UPPSALA
UNIVERSITET

Data Engineering - II
Course code: 1TD075 62033

M1 - Data Stream Processing (Part-II)

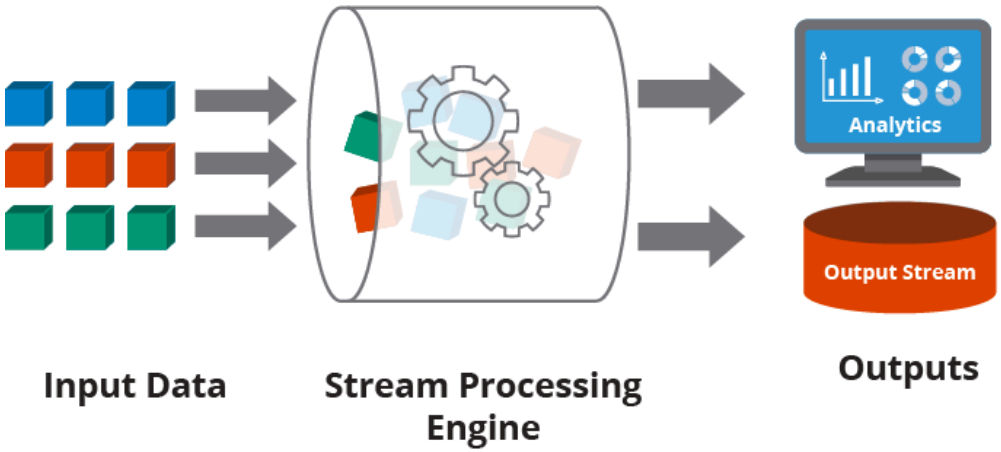
Salman Toor
salman.toor@it.uu.se

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

UPPSALA UNIVERSITET

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

https://hazelcast.com/glossary/stream-processing/

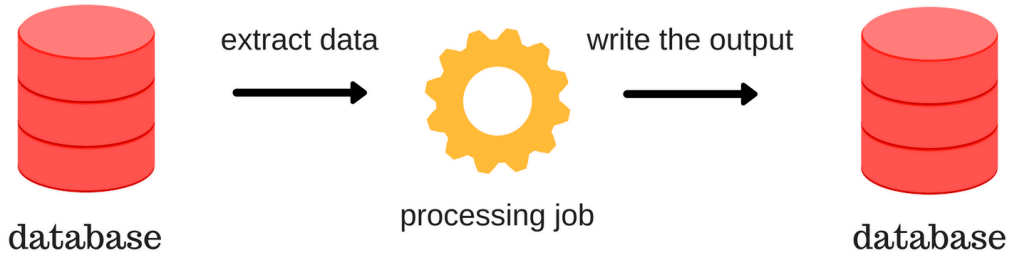
UPPSALA UNIVERSITET

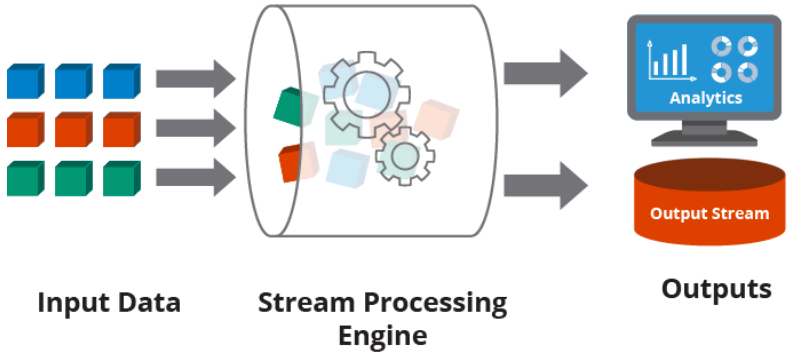
Stream processing

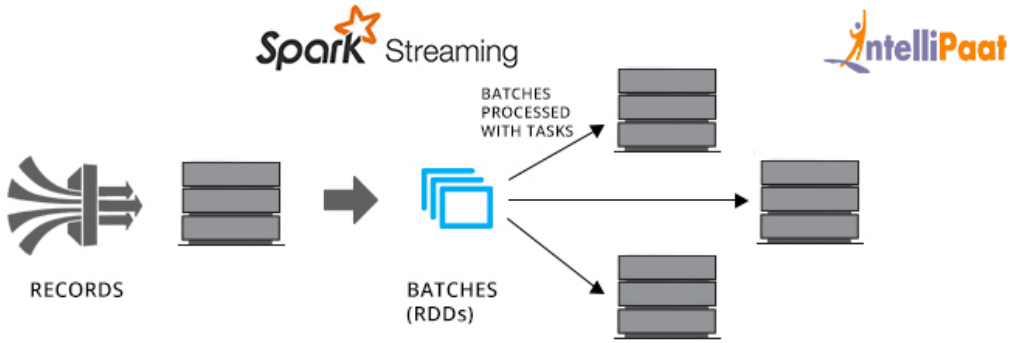
- Challenges
  - It is not possible to find the exact solution for the functions such as min or max
  - All blocking operations are difficult to execute in stream processing
