STREAMLINE. FAST REACTIVE ANALYTICS

RI. SE

Approximate Standing Queries on Apache Flink


@tobiaslindener


lindener@kth.se

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 688191.

Overview

1. Introduction
2. Background
3. Design & Implementation
4. Results

It is better to use a crude approximation and know the truth, plus or minus 10 percent, than demand an exact solution and know nothing at all

In Arthur Bloch, *The Complete Murphy's Law: A Definitive Collection* (1991), 126

Unbounded Stream

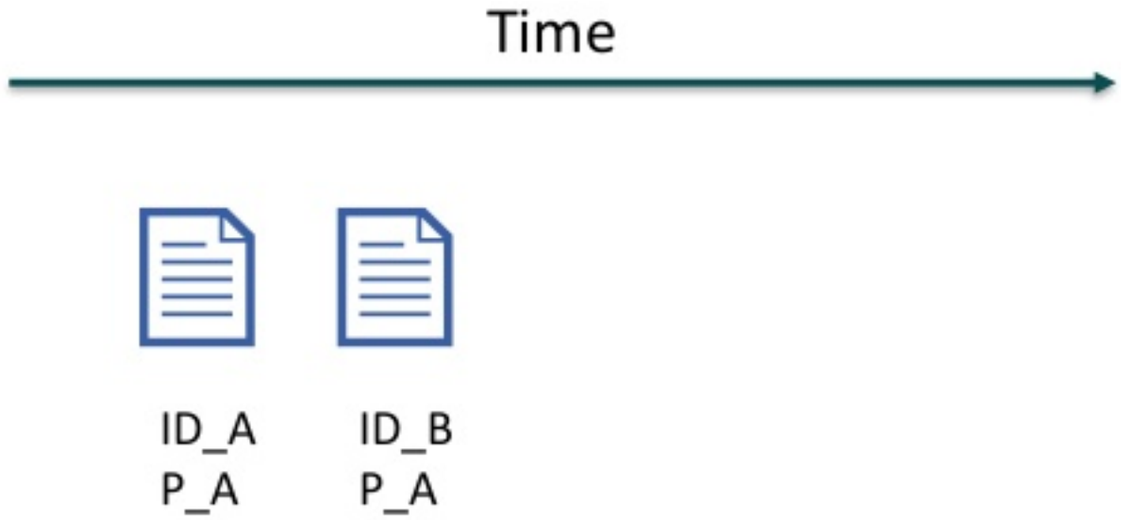
Time

ID_A
P_A

Key	Count
A	1

Infinite Growth

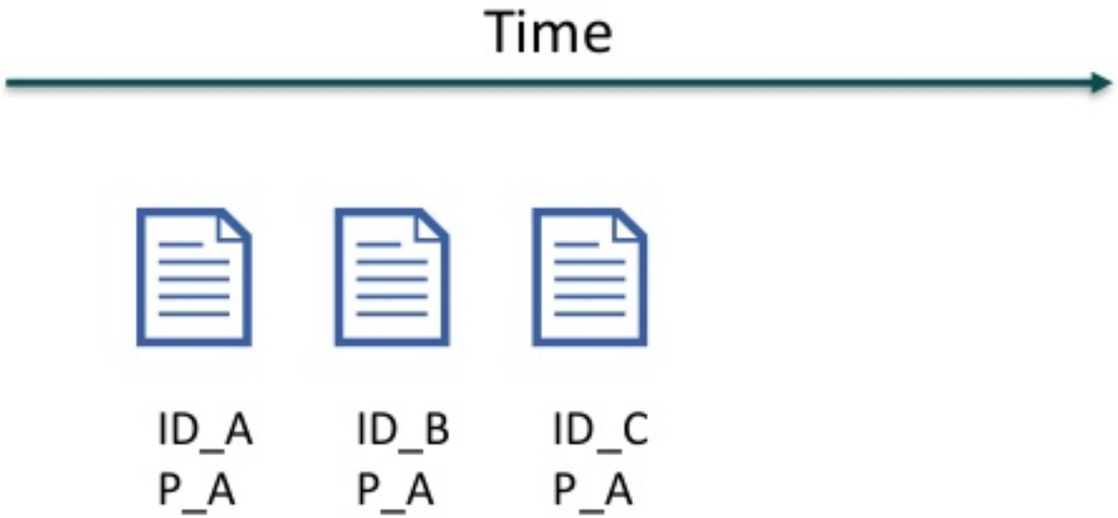
Unbounded Stream



Key	Count
A	1
B	1

Infinite Growth

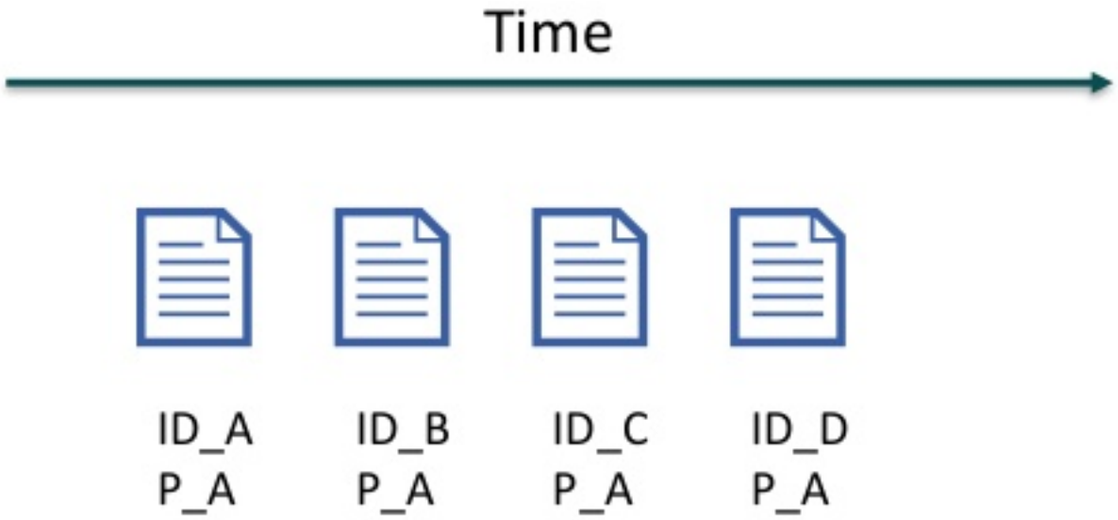
Unbounded Stream



Key	Count
A	1
B	1
C	1

Infinite Growth

