THE UNIVERSITY of EDINBURGH
informatics

Reconciling Strict SLOs with Low Cost for User-Facing Services in the Cloud

DILINA DEHIGAMA, SHYAM JESALPURA, ZEYU XU, MARTON NEMETH
MARIOS KOGIAS*, BORIS GROT

* IMPERIAL

Today's online applications & deployment

Web & API services

E-Commerce

Media processing

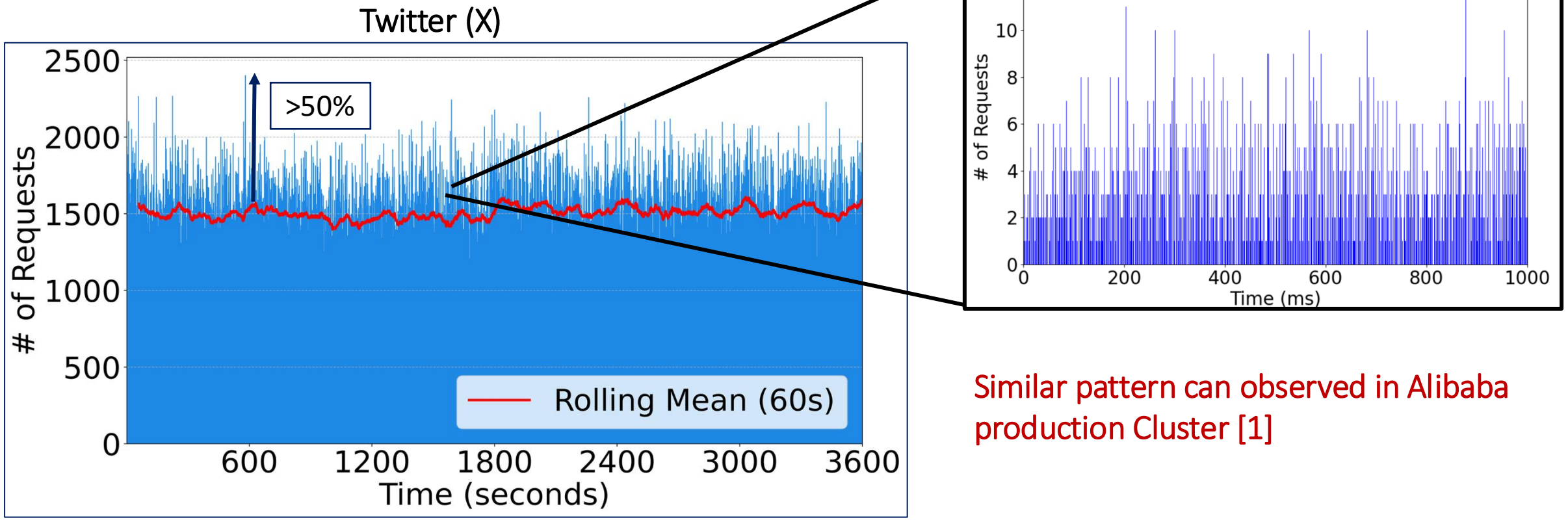
Data processing

User-facing → latency sensitive

Must meet strict latency SLOs (Service Level Objectives)