- Active research directions
  - Approximate query processing techniques
  - Novel architectures for blocking operators
  - Reliable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

UPPSALA UNIVERSITET

Difference between batch, micro-batch and stream processing

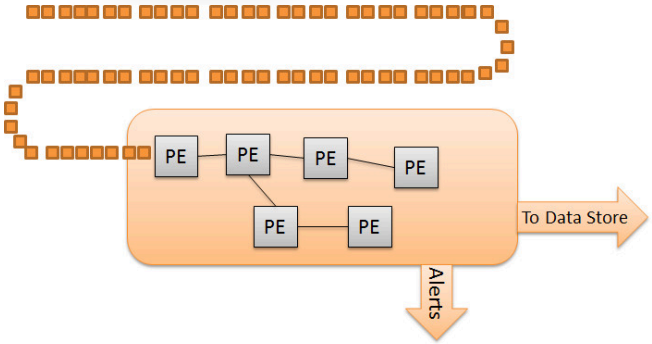




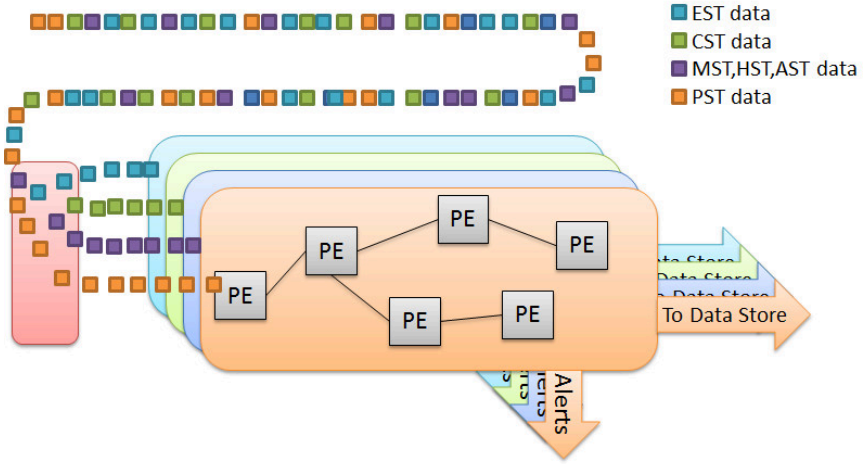


UPPSALA UNIVERSITET

Stream processing



1. A simple stream processing setup



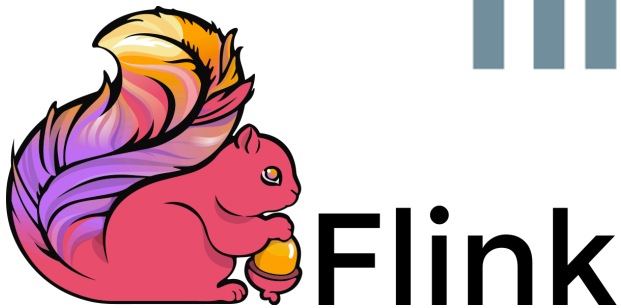
2. Partitioning Streams

- 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

https://www.linkedin.com/pulse/6-important-concepts-stream-processing-chandra-rajasekharaiah/