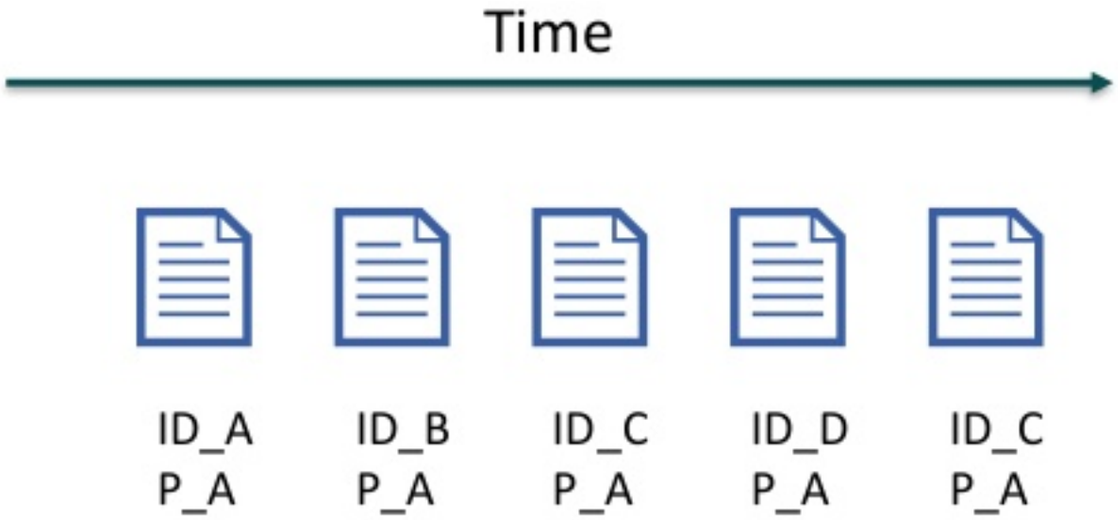
Unbounded Stream



Key	Count
A	1
B	1
C	1
D	1

Infinite Growth

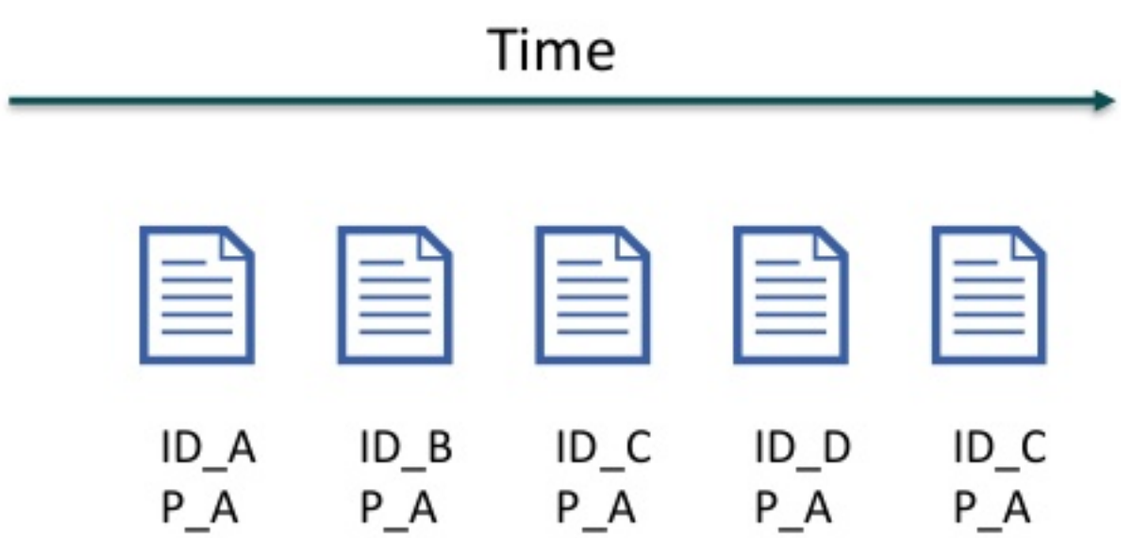
Unbounded Stream



Key	Count
A	1
B	1
C	2
D	1

Infinite Growth

Unbounded Stream



Key	Count
A	1
B	1
C	2
D	1

Infinite Growth

Approximation Algorithms

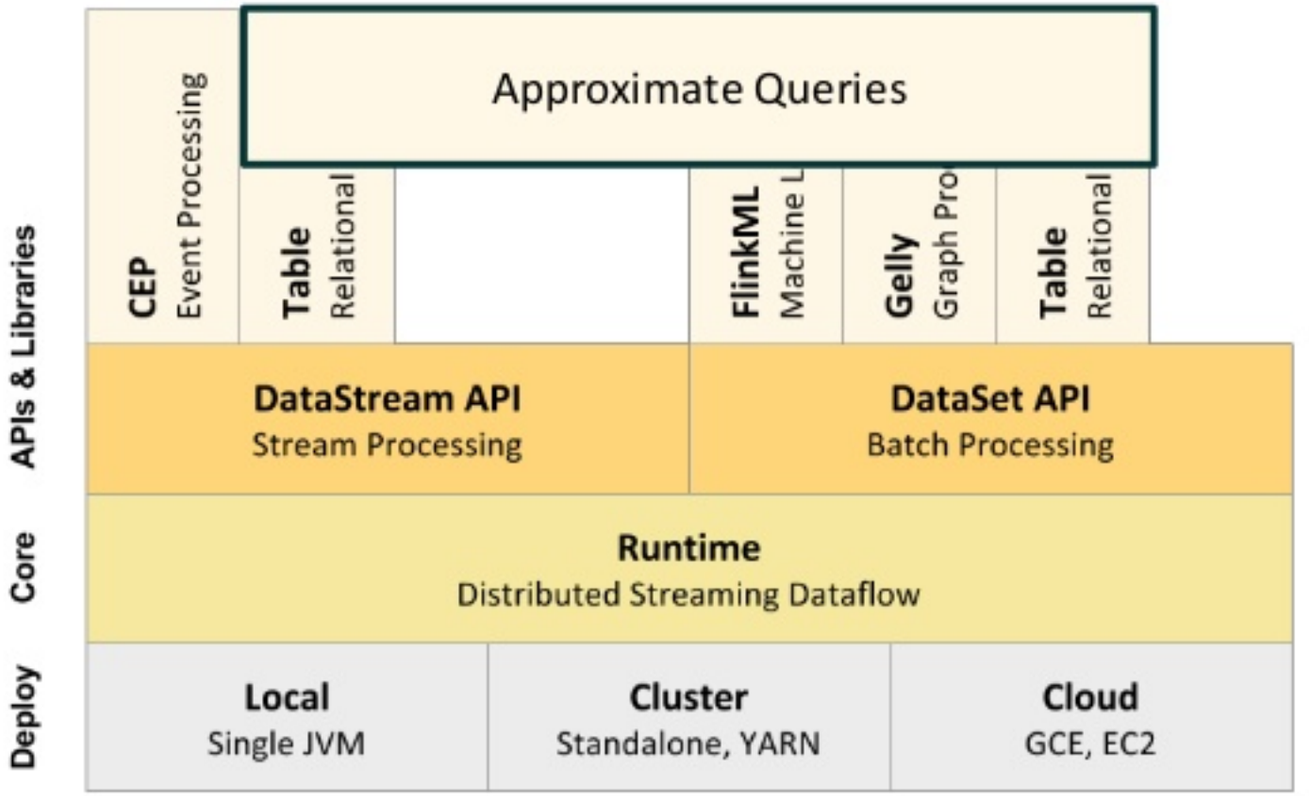
Use-cases

- Identify heavy hitters (Count)
- Cardinality Estimation (Count Distinct)

