0-to-hero

07/10 Mentors <> 07/10 Sessions

Elkhan Shahverdi

Senior Software Engineer at Insparx Gmbh

https://www.linkedin.com/in/elkhan-shahverdi-59356166/
shahverdiyev90@gmail.com
Twitter: @shahverdiyev90

Ankara University – Computer Engineering (BSc)
University of Tartu – Software Engineering (MSc)

Estonia

Study:
https://www.studyinestonia.ee/en

Work:
https://www.workinestonia.com/
https://devhunt.ee

Stream Processing

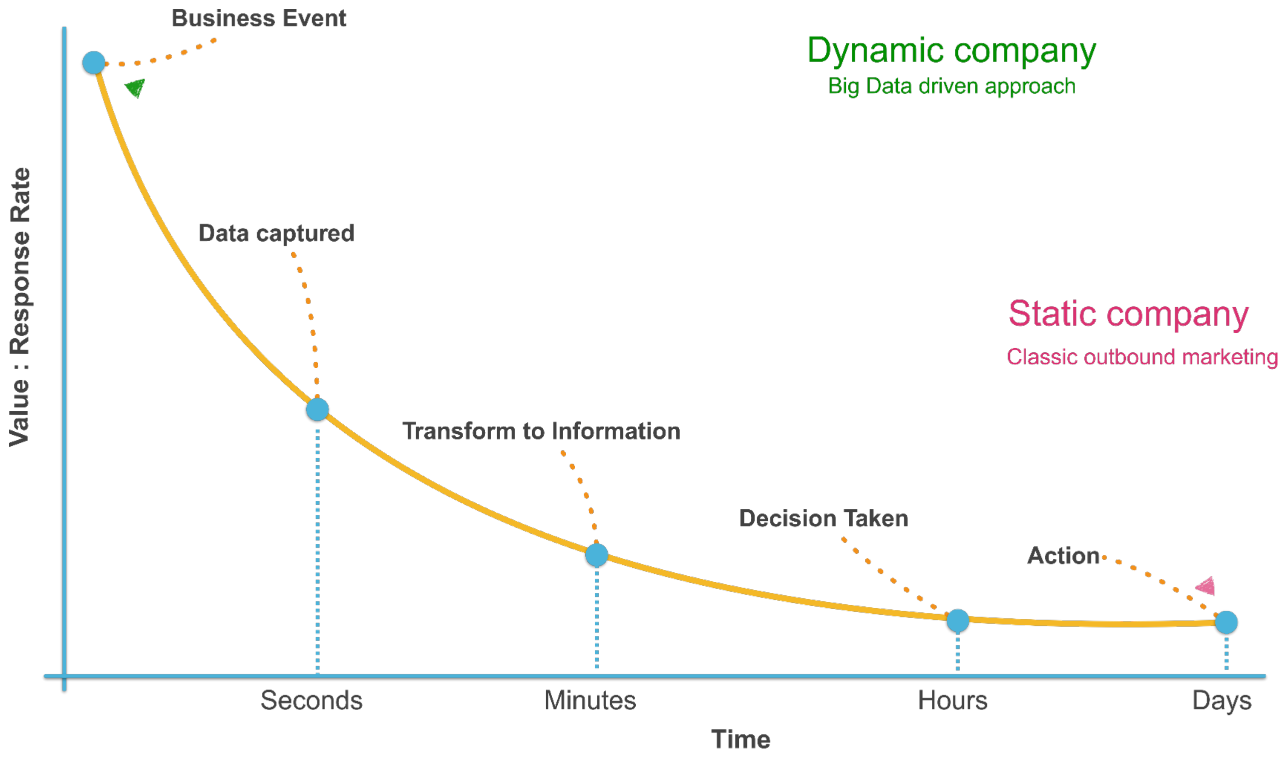
Agenda

- Derived data
- Stream processing
- Apache Flink Architecture
- Benchmark of Stream Processing engines.

Derived Data

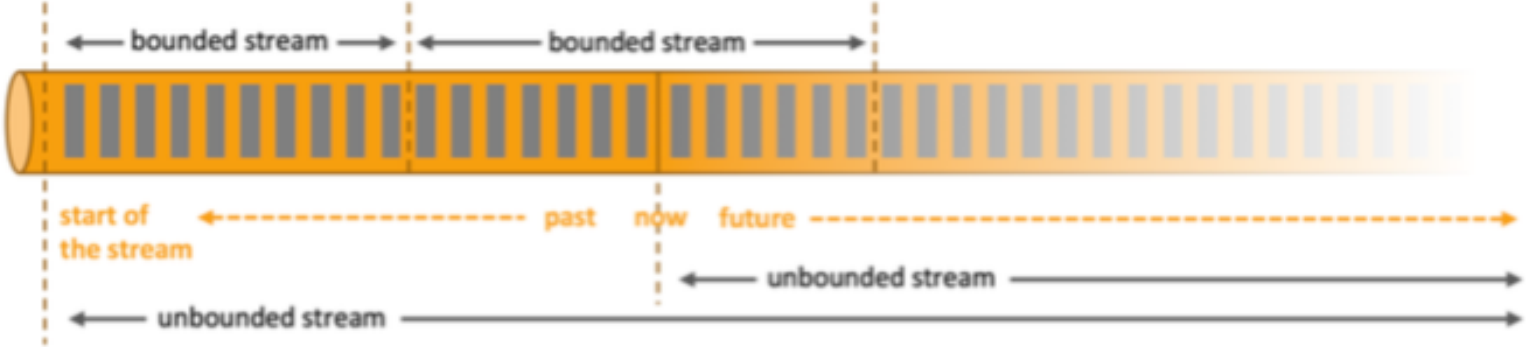
- Network Monitoring and Traffic engineering
- Sensor logs
- Telecom call-detail records
- Mobile Application logs
- Social network trends
- Website logs and clickstreams
- Other massive data sets…

Importance of real-time processing



What is Stream Processing

- Processing data in motion
- Querying continuous data stream
- Detect conditions fast within a small time period
- Related synonyms for Stream Processing are: Real-time analytics, Streaming analytics, Complex Event Processing, Event Processing

