11th Symposium on Software Performance 2020

Toward Efficient Scalability Benchmarking of Event-Driven Microservice Architectures at Large Scale

Sören Henning and Wilhelm Hasselbring

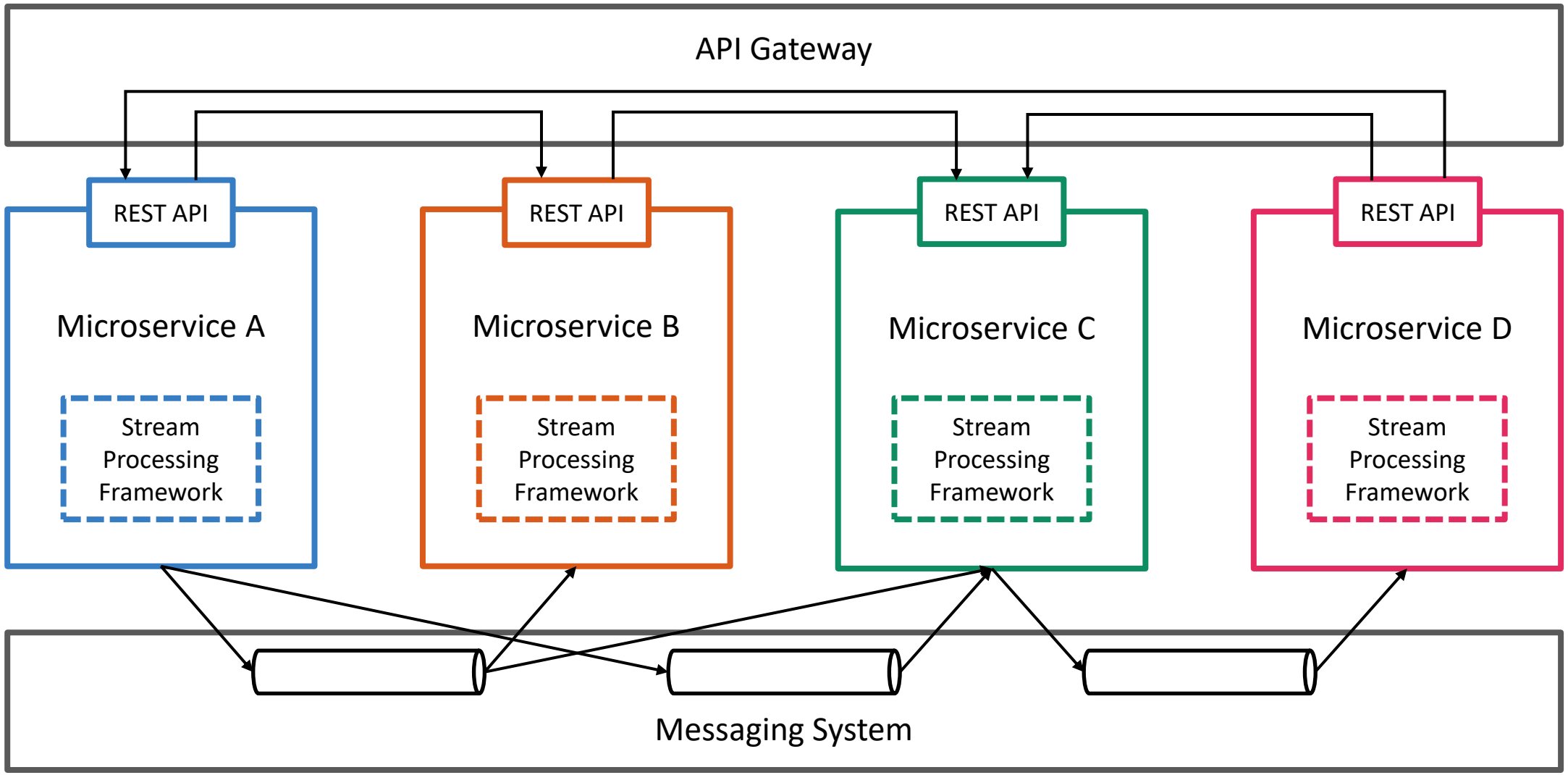
C | A | U
Kiel University
Christian-Albrechts-Universität zu Kiel

SPONSORED BY THE

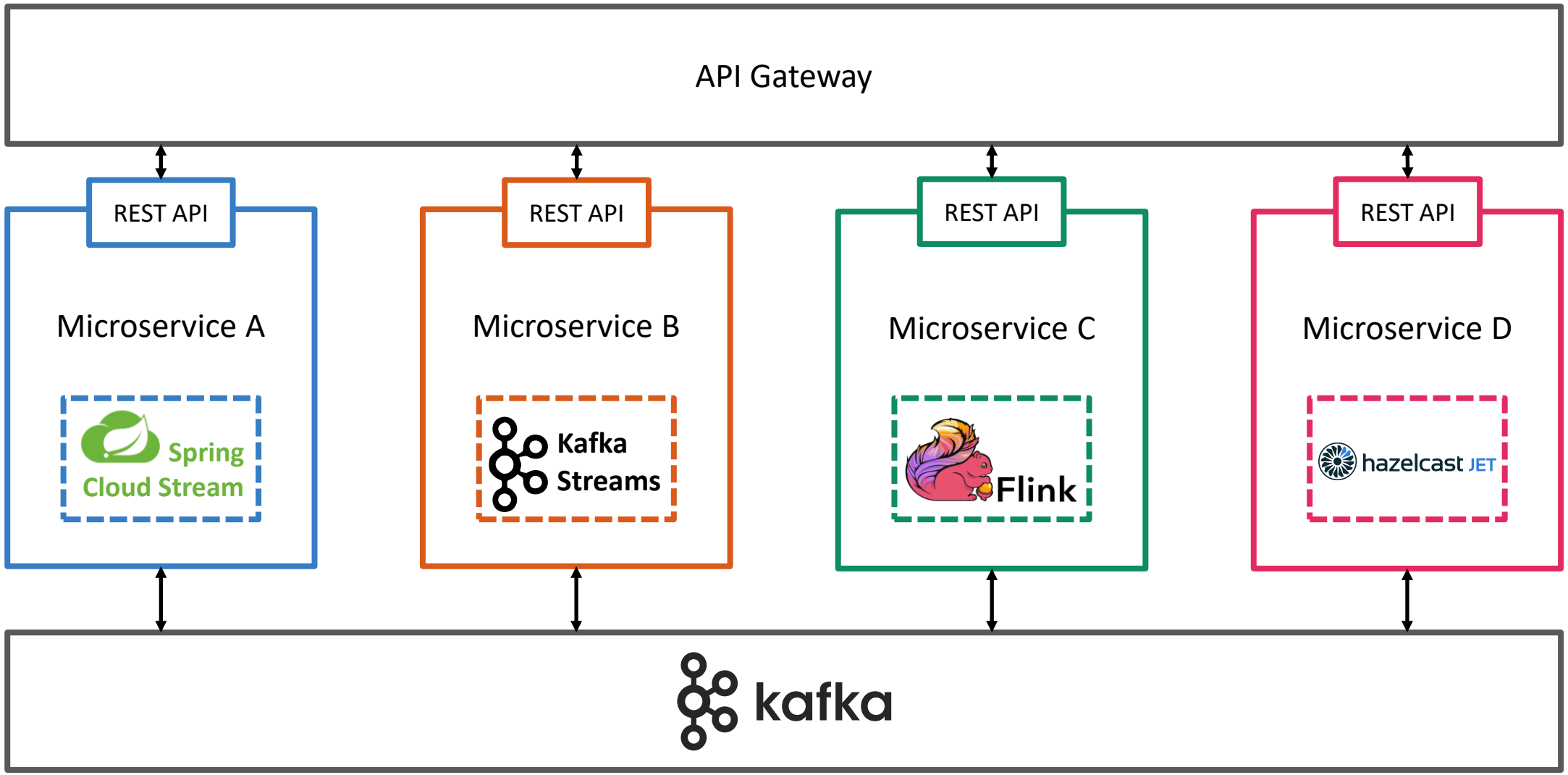
Federal Ministry
of Education
and Research

contract no. 01IS17084B

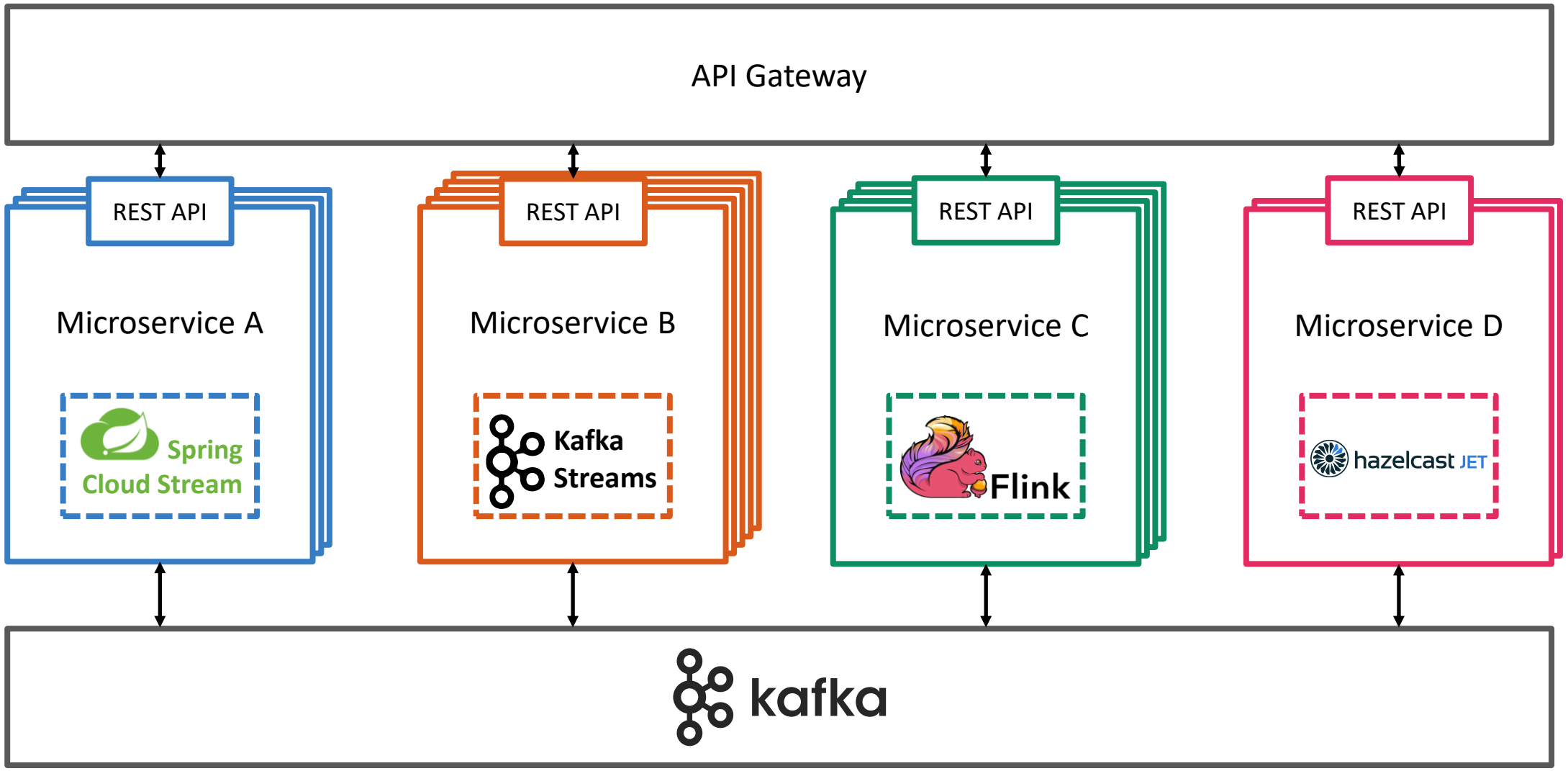
Event-Driven Microservice Architectures



Event-Driven Microservice Architectures



Event-Driven Microservice Architectures



Part 1:

The Theodolite Scalability Benchmarking Method

Theodolite's Scalability Metric

> *Scalability is the ability of the system to sustain increasing workloads by making use of additional resources […].*
>
> Herbst et al. 2013

Theodolite's Scalability Metric

> *Scalability is the ability of the system to sustain increasing workloads by making use of additional resources […].*
>
> Herbst et al. 2013

Load Intensity to be increased

Service Level to be sustained

Resource Amounts to be added

Weber et al. 2014

Theodolite's Scalability Metric

> *Scalability is the ability of the system to sustain increasing workloads by making use of additional resources […].*

