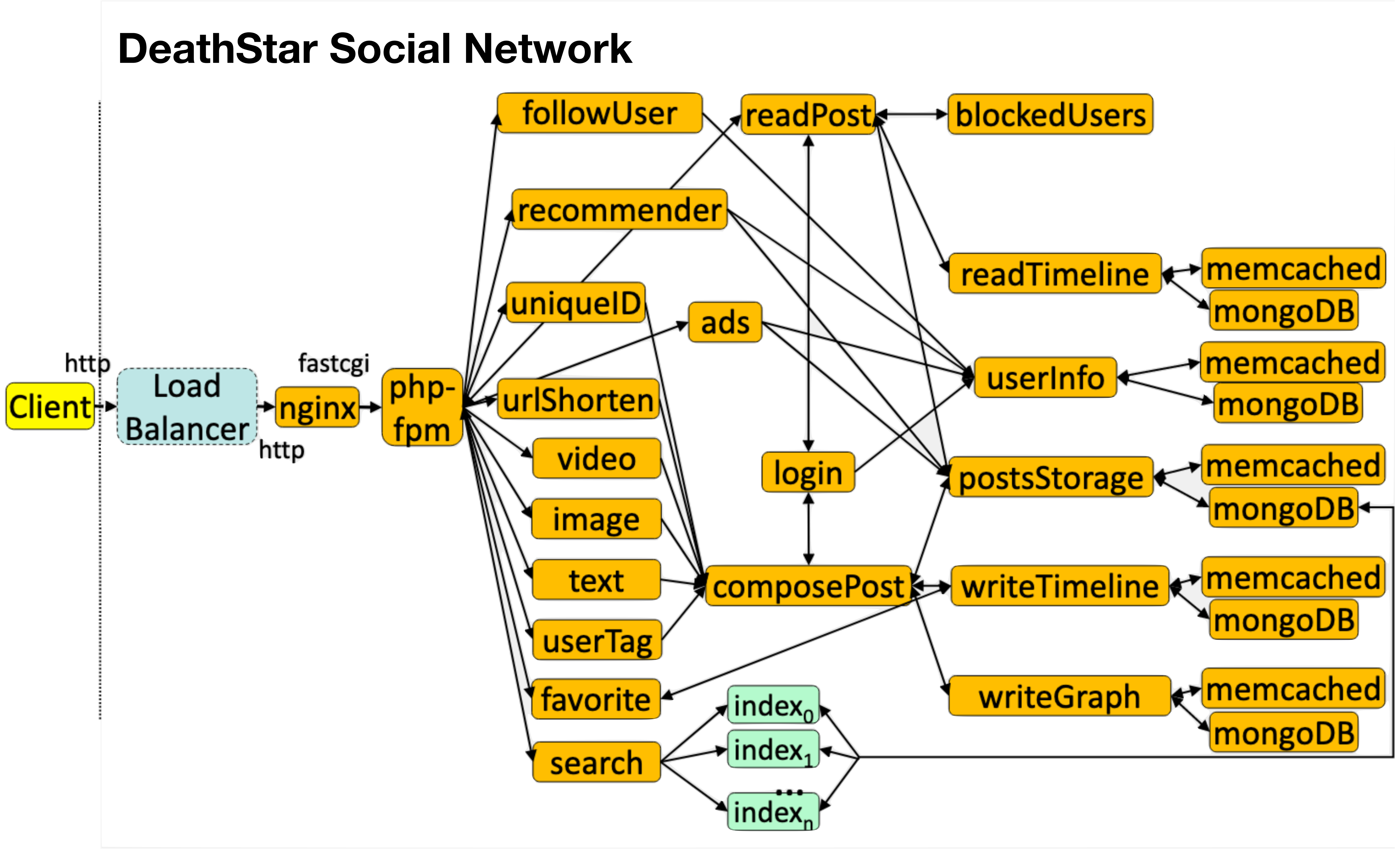
The Benefit of Hindsight: Tracing Edge-Cases in Distributed Systems

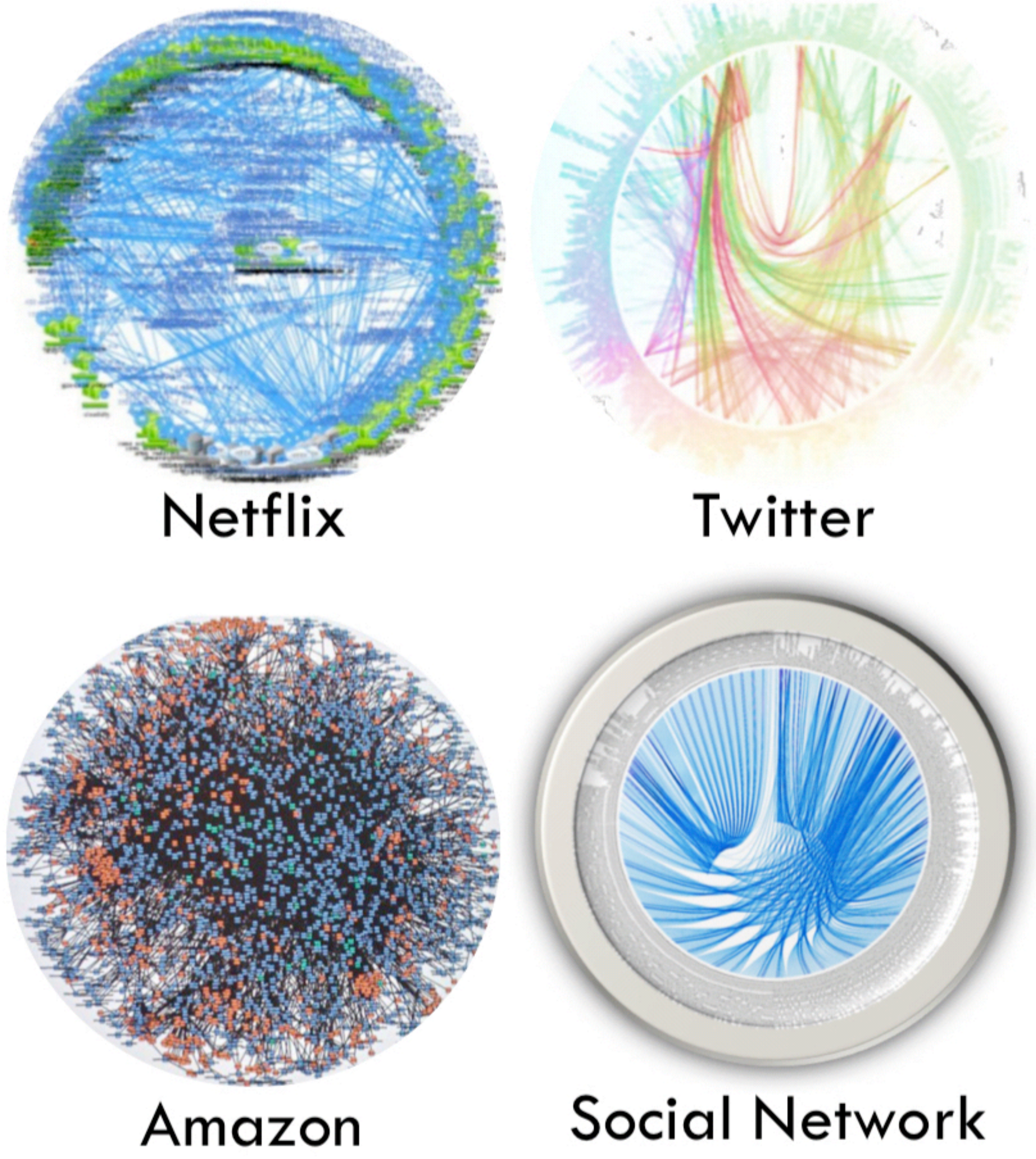
Lei Zhang, Zhiqiang Xie, Vaastav Anand, Ymir Vigfusson, Jonathan Mace

