Seer: Leveraging Big Data to Navigate The Complexity of Cloud Debugging

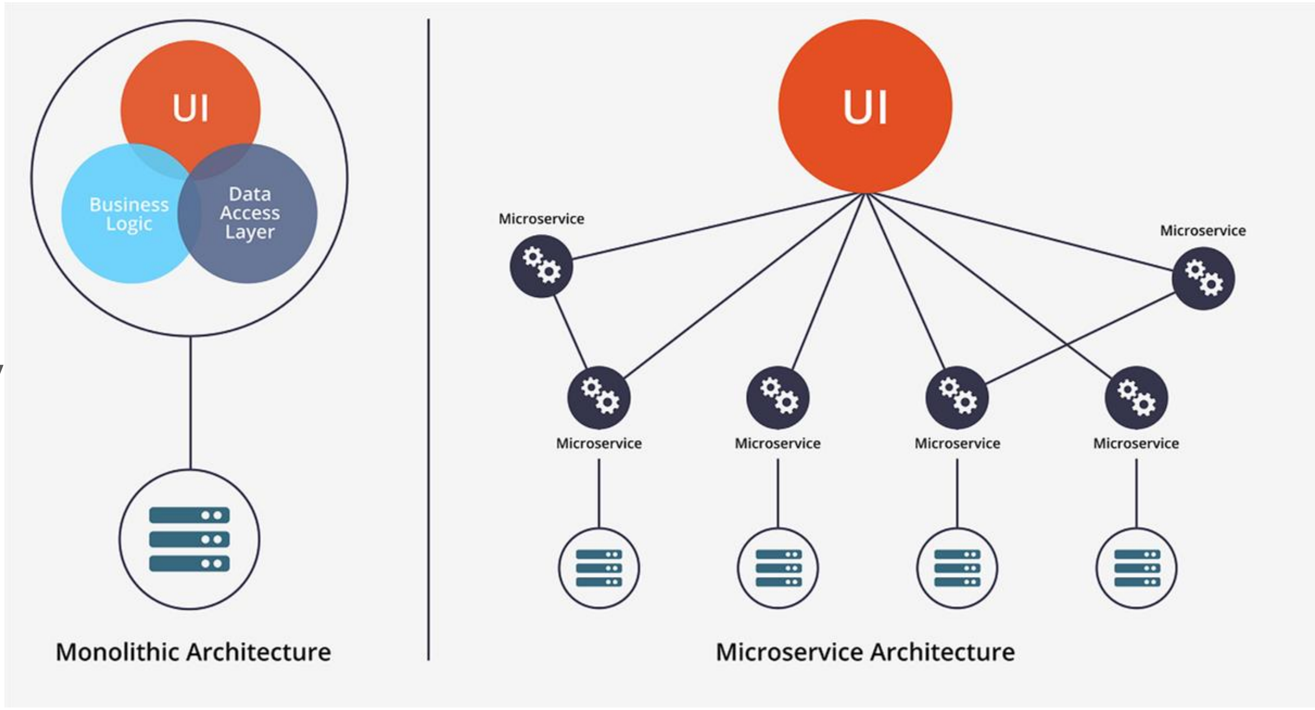
Yu Gan, Meghna Pancholi, Dailun Cheng, Siyuan Hu, Yuan He,
Christina Delimitrou

What is SEER?

A proactive performance debugging systems using machine learning to improve performance predictability of cloud systems hosting interacting micro services.

Microservices- better modularity, ease of development & deployment.

