Lifting the veil on Meta's microservices

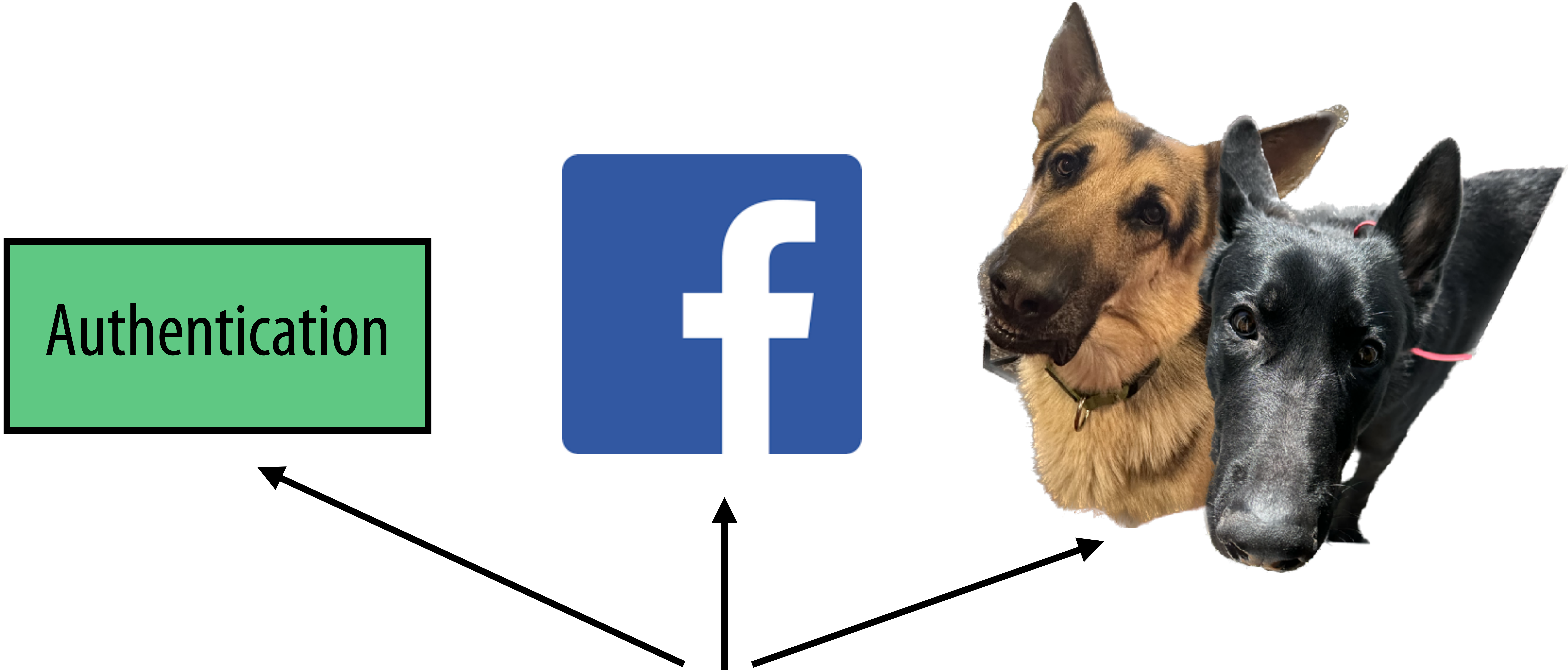
Analysis of topology and request workflows

***Darby Huye**, Yuri Shkuro, & Raja Sambasivan*

Tufts UNIVERSITY

Meta

Microservices: what are they?



Authentication

Is this a microservice???

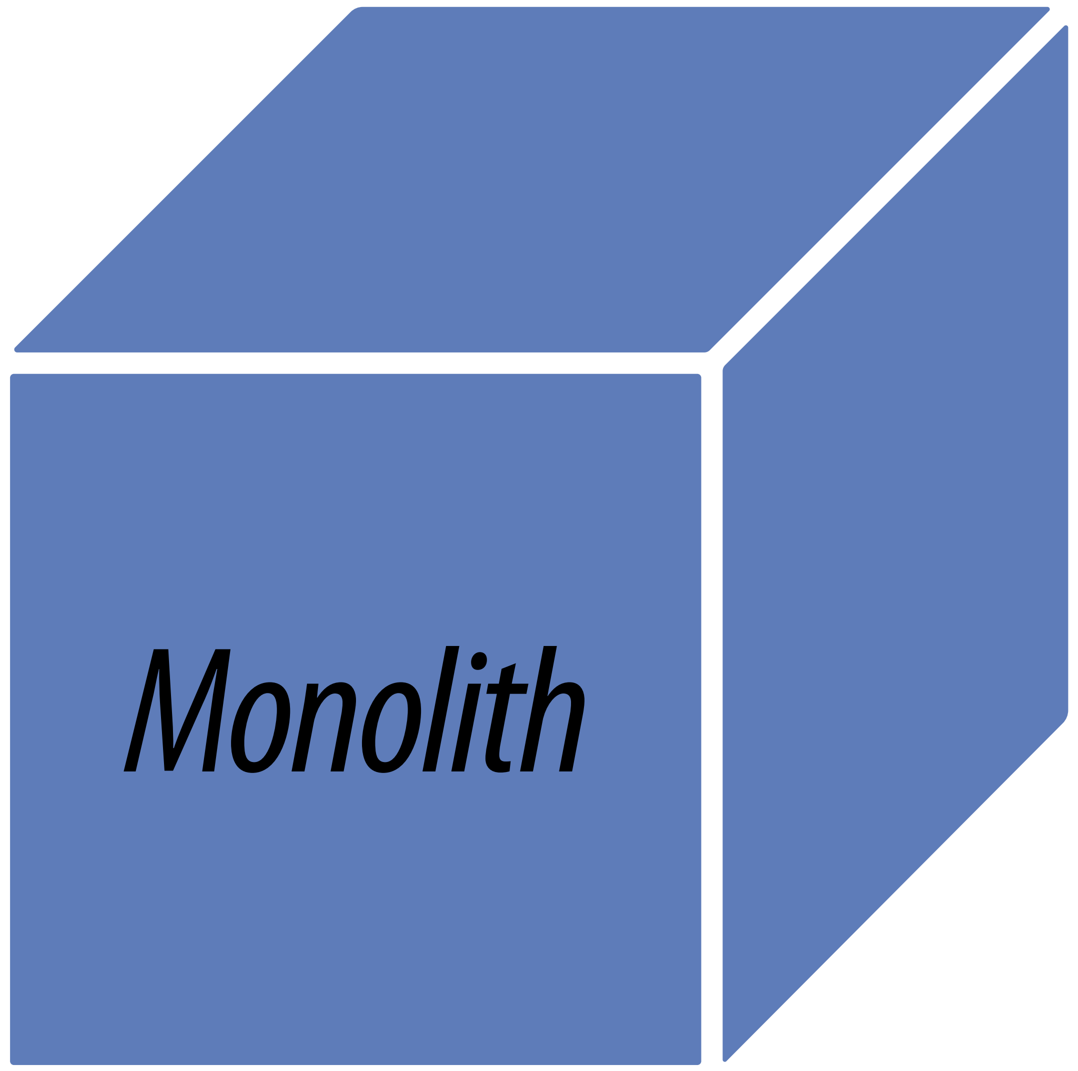
Foundational trends towards microservices

Organizational trends:

- desire for teams to work independently, want quick development, globalization of companies

Hardware trends:

- death of Moore's law leads to need for parallelization



Foundational trends towards microservices

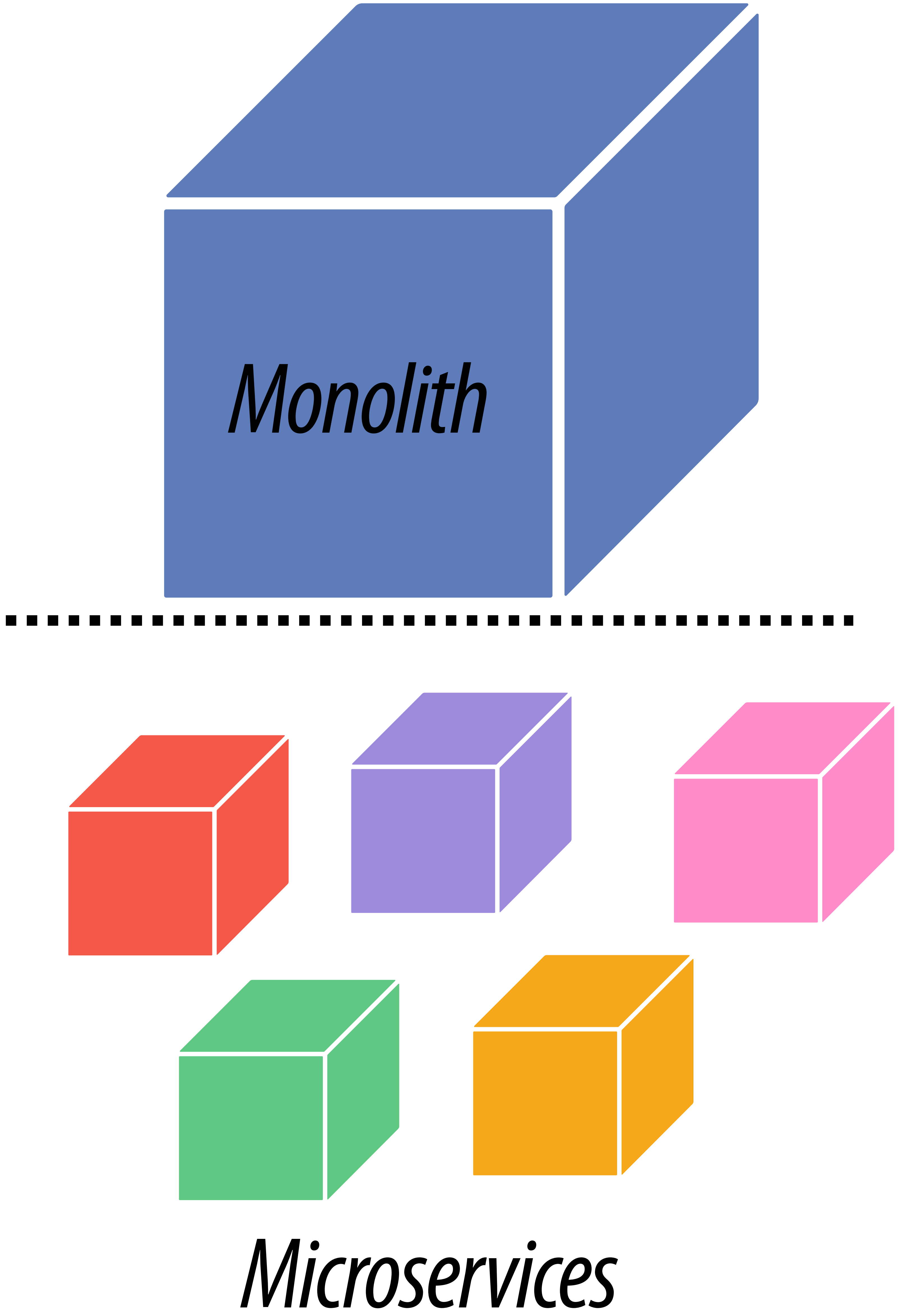
Organizational trends:

