Apache Spark Quick Guide
Python

Joshua Jansen Van Vueren

December 6, 2019

Introduction

This document seeks to outline and describe the programming model utilised by Apache Spark, and compare utility to other frameworks.

"Apache Spark is a fast and general-purpose cluster computing system."
"Apache Spark lets you use clusters of tens, hundreds, or thousands of servers to run simulations in a way that is intuitive and scales to meet your needs."
"Overall, Spark has been a flexible, robust and scalable data processing engine... Spark Streaming provides efficient fault-tolerant stateful stream processing. It provides high throughput and good integration with data stores like Cassandra, Kafka, Hadoop etc."

1 Terminology

- **Resilient Distributed Datasets - RDD:** Fault-tolerant collection of elements that can be operated on in parallel. Optimized for parallel processing.
- **Dataset:** Distributed collection of data, newer programming interface, better performance over RDD's.
- **DataFrame:** Dataset organized into named columns, conceptually equivalent to a relational DB table (supported in Scala,Java,Python,R).
- **DStreams:** Sequence of RDD's as defined by user based of time intervals (micro-batching).
- **Parallelization:** Members from collection are copied to form a distributed dataset, to be operated on in parallel. Slicing the data into a number of partitions to be used in the cluster.

- **External Datasets:** Spark can create distributed datasets from any storage source supported by Hadoop.
- **Cluster:** Groups of Compute Engine VM's with master and worker VM's.
- **Cloud Dataproc:** Google Cloud Dataproc lets you provision Apache Hadoop clusters and connect to underlying analytic data stores. "Can provision capacity on demand and pay for it by the minute". Spark jobs can be submitted and run on Dataproc.

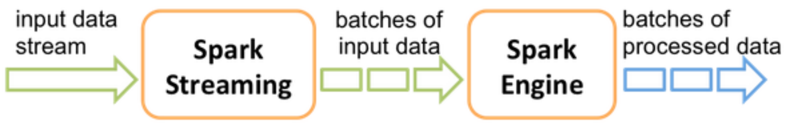
2 Systems

2.1 Beam

- Beam provides the ability to utilise multiple different runners, these runners are compared through a capability matrix - available here. From the matrix it is clear that all areas considered in data processing the number functionalities provided by Dataflow outweigh Spark.
- So one can utilise Spark to run Apache Beam pipelines, but Spark can also be run independently.

2.2 Spark

- Memory Intensive
- Streaming builds micro-batches (micro-batch size is a pre-set, fixed, time interval), processed by Spark engine, outputting processed data. Not per record stream.
- Data is structured in RDD's which is analogous to Beam's PCollection.
- Utility being a all-in-one solution for all processing needs.
- Can be run locally using servers or online utilising BigQuery, DataProc etc.
- Streaming Inputs: Kafka, Flume, Kinesis, or TCP sockets
- MLib can be applied to data-streams. There are streaming ML algorithms which learn from streaming data and are applied to streaming data, and there are models which can be learned off historical data (offline) and then applied to streaming data.
- Streaming Outputs: Filesystems, databases, and live dashboards



2.2.1 Spark Use Cases

- Streaming Data
  - Streaming Extract, Transform, Load
  - Data Enrichment
  - Trigger Event Detection
  - Complex Session Analysis - session activity easy to group and analyse
- ML
- Fog Computing

Industry Use Cases:

- Booking.com utilises spark for online ML features for real-time prediction of behaviour and preference of users.
- Walmart Labs Pipeline using Spark Streaming

2.2.2 Utilities:

- Able to read from and write to BigQuery, code here.
- Can run ML algorithms from BigQuery data, code here.
- Processes data from Pub/Sub using Dataproc and outputs to Datastore, code here
- PySpark does have some streaming capabilities, but is limited in connectors, connectors like Kafka are supported; Py Spark Streaming Library. I have yet to see a streaming example of PySpark that reads from and writes to any cloud storage, they all seem to implemented in a batch mode.

2.2.3 Libs:

- **MLib** `https://spark.apache.org/docs/2.0.0/ml-guide.html`

2.2.4 RDD Operations

- **Transformations:** Create new dataset from an existing one
- **Actions:** Return a value to the driver program after running a computation on the dataset.
- **Note:** All transformations are lazy, therefore they do not compute all results immediately, only when required - by an action. One would need to run a `collect()` or some other action.
- **Models:** ML models can be run on data through the spark structure, example here

2.3 Kafka

- Data Pipeline
- Record-at-a-time processing

2.4 Hive

- Horizontally scalable
- SQL Interface, operating on Hadoop
- Built for data warehousing operations

2.5 Comparisons

Spark vs Beam: Apache Beam can be classified as a tool in the "Workflow Manager" category, while Apache Spark is grouped under "Big Data Tools" [1].

Spark vs Hadoop: Hadoop is more focused towards batching, and spark towards streaming. Spark utilises HDFS for its file structuring. Spark was built over the Hadoop framework to achieve faster data processing in both batch and streaming applications, at the cost of being more memory intensive.

3 Other Documentation and Links

- Google Cloud Storage Connector with Apache Spark
- Apache Spark Use Cases

[1] https://stackshare.io/stackups/apache-beam-vs-spark

- Google Cloud Console Authentication Setup
- Caching in Spark
- Filtering Vs Enriching Data in Apache Spark; interesting use case.

4 Where to Go From Here

- I would shift over to implementing Spark in Scala. It seems required if one would like to access the full capabilities of the system. There are numerous examples of streaming pipeline examples written in Scala.
- It is my intuition that there may be a way around the constraint of only batch implementations in PySpark. It would seem that WorkFlows in DataProc would have the ability to schedule jobs, and the BigQuery connector provided one could implement a windowing query. Alternatively a cron job could do the trick.
- The new graph library in Spark could be of some utility, available here.