



UPPSALA
UNIVERSITET

Data Engineering - II

Course code: 1TD075 62033

M1 - Data Stream Processing (Part-II)

Salman Toor
salman.toor@it.uu.se

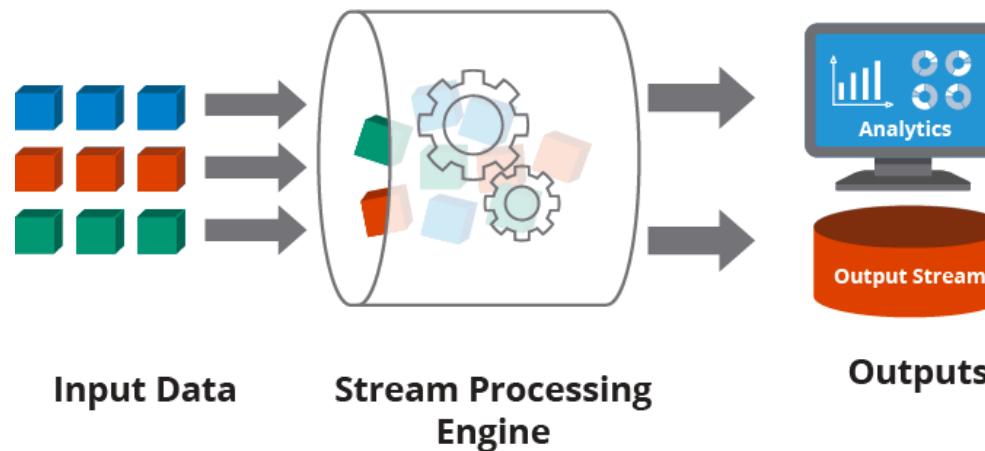


UPPSALA
UNIVERSITET

Introduction

- Data stream processing
- Differences between batch, micro-batch and stream processing
- Components and conventional architectures for data stream processing
- Streaming frameworks
- Performance metrics for streaming frameworks

Data stream processing



- Data sources
 - sensor networks
 - wireless networks
 - customer click streams
 - multimedia data
 - scientific data
- Data source characteristics
 - open-ended
 - flowing at high-speed
 - non stationary distributions in dynamic environments

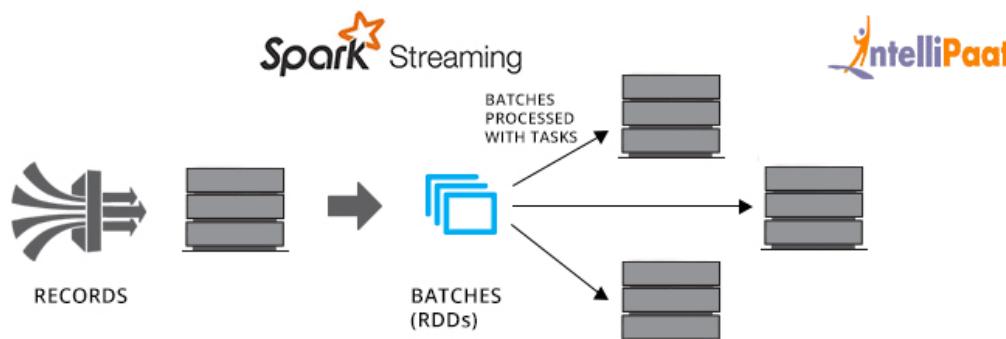
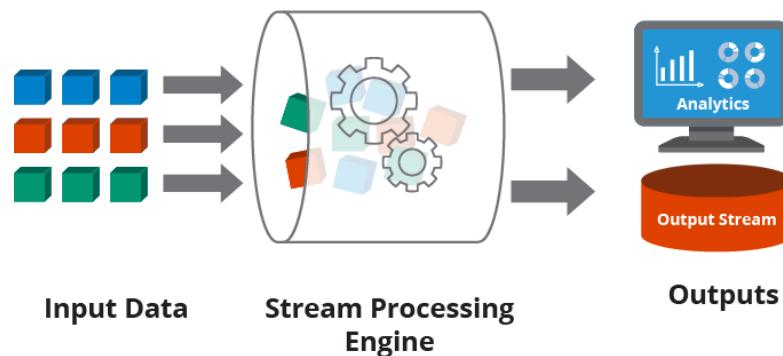


UPPSALA
UNIVERSITET

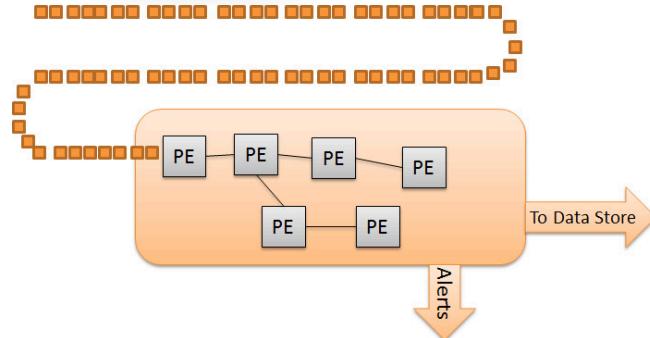
Stream processing

- Challenges
 - It is not possible to find the exact solution for the functions such as min or max
 - All blocking operations are difficult to execute in stream processing
- Active research directions
 - Approximate query processing techniques
 - Novel architectures for blocking operators
 - Reliable processing and management of high coming data rate

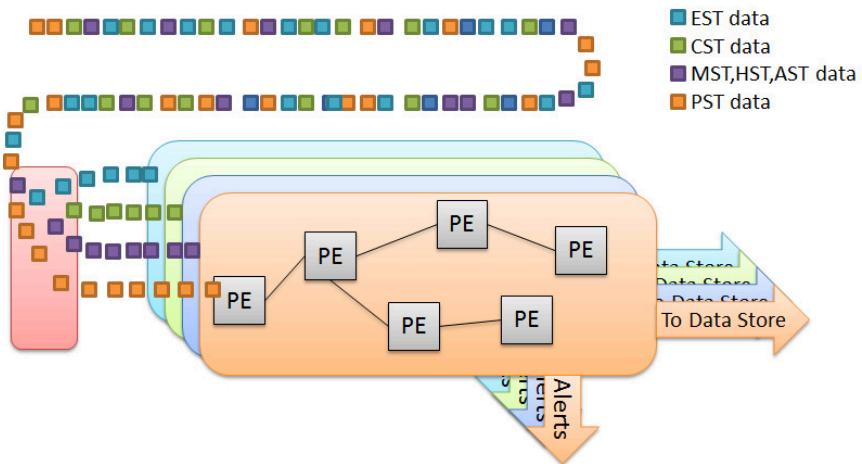
Difference between batch, micro-batch and stream processing



Stream processing



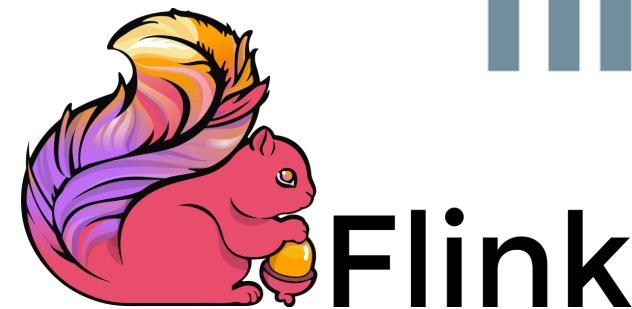
1. A simple stream processing setup



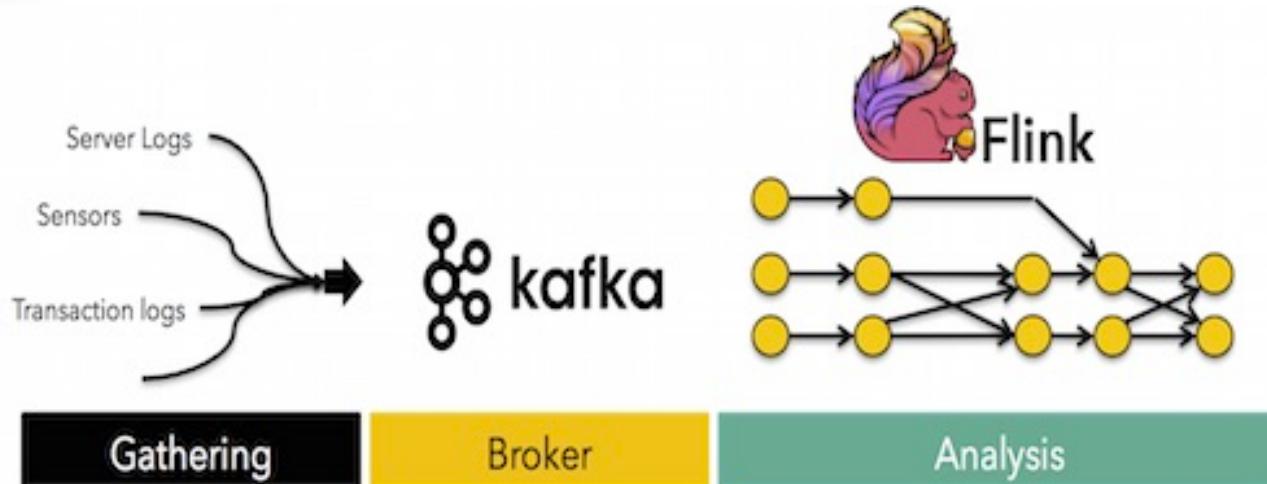
2. Partitioning Streams

- Important application design choices:
 - Avoid expensive external calls and dependencies
 - Avoid reprocessing historic events
 - Pull data immediately as they become available
 - Avoid complex time-consuming calculations as part of message processing
 - Discard unnecessary data
 - Carefully design topic partitions

