Large scale stream processing with Apache Flink

Nikolay Stoitsev
Sr. Software Engineer at Uber Tech Sofia
Stream Processing?
Stream Processing?

User Interaction Logs
Stream Processing?

User Interaction Logs

Application Logs
Stream Processing?

User Interaction Logs

Application Logs

Sensor Data
Stream Processing?

User Interaction Logs

Application Logs

Sensor Data

Database Commit Logs
Infinite Dataset
Big Latency
Apache Storm
storm.apache.org
High-latency & accurate

vs.

Low-latency & approximation
Lambda architecture
Questioning the Lambda Architecture

The Lambda Architecture has its merits, but alternatives are worth exploring.

By Jay Kreps. July 2, 2014

The call for proposals is now open for the Strata Data Conference in London, April 29-May 2, 2019.

Nathan Marz wrote a popular blog post describing an idea he called the Lambda Architecture (“How to beat

https://www.oreilly.com/ideas/questioning-the-lambda-architecture
Use Apache Kafka

Durable, scalable, fault-tolerant
# Customer Satisfaction

95% satisfaction rating

You're a customer champion
Customers love ordering from you! See what they're saying about your dishes below.

## Ratings

<table>
<thead>
<tr>
<th>Item</th>
<th>Satisfaction Rating</th>
<th>Negative Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cobb's Salad</td>
<td>100% (54)</td>
<td></td>
</tr>
<tr>
<td>Crispy Fried Chicken</td>
<td>100% (31)</td>
<td></td>
</tr>
<tr>
<td>Kettle Corn on the Cobb</td>
<td>100% (24)</td>
<td></td>
</tr>
<tr>
<td>Cobb's Special</td>
<td>100% (19)</td>
<td></td>
</tr>
<tr>
<td>Corn Meal</td>
<td>100% (9)</td>
<td></td>
</tr>
<tr>
<td>Spicy Fried Chicken</td>
<td>92% (11)</td>
<td></td>
</tr>
<tr>
<td>Corn on the Cobb</td>
<td>92% (10)</td>
<td></td>
</tr>
<tr>
<td>Cookies and Corn</td>
<td>89% (7)</td>
<td></td>
</tr>
</tbody>
</table>
Metrics we want to track

- Net payout
- Daily items sold
- Weekly items sold
- Order acceptance rate
- Order preparation speed
- Item rating
Real time
Scalable
Granular
Highly available
Order Stream
Payment Stream
User Rating Stream
What is Samza?

Apache Samza is a distributed stream processing framework. It uses Apache Kafka for messaging, and Apache Hadoop YARN to provide fault tolerance, processor isolation, security, and resource management.

- **Simple API**: Unlike most low-level messaging system APIs, Samza provides a very simple callback-based “process message” API comparable to MapReduce.
- **Managed state**: Samza manages snapshotting and restoration of a stream processor’s state. When the processor is restarted, Samza restores its state to a consistent snapshot. Samza is built to handle large amounts of state (many gigabytes per partition).
- **Fault tolerance**: Whenever a machine in the cluster fails, Samza works with YARN to transparently migrate your tasks to another machine.
- **Durability**: Samza uses Kafka to guarantee that messages are processed in the order they were written to a partition, and that no messages are ever lost.
- **Scalability**: Samza is partitioned and distributed at every level. Kafka provides ordered, partitioned, replayable, fault-tolerant streams. YARN provides a distributed environment for Samza containers to run in.
- **Pluggable**: Though Samza works out of the box with Kafka and YARN, Samza provides a pluggable API that lets you run Samza with other messaging systems and execution environments.
- **Processor isolation**: Samza works with Apache YARN, which supports Hadoop’s security model, and resource isolation through Linux CGroups.
Apache Flink

flink.apache.org
Everything is a batch

vs.

Everything is a stream
Dataflow graph
DataStream<String> lines = env.addSource(
    new FlinkKafkaConsumer<>(...));

DataStream<Event> events = lines.map((line) -> parse(line));

DataStream<Statistics> stats = events
    .keyBy("id")
    .timeWindow(Time.seconds(10))
    .apply(new MyWindowAggregationFunction());

stats.addSink(new RollingSink(path));
```java
DataStream<String> lines = env.addSource(
    new FlinkKafkaConsumer<>(...));
DataStream<Event> events = lines.map((line) -> parse(line));
DataStream<Statistics> stats = events
    .keyBy("id")
    .timeWindow(Time.seconds(10))
    .apply(new MyWindowAggregationFunction());
stats.addSink(new RollingSink(path));
```
Fault tolerant
Lightweight Asynchronous Snapshots for Distributed Dataflows

Paris Carbone,
Gyula Fóra,
Stephan Ewen
Seif Haridi
Kostas Tzoumas
Barrier

Snapshot Store

Operator

Msg  Barrier  Msg  Msg  Barrier  Msg  Msg  Msg
Exactly Once Processing
Can handle very large state
Joining Streams
Apache Flink

