Challenges and Directions in Modeling Cloud Performance

Modsim 2022

Abhishek Dhanotia

Meta

Agenda

01 Motivation - why model performance in private clouds?

02 Challenges with performance modeling @scale

03 Directions - Meta Perspective

Modeling private cloud performance

Goals

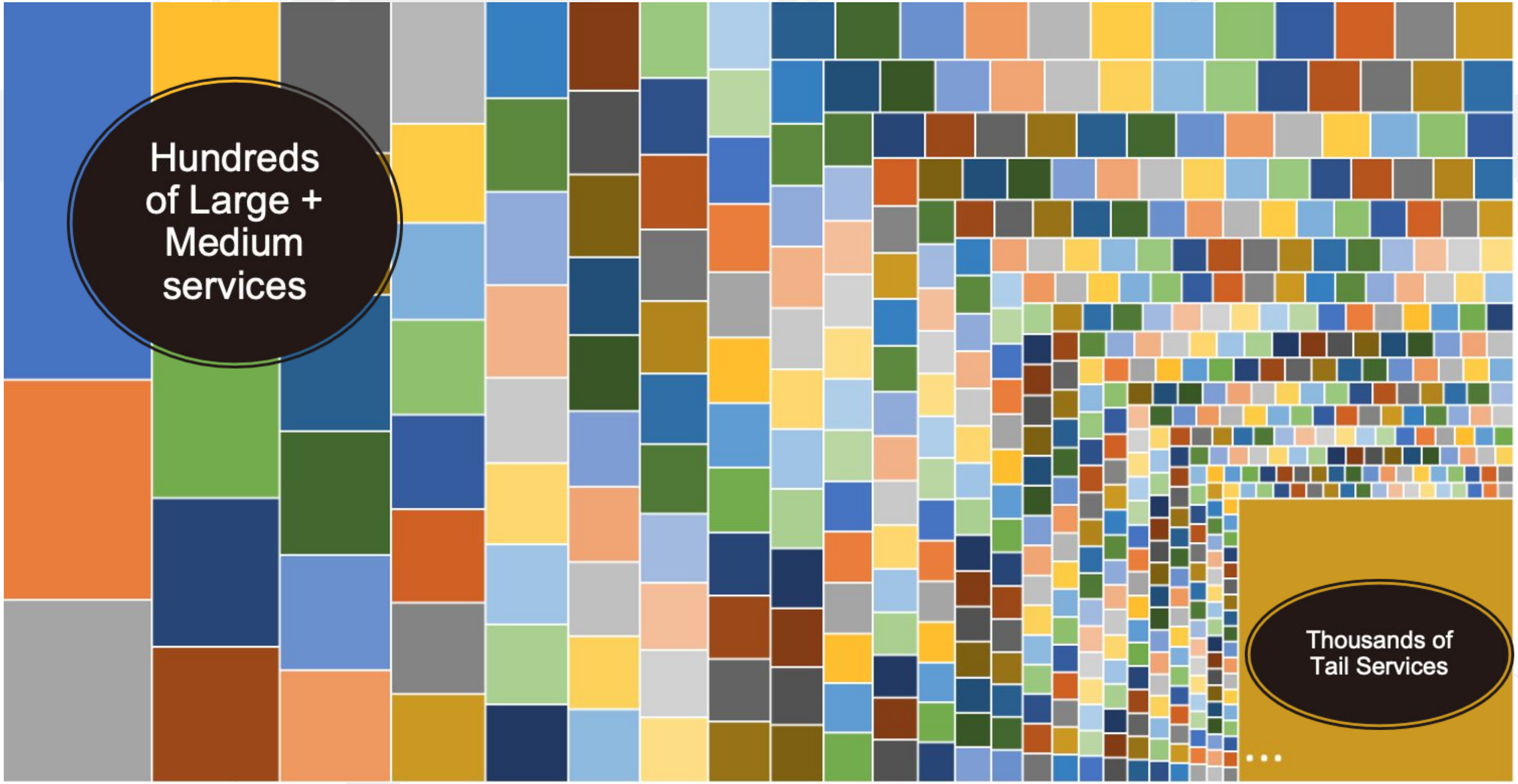
- ❖ Define and build a limited number of different server types (a.k.a system shapes)
- ❖ Optimize platforms for rack and fleet level efficiency
- ❖ Provide reliable performance and efficiency signals for capacity planning
- ❖ Ensure all possible workloads and use cases are supported via roadmap offerings

Non-Goals

- ❖ Build the biggest possible systems
- ❖ Maximize total available vCores or persistent bytes in datacenters

