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

Sören Henning and Wilhelm Hasselbring
Event-Driven Microservice Architectures

REST API
Microservice A
Stream Processing Framework

REST API
Microservice B
Stream Processing Framework

REST API
Microservice C
Stream Processing Framework

REST API
Microservice D
Stream Processing Framework

API Gateway

Messaging System
Event-Driven Microservice Architectures
Event-Driven Microservice Architectures

API Gateway

REST API

Microservice A
  - Spring Cloud Stream

REST API

Microservice B
  - Kafka Streams

REST API

Microservice C
  - Flink

REST API

Microservice D
  - Hazelcast JET

kafka
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
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
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

- Are there sufficient resources for the load?
- Does lag increase over time?

lag = queued messages
Theodolite’s Scalability Measurement Method

- Are there sufficient resources for the load?
- Does the lag increase over time?
Theodolite’s Scalability Measurement Method

sufficient resources for load? **No!**
lag increase over time? **Yes!**

resources

load intensity

[Graph showing queuing messages over time with observed and trend lines]
Theodolite’s Scalability Measurement Method

- Load intensity
- Resources

Is there sufficient resources for load?
Lag increase over time?
Theodolite’s Scalability Measurement Method

resources

load intensity

sufficient resources for load? Yes!

lag increase over time? No!

graph showing queued messages over time.
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
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

Benchmarksing 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


Event-Driven Microservice Architectures

API Gateway

Microservice A
- REST API
- Stream Processing Framework

Microservice B
- REST API
- Stream Processing Framework

Microservice C
- REST API
- Stream Processing Framework

Microservice D
- REST API
- Stream Processing Framework

Messaging System
Example: Kafka Streams

- Commit Interval
- Hierarchical Aggregation
- Database Storage
- Downsampling
- Aggregating Time Attributes
Theodolite’s Framework Architecture

Experiment Control

Workload Generator → Messaging System → Microservice (SUT) Implementation of a Use Case → Stream Processing Engine → Monitoring

- Dashboard
- Measurement Data
- Offline Analysis

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