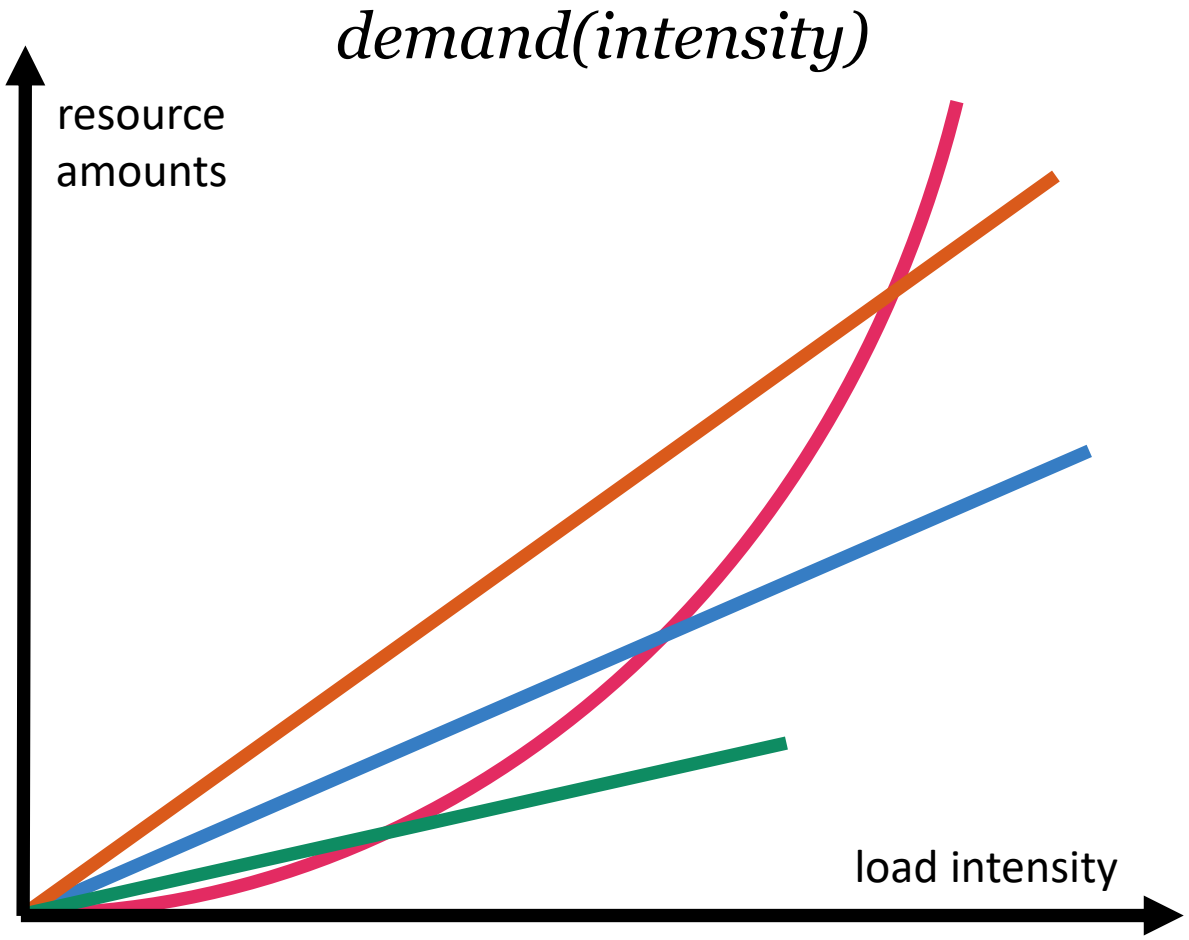
Herbst et al. 2013

Load Intensity to be increased

Service Level to be sustained

Resource Amounts to be added

Weber et al. 2014



Herbst et al. 2015

Theodolite's Scalability Metric

> *Scalability is the ability of the system to sustain increasing workloads by making use of additional resources […].*

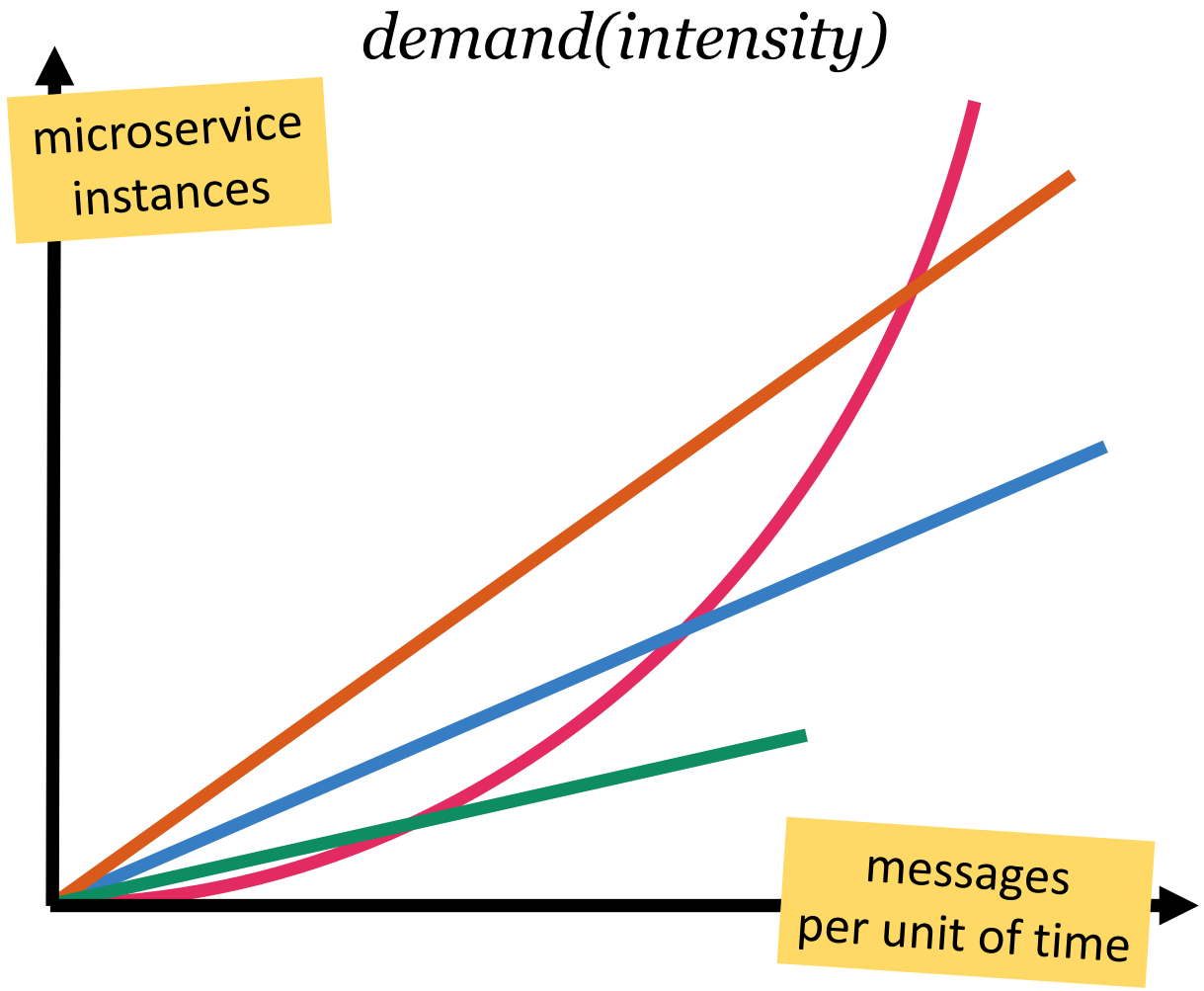
Herbst et al. 2013

Load Intensity to be increased

Service Level to be sustained

Resource Amounts to be added

Weber et al. 2014



Theodolite's Scalability Metric

> *Scalability is the ability of the system to sustain increasing workloads by making use of additional resources […].*