EMORY UNIVERSITY

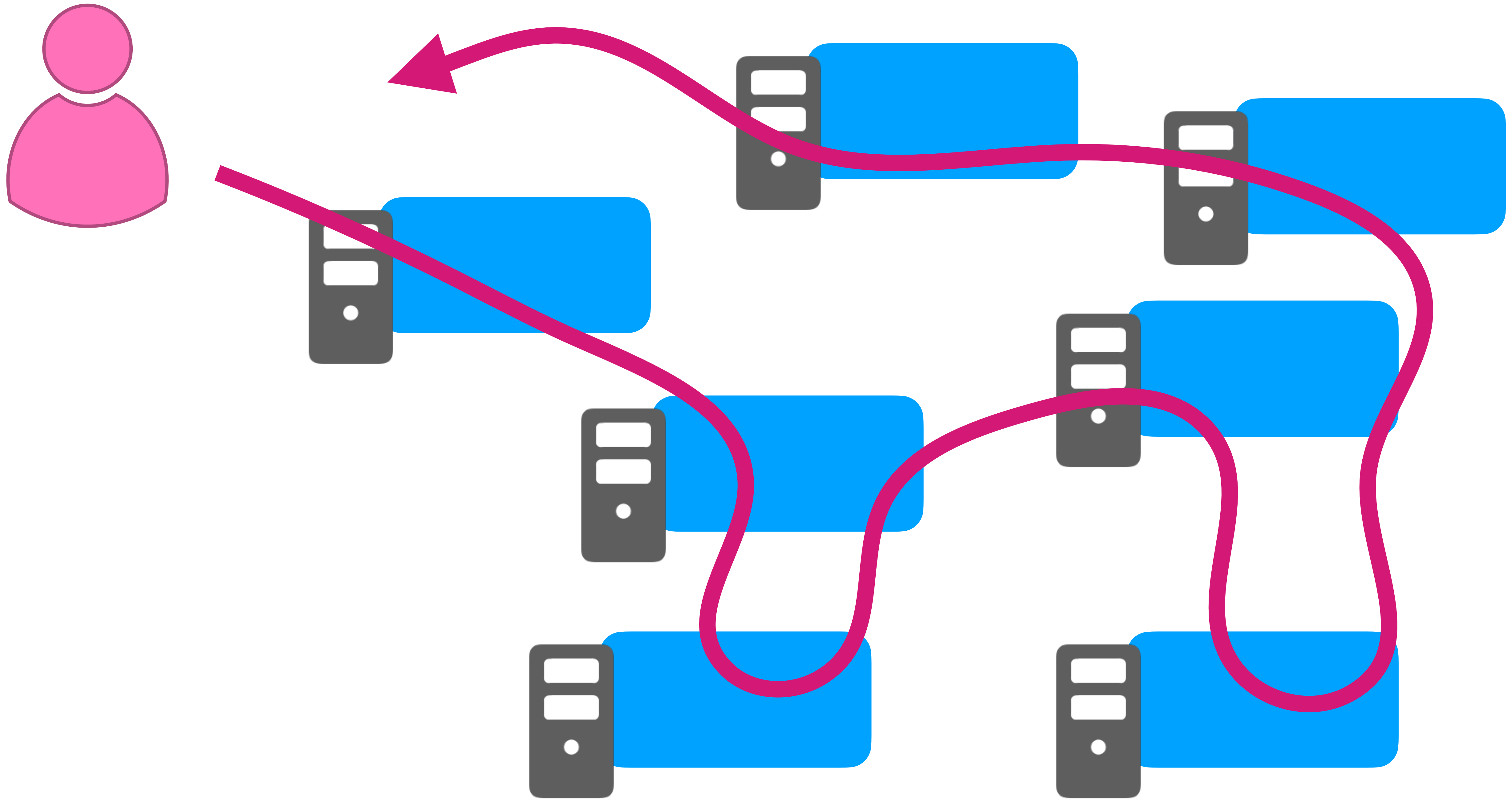
MAX PLANCK INSTITUTE FOR SOFTWARE SYSTEMS

Distributed Applications





Distributed Applications



Hard to understanding system behaviors

- End-to-end behavior can be affected by any component

Even hard when troubleshooting

- Symptoms and root causes can be far apart

Troubleshooting Edge-Cases

Infrequent 'rare' requests

Symptoms of a problem:
Erroneous responses
Tail latency
Uncommon request attributes

Why did we see these symptoms?
What was the root cause?

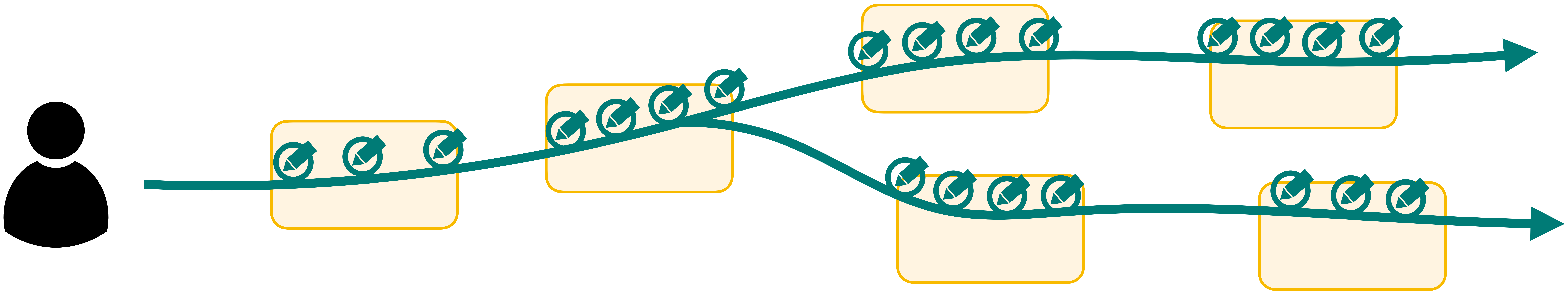
Troubleshooting requires execution details