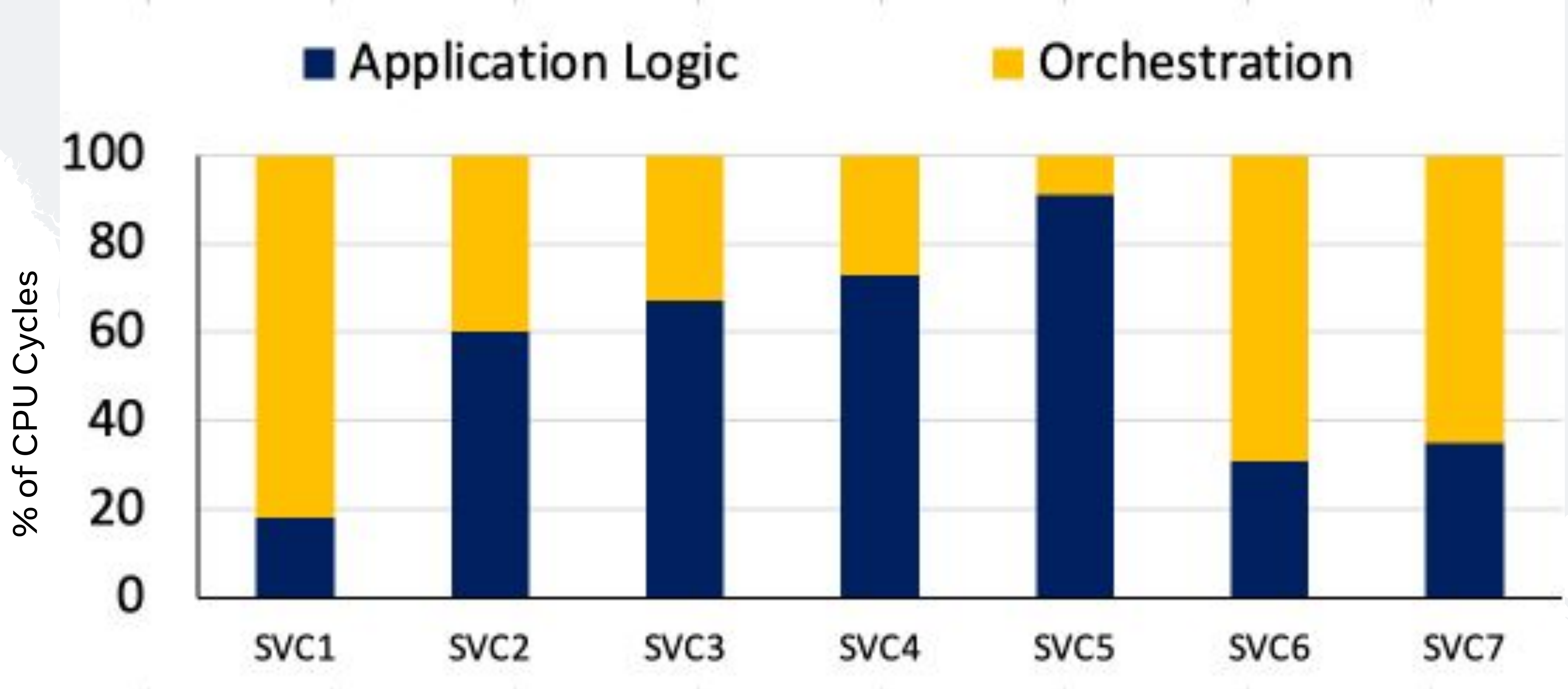
Challenges in modeling cloud performance

Challenge #1 – Workload Diversity



Looking at CPU cycles spent on one server type.
Multiple different server types to run workloads with different resource requirements
Practically impossible to model every individual service that runs in the fleet
Need creative ways to limit the scope of work needed for performance modeling

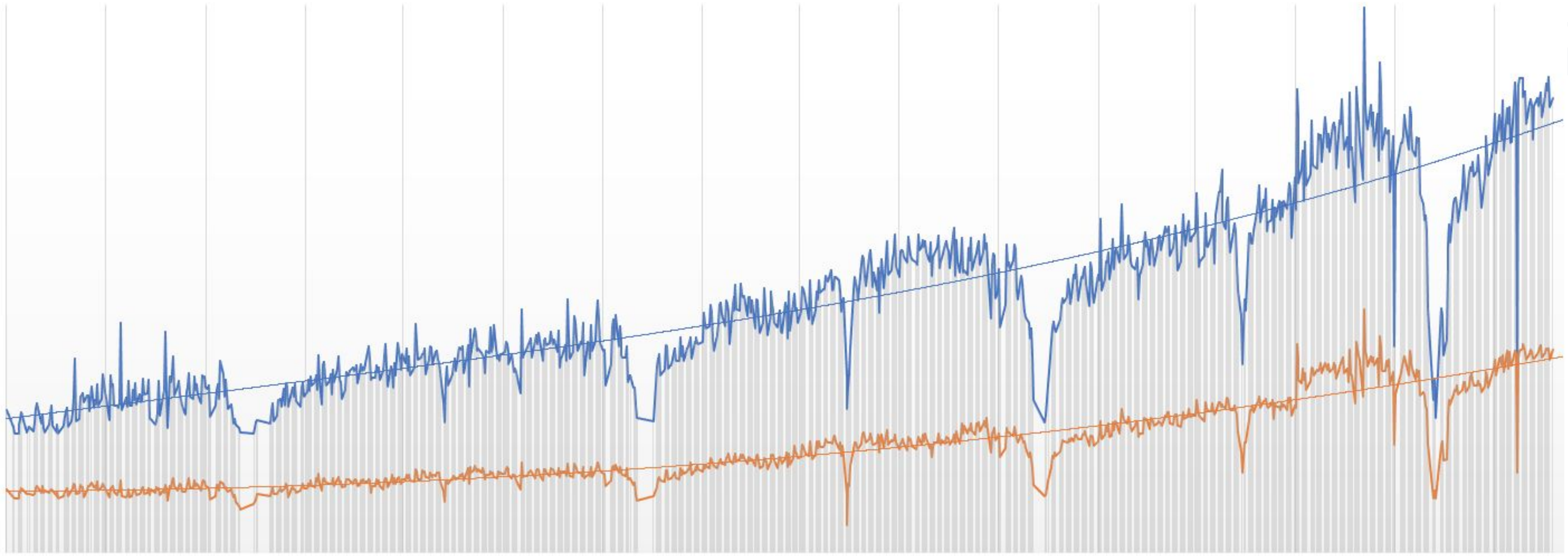
Challenge #2 - What qualifies as a cloud workload?



