ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

CRISP: Critical Path Analysis of Large-Scale Microservice Architectures

Zhizhou (Chris) Zhang, and Timothy Sherwood

Murali Krishna Ramanathan, Prithvi Raj, Abhishek Parwal, and Milind Chabbi

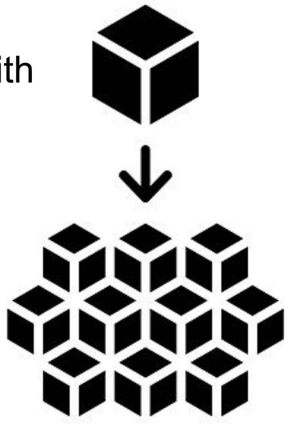
Department of Computer Science
University of California Santa Barbara

Uber

What is microservice architecture?

- Distributed system
- Independent business logic -> independent programs
- Communicate over well-defined APIs
- Loosely coupled
- Owned by small, self-contained team

Monolith



Microservice

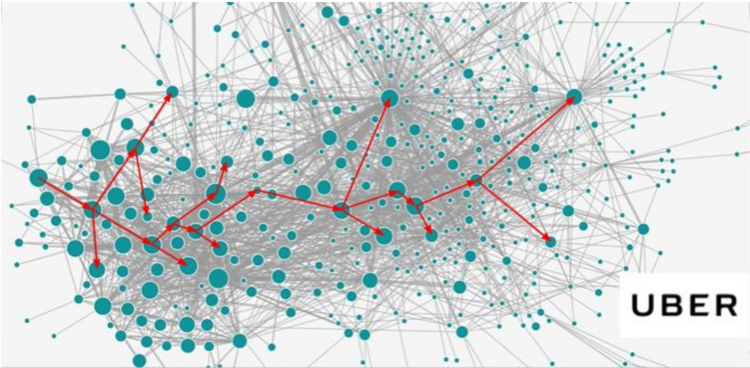
https://tinyurl.com/4zxbynt8

Why microservices?

- Scalable development
- Independent development
- Easier deployment
- 71% organizations adopted microservices in 2021

Microservice Challenges: Complexity

- Evolution of microservices often leads to complex interactions
- Extremely complicated to analyze
- Deeply nested
- Asynchronous
- Tens of thousands of endpoints interact with each other



