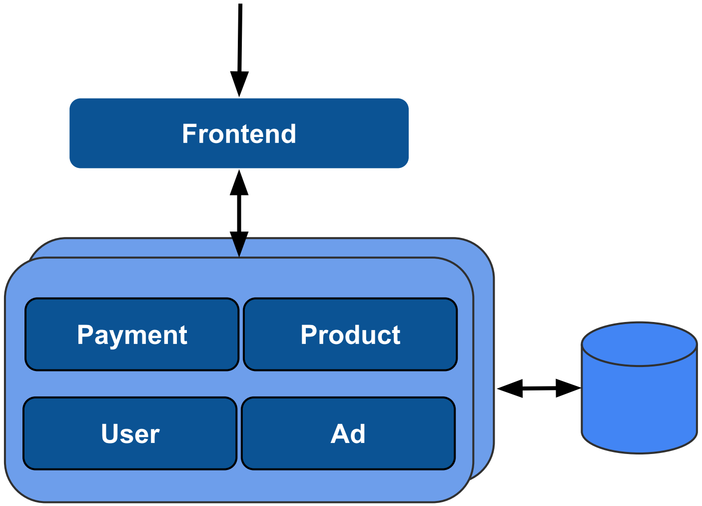
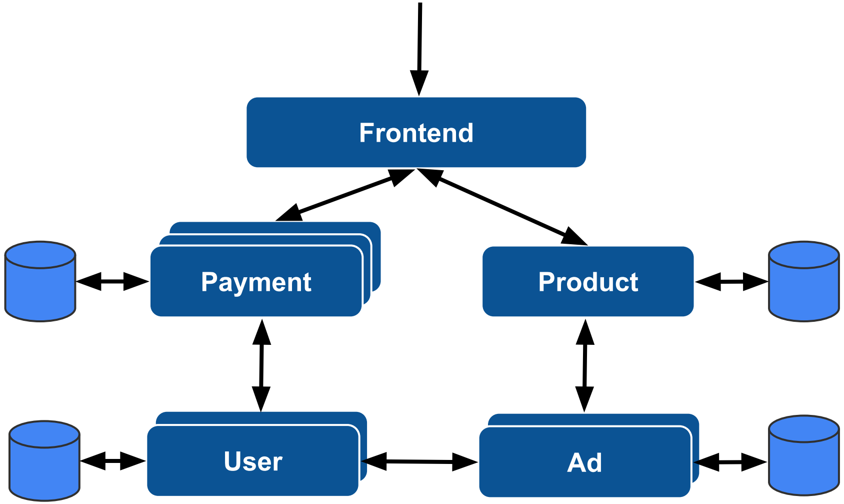
Dissecting Overheads of Service Mesh Sidecars

Xiangfeng Zhu, Guozhen She, Bowen Xue, Yu Zhang, Yongsu Zhang, Xuan Kelvin Zou, XiongChun Duan, Peng He, Arvind Krishnamurthy, Matthew Lentz, Danyang Zhuo, Ratul Mahajan