Deployed as containers on top of VMs

Bursty nature of load

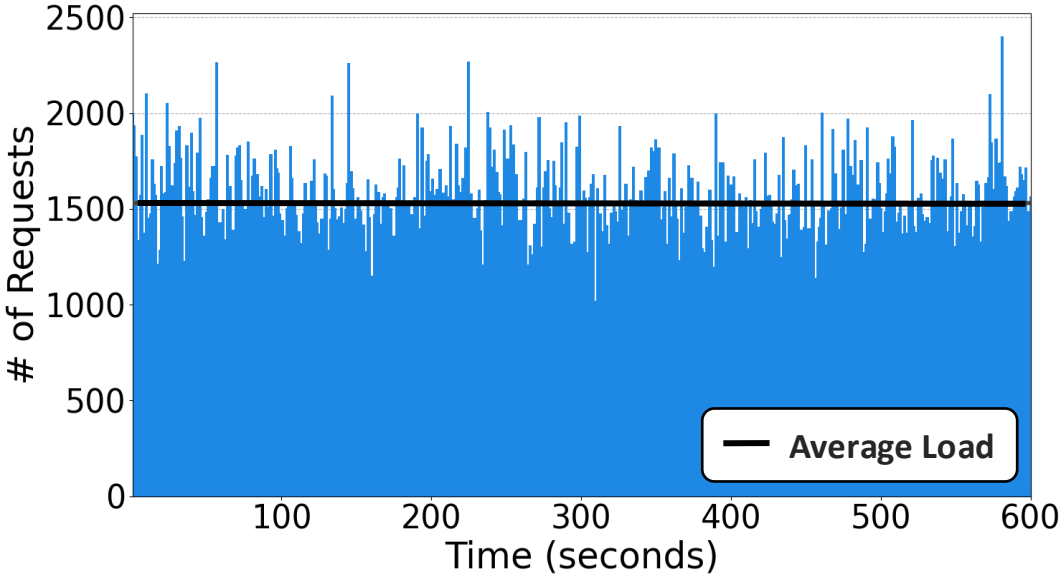


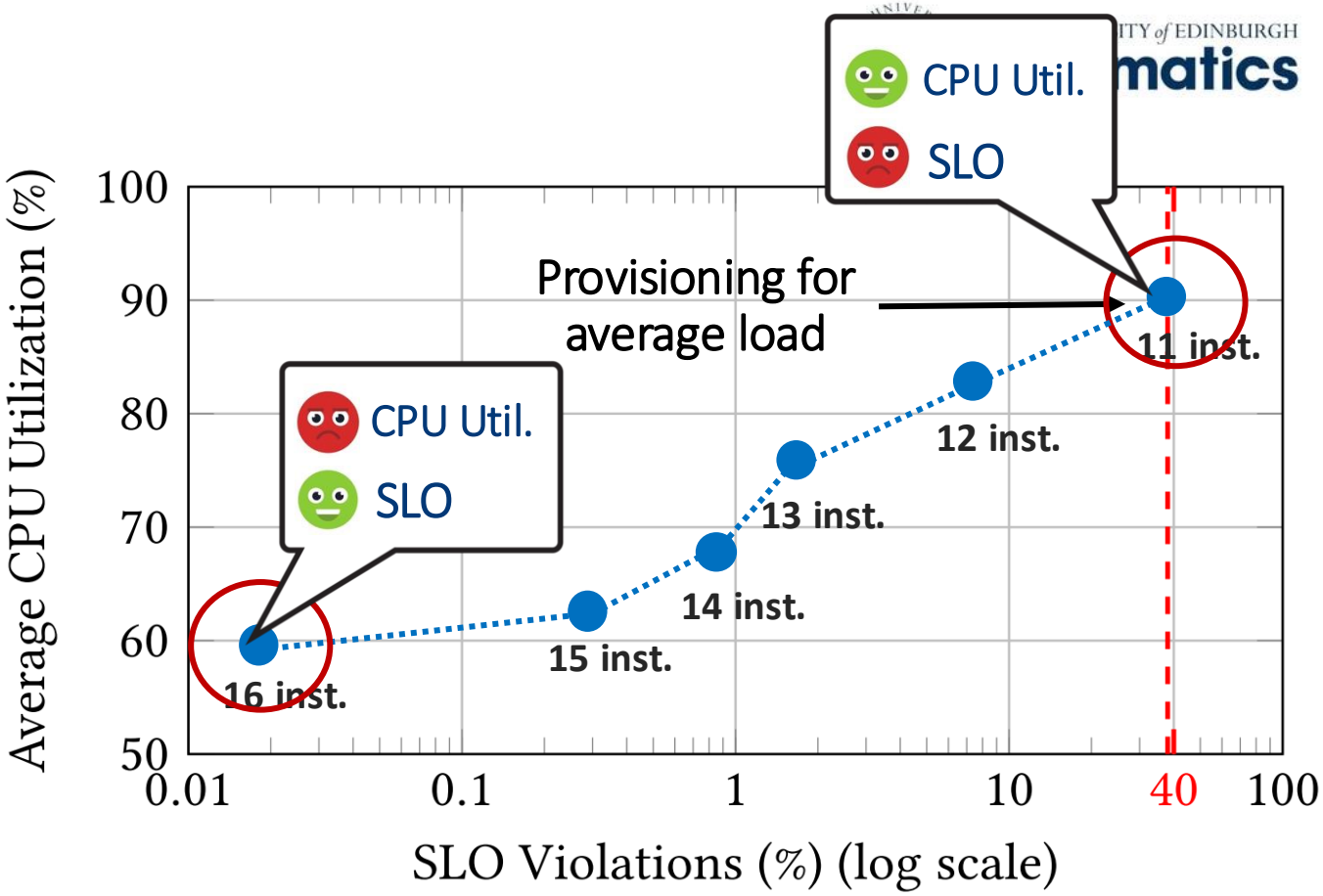
Similar pattern can observed in Alibaba production Cluster [1]

- Load is **relatively stable** at coarse grained intervals (60s rolling mean)
- Load fluctuates significantly at second-scale intervals (>50% deviation from the mean).
- Load is **highly volatile at sub-second** intervals (zoomed in right)