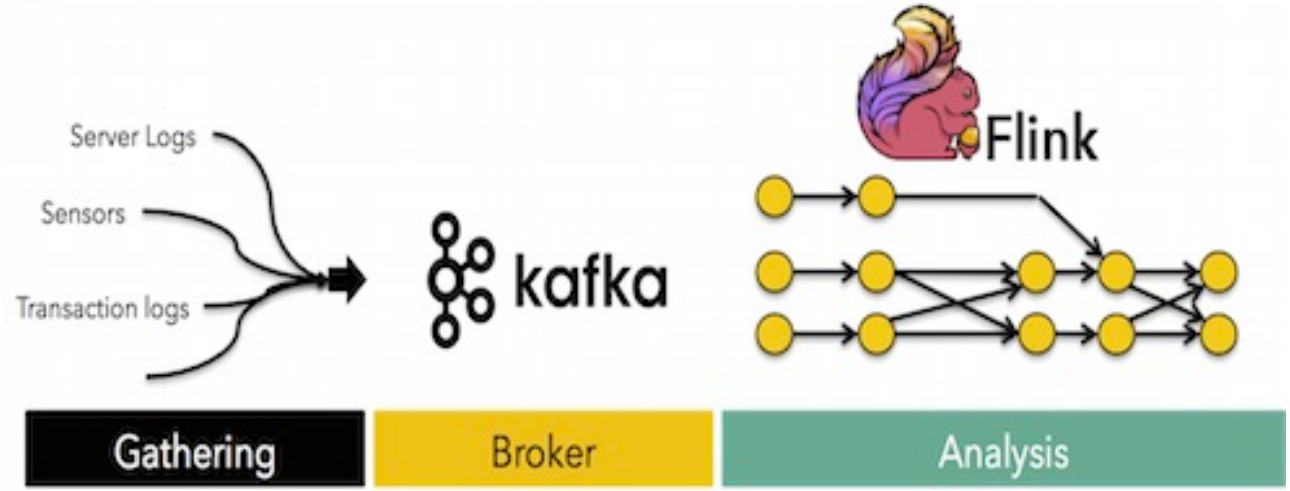
UPPSALA UNIVERSITET

Data stream processing platforms



UPPSALA UNIVERSITET

Combination of different streaming frameworks

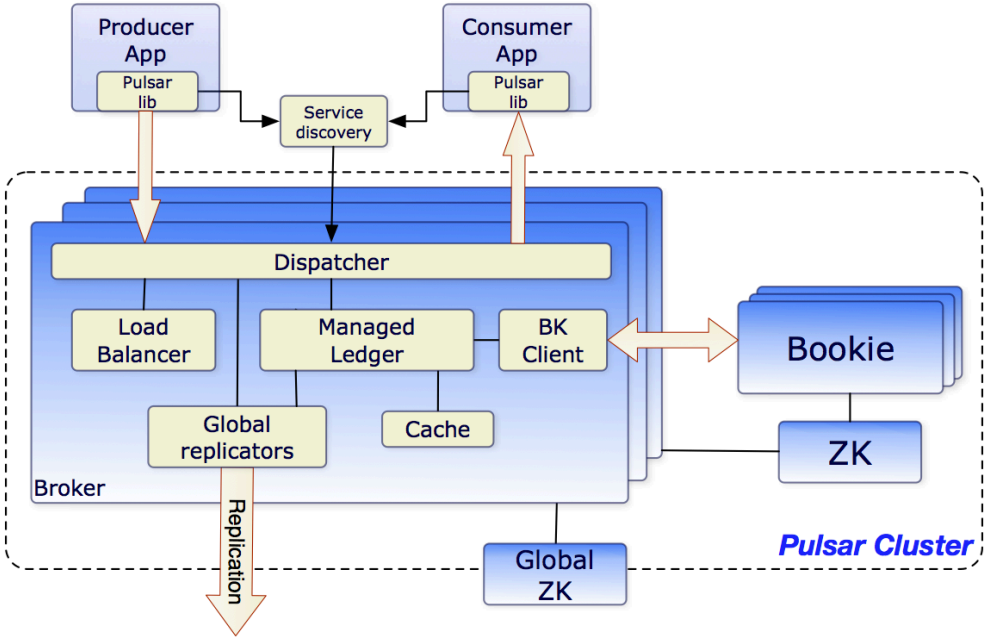


- Apache Flink supports batch analytics, continuous stream analytics, as well as machine learning and graph processing natively on top of a streaming engine
- Combination of different frameworks also allows to build highly scalable and robust processing platform