Data stream processing platforms

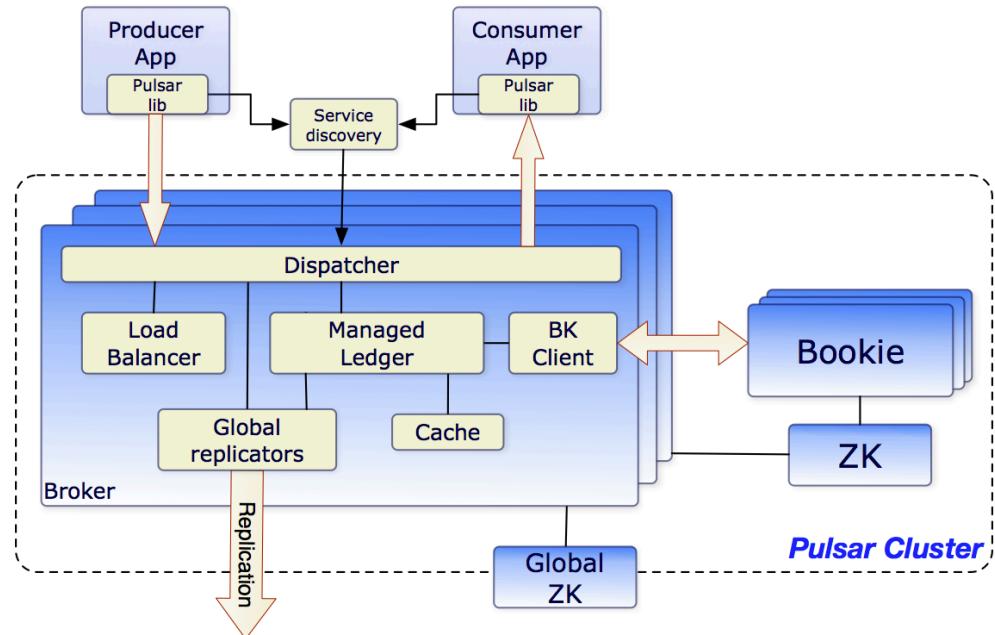


Combination of different streaming frameworks



- Apache Flink supports batch analytics, continuous stream analytics, as well as machine learning and graph processing natively on top of a streaming engine
- Combination of different frameworks also allows to build highly scalable and robust processing platform

- Components
 - Clusters
 - Broker(s)
 - HTTP server
 - Dispatcher
 - Service discovery
 - Bookies (persistent store for messages)
 - ZooKeeper (metadata store)
 - BookKeeper instances (Bookies)





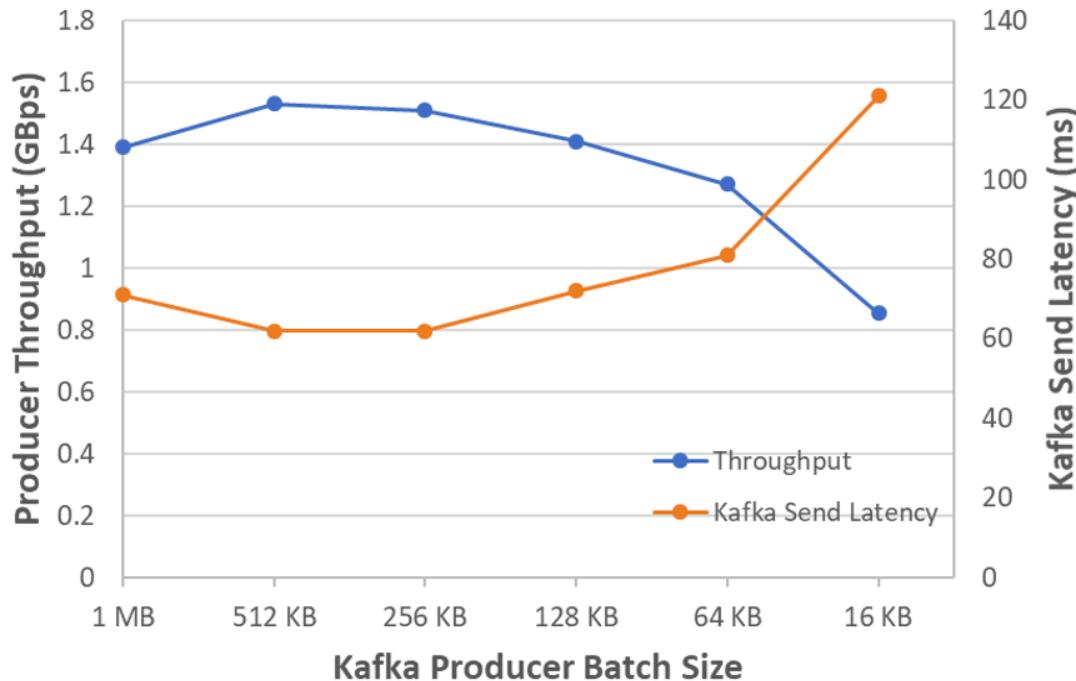
UPPSALA
UNIVERSITET

Two case studies

Data Stream frameworks:

- FenceX project
<https://github.com/bmd007/statefull-geofencing-faas>
- HarmonicIO
https://github.com/Snapple49/HarmonicIO/tree/adding_irm_real

Large vs small data objects



- Performance of Kafka cluster at Microsoft Azure infrastructure

HarmonicIO

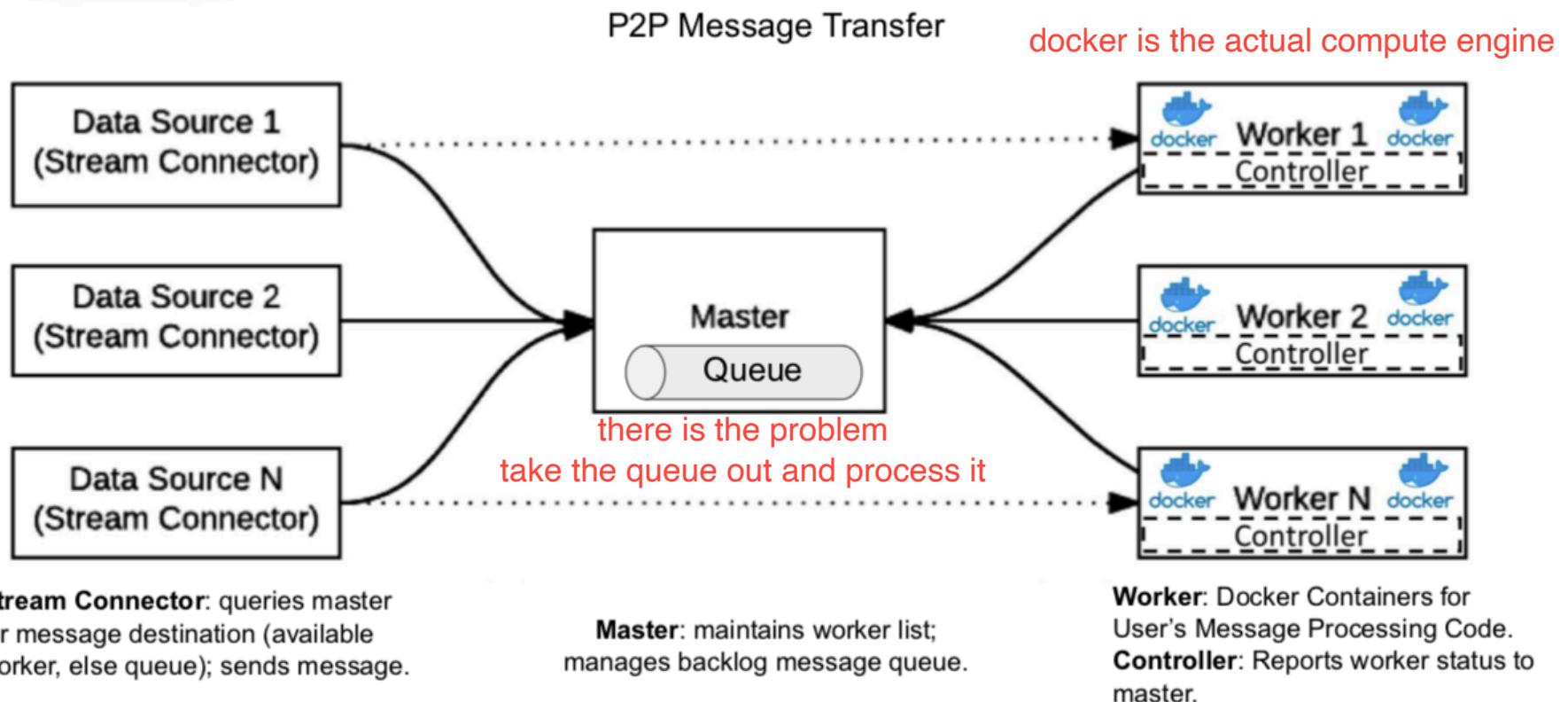
The aim of this study is to address the challenges related to smart and efficient resource utilization while processing large data streams based on scientific experiments.

Features:

- **High Throughput:** P2P transfer whenever possible, with processing engines on multiple worker nodes.
- **Smart Architecture:** Fallback message backlog queue absorbs fluctuations in source and/or processing rates, without blocking.
- **Custom Environment:** The user's application, encapsulated inside a Docker container, with very minimal dependency on HarmonicIO.
- **Simplicity:** almost zero configuration, robust; readily extensible - an ideal platform for research prototyping.