- desire for teams to work independently, want quick development, globalization of companies

Hardware trends:

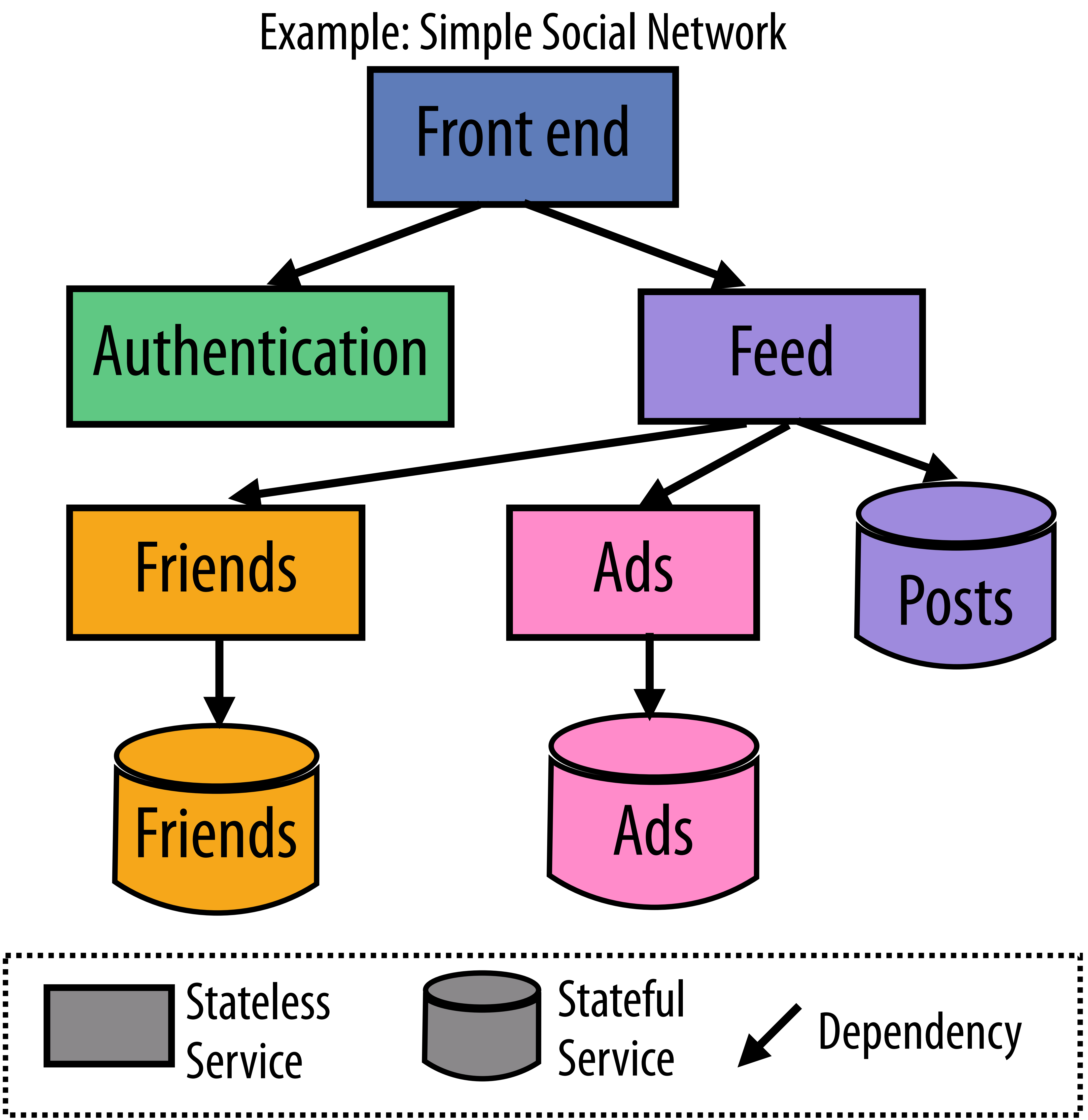
- death of Moore's law leads to need for parallelization

Basic Idea: apps composed of tiny pieces communicating over the network



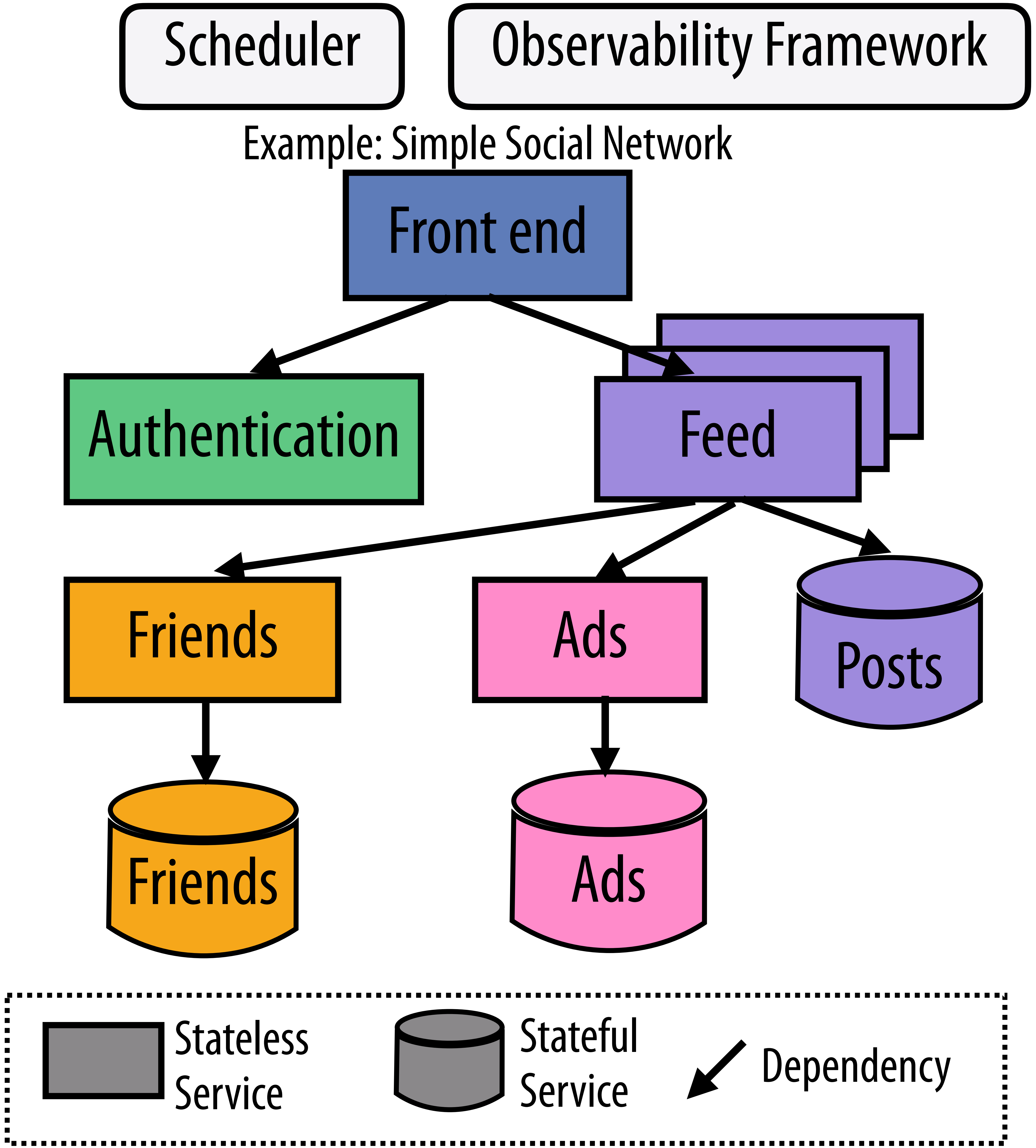
Microservices: current abstraction

- Concept of service is sufficient dimension for deployment, scaling, observability
- Independently deployable units
- Small, represent a single business capability
- Strictly hierarchical architecture
- Relatively stable topologies