UPPSALA UNIVERSITET

PULSAR

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



https://pulsar.apache.org/docs/en/concepts-architecture-overview/

UPPSALA UNIVERSITET

Two case studies

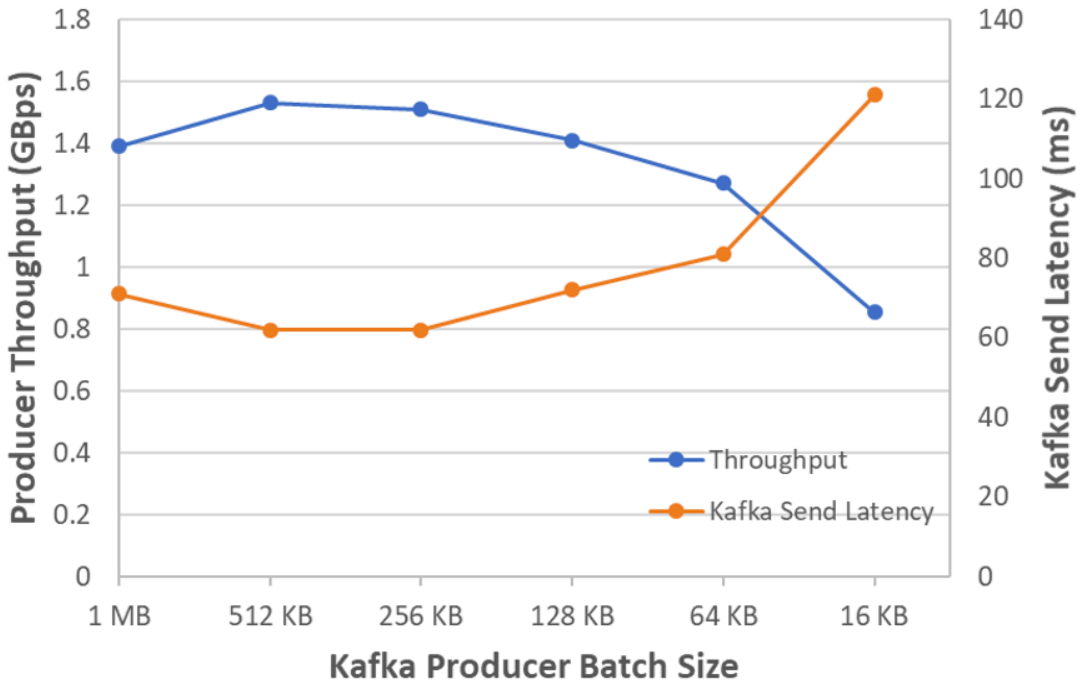
Data Stream frameworks:

- FenceX project
  https://github.com/bmd007/statefull-geofencing-faas

- HarmonicIO
  https://github.com/Snapple49/HarmonicIO/tree/adding_irm_real

UPPSALA UNIVERSITET

Large vs small data objects



- Performance of Kafka cluster at Microsoft Azure infrastructure

https://azure.microsoft.com/en-us/blog/processing-trillions-of-events-per-day-with-apache-kafka-on-azure/