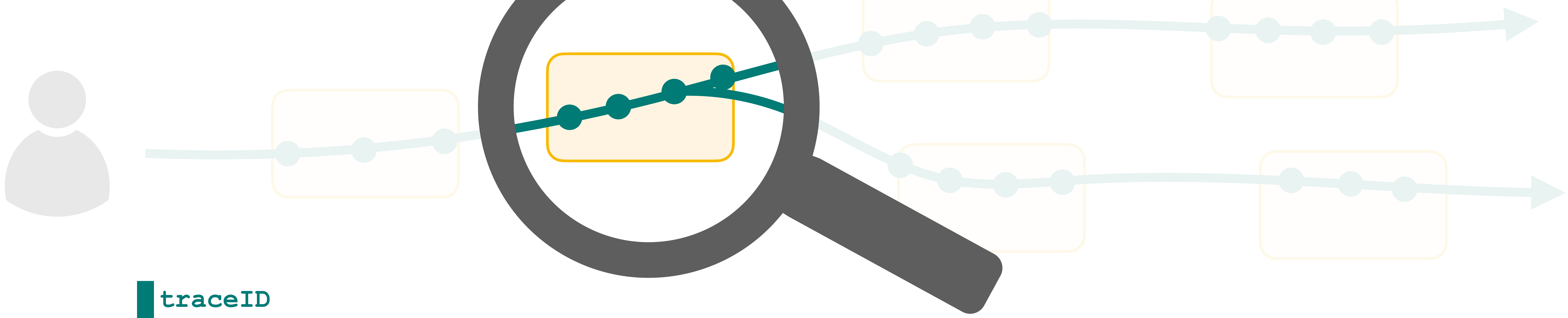
Distributed Tracing

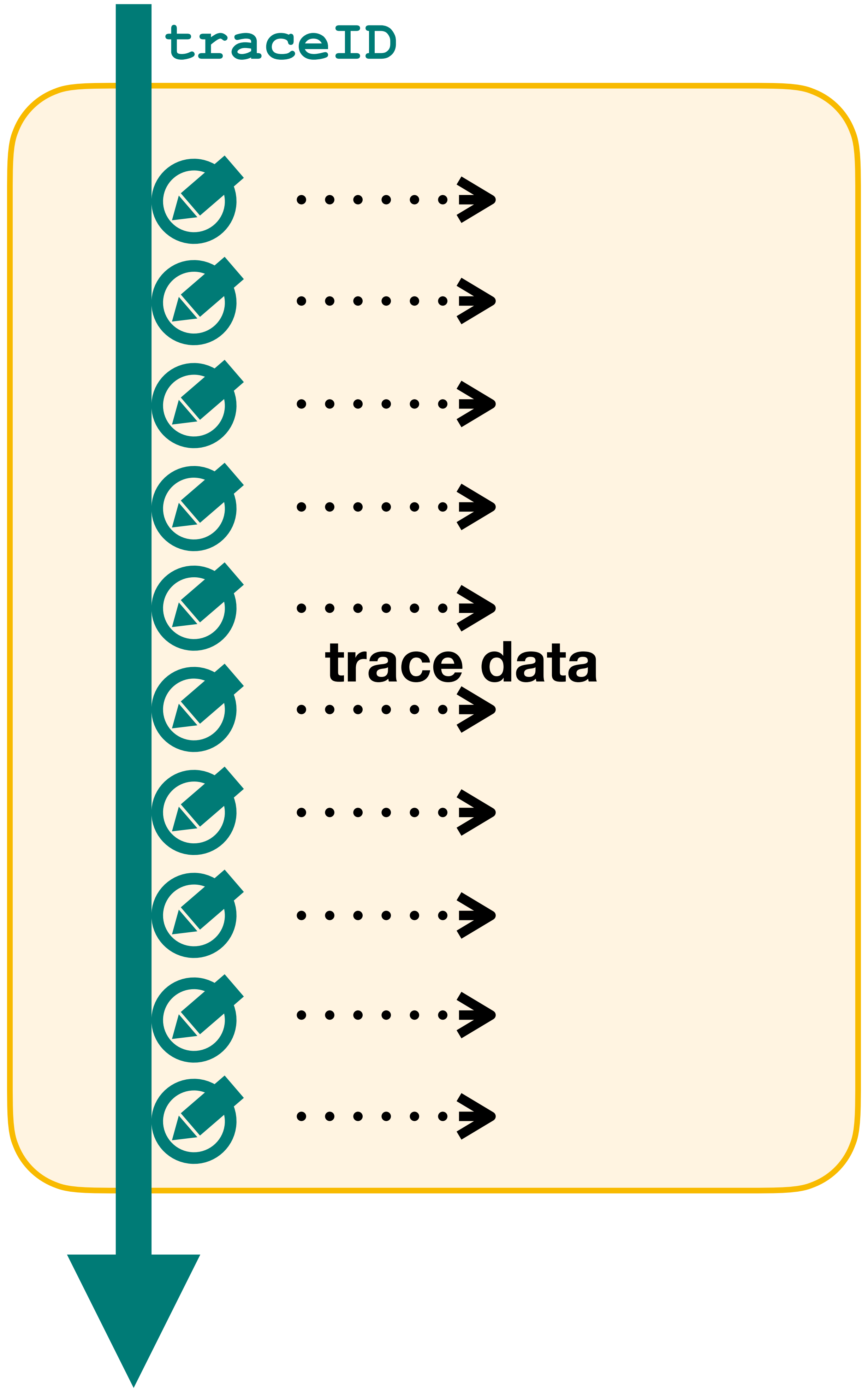
Recording of executions across all components