There is much more going on on the server than just the core application logic*
Not every part of the workload scales at the same rate and w/ same uArch or system architecture improvements
Optimizing just for the 'application logic' can lead to misleading results.

**Accelerometer paper (ASPLOS 2020) discusses orchestration logic and what it constitutes*

Challenge #3 - Evolving Codebases



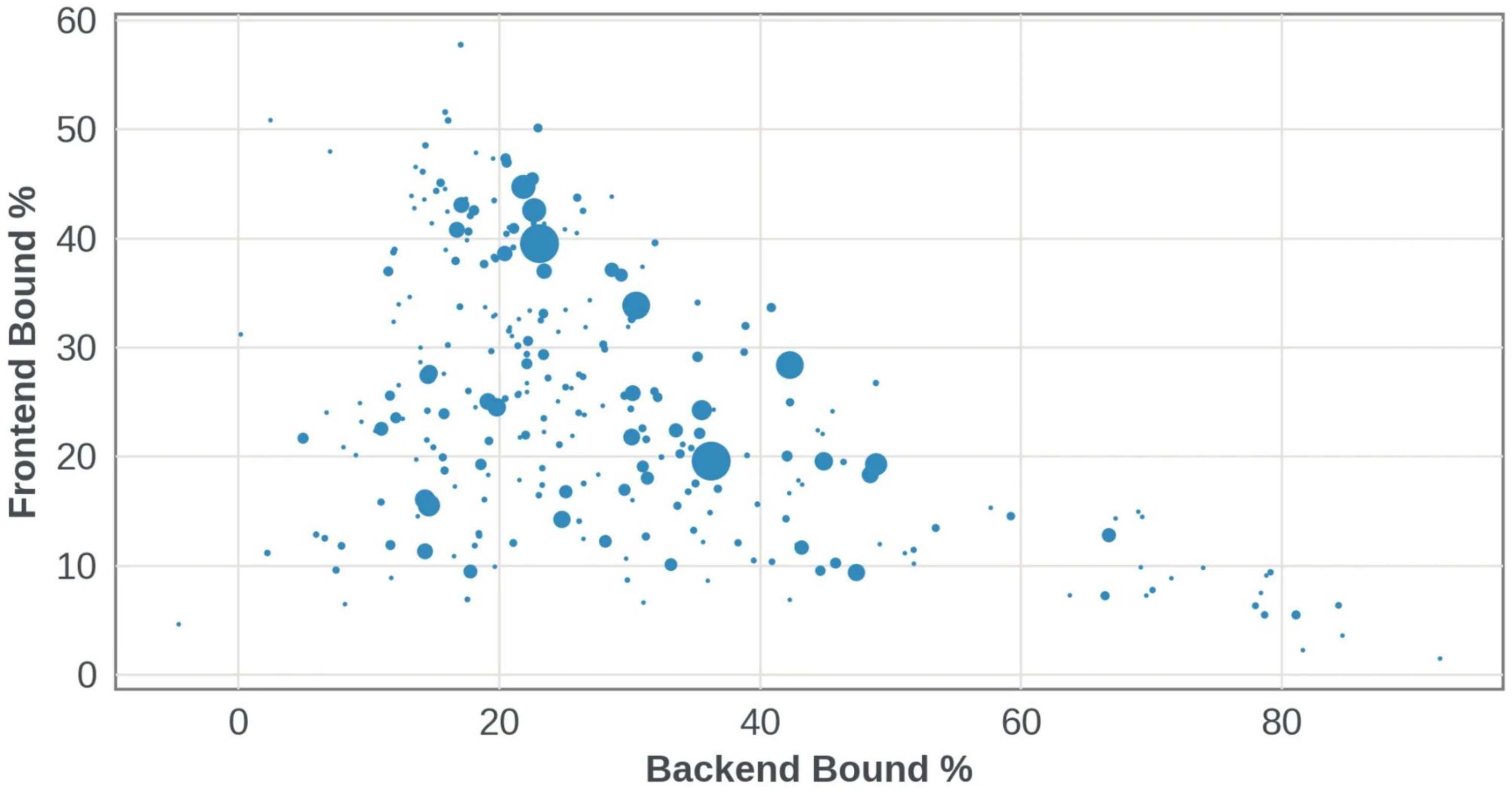
Committers and Commits per day on one stack over a year (large dips reflect holidays)
Multiple application stacks (usually one per service) are usually supported in every cloud environment
Scale of y-axis depends on the application stack and can vary between O(10) to O(1K)
Taking a snapshot of a workload at a point in time can lead to low fidelity results over medium-long term