[1] The power of prediction: microservice auto scaling via workload learning [SoCC'22]

The provisioning trade-off





The provisioning dilemma:

Strong SLO compliance but low utilization

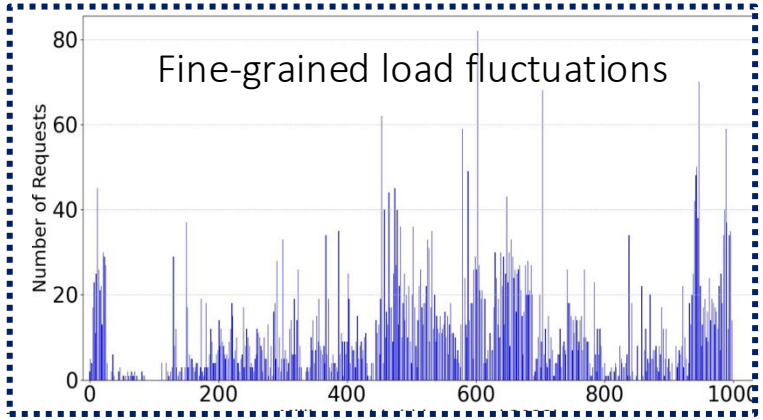
or

high utilization but poor SLO compliance

Dealing with fine-grained load fluctuations

VM Based Autoscaling

for



Reactive Auto scaling

Kubernetes Horizontal Pod Autoscaler (HPA)

- ✓ Adjusts capacity based on load changes
- ✗ Slow to respond to load changes
  - VM boot time (>30 seconds)
  - Built-in hysteresis to avoid frequent scaling

Proactive Auto scaling

Google's Autopilot , Alibaba's Madu

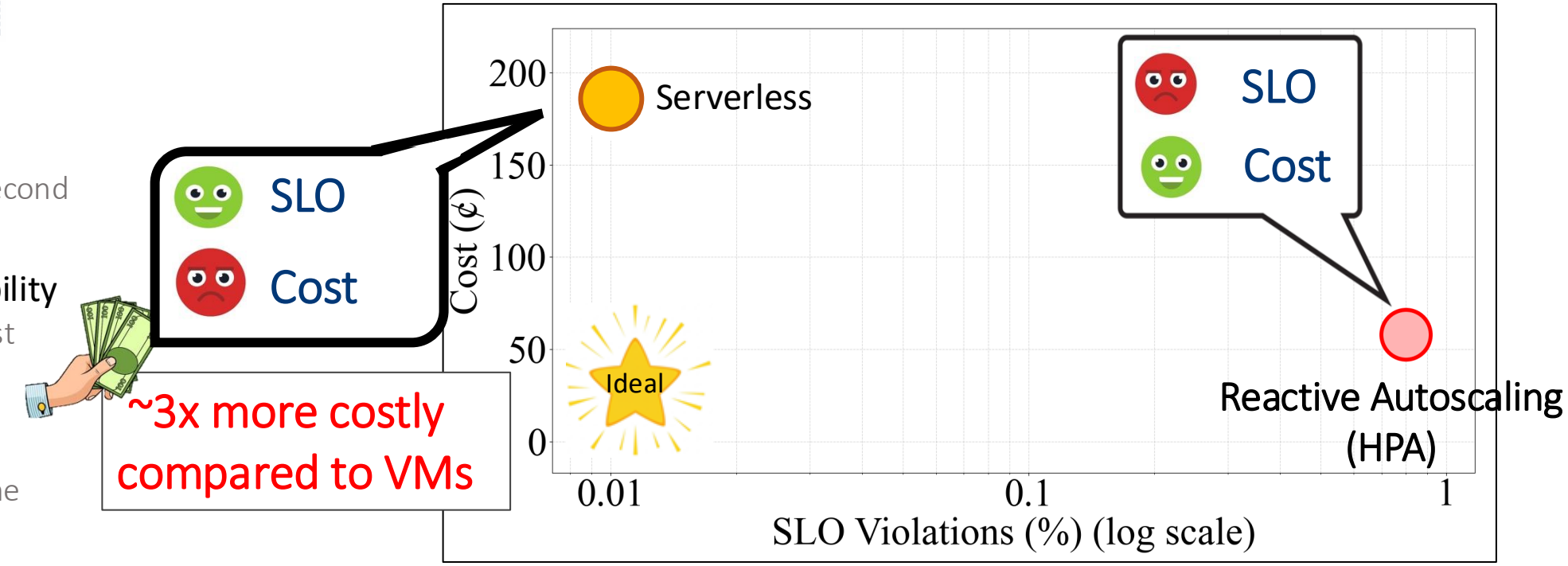
- ✓ Avoids detection delays & hides VM boot latency
- ✗ Works only for predictable load changes
  - Not effective under fine-grained load variability

A better alternative?