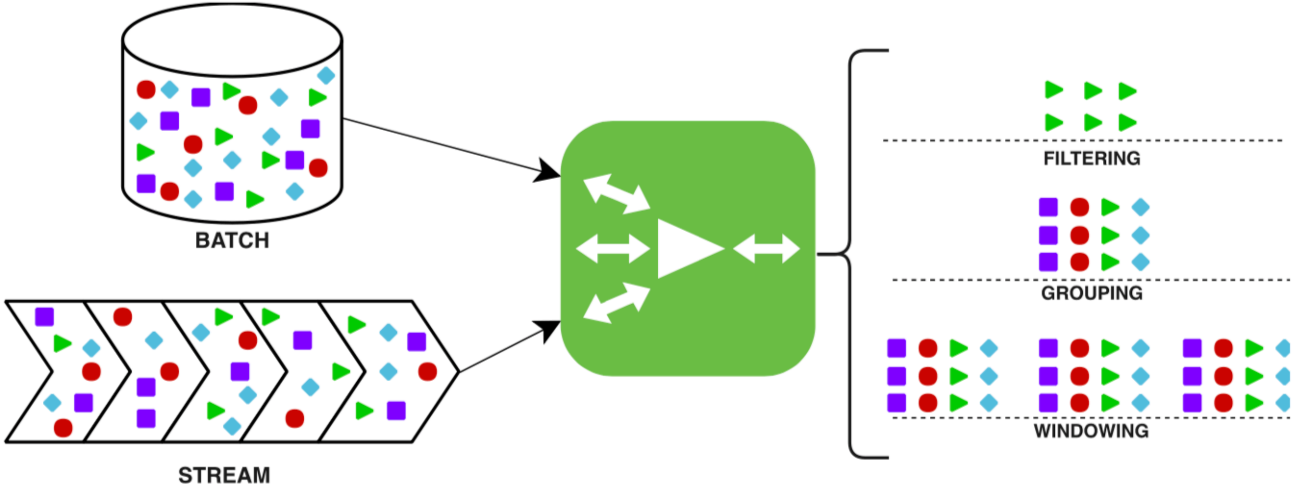


What is Stream Processing doing?

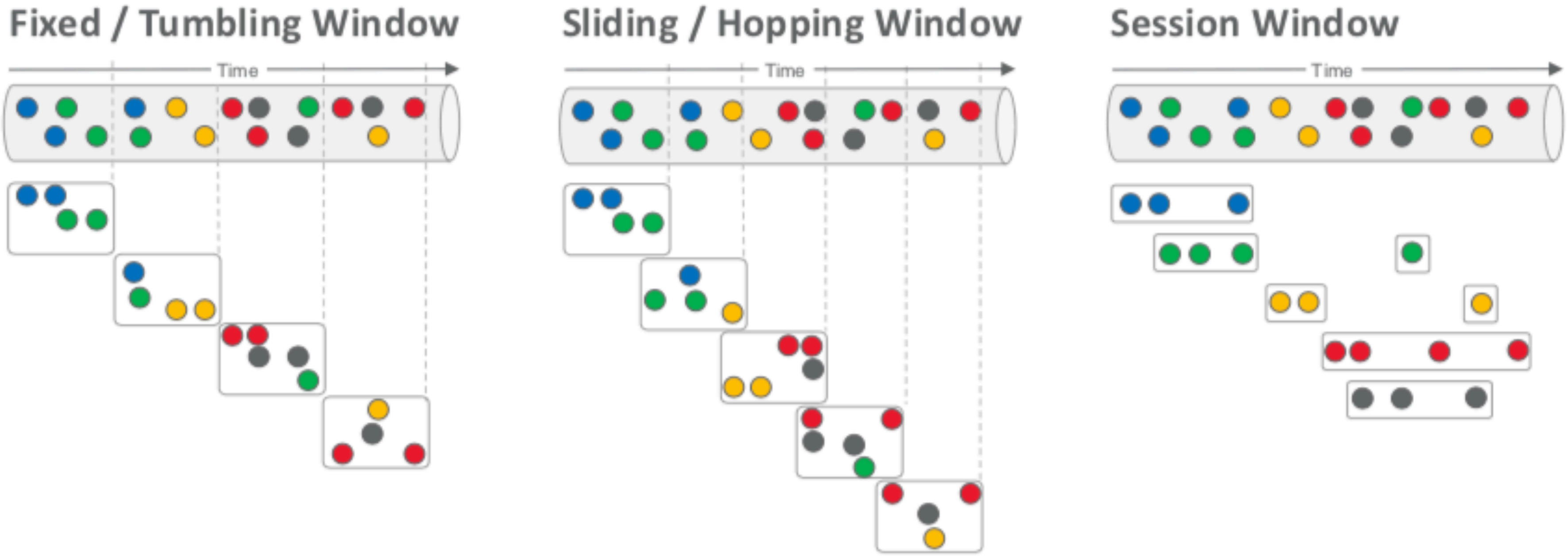
- Sourcing
- Filtering
- Transformation

- Map-Reduce
- Grouping
- Windowing

- Stream-Stream Join
- Stream-Table Join
- Sink

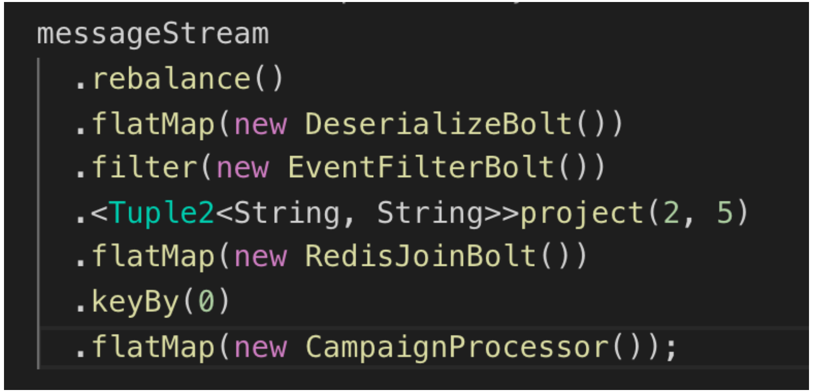


Windowing


Fixed / Tumbling Window
Time
Sliding / Hopping Window
Time
Session Window
Time

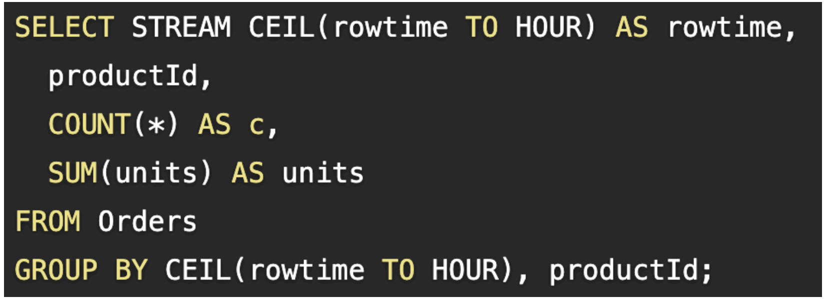
API

- Programmatic



```
messageStream
  .rebalance()
  .flatMap(new DeserializeBolt())
  .filter(new EventFilterBolt())
  .<Tuple2<String, String>>project(2, 5)
  .flatMap(new RedisJoinBolt())
  .keyBy(0)
  .flatMap(new CampaignProcessor());
```

