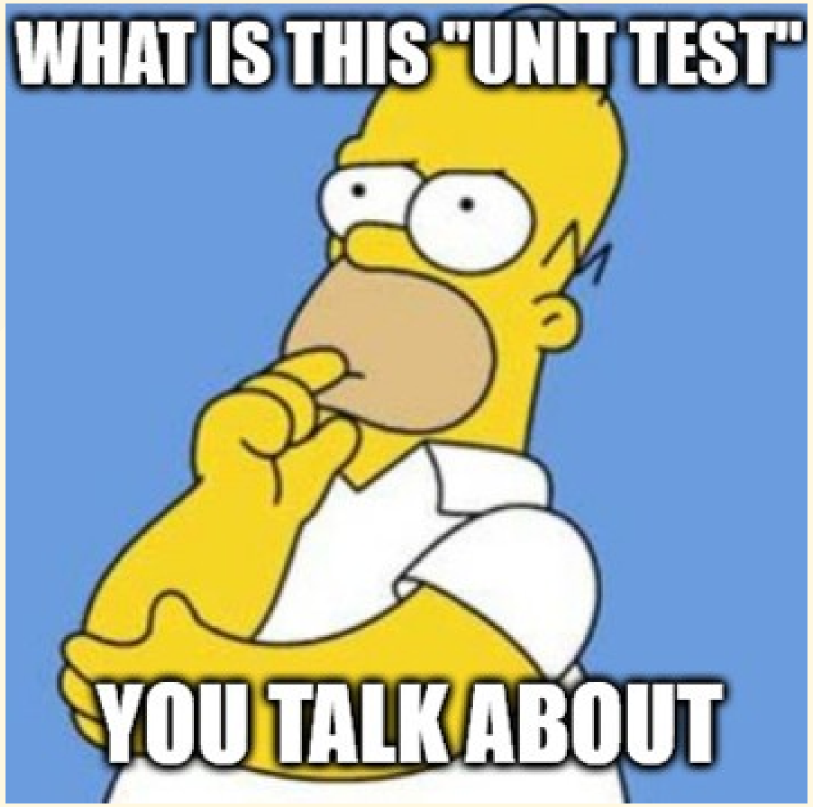
Real-Time Streaming Showdown

Sherif Behna
Data Architect
Hikari Data inc.

About me

WHAT IS THIS "JOIN"
YOU TALK ABOUT


WHAT IS THIS "UNIT TEST"
YOU TALK ABOUT

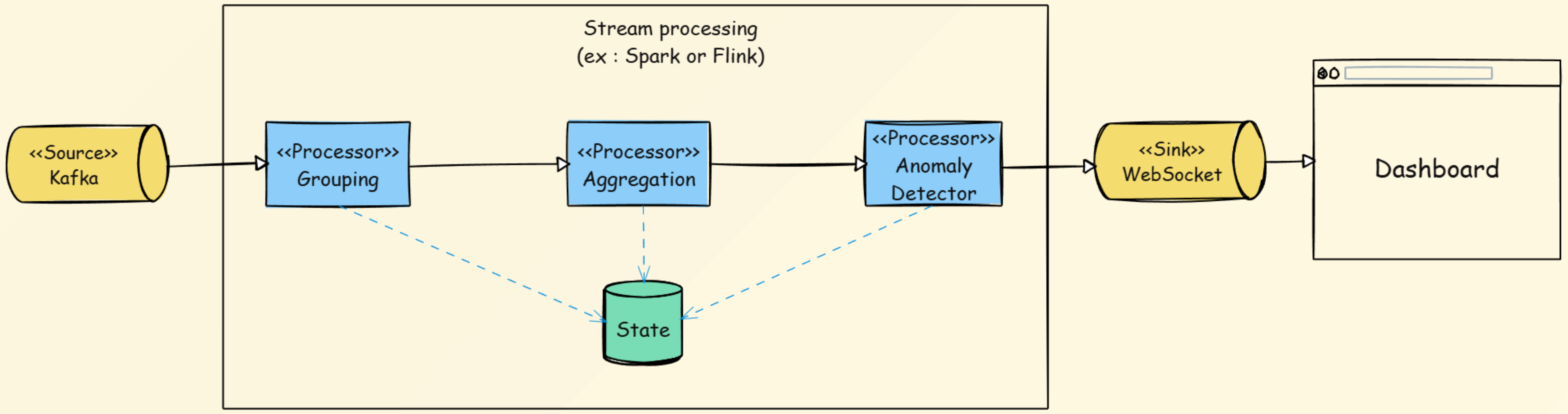
What is Stream Processing ?