From Monolith to Microservices

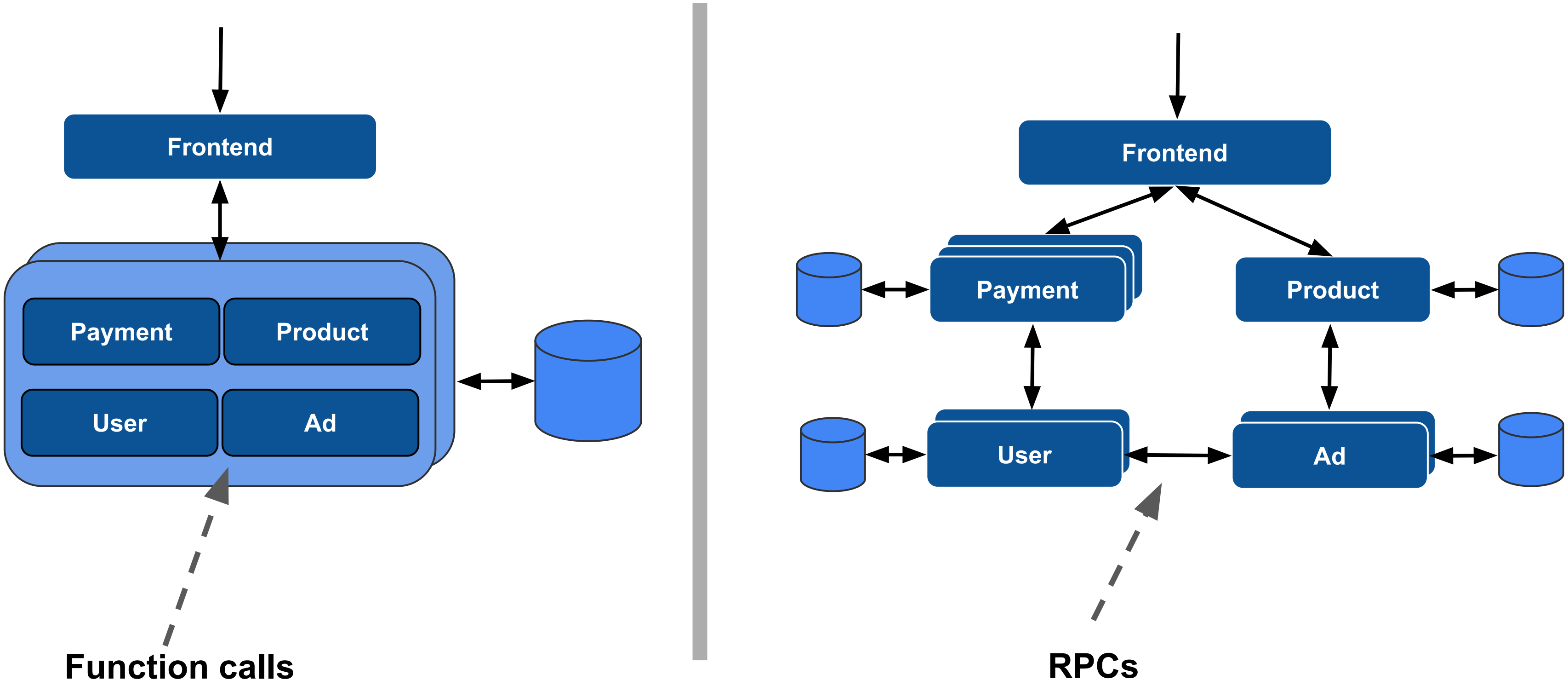


Monolithic Application



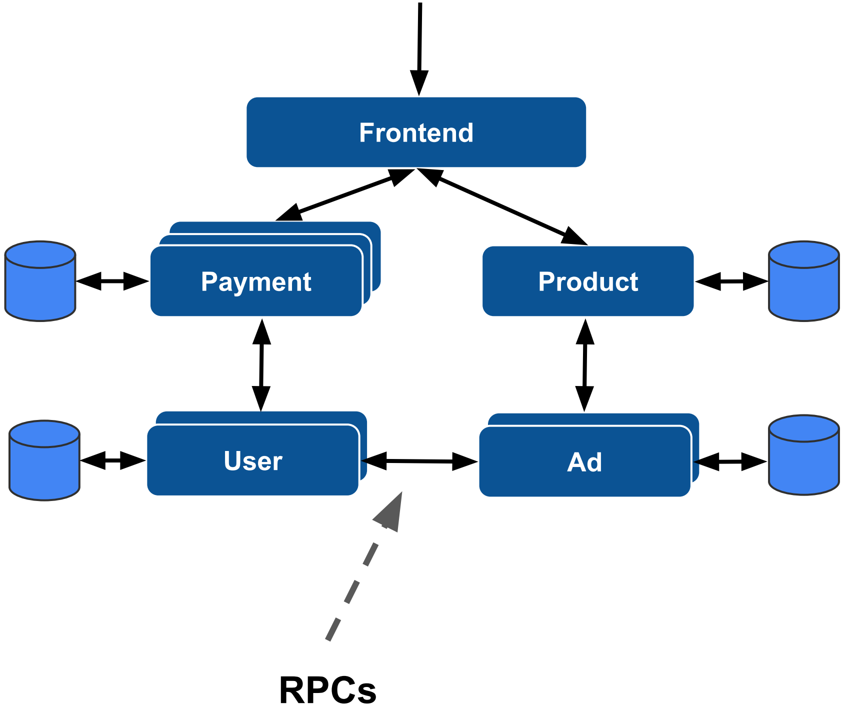
Microservices

From Monolith to Microservices



Microservices need rich message processing

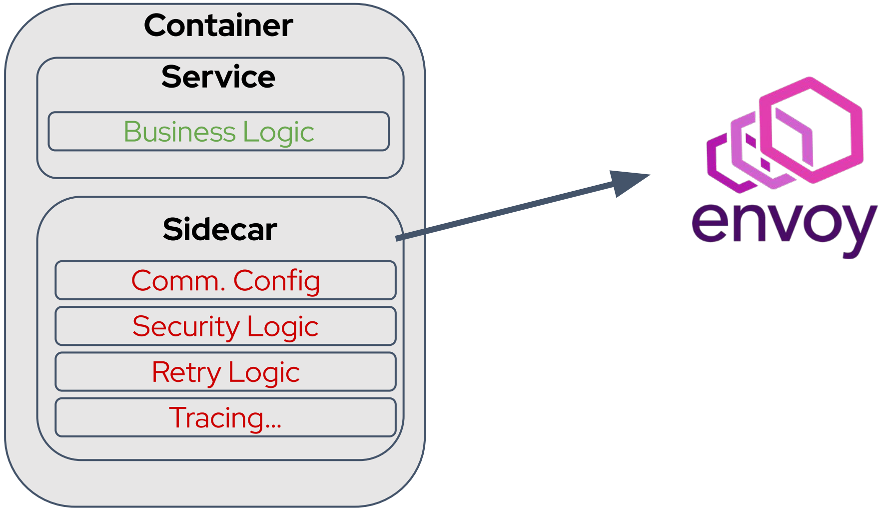
- Service discovery
- Access control
- Load balancing
- Observability
- Encryption
- Fault tolerance



Solution: Service Mesh with Sidecar Pattern

90% of the organizations either using or evaluating service mesh

- From a 2022 CNCF Survey[1]



[1]Service meshes are on the rise - CNCF '22

Performance Overhead vs. Functionality

"We are frequently asked how fast is Envoy? ... The answer is: it depends. Performance depends a great deal on which Envoy features are being used and the environment in which Envoy is run...."