Challenge #4 – Diverse resource bottlenecks across workloads

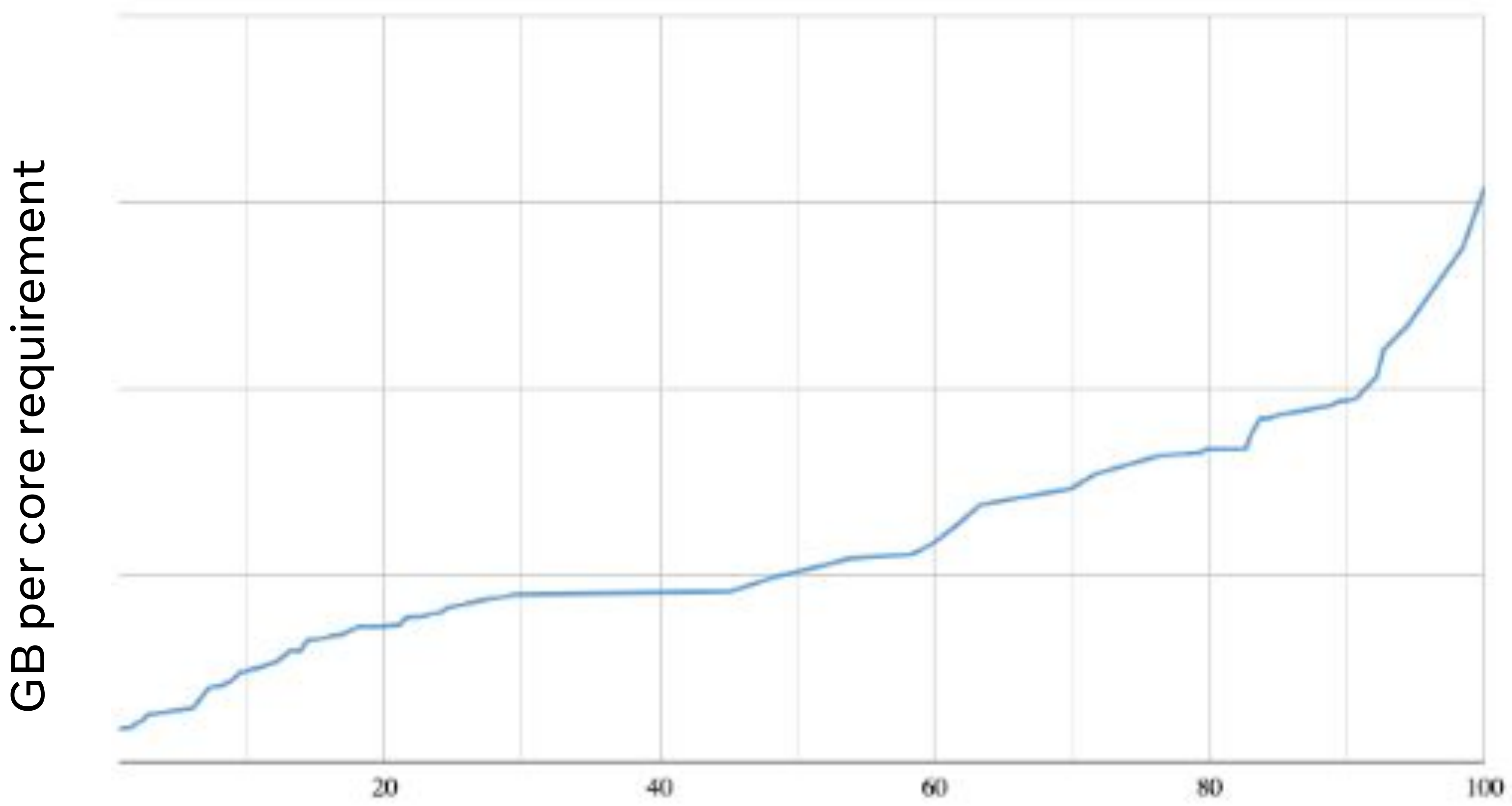
- Large variance in uArch level resource bottlenecks across workloads => over optimizing a particular use case can cause fleet level regressions

- Large variance in system level resource bottlenecks
  - important to find the sweet spot for fleet efficiency.
  - Important to stack (bin pack) as needed

- Subjective tradeoffs between building optimal HW system vs. SW resources



CPU FrontEnd vs. BackEnd bound across one server
SoftSKU Paper (ISCA 2020) discusses this in more detail



Memory per core usage across one server type
Transparent Page Placement Paper (arxiv) discusses this in more detail

Challenge #5 – What does efficiency mean? Which metrics to optimize?