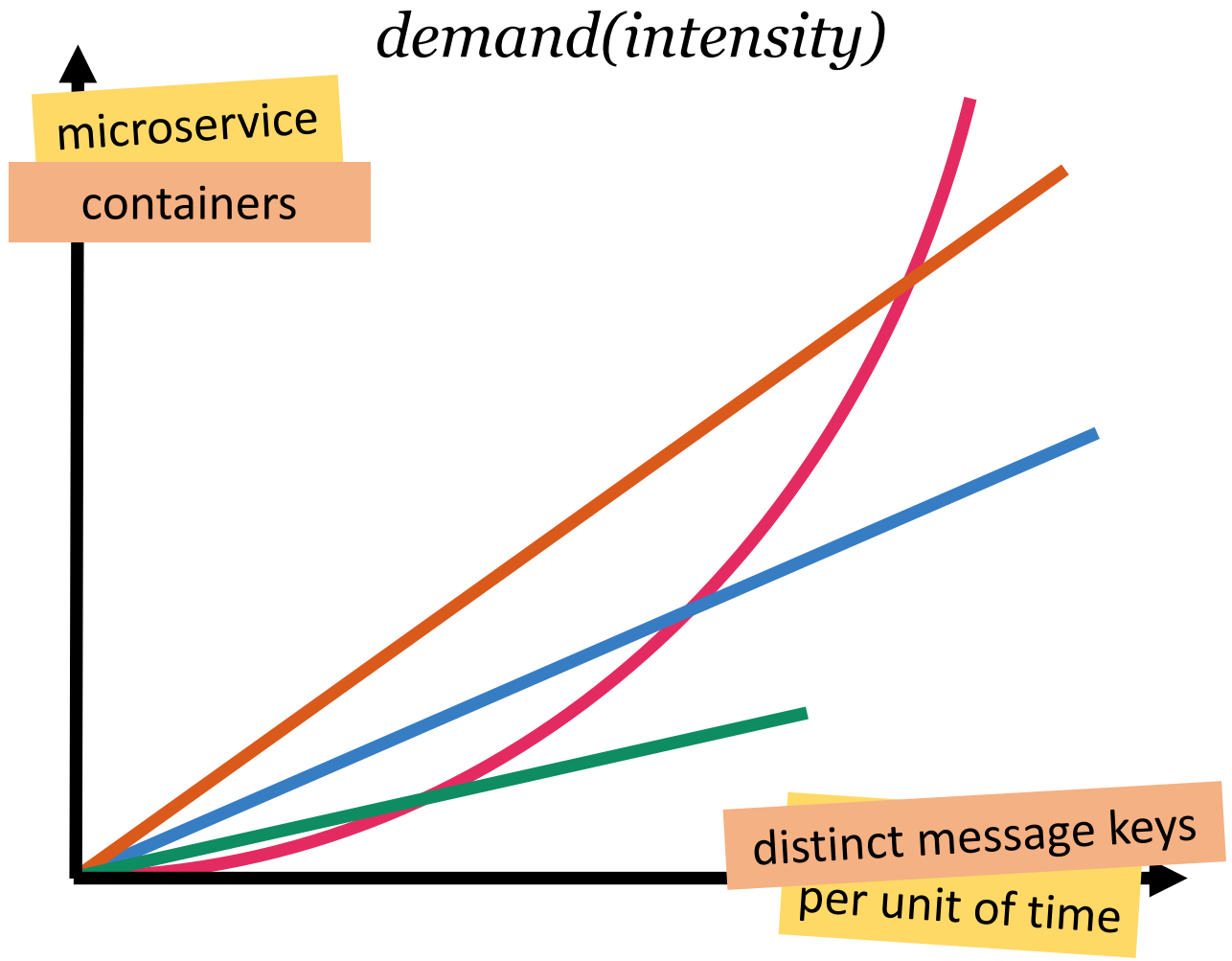
Herbst et al. 2013

Load Intensity to be increased

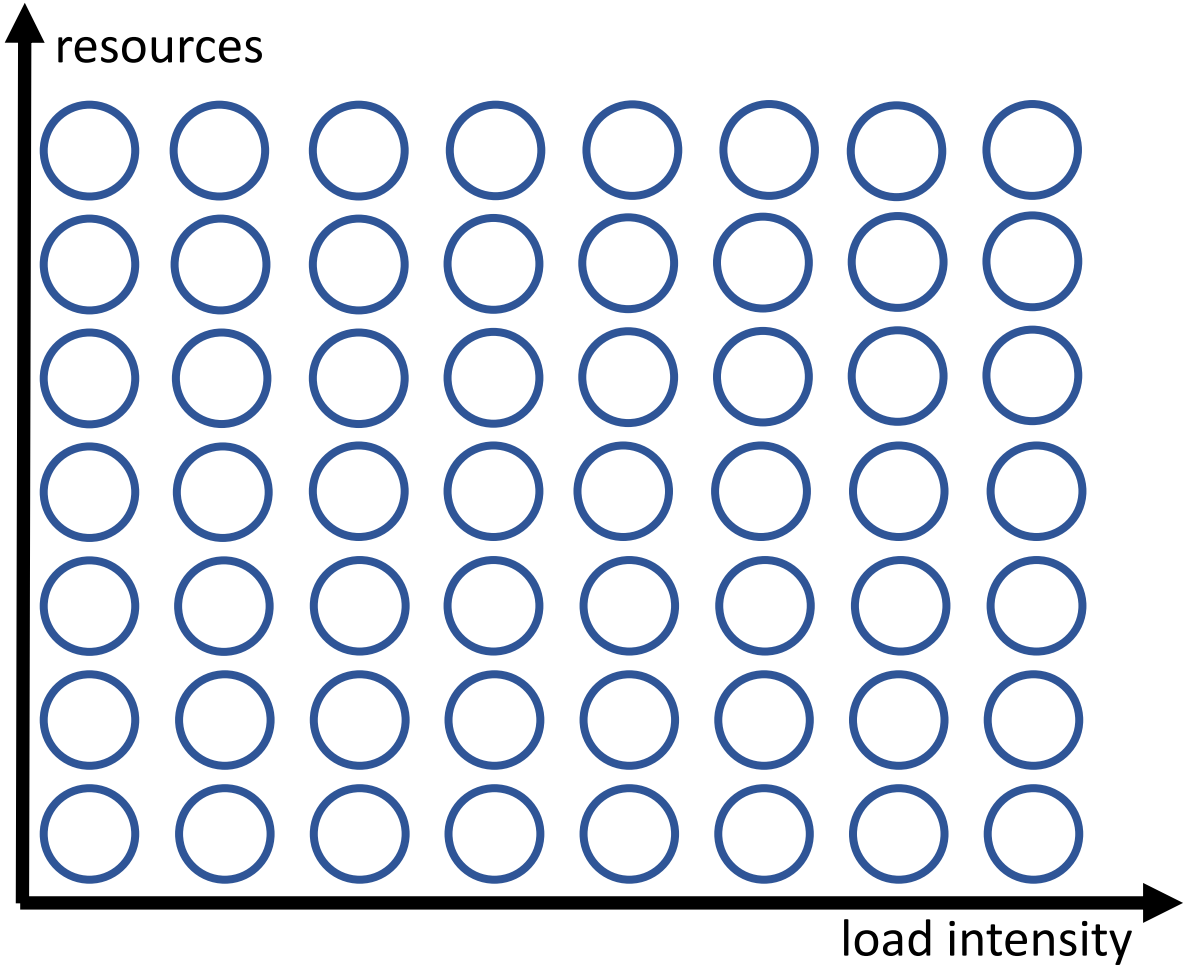
Service Level to be sustained

Resource Amounts to be added

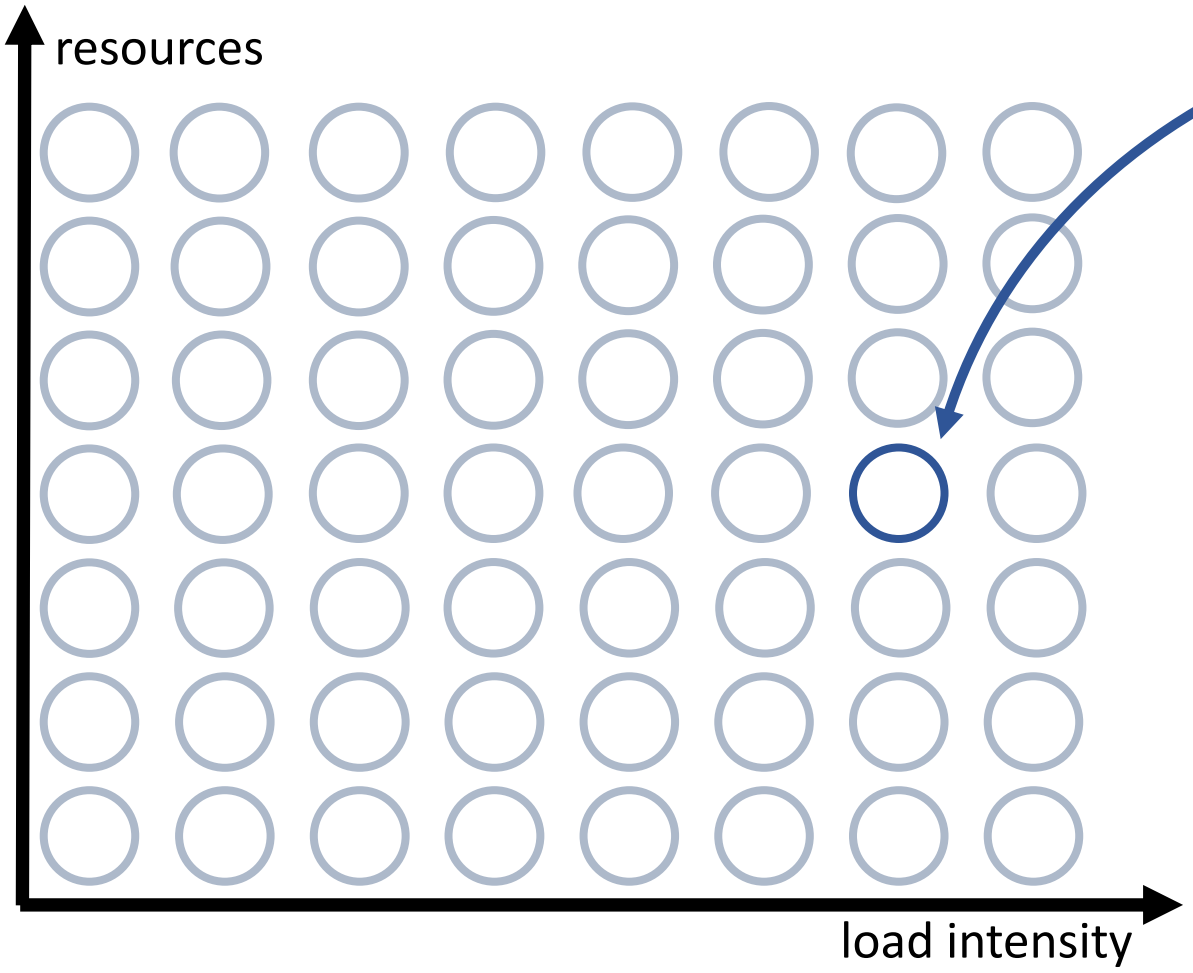
Weber et al. 2014



Theodolite’s Scalability Measurement Method



Theodolite's Scalability Measurement Method

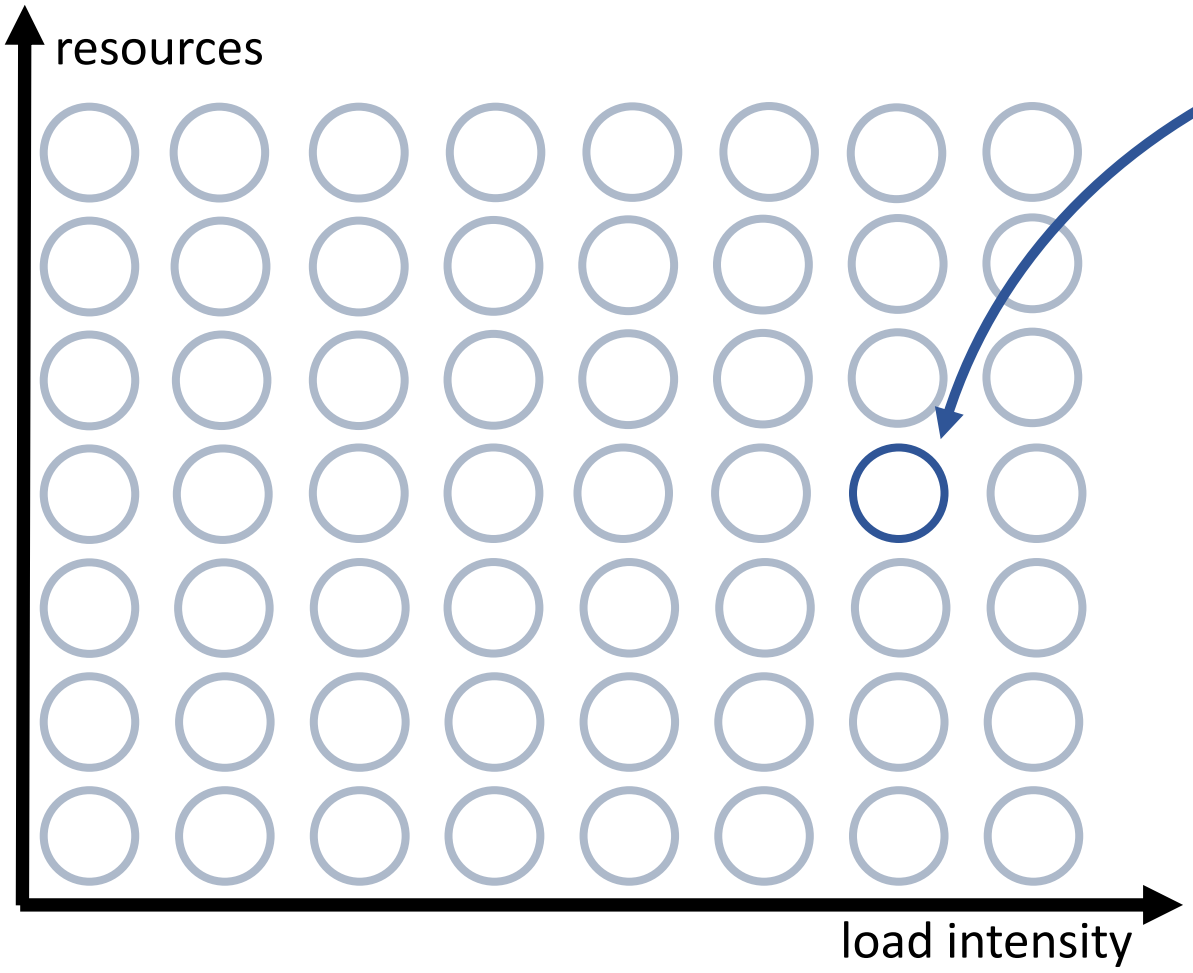


sufficient resources for load?

lag increase over time?