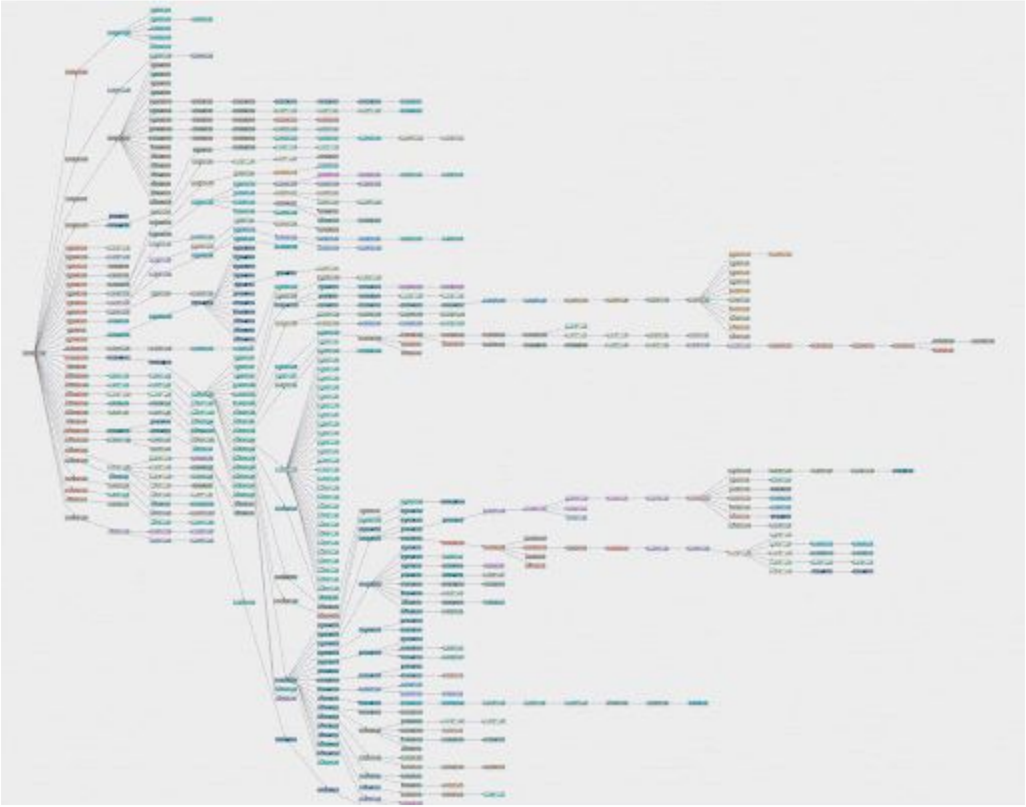
Distributed tracing

- Jaeger: System for capturing RPC caller-callee relationships among services
- Widely deployed at Uber
- Supports multiple languages: Go, Java, Python…
- Collect trace on sampling basis
- Retains in different storage systems
  - Cassandra, Elasticsearch, memory

https://www.jaegertracing.io/

How to **pinpoint** and **quantify** the root cause of end-to-end latency of a request?

Gives example visualize



Our solution

Critical Path Analysis (CPA) on distributed traces

It supports:

- Top-down: service owner debuggings and optimizations
- Bottom-up: systemic analysis and optimizations
- Anomaly detection: for building automatic alerting system

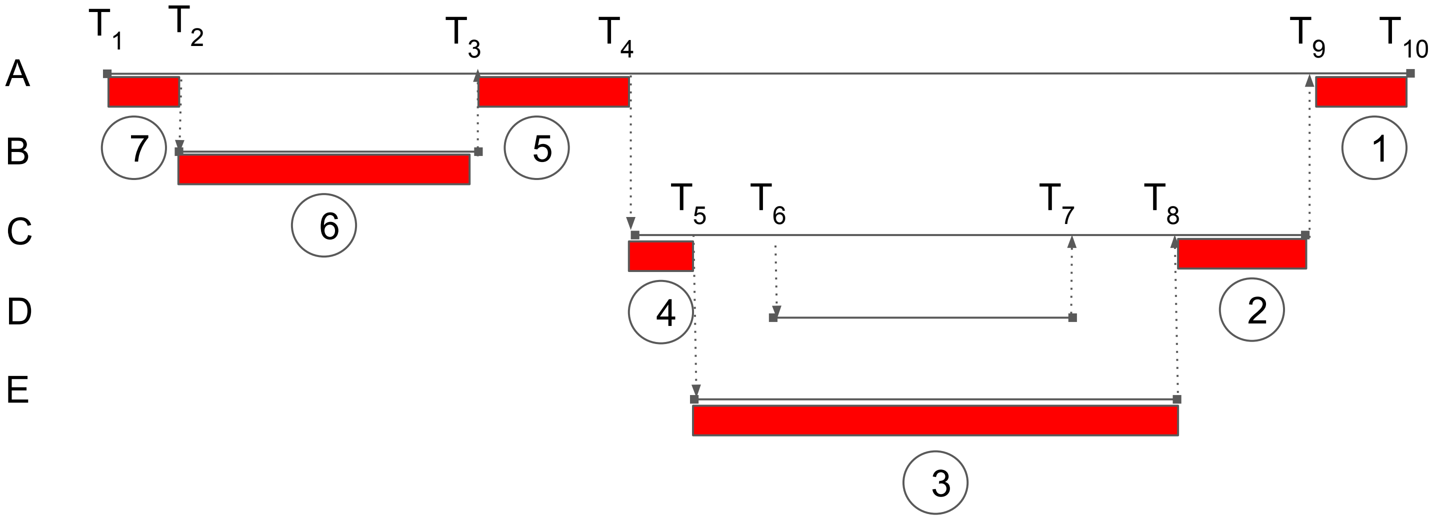
Outline

- Intro
- What is Critical Path Analysis
- Challenges applying CPA in real data center
- CRISP design
- Top-down analysis
- Bottom-up analysis
- Anomaly detection