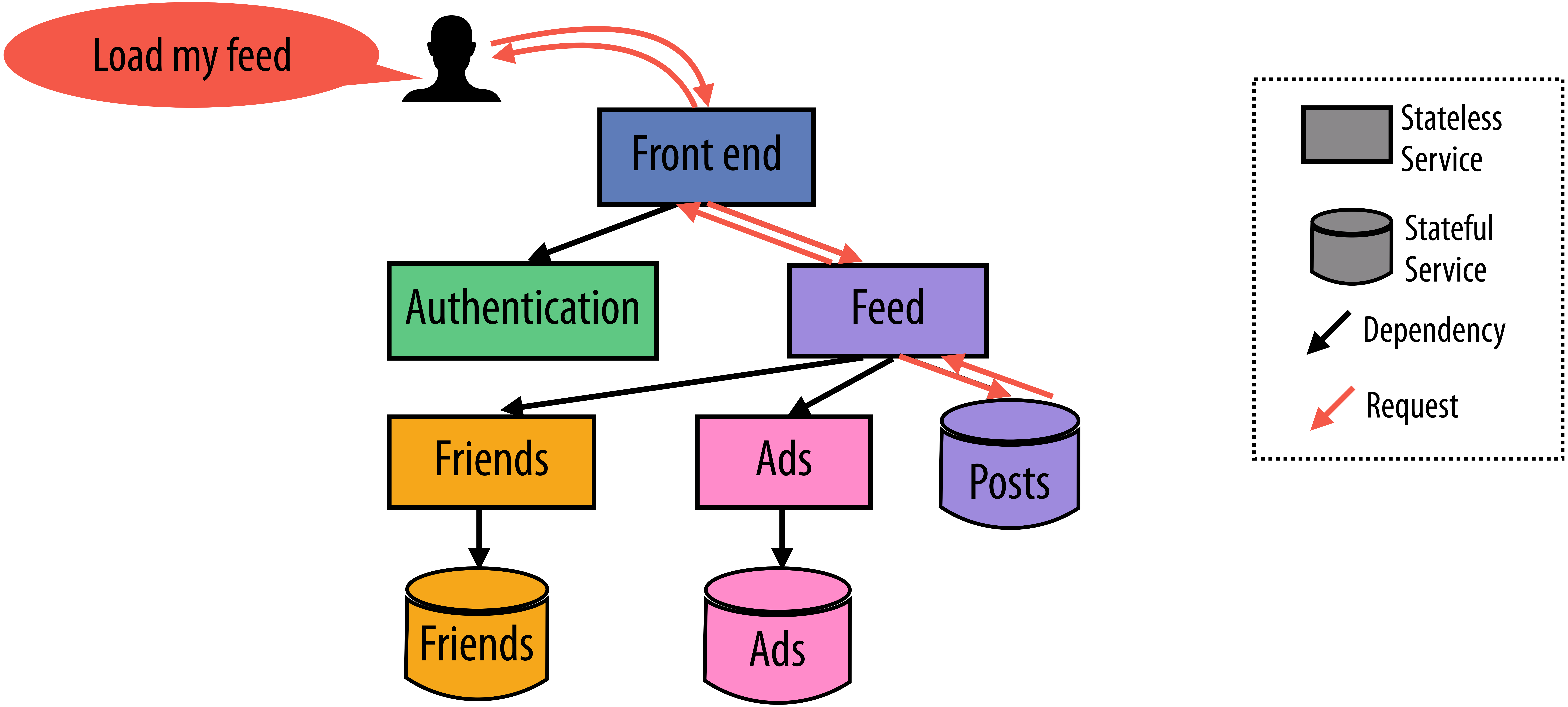


Microservices: current abstraction

- **Concept of service is sufficient dimension for deployment, scaling, observability**
- Independently deployable units
- Small, represent a single business capability
- Strictly hierarchical architecture
- Relatively stable topologies



Microservices: request workflow



Microservice Topology (Dependency Diagram)

Current state of microservice research

Microservice testbeds [ASPLOS'19, TSE'18, Bookinfo]

- small in scale and complexity

Tools evaluated on testbeds [OSDI'20, SINAN'21, ASPLOS'21]

- Focuses on topology and request workflows
- E.g., Sage: resource management using topological information
- TProf aggregate analysis of request workflows

How realistic is our abstraction?

Analysis of Meta's microservices

Topology

	Abstraction	Finding
X	Service is sufficient dimension	Service is not one size fits all
X	Topology is static	Long-term growth with daily churn
X	Services are simple	Long tail of complex services

Workflows

	Abstraction	Finding
✓	Wide & shallow	Wide & shallow
X	Traces rep. of workflows	Observability loss impacts deep traces
X	Depth predicts # calls	Variation in # calls, even locally
X	Workflows execute consistently	Variation in conc., decreased by children set

Analysis of Meta's microservices

Topology

	Abstraction	Finding
X	Service is sufficient dimension	Service is not one size fits all
X	Topology is static	Long-term growth with daily churn
X	Services are simple	Long tail of complex services

Workflows

	Abstraction	Finding
✓	Wide & shallow	Wide & shallow
X	Traces rep. of workflows	Observability loss impacts deep traces
X	Depth predicts # calls	Variation in # calls, even locally
X	Workflows execute consistently	Variation in conc., decreased by children set

Methodology: Topology

Service History (22 months)

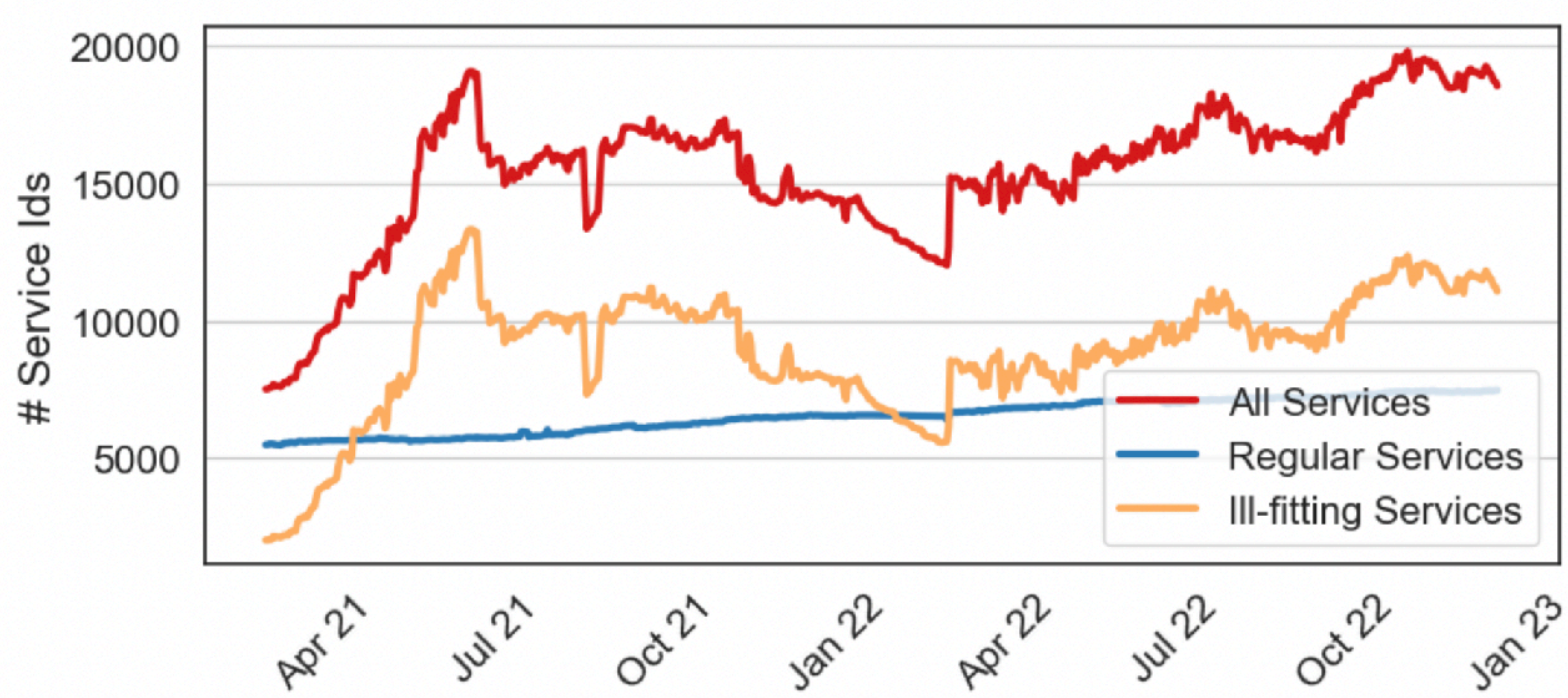
- Service deployment and lifetimes

Service Complexity (1 day)

- Endpoints exposed by deployed services, replication factors, and dependencies

Analysis granularity: ***service id,*** a unique name assigned to each service (e.g. authentication)

Is service a sufficient dimension?



60% of service ids are inference_platform + #####

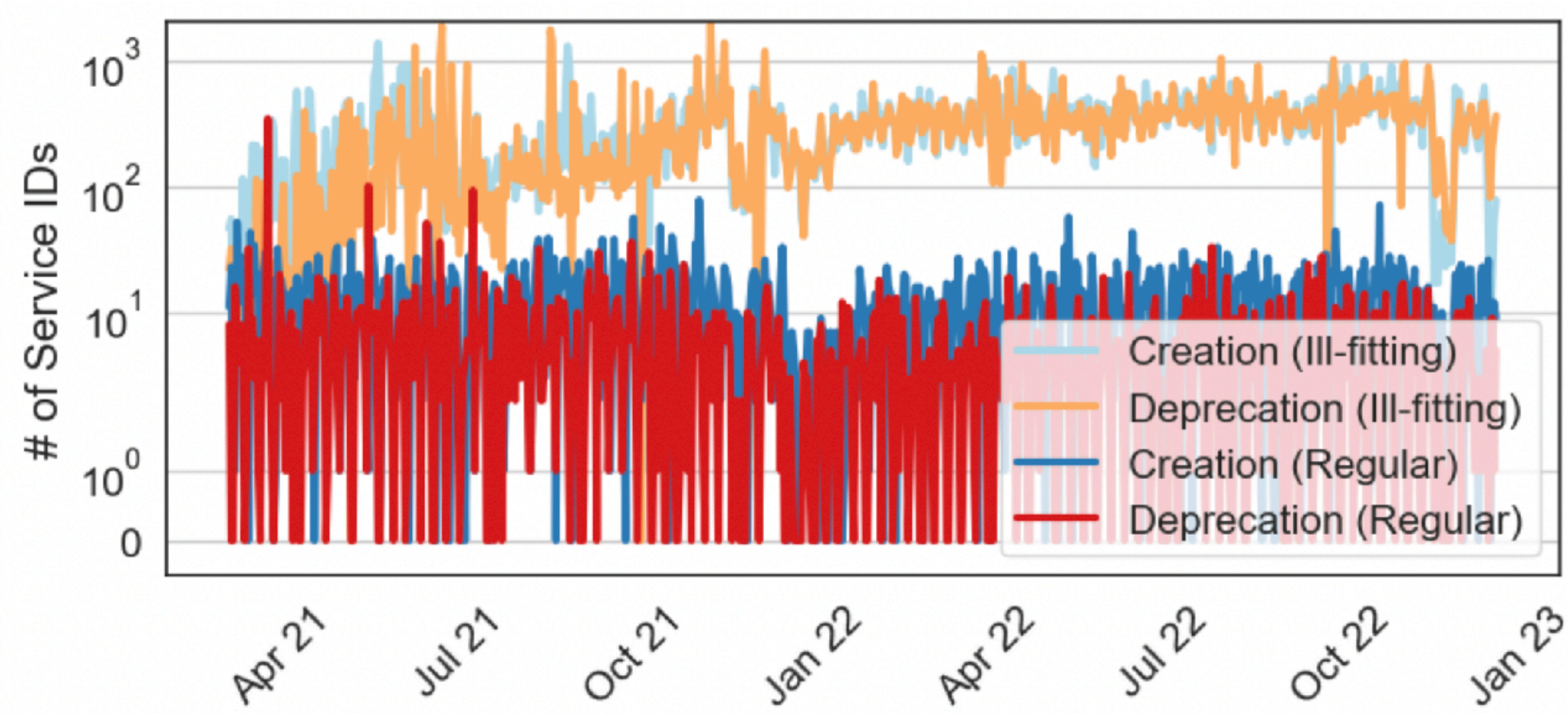
Inference platform: includes tenant info in service id to utilize infrastructure support