Framework architecture



Initial scalability approach

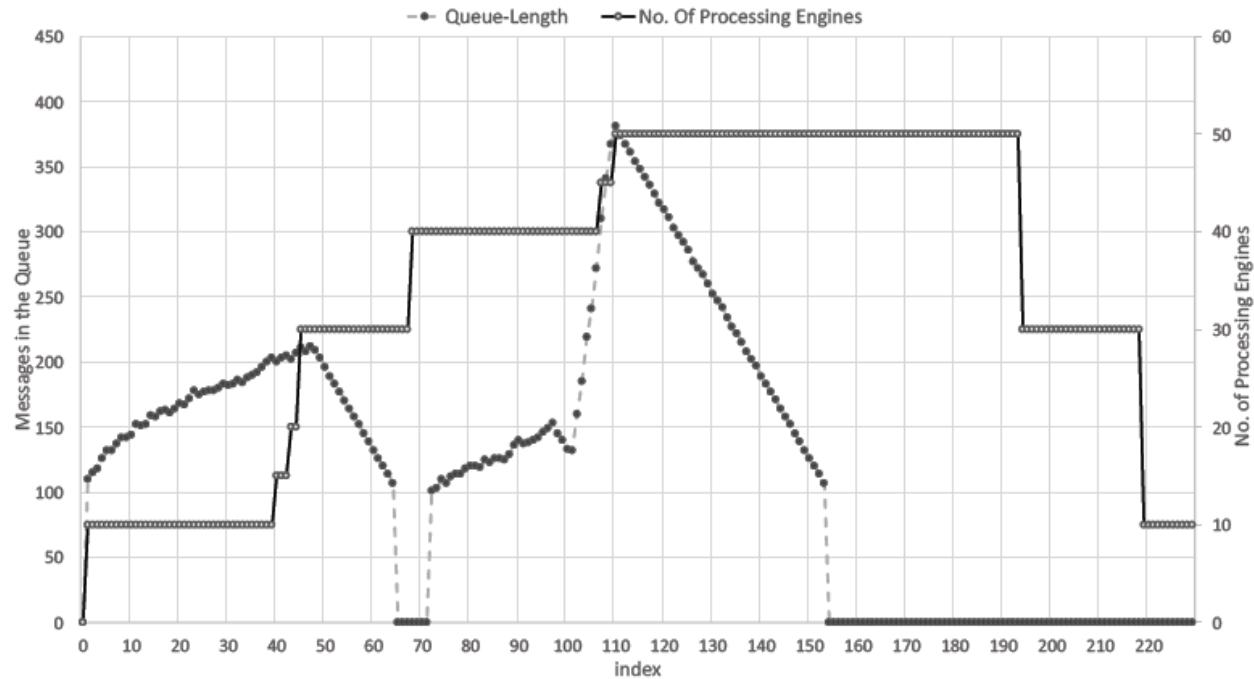


Fig. 7. Dynamic load management using different number of workers. The x-axis shows the time index and the two y-axes presents the messaging queue and the number of running containers in the framework.

Performance comparison

- Comparison based on maximum throughput frequency
- Frameworks:
 - Apache Spark framework
 - TCP streaming
 - File streaming
 - Kafka integration
 - HarmonicIO framework

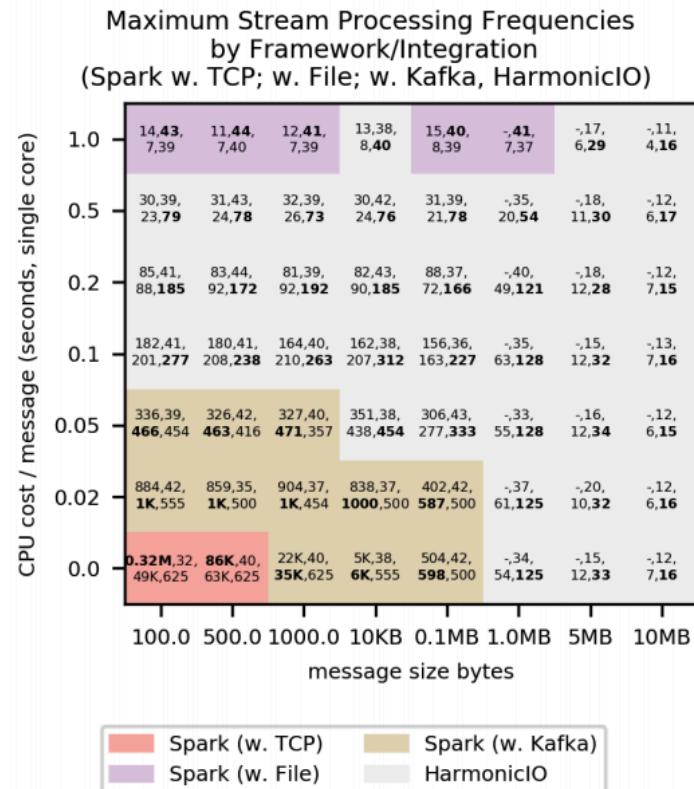


Fig. 3: Performance of Apache Spark (under TCP, File streaming, and Kafka integrations), and HarmonicIO, by maximum message throughput frequency over the domain under study. Under each setting, the highest frequency is shown in bold, with color-coding according to which framework/integration was able to achieve the best frequency. Compare with Fig. 1.



UPPSALA
UNIVERSITET

HarmonicIO - data stream framework

RECENT IMPROVEMENTS

Resource management

One lacking aspect in HarmonicIO was **resource management capabilities**. There are multiple challenges to address:

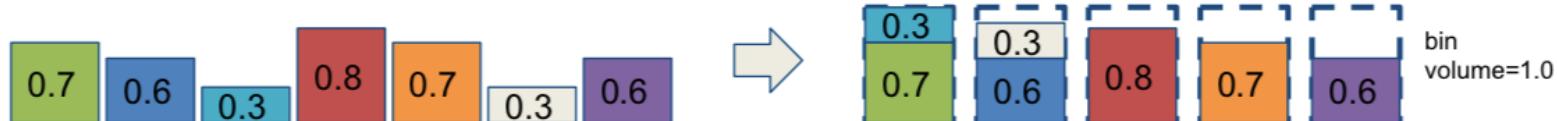
- Utilizing the computation resources efficiently
- Scaling the stream processing capacity based on the load
- Automatic resource management

Dynamic Online Bin-Packing

Bin-Packing: pack items of varying volumes into bins or containers with a fixed volume while keeping the number of bins needed to a minimum

Problem variations and challenges:

- **Online Bin-Packing:** pack items as they arrive
 - No advance information about item sizes
- **Dynamic Bin-Packaging:** items may be removed from inside container
 - Removed by external factors (time, user interaction etc.)
 - Defragmented bins
- **The First-Fit algorithm:** find first available bin that can fit item; else create new
- NP-complete
 - **Complexity:** $O(n \log n)$ time, $O(n)$ space

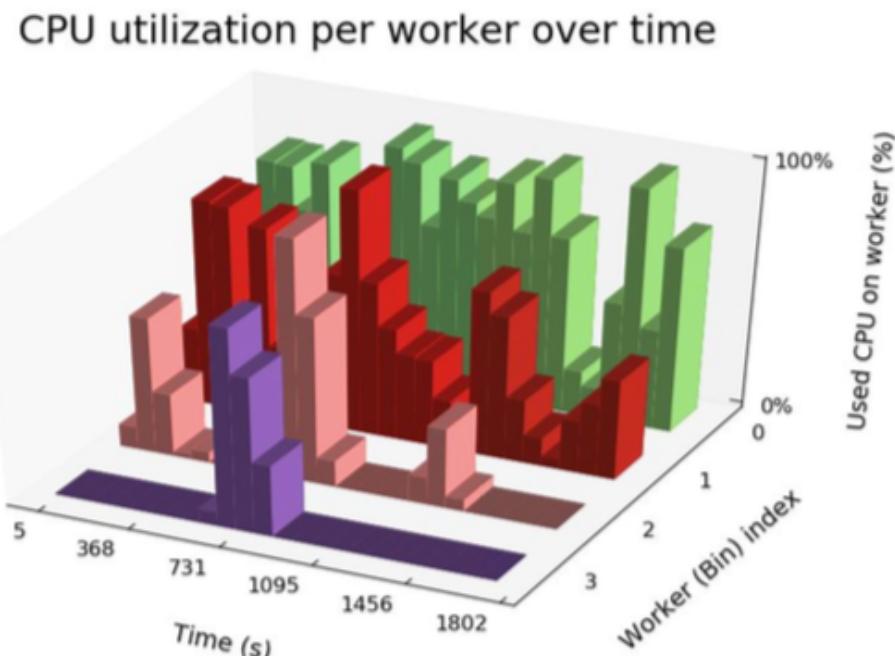


HarmonicIO with Bin-Packing

- We utilize Bin-Packing to maximize the resource utilization and add autoscaling of stream processing engines to optimally match the stream load?
- Introducing the Intelligent Resource Manager (IRM) to HarmonicIO:
 - Modeling processing engines (PEs) as items and workers as bins
 - Bin-Packing outcome indicates how many PEs needed and where to host
 - Enables autoscaling of PEs and workers
 - Profiling workloads run-time provides estimation of required resources
 - Average CPU usage
 - Configurable scaling response data streaming pressure

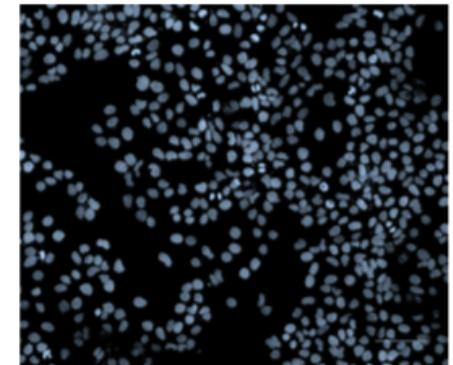
Results: synthetic workloads

- **Test scenario:** Synthetic workloads targeting 100% single-core CPU usage. 3D-plot shows the CPU usage per worker over time throughout the experiment
 - Bin-Packing pushes workloads to first available worker (lowest index bin)
 - Idle time could be allocated to other tasks
 - Peaks between 90-100% CPU utilization before moving to next worker



Results: CellProfiler workload

- **Test scenario:** Image analysis of cell microscopy images with CellProfiler, courtesy of AstraZeneca



