The Future of Real-Time in Spark

Reynold Xin @rxin
Spark Summit, New York, Feb 18, 2016
Why Real-Time?

Making decisions faster is valuable.

- Preventing credit card fraud
- Monitoring industrial machinery
- Human-facing dashboards
- …
Streaming Engine

Noun.

Takes an input stream and produces an output stream.
Spark Unified Stack

- SQL
- Streaming
- MLlib
- GraphX

Spark Core
Spark Unified Stack

- Streaming
- SQL
- MLlib
- GraphX

Spark Core

Introduced 3 years ago in Spark 0.7
50% users consider most important part of Spark
Spark Streaming

- First attempt at unifying streaming and batch
- State management built in
- Exactly once semantics
- Features required for large clusters
  - Straggler mitigation, dynamic load balancing, fast fault-recovery
Streaming computations don’t run in isolation.
Use Case: Fraud Detection

Ad-hoc analyze historic data

STREAM

ANOMALY

Machine learning model continuously updates to detect new anomalies
Continuous Application

noun.

An end-to-end application that acts on real-time data.
Challenges Building Continuous Applications

Integration with non-streaming systems often an after-thought
• Interactive, batch, relational databases, machine learning, …

Streaming programming models are complex
Integration Example

Stream
(home.html, 10:08)
(product.html, 10:09)
(home.html, 10:10)

Streaming engine

MySQL

What can go wrong?
• Late events
• Partial outputs to MySQL
• State recovery on failure
• Distributed reads/writes
• ...

<table>
<thead>
<tr>
<th>Page</th>
<th>Minute</th>
<th>Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>home</td>
<td>10:09</td>
<td>21</td>
</tr>
<tr>
<td>pricing</td>
<td>10:10</td>
<td>30</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Complex Programming Models

Data
Late arrival, varying distribution over time, …

Processing
Business logic change & new ops
/windows, sessions/

Output
How do we define output over time & correctness?
Structured Streaming
The simplest way to perform streaming analytics is not having to **reason** about streaming.
Spark 1.3
Static DataFrames

Spark 2.0
Infinite DataFrames

Single API!
Structured Streaming

High-level streaming API built on Spark SQL engine
  • Runs the same queries on DataFrames
  • Event time, windowing, sessions, sources & sinks

Unifies streaming, interactive and batch queries
  • Aggregate data in a stream, then serve using JDBC
  • Change queries at runtime
  • Build and apply ML models
Model

Time

1. data up to PT 1
2. data up to PT 2
3. data up to PT 3

Input

Query

Result

output for data at 1
output for data at 2
output for data at 3

Output

complete output

Trigger: every 1 sec
Model Details

Input sources: append-only tables

Queries: new operators for windowing, sessions, etc

Triggers: based on time (e.g. every 1 sec)

Output modes: complete, deltas, update-in-place
Example: ETL

**Input:** files in S3

**Query:** map (transform each record)

**Trigger:** “every 5 sec”

**Output mode:** “new records”, into S3 sink
Example: Page View Count

**Input:** records in Kafka

**Query:**
```
select count(*) group by page, minute(evtime)
```

**Trigger:** “every 5 sec”

**Output mode:** “update-in-place”, into MySQL sink

**Note:** this will automatically update “old” records on late data!
Execution

**Logically:**
DataFrame operations on static data (i.e. as easy to understand as batch)

**Physically:**
Spark automatically runs the query in streaming fashion (i.e. incrementally and continuously)
Example: Batch Aggregation

```python
logs = ctx.read.format("json").open("s3://logs")

logs.groupBy(logs.user_id).agg(sum(logs.time))
   .write.format("jdbc")
   .save("jdbc:mysql//...")
```
Example: Continuous Aggregation

```scala
logs = ctx.read.format("json").stream("s3://logs")

logs.groupBy(logs.user_id).agg(sum(logs.time))
  .write.format("jdbc")
  .stream("jdbc:mysql://...")
```
Automatic Incremental Execution

T = 0

T = 1

T = 2

...
Rest of Spark will follow

- Interactive queries should just work
- Spark’s data source API will be updated to support seamless streaming integration
  - Exactly once semantics end-to-end
  - Different output modes (complete, delta, update-in-place)
- ML algorithms will be updated too
What can we do with this that’s hard with other engines?

Ad-hoc, interactive queries

Dynamic changing queries

Benefits of Spark: elastic scaling, straggler mitigation, etc
Use Case: Fraud Detection

- Analyze Historic Data
- Machine Learning Model continuously updates to detect new anomalies
Timeline

Spark 2.0
• API foundation
• Kafka, file systems, and databases
• Event-time aggregations

Spark 2.1 +
• Continuous SQL
• BI app integration
• Other streaming sources / sinks
• Machine learning
Thank you.

@rxin