- Stream : Continuous sequence of events over time
- Stream processing : Processing these events to get insights

Typical Use Cases

- Real-time analytics
- Event-driven applications
- Sensor data processing (IoT)
- Business monitoring and alerting
- Anomaly and fraud detection

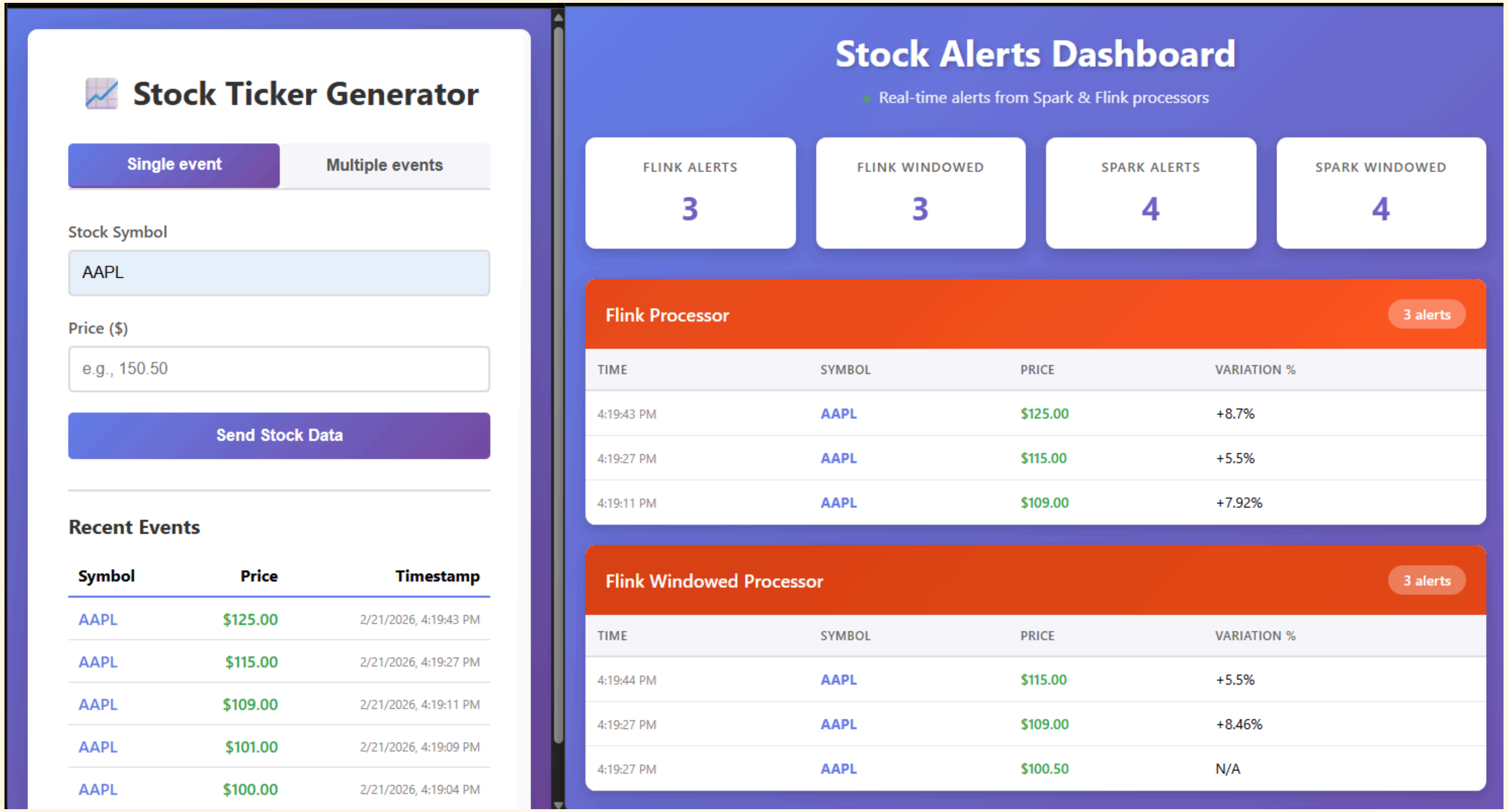
Typical Architecture



Do you really need "real-time"

- Real-time means milliseconds to a few seconds (for example : 5s end-to-end)
- Leads to extra complexity (vs batch or "slow real-time")
- Requires better monitoring and alerting
- Is there a business case for it ?

