



# A Big Data Analytics Framework for Evaluating Automated Elastic Scalability of the SMACK-Stack

Diploma Examination

Benedikt Wedenik

October 25, 2018

# Table of contents

# Motivation & Research Challenges

# Motivation

In the last years the demand for information availability and shorter response times has been increasing. Today's business requirements are changing: Waiting hours or even days for the result of a query is not acceptable anymore in many sectors.
The response needs to be immediate, or the query is discarded [**?**]. This is why "Fast Data", as an approach to solve those problems, increases its popularity, as being "big data, but fast" [**?**].

# Research Challenges

- **Deploying large scale applications**
  Requires multiple instances of different technologies to be deployed in a defined sequence to fulfill subsequent dependencies, often including manual steps.
- **Initial setup**
  The decision of how to configure the instances of an application is a non-trivial task, as there are almost infinite combination possibilities and the impact can be drastic.

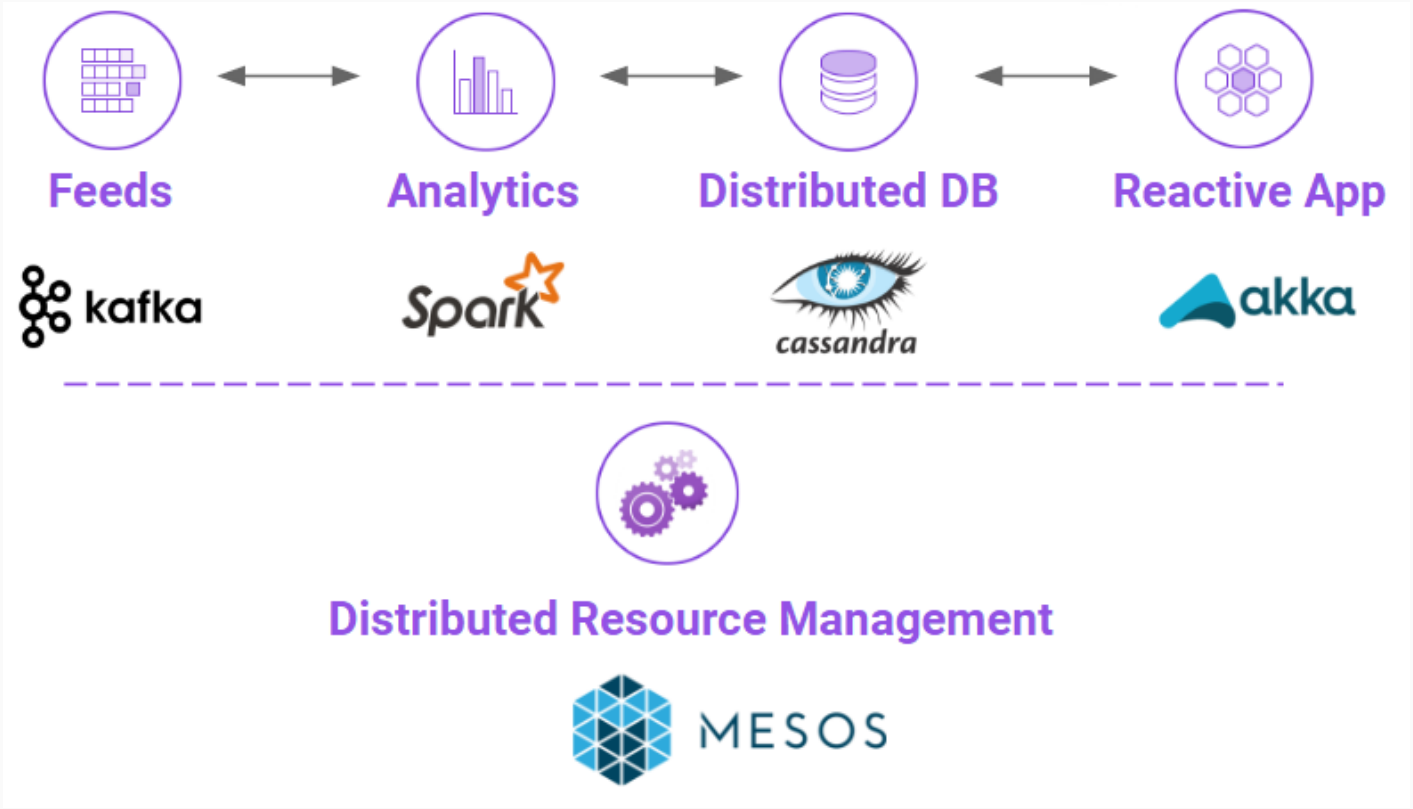
# Research Challenges

- **Monitoring**
  Considering just RAM, CPU and disk usage is in most cases insufficient, as a deeper understanding of the used frameworks is required, which introduces a new layer of complexity.
- **Scaling when needed**
  Understanding whats going on in a cluster and reacting accordingly is crucial for the success of any large scale application.

# Background

# Background

The SMACK-Stack consists of five technologies combined to a lightning fast data pipeline for today's needs of big data applications, as shown below.




mesosphere.com [?]

# Background

- Apache **S**park is the engine of the pipeline, providing batch-, as well as stream-processing power for large-scale data processing.
- **M**esos is a datacenter operating system with the aim to reduce complexity and ease the deployment and maintenance of large-scale distributed applications.
- Apache **A**kka can be seen as the model, providing the possibility to build powerful reactive distributed message-driven applications.

# Background

- Apache **C**assandra is a highly distributed database which is a hybrid between a column-oriented and a key-value DBMS, which is implemented avoiding a single point of failure.
- Apache **K**afka serves as publish-subscribe message broker, which is usually the ingestion point of the pipeline.