Microscopy image,
courtesy AstraZeneca

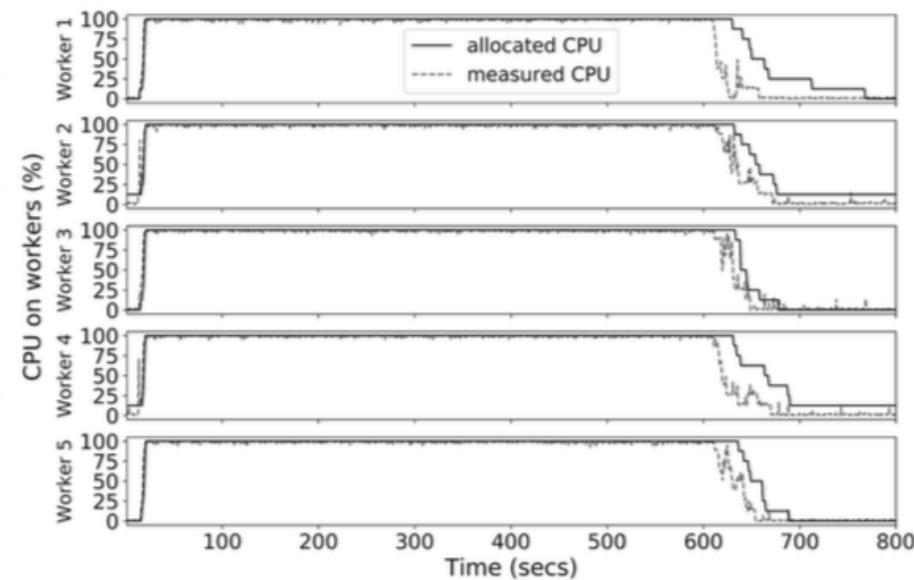
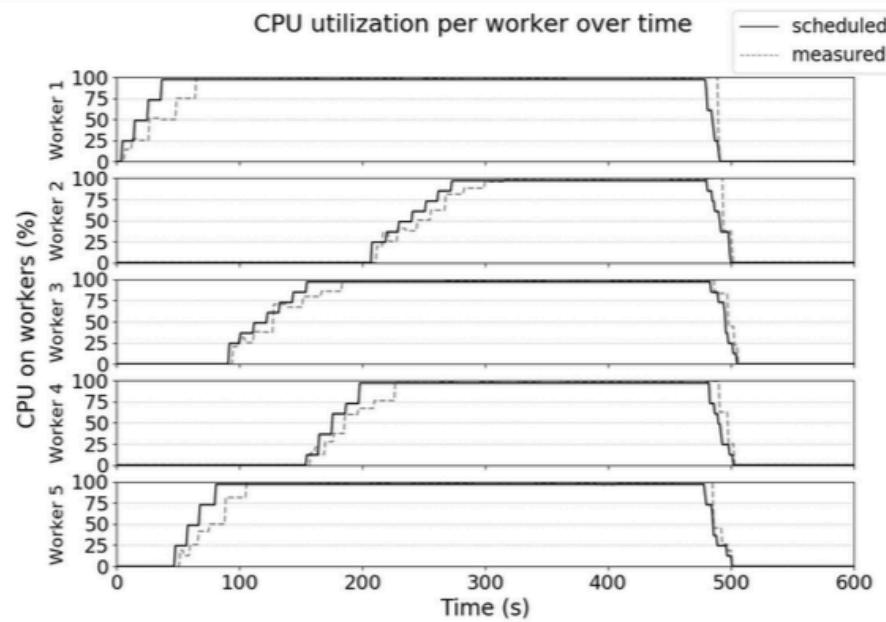
- Stream microscopy images and run analysis pipeline
 - Full single-core CPU utilization per task
- Benchmark test: comparing equivalent workload on HarmonicIO and Spark Streaming
- Comparing the allocated and measured CPU usage for both
 - Different scaling strategies



UPPSALA
UNIVERSITET

Results: CellProfiler workload

HarmonicIO



Articles

- HarmonicIO: Scalable Data Stream Processing for Scientific Datasets
<https://ieeexplore.ieee.org/document/8457894>
- Apache Spark Streaming, Kafka and HarmonicIO: A Performance Benchmark and Architecture Comparison for Enterprise and Scientific Computing
<https://arxiv.org/pdf/1807.07724.pdf>
- Smart Resource Management for Data Streaming using an Online Bin-packing Strategy
<https://ieeexplore.ieee.org/document/9378241>

What is Geofencing?

- Geofencing uses the global positioning system (GPS) and/or radio frequency identification (RFID) to create a virtual geographic boundary around a particular area
- Possible use cases:
 - Elderly care system
 - Online taxi booking
 - **Monitor the behaviour of grazing animals**
- It is also known as High-tech animal tracking technique





UPPSALA
UNIVERSITET

Types of Geofencing

- On-demand geofencing
 - Administrators can set new fences based on different needs
 - Example, farmer needs to set a new area for grazing
- Real-time geofencing
 - Fences automatically get created based on different requirements
 - Example: Online taxi booking system



UPPSALA
UNIVERSITET

Pros and Cons of Geofencing

- Pros
 - Effective animal tracking for outdoor grazing
 - Accurate data collection based on the animals in forests
 - Reliable interaction traces between animals. Such datasets are helpful to study disease spreads or social interaction between animals
- Cons
 - Highly dependent on the sensor technology
 - Change of batteries can be a serious challenge
 - Security threats
 - Good communication is required



UPPSALA
UNIVERSITET

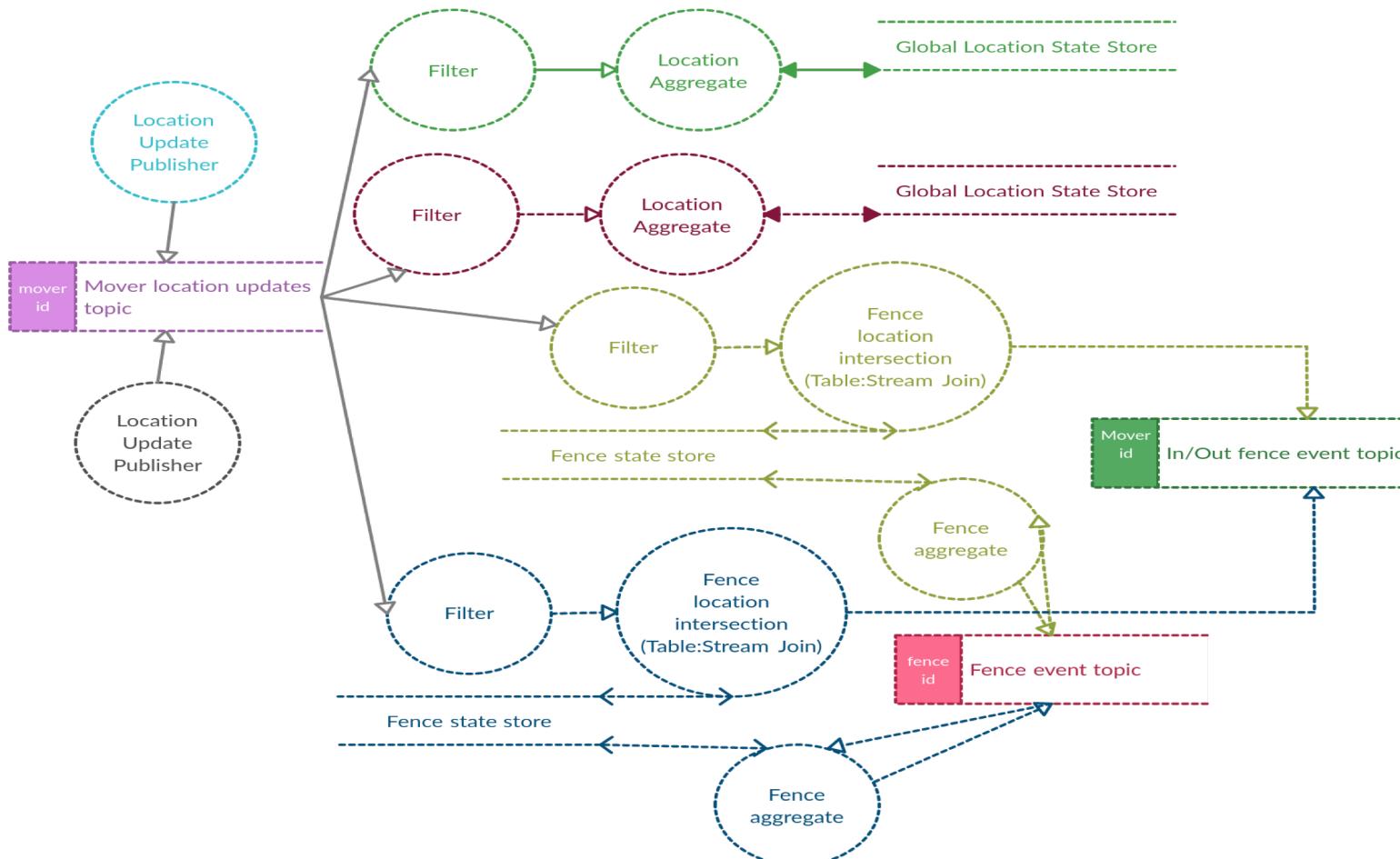
FenceX Framework

- FenceX is an open source software framework for geofencing based on GPS based coordinate system
- It is a data stream processing framework to handle a large number of events in real-time
- The aim is to design a highly scalable, resilient and efficient software framework for geofencing