Trace events: timing, operations, messages, attributes

End-to-end requests show where the request went, and what it did

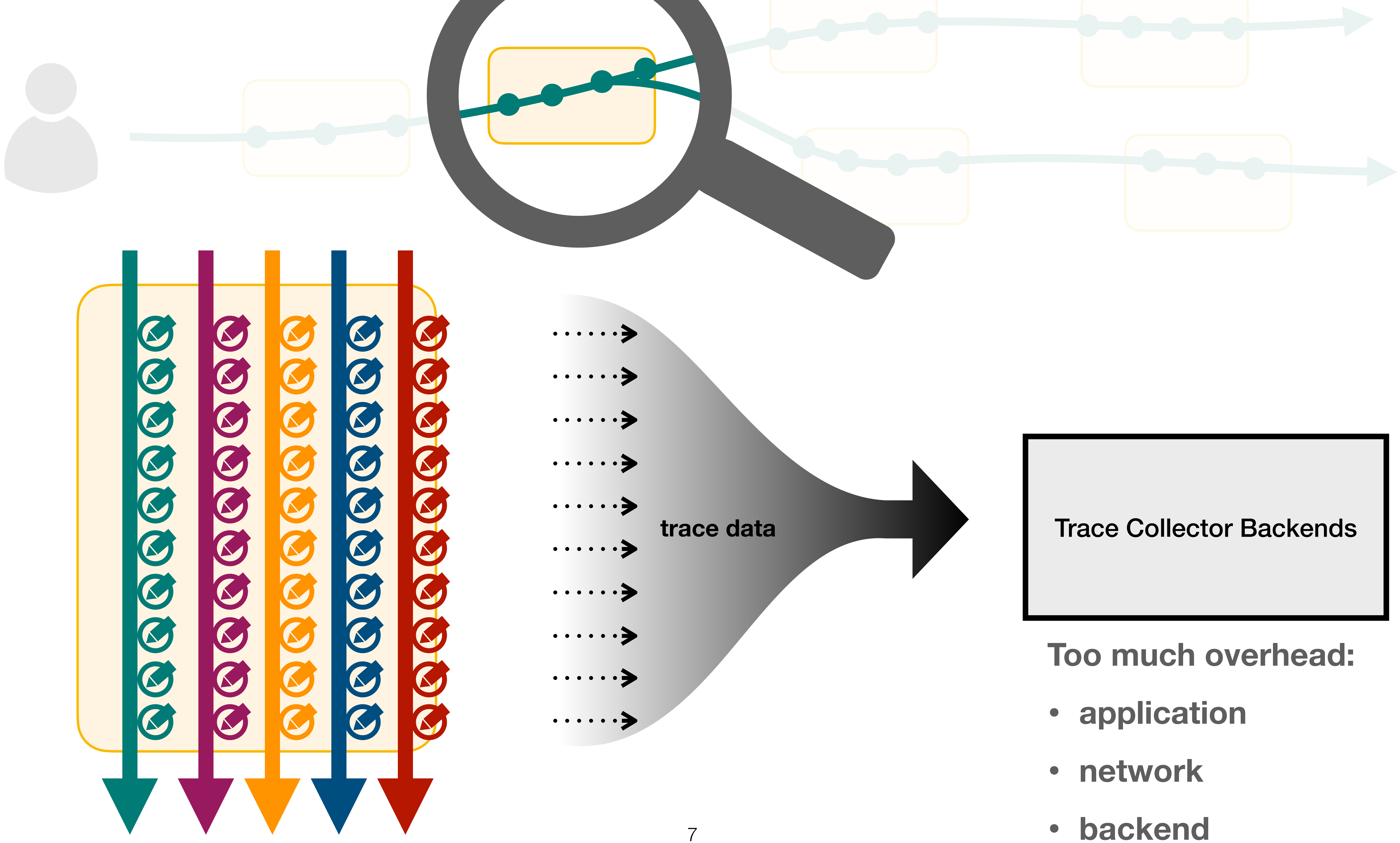






Google: debug-level logging

Facebook: up to 10MB for home-timeline trace

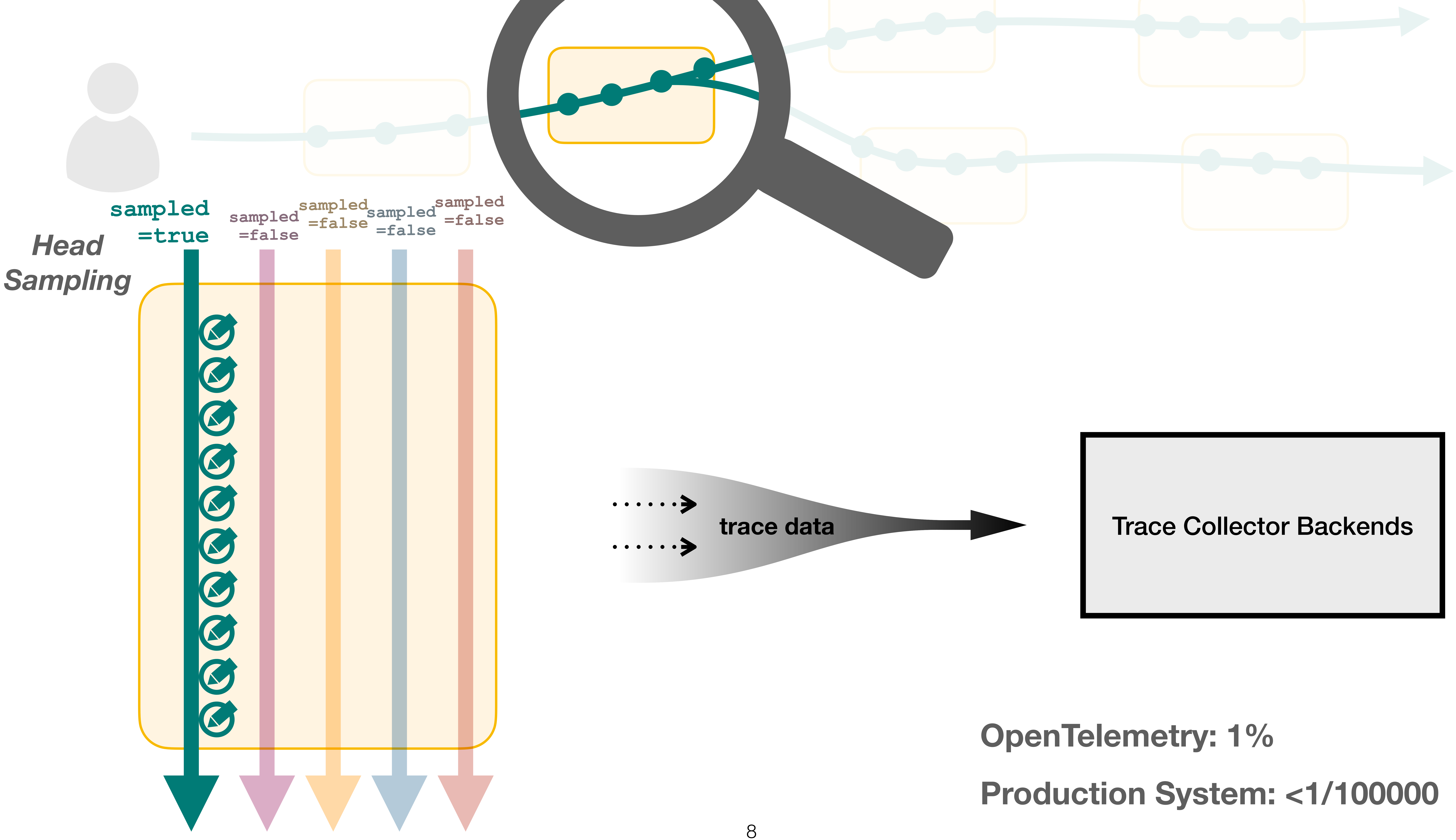


trace data

Trace Collector Backends

Too much overhead:

- **application**
- **network**
- **backend**



Head Sampling

sampled =true

sampled =false

sampled =false

sampled =false

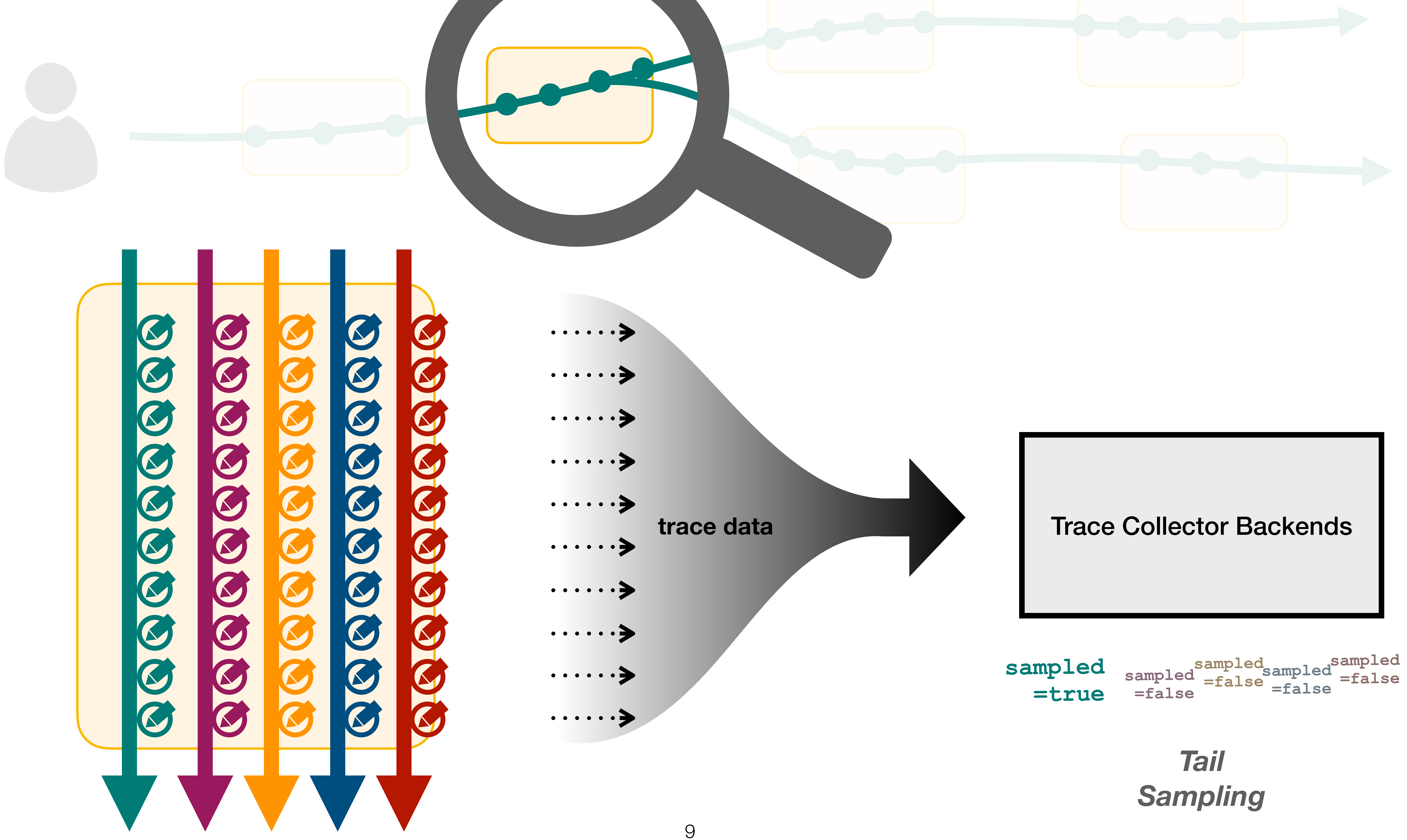
sampled =false

trace data

Trace Collector Backends

OpenTelemetry: 1%

Production System: <1/100000



trace data

Trace Collector Backends

sampled =true

sampled =false

sampled =false

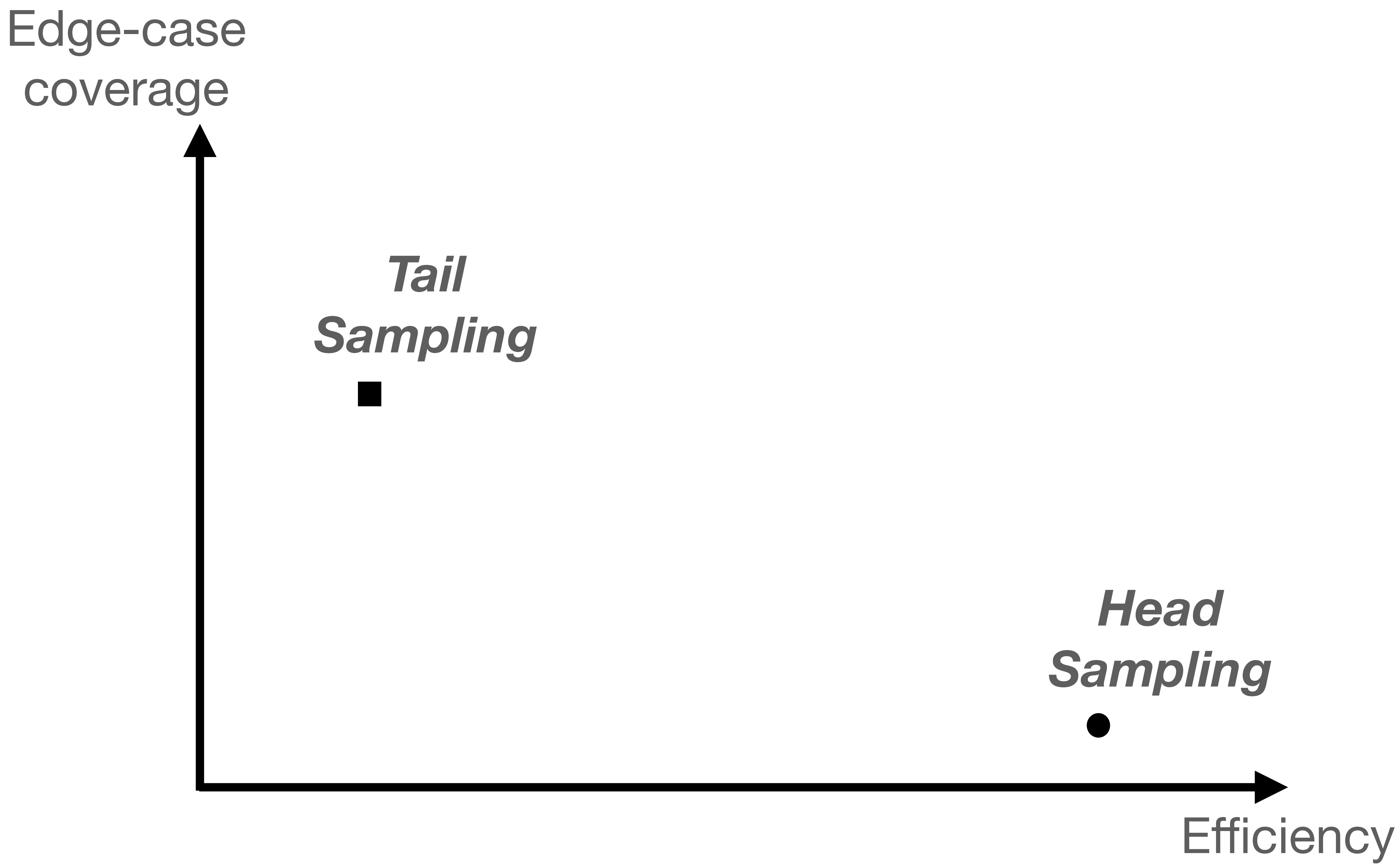
sampled =false

sampled =false

Tail Sampling

Sampling vs. Edge-Cases

- Edge-cases are rare
- We don't know edge-cases before they happen
- But we need to trace events before the symptoms



Today, edge-case trace data availability relies on luck

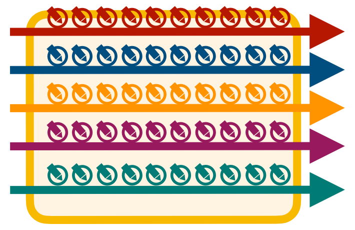
Observations

1. **Data generation is cheap**

2. **Edge-case trace data is a small set**

3. **Symptoms can be programmatically detected**

Retroactive Sampling



- Trace every request, leave data in memory, ingest later
  - Trace data of a request is scattered across machines

Retroactive Sampling

- Trace every request, leave data in memory, ingest later
- Each component detects symptoms, and fires **triggers**
    - Any time during or shortly after request

Retroactive Sampling

- Trace every request, leave data in memory, ingest later
- Programmatically detect symptoms, and fire **triggers**
- Requests propagate and deposit **breadcrumbs**
    - Inform all relevant machines of a triggered trace