Each Microservice is easily deployed, rebuilt, redeployed and managed independently



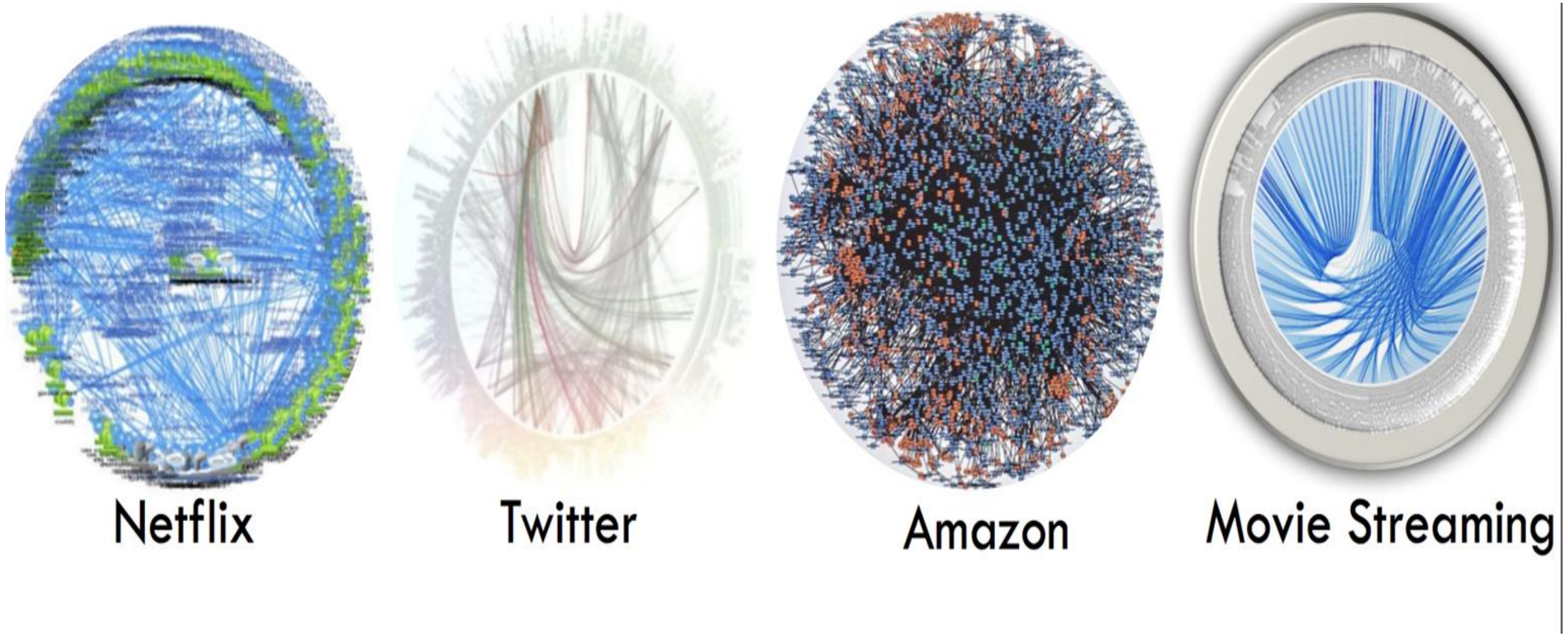
Goals

Cloud computing services are governed by strict quality of service (QoS) constraints in terms of throughput, and more critically tail latency

(i) can QoS violations be anticipated in cloud systems that host microservices-based application

(ii) can we pinpoint which microservice is the culprit of an upcoming QoS violation early enough to take corrective action?

Microservices graphs



Tracing

- RPC Level Tracing using thrift
- Timestamp start-end for each microservice
- No sampling
- Overhead: <0.1% in throughput and <0.2% in tail latency

Why Neural Networks?

- Recognize queueing patterns between microservices that result in QoS violations
- Neural networks have shown good results in pattern recognition with massive datasets
- Conditions resulting in QoS violations are not known or easy to annotate
- Assumes no prior knowledge about dependencies between individual microservices

