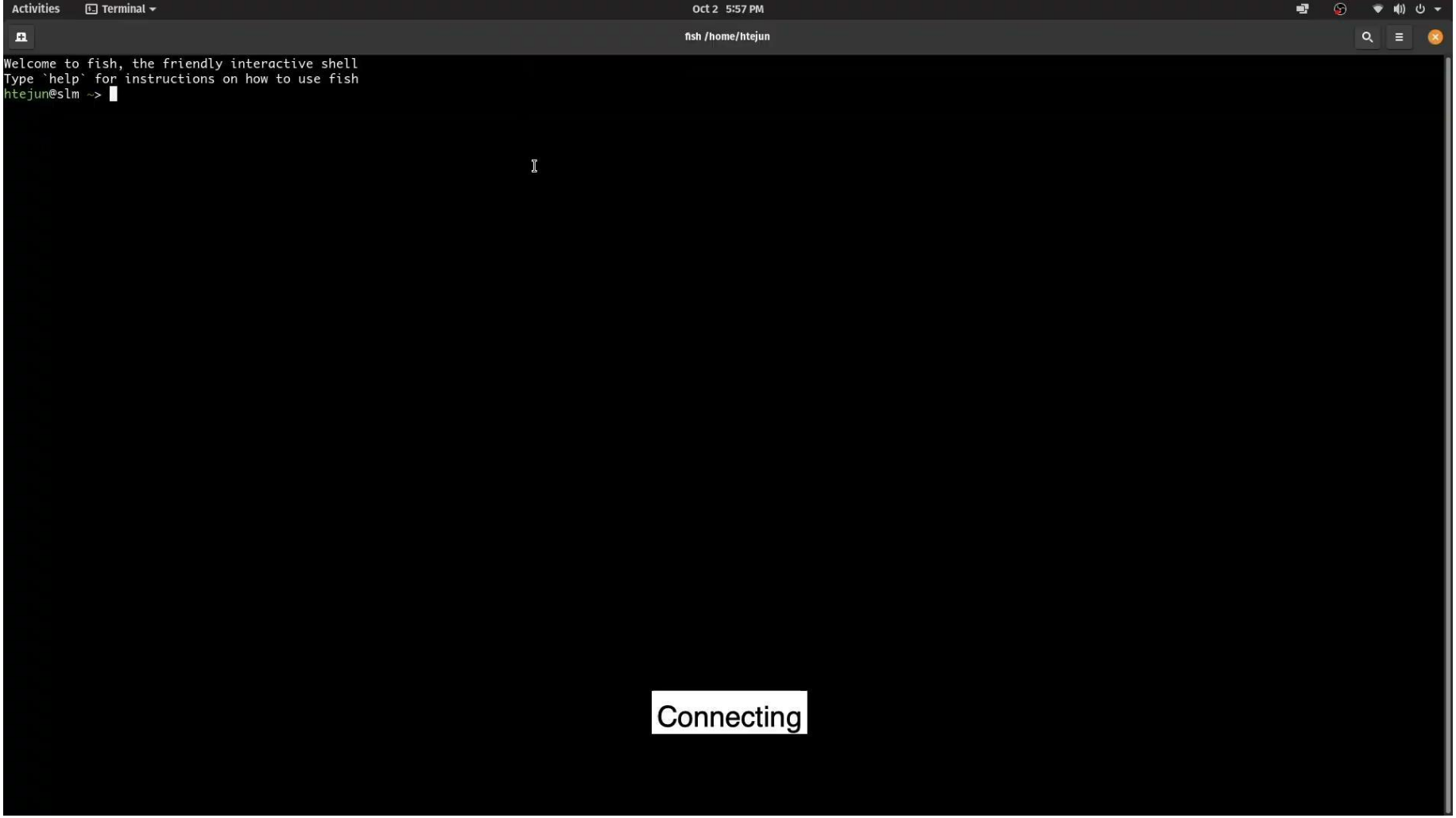
Demo:

- Facebook Resource Control Demo by Tejun Heo and collaborators

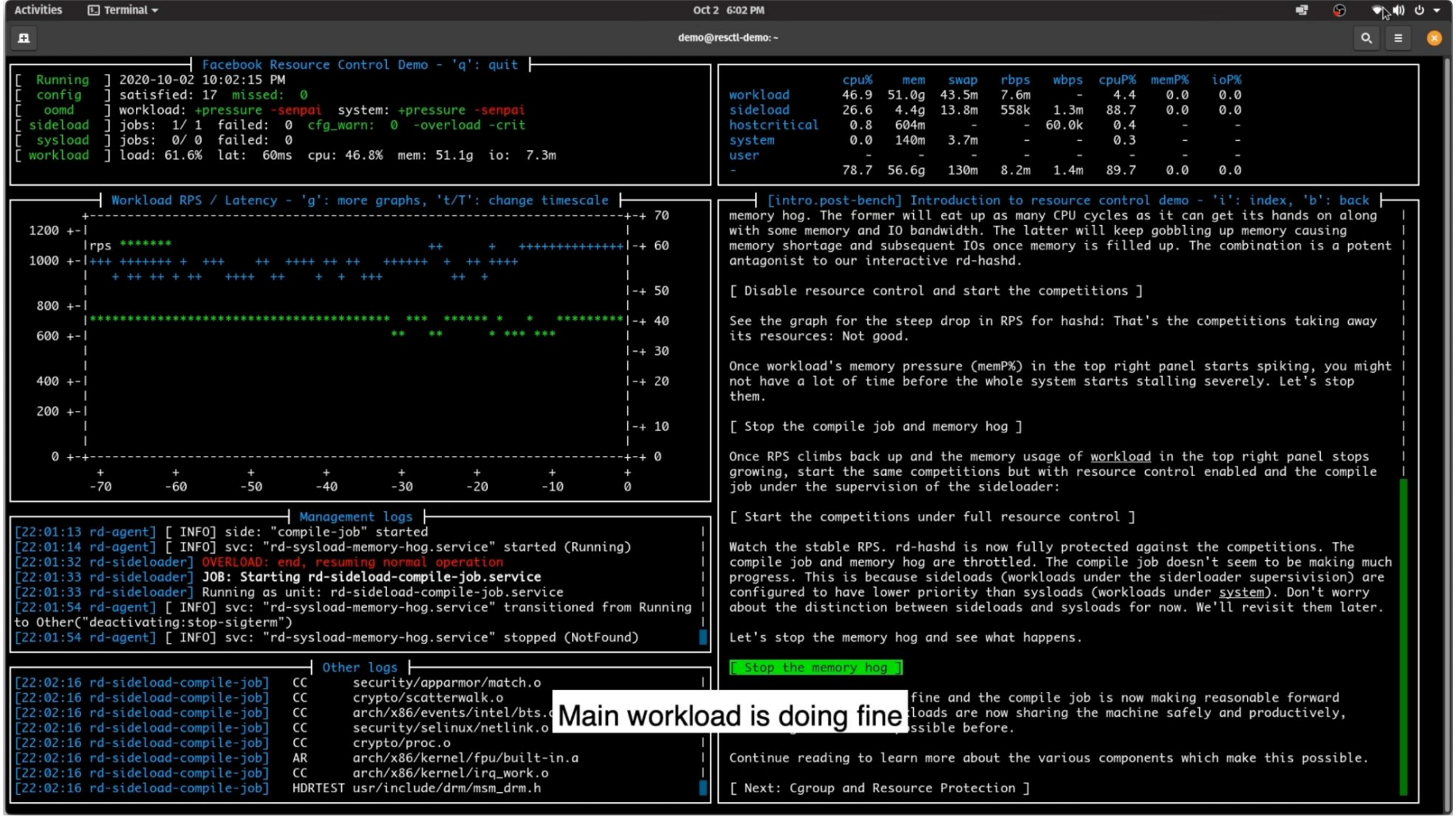
Linux resctrl CPU support

	Kernel
Intel RDT Intel Resource Director Technology	v4.10, v4.12
AMD QoS AMD Platform Quality of Service	v5.0
ARM MPAM Memory System Resource Partitioning and Monitoring	WIP, x86→generic Review needed

AMD, ARM



Credit: Meta; Tejun Heo and collaborators. [resctl-demo on GitHub](#).



Credit: Meta; Tejun Heo and collaborators. [resctl-demo on GitHub](https://github.com/facebookexperimental/resctl-demo).