Demo



Stock Symbol: AAPL

Price ($): e.g., 150.50

Recent Events

Symbol	Price	Timestamp
AAPL	$125.00	2/21/2026, 4:19:43 PM
AAPL	$115.00	2/21/2026, 4:19:27 PM
AAPL	$109.00	2/21/2026, 4:19:11 PM
AAPL	$101.00	2/21/2026, 4:19:09 PM
AAPL	$100.00	2/21/2026, 4:19:04 PM

Stock Alerts Dashboard

Real-time alerts from Spark & Flink processors

FLINK ALERTS	FLINK WINDOWED	SPARK ALERTS	SPARK WINDOWED
3	3	4	4

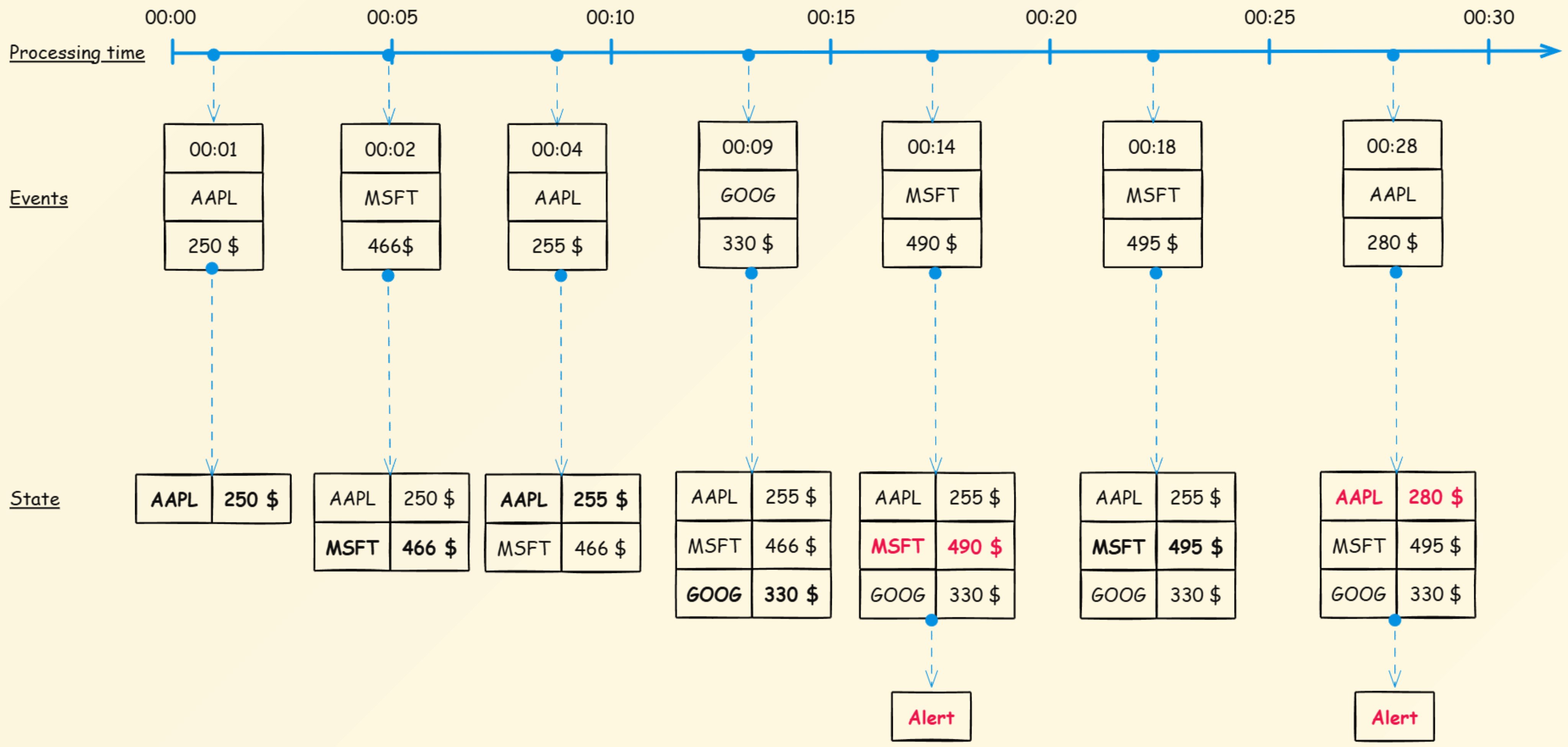
Flink Processor — 3 alerts

TIME	SYMBOL	PRICE	VARIATION %
4:19:43 PM	AAPL	$125.00	+8.7%
4:19:27 PM	AAPL	$115.00	+5.5%
4:19:11 PM	AAPL	$109.00	+7.92%

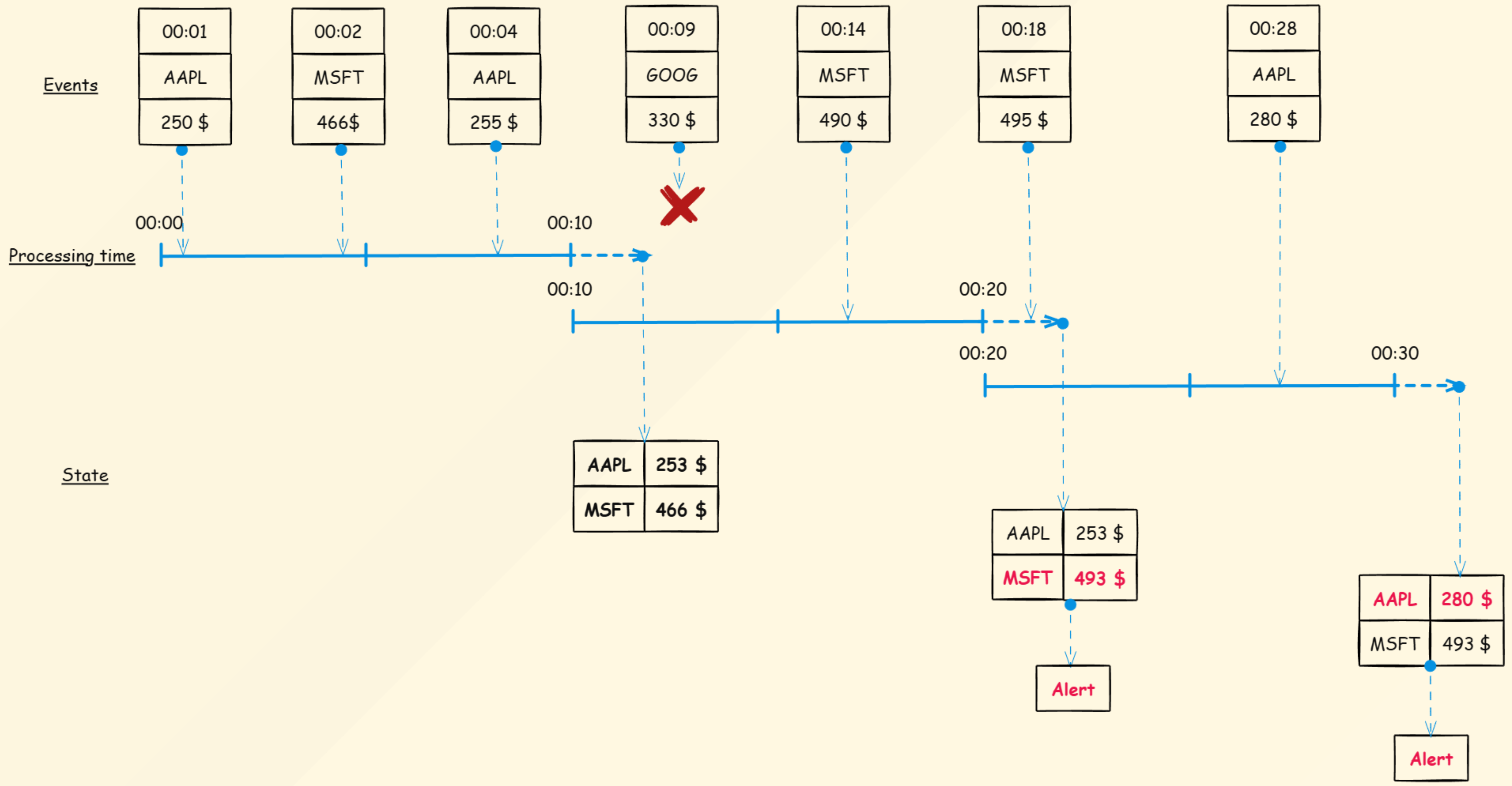
Flink Windowed Processor — 3 alerts