- From Envoy FAQ

- Balancing performance and functionality
  - What is the latency and CPU usage impact with a given configuration
- Evaluating the impact of optimizations
  - What are the primary sources of overhead of a given configuration?
  - What if I use an alternative the IPC mechanism between application and sidecars?

Ad-hoc Configuration Tuning is inefficient

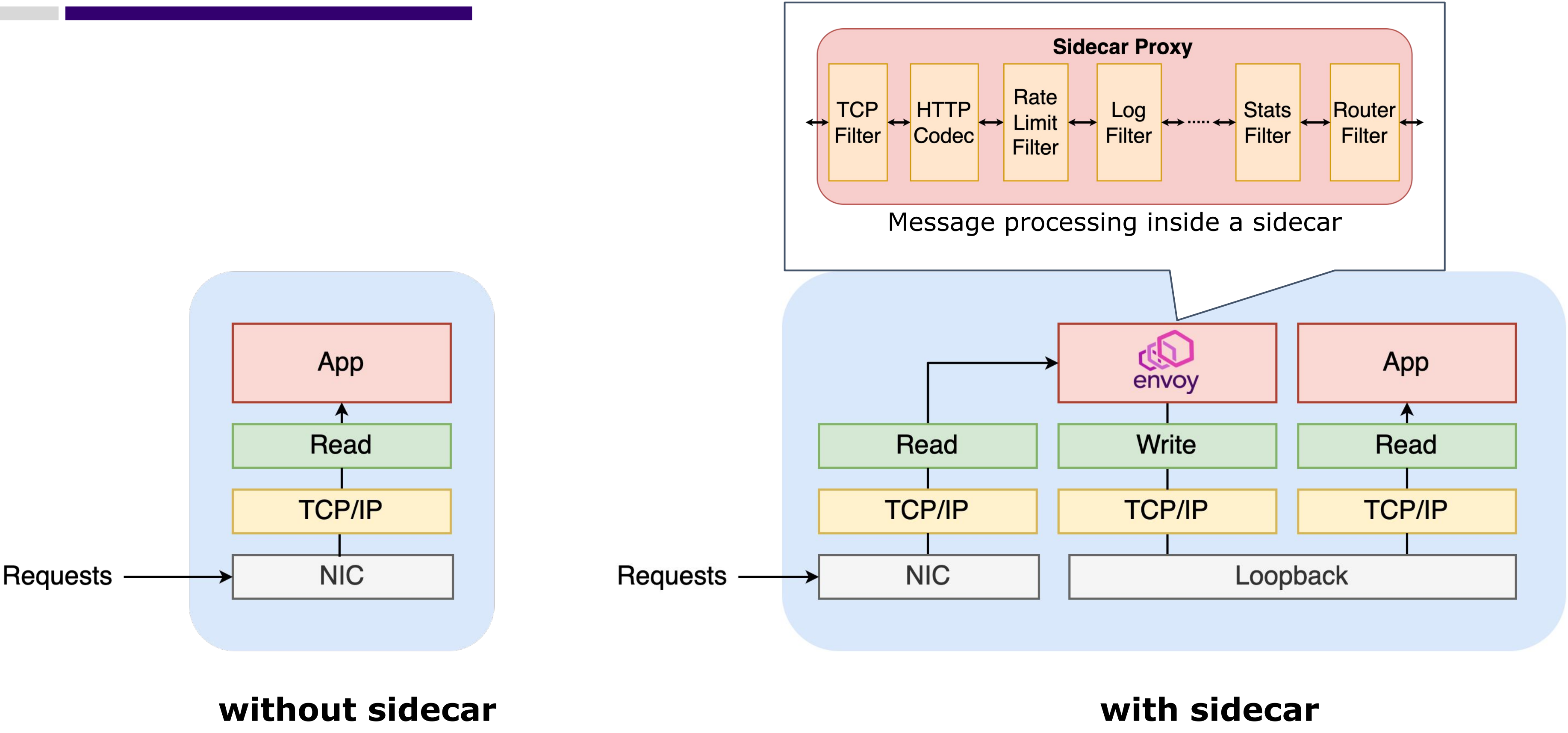
- Black-box measurement of different configurations and workloads
- Key limitation:
  - Combinatorial configuration space
    - Protocol
    - Envoy features
    - Workload (Request size and rate)
    - …