- Can join streams
- Fault tolerant
- Exactly Once Processing
- Combines stream and batch processing
... but it requires Java/Scala code
Scalable, efficient and robust
SQL-based streaming analytics platform at scale

calcite  flink  sql  uber  streaming  stream  analytics  data

19 commits  1 branch  0 releases  6 contributors  Apache-2.0

walterdrr change YARN setup Java Doc for 1.5

athenax-backend change YARN setup Java Doc for 1.5

athenax-tests Upgrade AthenaX to Apache Flink 1.5.0 (#24)

athenax-vm-api Upgrade AthenaX to Apache Flink 1.5.0 (#24)

athenax-vm-compiler Upgrade AthenaX to Apache Flink 1.5.0 (#24)

athenax-vm-connectors Upgrade AthenaX to Apache Flink 1.5.0 (#24)
SQL → what data to analyze

Flink → how to analyze it
SELECT
    HOP_START(rowtime, INTERVAL '1' MINUTE, INTERVAL '15' MINUTE) AS window_start,
    restaurant_uuid,
    COUNT(*) AS total_order
FROM ubereats_workflow
WHERE state = 'CREATED'
GROUP BY
    restaurant_uuid,
    HOP(rowtime, INTERVAL '1' MINUTE, INTERVAL '15' MINUTE)
CREATE FUNCTION AirportCode AS ...;

SELECT

    AirportCode(location.lng,location.lat) AS airport
    driver_id AS driver_id,

FROM

    event_user_driver_app

WHERE

    NAME = 'trip_start'
SELECT
    w.created_timestamp,
    w.datestr,
    w.restaurant_uid,
    w.order_job_uuid,
    o.price,
    o.currency,
FROM
    ubereats_workflow_etd_summary w
JOIN
    ubereats_order_state_changes o
ON
    o.job_uuid = w.order_job_uuid
WHERE
    w.status IN ('CANCELED_BY_EATER', 'UNFULFILLED')
    AND
    w.proctime
    BETWEEN
        o.proctime - INTERVAL '60' SECOND
    AND
        o.proctime + INTERVAL '60' SECOND
Project:
\[ r' = (\text{created\_timestamp}, \text{datestr}, \text{restaurant\_uuid}, \text{order\_job\_uuid}, \text{price}, \text{currency}) \]

Project:
\[ r' = (\text{created\_timestamp}, \ldots, \text{price}, \text{currency}, \ldots) \]

Join:
\[ r.\text{order\_uuid} = o.\text{job\_uuid}, \text{status} \in (\ldots) = \text{true}, w.\text{proctime} \in [o.\text{proctime} - 60000, o.\text{proctime} + 60000] \]

Project:
\[ r = (\text{order\_job\_uuid}, \ldots, \text{status} \in (\ldots)) \]

TableScan:
\[ \text{ubereats\_order\_state\_changes} \]

TableScan:
\[ \text{ubereats\_workflow\_etd\_summary} \]

(a) original logical plan
Project:
\[ r' = (\text{created\_timestamp}, \text{datestr}, \text{restaurant\_uuid}, \text{order\_job\_uuid}, \text{price}, \text{currency}) \]

Join:
\[
\begin{align*}
& r.\text{order\_job\_uuid} = r'.\text{job\_uuid}, \\
& r.\text{proctime} \in [r'.\text{proctime} - 60000, r'.\text{proctime} + 60000]
\end{align*}
\]

Calc:
\[
\begin{align*}
& r = (..., \text{order\_job\_uuid}), \\
& \text{condition} = w.\text{status} \in (...))
\end{align*}
\]

TableScan:
\[ \text{ubereats\_workflow\_etd\_summary} \]

Calc:
\[
\begin{align*}
& r' = (\text{job\_uuid}, \text{price}, \text{currency})
\end{align*}
\]

TableScan:
\[ \text{ubereats\_order\_state\_changes} \]
TwoInputStreamTask:

\[ r' = (\text{created\_timestamp}, \text{datestr}, \text{restaurant\_uuid}, \text{order\_job\_uuid}, \text{price}, \text{currency}) \]

SourceStreamTask:

\text{ubereats\_workflow\_etd\_summary} \rightarrow 
\[ r = (\text{created\_timestamp}, ...) \]

SourceStreamTask:

\text{ubereats\_order\_state\_changes} \rightarrow r'

= (\text{job\_uuid}, \text{price}, \text{currency})

(c) compiled data flow program
Resource estimation and auto scaling
Monitoring and automatic failure recovery
Introducing AthenaX, Uber Engineering’s Open Source Streaming Analytics Platform

By Haohui Mai, Bill Liu, & Naveen Cherukuri

October 9, 2017

eng.uber.com/athenax
Thanks!

Nikolay Stoitsev @ Uber