Algorithms

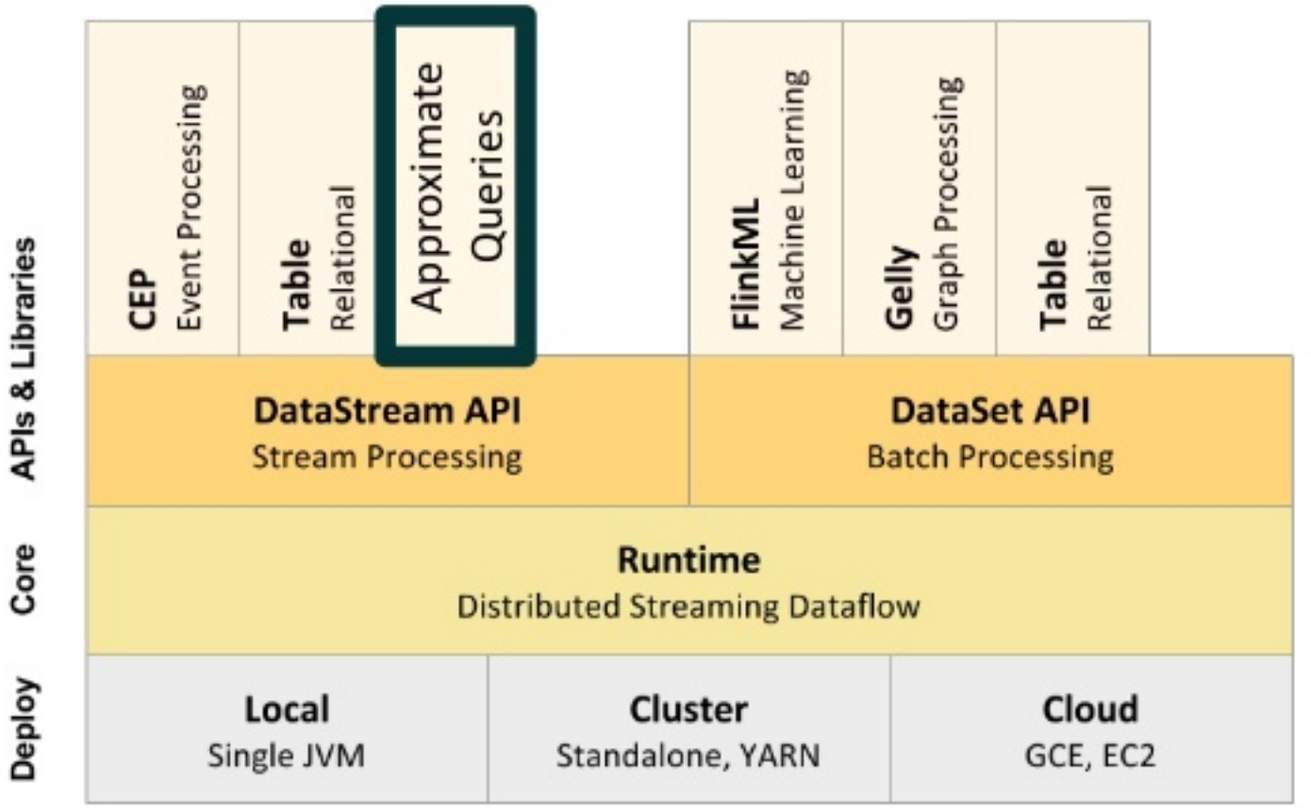
- Frequent Item Estimation
- HyperLogLog
- Quantile Estimation

Processing



Flink Architecture (Apache Software Foundation, 2018)