Service granularity is not sufficient for all management tasks: at least multi-tenancy and data placement must be considered

Daily churn of deployed services

Creation:

new service id deployed for first time

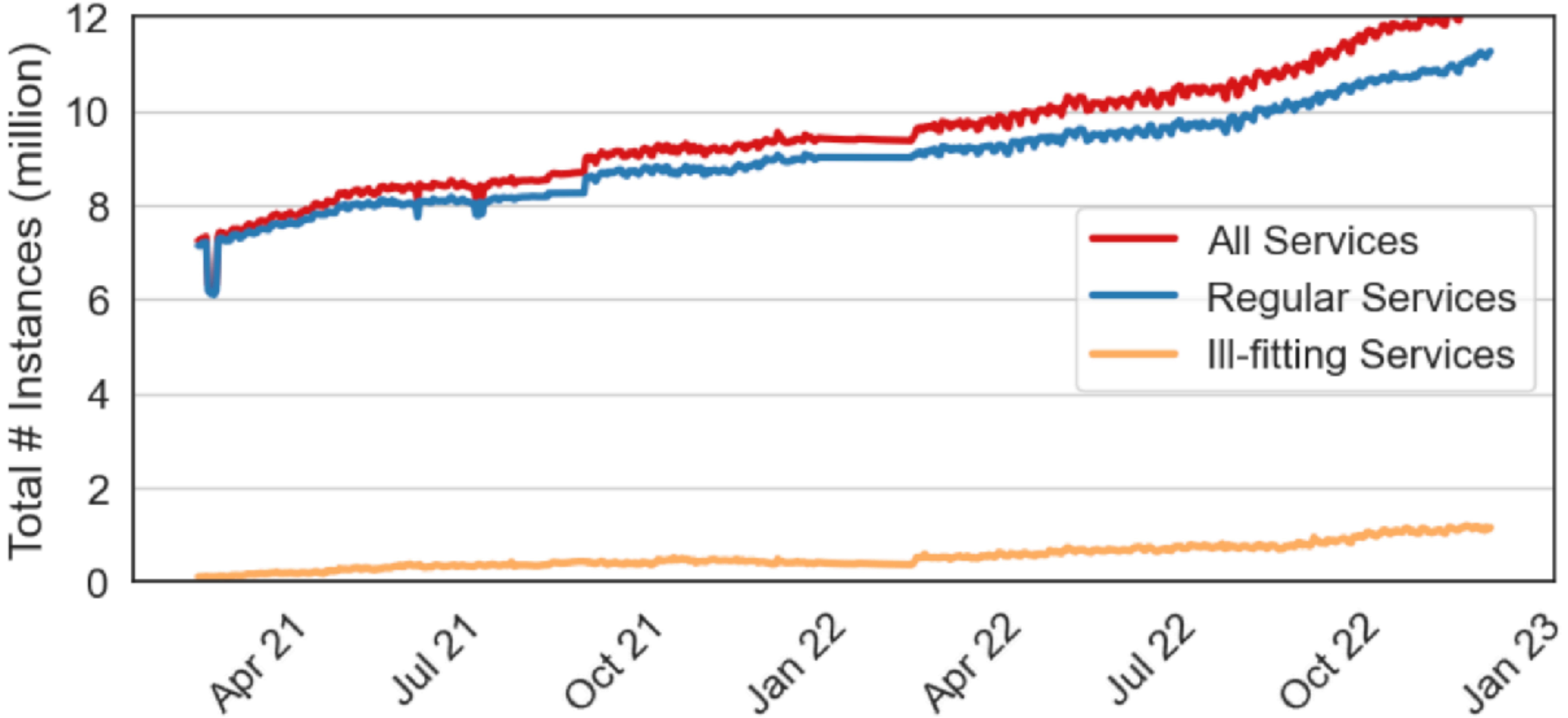


Deprecation:

last time service id deployed

- 89% of new services deployed were also deprecated
- 40% of regular services lived the entire time range

Long-term growth in total deployed instances



- Total number of deployed service instances nearly doubled
- Growth is due to new (regular) service ids, not an increase in replication factors for existing services

Analysis of Meta’s microservices

Topology

	Abstraction	Finding
X	Service is sufficient dimension	Service is not one size fits all
X	Topology is static	Long-term growth with daily churn
X	Services are simple	Long tail of complex services

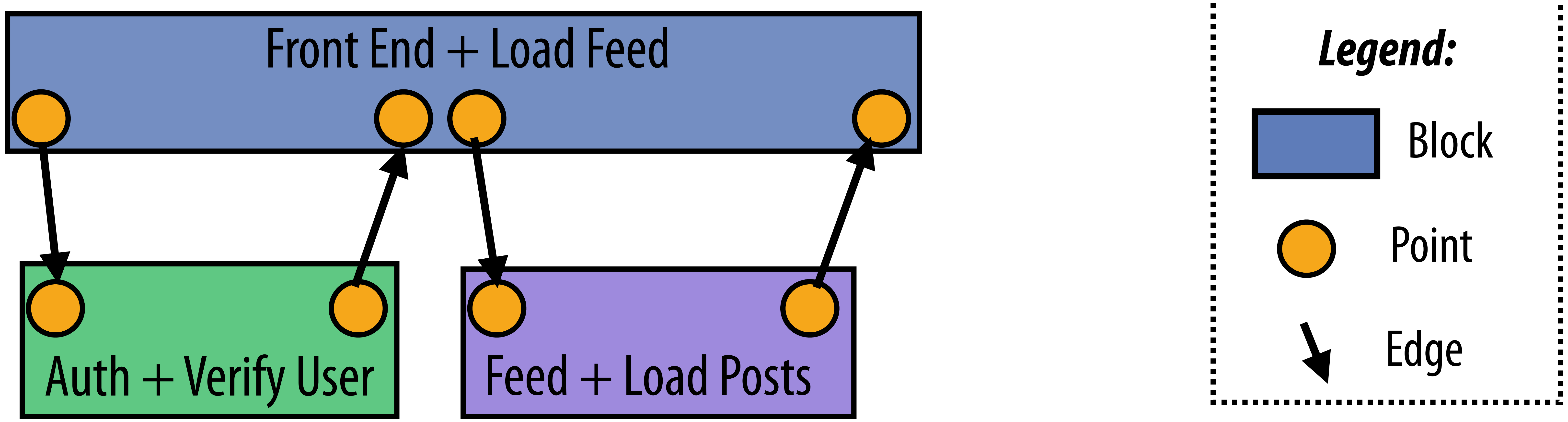
Workflows

	Abstraction	Finding
✓	Wide & shallow	Wide & shallow
X	Traces rep. of workflows	Observability loss impacts deep traces
X	Depth predicts # calls	Variation in # calls, even locally
X	Workflows execute consistently	Variation in conc., decreased by children set

Methodology: Workflows

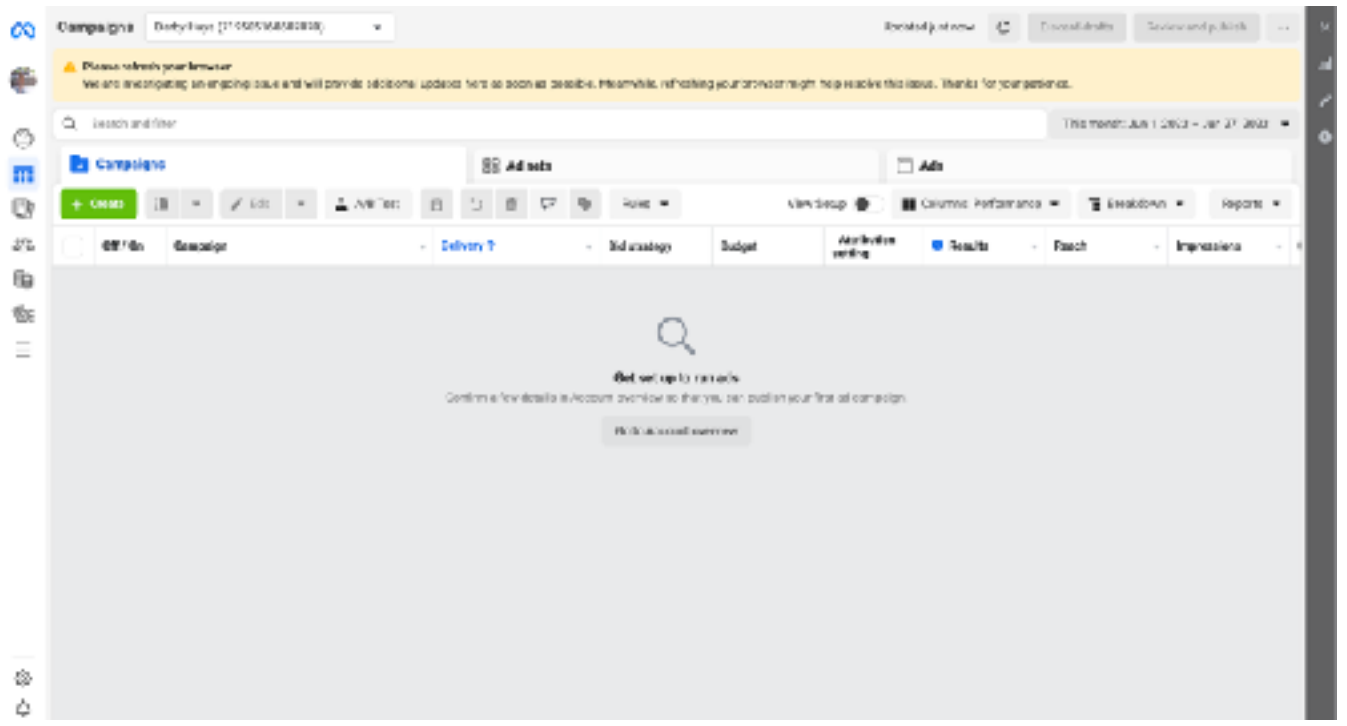
- ***Distributed tracing***: graphs capturing the work done on behalf of a request
- Canopy [SOSP'17]: Meta's distributed tracing framework
- Traces can be sampled anywhere in the topology