Performance: ***But, what to use to measure performance given the large variety of workloads?***

Performance per Watt: ***But, what constitutes watt here? Is this max power or current power? How to factor in utilization levels and diurnal workload patterns?***

Total Cost of Ownership (TCO): ***But how to measure TCO? What constitutes cost in a cloud environment?***

vCores per Watt or per $: ***But, what if cores have different capabilities or workload has dependencies on multiple HW resources (CPU, GPU, Flash, Network)***

How to balance multiple metrics of interest?

Directions in modeling cloud performance

Direction #1 - Workloads

Evaluate a wide variety

- Data Ingestion - At line rate to keep trainers busy
- Distributed Training - Offline training w/ multiple model sizes and object size distributions
- Ranking and real time inference - Online services with varying sizes of ML models performing low latency ranking or inference.
- Warehousing and Analytics - In-memory and In-storage analytics with varying amount of back-end data to deal with.
- Caching - Multiple flavors (look aside, look through), varying object sizes, retention times and latency SLAs.
- End user (Web) Serving

Use data science and ML to find similarities and differences

- Not every workload has unique characteristics.
- No need to model performance for every application

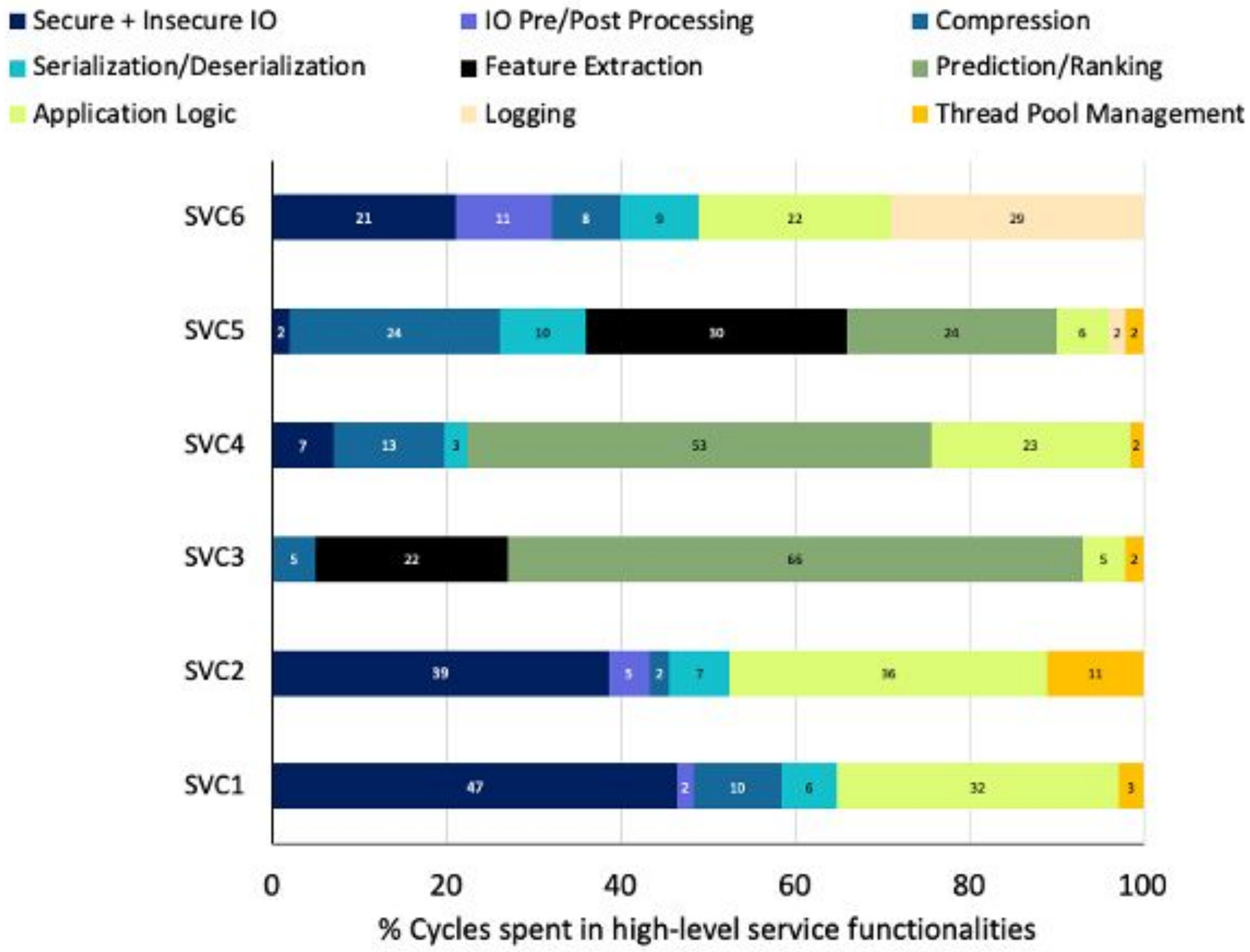
Direction #2 - Factor in orchestration overheads

- (u)Service oriented architecture of cloud applications => portion of cycles are spent talking to other applications

- These ‘orchestration’ work can take up significant CPU cycles, more than application logic in some cases

- Just speeding up application logic can lead to diminishing returns, might even cause performance regressions.