Critical Path Analysis (CPA)

- Technique to identify longest stretch of dependent tasks
- End-to-end latency = length (CP)
- ↓length (CP) ⇒ ↓end-to-end latency
- Naturally simplifies the complex dependency graph from distributed tracing
- How to compute: iterate backwards and recursively

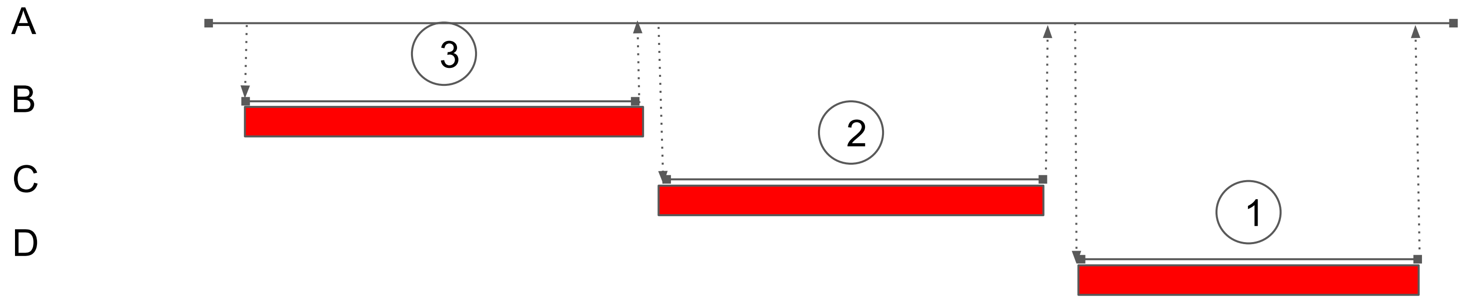
CPA example



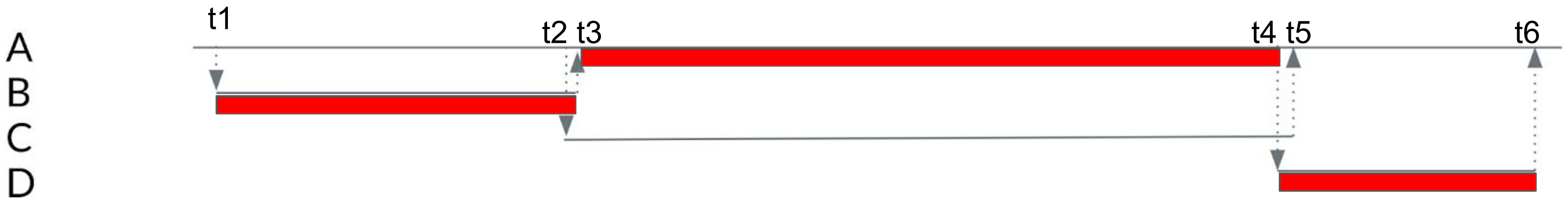
Challenges applying CPA on real-world traces

- “sync” (last arriver) are NOT designated events in Jaeger traces
  - “Sync” needs to be inferred via timestamp
- Machine clocks are not synchronized
- Missing spans