Hypervisors support allocation

- VMware in Telco Cloud Automation

  → Telco + Finance?
- Static

Intel RDT support in hypervisors

	Cache partitioning	Memory bandwidth
Xen	✔	✔
KVM	✔	
VMware	✔	Monitor (vSphere)
Hyper-V		Monitor PMU
ACRN	✔	✔

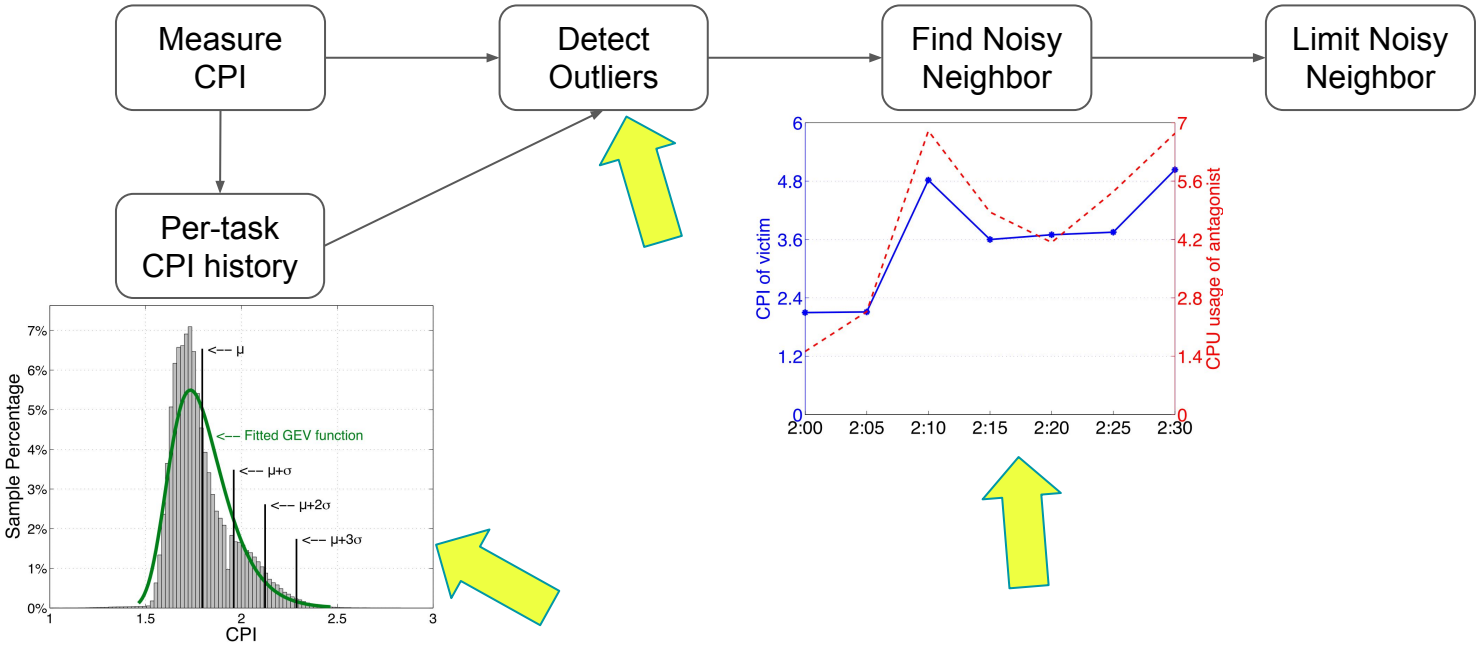
Telco Cloud Automation, KVM libvirt NUMA tuning, Xen memory bandwidth, Xen LLC, ACRN, Hyper-V PMU

Mitigation Systems

- Type 1: cycles per instruction (CPI)
- Type 2: latency control
- Type 3: usage control

Type 1: Cycles Per Instruction (CPI)

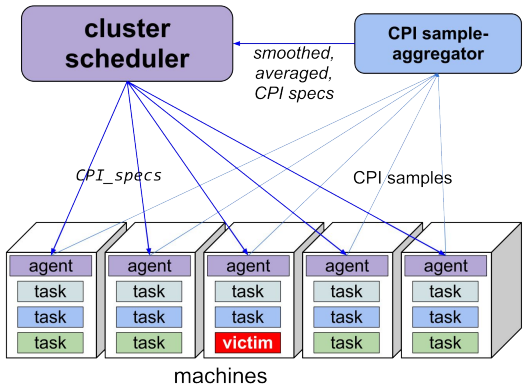
Uses: High interference → high CPI



X. Zhang et al., "CPI [2]: CPU performance isolation for shared compute clusters," in *EuroSys 2013*. doi: 10.1145/2465351.2465388.


I'll say when to stop!