MeshInsight: Dissecting Sidecar Overheads

Goal: systematically quantify the service mesh sidecar overheads for developers to

- Navigate the performance and functionality tradeoff
- Estimate the impact of their optimizations

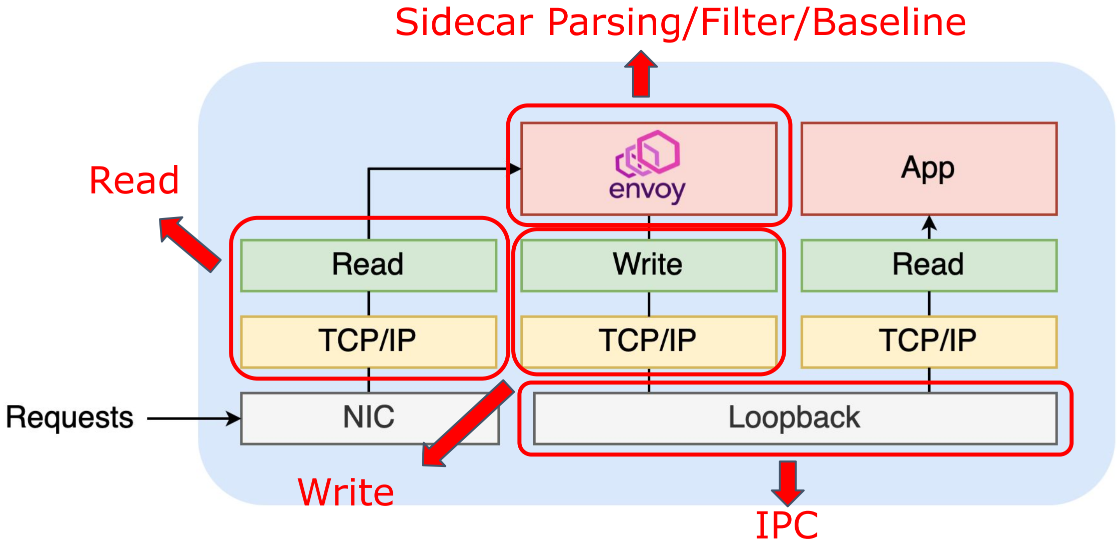
Service Mesh Data Path



Modeling per-Sidecar Overheads

- Breakdown the overhead into fine-grained components
  - Independent from each other
  - Total overhead = Sum (overhead of each component)