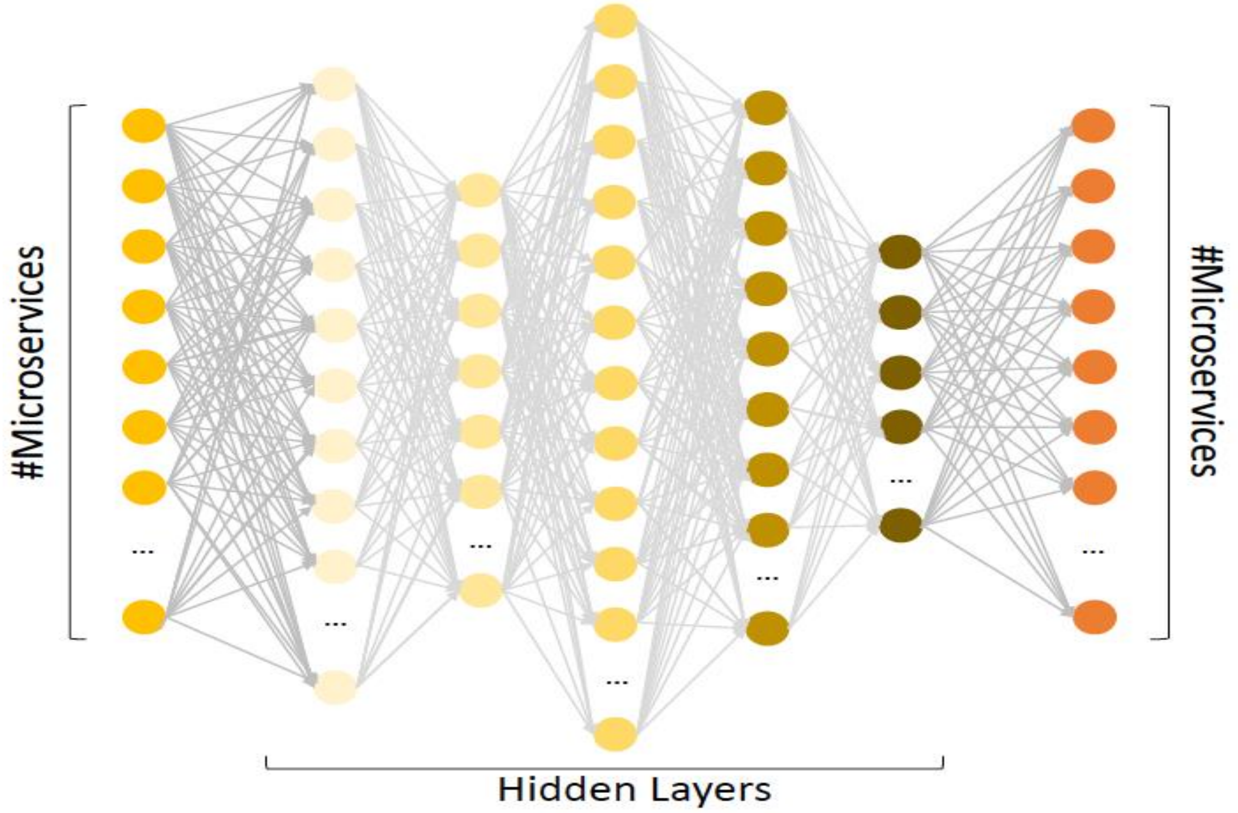
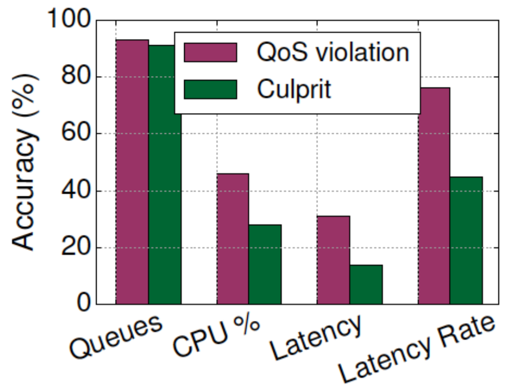
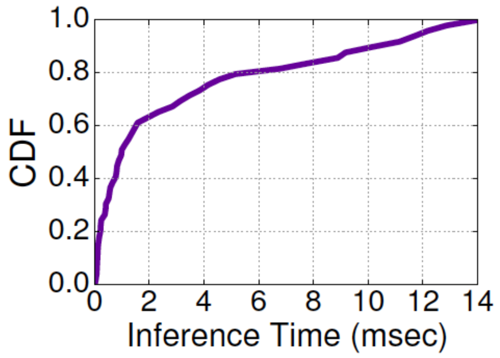
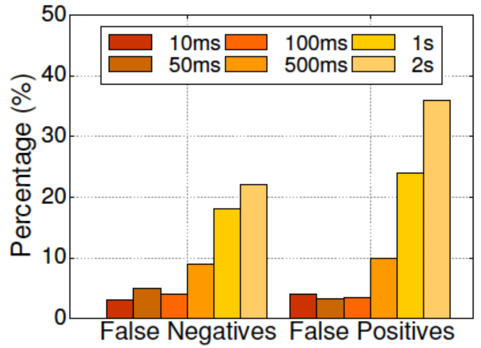