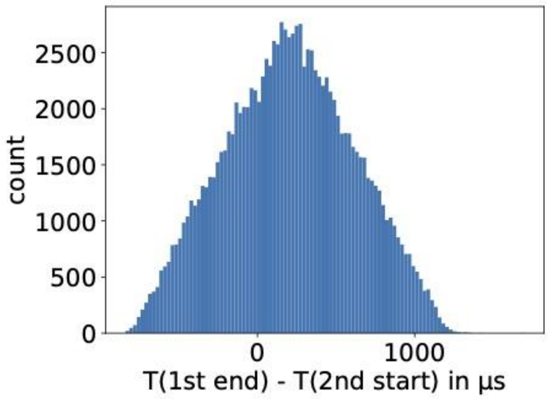
Critical Path on Perfect Traces



Critical Path on Real Traces

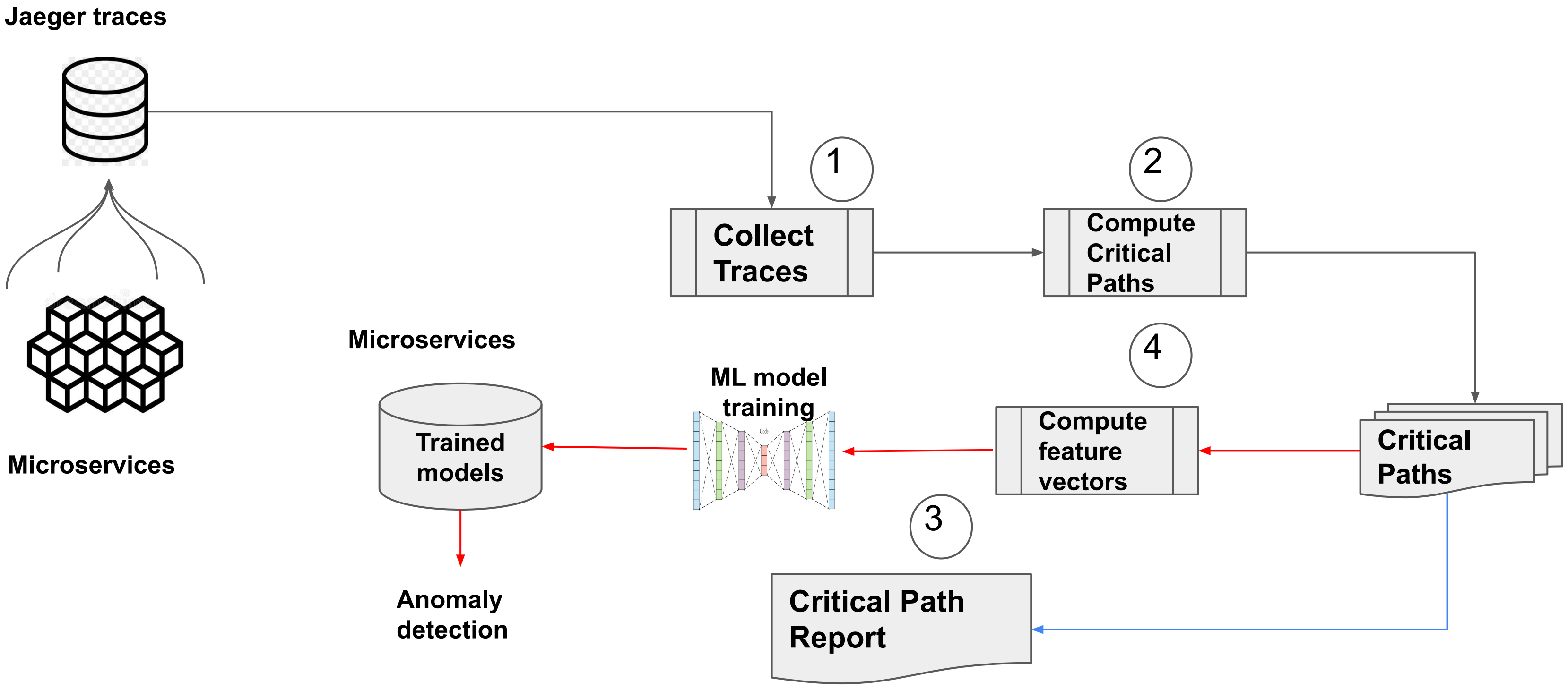


C is **<u>NOT</u>** on critical path!