- Metrics
  - Latency (service time)
  - CPU usage



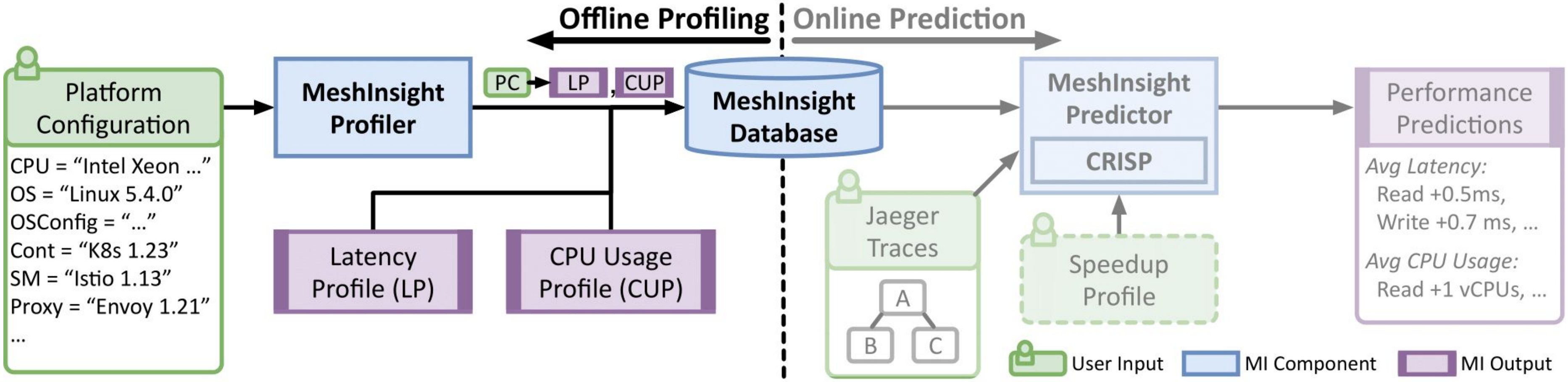
Building Component Profiles

- Component profile is a linear function of message size and rate
  - Latency: $L_x + message\_size \times l_x$
  - CPU: $message\_rate \times (C_x + message\_size \times c_x)$

Predicting the End-to-end the Overhead

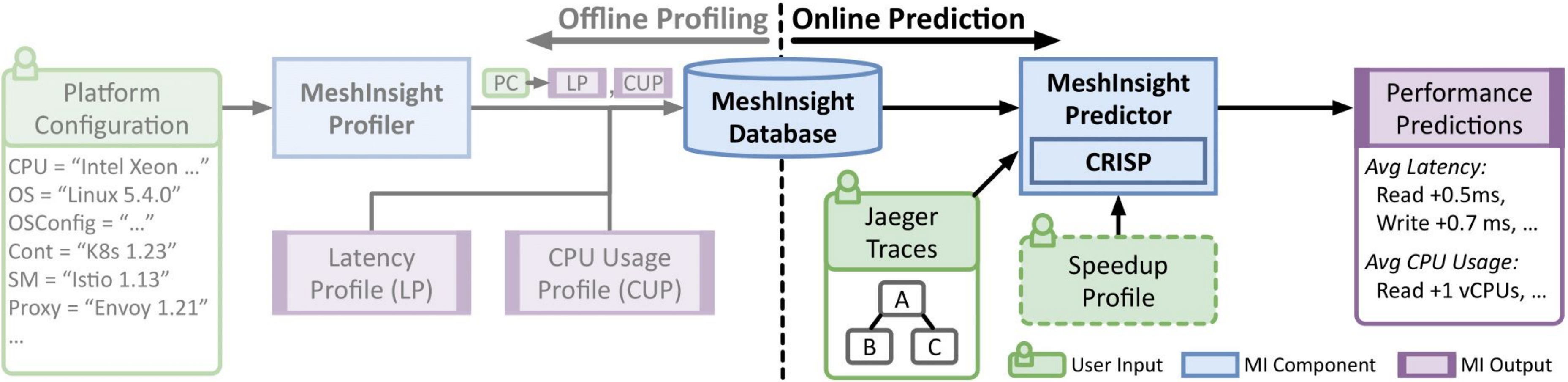
- Extended Call Graph (ECG)
  - Captures service communication patterns, platform specs, sidecar configurations, and workload information
- E2E overhead = sum of all sidecar-level overheads in ECG
  - Latency overhead requires critical path analysis

MeshInsight Architecture



https://github.com/UWNetworksLab/meshinsight

MeshInsight Architecture



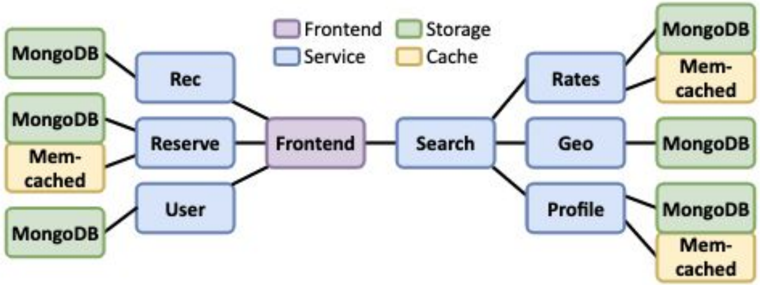
https://github.com/UWNetworksLab/meshinsight