Type 1: Cycles Per Instruction



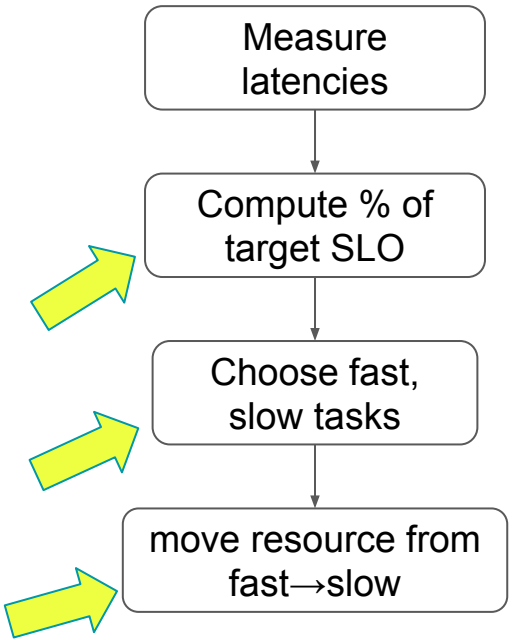
"We have rolled out CPI^2 to all of Google's shared compute clusters." – paper authors, 2013 @Google

	Type 1: CPI
Measurement	Cycles, Instructions
Averaging	High
Cluster components	Aggregator
Pros	• Simple to measure
Cons	• Complex deployment • Averaging → Slow reaction

X. Zhang et al., "CPI ^{2}: CPU performance isolation for shared compute clusters," in *EuroSys 2013*. doi: 10.1145/2465351.2465388.

Type 2: Latency Control

Example algorithm:



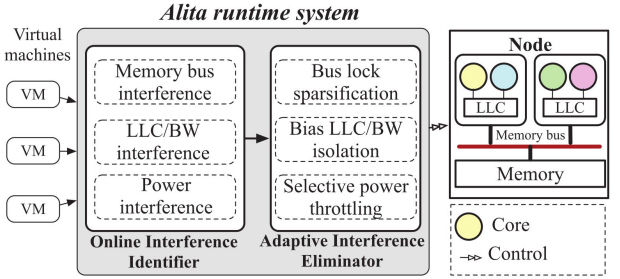
	Type 2: Latency
Measurement	App latency
Averaging	Medium
Cluster components	Node only
Pros	• No profiling → Node-local • Control what you care about
Cons	• High developer effort • Noisy signal → Averaging

S. Chen et al. “PARTIES: QoS-Aware Resource Partitioning for Multiple Interactive Services,” in ASPLOS ’19. doi: 10.1145/3297858.3304005.

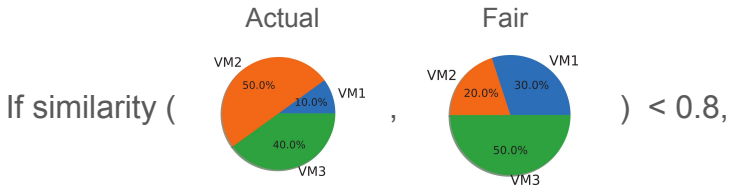
Type 3: Usage Control

How:

- Measure per-app resource usage
- Find unfair allocation
- Limit offender



Cache example:



then limit cache polluters to their fair share

Q. Chen *et al.*, “Alita: Comprehensive Performance Isolation through Bias Resource Management for Public Clouds,” in *SC20:* link.

Do we really want a fair allocation?

Type 3: Usage Control (cont'd)

	Type 3: Usage
Measurement	CPU counters
Averaging	Low
Cluster components	Node only
Pros	• Node-local • Measure directly → Fast reaction → Easy to reason
Cons	• Do we really want fair allocation?

Deployed in production, > 2 years:

- 30k nodes (24 to 48 cores each)
- 250k VMs

- authors, 2020 @Alibaba Cloud

Q. Chen *et al.*, "Alita: Comprehensive Performance Isolation through Bias Resource Management for Public Clouds," in *SC20:* link.