UPPSALA UNIVERSITET

HarmonicIO

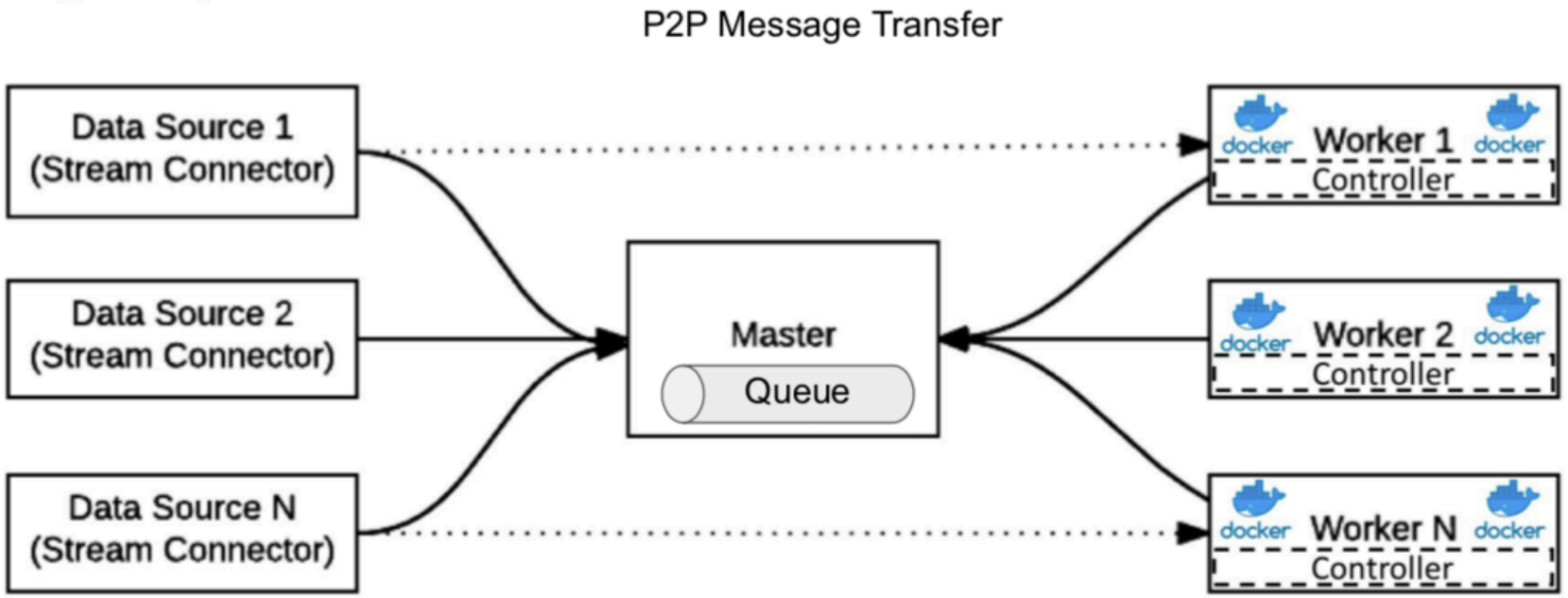
The aim of this study is to address the challenges related to smart and efficient resource utilization while processing large data streams based on scientific experiments.

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.

UPPSALA UNIVERSITET

Framework architecture



Stream Connector: queries master for message destination (available worker, else queue); sends message.

Master: maintains worker list; manages backlog message queue.

Worker: Docker Containers for User's Message Processing Code.
Controller: Reports worker status to master.