# Contribution

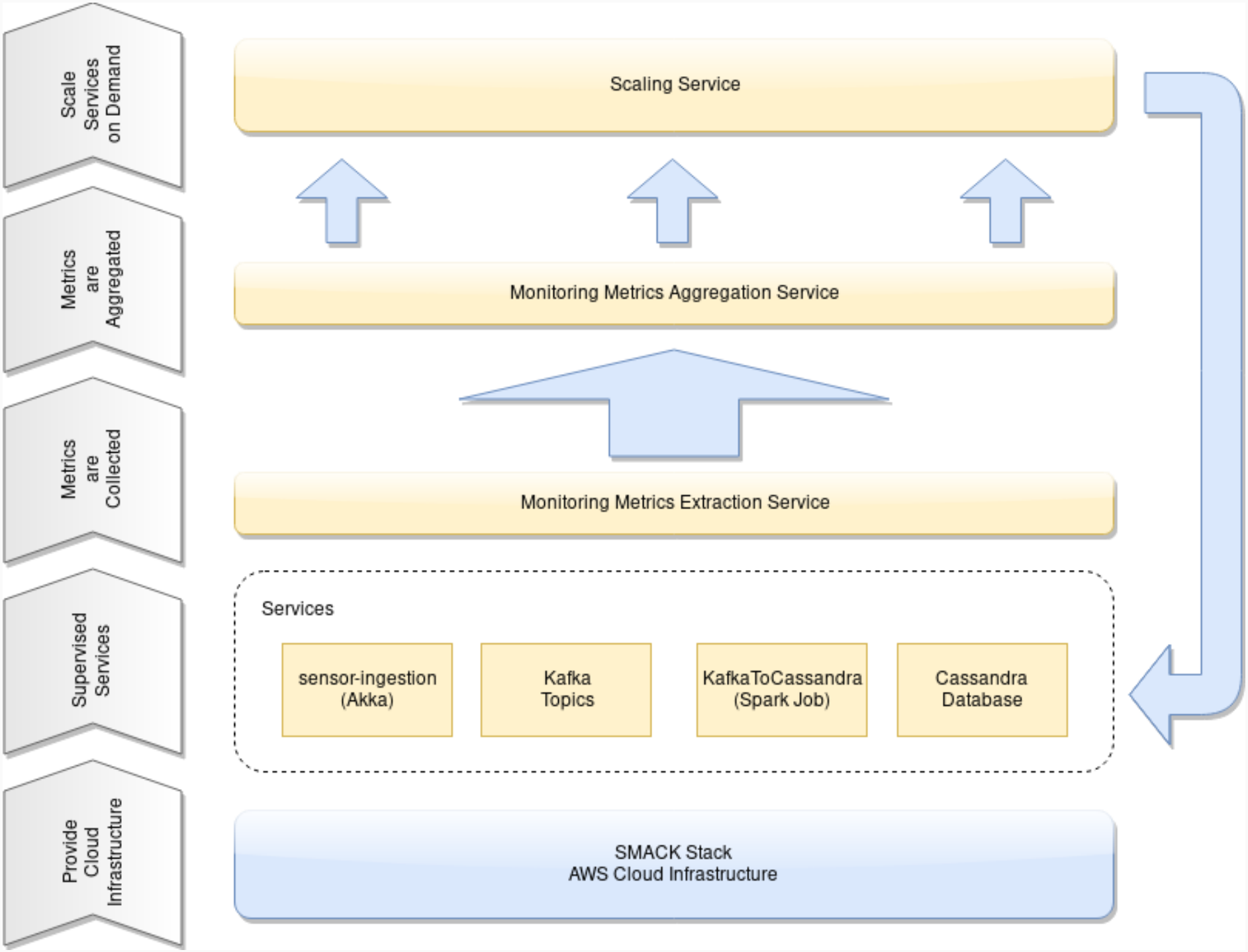
# Contribution

To solve the mentioned problems, a framework has been developed, whereas the single components can be described as follows:

- **Framework to Easily Launch SMACK in AWS**
  With the help of this framework it takes just a few command line calls to launch and deploy the whole SMACK stack in the cloud.
- **JMX Extraction Tool**
  This tool is designed to automatically extract interesting metrics from SMACK components via JMX and sending them to a central service, in this case the REST monitoring service.

# Contribution

- **REST Service Collecting Monitoring Information**
  This is the service which collects all the extracted metrics and compiles them into a useful format. In addition, there is the possibility to generate plots at runtime.
- **Automated Scaling Tool for SMACK**
  The scaling tool evaluates the collected metrics from the REST service and scales the individual parts of the SMACK stack up or down.
- **Deployment Blueprints**
  Those reference architecture and configuration recommendations help to launch the SMACK stack and getting most out of the available resource.

