Stream Processing Beyond Streaming Data with Apache Flink

Stephan Ewen
Co-creator and PMC of Apache Flink
Ververica (formerly dataArtisans, now part of Alibaba Group)
About Ververica

Original creators of Apache Flink®

Enterprise Stream Processing
Apache Flink at Alibaba Group

The "Singles Day" (11/11/2018)

<table>
<thead>
<tr>
<th>machines</th>
<th>queries</th>
<th>throughput</th>
<th>latency</th>
<th>state size</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K</td>
<td>10K</td>
<td>1.7B</td>
<td>Sub-Second</td>
<td>100TB</td>
</tr>
</tbody>
</table>
Some Apache Flink Users

<table>
<thead>
<tr>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>airbnb</td>
</tr>
<tr>
<td>Airbus Defense &amp; Space</td>
</tr>
<tr>
<td>Alibaba Group</td>
</tr>
<tr>
<td>aws</td>
</tr>
<tr>
<td>BetterCloud</td>
</tr>
<tr>
<td>bol.com</td>
</tr>
<tr>
<td>bouygues</td>
</tr>
<tr>
<td>CapitalOne</td>
</tr>
<tr>
<td>Comcast</td>
</tr>
<tr>
<td>criteo</td>
</tr>
<tr>
<td>John Deere</td>
</tr>
<tr>
<td>DELL EMC</td>
</tr>
<tr>
<td>DiDi</td>
</tr>
<tr>
<td>eBay</td>
</tr>
<tr>
<td>Ericsson</td>
</tr>
<tr>
<td>gojek</td>
</tr>
<tr>
<td>Goldman Sachs</td>
</tr>
<tr>
<td>Huawei</td>
</tr>
<tr>
<td>King</td>
</tr>
<tr>
<td>ING</td>
</tr>
<tr>
<td>lyf</td>
</tr>
<tr>
<td>Klaviyo</td>
</tr>
<tr>
<td>MediaMath</td>
</tr>
<tr>
<td>Netflix</td>
</tr>
<tr>
<td>Oppo</td>
</tr>
<tr>
<td>OVH.com</td>
</tr>
<tr>
<td>Pinterest</td>
</tr>
<tr>
<td>Razorpay</td>
</tr>
<tr>
<td>SK telecom</td>
</tr>
<tr>
<td>Splunk</td>
</tr>
<tr>
<td>Stripe</td>
</tr>
<tr>
<td>Telefonica</td>
</tr>
<tr>
<td>Tencent 腾讯</td>
</tr>
<tr>
<td>theTradeDesk</td>
</tr>
<tr>
<td>Uber</td>
</tr>
<tr>
<td>Workday</td>
</tr>
<tr>
<td>Xiaomi</td>
</tr>
<tr>
<td>Yelp</td>
</tr>
</tbody>
</table>

Sources: Powered by Flink, Speakers – Flink Forward San Francisco 2019, Speakers – Flink Forward Europe 2019
The Spectrum of Streaming Data Use Cases

- Batch Processing
- Continuous Processing
- Data Pipelines
- Streaming Analytics
- Event-driven Applications
- Transactional Applications

more lag time  more real time
Apache Flink: Analytics and Applications on Streaming Data

Stateful Computations over Data Streams

Stateful Stream Processing
- Streams, State, Time

Event-driven Applications
- Stateful Functions

Flink Runtime
- Stateful Computations over Data Streams

Streaming Analytics
- SQL and Tables
Everything is a Stream

Streams Of Records in a Log or MQ
[e.g., Apache Kafka or AWS Kinesis ...]
Everything is a Stream

Stream of Requests/Responses to/from Services

→ event sourcing architecture
Everything is a Stream

Stream of Rows in a Table or in Files

- 2016-3-1 12:00 am
- 2016-3-1 1:00 am
- 2016-3-1 2:00 am
- ... 2016-3-11 10:00 pm
- 2016-3-11 11:00 pm
- 2016-3-12 12:00 am
- 2016-3-12 1:00 am
- 2016-3-12 2:00 am
- 2016-3-12 3:00 am
Everything is a Stream

Stream of Rows in a Table or in Files

a batch
Bounded and Unbounded Streams
Programs are DAGs of Stateful Computation Steps
val lines: DataStream[String] = env.addSource(new FlinkKafkaConsumer011(...))
val events: DataStream[Event] = lines.map((line) => parse(line))
val stats: DataStream[Statistic] = stream 
  .keyBy("sensor")
  .timeWindow(Time.seconds(5))
  .sum(new MyAggregationFunction())
stats.addSink(new RollingSink(path))
Event Sourcing + Memory Image

- Event / command
- Event stream
  - Persists events (temporarily)
- Periodically snapshot the memory

Also works with RocksDB (LSM trees)

Process

- Main memory
  - Update local variables/structures
Event Sourcing + Memory Image

Recovery: Restore snapshot and replay events since snapshot

Process

event stream
persists events (temporarily)
Consistent Distributed Snapshots
Checkpoints for Recovery / Rollback / Evolution / Cloning / ...