Implementation

- Implementation:
  - Istio v1.13 and Envoy v1.21
  - Each component is profiled using an echo server
    - *eBPF* for latency
    - *perf* for CPU usage

Evaluation

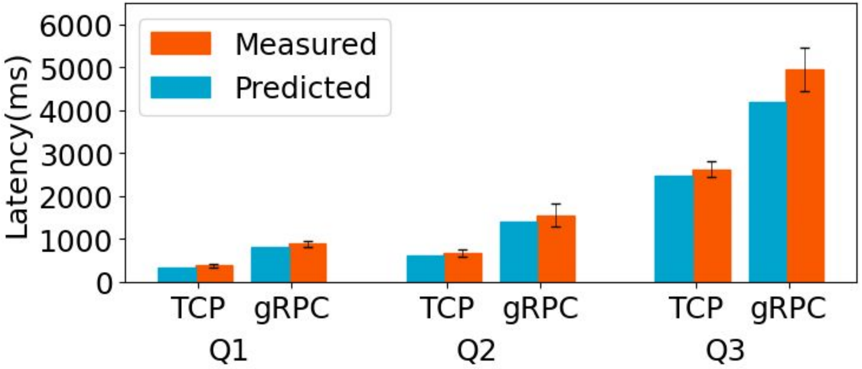
- Synthetic benchmark
  - Hotel Reservation
- Real-world microservice traces
  - Alibaba microservice trace (~20K services, >20M call graphs)

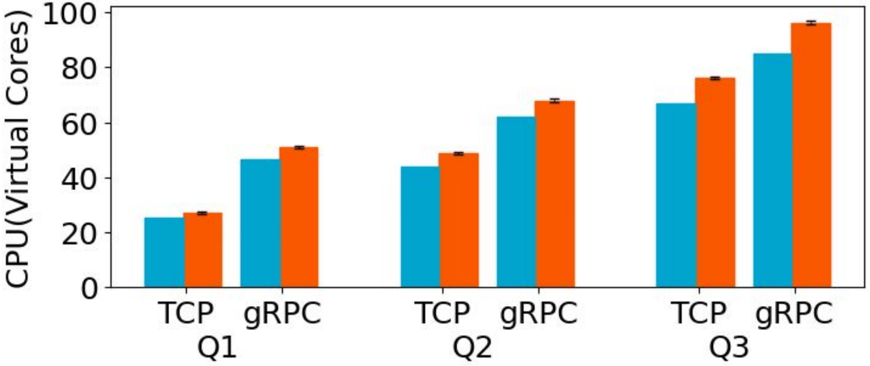


Hotel Reservation Architecture

How Well Can MeshInsight Predict Overhead?