Processing

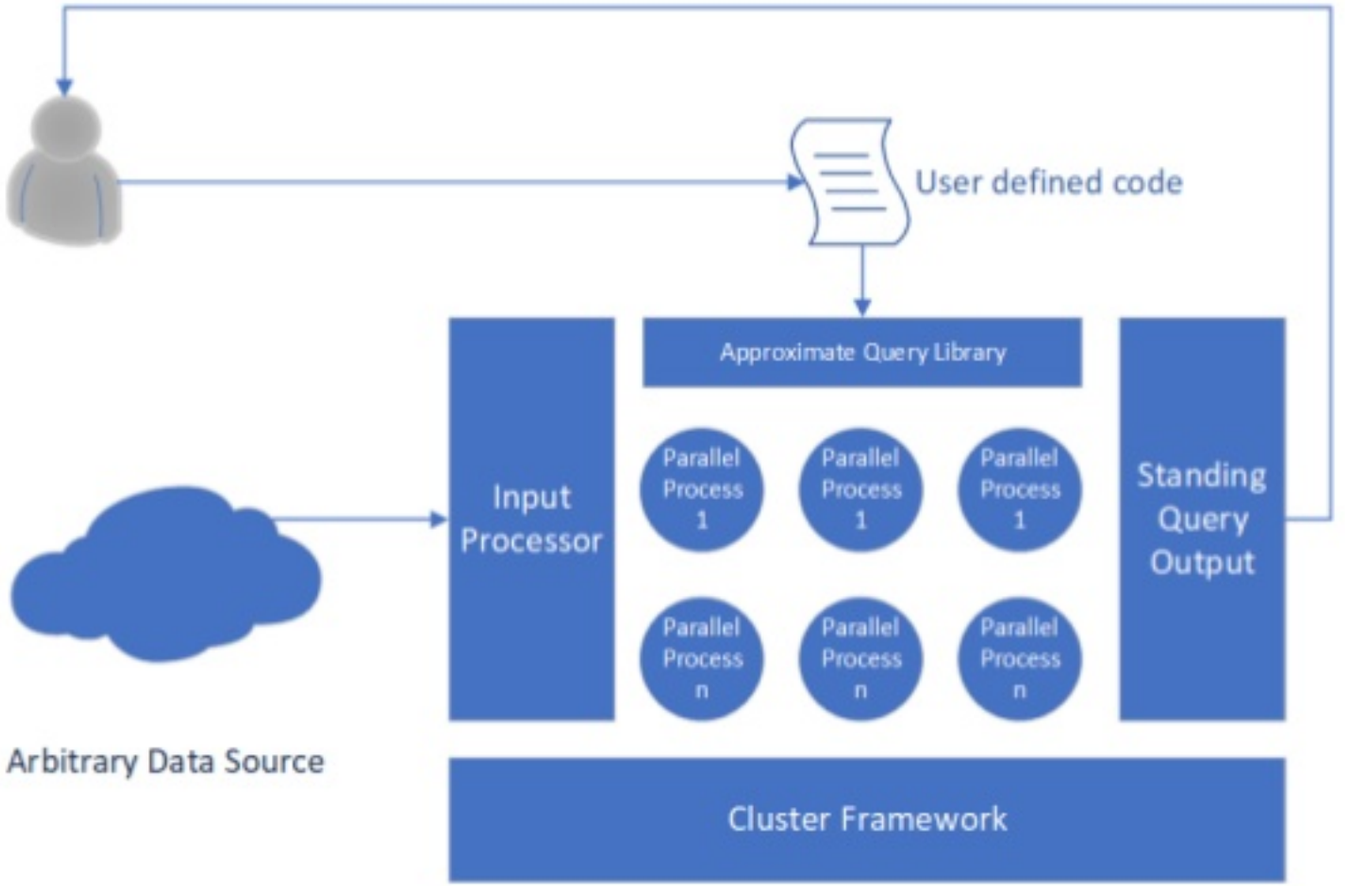


Flink Architecture (Apache Software Foundation, 2018)

Approximate Query API (1)

```
1 public static <T> DataStream<TopNQueryResult>
    ↪ continuousFrequentItems(DataStream<T> inputStream,
    ↪ KeySelector valueSelector, int maxItems, int
    ↪ emitMin)
2
3 public static <T> DataStream<QuantileQueryResult>
    ↪ continuousQuantiles(DataStream<T> inputStream,
    ↪ KeySelector valueSelector)
```

Approximate Query API (2)



Sketch Capacity

Input of the dataset from file or external system

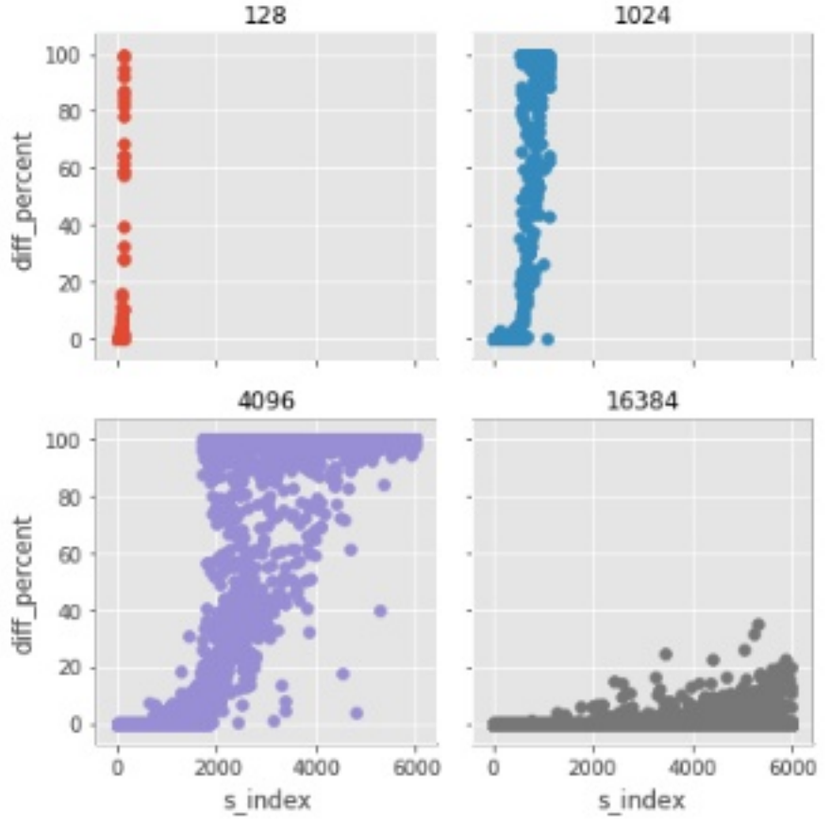
Distribute data over multiple worker using consistent hashing