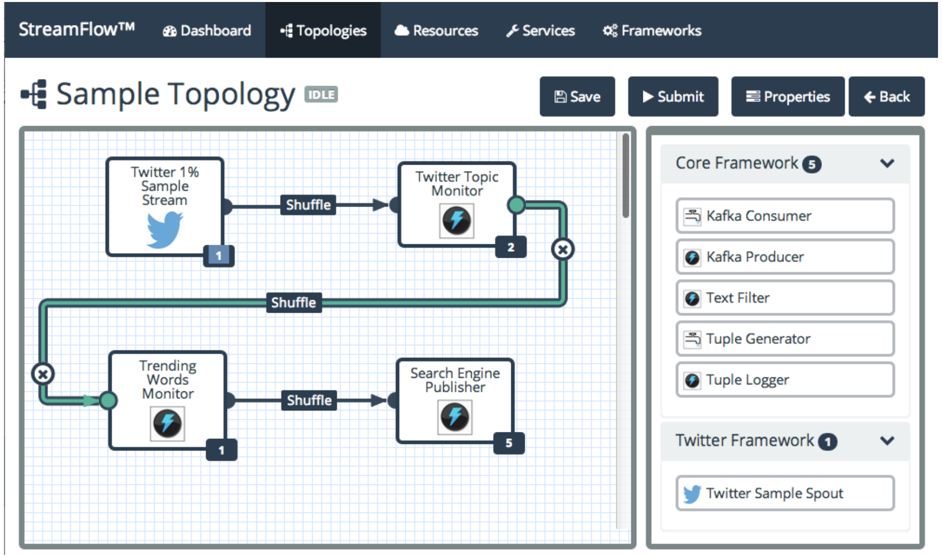
- Streaming SQL

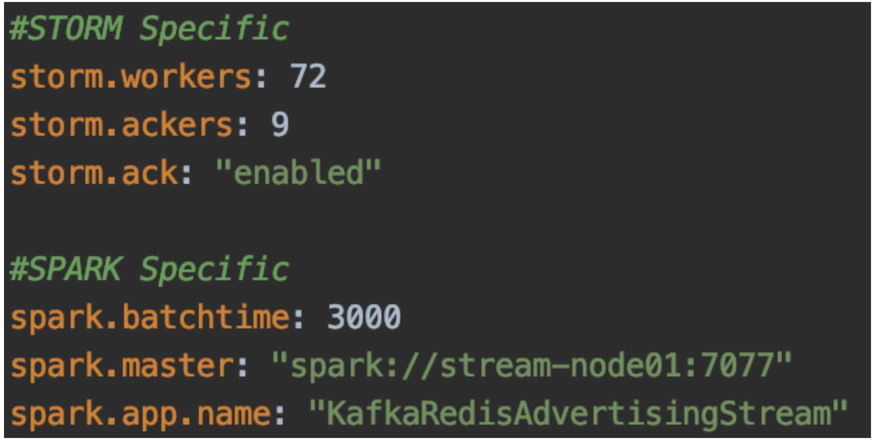


```
SELECT STREAM CEIL(rowtime TO HOUR) AS rowtime,
  productId,
  COUNT(*) AS c,
  SUM(units) AS units
FROM Orders
GROUP BY CEIL(rowtime TO HOUR), productId;
```

API

- GUI-based / drag and drop
- Declarative (Json, Yaml)





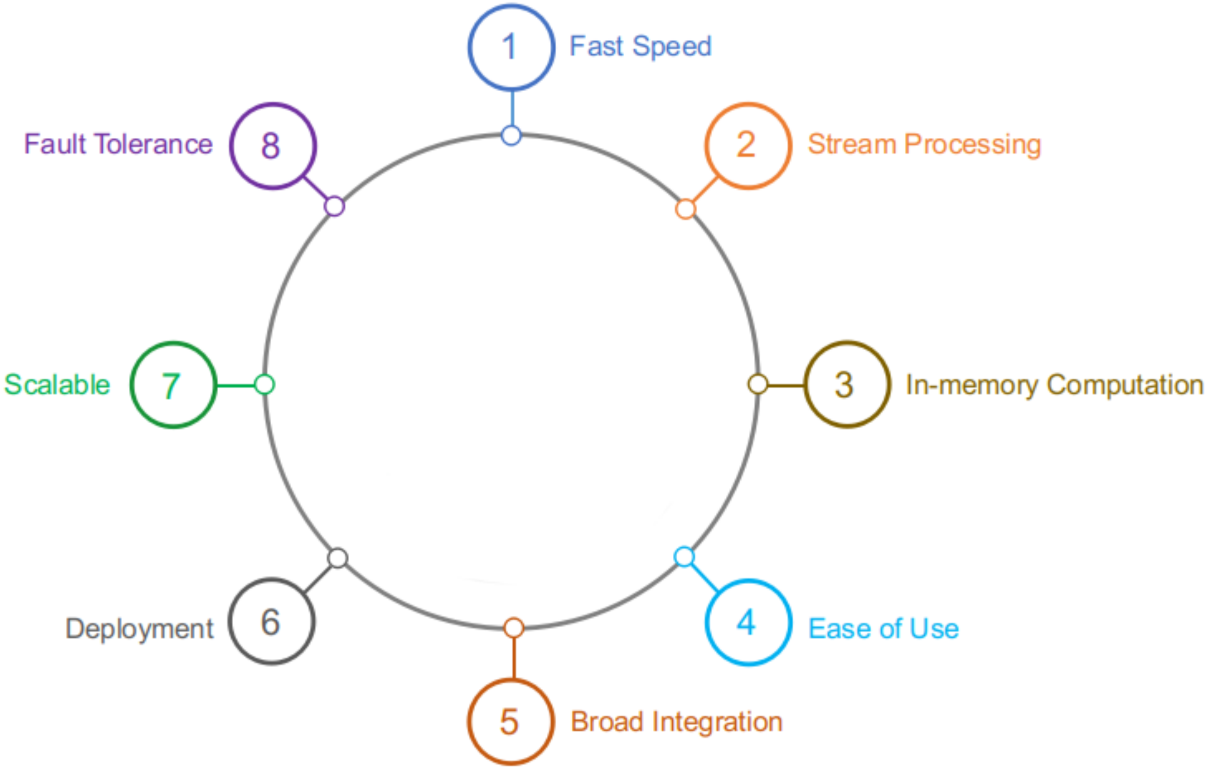
```
#STORM Specific
storm.workers: 72
storm.ackers: 9
storm.ack: "enabled"

#SPARK Specific
spark.batchtime: 3000
spark.master: "spark://stream-node01:7077"
spark.app.name: "KafkaRedisAdvertisingStream"
```