- MeshInsight provides accurate prediction of latency and CPU usage overhead

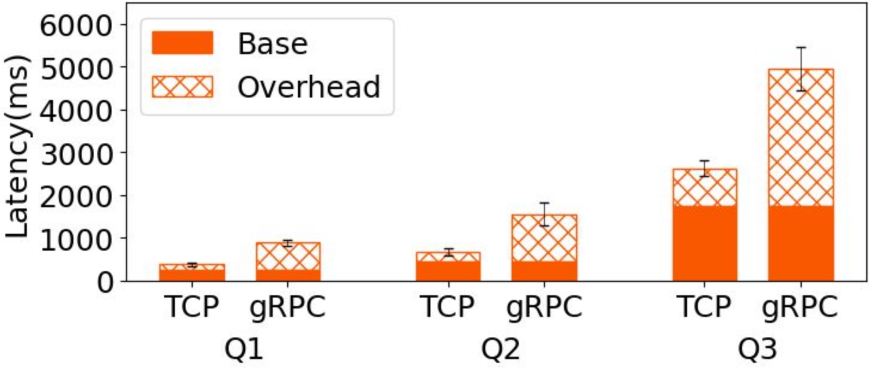


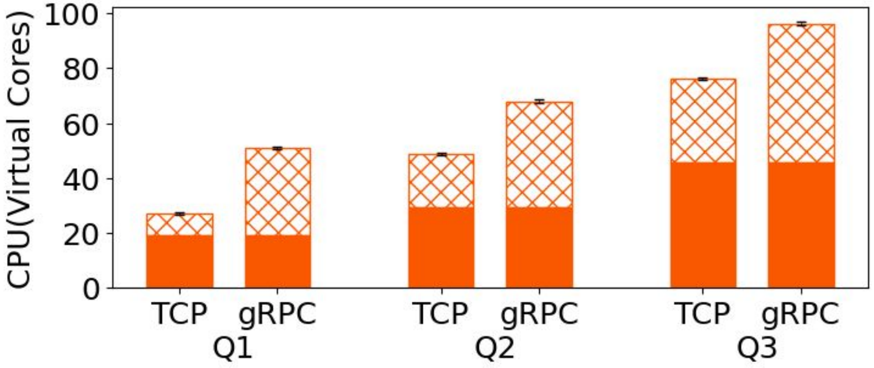


Latency and CPU usage of Hotel Reservation Benchmark

How Much Overhead Do Sidecars add?

- gRPC mode can increase the latency by up to 2.7X and CPU usage by up to 1.6X
- TCP mode overhead are lower but still noticeable

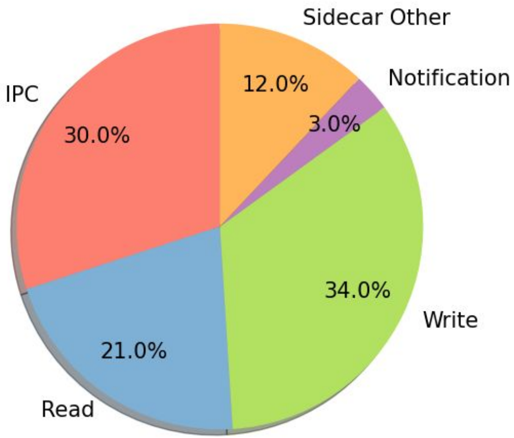




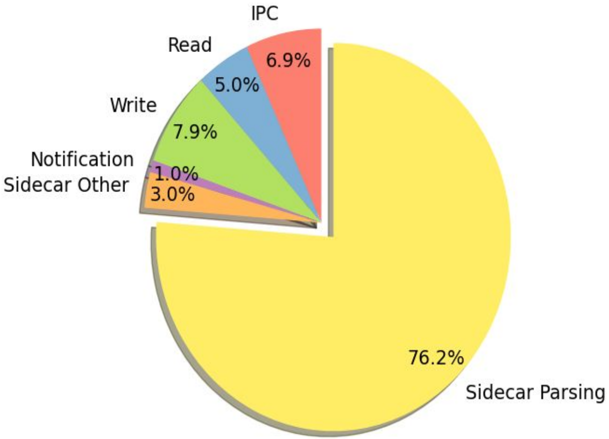
Latency and CPU usage of Hotel Reservation Benchmark

What Are the Primary Contributors of Latency Overhead?

- IPC, Read, and Write are the major sources of overhead in TCP mode
- Parsing dominates in HTTP mode



TCP Mode (+41μs)

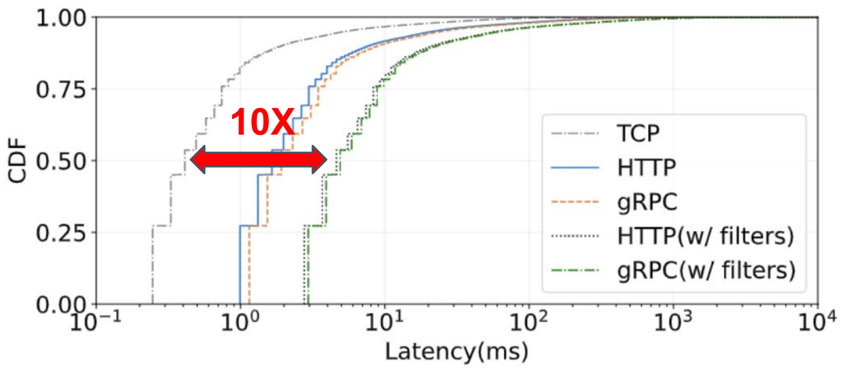


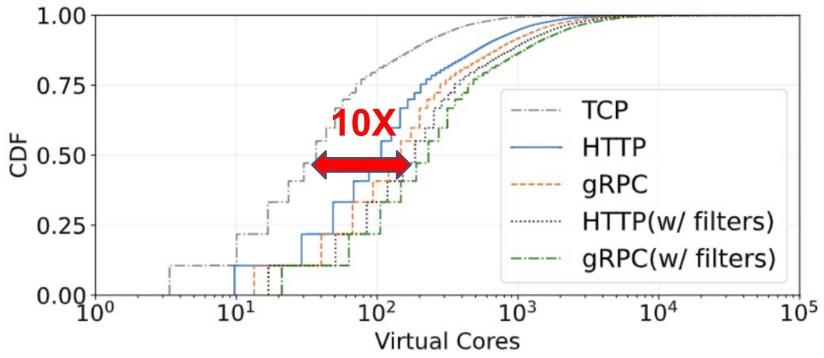
HTTP Mode (+165μs)

Message size is set to 100B

How Do Overheads Vary Across Configurations/workloads?