Serverless / Function-as-a-Service (FaaS)

Eg: AWS Lambda

- **Rapid elasticity**
  Starts well under a second
- **On-demand scalability**
  1 instance per request
- **Pay-per-use billing**
  No charge for idle time



Can we have the best of both worlds?

VMs + Serverless: A hybrid approach

Best of both worlds → Leverage **VM Cost-Efficiency** + **Serverless Elasticity**

Existing works

Spock, Mark , Feat

Use serverless only during VM scale-out

- ✓ Handle prolonged load spikes
- ✗ Ignores fine-grained fluctuations

Libra

Continuously offload to serverless

- ✓ Fixed fraction of requests offloaded to serverless determined at a 1-second interval → >1500 Requests
- ✗ Can't adapt to microbursts within the interval

Existing systems fail to unlock hybrid's full potential

Insights

THE UNIVERSITY of EDINBURGH informatics

Insight 1: Decisions must be per-request

- Coarse-grained request assignment leads to low VM utilization (costly!) or SLO violations

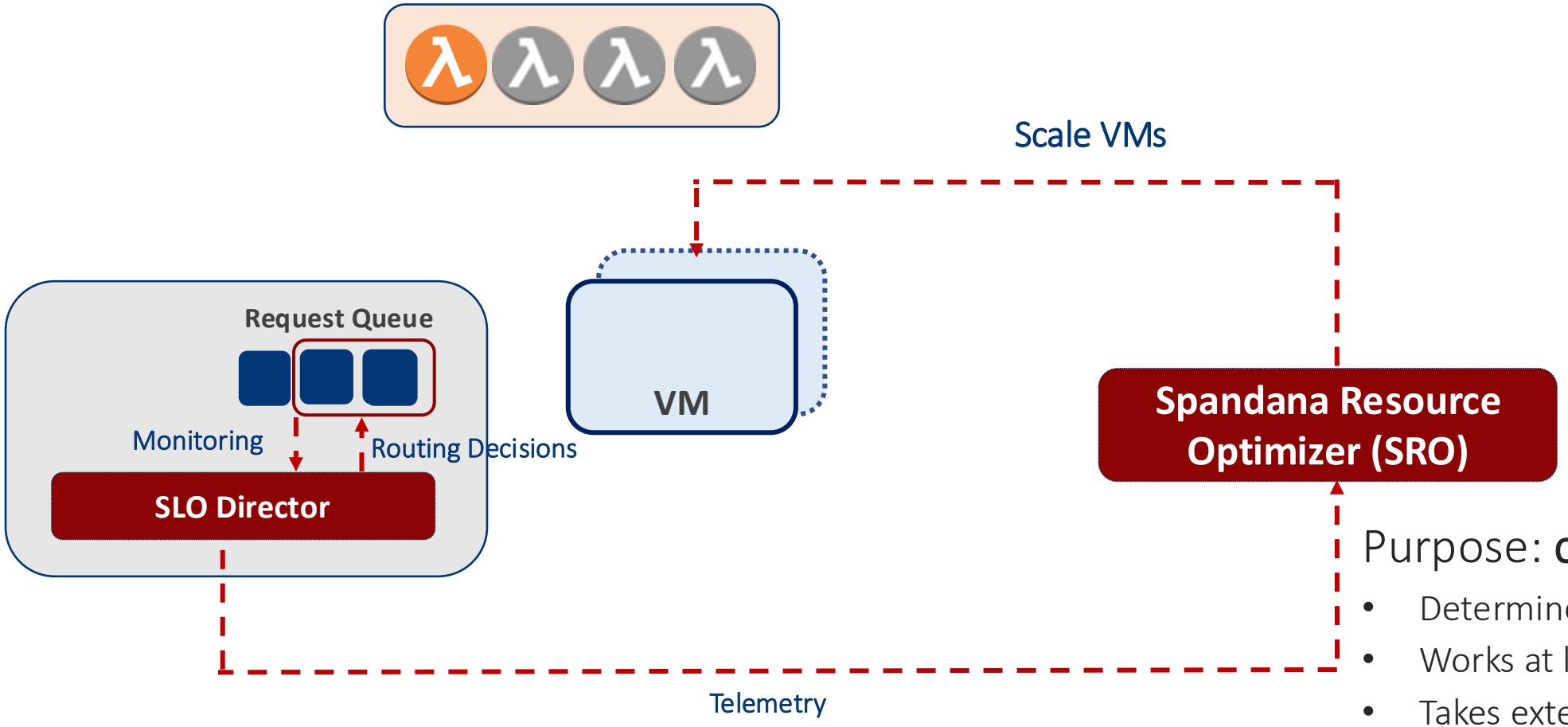
Insight 2: Queueing is not all bad

- A busy VM does not automatically imply an SLO violation
- Offload to serverless *only if* (queueing time + service time) > SLO target