Stream Processing Engines

Airflow
kafka
samza
APEX
beam
amazon KINESIS
Flink
HERON
EsperTech
hazelcast JET
Apache STORM
APACHE Spark™

Features of Stream Processing Engines



1 Fast Speed

2 Stream Processing

3 In-memory Computation

4 Ease of Use

5 Broad Integration

6 Deployment

7 Scalable

8 Fault Tolerance

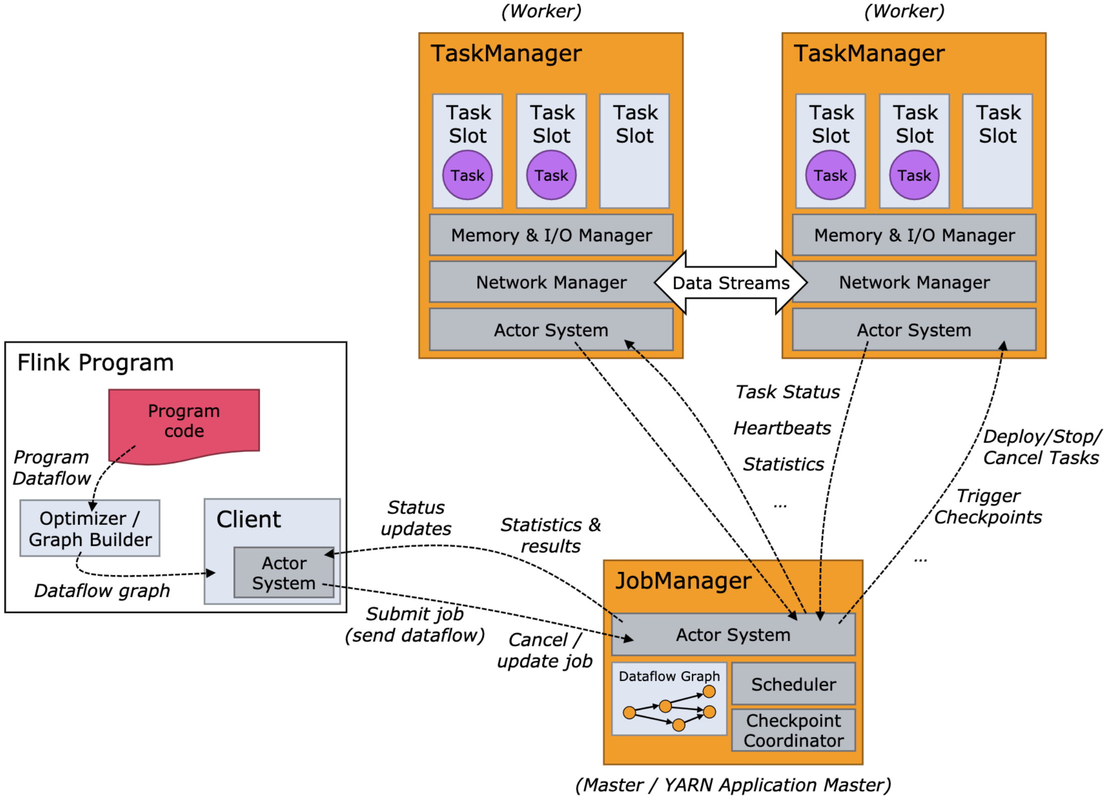
Delivery Guarantees

- At most once - fire and forget
- At least once
- Exactly once

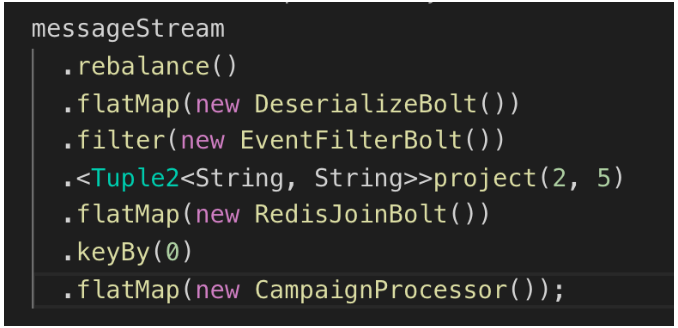
Apache Flink Architecture

- Program
  - It is a piece of code, which you run on the **Flink Cluster**.
- Client
  - It is responsible for taking code (program) and constructing job dataflow graph, then passing it to **JobManager**.
- Job Manager
  - Also called **masters**.
- Task Manager
  - Also called **workers** or **slaves**.