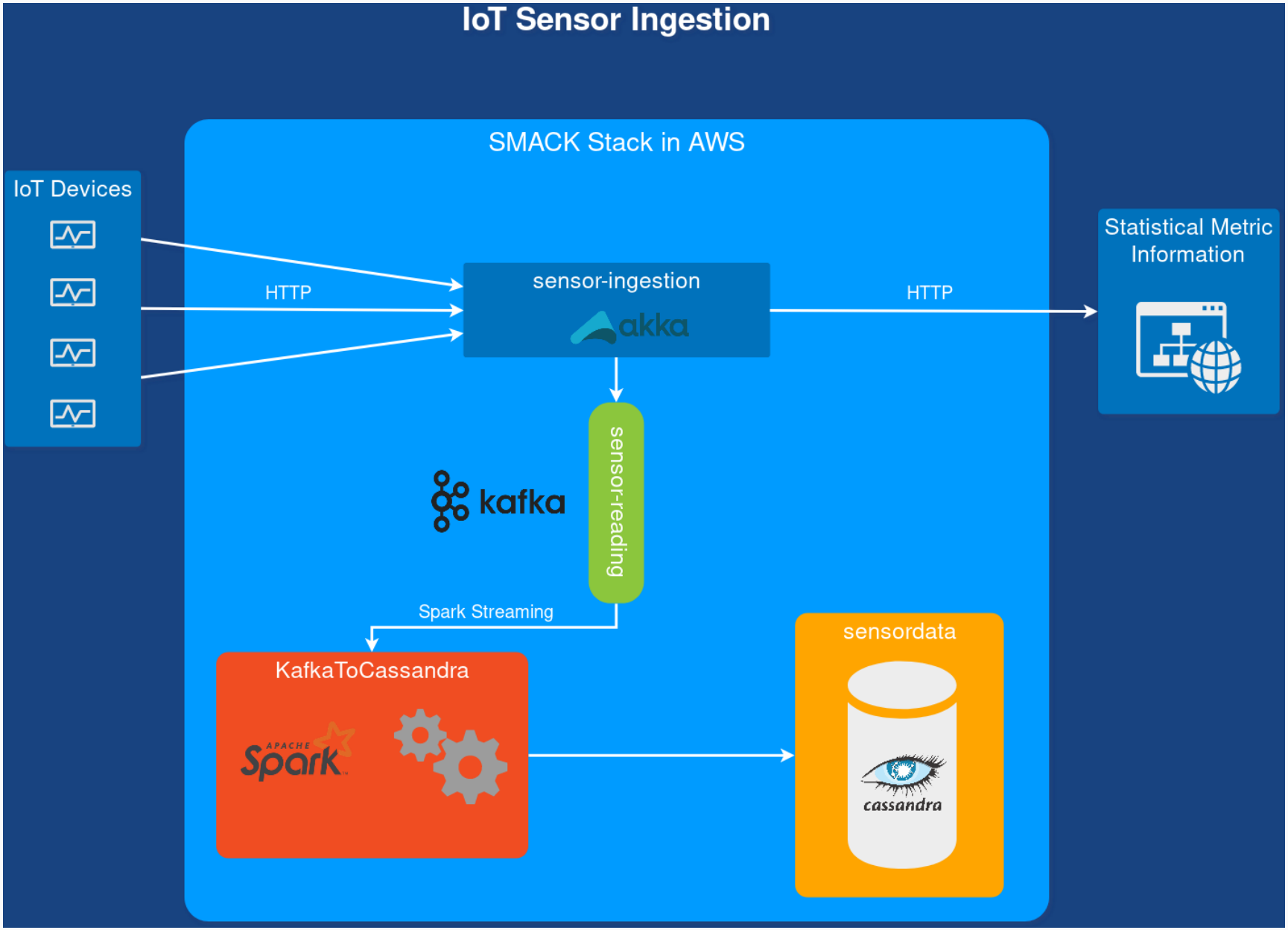

Scale Services on Demand
Metrics are Aggregated
Metrics are Collected
Supervised Services
Provide Cloud Infrastructure
Scaling Service
Monitoring Metrics Aggregation Service
Monitoring Metrics Extraction Service
Services
sensor-ingestion (Akka)
Kafka Topics
KafkaToCassandra (Spark Job)
Cassandra Database
SMACK Stack
AWS Cloud Infrastructure