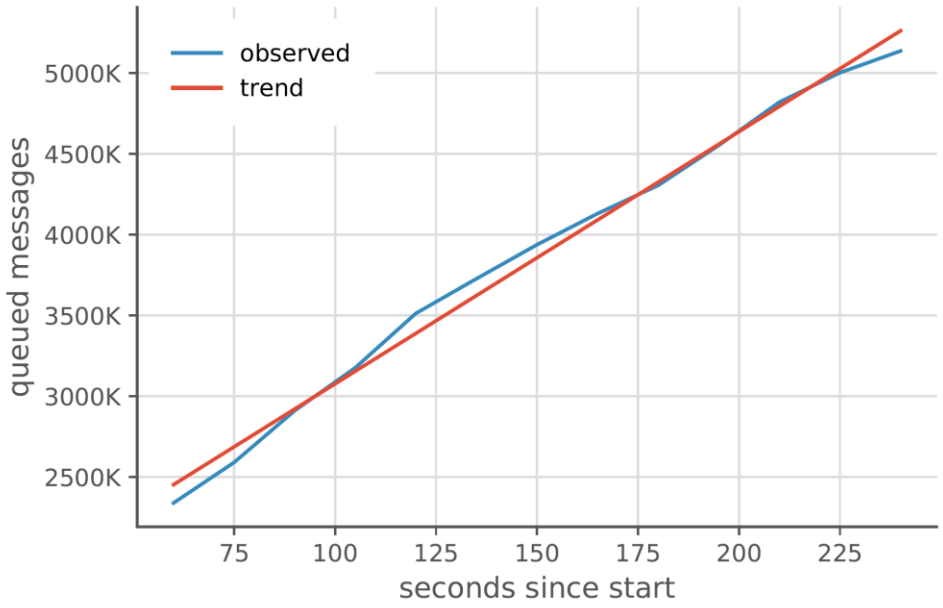
lag = queued messages

Theodolite's Scalability Measurement Method

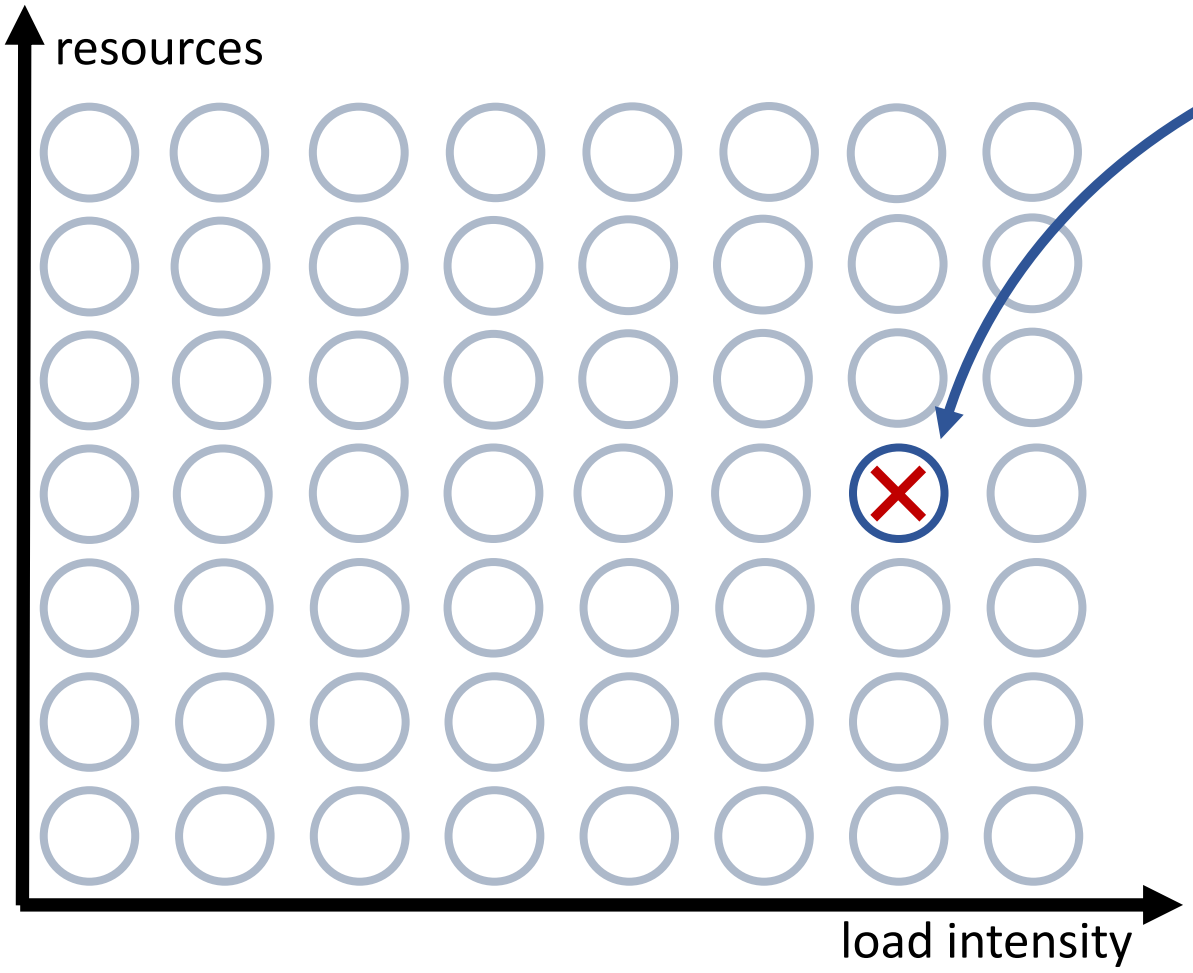


sufficient resources for load?

lag increase over time?

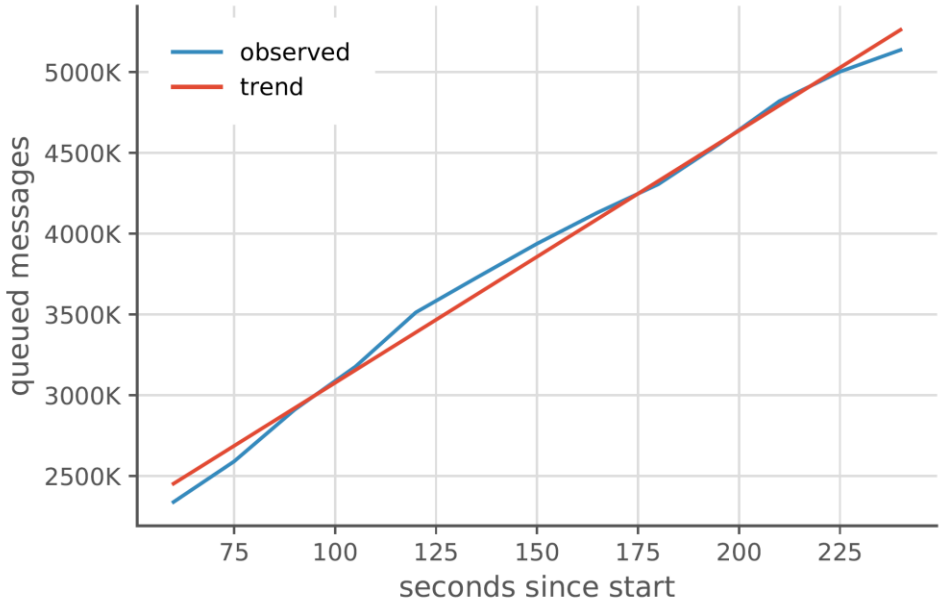


Theodolite's Scalability Measurement Method

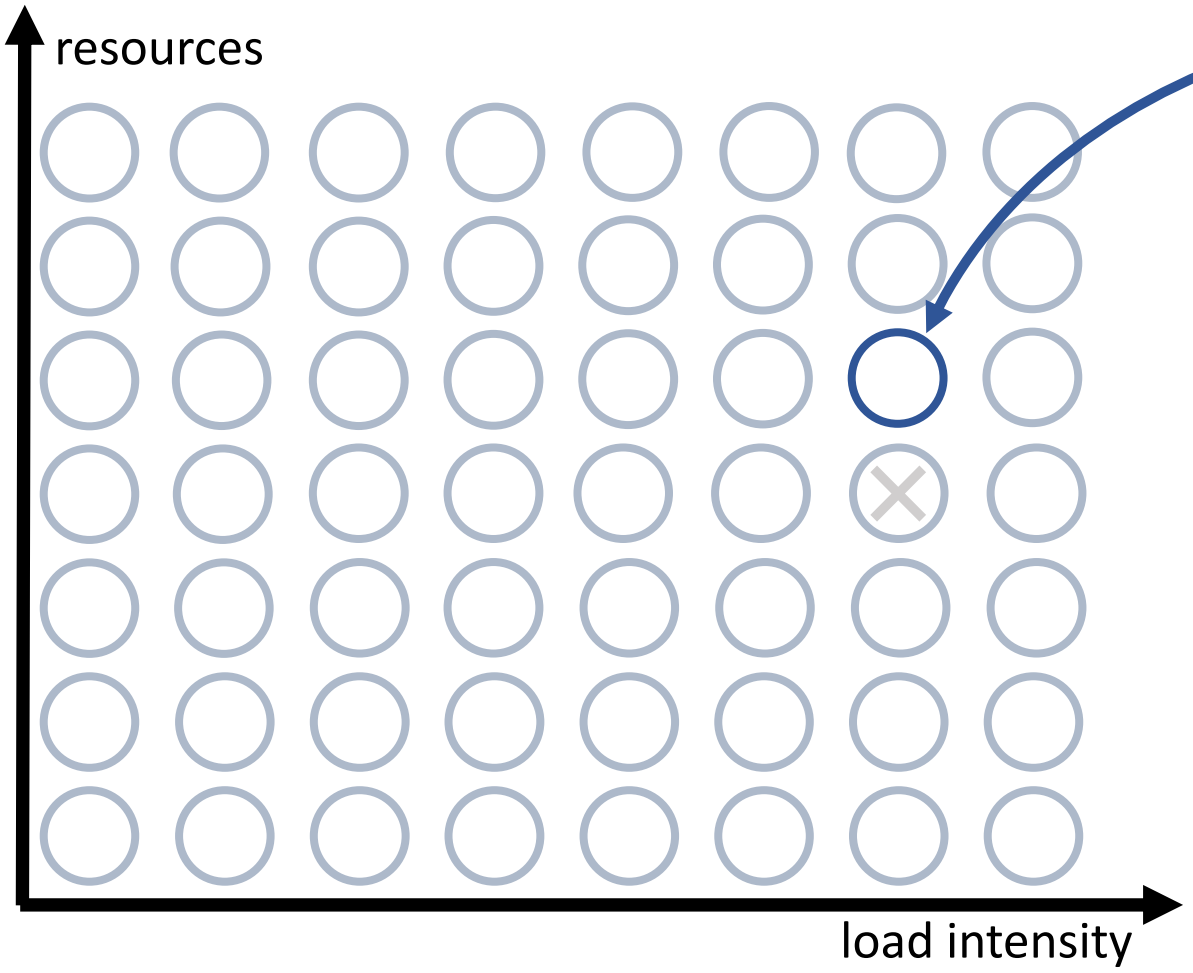


sufficient resources for load? **No!**

lag increase over time? **Yes!**

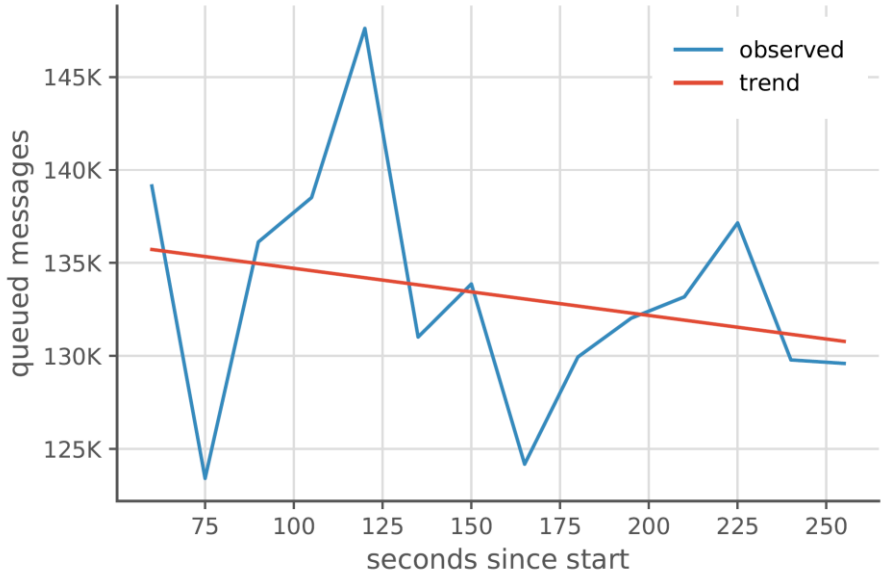


Theodolite's Scalability Measurement Method

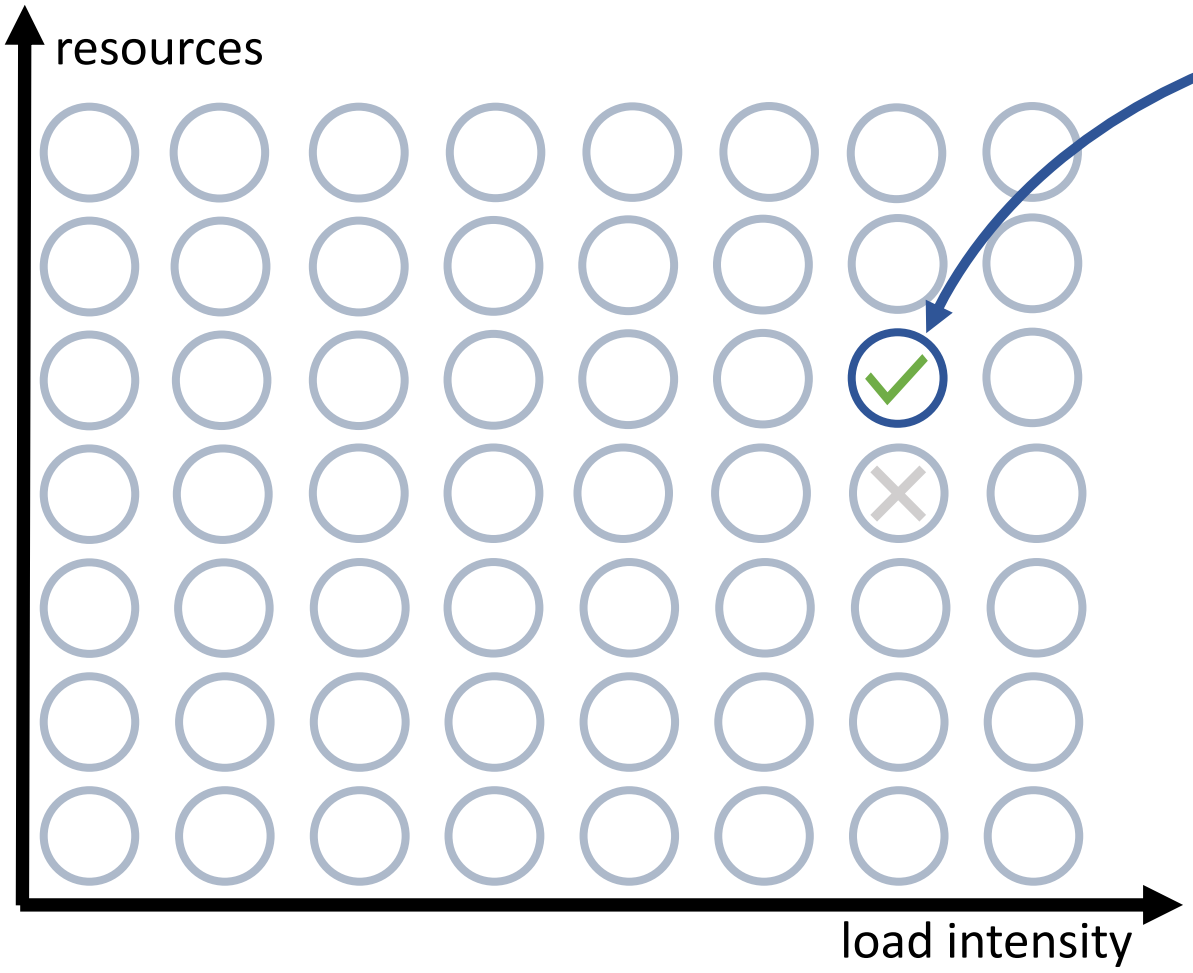


sufficient resources for load?

lag increase over time?