UPPSALA UNIVERSITET

Initial scalability approach

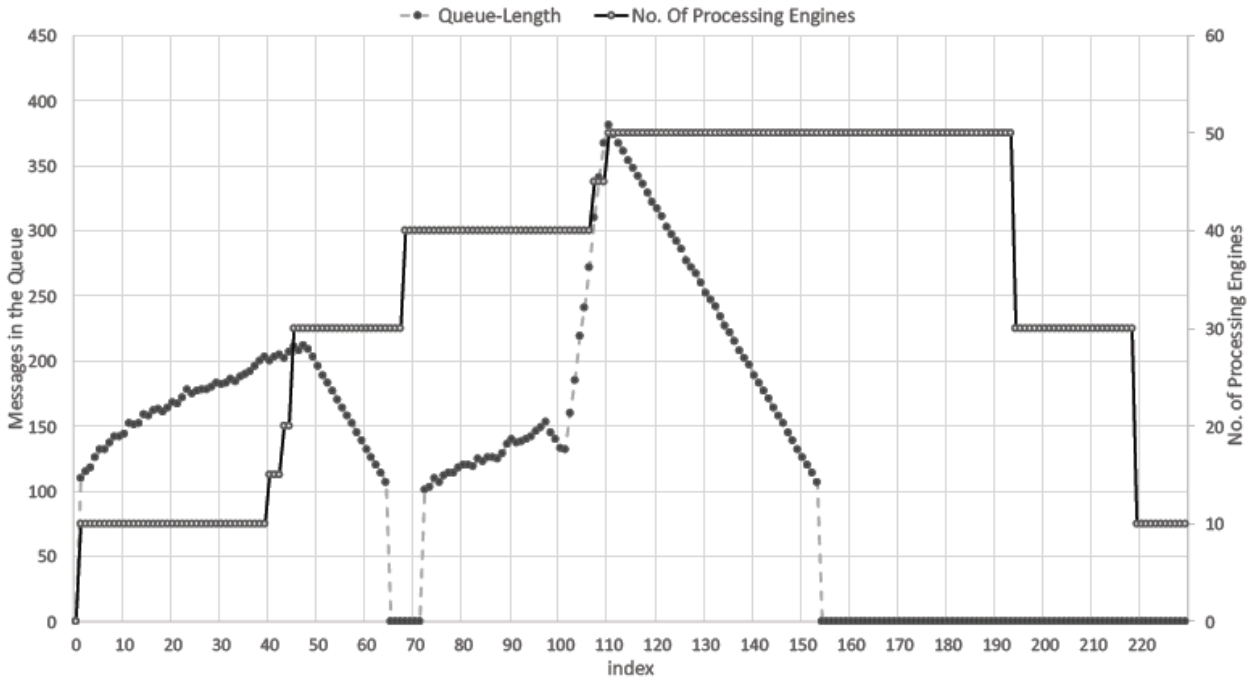


Fig. 7. Dynamic load management using different number of workers. The x-axis shows the time index and the two y-axes presents the messaging queue and the number of running containers in the framework.

UPPSALA UNIVERSITET

Performance comparison

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

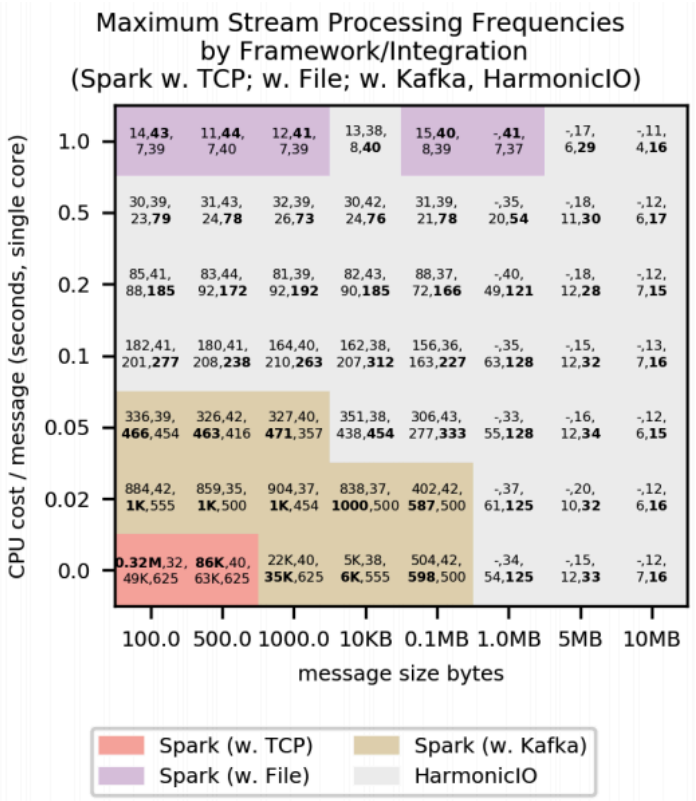


Fig. 3: Performance of Apache Spark (under TCP, File streaming, and Kafka integrations), and HarmonicIO, by maximum message throughput frequency over the domain under study. Under each setting, the highest frequency is shown in bold, with color-coding according to which framework/integration was able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

Apache Spark Streaming, Kafka and HarmonicIO: A Performance Benchmark and Architecture Comparison for Enterprise and Scientific Computing, https://arxiv.org/pdf/1807.07724.pdf

UPPSALA
UNIVERSITET

HarmonicIO - data stream framework

RECENT IMPROVEMENTS

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

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-Packing:** 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

UPPSALA UNIVERSITET

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?