Rolling back computation

Re-processing

Reset positions in input streams

Re-load state
Versioning the state of applications

Time

App. A

Savepoint

App. B

Savepoint

Savepoint

App. C

Savepoint
Apache Flink: Analytics and Applications on Streaming Data

Streaming Analytics

SQL and Tables

Stateful Stream Processing

Streams, State, Time

Event-driven Applications

Stateful Functions

Flink Runtime

Stateful Computations over Data Streams
SELECT
   room,
   TUMBLE_END(rowtime, INTERVAL '1' HOUR),
   AVG(temperature)
FROM
   sensors
GROUP BY
   TUMBLE(rowtime, INTERVAL '1' HOUR), room
Interpreting Streams as Tables

<table>
<thead>
<tr>
<th>user</th>
<th>lastLogin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>2017-07-01</td>
</tr>
<tr>
<td>Bob</td>
<td>2017-06-01</td>
</tr>
<tr>
<td>Liz</td>
<td>2017-05-01</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
SELECT
  room,
  TUMBLE_END(rowtime, INTERVAL '1' HOUR),
  AVG(temperature)
FROM
  sensors
GROUP BY
  TUMBLE(rowtime, INTERVAL '1' HOUR), room
SELECT
    room,
    TUMBLE_END(rowtime, INTERVAL '1' HOUR),
    AVG(temperature)
FROM
    sensors
GROUP BY
    TUMBLE(rowtime, INTERVAL '1' HOUR), room
update database with changes
output result changes as stream
Many handy SQL features: Temporal Joins, Pattern Matching, ...

```
SELECT tf.time, tf.price * rh.rate as conv_fare
FROM taxiFare AS tf
LATERAL TABLE (Rates(tf.time)) AS rh
WHERE tf.currency = rh.currency;
```

```
SELECT rideId, timeDiff(startT, endT) / 60000 AS durationMin
FROM Rides
MATCH_RECOGNIZE (PARTITION BY rideId
ORDER BY rideTime
MEASURES
  s.rideTime AS startT,
  E.rideTime AS endT
AFTER MATCH SKIP PAST LAST ROW
PATTERN (S E)
DEFINE
  S AS S.isStart,
  E AS NOT E.isStart
);
```
Apache Flink: Analytics and Applications on Streaming Data

- Streaming Analytics
  - SQL and Tables

- Stateful Stream Processing
  - Streams, State, Time

- Event-driven Applications
  - Stateful Functions

- Flink Runtime
  - Stateful Computations over Data Streams
Stateful Functions is currently a standalone project

https://statefun.io/
https://github.com/ververica/stateful-functions

We are contributing it to Apache Flink during these weeks

The project is still new and dynamic.
A good time to get involved to get traction ;-)
Stream Processing

- Can we combine some of these properties?

- composability

- state management

- performance

F-a-a-S

- event-driven

- simplicity / generality

- lightweight resources
Stateful Functions

mass storage (S3, GCF, ECS, HDFS, ...)

event ingress

f(a, b) → f(a, b) → f(a, b) → f(a, b) → f(a, b) → f(a, b)

event egress

snapshot state

© 2019 Ververica
Event ingresses supply events that trigger functions
Multiple functions send event to each other
Arbitrary addressing, no restriction to DAG
Functions have locally embedded state
Stateful Functions

State and messaging are consistent with exactly-once semantics
Stateful Functions

No database required
All persistence goes directly to blob storage
Stateful Functions

Event egresses to respond via event streams
Logical/Virtual Instances

Shard 1

- Memory:
  - A
  - B
  - C
  - F
  - G
  - H
  - I

- Secondary storage:
  - K
  - L
  - M
  - N

Shard 2

- Function virtual instance:
  - D
  - E
Logical/Virtual Instances

Shard 1

Shard 2

message to "K"

K.invoke(message)

possibly evict other

load "K"
Apache Flink is the State and Event Streaming Fabric

Conceptual Dataflow

Apache Flink Dataflow Graph
Running Stateful Functions on Apache Flink

Exactly-once checkpointing for streaming loops

loop feedback
Example: Ride Sharing App

Driver status updates

Passenger ride requests

Driver

Geo-index

Ride

Passenger

Ride status update

update

bidding

free

booked

bid

Inform / Book

update cell

lookup

create

bill

confirmed

riding

seeking

confirmed

billing

update

confirmed

update cell

update cell

update cell
Stream Processing
Streaming SQL

- event/stream-centric

- data preparation
  combining knowledge/information
  filtering, enriching,
  aggregating, joining events

Stateful Functions

- state-centric

- coordination,
  (interacting) state machines
  complex event/state
  interactions

F-a-a-S

- stateless / compute-centric

- "occasional" actions or
  spiky loads
  compute-intensive
  or blocking
Putting it all together: Ridesharing again

Stream Processing
- traffic models
- demand forecast & pricing

Driver position updates

Passenger updates

Driver status updates

Stateful Functions
- ride life-cycle
- driver-to-ride matching

Billing

Faas
- render map/route image
- create a receipt PDF
- send email

© 2019 Ververica
Thank you!

If you liked this, engage with the Apache Flink® community

- Try Flink and help us improve it
- Contribute docs, code, tutorials
- Share your use cases and ideas
- Join a Flink Meetup
- Join the Flink Forward conference

@StephanEwen
@ApacheFlink

https://flink.apache.org/