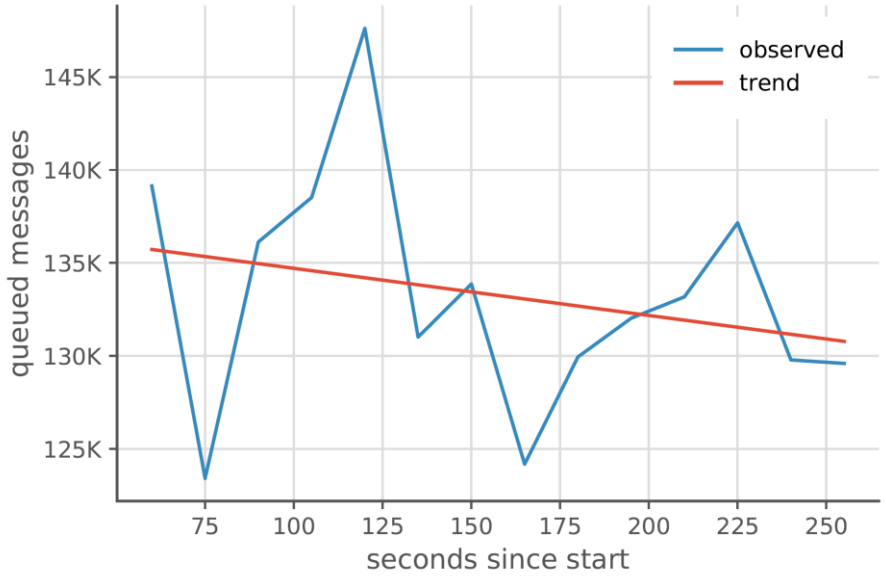


Theodolite's Scalability Measurement Method

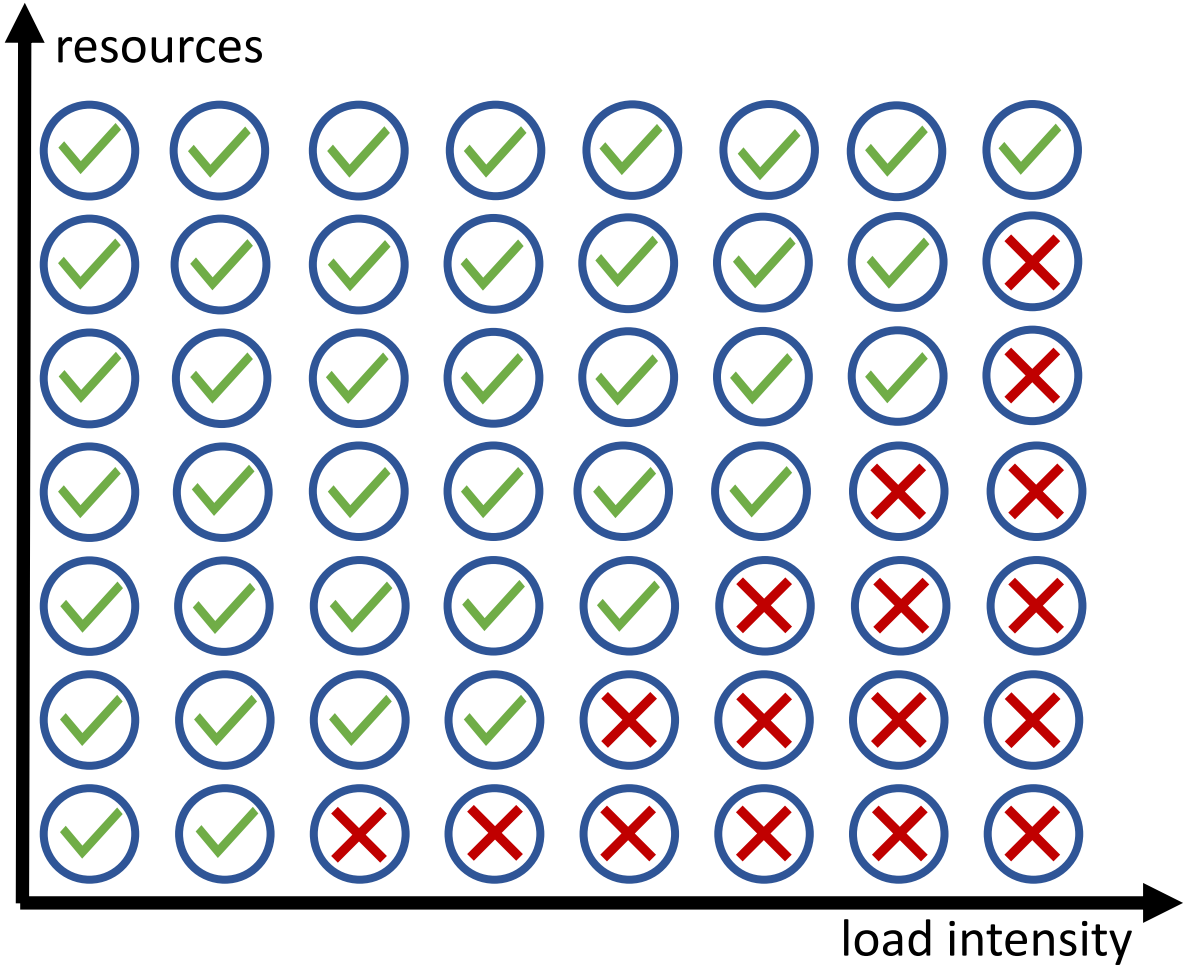


sufficient resources for load? Yes!

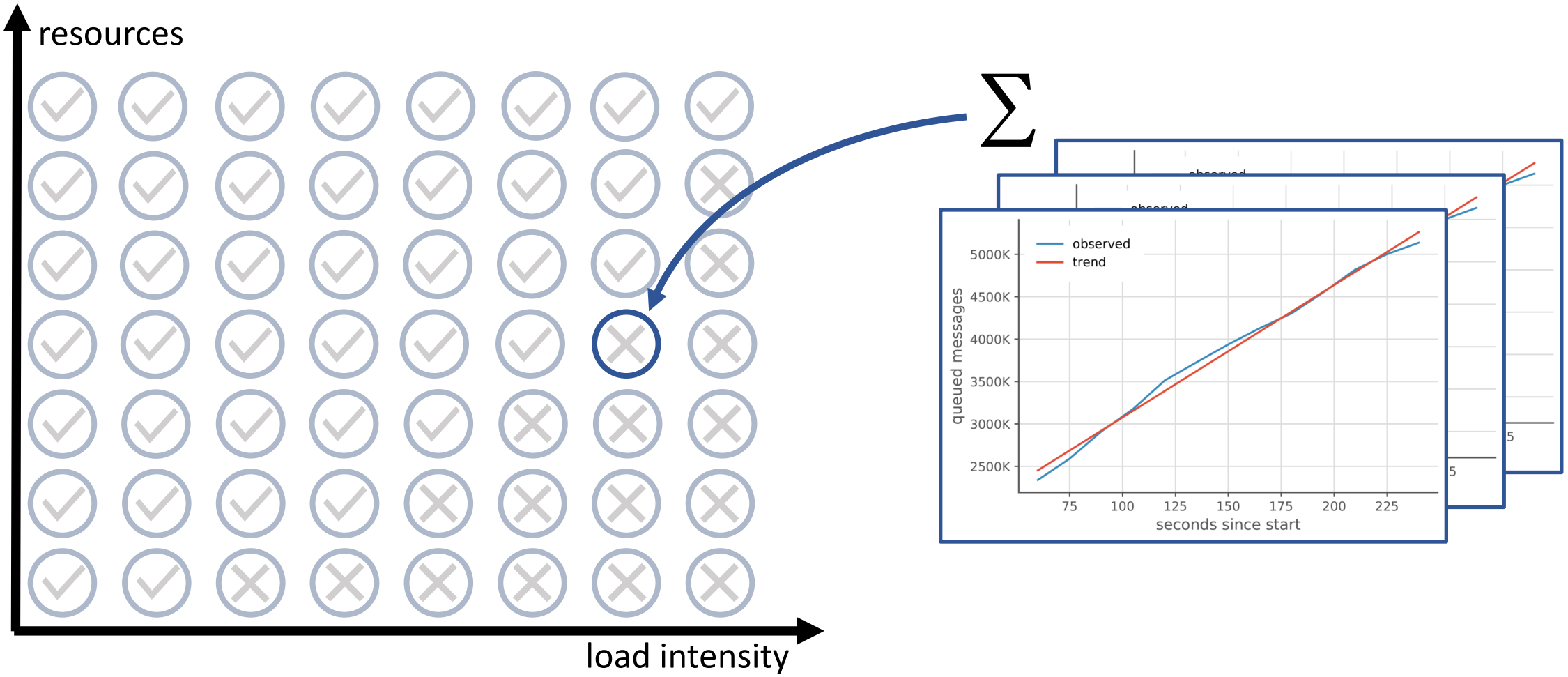
lag increase over time? No!



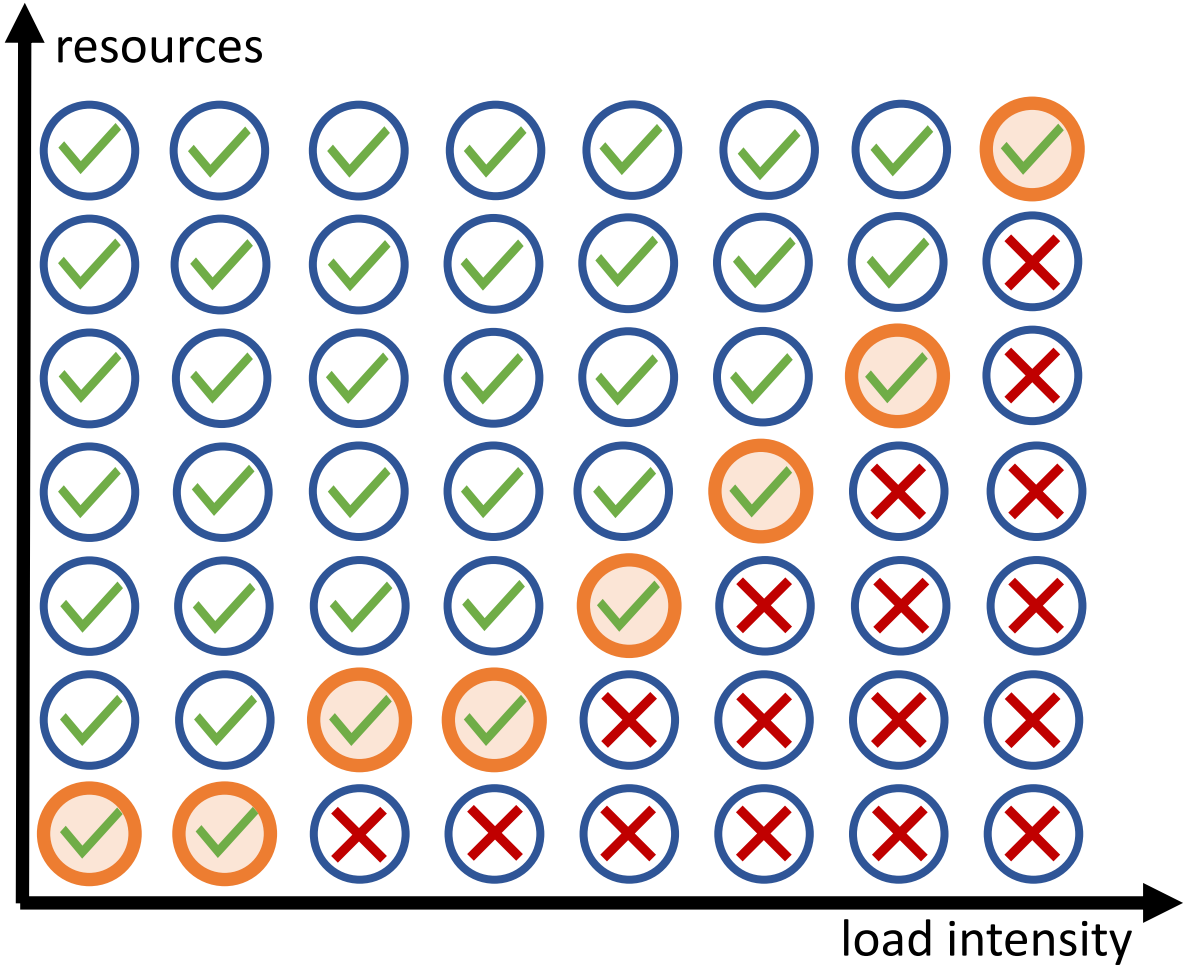
Theodolite’s Scalability Measurement Method