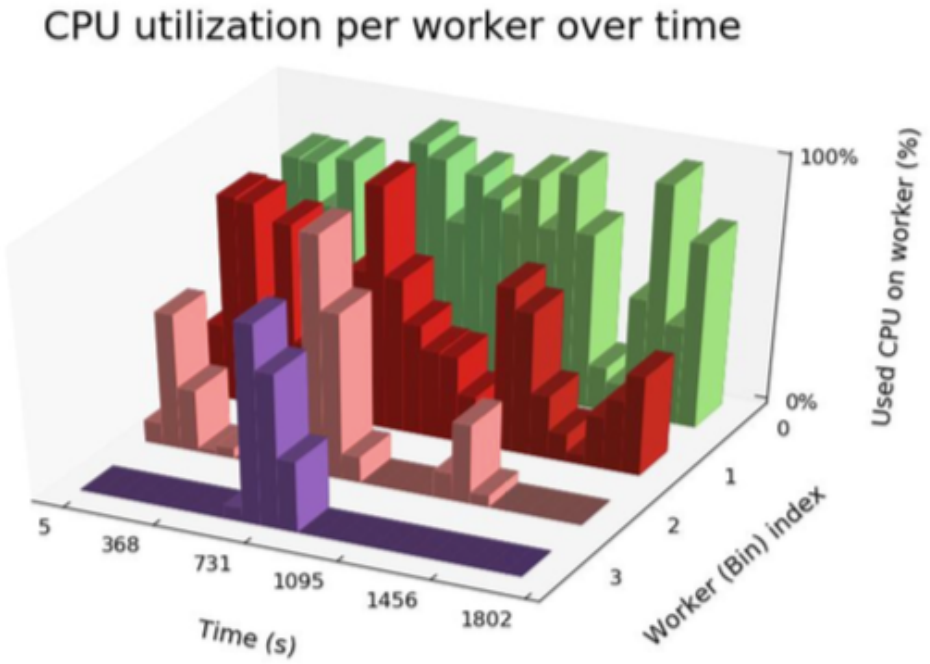
- Introducing the Intelligent Resource Manager (IRM) to HarmonicIO:
  - Modeling processing engines (PEs) as items and workers as bins
  - Bin-Packing outcome indicates how many PEs needed and where to host
    - Enables autoscaling of PEs and workers
  - Profiling workloads run-time provides estimation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

Smart Resource Management for Data Streaming using an Online Bin-packing Strategy
https://ieeexplore.ieee.org/document/9378241

UPPSALA UNIVERSITET

Results: synthetic workloads

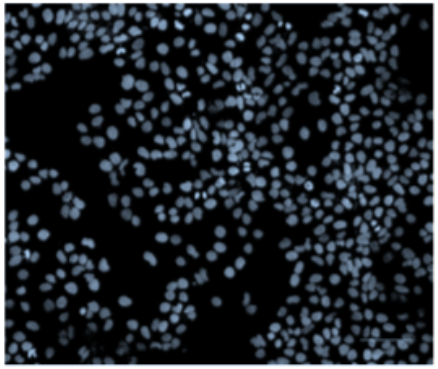
- **Test scenario:** Synthetic workloads targeting 100% single-core CPU usage. 3D-plot shows the CPU usage per worker over time throughout the experiment
  - Bin-Packing pushes workloads to first available worker (lowest index bin)
  - Idle time could be allocated to other tasks
  - Peaks between 90-100%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