UPPSALA
UNIVERSITET

FenceX Architecture

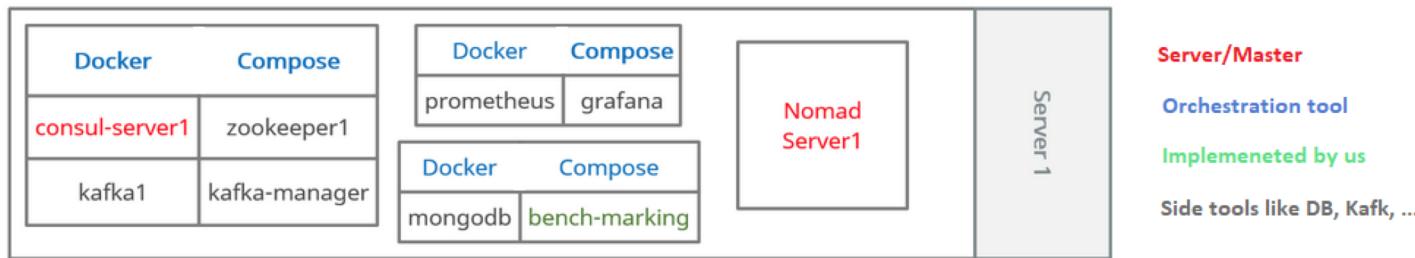




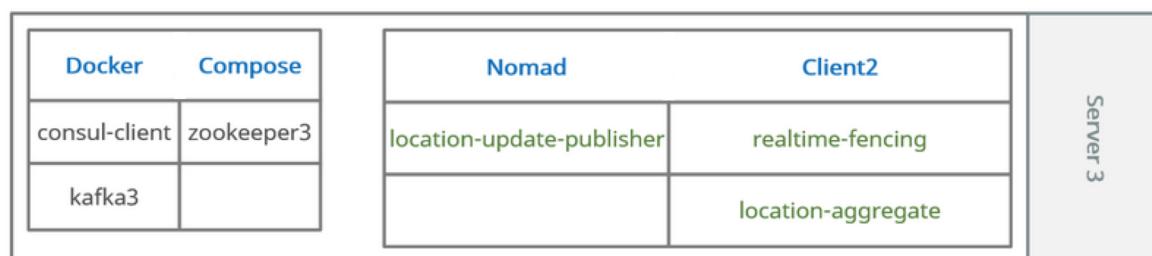
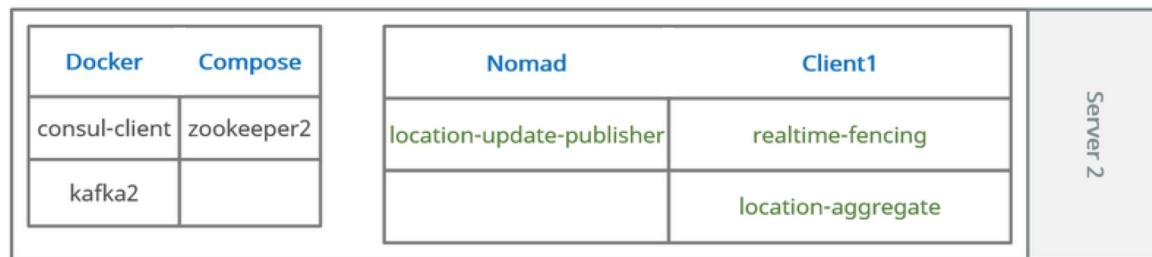
UPPSALA
UNIVERSITET

Deployment Settings

- 4 xlarge
Snic cloud VMs



- +3Gbps
bandwidth
- 16GB of RAM for
workers
- +16GB of RAM



Framework Evaluation

- Throughput



- Availability



- Scalability





UPPSALA
UNIVERSITET

Throughput

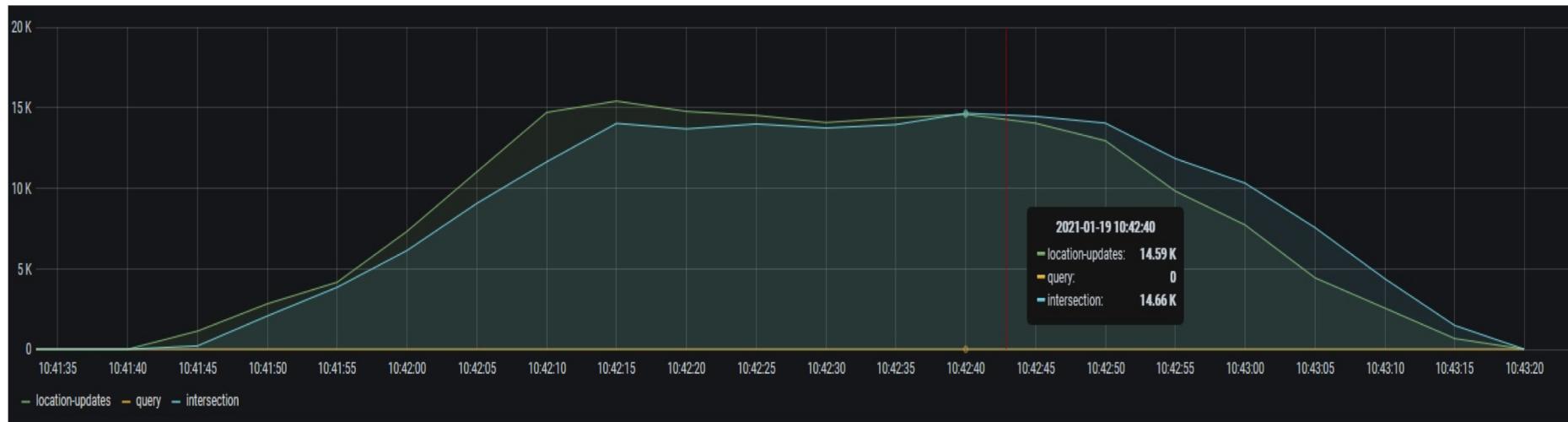
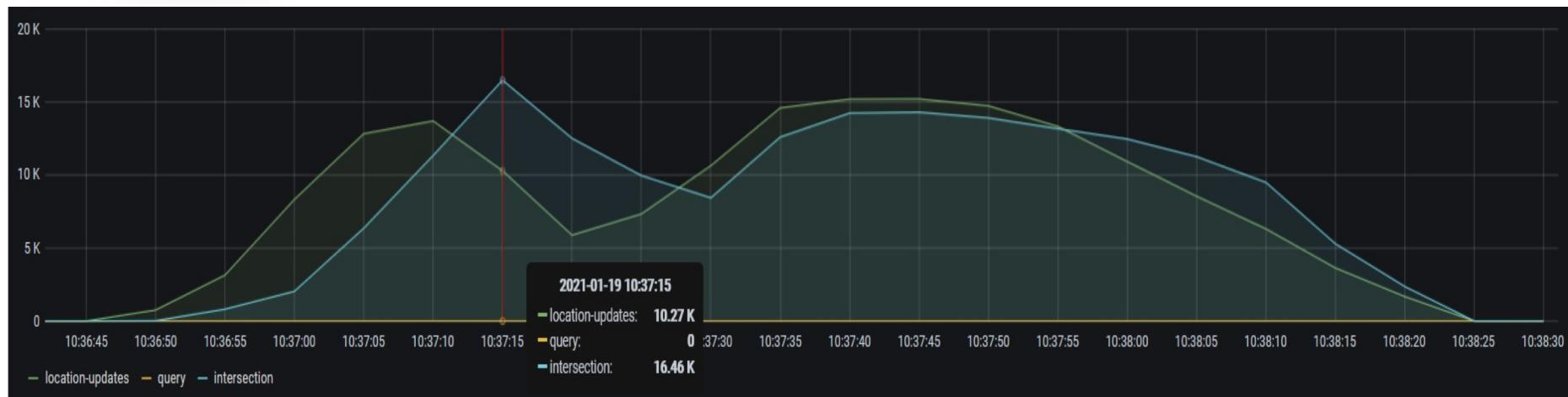
System's ability to successfully run the operations and transfer data from one location to another in a given time.

Features	Input	Throughput
Push	Location update report	Number of fence-point intersections /sec
Pull	Query by fence request	Number of successfully answered queries /sec



UPPSALA
UNIVERSITET

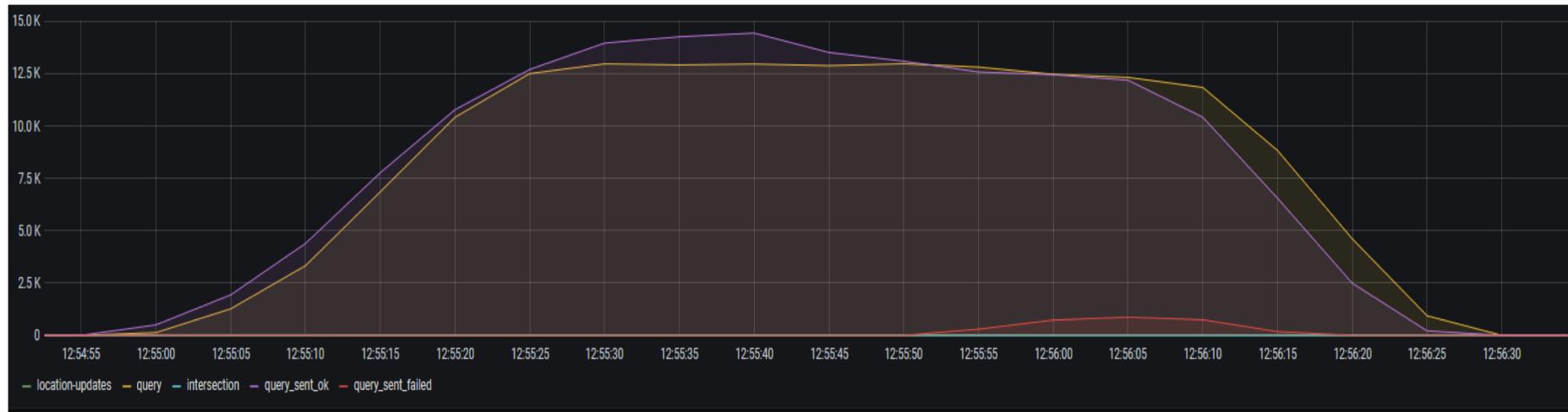
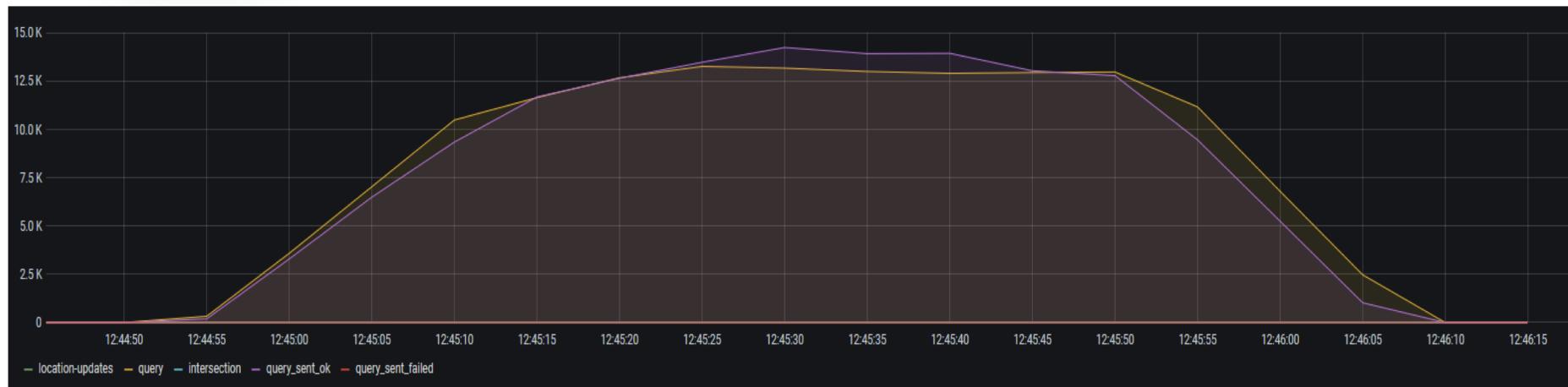
Throughput Push case