Update sketch with incoming stream elements

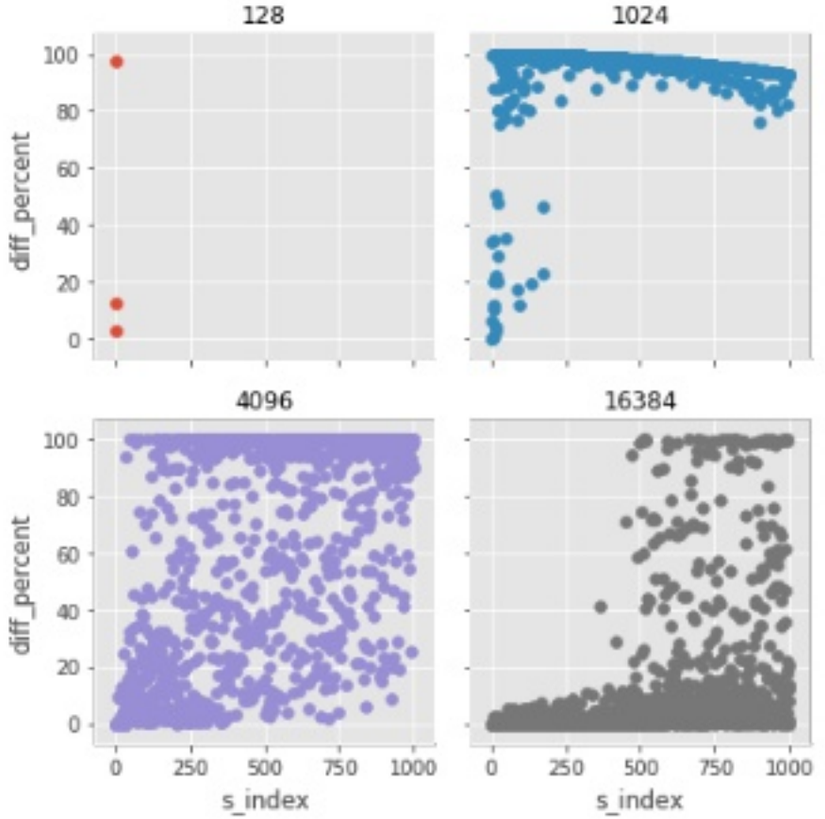
Persist the result of the updated sketch on disk