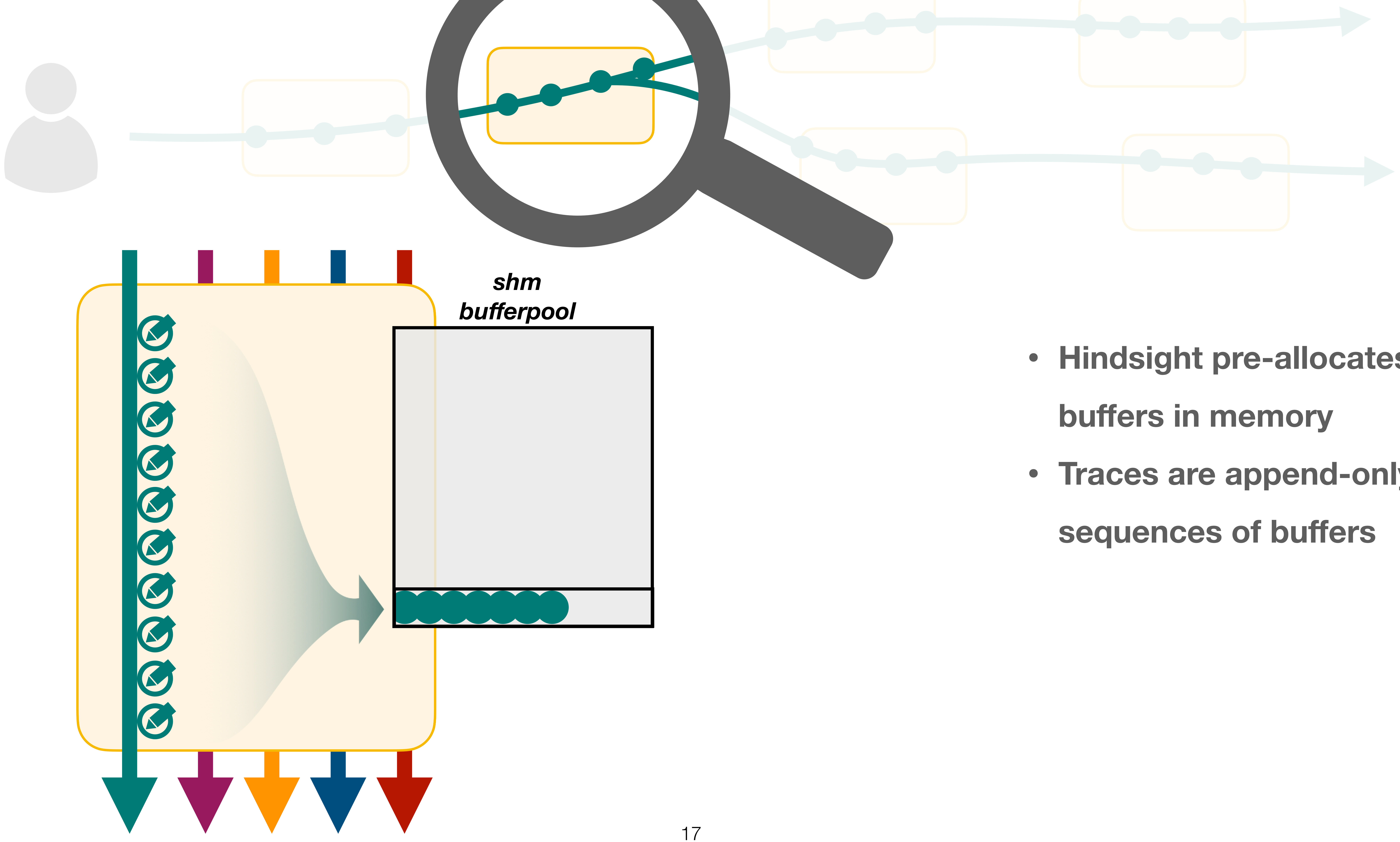
Retroactive Sampling

- Trace every request, leave data in memory, ingest later
- Programmatically detect symptoms, and fire **triggers**
- Requests propagate and deposit **breadcrumbs**
- Collect triggered request in time
  - If not triggered, old data is overwritten with new data

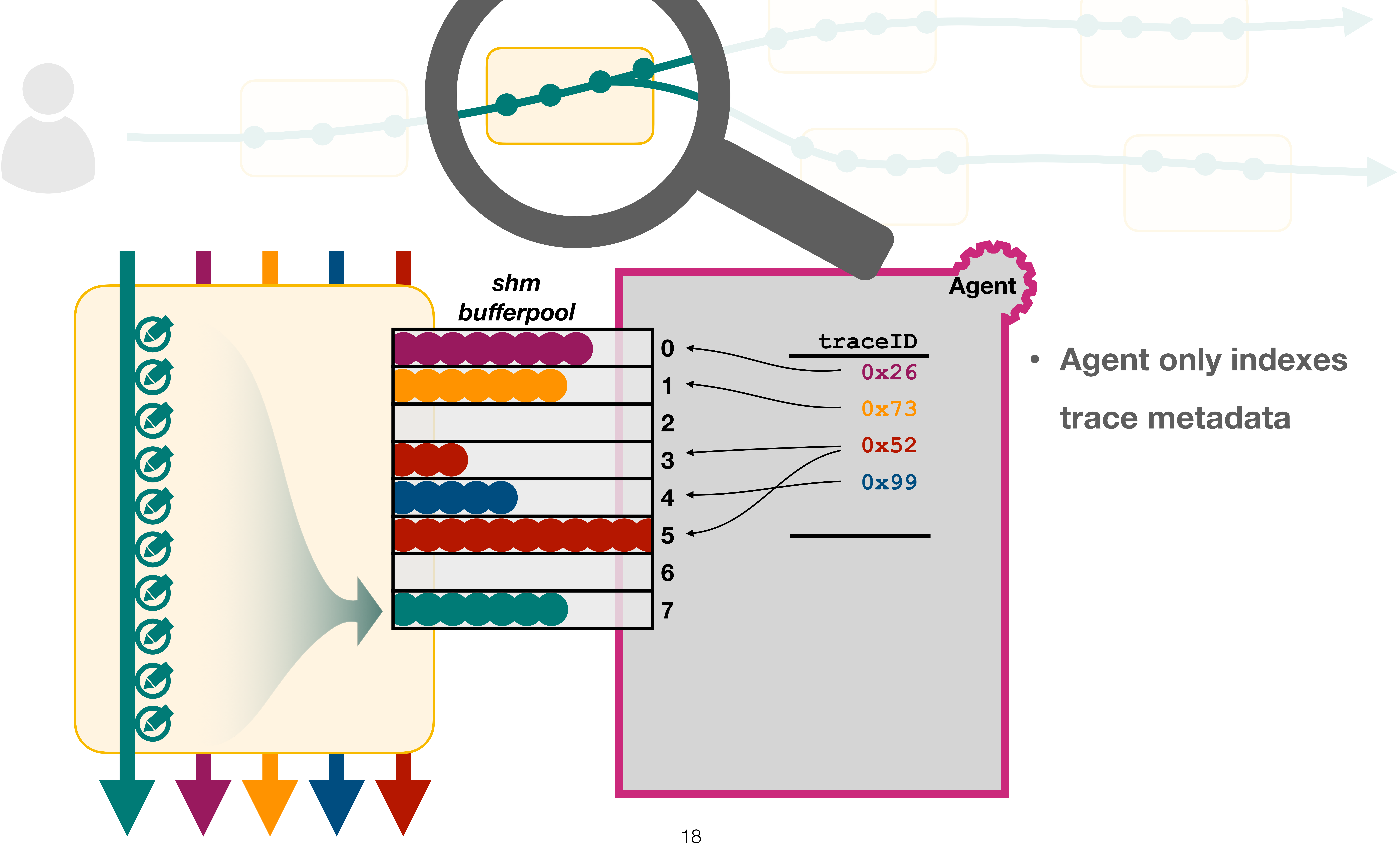
Hindsight

- Hindsight is designed to trace 100% requests ✓
- Split control and data plane to manage large data volume ✓
- AutoTrigger library to support symptom detection
- Scalable breadcrumb mechanism for triggered traces