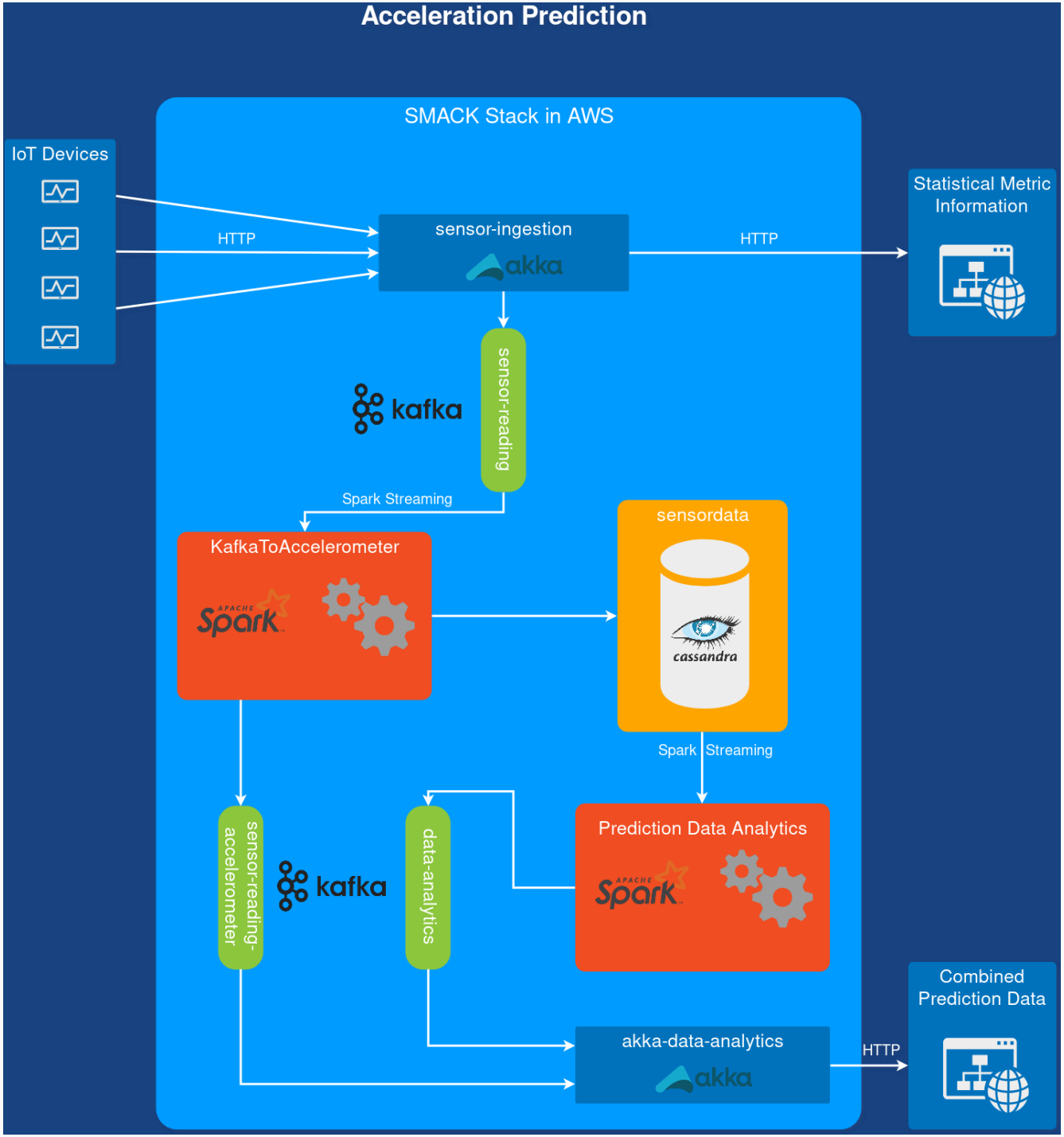
# Evaluation

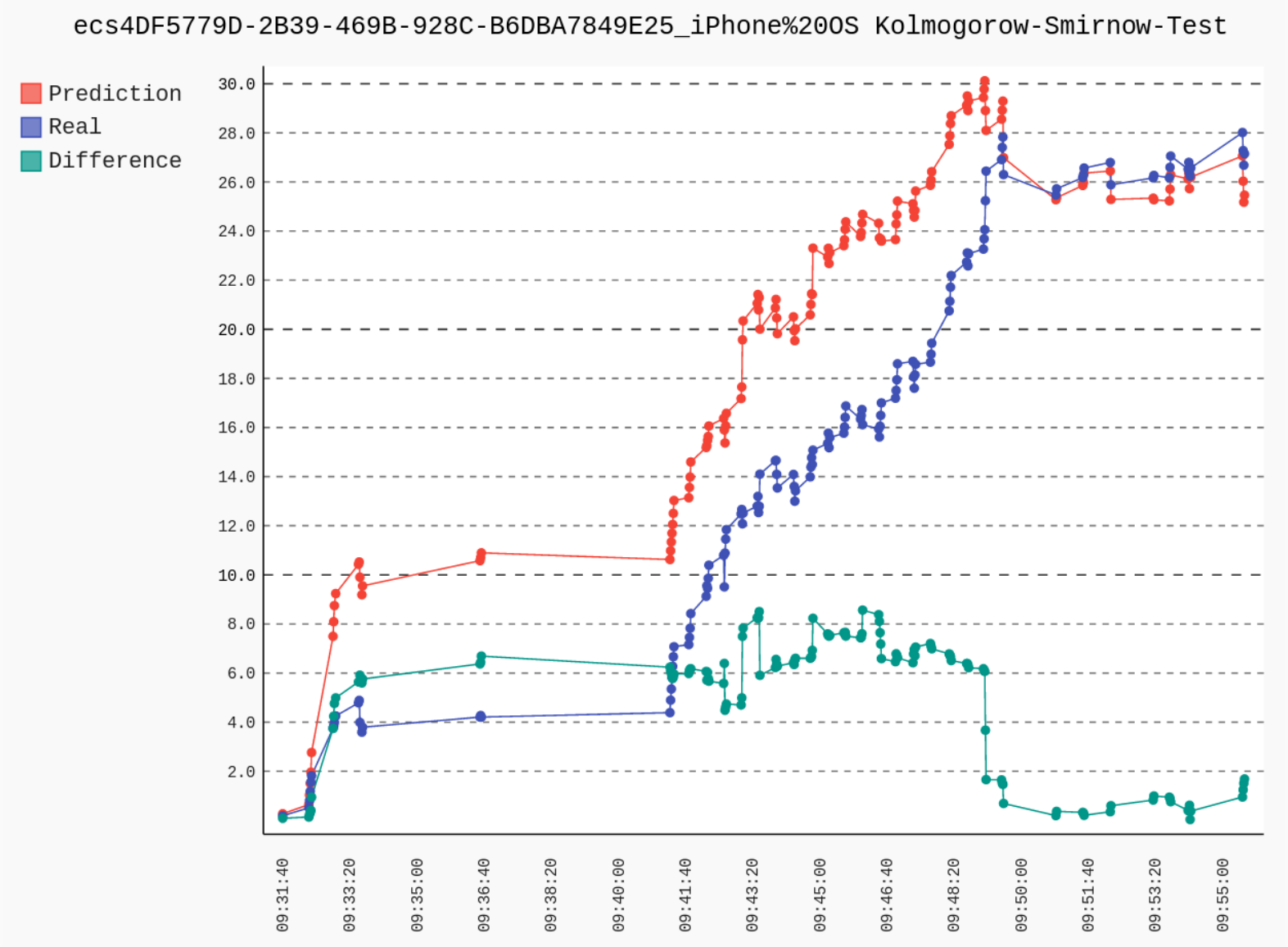
# Evaluation

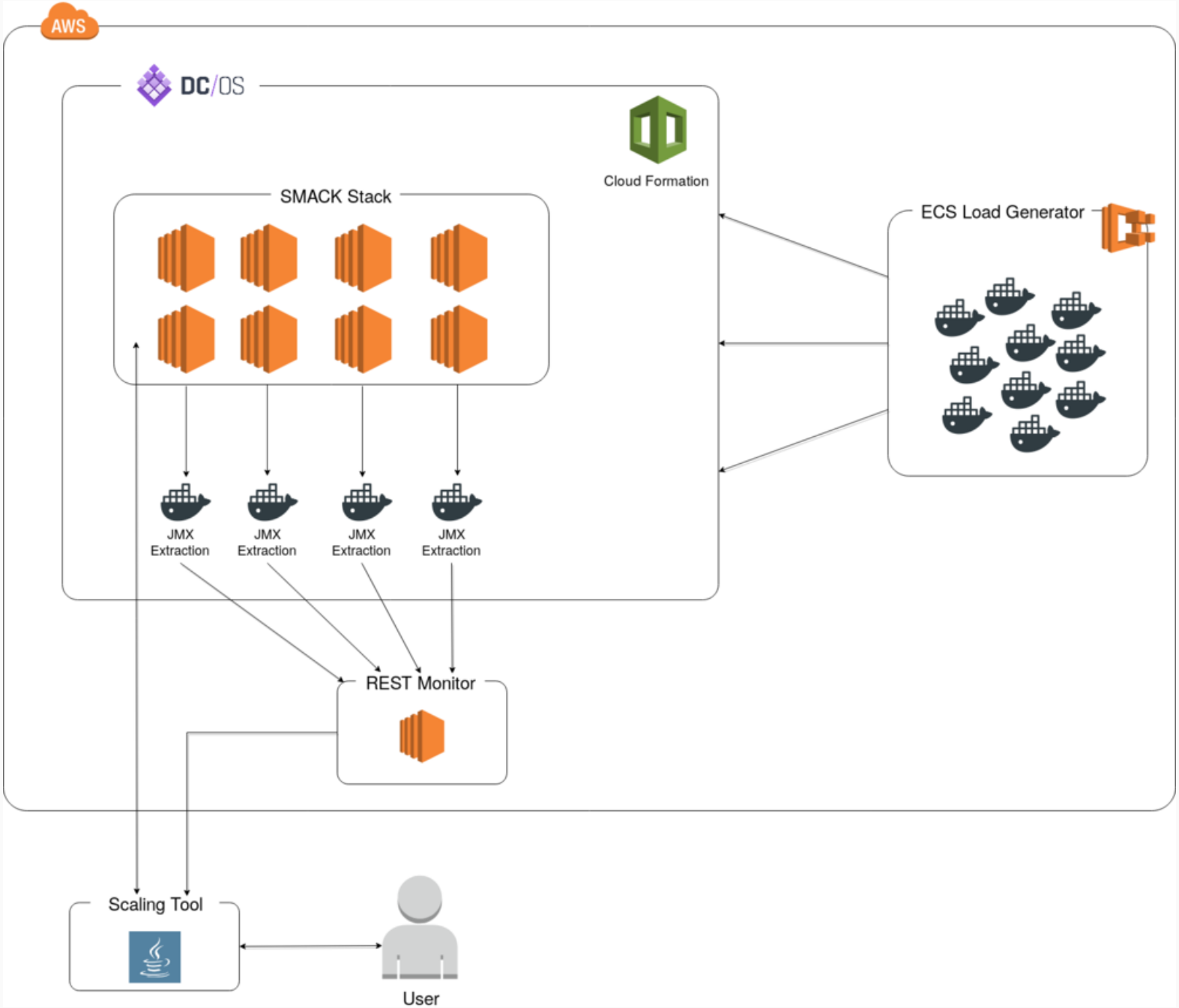
In order to evaluate the framework's performance, a use-case based approach with two real-world applications is chosen.
As part of the contribution, two real world applications have been developed in order to provide a valid base for the evaluation of the framework.

- The *IoT Data Storage Application* is mainly I/O bound and represents applications which require high throughput.
- In case of the *Acceleration Prediction Application*, a prediction based on IoT data is performed, which is heavily computational intense.


IoT Sensor Ingestion
SMACK Stack in AWS
IoT Devices
HTTP
sensor-ingestion
akka
HTTP
Statistical Metric Information
kafka
sensor-reading
Spark Streaming
KafkaToCassandra
APACHE Spark
sensordata
cassandra


Acceleration Prediction
SMACK Stack in AWS
IoT Devices
HTTP
sensor-ingestion
akka
HTTP
Statistical Metric Information
kafka
sensor-reading
Spark Streaming
KafkaToAccelerometer
Spark
sensordata
cassandra
Spark Streaming
sensor-reading-accelerometer
kafka
data-analytics
Prediction Data Analytics
Spark
akka-data-analytics
akka
HTTP
Combined Prediction Data


ecs4DF5779D-2B39-469B-928C-B6DBA7849E25_iPhone%20OS Kolmogorow-Smirnow-Test
Prediction
Real
Difference
30.0
28.0
26.0
24.0
22.0
20.0
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
09:31:40
09:33:20
09:35:00
09:36:40
09:38:20
09:40:00
09:41:40
09:43:20
09:45:00
09:46:40
09:48:20
09:50:00
09:51:40
09:53:20
09:55:00