Estimate Deviation - Frequent Items

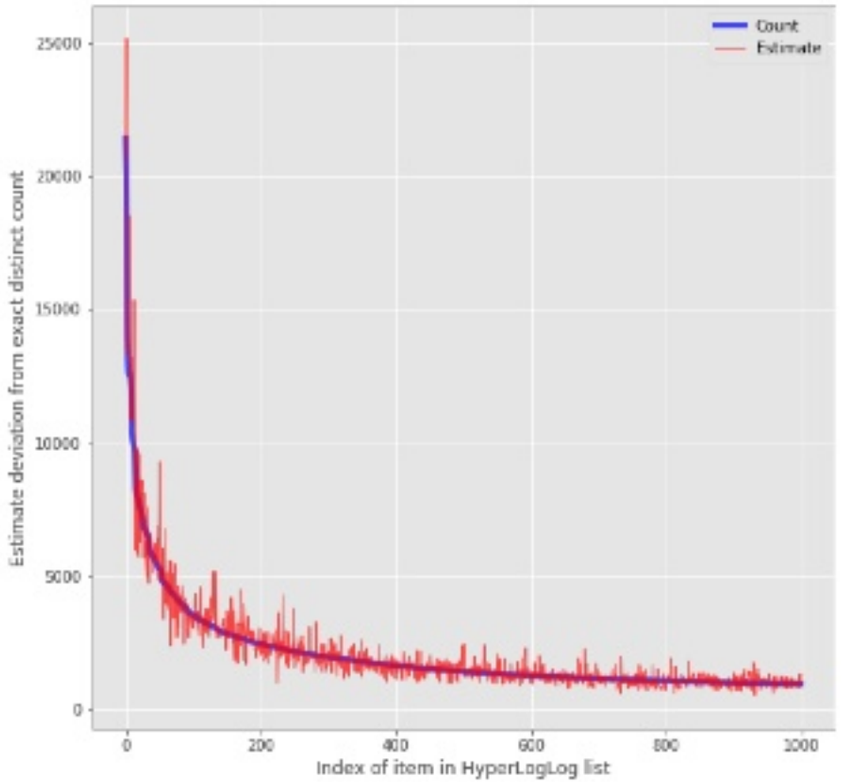
WIKITRACE DATASET TOP 6000 URL



AMAZON DATASET TOP 1000 REVIEWER

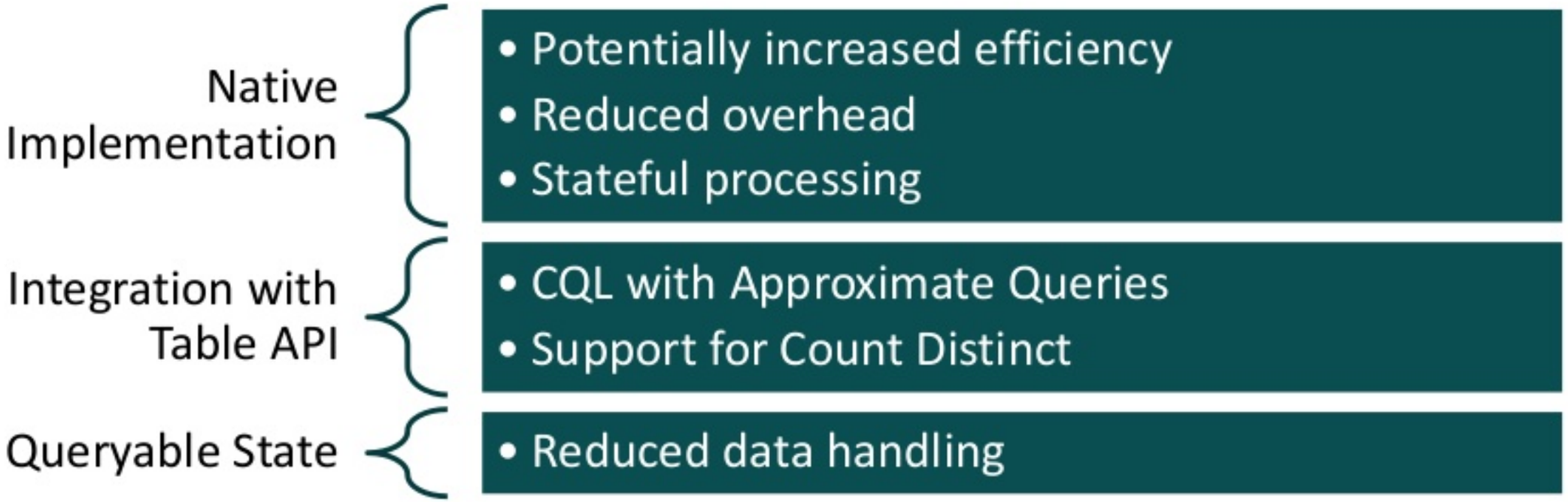


Estimate Deviation - HyperLogLog

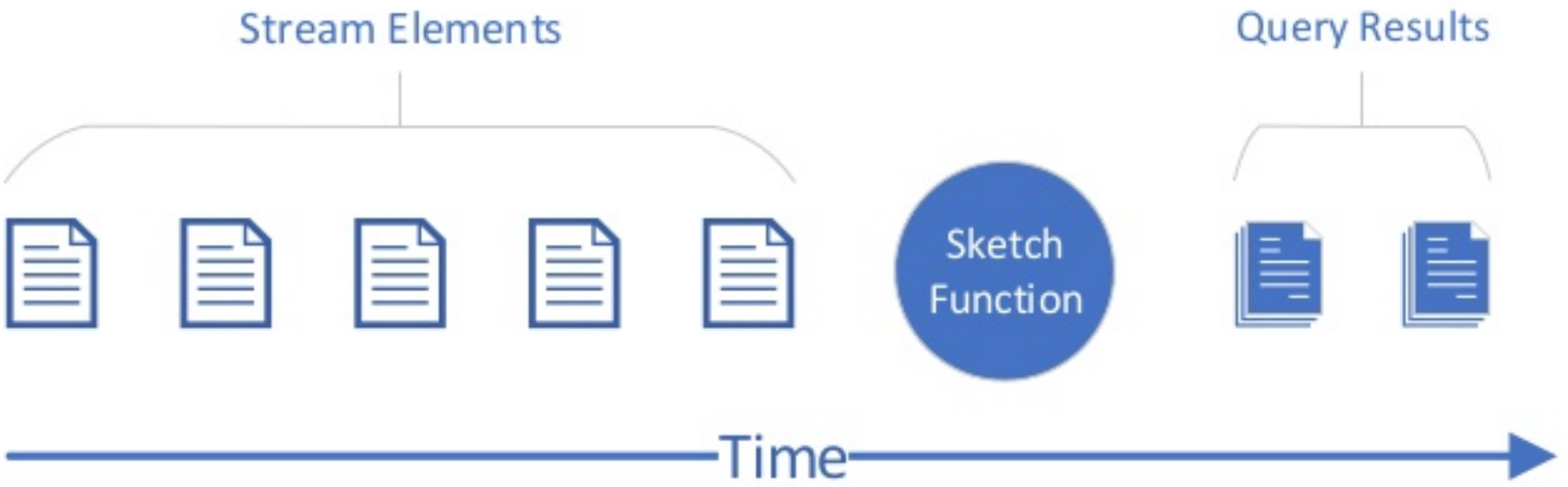



```
SELECT COUNT(DISTINCT(Reviewer)) FROM ratings GROUP BY
↪ Rating
```