Example Canopy Trace



Methodology: Workflows

Used traces collected on a single day from three important trace profiles:



Ads **Manager**

3.2M traces

Random Sampling

(0.01%)

Fetch **Notifications**

87,000 traces

Adaptive Sampling

(1 trace/second)

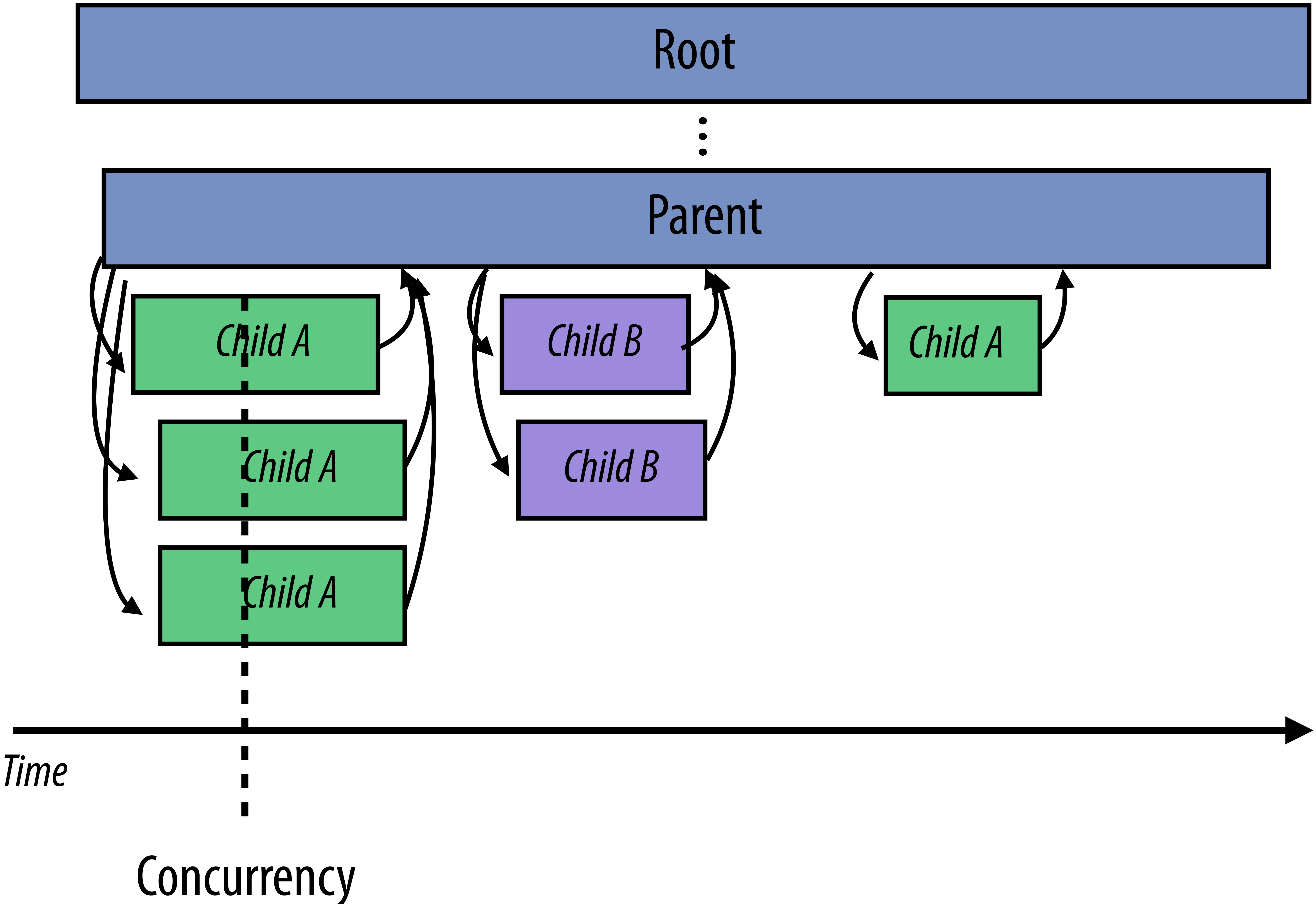
RaaS (Ranking of items)

3.3M traces

Adaptive Sampling

(25 trace/second)

Description of analyzed workflow properties



Node names:
service id + endpoint name

Parent's characteristics:

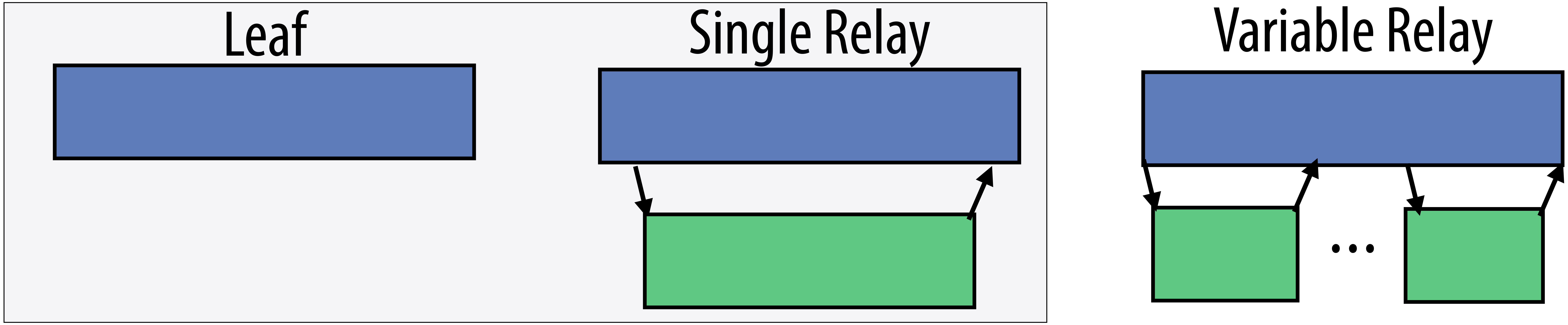
Children set: A B

Number of calls: 6

Max concurrency rate: 0.5 (3/6)

Predicting number of children

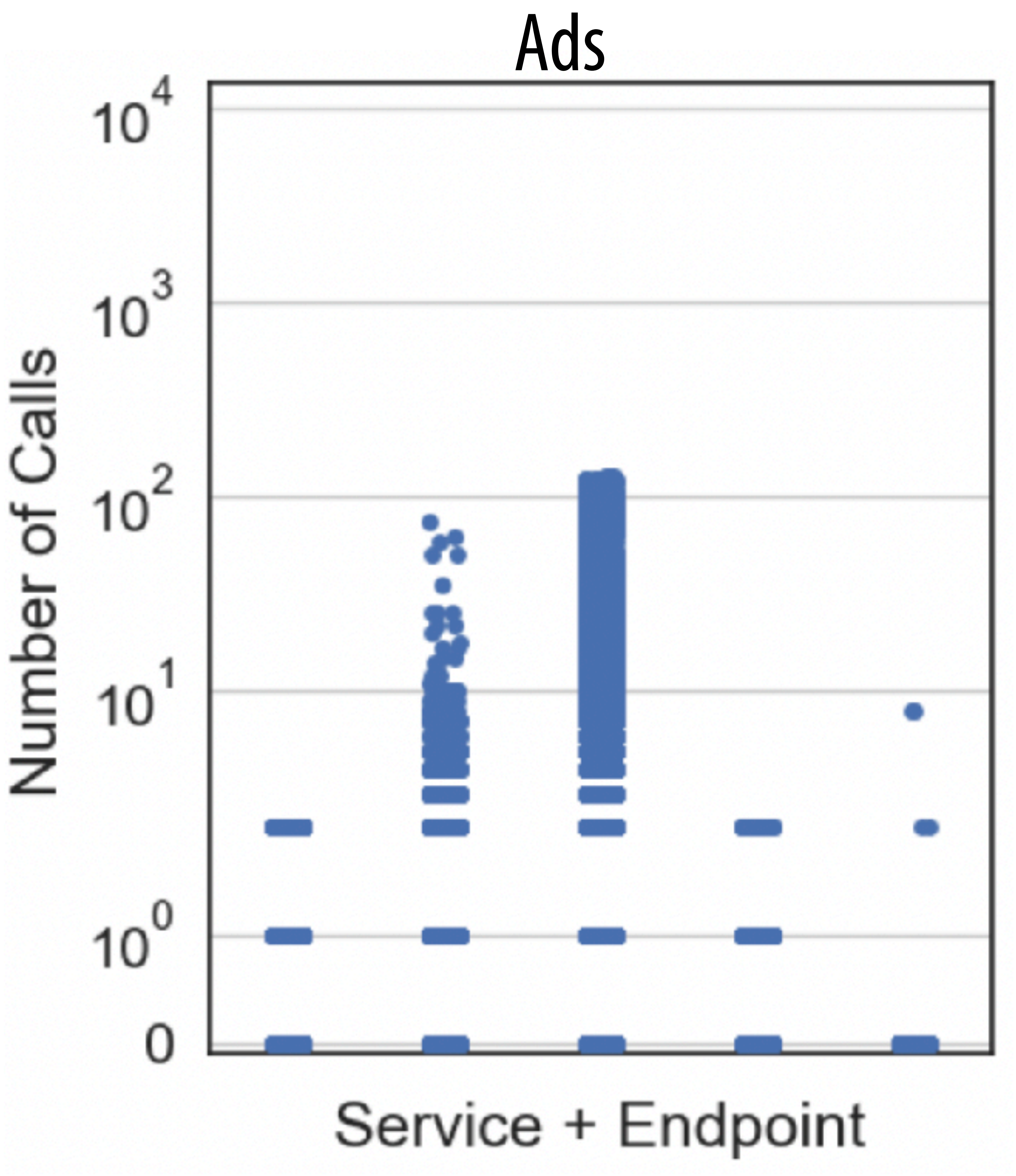
Identified three categories of nodes:



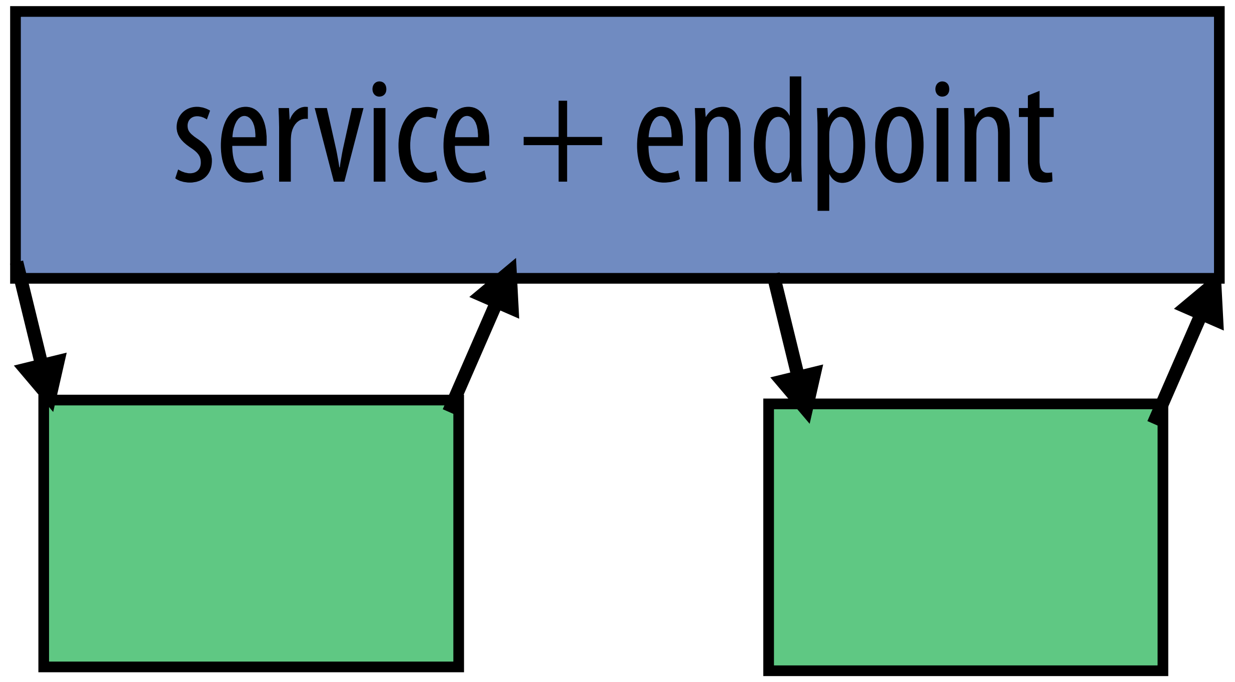
The majority of service + endpoints are leaves or single relays:

- Ads Manager: 54%
- Fetch Notifications: 66%
- RaaS: 72%

Predicting # of children for variable relays



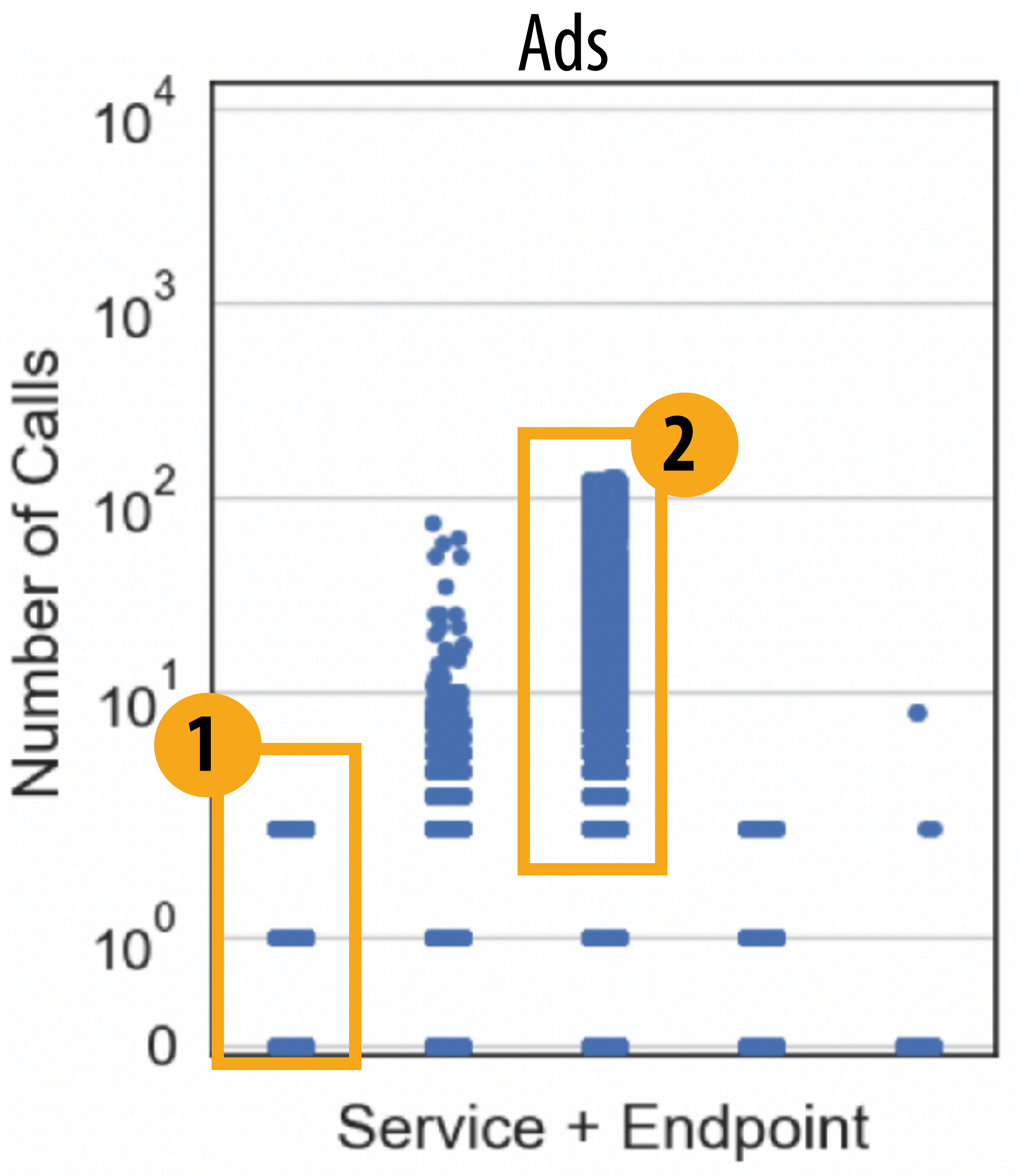
Number of calls issued by a service + endpoint



Predicting # of children for variable relays

Ads
Fetch
RaaS
Number of Calls
10^4
10^3
10^2
10^1
10^0
0
Service + Endpoint
Service + Endpoint
Service + Endpoint

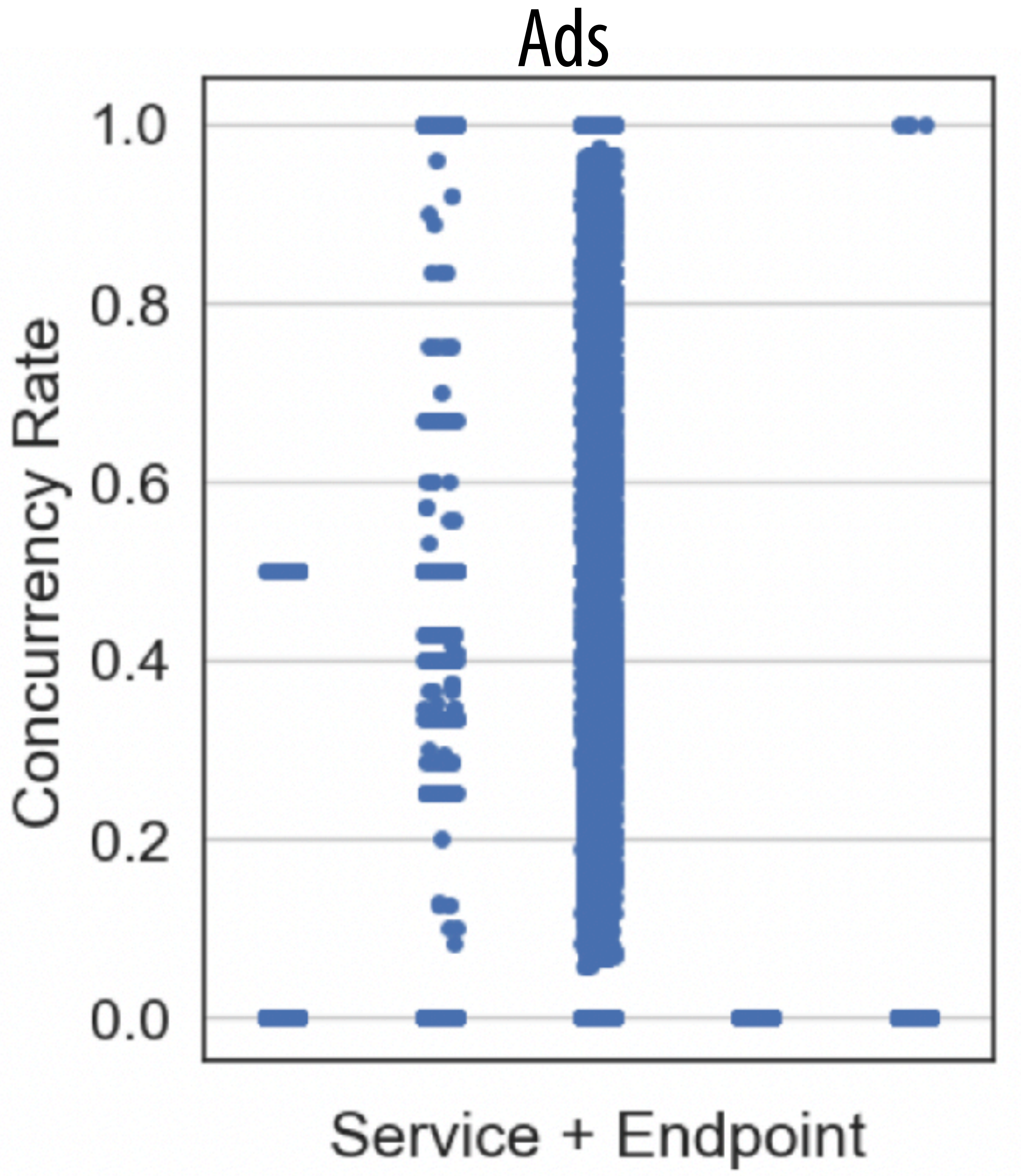
Predicting # of children for variable relays