Insight 3: Provisioning must be fluctuation-aware

- Average load metrics are insufficient
- VM allocation must explicitly account for load variance for cost efficiency

High-level architecture of Spandana



Purpose: **cost optimization**

- Determines cost-optimal # VMs
- Works at longer intervals (minutes)
- Takes extent of fluctuations into account

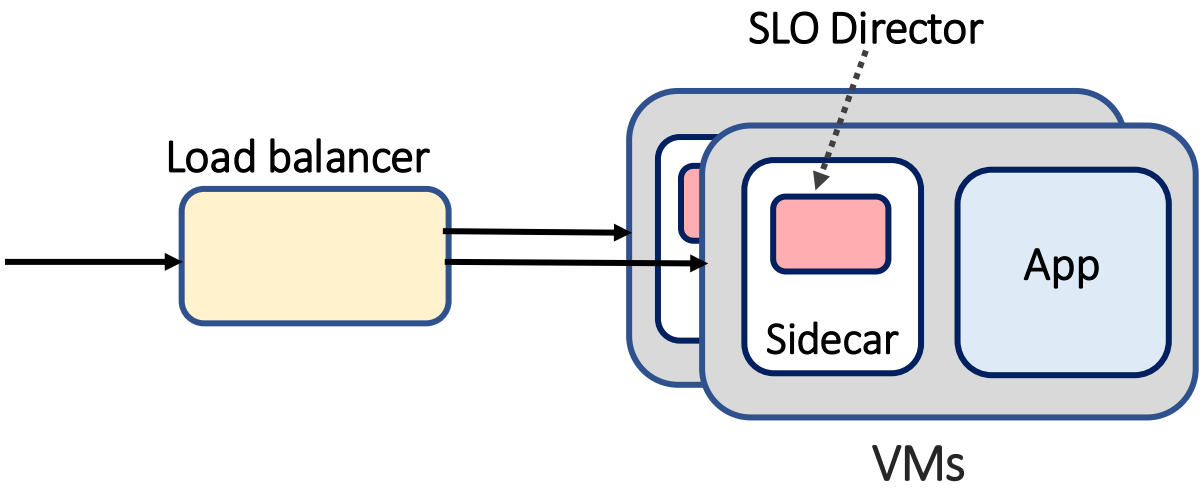
Spandana SLO Director

Purpose: **SLO enforcement**

- Real-time **per-request** routing decisions
- Maintains a bounded FIFO queue of requests
- Works with a **centralized** or **decentralized** setup

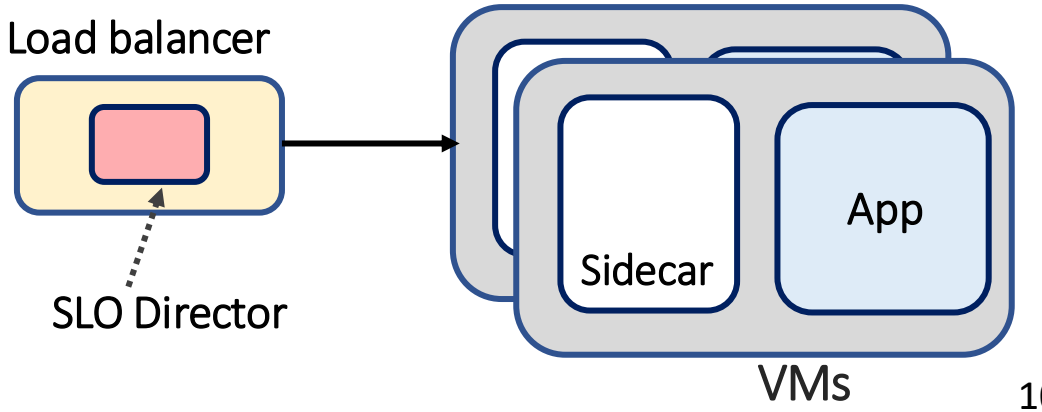
Decentralized Implementation

Co-located with the application instance



Centralized Implementation

Co-located with a centralized load balancer



Experimental setup

THE UNIVERSITY of EDINBURGH informatics

- **Kubernetes Cluster** - AWS EKS
- **Serverless Platform** - AWS Lambda (no pre-warmup)
- **Studied Applications** -
  1. Ratings service
  2. Details service

