Introduction to Spark

Marcin Tustin - Consultant Data Engineer
https://medium.com/@albamus
marcin.tustin@gmail.com
<table>
<thead>
<tr>
<th>Topic</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who am I</td>
<td>1 min</td>
</tr>
<tr>
<td>Who are you</td>
<td>4 min</td>
</tr>
<tr>
<td>Agenda</td>
<td>2 min</td>
</tr>
<tr>
<td>The big data landscape</td>
<td>20 min</td>
</tr>
<tr>
<td>What is Spark</td>
<td>20 min</td>
</tr>
<tr>
<td>Why not Spark</td>
<td>10 min</td>
</tr>
<tr>
<td>Why Spark</td>
<td>3 min</td>
</tr>
<tr>
<td>How does Spark work?</td>
<td>5 min</td>
</tr>
<tr>
<td>Let’s make a Spark program!</td>
<td>Rest of session</td>
</tr>
</tbody>
</table>
Who am I?

• Contribute to Spark

• Built Handy’s data pipeline - used to use Spark every day

• Blog at https://medium.com/@albamus

• Available for consulting on big data technology, governance, and processes
Who are you?
<table>
<thead>
<tr>
<th>Processing type</th>
<th>Bounded data</th>
<th>Unbounded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>Storm, Heron, Flink, Millwheel, Kafka Streams, Spark</td>
<td>NA</td>
</tr>
<tr>
<td>Batch</td>
<td>Spark, Hadoop, Hive</td>
<td>NA</td>
</tr>
</tbody>
</table>
### The Big Data Landscape (2): Application types

<table>
<thead>
<tr>
<th>Application types</th>
<th>Bounded data</th>
<th>Unbounded data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query processing/data retrieval</strong></td>
<td>Hive, pig, spark, teradata, presto, redshift, bigtable, elasticsearch, cassandra etc.</td>
<td>Influx, storm, heron, spark, geode, kinesis stream analytics, elasticsearch</td>
</tr>
<tr>
<td><strong>Analytic</strong></td>
<td>Spark, Map Reduce, Pig, Flink</td>
<td>Spark, Storm, Heron, Flink</td>
</tr>
<tr>
<td><strong>Service-oriented</strong></td>
<td>Web services</td>
<td>Geode, Kafka Streams, Actors systems</td>
</tr>
</tbody>
</table>
The Big Data Landscape (3): The stack

Application framework
- Spark
- Flink
- Heron
- Hive
- Map Reduce

Caching
- Alluxio
- Geode

Execution
- YARN
- MESOS
- Kubernetes
- ECS

Storage
- HDFS
- S3 (etc)
- Cassandra
- RDBMS

ONE DOES NOT SIMPLY PROCESS DATA WITHOUT A CLUSTER
## What is Spark (1)

<table>
<thead>
<tr>
<th>Processing type</th>
<th>Bounded data</th>
<th>Unbounded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>Storm, Heron, Flink, Millwheel, Kafka Streams, Spark</td>
<td>NA</td>
</tr>
<tr>
<td>Batch</td>
<td>Spark, Hadoop, Hive</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processing type</th>
<th>Bounded data</th>
<th>Unbounded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query processing/data</td>
<td>Hive, pig, spark, teradata, drill, redshift, bigtable</td>
<td>Influx, storm, heron, spark, geode, kinesis</td>
</tr>
<tr>
<td>Analytic</td>
<td>Spark, Map Reduce, Pig, Flink</td>
<td>Spark, Storm, Heron, Flink</td>
</tr>
<tr>
<td>Service-oriented</td>
<td>Web services</td>
<td>Geode, Kafka Streams, Actors systems</td>
</tr>
</tbody>
</table>
What is Spark (2)

- Program/Library NOT infrastructure
- Analytics and querying engine for big data
  - Cluster executed
  - **Max efficiency + speed at high volume**
- Stream and batch oriented
- NOT a database or storage engine
What is Spark (3): Program and Framework

- Provides a complete interface for programming querying and analytic applications
- Declarative, functional programming model
- Has many useful integrations ("window to the big data world")
- Good for "plumbing" applications like data ingestion
- Machine learning and graph libraries built in
What is Spark (4): Analytics and querying engine

- Handles details of resilience, distribution, execution scheduling for you

- **Awesome** performance compared to map reduce

- Not a complete job runner (e.g. you want airflow to run things regularly) or activity monitor (e.g. use marathon/aurora or ECS to ensure your task is running)

- Cluster execution allows programmer to ignore resiliency and resource limits - frameworks handle resource limiting
How cluster execution works

http://spark.apache.org/docs/latest/cluster-overview.html
### Why not Spark (1)

<table>
<thead>
<tr>
<th>Processing type</th>
<th>Bounded data</th>
<th>Unbounded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>Storm, Heron, Flink, Millwheel, Kafka Streams, Spark</td>
<td>NA</td>
</tr>
<tr>
<td>Batch</td>
<td>Spark, Hadoop, Hive</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Bounded data</th>
<th>Unbounded data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query processing/data</td>
<td>Hive, pig, spark, teradata, drill, redshift, bigtable</td>
<td>Influx, storm, heron, spark, geode, kinesis</td>
</tr>
<tr>
<td>Analytic</td>
<td>Spark, Map Reduce, Pig, Flink</td>
<td>Spark, Storm, Heron, Flink</td>
</tr>
<tr>
<td>Service-oriented</td>
<td>Web services</td>
<td>Geode, Kafka Streams, Actors systems</td>
</tr>
</tbody>
</table>
Why not Spark (2): Latency

• If latency is more important than resilience:
  • For programmed analytic workloads, choose Flink or Heron
  • For simple SQL based querying choose Presto
Why not Spark (3): Clusterless execution

• If you don’t want to have an execution cluster:

  • For simple SQL based querying choose Presto (it has its own execution engine; also Amazon has a hosted Presto service branded as Athena)

  • Kafka Streams is good for streaming execution without a cluster

  • Deploy some web services behind a load balancer

  • Use ECS or Kubernetes?
Why not Spark (4): Many very small tasks

- Consider using something like lambda, or just writing with any cluster-free framework
  - Node.js?
  - Kubernetes?
  - Mesos?
Why not Spark (5): Data Storage

- It’s not a database
- Can be useful for warehouse maintenance tasks
- Can be useful for low frequency data ingestion (once a day or less)
- IO against non scalable data source/sink
  - E.g. single non-scalable database will be a bottleneck
  - Great for scalable IO against e.g. s3 or hdfs
Why not Spark (6): Transactional + application work

- It’s not a database
- Don’t try to use it for OLTP workloads
Why Spark

• Process a whole bunch of data with maximum throughput

• Use the many libraries built in an external

• Get started fast on programming (no architecting)

• Get started fast analyzing (interactive mode)
How Spark Works

http://spark.apache.org/docs/latest/cluster-overview.html

How cluster execution works
Let’s make a spark application

• Get Zeppelin: https://zeppelin.apache.org/download.html

• If you’re on the hotel wifi, it’s likely fastest to get the source distribution and compile

• Zeppelin quick start: https://zeppelin.apache.org/docs/0.7.1/install/install.html

• Get the mushroom dataset: https://www.kaggle.com/uciml/mushroom-classification/downloads/mushroom-classification.zip

• We’re going to work together to do some machine learning

• Pre-made Zeppelin notebook: https://goo.gl/RvTm4V
Let’s make a spark application

- Last year’s exercise is also available: http://conferences.infotoday.com/documents/245/W2_Tustin.pdf