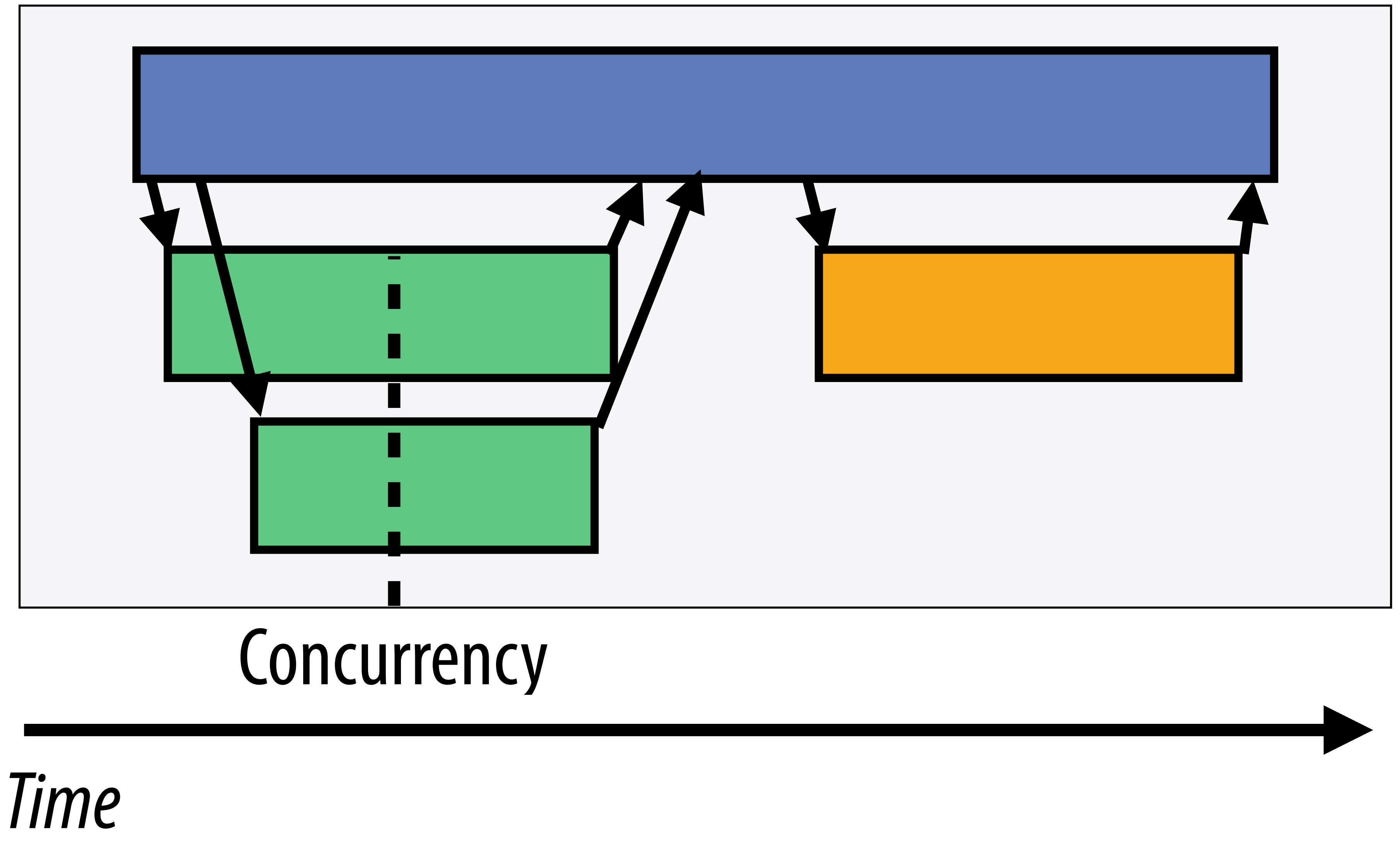


Variation in number of calls is often attributed to:

1. Different children sets
2. Database accesses

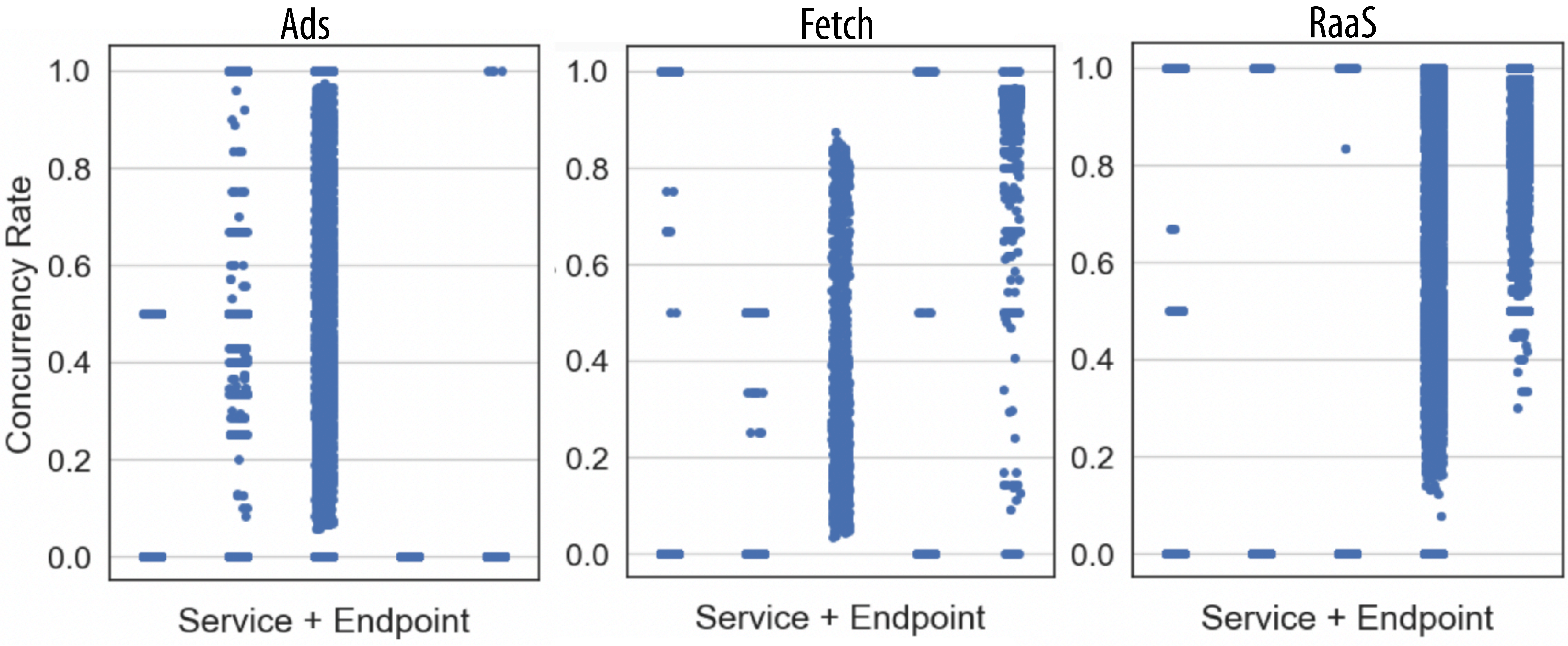
Predicting concurrency rates of variable relays