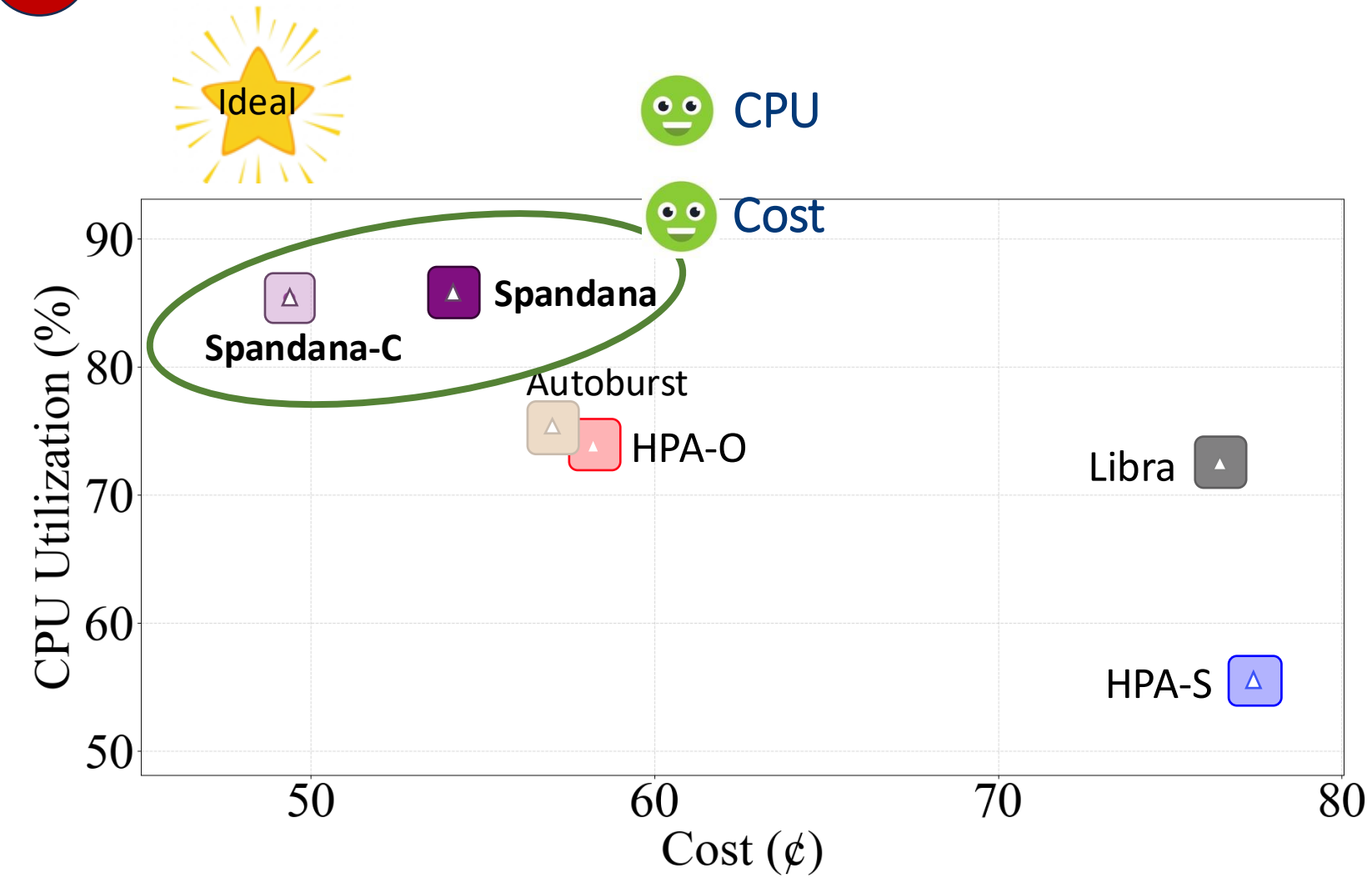
  I/O Bound *(BookInfo)*
  3. Img. Proc
  4. Compression

  CPU Bound *(SeBS)*

Evaluated systems

1. **Kubernetes HPA**
   - Standard (HPA-S)
   - Oracle (HPA-O)
2. **AutoBurst**
   - SOTA for hybrid VMs: standard + burstable VMs
3. **Libra**
   - SOTA for VMs + Serverless
4. **Spandana**
   - Spandana (Distributed)
   - Spandana-C (Centralized)