UPPSALA
UNIVERSITET

Throughput Pull case





UPPSALA
UNIVERSITET

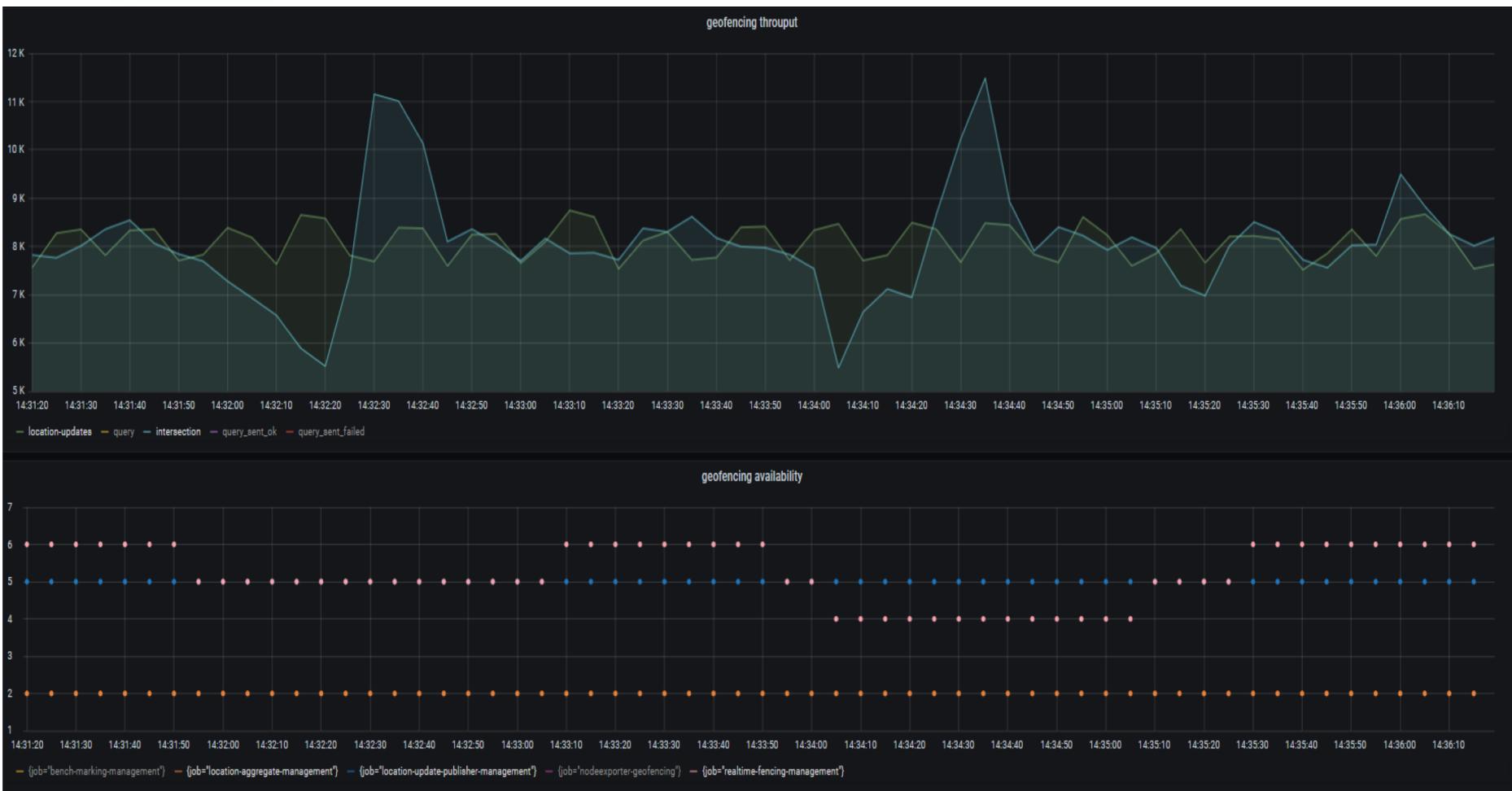
Availability

The probability that the system does not experience many failures in a given time.