Theodolite's Scalability Measurement Method

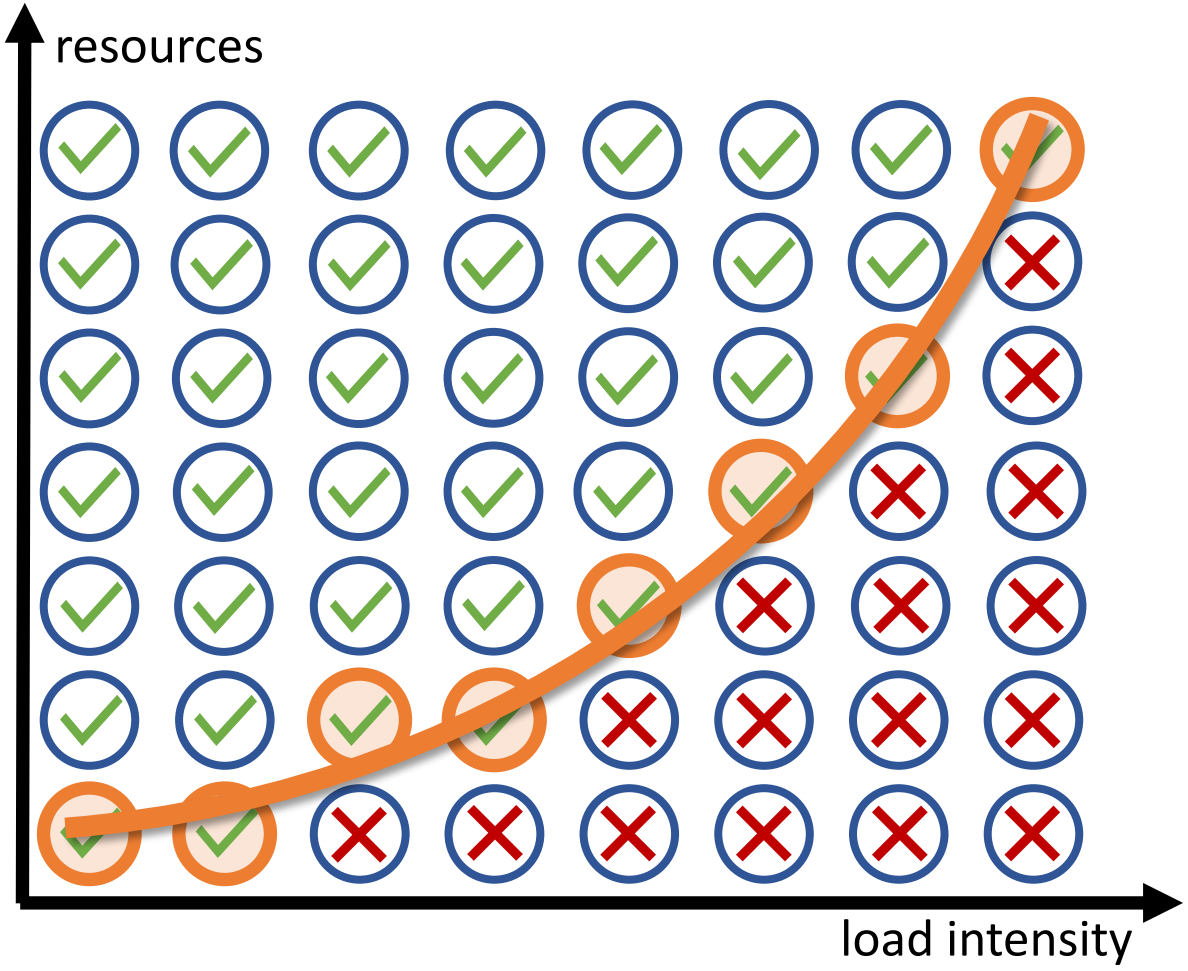


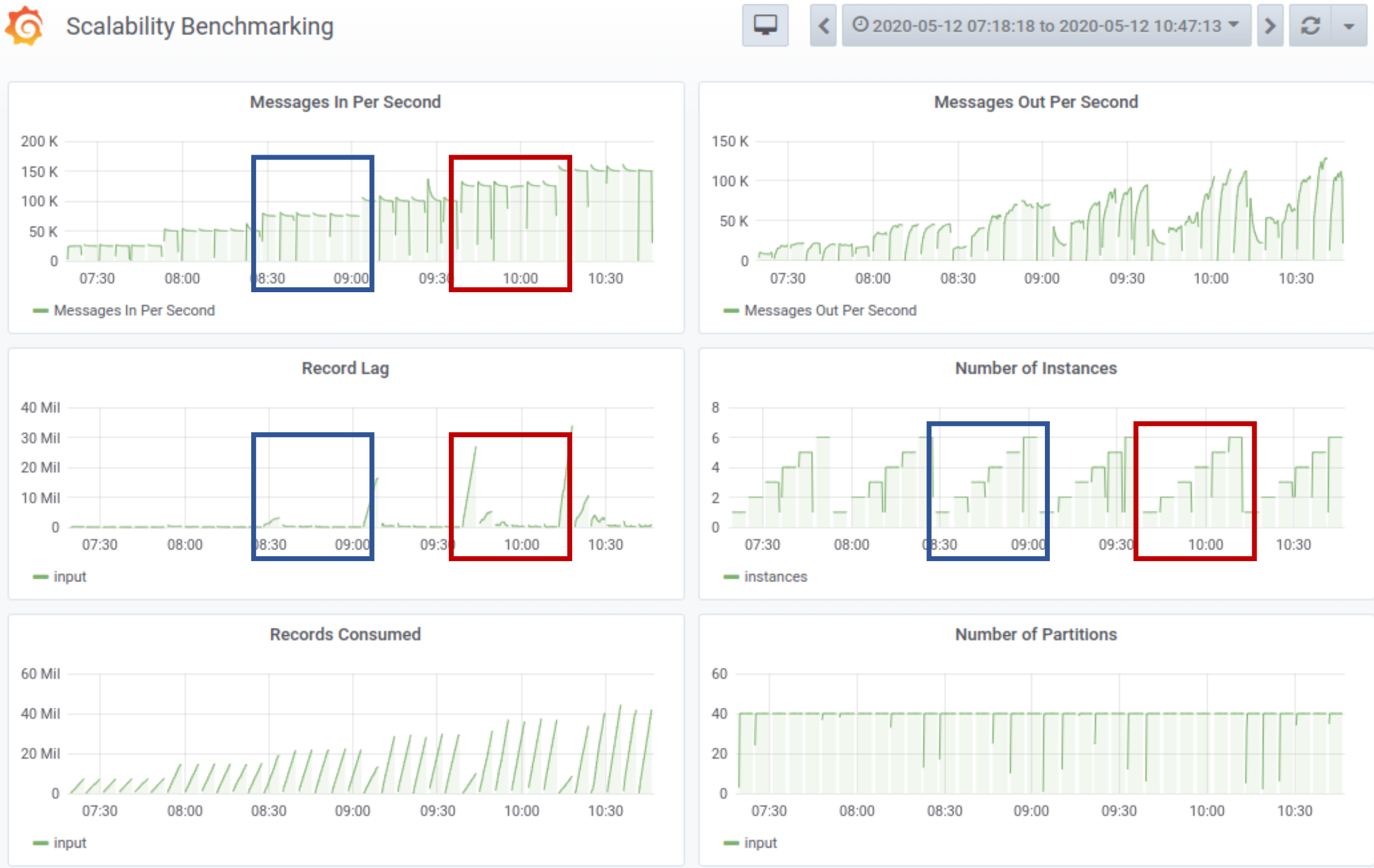
Theodolite's Scalability Measurement Method



Identify minimal required resources per load intensity

Theodolite's Scalability Measurement Method

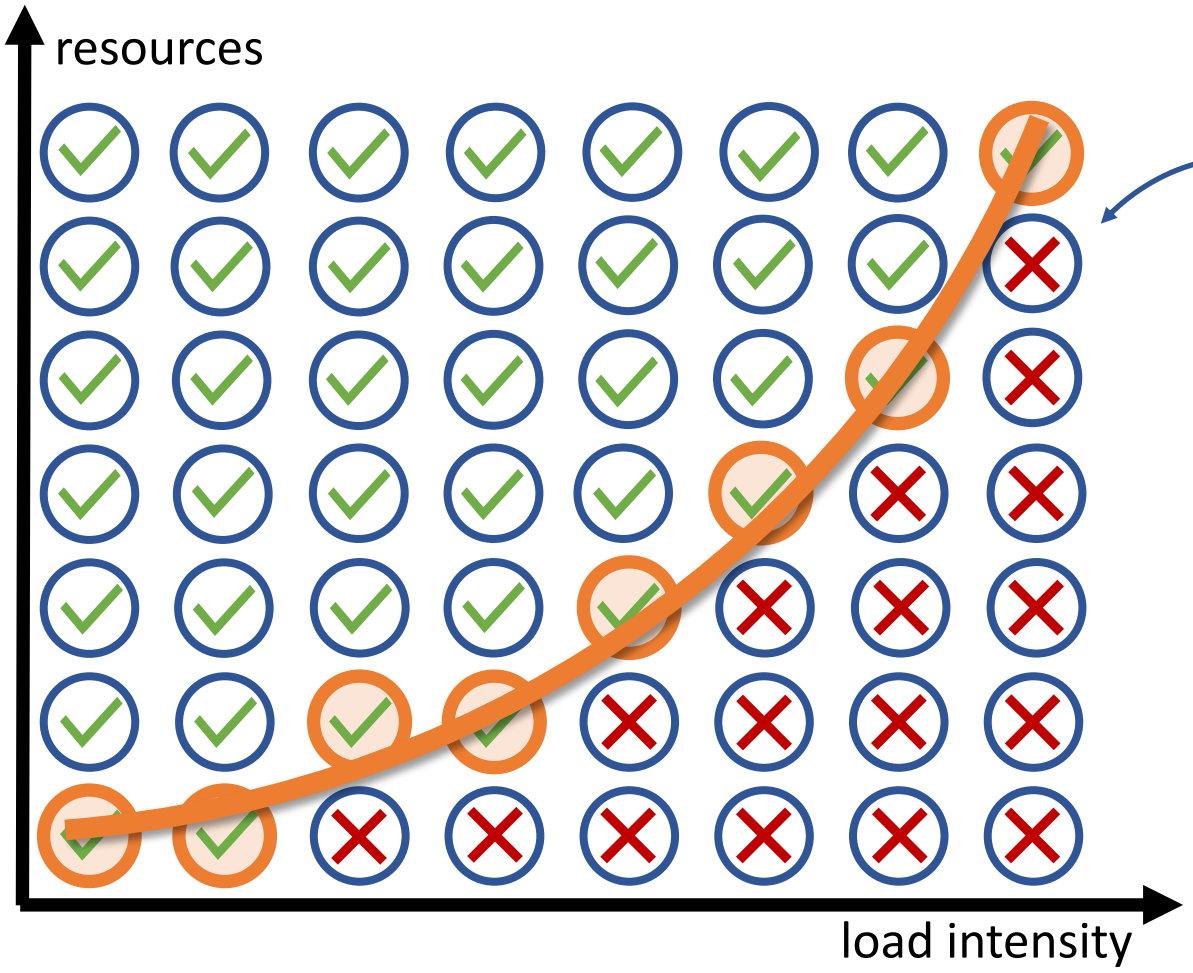


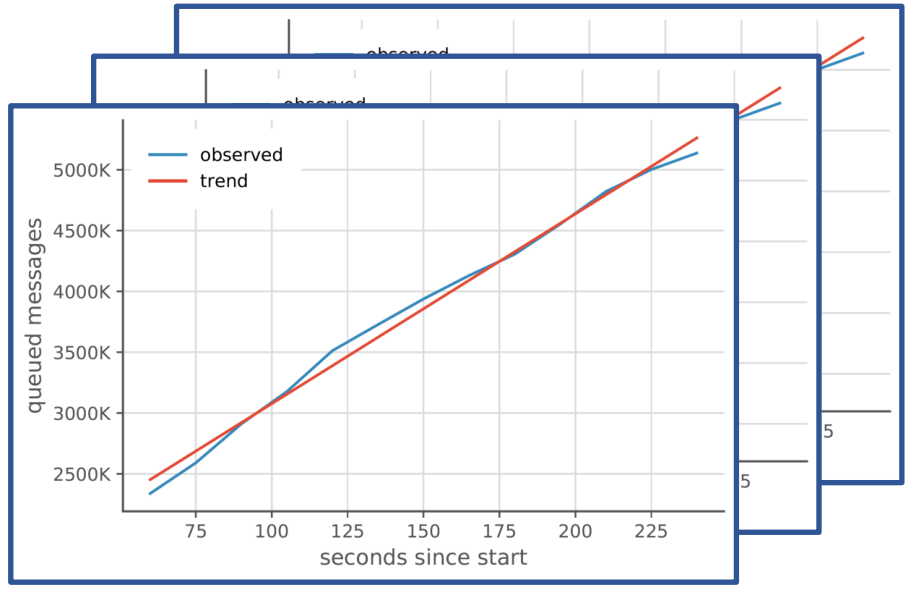

Scalability Benchmarking
2020-05-12 07:18:18 to 2020-05-12 10:47:13
Messages In Per Second
200 K
150 K
100 K
50 K
0
07:30
08:00
08:30
09:00
09:30
10:00
10:30
Messages In Per Second
Messages Out Per Second
150 K
100 K
50 K
0
07:30
08:00
08:30
09:00
09:30
10:00
10:30
Messages Out Per Second
Record Lag
40 Mil
30 Mil
20 Mil
10 Mil
0
07:30
08:00
08:30
09:00
09:30
10:00
10:30
input
Number of Instances
8
6
4
2
0
07:30
08:00
08:30
09:00
09:30
10:00
10:30
instances
Records Consumed
60 Mil
40 Mil
20 Mil
0
07:30
08:00
08:30
09:00
09:30
10:00
10:30
input
Number of Partitions
60
40
20
0
07:30
08:00
08:30
09:00
09:30
10:00
10:30
input

Part 2:

Scalability Benchmarking at Large Scale

Improve Time Efficiency!