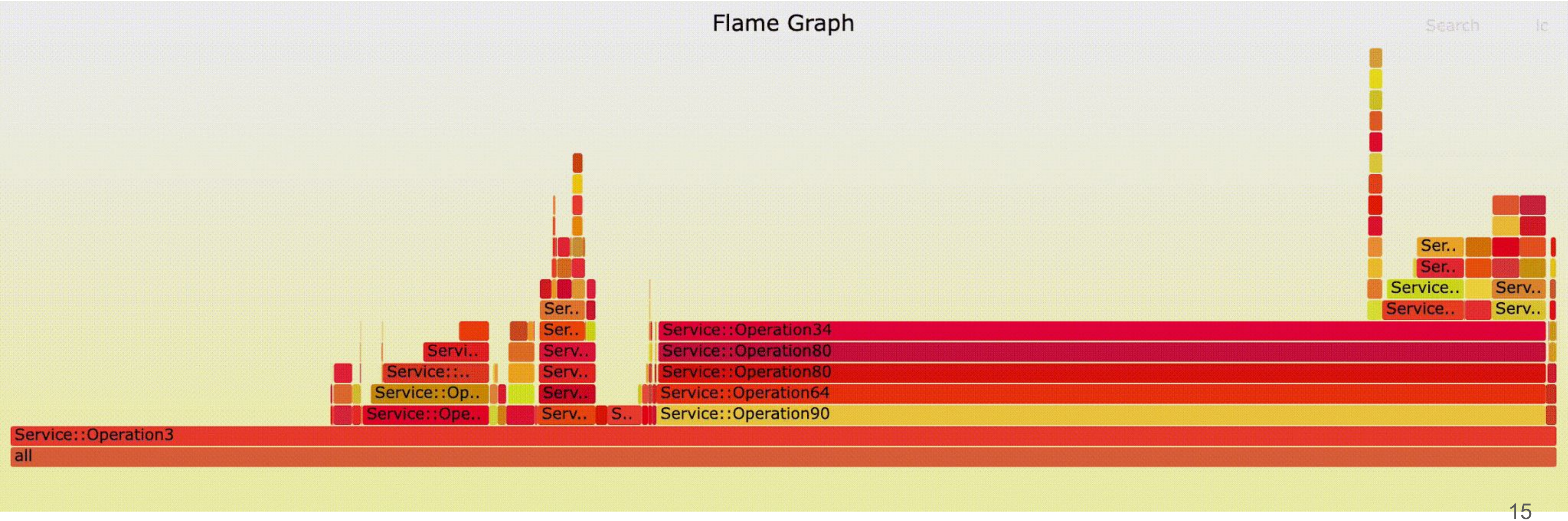


- Solution: allow some degree of overlap between child endpoints

Design of CRISP (Critical path and Span)