Performance models should capture end to end performance



Accelerometer paper (ASPLOS 2021) talks about this in more detail

Direction #3 - Representative benchmarks and right operating points

(for black-box environments or when production testing is not possible)

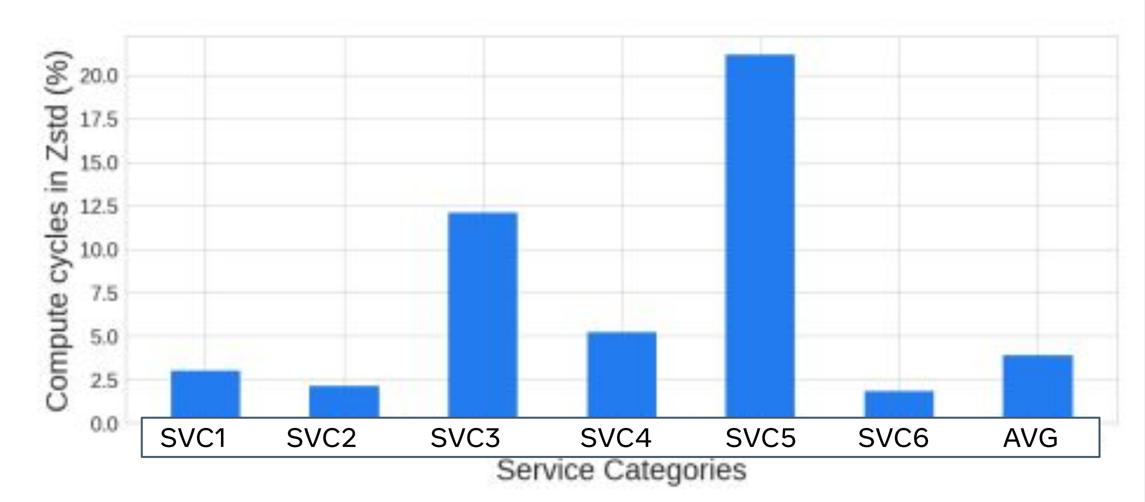
Taking data compression acceleration as an example

- ❖ Only running compression on the system is not useful
- ❖ Only running a single thread to extract max compression throughput is not useful
- ❖ Only running on very large block sizes or high compression level is not useful

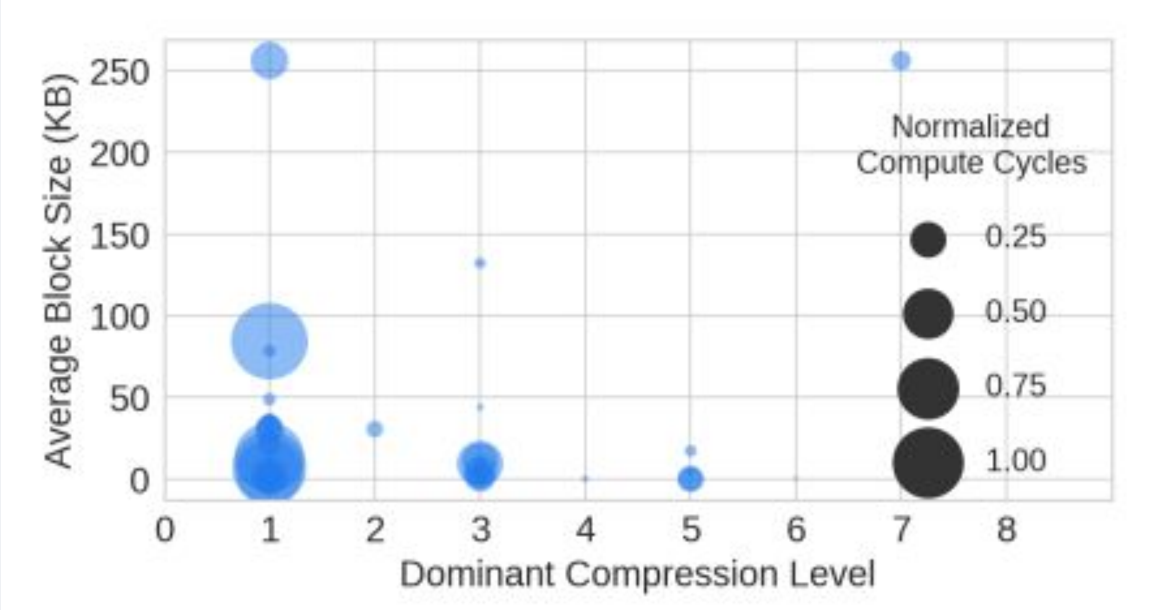
What **is** useful?