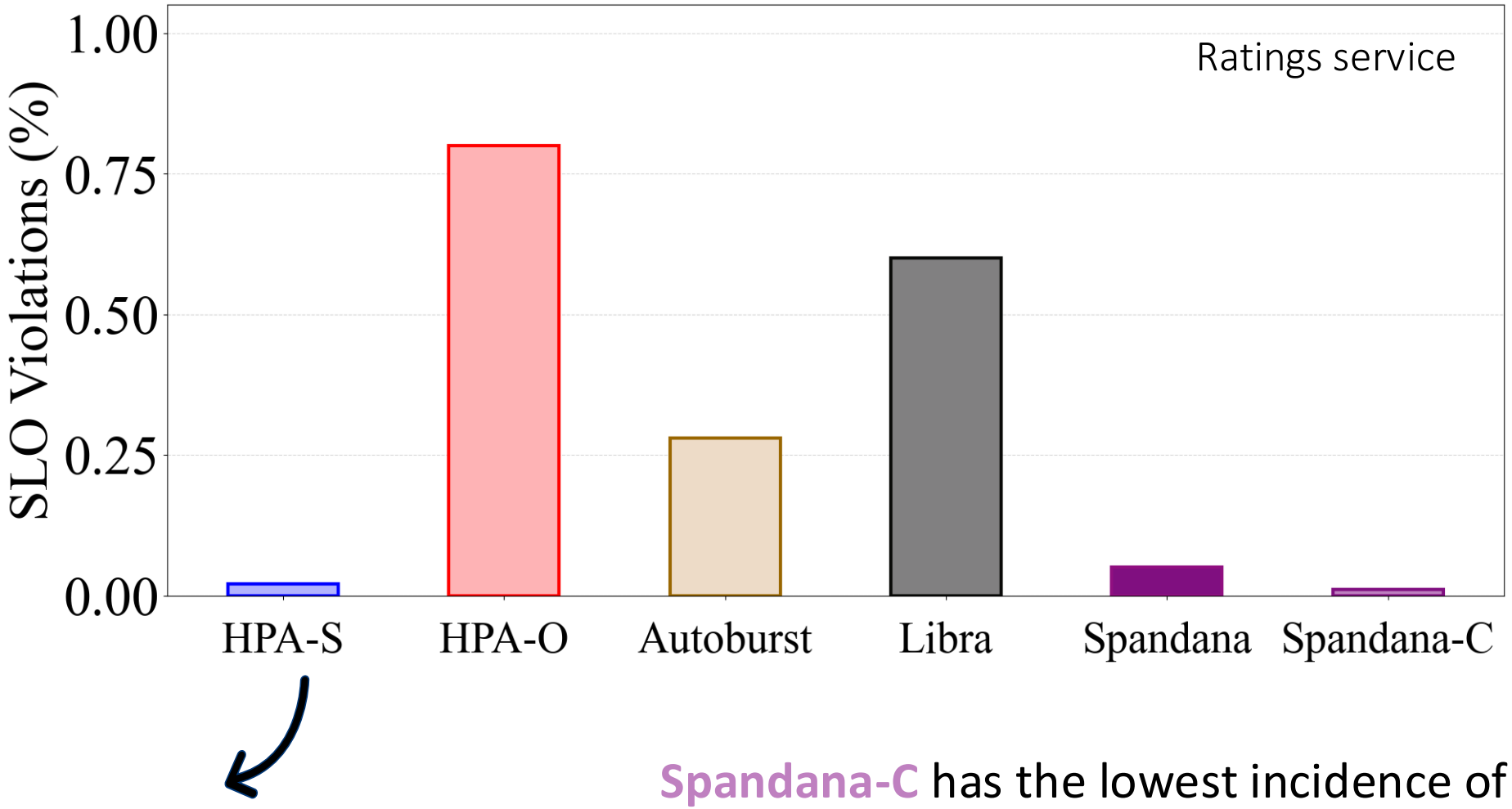
1 Main results : Cost & CPU Utilization



Spandana variants achieve the highest CPU utilization (**>85%**) among all systems