UPPSALA UNIVERSITET

Results: CellProfiler workload

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



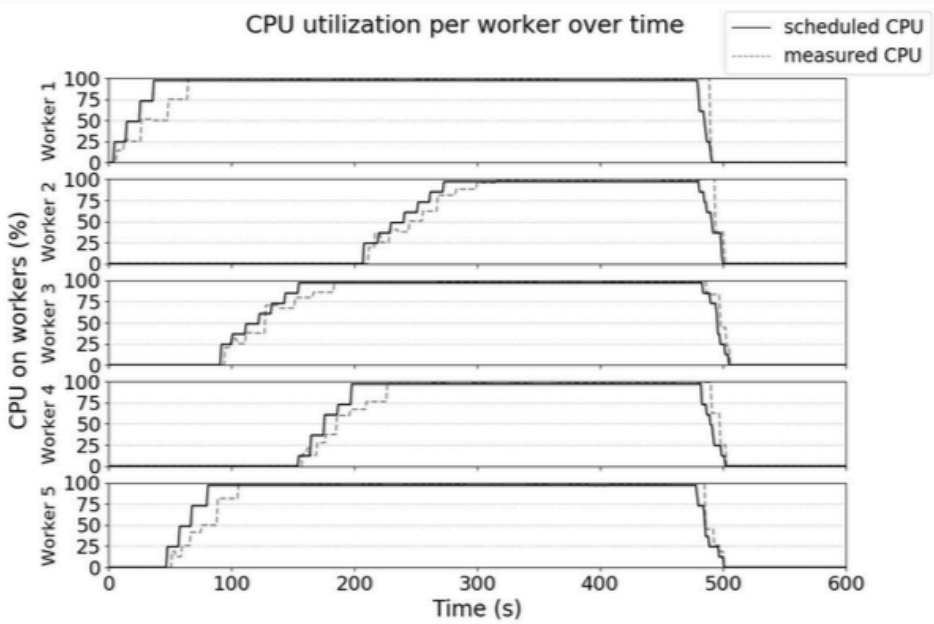
Microscopy image, courtesy AstraZeneca

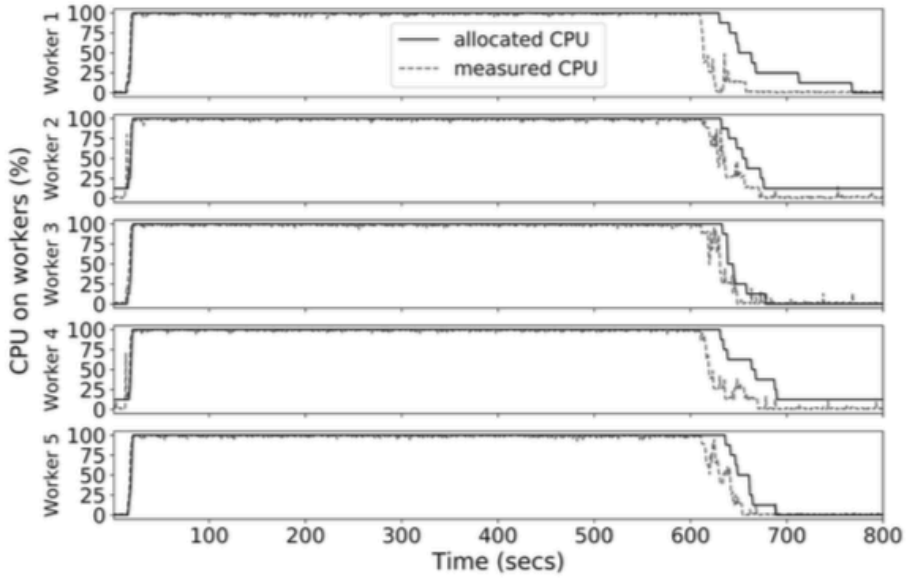
- Stream microscopy images and run analysis pipeline ○
  - Full single-core CPU utilization per task
- Benchmark test: comparing equivalent workload on HarmonicIO and Spark Streaming
- Comparing the allocated and measured CPU usage for both
  - Different scaling strategies

UPPSALA UNIVERSITET

Results: CellProfiler workload

HarmonicIO





Spark Streaming

UPPSALA UNIVERSITET

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