- ❖ Demonstrating benefit on an end-to-end representative benchmark that ***also*** does compression
- ❖ Demonstrating benefit across different use cases of compression in the fleet

We use end-to-end representative benchmarks developed in-house to capture these aspects



Percentage of CPU cycles spent doing compression



Relative CPU cycles spent on different compression levels and block sizes

Example use case for compression at Meta
ISPASS 2022 abstract talks about this in a bit more detail

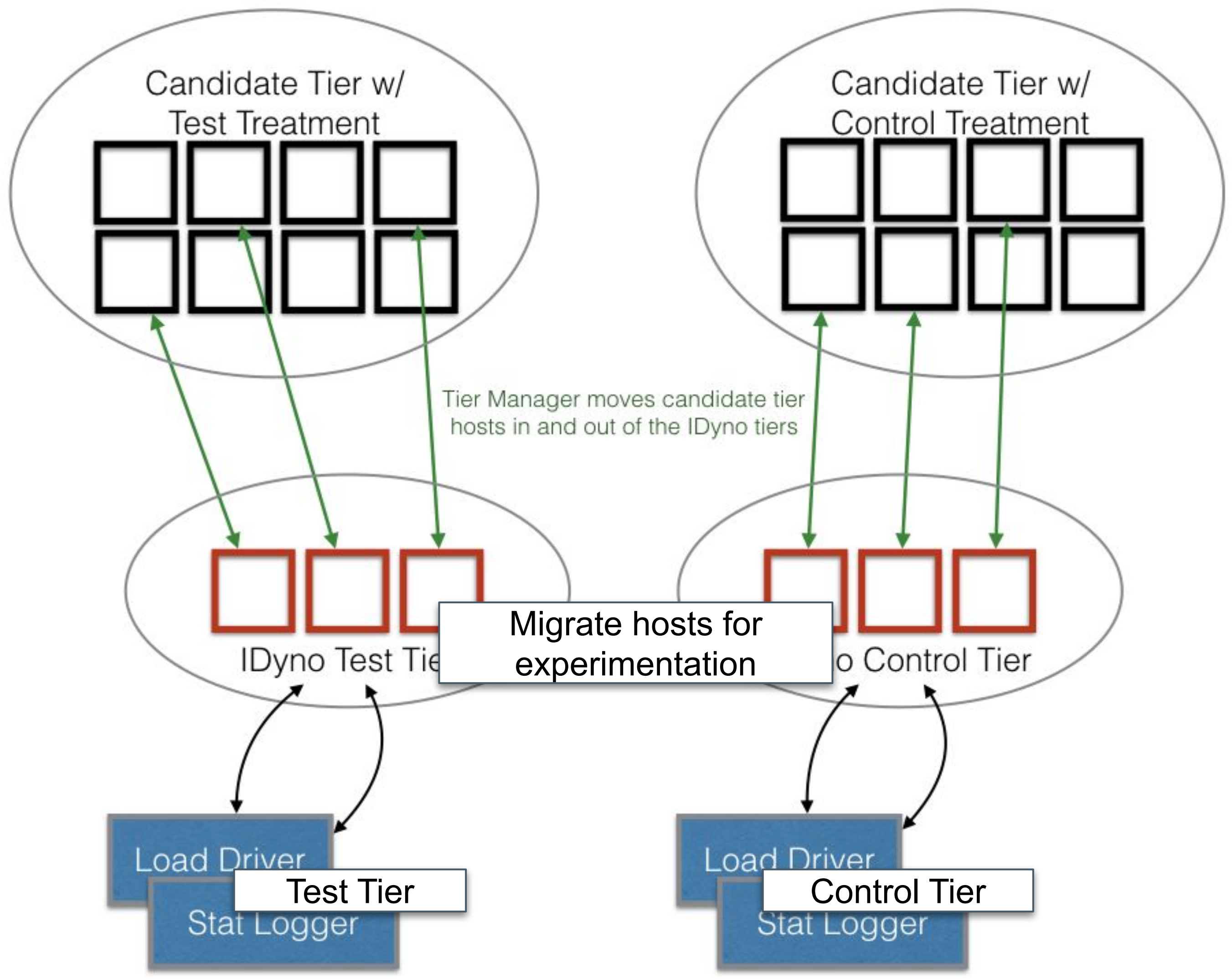
Direction #4 - Systematic load testing frameworks

(for white-box testing in production environments)

- Test with real binaries and production traffic (shadowed, replayed or live) when possible.
  - Helps naturally evolve the perf work with ‘current’ codebase and traffic patterns

- Push workload to the right saturation points
  - Latency SLAs mean that the machine is likely not going to run at 100% utilization
  - Fault tolerance SLAs mean leaving headroom for traffic spikes