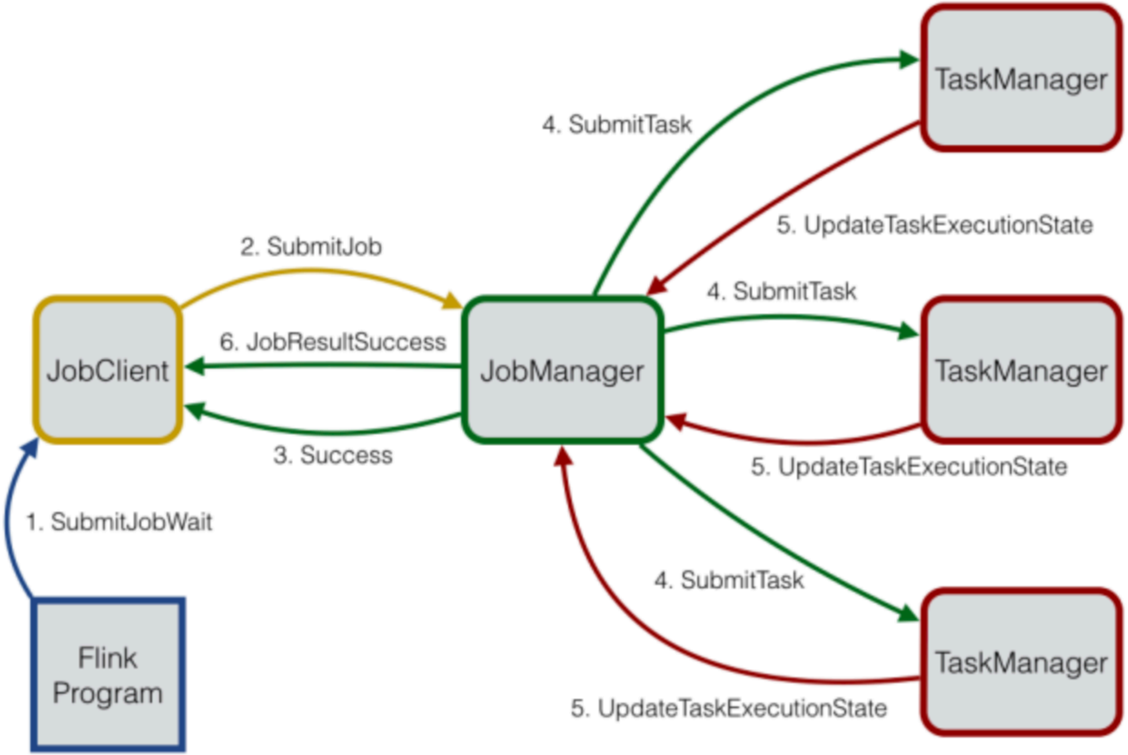
Apache Flink Architecture



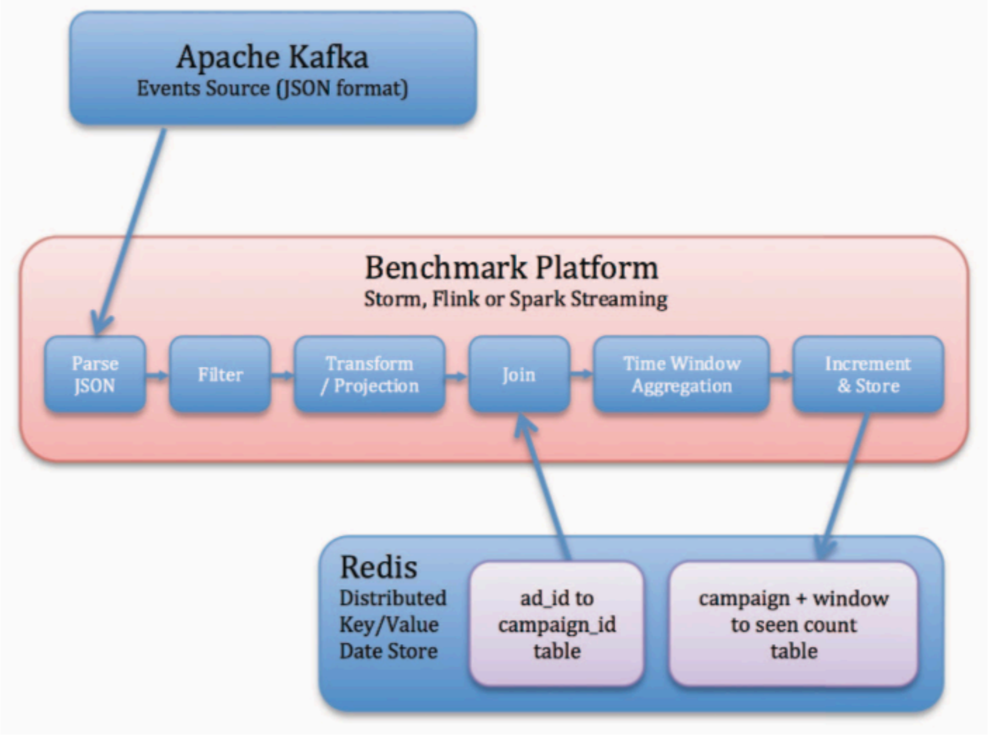
Apache Flink Architecture



```
messageStream
  .rebalance()
  .flatMap(new DeserializeBolt())
  .filter(new EventFilterBolt())
  .<Tuple2<String, String>>project(2, 5)
  .flatMap(new RedisJoinBolt())
  .keyBy(0)
  .flatMap(new CampaignProcessor());
```



Experiment Design


Apache Kafka
Events Source (JSON format)
Benchmark Platform
Storm, Flink or Spark Streaming
Parse JSON
Filter
Transform / Projection
Join
Time Window Aggregation
Increment & Store
Redis
Distributed Key/Value Date Store
ad_id to campaign_id table
campaign + window to seen count table

1. C. Sanket, D. Derek, E. Bobby, F. Reza, G. Thomas, H. Mark, L. Zhuo, N. Kyle, P. Kishorkumar, J. P. Boyang ja P. Paul, „Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming.,“ IEEE, 2016.

Comparison with Yahoo Streaming Benchmark

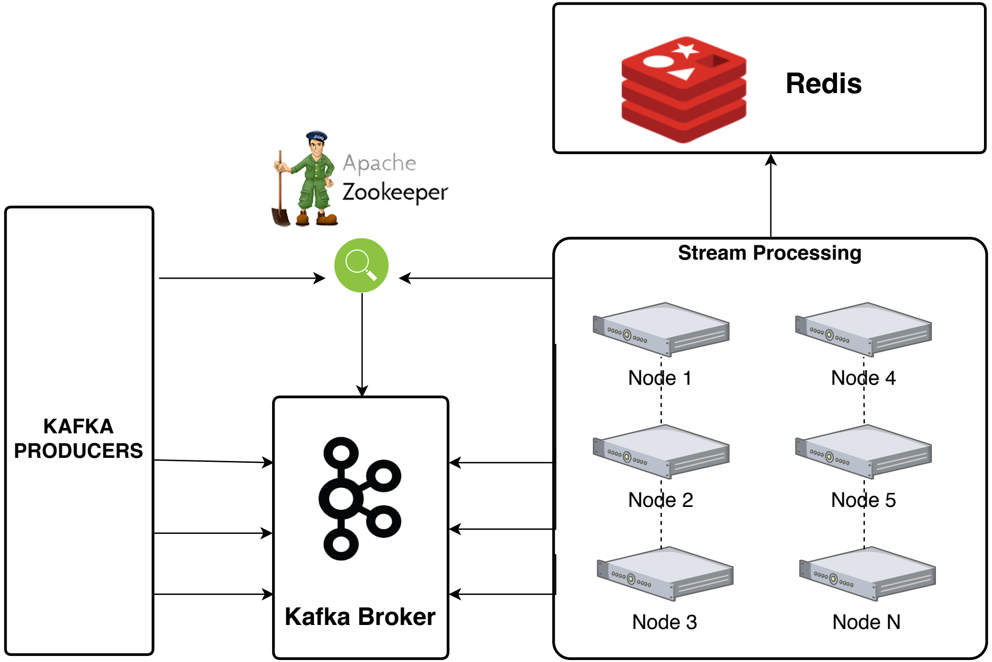
Tools	Yahoo’s version (2015)	My version (2018)
Flink	1.1.3	1.5.0
Spark DStream	1.6.2	2.3.0
Strom	0.9.7	1.2.1
Redis	3.0.5	4.0.8
Kafka Broker	0.8.2.1	0.11.0.2
Spark Structured Streaming	-	2.3.0
Kafka Stream	-	1.1.0