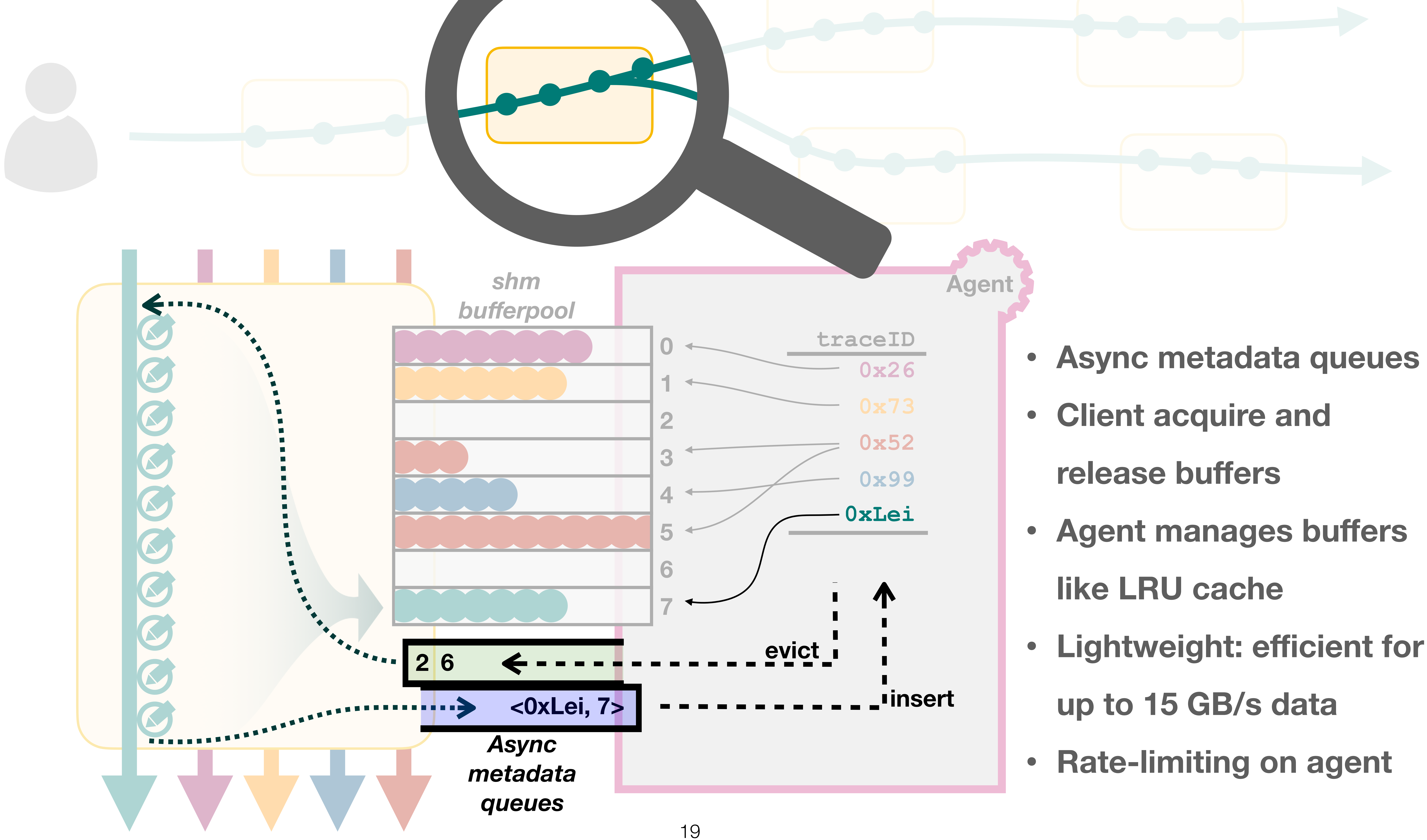
📄



- Hindsight pre-allocates buffers in memory
- Traces are append-only sequences of buffers



- Agent only indexes trace metadata



- Async metadata queues
- Client acquire and release buffers
- Agent manages buffers like LRU cache
- Lightweight: efficient for up to 15 GB/s data
- Rate-limiting on agent

Evaluation

- Benchmarks: DeathStarBench, HDFS, MicroBricks
  - A configurable RPC benchmark with 93 service applications
- Baseline: OpenTelemetry (with Jaeger), with no-tracing, or head/tail sampling
- Evaluation:

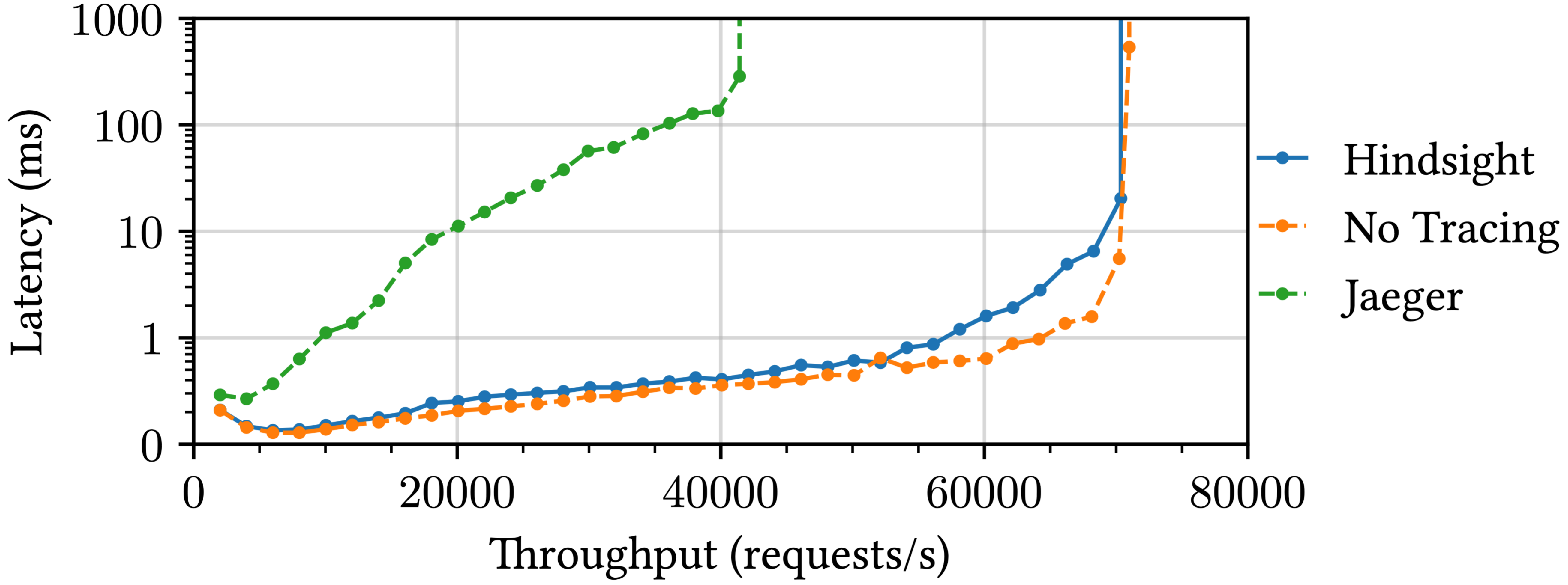
✓ Overhead

📄 Scalability

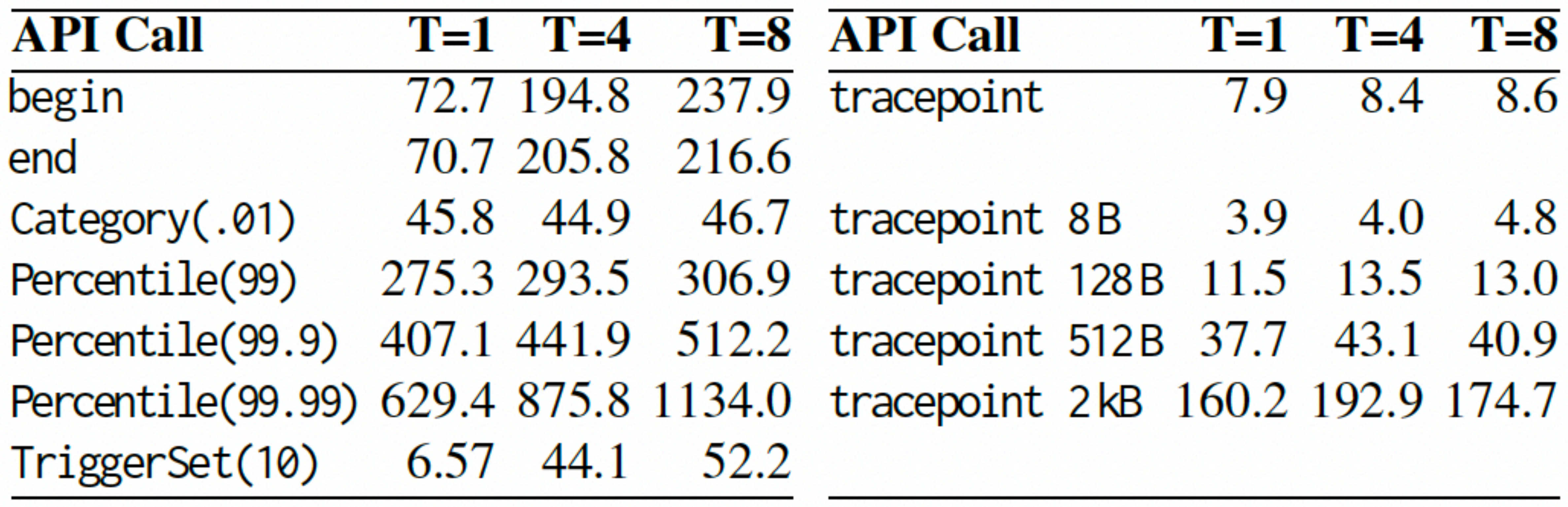
📄 Real-world use cases

Overhead with 100% Requests

- Hindsight’s data generation adds minimum end-to-end application overheads
  - Tail sampling has 10-100x latency with 50% reduced peak throughput
- Nanosecond-level tracing APIs