TIME	SYMBOL	PRICE	VARIATION %
4:19:44 PM	AAPL	$115.00	+5.5%
4:19:27 PM	AAPL	$109.00	+8.46%
4:19:27 PM	AAPL	$100.50	N/A

State management under the hood



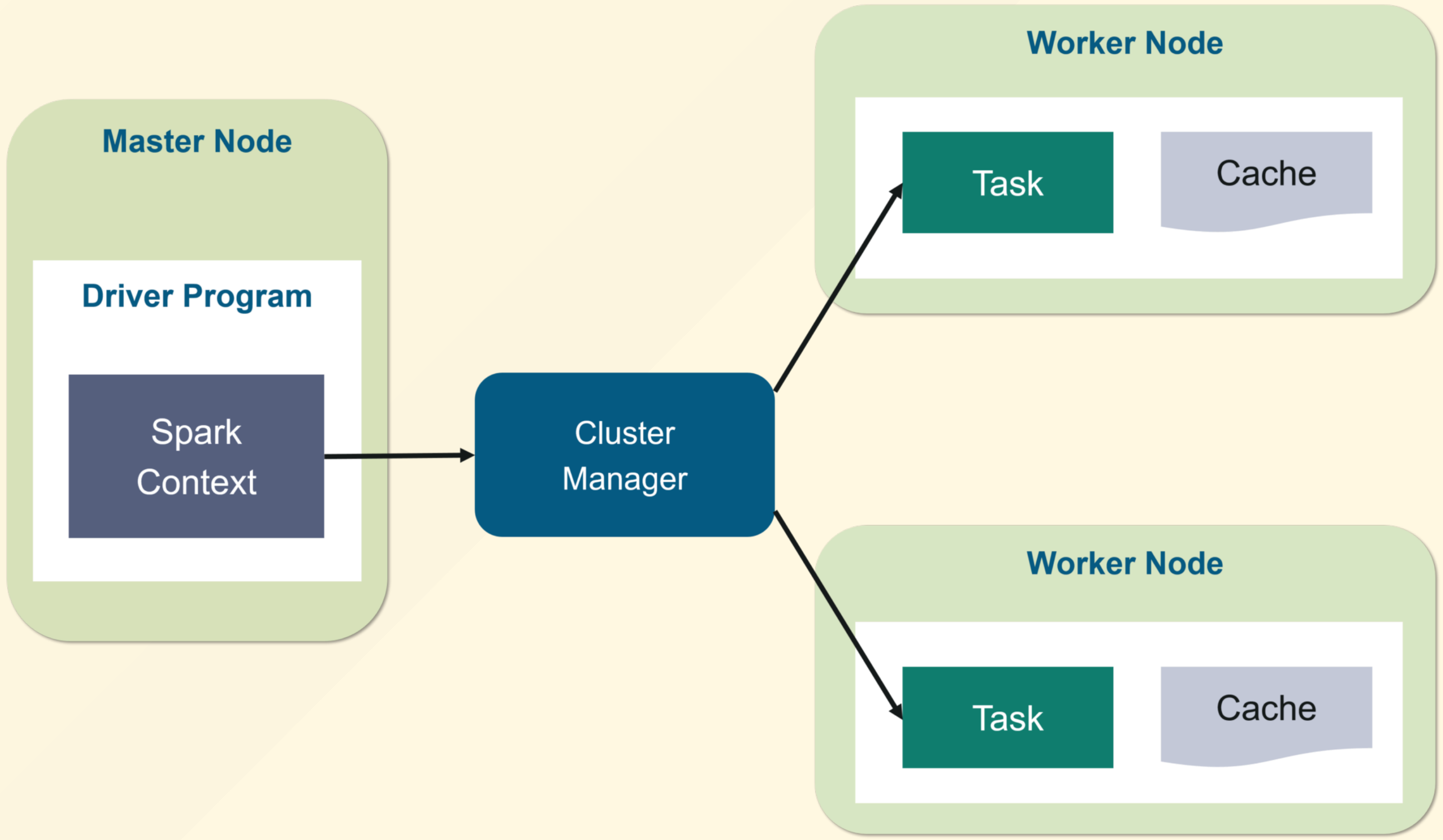
More advanced state management



Apache Spark

- **Micro-batch processing** - Divides stream into small batches
- **APIs**:
  - DStream API (legacy)
  - Structured Streaming (DataFrame API)
  - Spark SQL integration
- **Additional libraries**: MLlib, GraphX, Spark Connect, Pandas Spark
- **Mature ecosystem**

Spark Architecture



Spark Code Snippet

```
Dataset<Row> lines = spark
    .readStream()
    .format("socket")
    .option("host", SOCKET_HOST)
    .option("port", SOCKET_PORT)
    .load();

Dataset<StockData> stockData = lines
    .flatMap(new JsonParser(), Encoders.bean(StockData.class));

Dataset<String> alerts = stockData
    .groupByKey(
        (MapFunction<StockData, String>) StockData::getSymbol,
        Encoders.STRING()
    )
    .mapGroupsWithState(
        new PriceChangeDetector(),
        Encoders.bean(StockState.class),
        Encoders.STRING(),
        GroupStateTimeout.NoTimeout()
    )
    .filter((FilterFunction<String>) alert -> alert != null && !alert.isEmpty());

StreamingQuery query = alerts
    .writeStream()
    .outputMode("update")
    .format("console")
    .start();
```