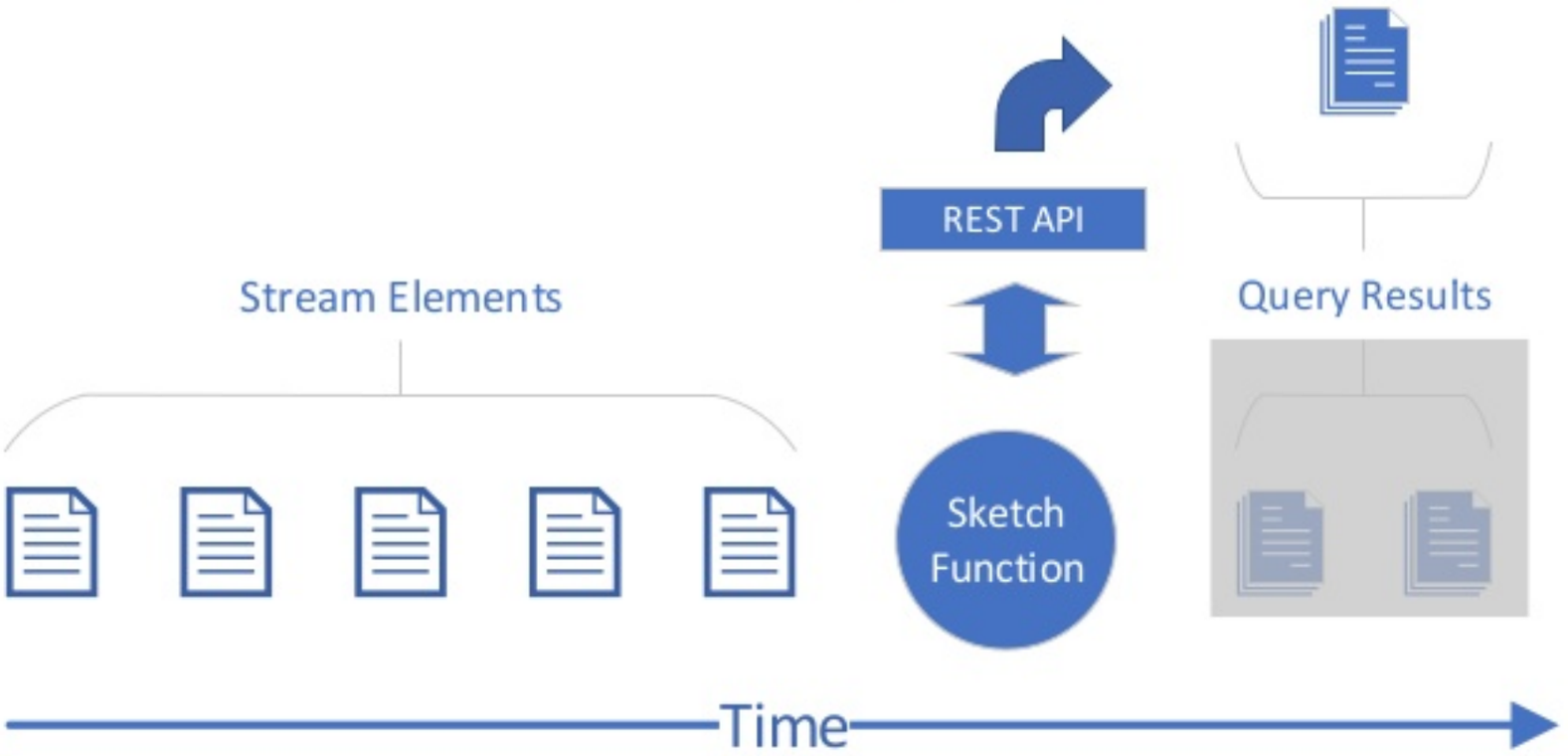
Future Work


Native Implementation
• Potentially increased efficiency
• Reduced overhead
• Stateful processing
Integration with Table API
• CQL with Approximate Queries
• Support for Count Distinct
Queryable State
• Reduced data handling

Queryable State



Queryable State



Conclusion

CHALLENGES

- Efficient Grouping (HLL)
- Stateful Native Implementation
- Skewed Datasets

LEARNINGS

- Importance of Data Distribution
- Performance Advantages
- Algorithm Parameters

Team

Theodore Vasiloudis
Researcher @ RISE

Tobias Lindener
Consultant @ Netlight

Paris Carbone
PhD candidate @ KTH

Slides
bit.ly/2LULyoZ

References & Links

- https://github.com/tlindener/ApproximateQueries
- https://datasketches.github.io/
- Daniel Anderson, Pryce Bevan, Kevin Lang, Edo Liberty, Lee Rhodes, Justin Thaler. A High-Performance Algorithm for Identifying Frequent Items in Data Streams.
- Kevin Lang, Back to the Future: an Even More Nearly Optimal Cardinality Estimation Algorithm.

Evaluation Environment

- WikiTrace Dataset (9 GB)
  - Address (6,708,723 distinct urls)
- Amazon Rating Dataset (3 GB)
  - ProductId (9,874,210 distinct items)
  - ReviewerId (21,176,521 distinct users)

- Ryzen 1600 (6C/12T)
- 16GB RAM
- Ubuntu 18.04
- OpenJDK 8
  - JVM tuned for max 10 GB heap
- Flink 1.4.2
  - Standalone mode
  - Evaluation through python scripts