(Nanoseconds)

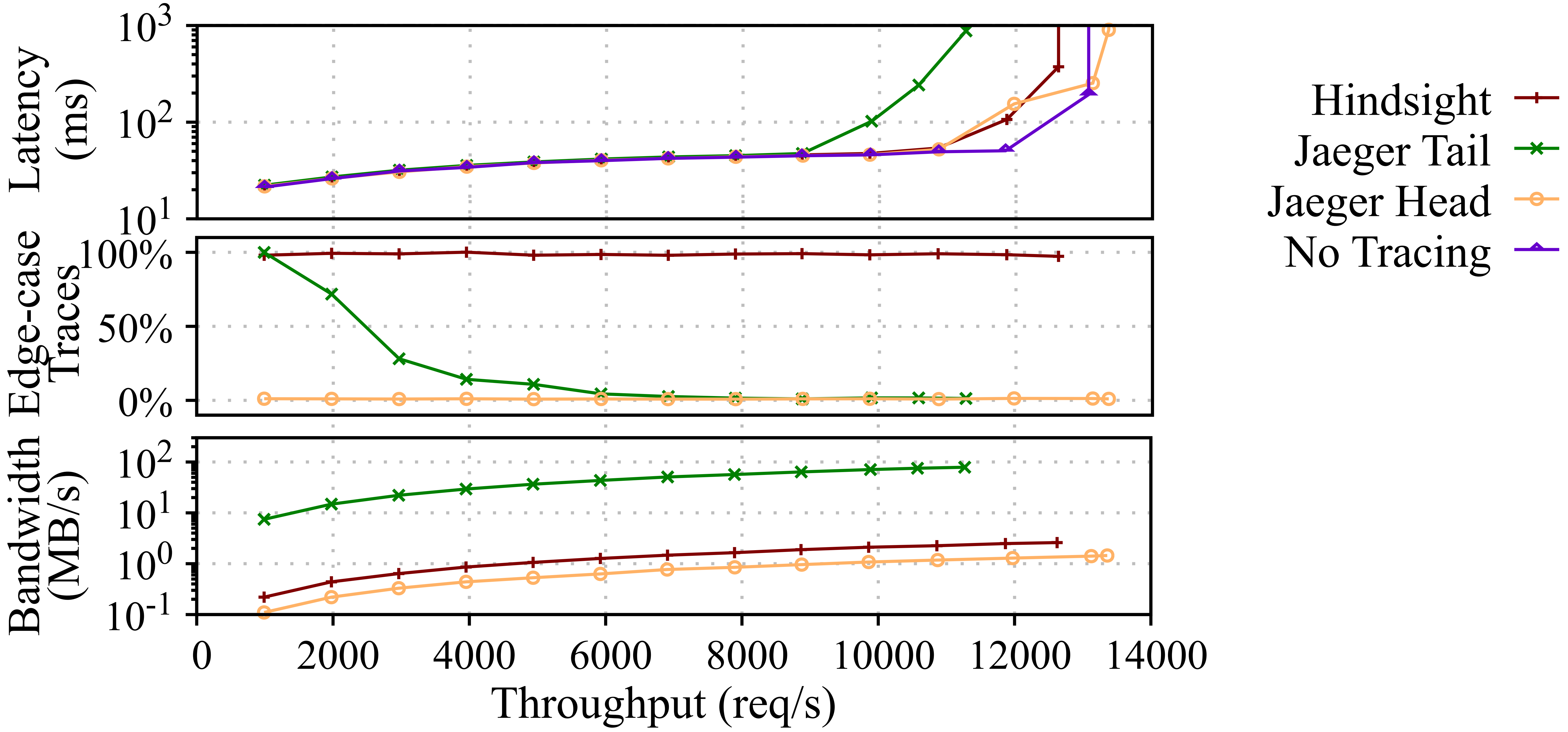


API Call	T=1	T=4	T=8
begin	72.7	194.8	237.9
end	70.7	205.8	216.6
Category(.01)	45.8	44.9	46.7
Percentile(99)	275.3	293.5	306.9
Percentile(99.9)	407.1	441.9	512.2
Percentile(99.99)	629.4	875.8	1134.0
TriggerSet(10)	6.57	44.1	52.2

API Call	T=1	T=4	T=8
tracepoint	7.9	8.4	8.6
tracepoint 8B	3.9	4.0	4.8
tracepoint 128B	11.5	13.5	13.0
tracepoint 512B	37.7	43.1	40.9
tracepoint 2kB	160.2	192.9	174.7

Overhead vs. Edge-Cases

- 93 application microservices
- Hindsight: <3.5% peak throughput reduction than no-tracing
- Hindsight captures almost all edge-cases, with low tracing bandwidth



Conclusion

*Today's tracing systems rely on **luck***

for edge-cases

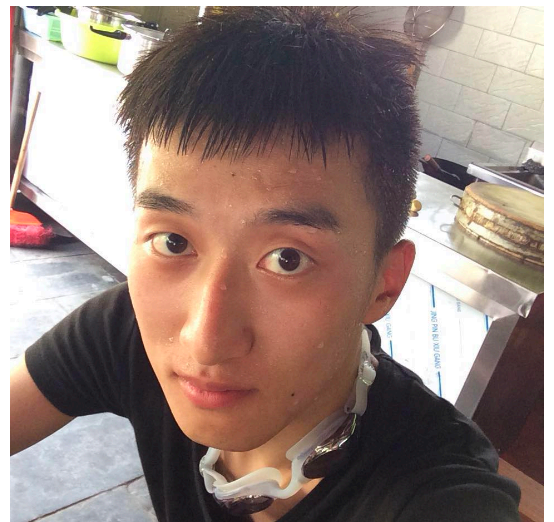
Hindsight: lightweight always-on tracing system

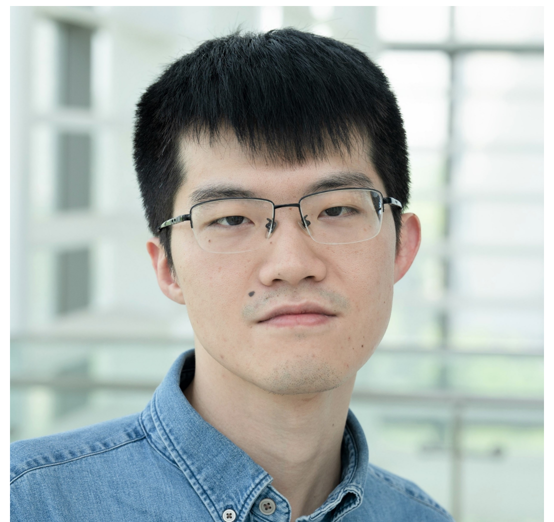
Solution: ***Retroactive Sampling***

- Trace every request, ingest later
- Programmatically detect symptoms and fire **triggers**
- Requests propagate and deporit **breadcrumbs**
- Collect triggered request in time

Hindsight: https://gitlab.mpi-sws.org/cld/tracing/hindsight

Microbricks: https://gitlab.mpi-sws.org/cld/tracing/hindsight-grpc





Thanks!