- Enable A/B and B/A testing to account for variance and eliminate noise
  - Performance variance can be as high as 10% in @scale deployments



Kraken paper (OSDI 2016) provides an overview of Meta’s load testing framework

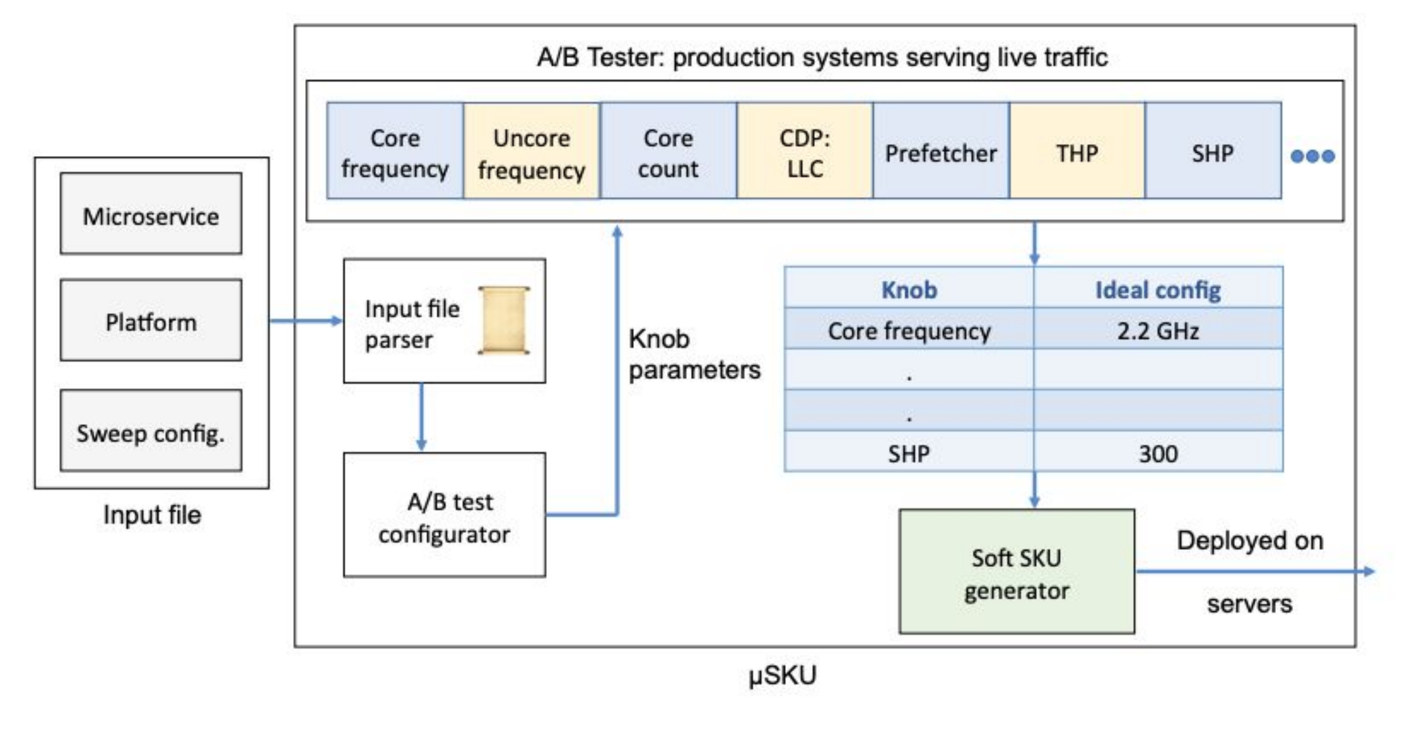
Direction #5 - System configuration

❖ Non-Goal: Build the largest system => The system might not have the max memory or CPU frequency or network bandwidth

➢ Using the right system configuration for perf modeling is important.

❖ Modern platforms provide many configurable knobs in HW and SW. Production applications can and do use them to further improve efficiency (E.g different turbo modes or different page sizes)

➢ Learning about and using the right HW/SW setup per application helps improve fidelity of results



SoftSKU paper (ISCA 2020) and Twine Paper (OSDI 2020) discuss flexible system configuration

Direction #6 – Appropriate use of metrics

01

Performance

As measured by rack level perf that can be achieved while meeting all SLA requirements.

Rack performance at fleet level is derived from blending performance of large workloads.

02

Power Efficiency

Power here is the budgeted power that a rack consumes @ peak load. This includes servers, sleds, network switches, power supply and cooling.

Power efficiency is Rack Performance normalized by budgeted power.

03

TCO Efficiency

Cost here is the total expense to keep the rack alive in the datacenter. This includes CAPEX (cost of all components) + OPEX (Network provisioning, DC space, technicians, electricity and spares)

TCO efficiency is Rack performance normalized by Rack TCO.

Priority among metrics depends on many external factors and can change over time

Questions?

THANK YOU FOR YOUR TIME