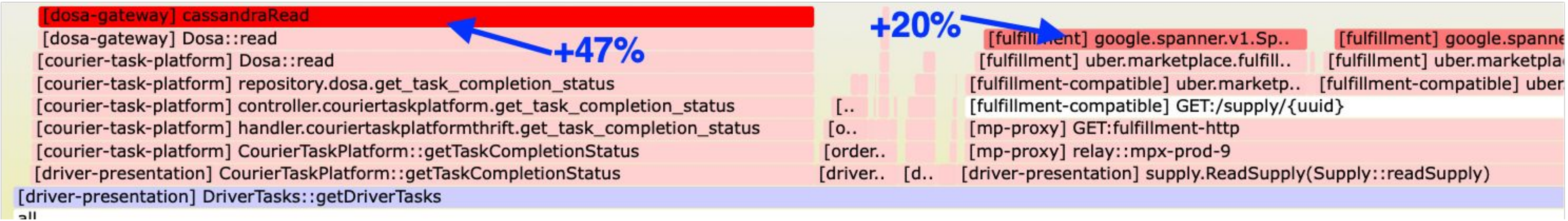
Top-Down Analysis



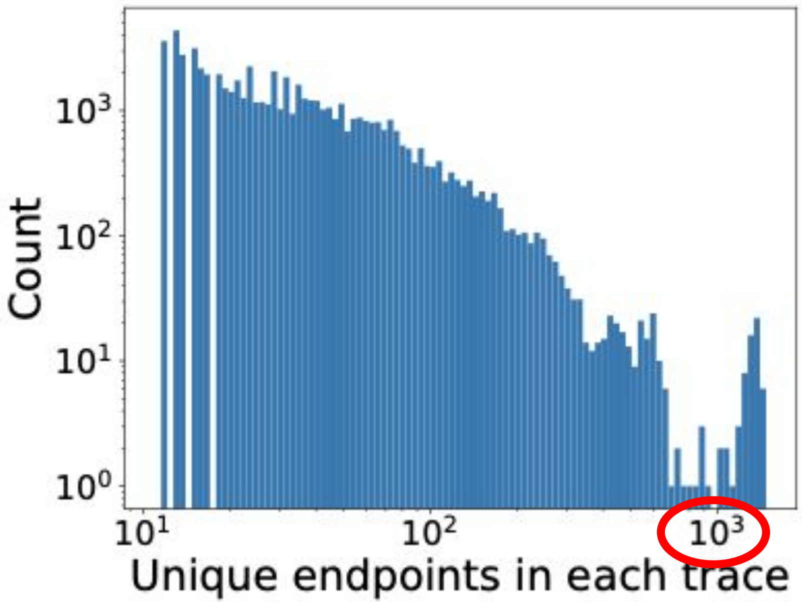
Differential Analysis

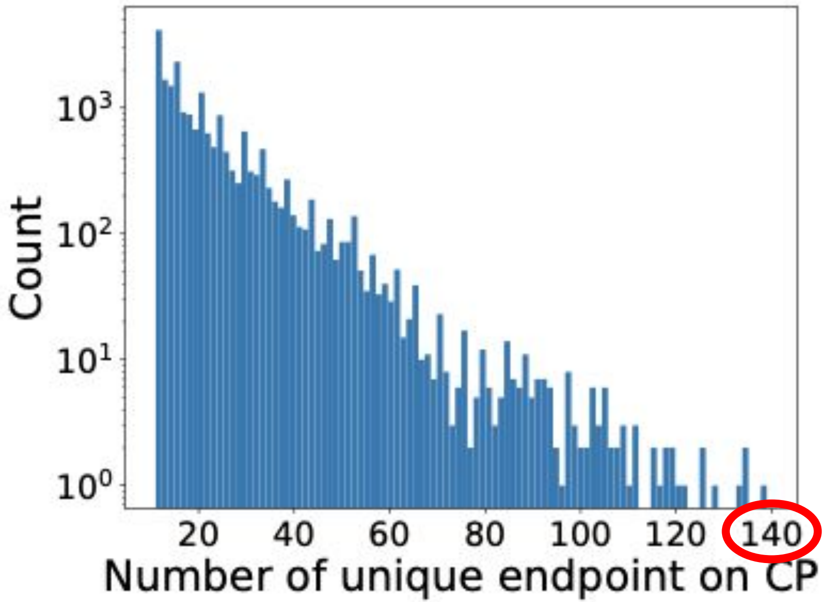
Root cause the tail latency by diffing P50 vs. P95

Recommend developer to cache the result instead of query database



Bottom-Up Analysis





Almost 10X difference!

