Apache Spark

Technology Paper for I524

Saurabh Swaroop
HID-SP18-522
March 20, 2018

Contents

1 Introduction

The amount of data consumed and generated by enterprise are huge in current world. Such large volume of data has created a necessity for the companies to make the best use of the diversity and hidden information associated with them.Not only these informations improvise the productivity but enables enterprises to identify new opportunities.With the advent of big data, more enterprises are incorporating data analytics when building strategic applications and making decisions. Data processing and its effective analysis plays a very important role in functioning and getting ahead of the competition for almost every enterprise now a days.

Enterprise need to have a standard process and infrastructure to perform data analysis which involves several stages like Data Collection, Data Cleaning, Data Storage, Data Processing, Data Analysis. On top of these stages lies the models on which enterprise map their business to get ahead of their peers.

Apache Spark is cluster computing framework which is used for data processing and analysis over large scale with great processing rate.

2 Spark Overview

Spark is an Apache open source project originally developed in AMPLab at UC Berkeley. It is a cluster computing framework which is built on top of Hadoop distributed file system HDFS for data analysis and processing. Unlike Hadoop, Spark does not use Map-reduce engine to read and write to hard disk. It is based on, in memory computation process which increases its speed over 100X as compared to Hadoop.

In memory cluster computing framework allows applications to load data into a cluster?s memory. This process enables multiple recursive queries without performance overhead. It allows both batch and real-time data analytic and processing pay loads.

Spark is available for Linux, OSX and Windows. It integrates well with several languages like Java, Scala and Python. It provides an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

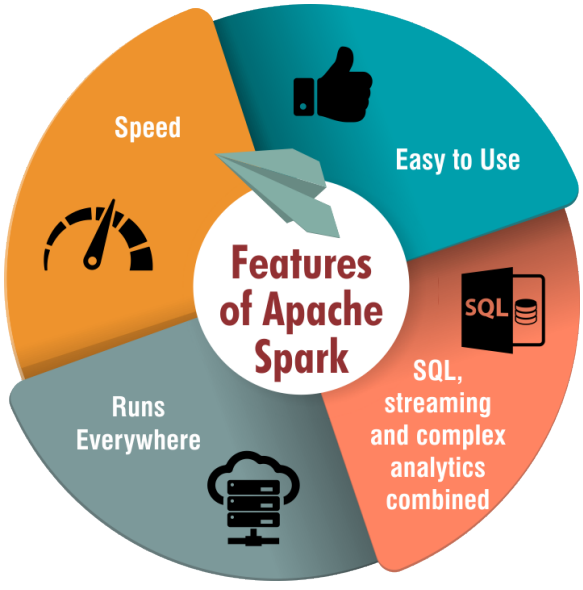

Fig. 1: Real, Spak Features [1]

3 Spark Engine

Spark Core is the base engine for large-scale parallel and distributed data processing. It is responsible for:

- memory management and fault recovery
- scheduling, distributing and monitoring jobs on a cluster
- interacting with storage systems

Spark is built around RDD (Resilient Distributed Dataset), an immutable fault-tolerant, distributed collection of objects that can be operated on in parallel. An RDD can contain any type of object and is created by loading an external dataset or distributing a collection from the driver program. RDDs support two types of operations:[2]

- Transformations are operations (such as map, filter, join, union, and so on) that are performed on an RDD and which yield a new RDD containing the result.
- Actions are operations (such as reduce, count, first, and so on) that return a value after running a computation on an RDD.

Transformations in Spark are lazy, meaning that they do not compute their results right away. Instead, they just remember the operation to be performed and the dataset (e.g., file) to which the operation is to be performed. The transformations are only actually computed when an action is called, and the result is returned to the driver program. This design enables Spark to run more efficiently. For example, if a big file was transformed in various ways and passed to first action, Spark would only process and return the result for the first line, rather than do the work for the entire file.[2]

By default, each transformed RDD may be recomputed each time you run an action on it. However, you may also persist an RDD in memory using the persist or cache method, in which case Spark will keep the elements around on the cluster for much faster access the next time you query it.[2]

The Spark core comes bundled with some powerful libraries/components which could be seamlessly used in the same application. These libraries currently include SparkSQL, Spark Streaming, MLlib (for machine learning), and GraphX, each of which is further detailed in this article. Additional Spark libraries and extensions are currently under development as well.[2]