UPPSALA
UNIVERSITET

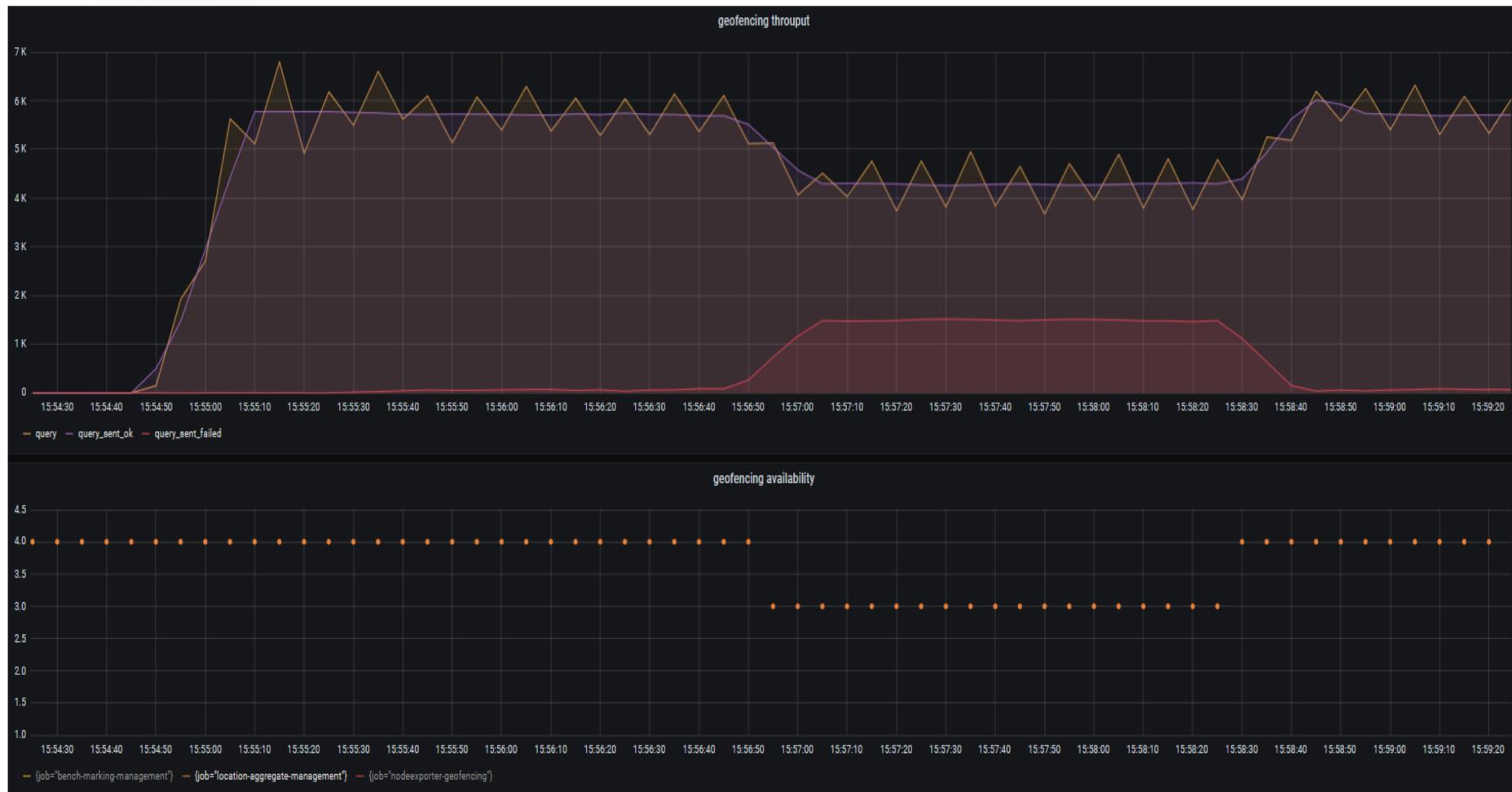
Availability Push case





UPPSALA
UNIVERSITET

Availability Pull case





UPPSALA
UNIVERSITET

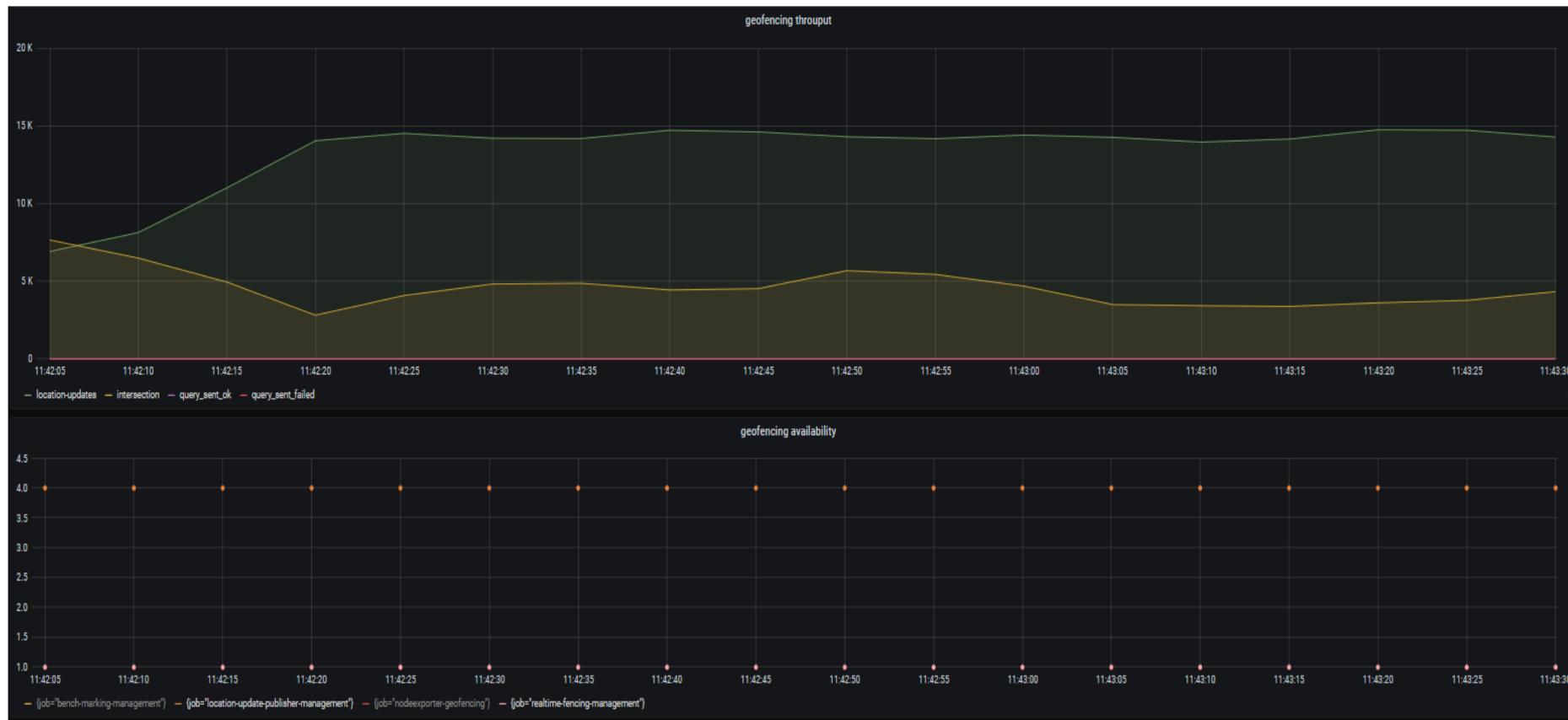
Strong Scalability

The ability of a system to efficiently handle the growing amount of workload.