Benchmark	Benchmark Metrics	Servers Type	Servers CPU	Servers memory
Yahoo’s version	Latency, Throughput	Dedicated server	16 cores (8 physical, 16 hyperthreading)	24GB memory
My Version	Latency, Throughput, Resource Consumption	Virtual Private Server	16 virtual cores	32GB memory

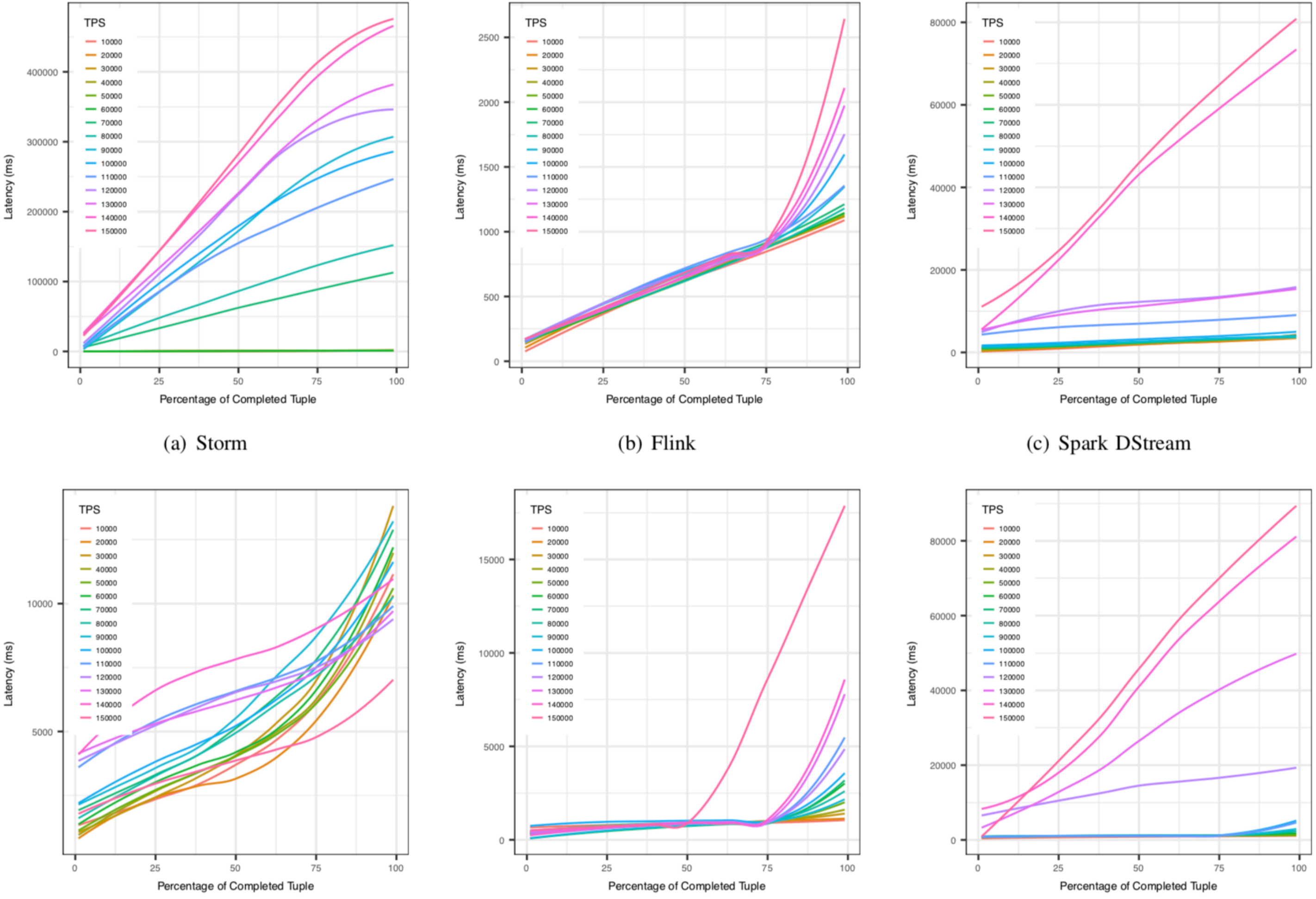
GitHub Repository of the project https://github.com/elkhan-shahverdi/streaming-benchmarks

Experiment Topology

- 10 Stream Processing nodes
- 5 Kafka nodes
- 3 Zookeeper nodes
- 10 Producer nodes
- 1 Redis node