Summary: Mitigation Systems

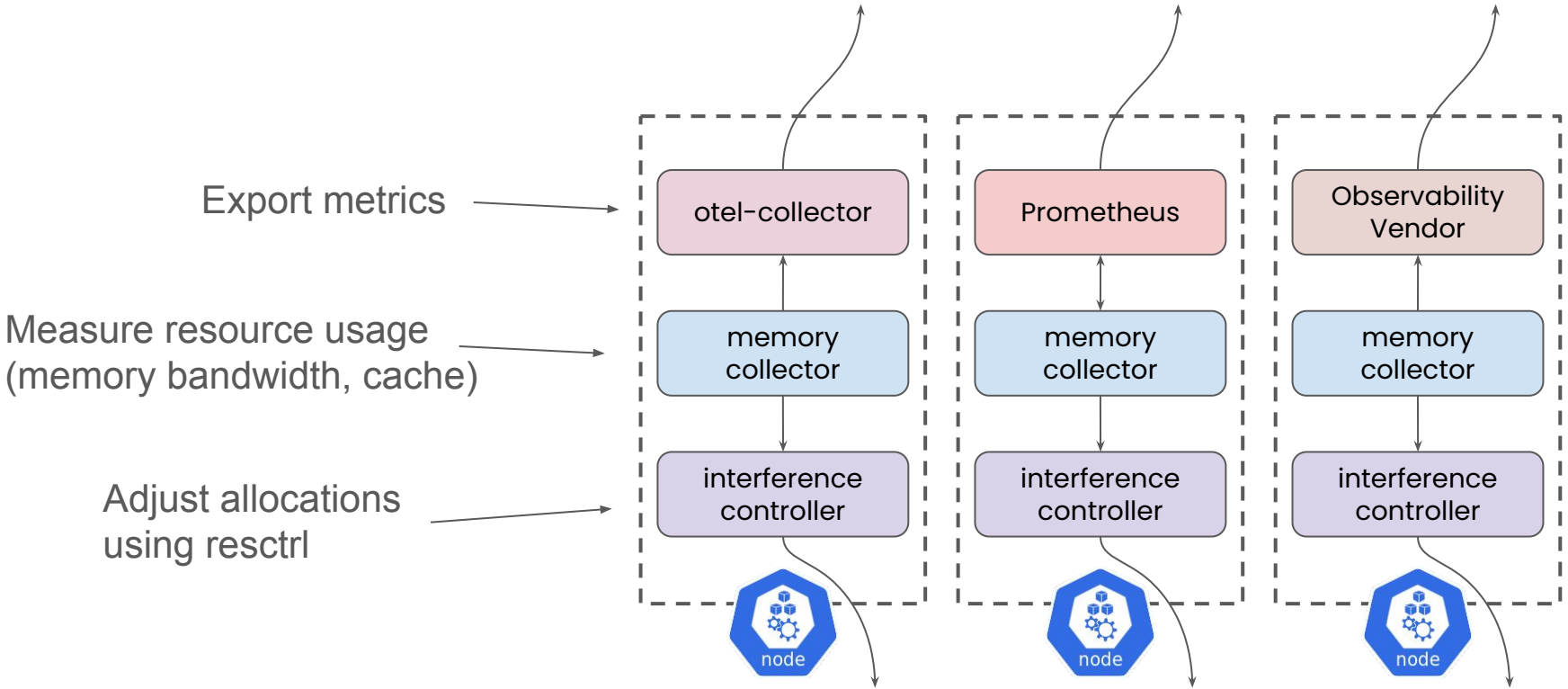
	Type 1: CPI	Type 2: Latency	Type 3: Usage
Measurement	Cycles, Instructions	App latency	CPU counters
Averaging	High	Medium	Low
Cluster components	Aggregator	Node only	Node only
Pros	• Simple to measure	• No profiling → Node-local • Control what you care about	• Node-local • Measure directly → Fast, Easy to reason
Cons	• Complex deployment • Averaging → Slow reaction	• High developer effort • Noisy signal → Averaging	• Do we really want fair allocation?

- The three “categories” of published systems
- Many good ideas → general purpose
- Usage control (#3) is promising

Next Steps

- Kubernetes Deployment
- Observability
- Community Next Steps

Kubernetes Deployment

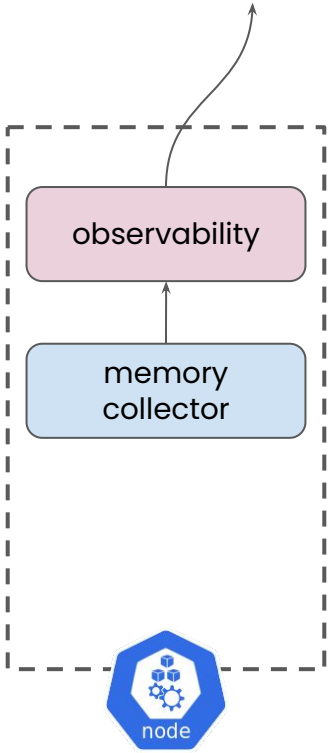


First step

Start with observability, because:

To deploy with confidence, need

- Quantify the benefit
- Metrics to show it is behaving correctly



Community Next Steps

perfpod/memory-collector

Call for:

- Contributors
  - Collector
  - Kubernetes benchmarks
- Potential users
  - Want to hear more
  - Deploy in test/staging

Contact: yonch@yonch.com

Set up time: yonch.com/collector