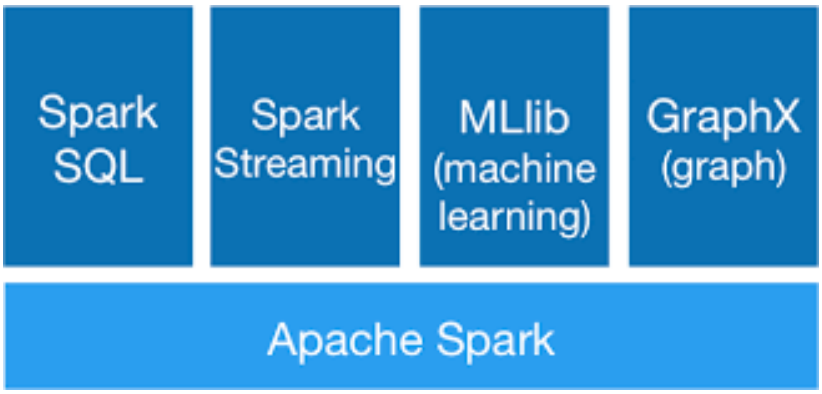

Fig. 2: Spark Architecture

3.1 SparkSQL

SparkSQL is a Spark component that supports querying data either via SQL or via the Hive Query Language. It originated as the Apache Hive port to run on top of Spark (in place of MapReduce) and is now integrated with the Spark stack. In addition to providing support for various data sources, it makes it possible to weave SQL queries with code transformations which results in a very powerful tool. Below is an example of a Hive compatible query:[2]

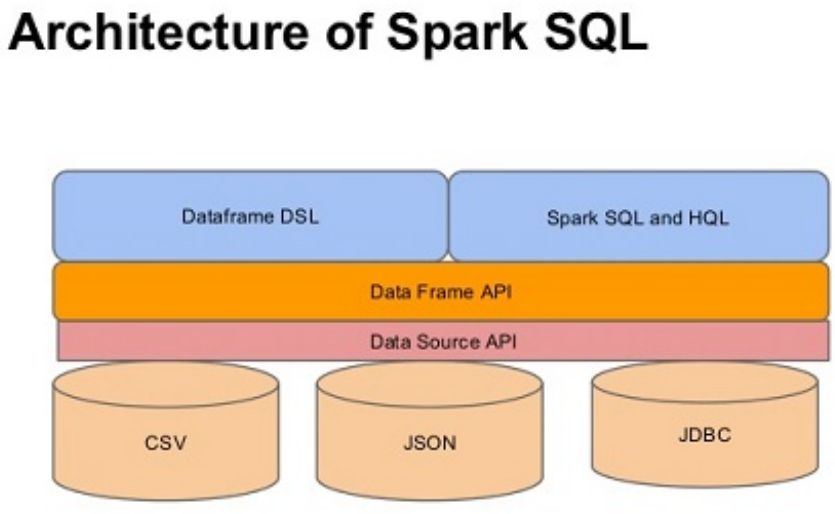

Fig. 3: Spark Sql Architecture[3]

3.2 Spark Streaming

Spark Streaming supports real time processing of streaming data, such as production web server log files (e.g. Apache Flume and HDFS/S3), social media like Twitter, and various messaging queues like Kafka. Under the hood, Spark Streaming receives the input data streams and divides the data into batches.[2]

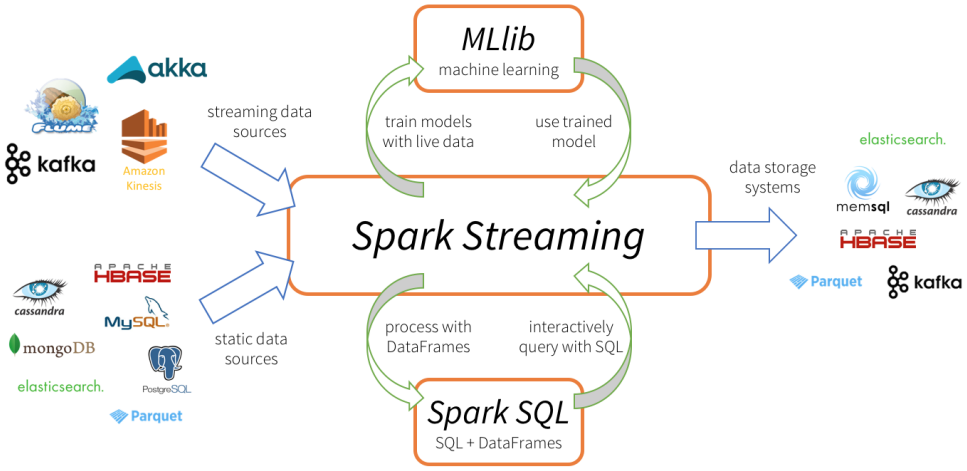

Fig. 4: Spark Streaming [4]