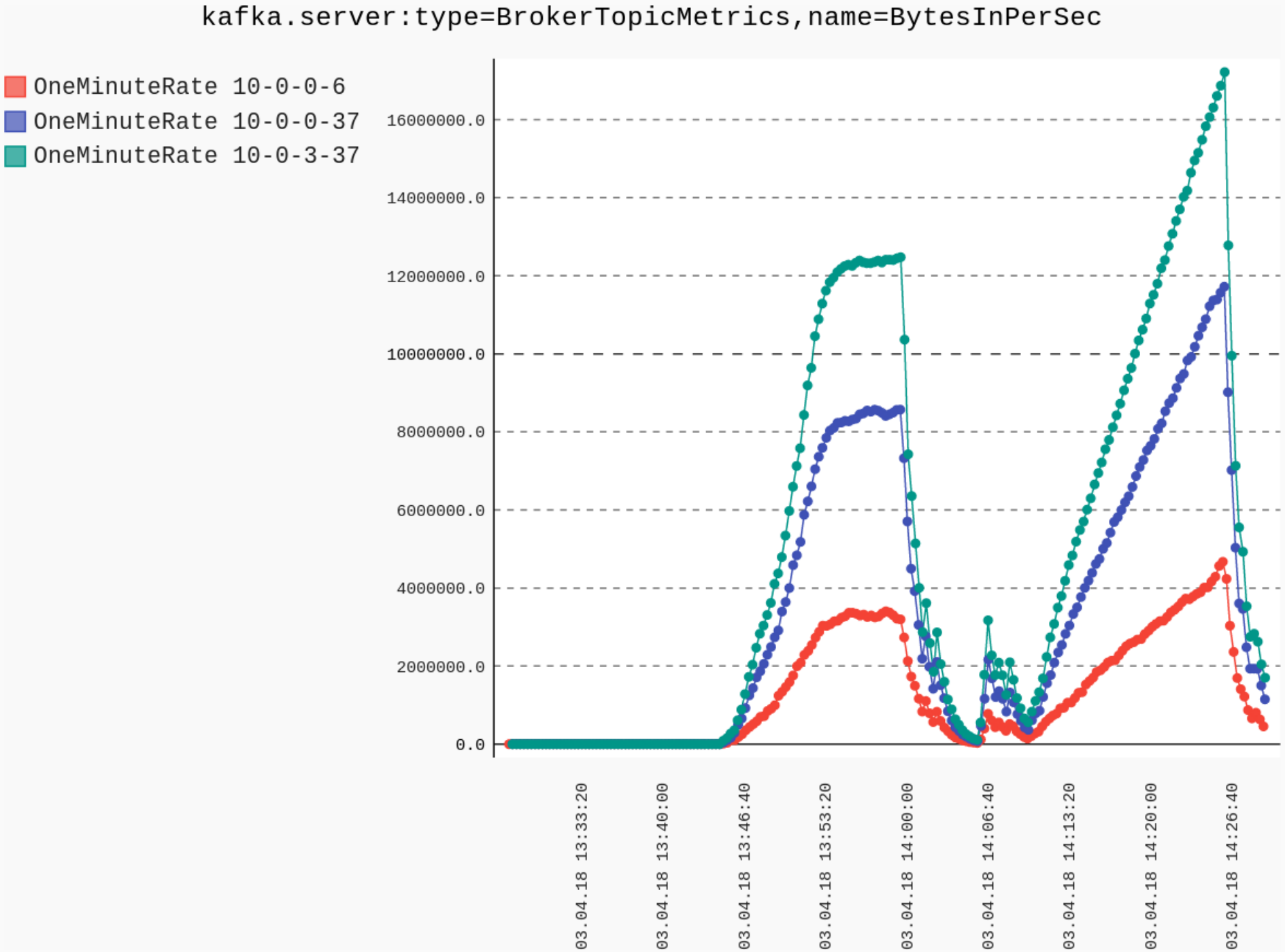

AWS
DC/OS
Cloud Formation
SMACK Stack
ECS Load Generator
JMX Extraction
JMX Extraction
JMX Extraction
JMX Extraction
REST Monitor
Scaling Tool
User