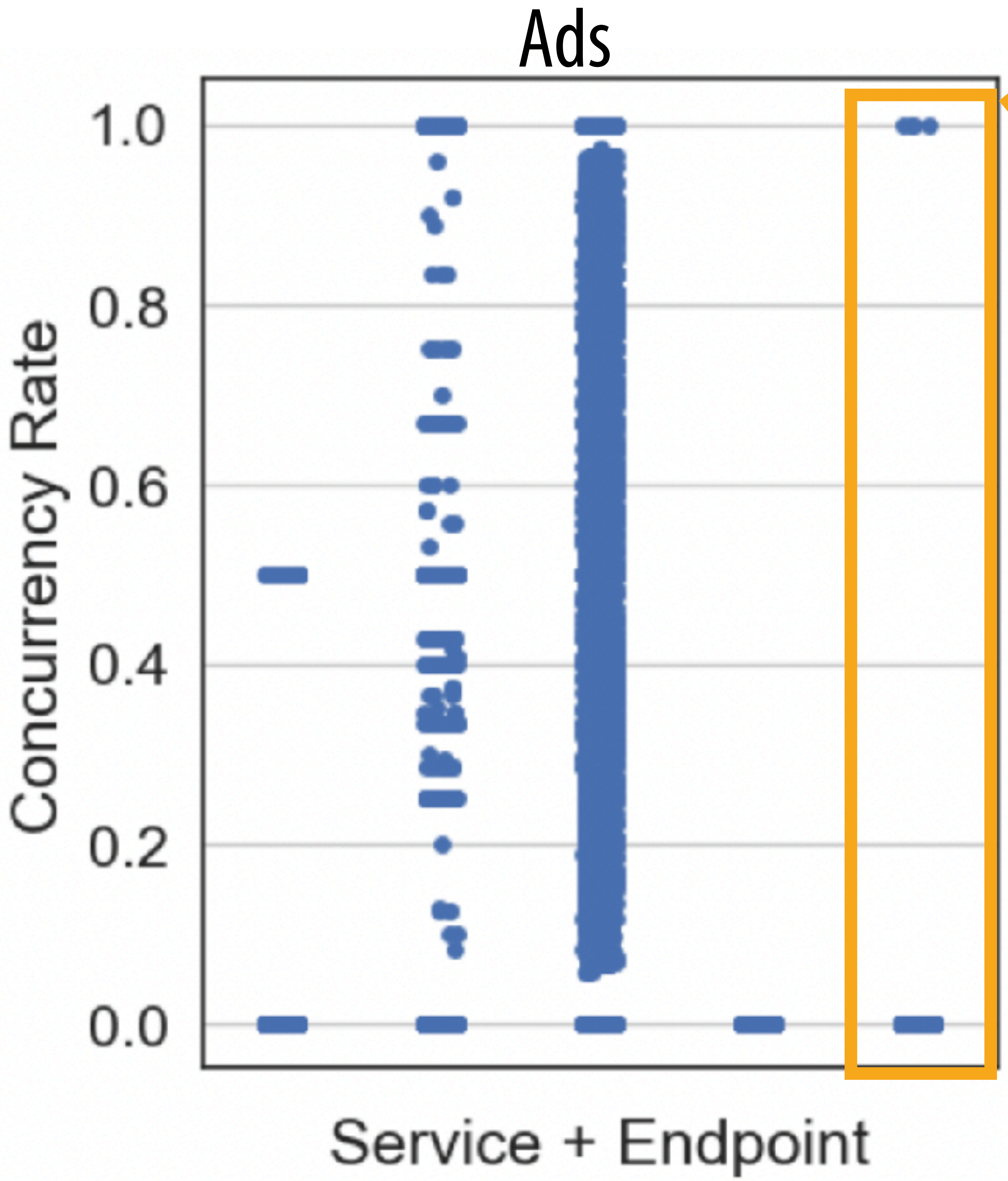


Max Concurrency Rate: 2/3 (0.67)

Predicting concurrency rates of variable relays



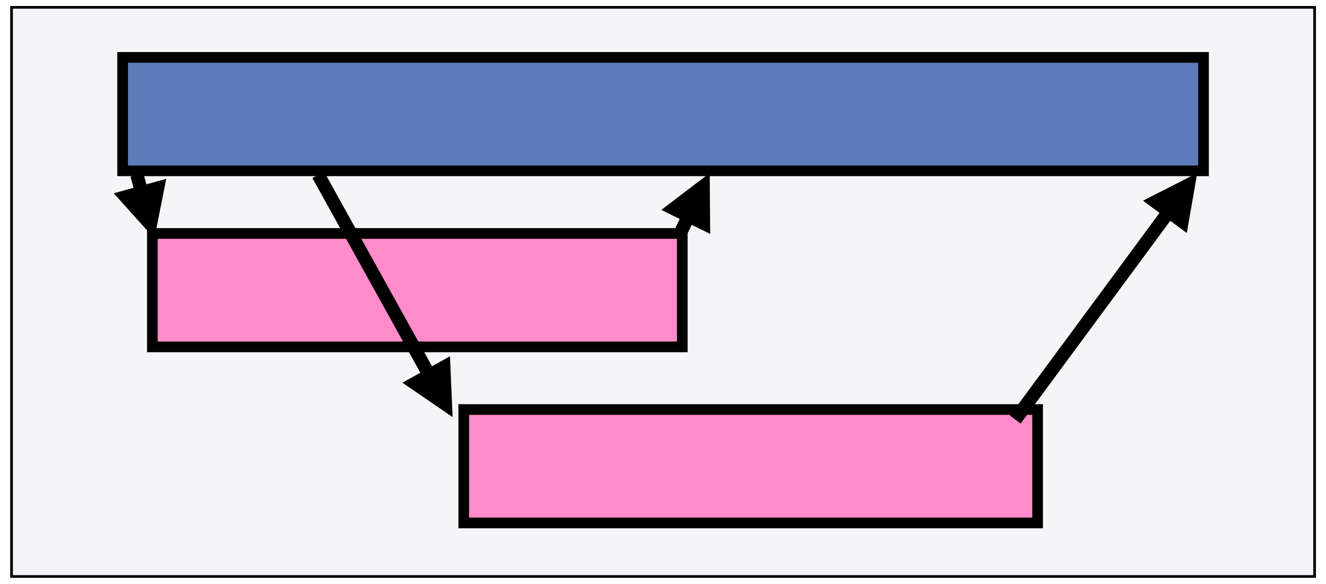
Predicting concurrency rates of variable relays



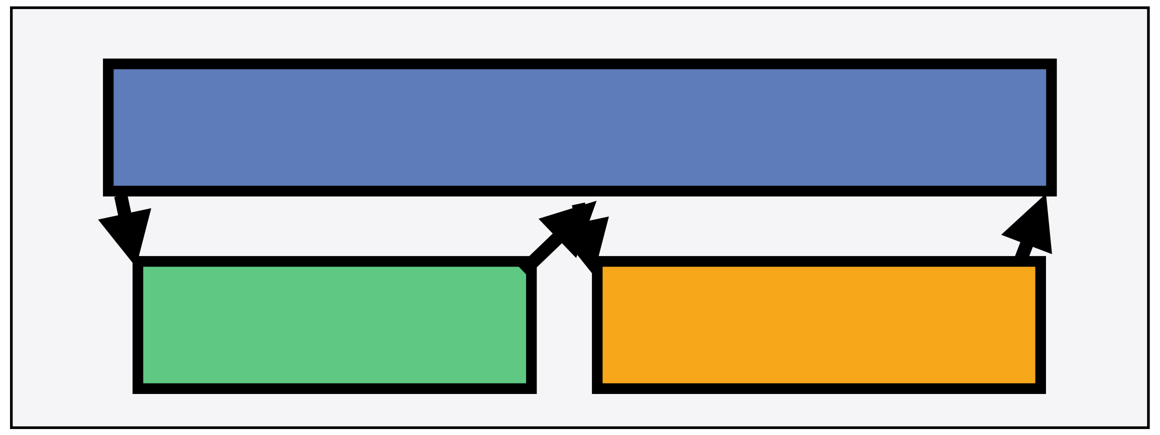
service + endpoint

Children are either 100% concurrent or 0% concurrent

Children sets: 1. 🟪 2. 🟩 🟧



Always 100% concurrent



Always 0% concurrent

Children set provides visibility into code logic, explaining dependencies

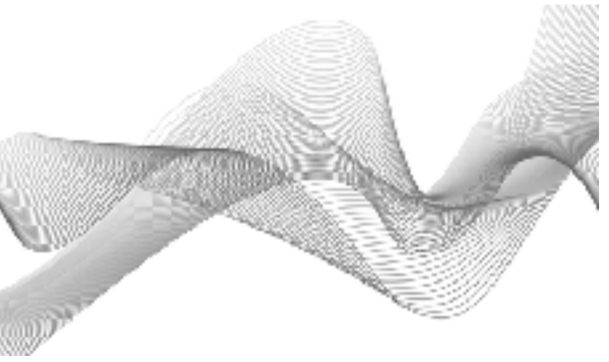
Outline

- Introduction
- Overview of Findings
- Topology
- Workflows
- **Implications**

Implications

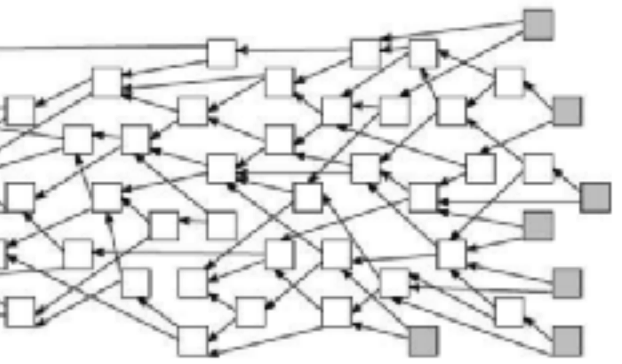
Testbeds should be extended to provide support for:

- Heterogeneity of services, churn & growth of deployed instances
- Variable concurrency, number of children, and children sets



Tooling that uses topology for resource management [ASPLOS'21, OSDI'20, SINAN'21]:

- Should be adaptable to dynamic topology



Tooling that uses workflows for performance prediction, diagnosis, capacity planning [Tprof'21, SoCC'19, VAIF'21, ATC '22]:

- Need to assume significant diversity in workflows

Summary

Microservice abstraction should be extended to support different types of archs.

Topology

Service is not one size fits all

Long-term growth with daily churn

Long tail of complex services

Workflows

Wide & shallow

Observability loss impacts deep traces

Variation in # calls, even locally

Variation in conc., decreased by children set

Data available @ github.com/facebookresearch/distributed_traces

References

- [SOSP’17]: Canopy: An End-to-End Performance Tracing And Analysis System, SOSP’17, Kaldor et al.
- [ATC ’22]: CRISP: Critical Path Analysis of Large-Scale Microservice Architectures, ATC’22, Zhang et al.
- [SoCC ‘21]: Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis, SoCC’21, Lou et al.
- [JSEP ‘22]: Characterizing and synthesizing the workflow structure of microservices in ByteDance Cloud, JSEP’22, Wen et al.
- [JSys’22]:[SoK] Identifying Mismatches Between Microservice Testbeds and Industrial Perceptions of Microservices, JSYS’22, Huye & Shesagiri et al.
- [ASPLOS’19]: An Open-Source Benchmark Suite for Microservices and Their Hardware-Software Implications for Cloud & Edge Systems, ASPLOS’19, Gan et al.
- [TSE’18]: Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study, TSE’18, Zhou et al.
- [ASPLOS’21]: Sage: Practical & Scalable ML-Driven Performance Debugging in Microservices, ASPLOS’21, Gan et al.
- [Tprof’21]: tprof: Performance profiling via structural aggregation and automated analysis of distributed systems traces, SoCC’21, Huang et al.

References

- [OSDI’20]: FIRM: An intelligent fine-grained resource management framework for slo-oriented microservices, OSDI’20, Qiu et al.
- [VAIF’21]: Automating instrumentation choices for performance problems in distributed applications with VAIF, SoCC’21, Toslali et al.
- [SINAN’21]: Sinan: ML-based and qos-aware resource management for cloud microservices. ASPLOS’21, Zhang et al.
- [NSDI’11]: Diagnosing performance changes by comparing request flows, NSDI’11, Sambasivan et al.
- [SoCC’19] Sifter: Scalable Sampling for Distributed Traces, without Feature Engineering, SoCC’19, Las-Casas et al.
- [BookInfo] Istio, https://istio.io/latest/docs/examples/bookinfo/