Spandana variants are the most cost-effective solutions

2 Main Results : SLO Violations (%)

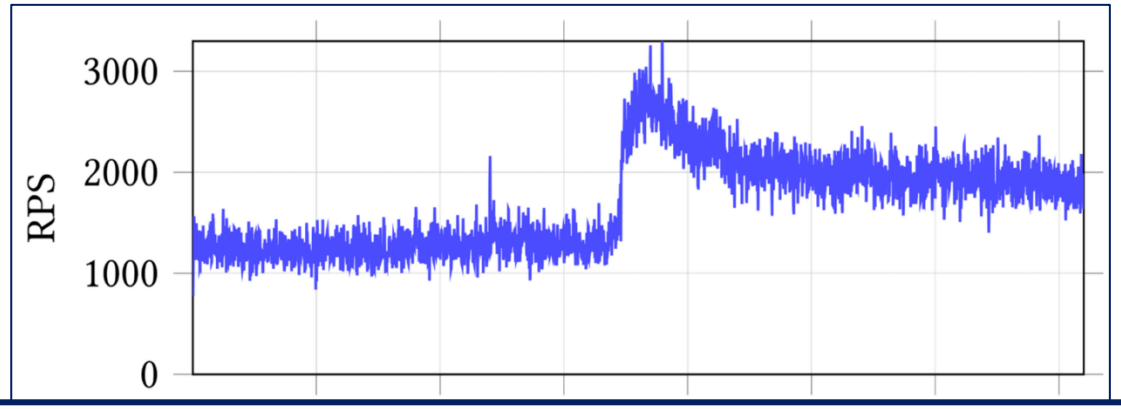


44% more costly than **Spandana**

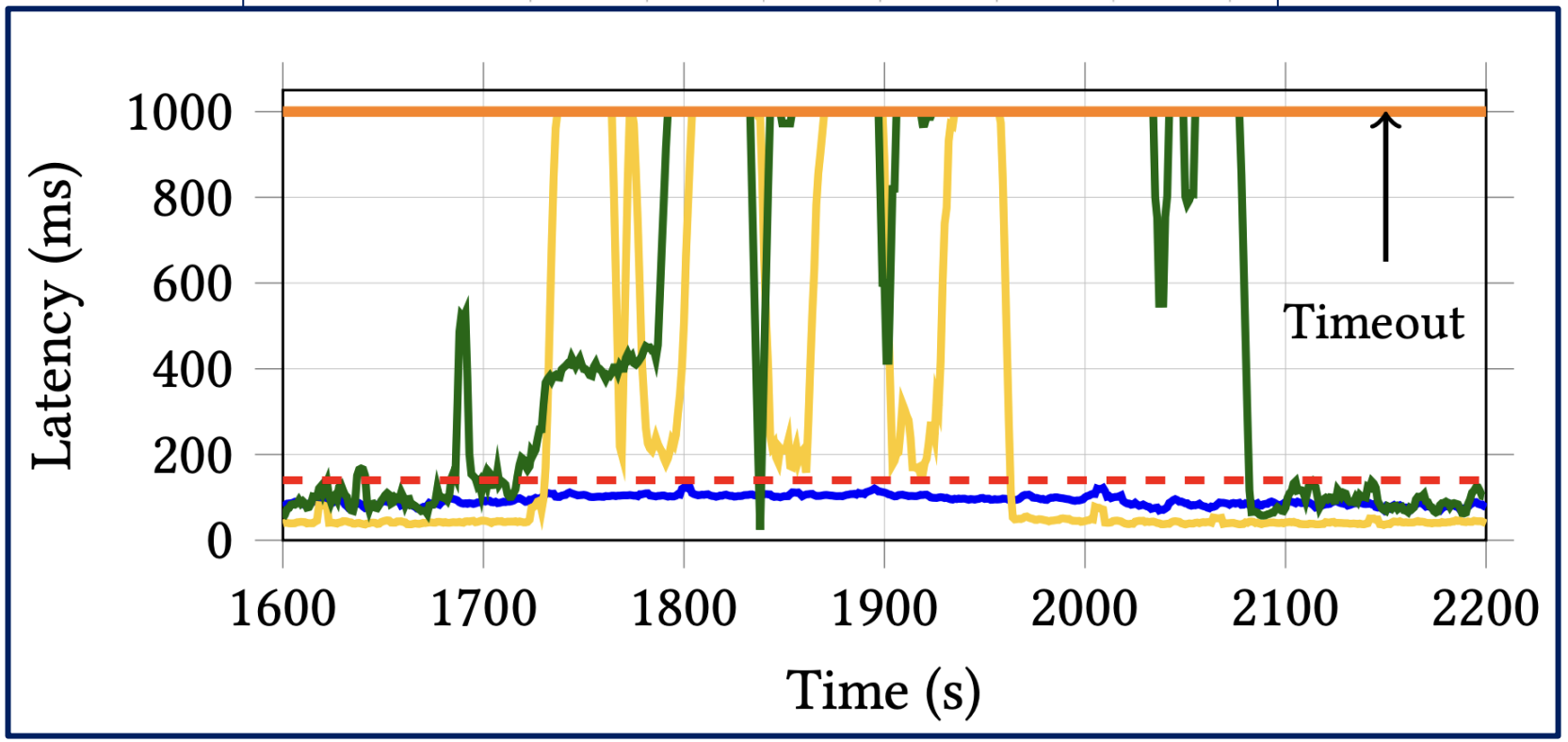
Spandana-C has the lowest incidence of SLO violations

Spandana is comparable to HPA-S on SLO but is less costly

3 Handling Load Spikes : P99 Latency



Twitter trace with a sudden **>2x** load spike.



Spandana never violates the strict SLO target

Thank you!

https://ease-lab.github.io