$8 \times 7 \times 3 \times 5 \text{ min} = 14 \text{ h}$

Open Research Questions

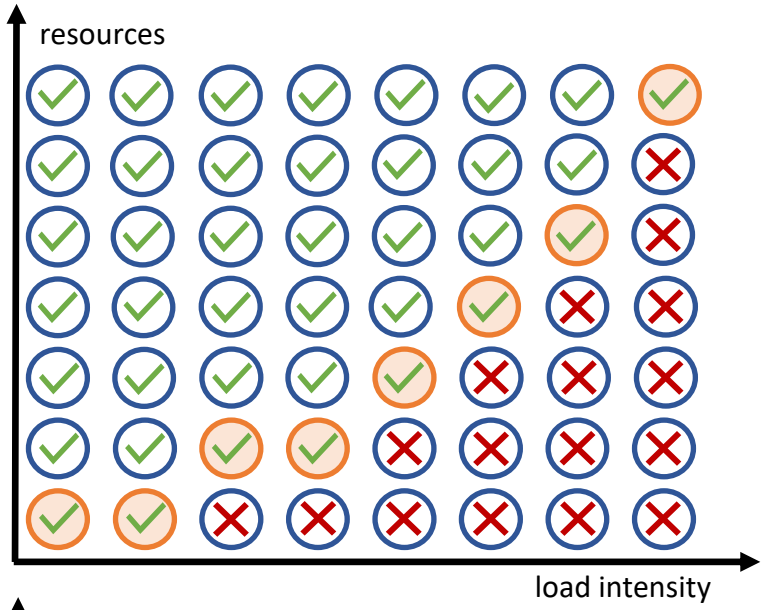
RQ1 How can the scalability metric be measured more efficiently?

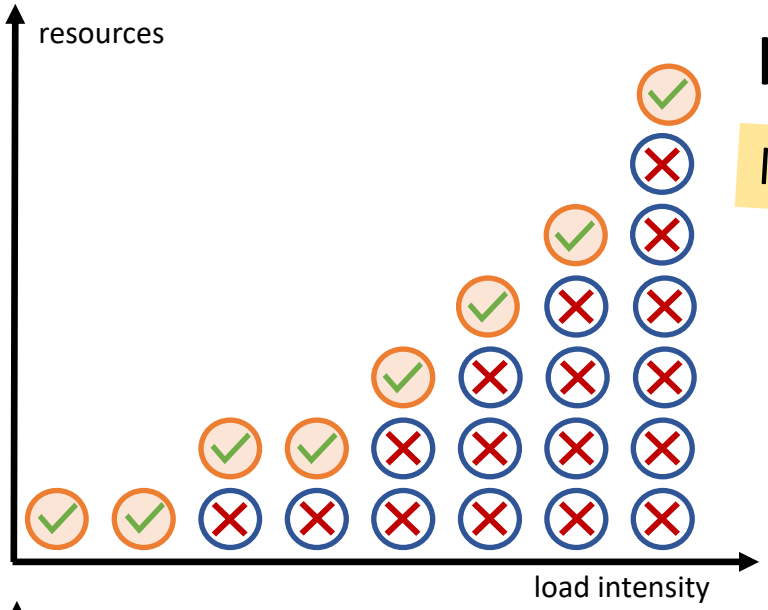
RQ2 For how long should the lag be monitored?

RQ3 How many repetitions are required?

RQ1

Search Heuristics



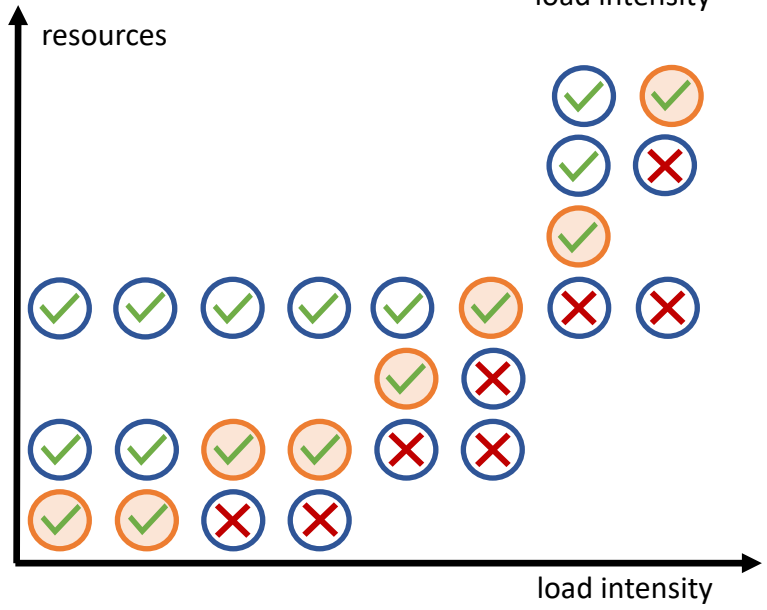


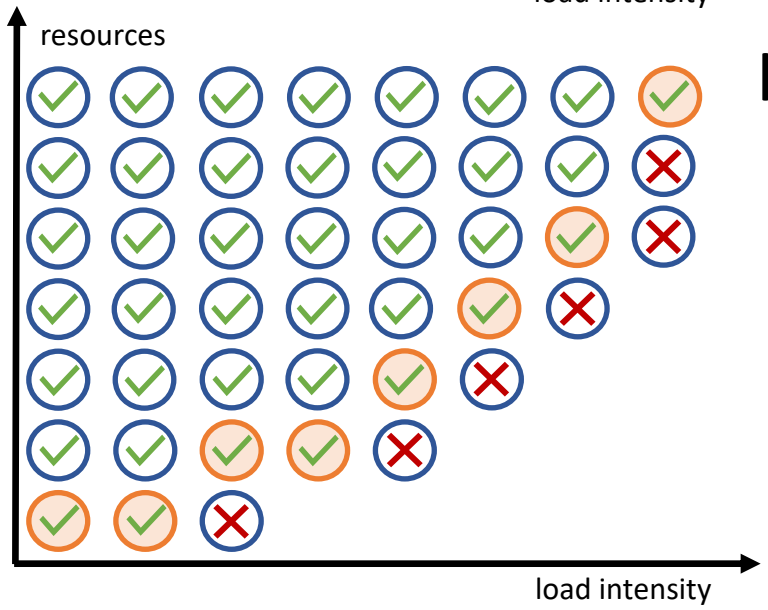
H1

linear search

H2

binary search





H3

assumption:
resource demand monotonically increasing

Research Agenda

RQ1

How can the scalability metric be measured more efficiently?

Use heuristics to execute less experiments.

RQ2

For how long should the lag be monitored?

Identify duration for stable lag trend.

RQ3

How many repetitions are required?

Quantify scattering among experiments.