3.3 MLlib

MLlib is Spark's open-source distributed machine learning library. MLlib provides efficient functionality for a wide range of learning settings and includes several underlying statistical, optimization, and linear algebra primitives. Shipped with Spark, MLlib supports several languages and provides a high-level API that leverages Spark's rich ecosystem to simplify the development of end-to-end machine learning pipelines. MLlib has experienced a rapid growth due to its vibrant open-source community of over 140 contributors, and includes extensive documentation to support further growth and to let users quickly get up to speed[5]

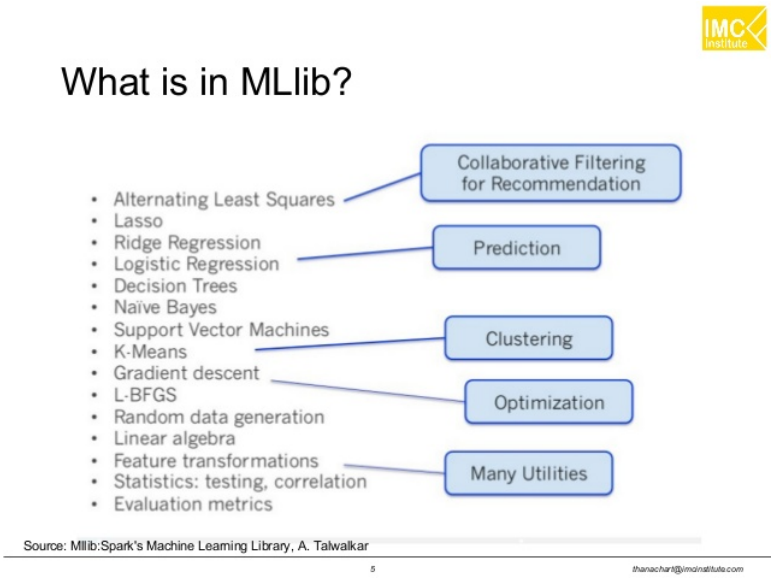

Fig. 5: Mlib [4]

3.4 GraphX

GraphX is a new component in Spark for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing a new Graph abstraction: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, join Vertices, and aggregate Messages) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.[6]

4 Other Apache Spark Use Cases

4.1 Gaming Industry

In the game industry, processing and discovering patterns from the potential firehose of real-time in-game events and being able to respond to them immediately is a capability that could yield a lucrative business, for purposes such as player retention, targeted advertising, auto-adjustment of complexity level, and so on.[7]

4.2 Finance Industry

In the finance or security industry, the Spark stack could be applied to a fraud or intrusion detection system or risk-based authentication. It could achieve top-notch results by harvesting huge amounts of archived logs, combining it with external data sources like information about data breaches and compromised accounts and information from the connection/request such as IP geolocation or time.[7]

4.3 IoT

IoT is rapidly emerging as a leading area for Apache Spark applications. In the real-world data analysis pipelines, where real-time streams are collected from edge devices, gateways, or other clouds, and then processed by Spark Streaming applications, which in turn generate derived streams for further processing, data aggregates, or trigger other real-time events. Spark's capability of using similar code for both stream and batch processing can simplify a number of data management issues. Modern IoT operations can drive digital transformation by analyzing the unprecedented amounts of data generated from devices and sensors in real-time.[8]

Apache Spark is a widely used stream processing engine for real-time IoT applications. Spark streaming offers a rich set of APIs in the areas of ingestion, cloud integration, multi-source joins, blending streams with static data, time-window aggregations, transformations, data cleansing, and strong support for machine learning and predictive analytics.[7]

References

[1] [Online]. Available: https://www.collaberatact.com/apache-spark-hadoop-locking-horns/

[2] [Online]. Available: https://www.toptal.com/spark/introduction-to-apache-spark

[3] [Online]. Available: https://www.dezyre.com/article/spark-sql-for-relational-big-data-processing/355

[4] [Online]. Available: https://databricks.com/product/getting-started-guide/streaming

[5] [Online]. Available: https://spark.apache.org

[6] [Online]. Available: https://spark.apache.org/docs/latest/graphx-programming-guide.html

[7] [Online]. Available: https://www.deepcoredata.com/introduction-apache-spark-examples/

[8] [Online]. Available: https://databricks.com/session/spark-streaming-and-iot