Figure 1: The neural network design in Seer.

Metrics and Performances



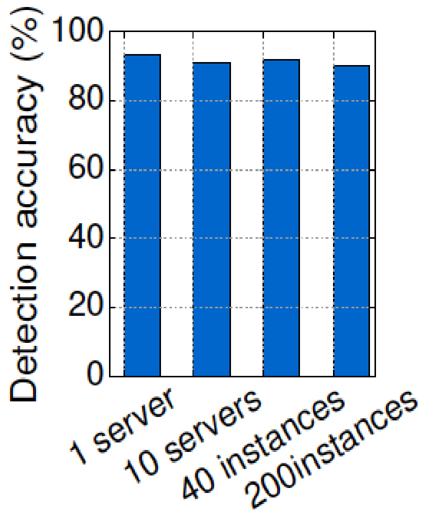


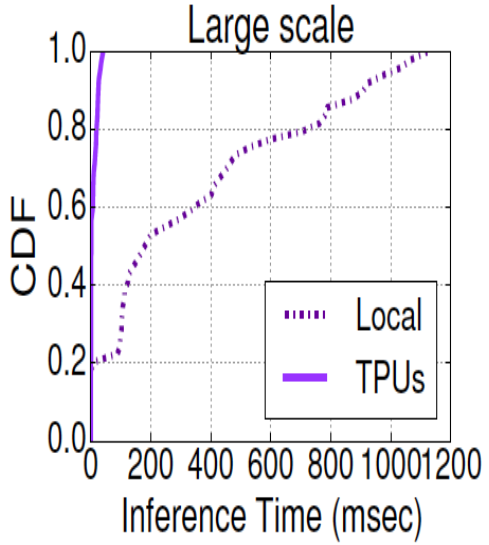


Methodology

3 end-to-end applications using popular open-source microservices

- 30-40 microservices per app
- Social Network
- Movie Reviewing
- E-commerce





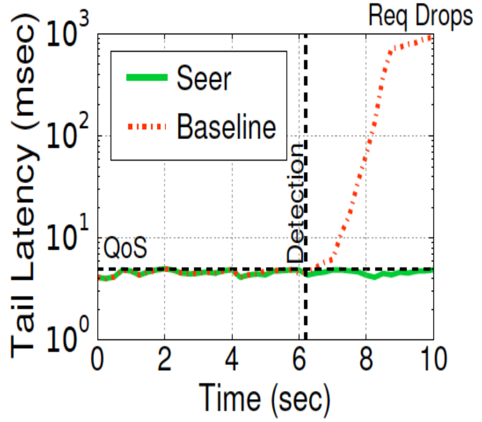


Figure 4: Tail latency with and without Seer.

QoS Violation Prevention

- Resizing Docker container
- Uses hardware performance counters for detecting problematic resources. But public clouds don’t provide access
- A set of contentious microbenchmarks, each targeting a different system resource to pinpoint problematic resources
  - For example, a cache thrashing microbenchmark for cache saturation,or a network bandwidth-demanding microbenchmark will reveal insufficient bandwidth allocations

Future work

- Security Concerns
  - Sensitive data of different resources can be leaked
- Increase in cost
  - Storage for trace data, TPUs
  - Evaluate trade-off and determine if it is really effective
- QoS violation prevention
  - Errors in problematic resources detection can cost high
  - Without accurate prevention, will detection be helpful?

Questions?