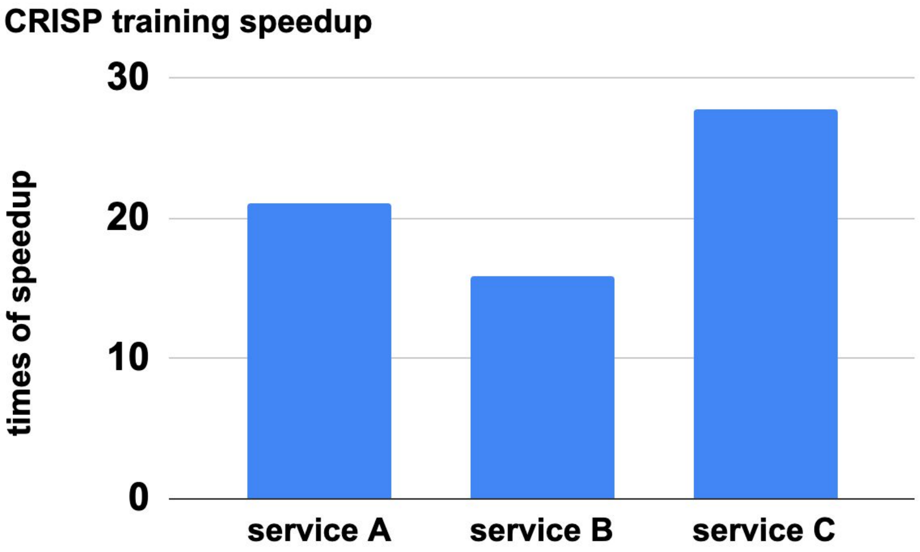
Anomaly detection

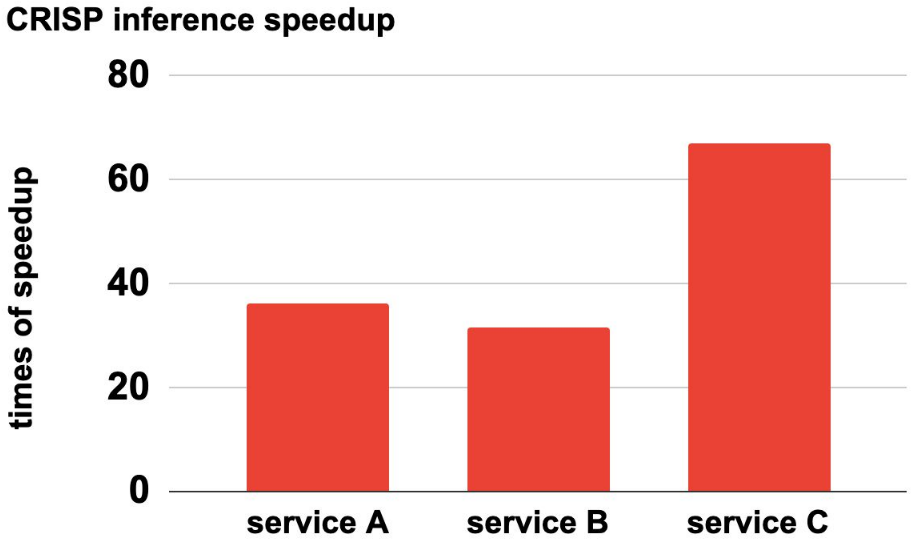
- Important to detect anomaly to debug
- Auto-encoder decoder (Liu et al. ISSRE 2020)
- Use critical path as the training data instead of full graph
- Run on numerous real important services from Uber
  - 200~1500 unique endpoints on each service
  - 1500~11000 spans in the trace

Recall Improvement

	recall	
	Liu et al. (SOTA)	CRISP
service 1	0.986	0.992
service 2	0.958	0.984
service 3	0.5	0.982
service 4	0.928	0.978
service 5	0.5	0.982
service 6	0.912	0.977

Training and Inferencing Speedup





Conclusion

- CRISP: critical path to analyze complex microservice traces
- Top-down for service-level insights
- Bottom-up for system-wide insights
- Anomaly detection to aid alerting systems

Available at: https://github.com/uber-research/CRISP

Contact: zhizhouzhang@ucsb.edu or milind@uber.com

Thanks!

Questions?