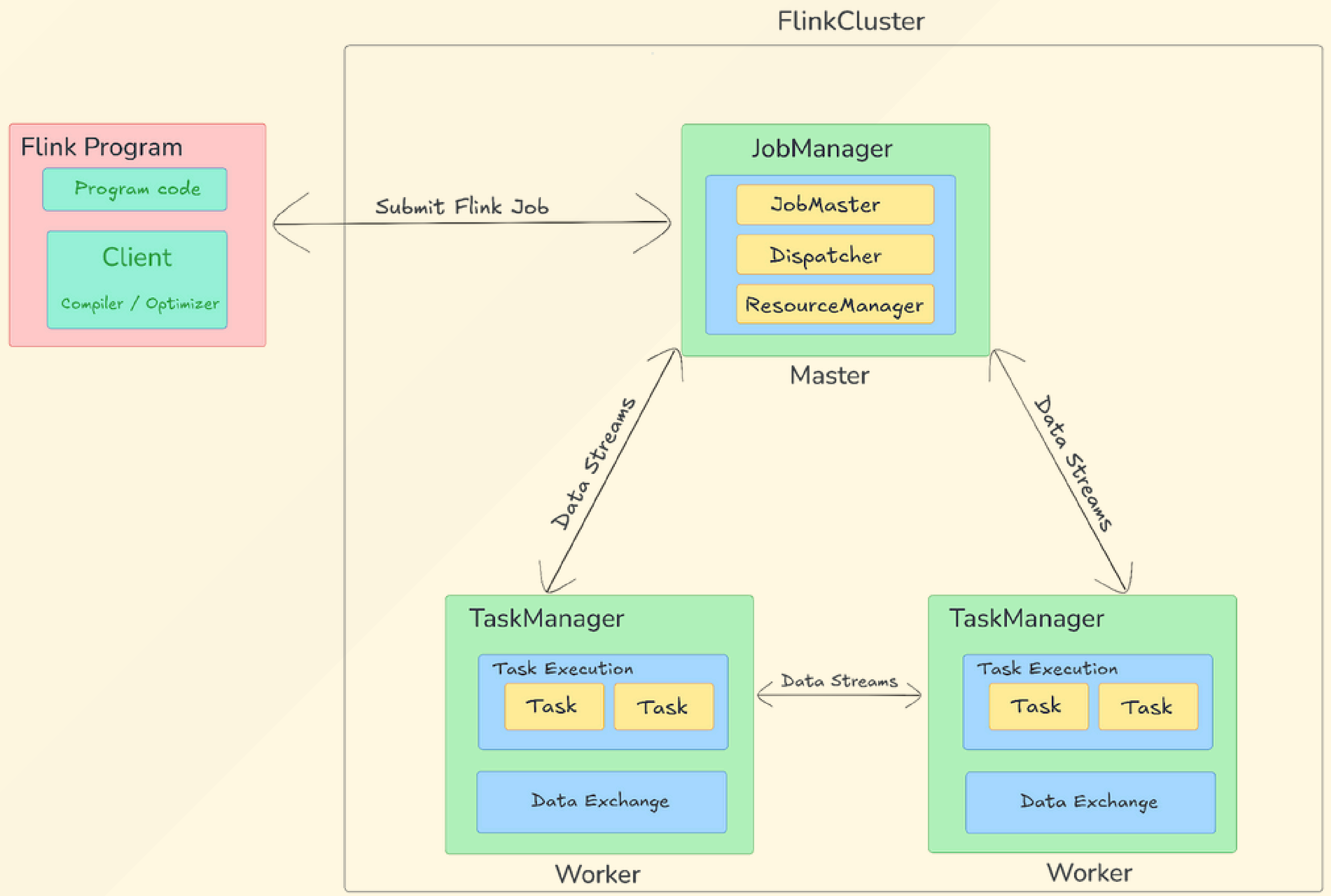
Apache Flink

Flink

- **Designed for real-time stream processing**
- **Processes events independently (true streaming)**
- **APIs**:
    - DataStream API (low-level, event-by-event processing)
    - Table API (relational operations)
    - Flink SQL
- **Additional libraries**: Flink CDC, Flink ML, Flink CEP, Flink Agents
- **Mature ecosystem (less than Spark but mature enough)**