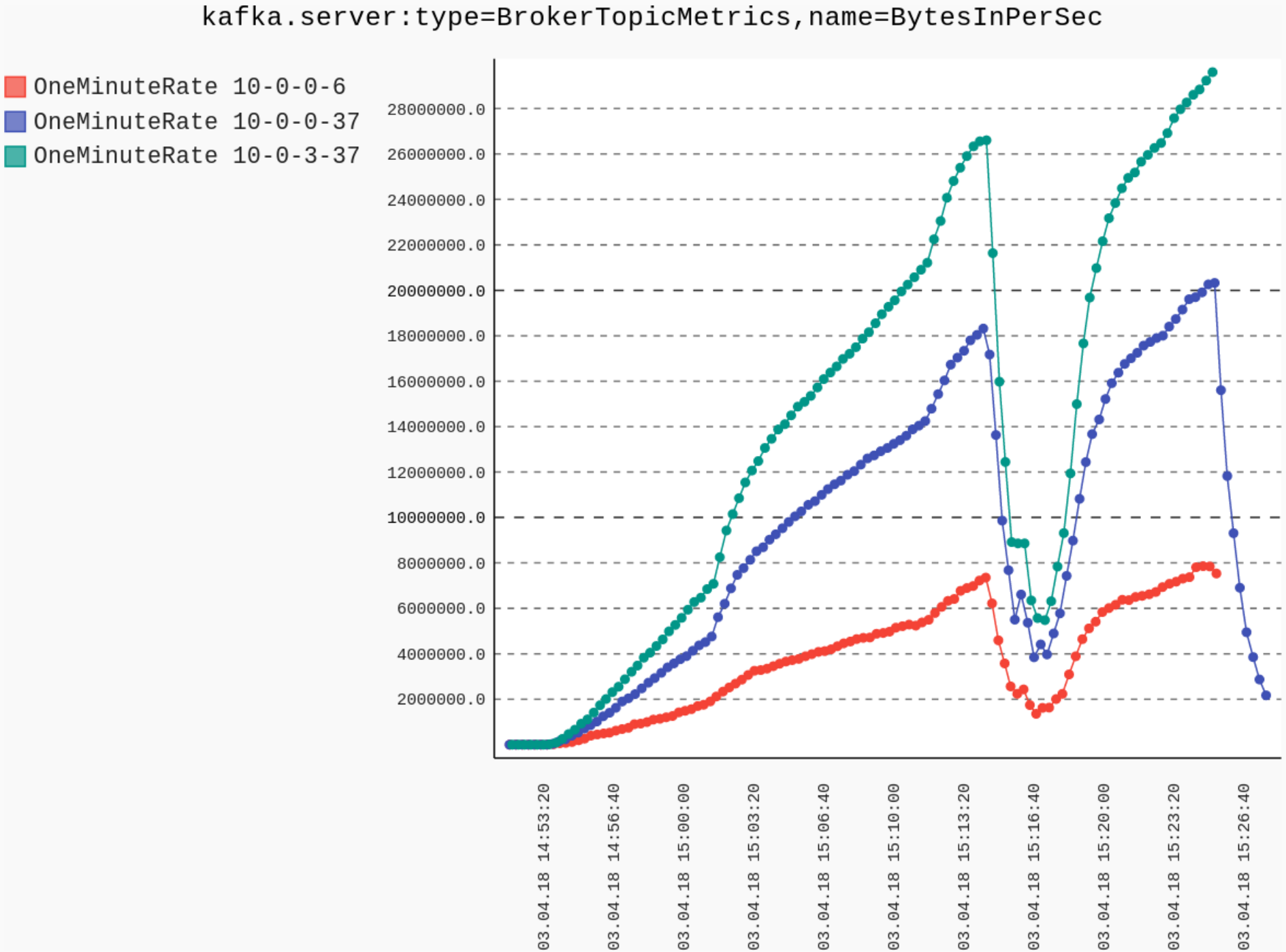
# Results & Conclusion

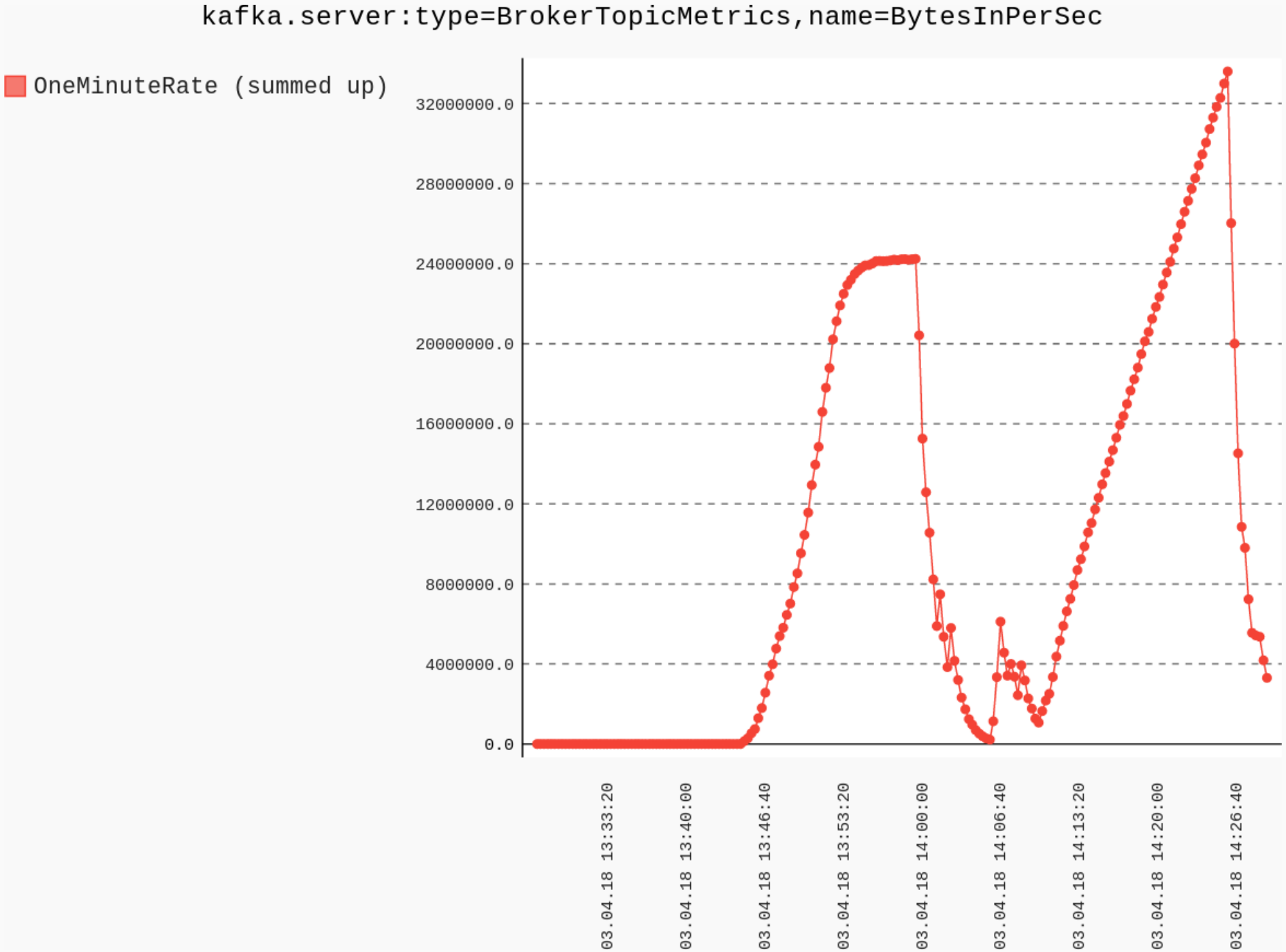
# Results

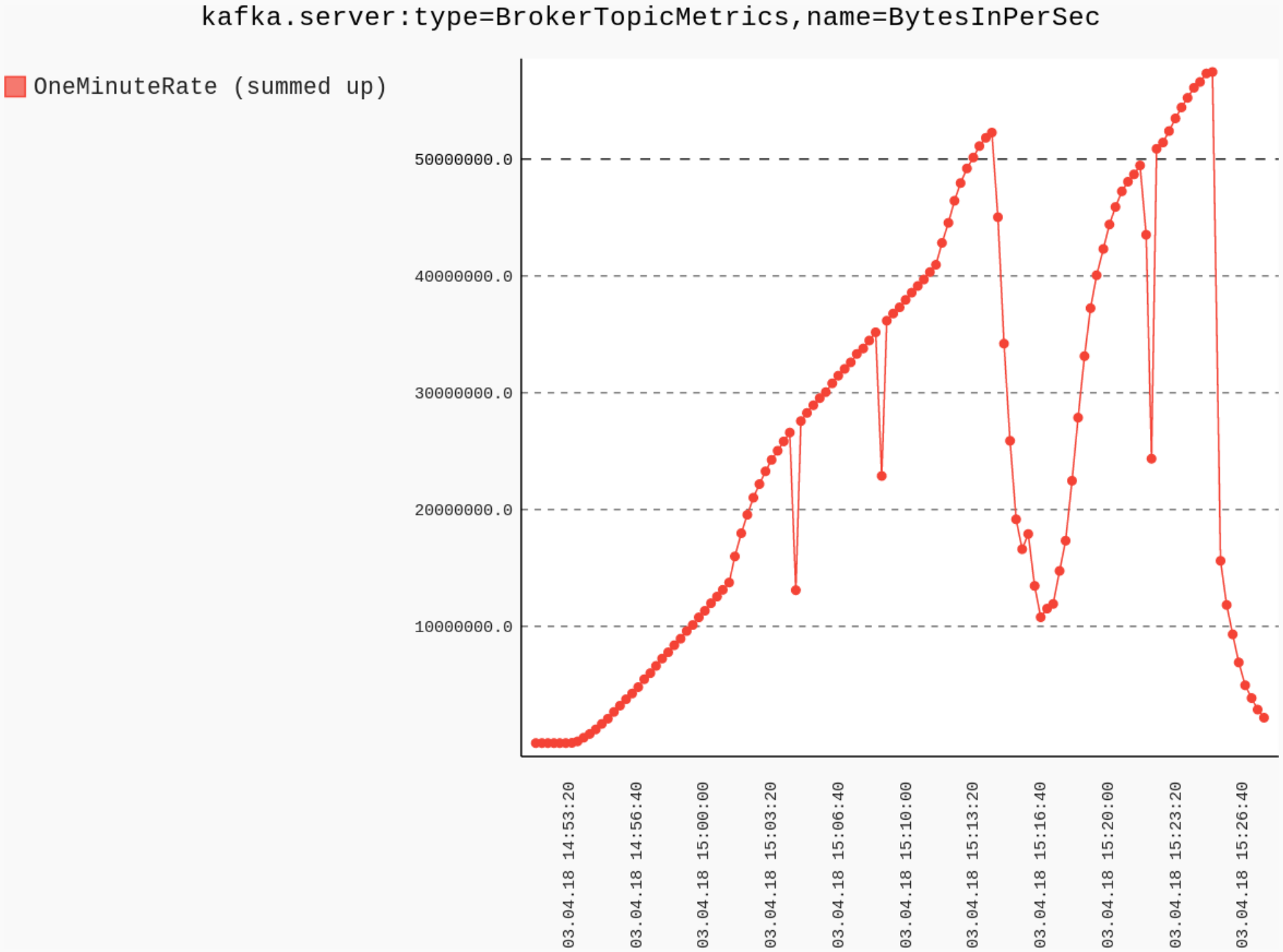
Empirical experiments have been conducted for evaluation.
The setup comprises two runs: **unsupervised**, which serves as a baseline, and **supervised**, in which the framework automatically scales the respective services.
Both applications benefit from the scaling service, where the following improvements can be highlighted:

- An **increase from 272 to 472 MBit/s** was measured when running the *IoT Data Storage Application*, as well as other **metrics like the time-in-mailbox of a message were improved**.
- The *Acceleration Prediction Application* benefited in form of **shorter message processing times** and an overall **faster task completion**. Based on the calculated average of the task completion, the run with the enabled Scaling Tool was about 169% faster.


kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec
OneMinuteRate 10-0-0-6
OneMinuteRate 10-0-0-37
OneMinuteRate 10-0-3-37
16000000.0
14000000.0
12000000.0
10000000.0
8000000.0
6000000.0
4000000.0
2000000.0
0.0
03.04.18 13:33:20
03.04.18 13:40:00
03.04.18 13:46:40
03.04.18 13:53:20
03.04.18 14:00:00
03.04.18 14:06:40
03.04.18 14:13:20
03.04.18 14:20:00
03.04.18 14:26:40


kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec
OneMinuteRate 10-0-0-6
OneMinuteRate 10-0-0-37
OneMinuteRate 10-0-3-37
28000000.0
26000000.0
24000000.0
22000000.0
20000000.0
18000000.0
16000000.0
14000000.0
12000000.0
10000000.0
8000000.0
6000000.0
4000000.0
2000000.0
03.04.18 14:53:20
03.04.18 14:56:40
03.04.18 15:00:00
03.04.18 15:03:20
03.04.18 15:06:40
03.04.18 15:10:00
03.04.18 15:13:20
03.04.18 15:16:40
03.04.18 15:20:00
03.04.18 15:23:20
03.04.18 15:26:40


kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec
OneMinuteRate (summed up)
32000000.0
28000000.0
24000000.0
20000000.0
16000000.0
12000000.0
8000000.0
4000000.0
0.0
03.04.18 13:33:20
03.04.18 13:40:00
03.04.18 13:46:40
03.04.18 13:53:20
03.04.18 14:00:00
03.04.18 14:06:40
03.04.18 14:13:20
03.04.18 14:20:00
03.04.18 14:26:40


kafka.server:type=BrokerTopicMetrics,name=BytesInPerSec
OneMinuteRate (summed up)
50000000.0
40000000.0
30000000.0
20000000.0
10000000.0
03.04.18 14:53:20
03.04.18 14:56:40
03.04.18 15:00:00
03.04.18 15:03:20
03.04.18 15:06:40
03.04.18 15:10:00
03.04.18 15:13:20
03.04.18 15:16:40
03.04.18 15:20:00
03.04.18 15:23:20
03.04.18 15:26:40

# Open Issues & Future Work

- Kolmogorov Smirnov test shows, that the prediction correlates with the real measured values. Still there is an obvious bias, which could be eliminated or at least decreased.
- Extended REST API for Acceleration Prediction Application.
- Unfortunately scaling down is not supported for Kafka and Cassandra, due to the restrictions of DC/OS [**?**], [**?**].
- The selection of what to extract could be refined further to reduce overhead and only focus on what is important.
- Currently the threshold management in the Scaling Tool is hard-coded and static, which is an obvious place for further improvements.

# Questions

I'm happy to answer your questions!

# References


Cassandra Limitations, GitHub.com.

**https://github.com/mesosphere/dcos-cassandra-service/blob/master/docs/limitations.md.**

Accessed: 24.07.2018.

Mesosphere.

**https://mesosphere.com/.**

Accessed: 24.07.2018.

R. Estrada and I. Ruiz.

**Big data, big challenges.**

In *Big Data SMACK*, pages 3–16. Springer, 2016.

G. Mishne, J. Dalton, Z. Li, A. Sharma, and J. Lin.

**Fast data in the era of big data: Twitter's real-time related query suggestion architecture.**

In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 1147–1158. ACM, 2013.