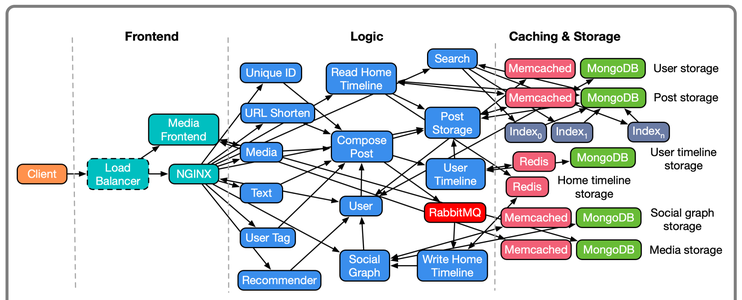
Autothrottle: A Bi-Level Approach to Microservice Resource Management

Francis Y. Yan, Mike Chieh-Jan Liang

USENIX NSDI 2024 Outstanding Paper

1. Background: Microservices

- Cloud apps are shifting toward microservices



Example: Social Network (benchmark)

- Imposed SLO on the application latency
  - e.g., P99 end-to-end latency < 200 ms
- Traditionally, CPU is overprovisioned
  - ☑ satisfied SLO ☒ resource waste

2. Problem: Resource Management

How to minimize CPU allocation while meeting SLO?

Challenges

- complex and evolving service dependencies
- delayed end-to-end latency feedback

As a result, we find it *impractical* to

- maintain an up-to-date view of dependencies
- reliably predict the impact of resource changes

3. Insight: Bi-Level Control

Microservices exhibit *distinct levels* of behavior:

- end-to-end application latency
- per-service resource usage

Our methodology:

1. Decouple application-level latency/SLO feedback from service-level resource control
2. Bridge these two levels through (machine-learned) performance targets

CPU throttle ratios

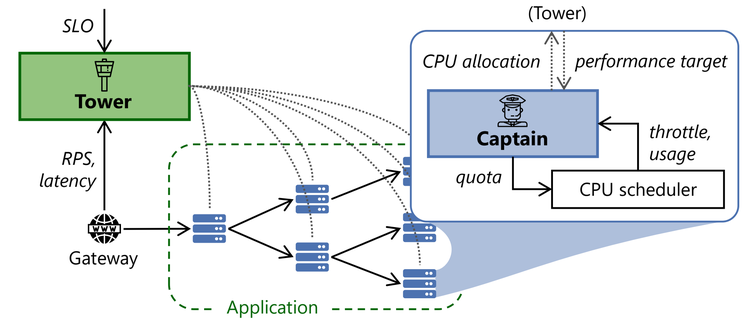
- we find they exhibit a high correlation with latencies

4. Solution: Autothrottle

Per-service closed-loop control to autonomously maintain a CPU throttle target

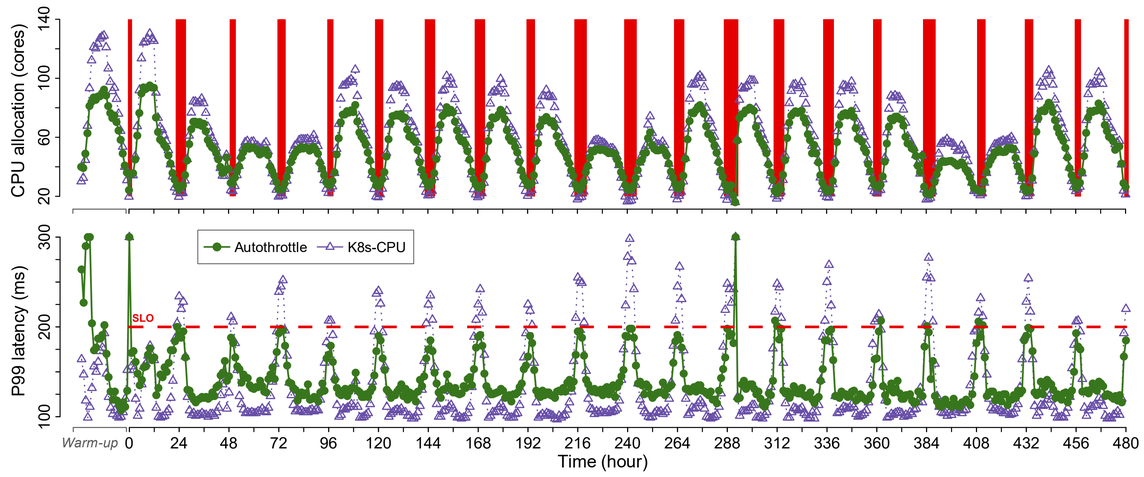
Application-wide online learning to periodically compute optimal CPU throttle targets

ML-assisted



5. Evaluation: Fewer SLO Violations, Lower CPU Allocations

Autothrottle reduces 90% SLO violations while saving 5.9 cores per hour



Experiment settings

App ▸ Social Network
Cluster ▸ 160 CPU cores
SLO ▸ 200 ms P99
Workload ▸ Bing traces
Baseline ▸ Kubernetes

Scan QR code for more details

Microsoft