- Performance overhead varies by orders of magnitude for
  - Different configurations



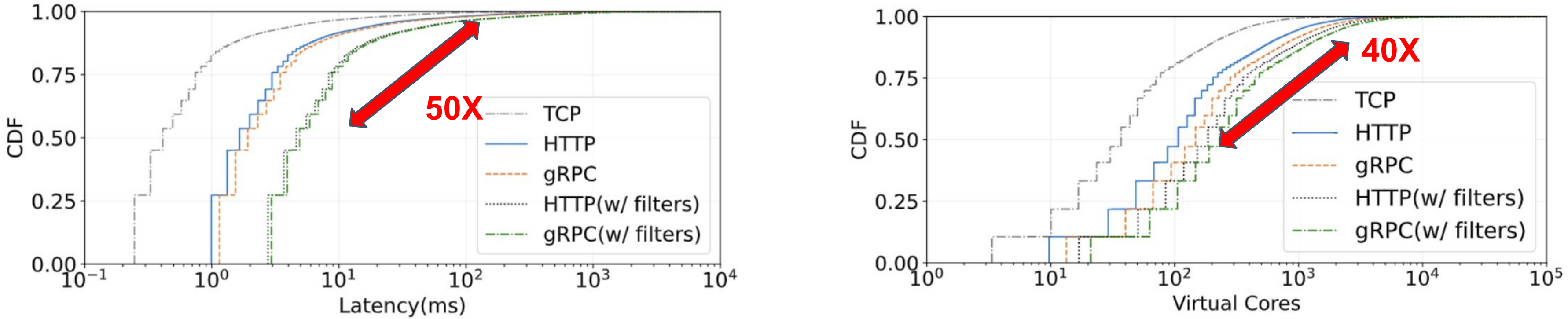


CDF of Latency and CPU usage overhead for Alibaba Microservice Trace

Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis - SoCC' 21

How Do Overheads Vary Across Configurations/workloads?

- Performance overhead varies by orders of magnitude for
  - Different configurations
  - Different call graphs

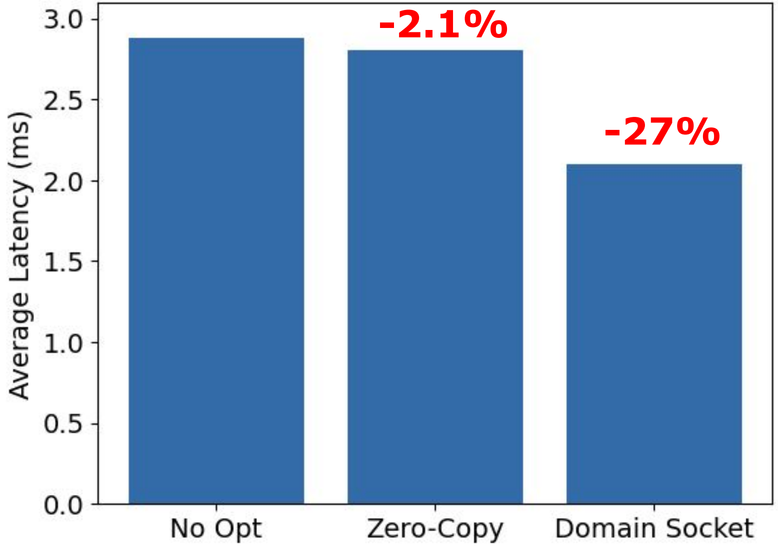


Absolute Latency and CPU usage overhead for Alibaba Microservice Trace

Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis - SoCC' 21

What Are the Impacts of Common Linux Features?

- Different optimizations have different impacts
  - Zero-copy write offers negligible improvement
  - Unix Domain socket can provide notable benefits



Conclusion

- Present MeshInsight, a tool to systematically quantify overhead of service mesh sidecars
  - Breaks down the data path into independent components
  - Predict the overhead based on extended call graphs
- Total Overhead and components' contribution varies substantially in different settings
- Some optimizations can help reduce overhead others do not
- MeshInsight is available at: **github.com/UWNetworksLab/meshinsight**