Flink Architecture

Flink



Flink Code Sample

Flink

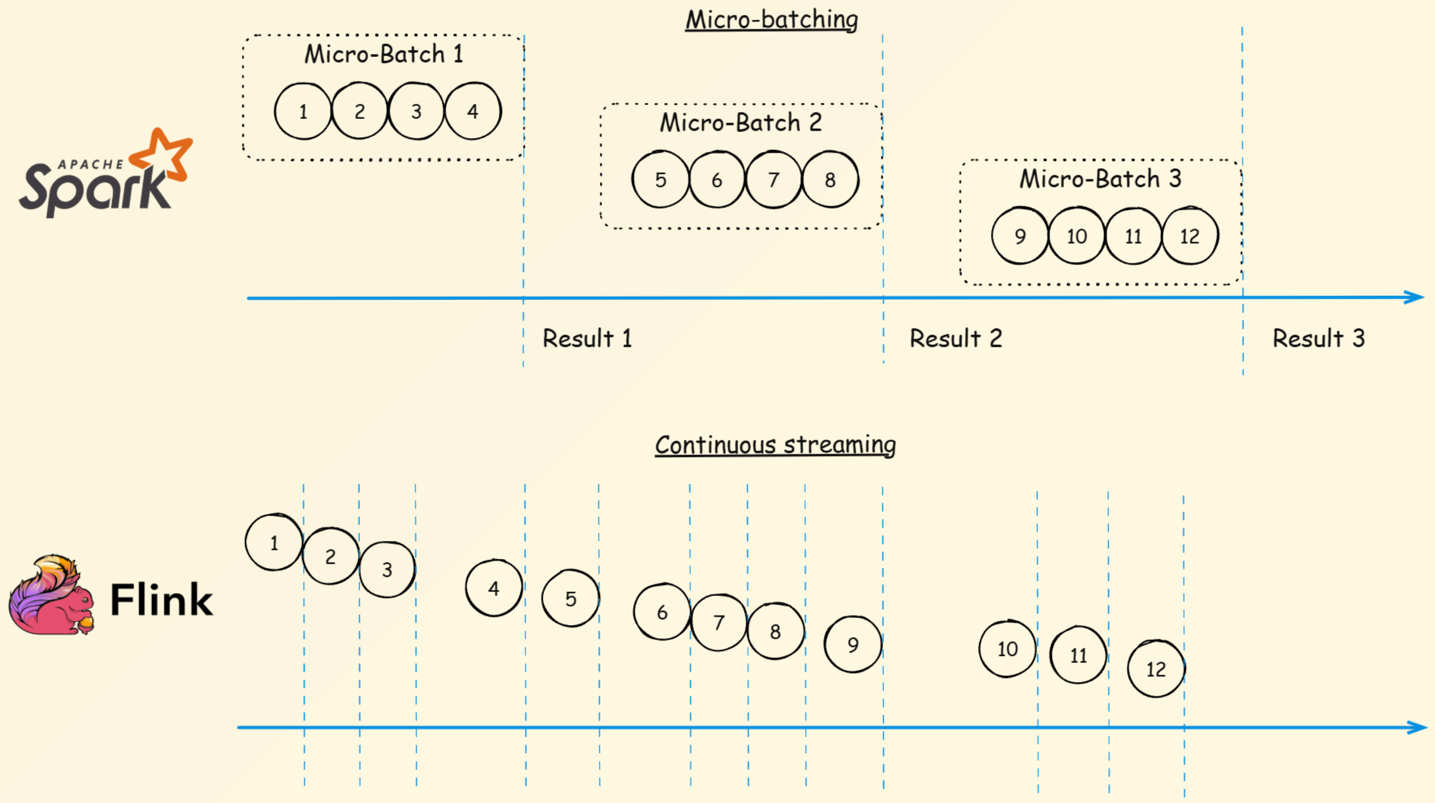
```java
final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

DataStream<String> socketStream = env.socketTextStream(SOCKET_HOST, SOCKET_PORT);

socketStream
    .flatMap(new JsonParser())
    .keyBy(StockData::getSymbol)
    .flatMap(new PriceChangeDetector())
    .print();

env.execute("Flink Stock Price Alert Processor");
```