Conclusions

Benchmarking stream processing frameworks & deployment options at large scale

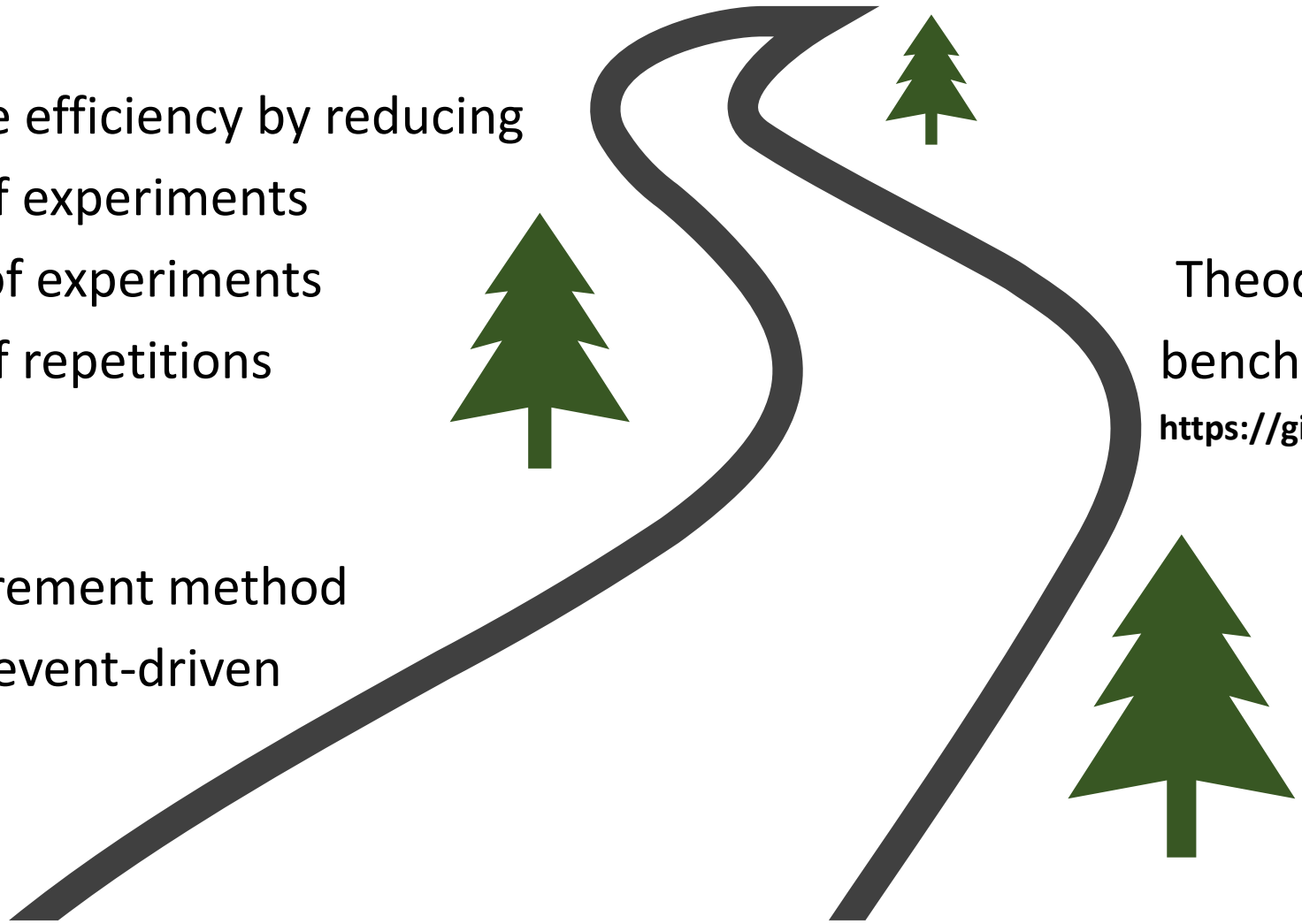
Improve time efficiency by reducing

- number of experiments
- duration of experiments
- number of repetitions

Theodolite: cloud-native benchmarking framework

https://github.com/cau-se/theodolite

Metric & measurement method for scalability in event-driven microservices



References

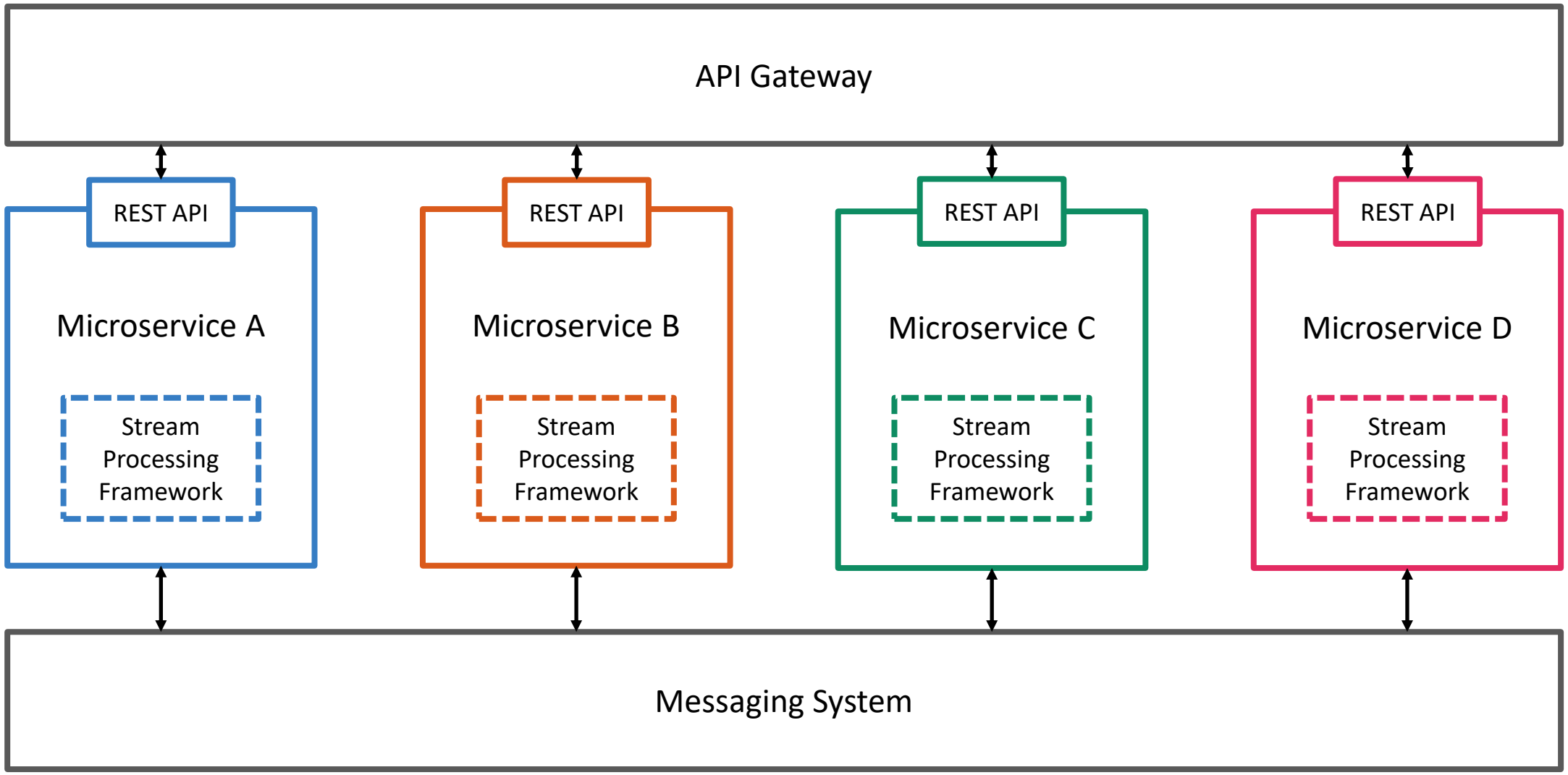
[Herbst et al. 2013] N. Herbst, S. Kounev, and R. Reussner, “Elasticity in Cloud Computing: What it is, and What it is Not” in Proc. International Conference on Autonomic Computing, San Jose, 2013.

[Weber et al. 2014] A. Weber, N. Herbst, H. Groenda, and S. Kounev, “Towards a Resource Elasticity Benchmark for Cloud Environments” in Proc. Int. Workshop on Hot Topics in Cloud Service Scalability, 2014.

[Herbst et al. 2015] N. Herbst, A. Weber, H. Groenda, S. Kounev. “BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments” in Proc. IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2015.

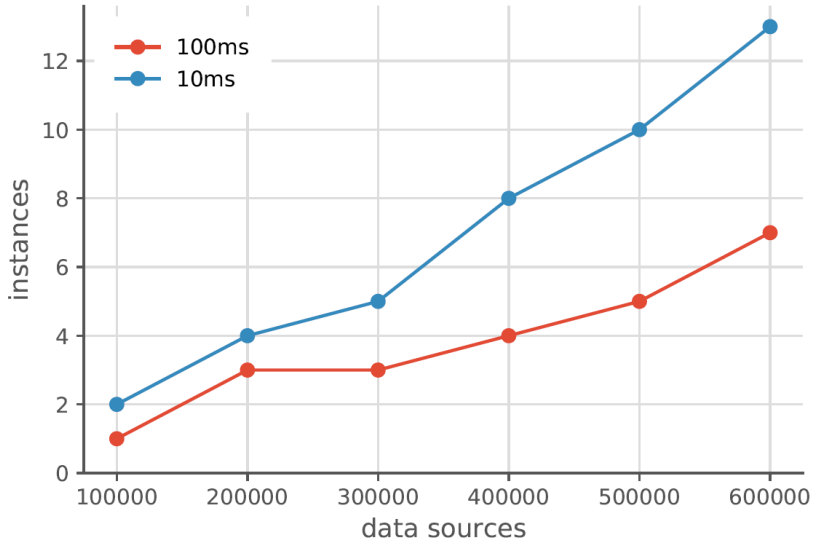
[Henning and Hasselbring 2020] S. Henning and W. Hasselbring. “Theodolite: Scalability Benchmarking of Distributed Stream Processing Engines” in arXiv preprints, arXiv: 2009.00304, 2020.

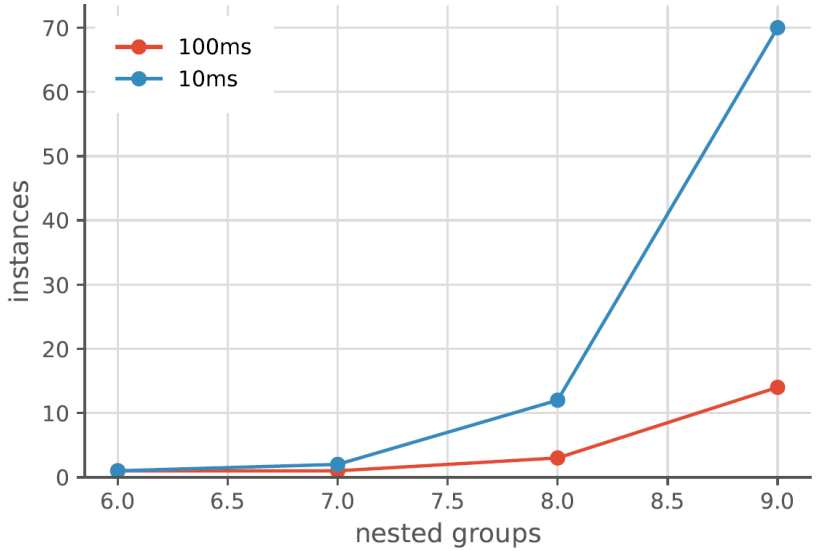
Event-Driven Microservice Architectures



Example: Kafka Streams Commit Interval

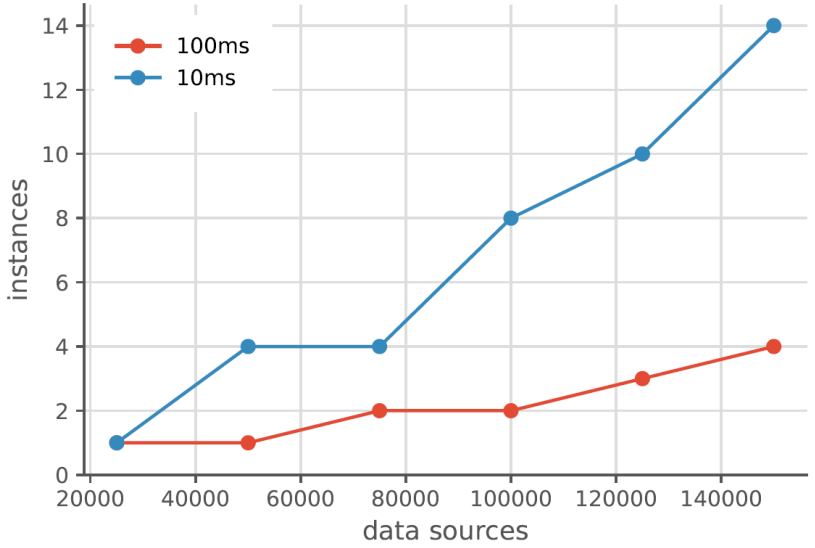
Database Storage

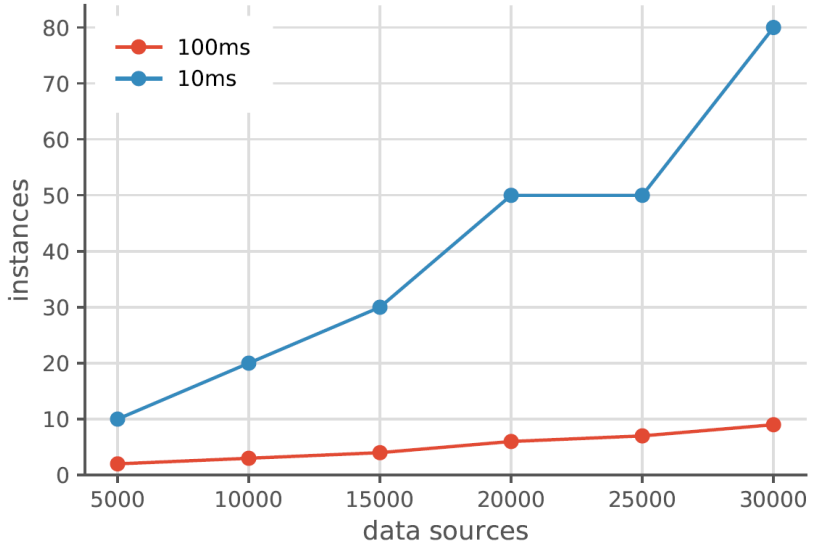




Hierarchical Aggregation

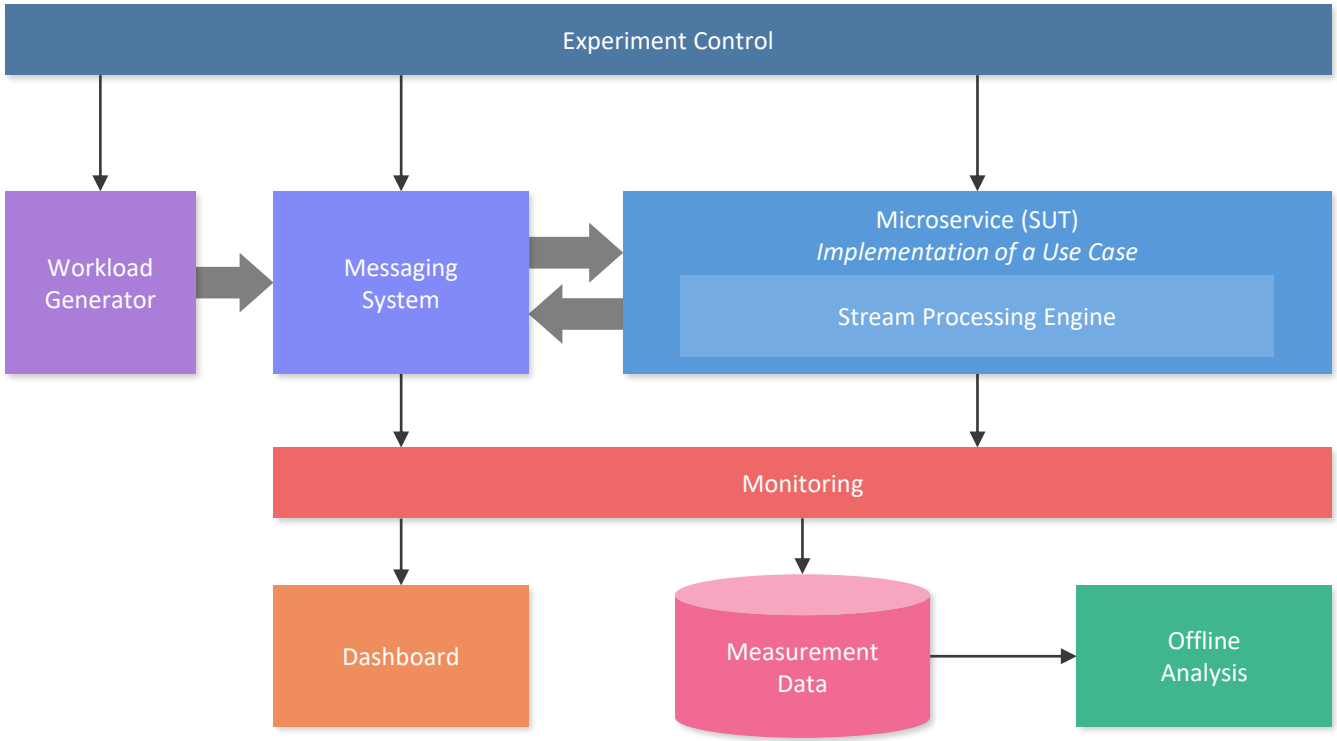
Downsampling





Aggregating Time Attributes

Theodolite's Framework Architecture



https://github.com/
cau-se/theodolite