Experiment Results



(a) Storm

(b) Flink

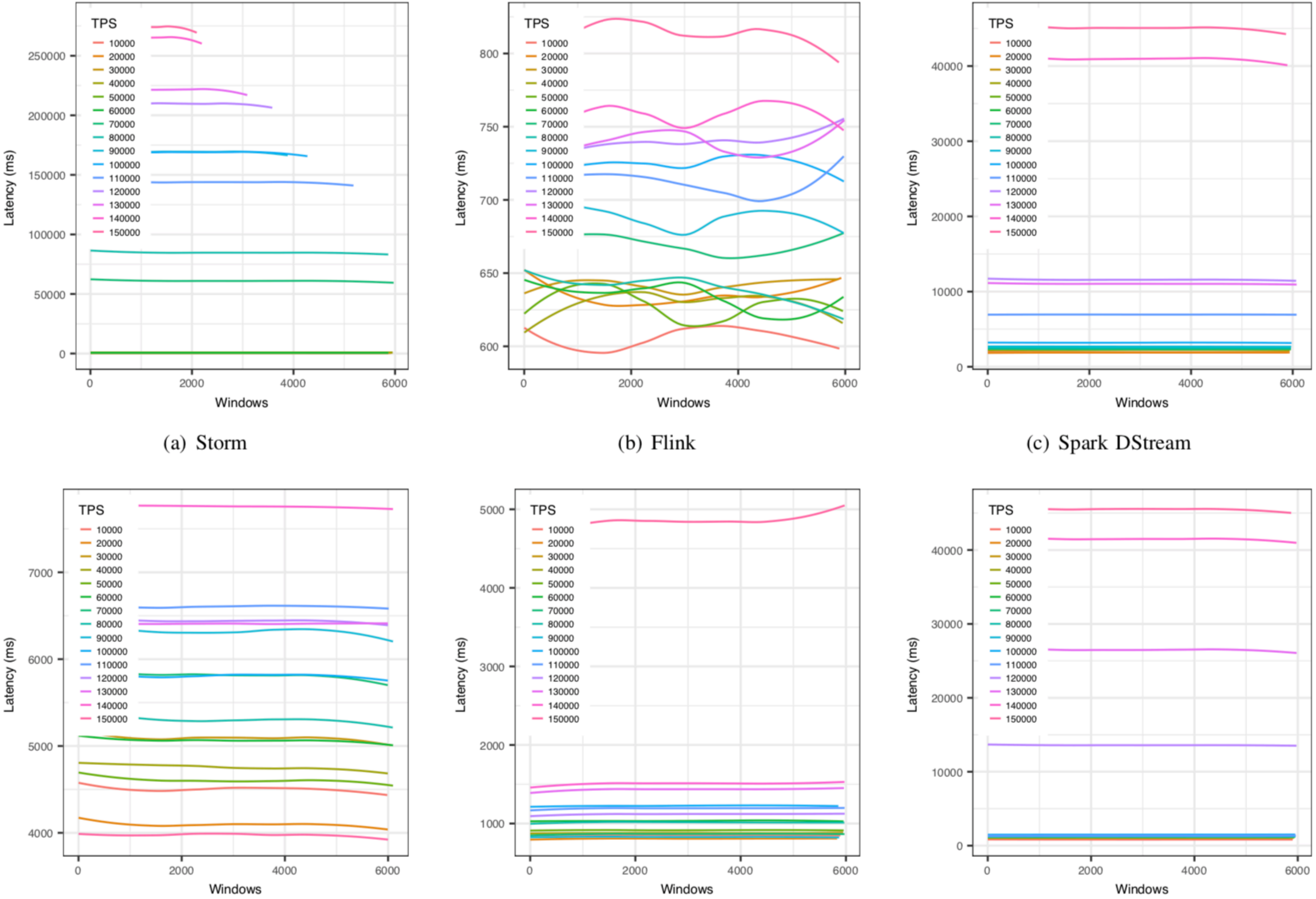
(c) Spark DStream

(d) Spark Structured Streaming

(e) Kafka Streams

(f) Hazelcast Jet

Fig. 2. Latency versus percentage of completed tuples



(a) Storm

(b) Flink

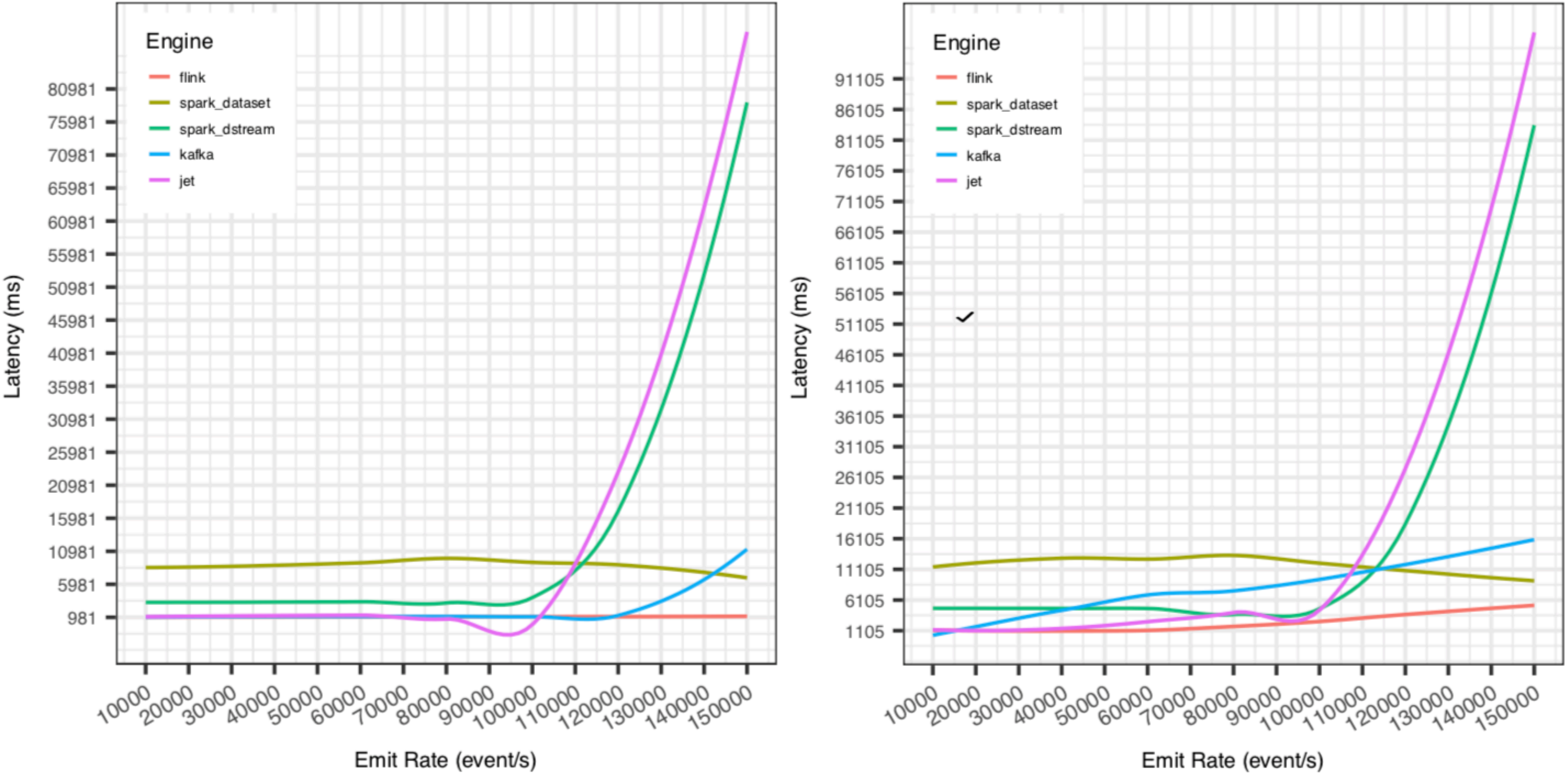
(c) Spark DStream

(d) Spark Structured Streaming

(e) Kafka Streams

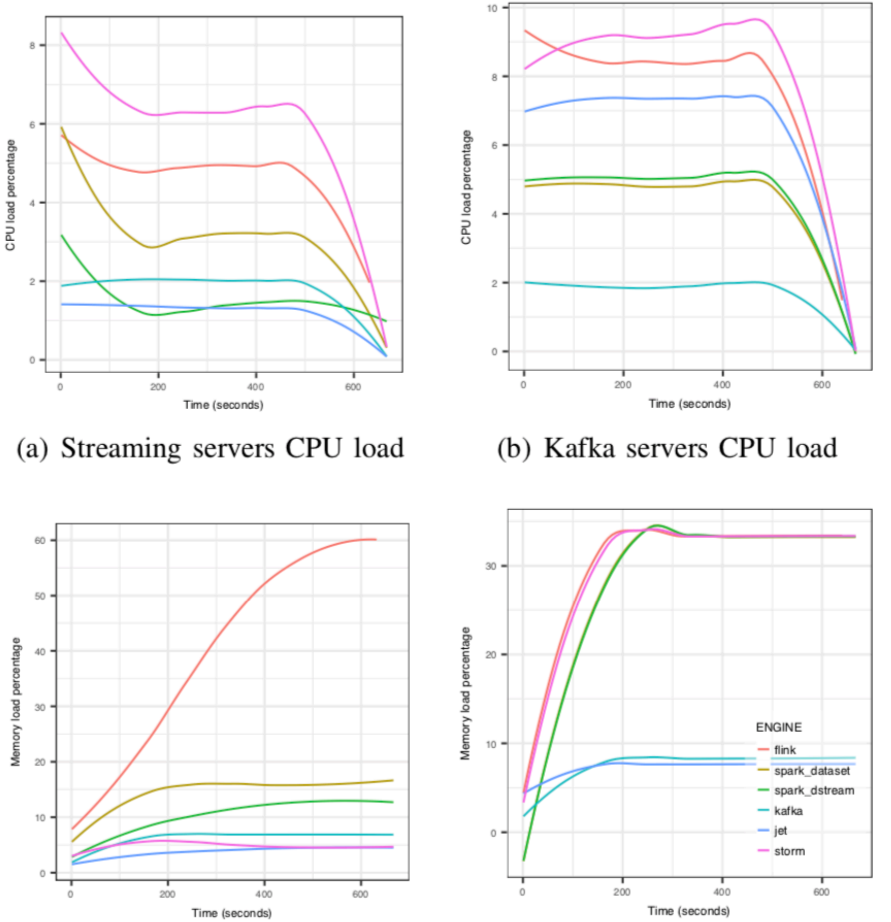
(f) Hazelcast Jet

Fig. 3. Loess Regression of Latencies



(a) 90% Percentile Latency

(b) 99% Percentile Latency



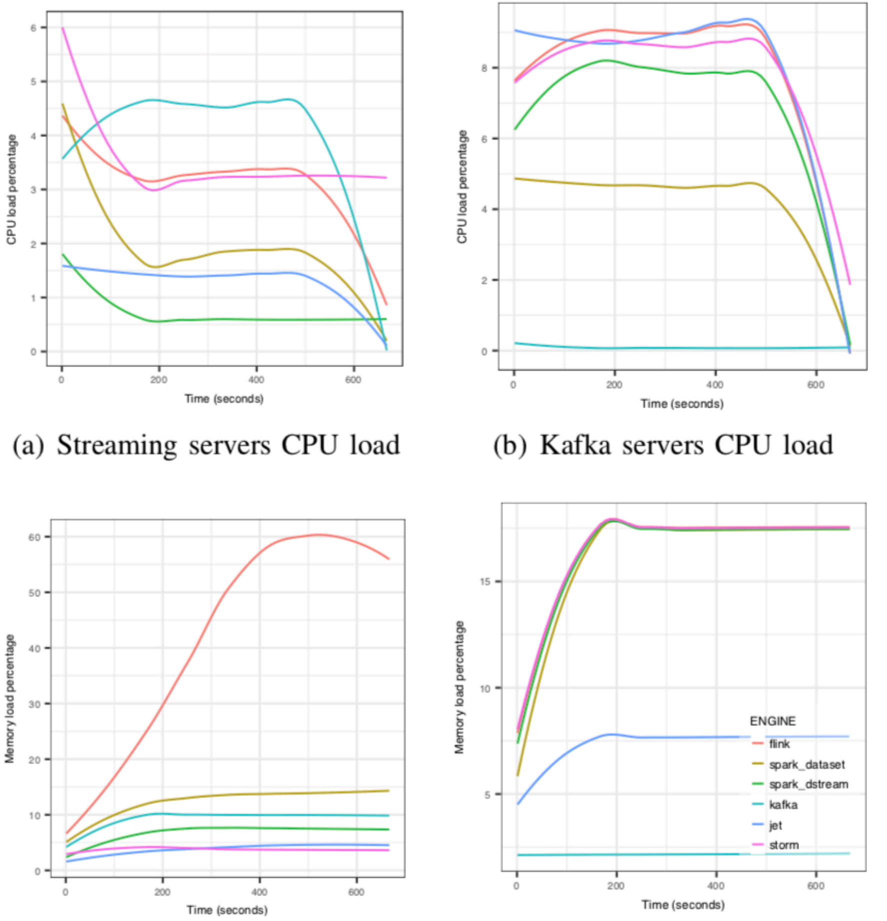
(a) Streaming servers CPU load

(b) Kafka servers CPU load

(c) Streaming servers memory consumption

(d) Kafka servers memory consumption

Fig. 5. Resource Consumption at $60K$ TPS



(a) Streaming servers CPU load

(b) Kafka servers CPU load

(c) Streaming servers memory consumption

(d) Kafka servers memory consumption

Fig. 6. Resource Consumption at $150K$ TPS

Experiment Conclusion

- Flink and Kafka are most noticeable real-time stream processors.
- Flink has lower latency than Kafka.
- Apache Strom latency can race with Flink with the low amount of data.
- Flink custom memory management is not better than Java GC.
- Spark Structured Streaming is the more resistant than DStream under huge amount of data.
- Flink owes the highest performance for more resource consumptions

Conclusion

- Principle is important than tools

References

---

1. Big Stream Processing Systems: An Experimental Evaluation by Elkhan Shahverdi, Ahmed Awad, Sherif Sakr
2. Designing Data-Intensive Applications. By Kleppmann, Martin
3. Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing. by Tyler Akidau, Slava Chernyak, Reuven Lax
4. https://cwiki.apache.org/confluence/display/FLINK/Flink+Internals
5. https://github.com/elkhan-shahverdi/streaming-benchmarks