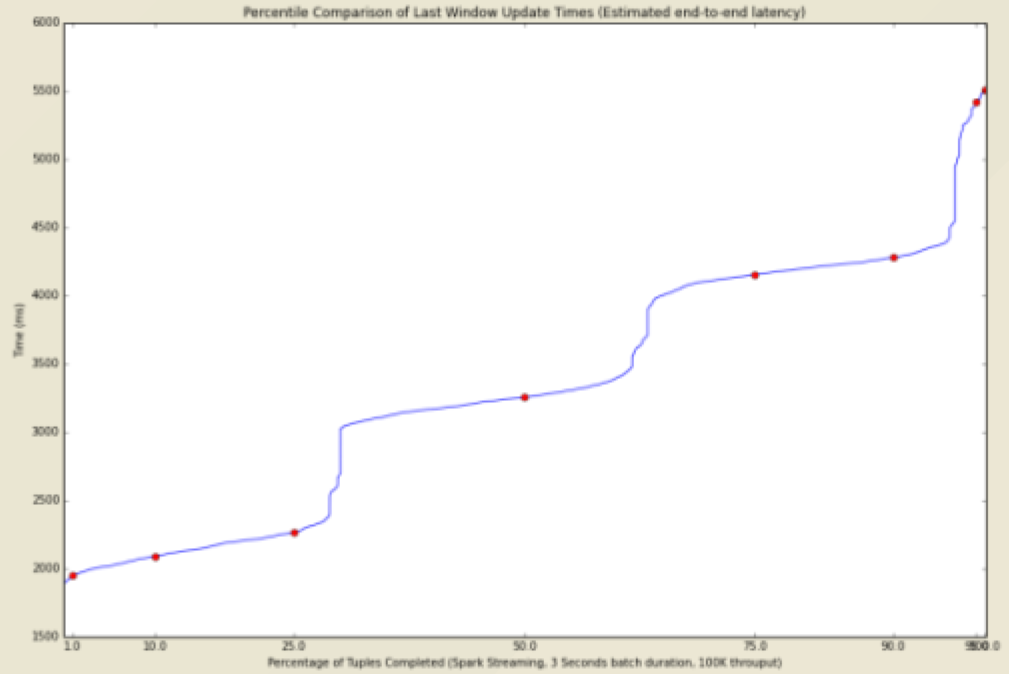
Micro-batch vs Continuous

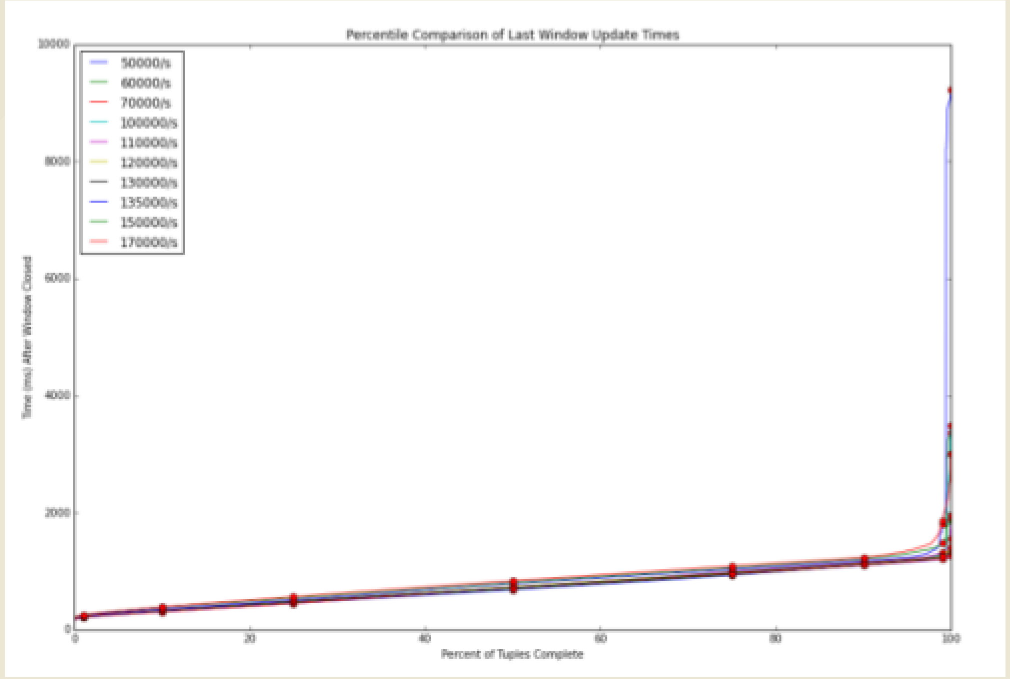


Performance

APACHE Spark

Flink





Source: https://yahooeng.tumblr.com/post/135321837876/benchmarking-streaming-computation-engines-at

Expected Features of a Streaming Platform

Feature	APACHE Spark	Flink
Supports multiple sources and sinks	✓	✓
Multiple connectors	✓	✓
Integration with monitoring and management tools	✓	✓
Language support (streaming)	Java, Scala, Python, SQL, R	Java, Scala, Python
Scalability (distributed)	✓	✓

Stateless Operators

Feature	APACHE Spark	Flink
Transformations (map / flatMap)	✓	✓
Filters	✓	✓
Projections (select)	✓	✓
Join with static data	✓	✓
Partitioning	✓	✓

Stateful Operators

Feature	APACHE Spark	Flink
Grouping	✓	✓
Aggregations (sum, count, avg, custom)	✓	✓
Windows (tumbling and sliding)	✓	✓
Stream to stream join	✓	✓
Sessionalizing	✓	✓

Handling the Pitfalls of Streaming

Feature	APACHE Spark	Flink
Time management (event time vs processing time)	✓	✓
Late-arriving data (Watermarks)	✓	✓
Fine-grained state management	Limited	✓
Delivery semantics (at-most once, at-least once, exactly once)	at-least once	exactly once
Fault tolerance	limited	✓
Handle backpressure	limited	✓

Bread and Butter Considerations

Feature	APACHE Spark	Flink
Learning curve	Moderate	Steep
Product maturity	High	Medium
Ecosystem maturity	High	Medium
Vendor support	High	Low
Scalability	Medium	High

And the winner is...

APACHE Spark	Flink
Mature and stable ecosystem	Low latency and predictable performance
More vendor / tooling support	Responsive event-driven apps
Data engineering / ML use cases	Robust state management and fault tolerance

Q & A

Repo : https://github.com/shbehna/streaming-spark-flink