UPPSALA
UNIVERSITET

Strong Scalability Push case

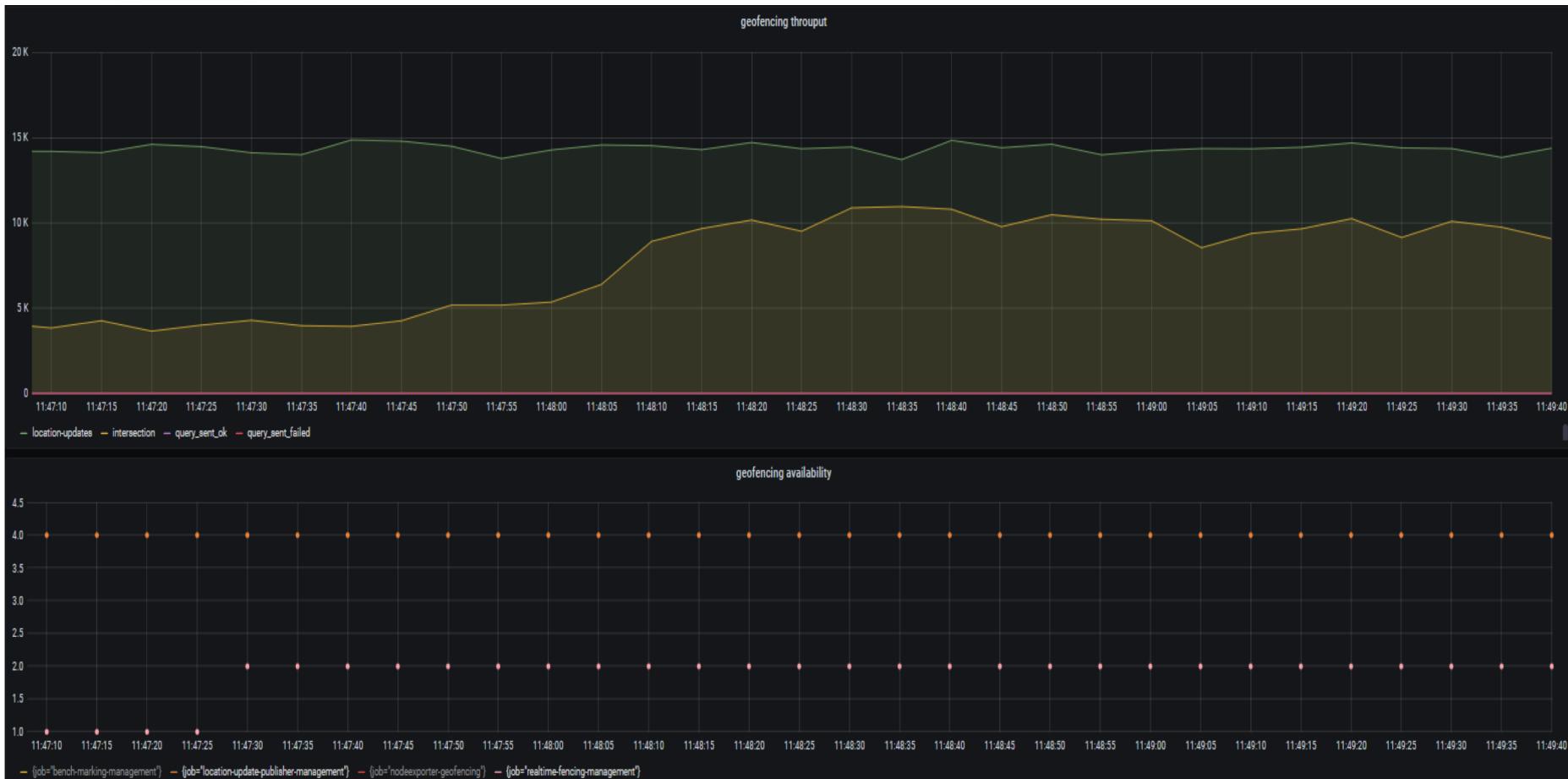


Processing nodes = 1



UPPSALA
UNIVERSITET

Strong Scalability Push case

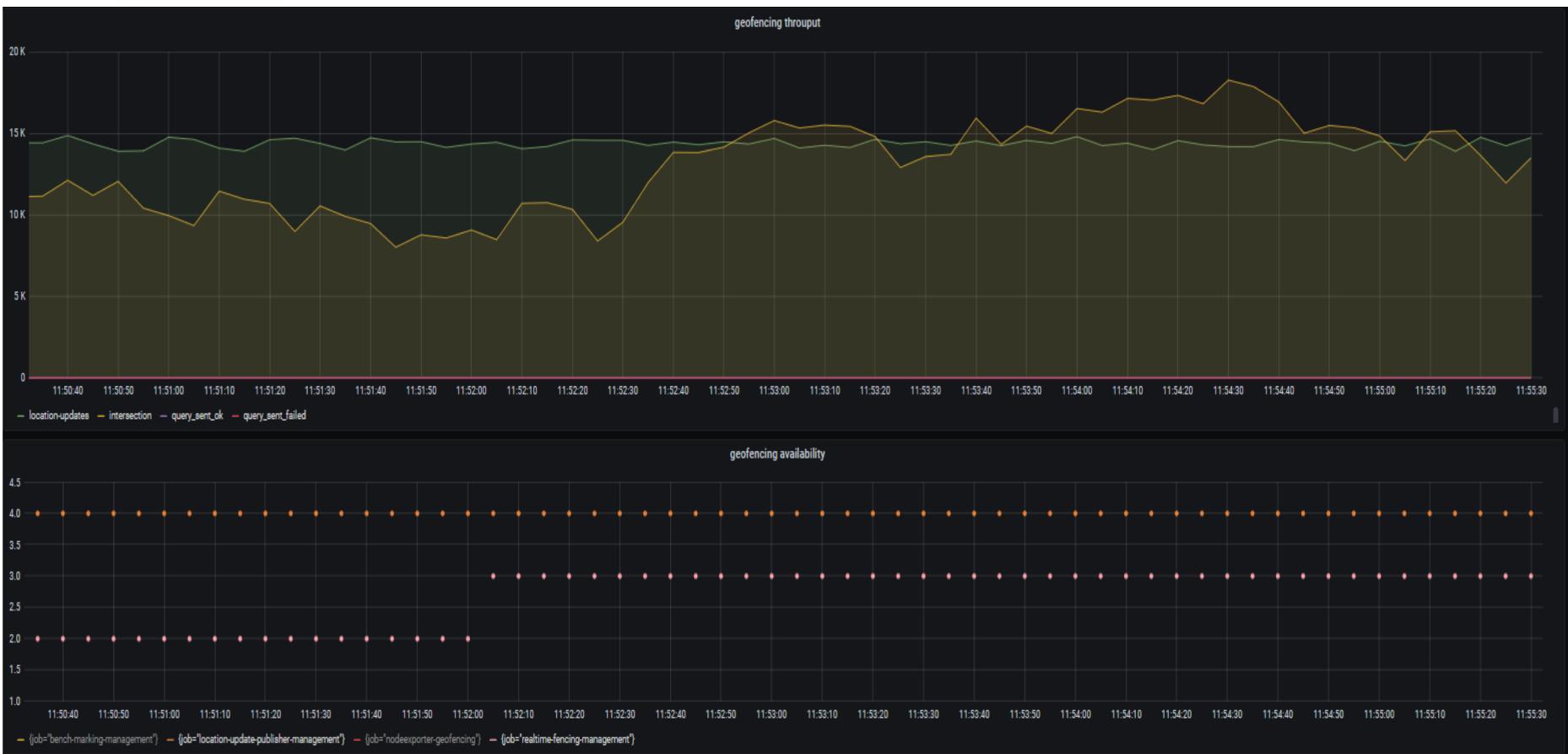


Processing nodes = 2



UPPSALA
UNIVERSITET

Strong Scalability Push case

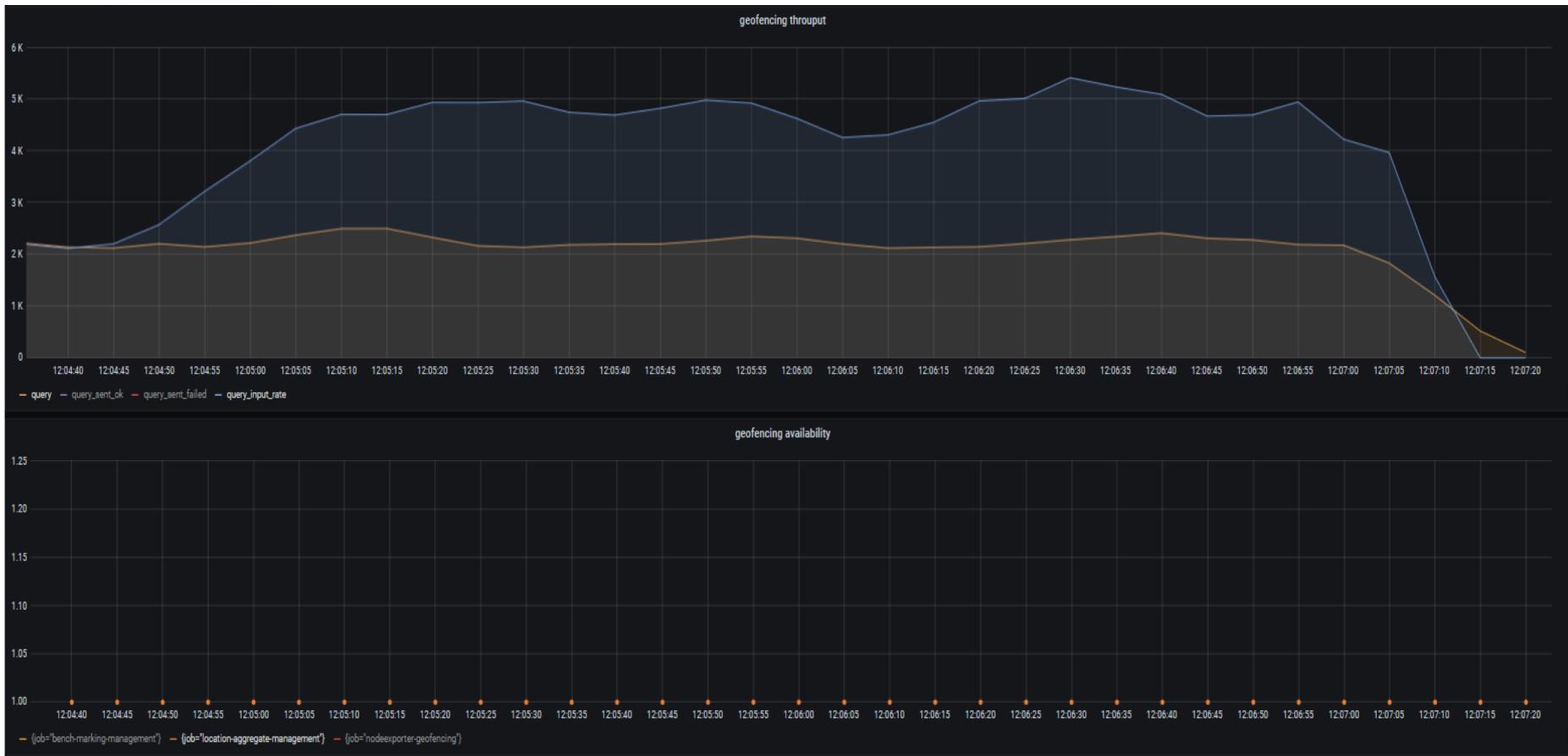


Processing nodes = 3



UPPSALA
UNIVERSITET

Strong Scalability Pull case

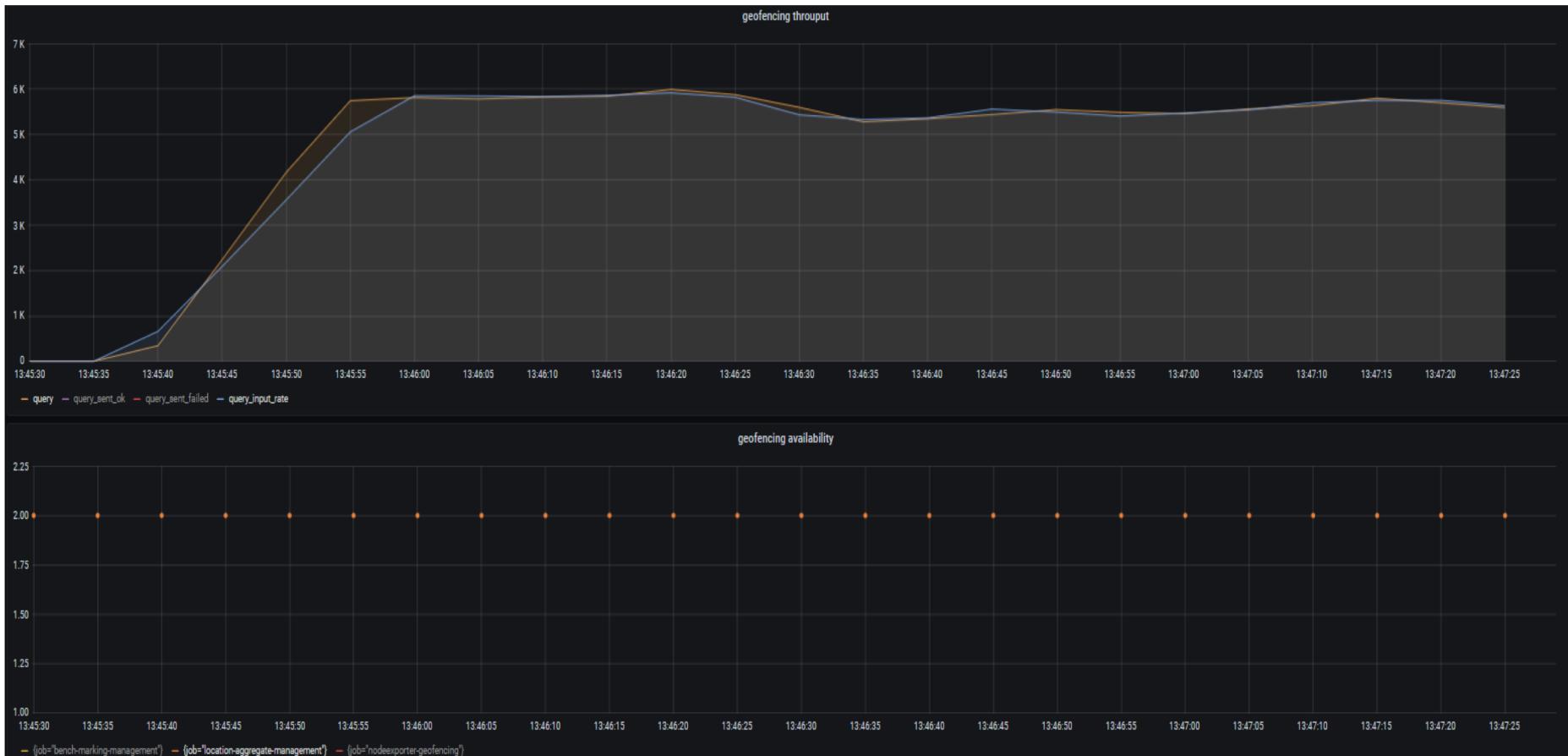


Processing nodes = 1



UPPSALA
UNIVERSITET

Strong Scalability Pull case



Processing nodes = 2

Why are these features important for the domain experts?

- A number of commercial companies offer frameworks for herd management using geofencing
- It is important to have an easy-to-use platform with accessibility based on different devices (mobiles, tablets and laptops). At the same time, it is also important to understand the underlying technical solution in terms of the following features:
 - Data storage and availability plan
 - Resource usage of the platform
 - Comparison with similar platforms based on
 - scalability (horizontal and vertical)
 - throughput
 - high availability
 - and security
- All these factors affect the cost of using a framework
- Good understanding of these features will help to better estimate the yearly budget