What is Apache Spark?

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size. It provides development APIs in Java, Scala, Python and R, and supports code reuse across multiple workloads—batch processing, interactive queries, real-time analytics, machine learning, and graph processing. You'll find it used by organizations from any industry, including at FINRA, Yelp, Zillow, DataXu, Urban Institute, and CrowdStrike.

What is the history of Apache Spark?

Apache Spark started in 2009 as a research project at UC Berkley's AMPLab, a collaboration involving students, researchers, and faculty, focused on data-intensive application domains. The goal of Spark was to create a new framework, optimized for fast iterative processing like machine learning, and interactive data analysis, while retaining the scalability, and fault tolerance of Hadoop MapReduce. The first paper entitled, "Spark: Cluster Computing with Working Sets" was published in June 2010, and Spark was open sourced under a BSD license. In June, 2013, Spark entered incubation status at the Apache Software Foundation (ASF), and established as an Apache Top-Level Project in February, 2014. Spark can run standalone, on Apache Mesos, or most frequently on Apache Hadoop.

How does Apache Spark work?

Hadoop MapReduce is a programming model for processing big data sets with a parallel, distributed algorithm. Developers can write massively parallelized operators, without having to worry about work distribution, and fault tolerance. However, a challenge to MapReduce is the sequential multi-step process it takes to run a job. With each step, MapReduce reads data from the cluster, performs operations, and writes the results back to HDFS. Because each step requires a disk read, and write, MapReduce jobs are slower due to the latency of disk I/O.

Spark was created to address the limitations to MapReduce, by doing processing in-memory, reducing the number of steps in a job, and by reusing data across multiple parallel operations. With Spark, only one-step is needed where data is read into memory, operations performed, and the results written back—resulting in a much faster execution. Spark also reuses data by using an in-memory cache to greatly speed up machine learning algorithms that repeatedly call a function on the same dataset. Data re-use is accomplished through the creation of DataFrames, an abstraction over Resilient Distributed Dataset (RDD), which is a collection of objects that is cached in memory, and reused in multiple Spark operations. This dramatically lowers the latency making Spark multiple times faster than MapReduce, especially when doing machine learning, and interactive analytics.

Key differences: Apache Spark vs. Apache Hadoop

Outside of the differences in the design of Spark and Hadoop MapReduce, many organizations have found these big data frameworks to be complimentary, using them together to solve a broader business challenge.

Hadoop is an open source framework that has the Hadoop Distributed File System (HDFS) as storage, YARN as a way of managing computing resources used by different applications, and an implementation of the MapReduce programming model as an execution engine. In a typical Hadoop implementation, different execution engines are also deployed such as Spark, Tez, and Presto.

Spark is an open source framework focused on interactive query, machine learning, and real-time workloads. It does not have its own storage system, but runs analytics on other storage systems like HDFS, or other popular stores like Amazon Redshift, Amazon S3, Couchbase, Cassandra, and others. Spark on Hadoop leverages YARN to share a common cluster and dataset as other Hadoop engines, ensuring consistent levels of service, and response.

What are the benefits of Apache Spark?

There are many benefits of Apache Spark to make it one of the most active projects in the Hadoop ecosystem. These include:

Fast

Through in-memory caching, and optimized query execution, Spark can run fast analytic queries against data of any size.

Developer friendly

Apache Spark natively supports Java, Scala, R, and Python, giving you a variety of languages for building your applications. These APIs make it easy for your developers, because they hide the complexity of distributed processing behind simple, high-level operators that dramatically lowers the amount of code required.

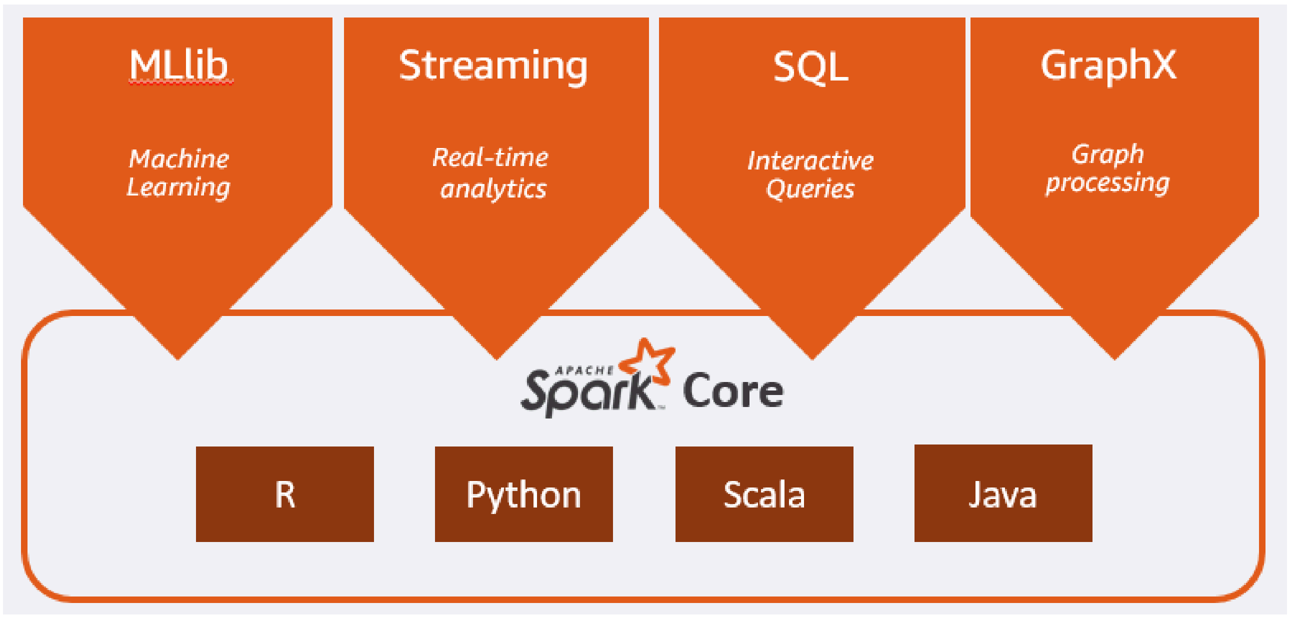
Multiple workloads

Apache Spark comes with the ability to run multiple workloads, including interactive queries, real-time analytics, machine learning, and graph processing. One application can combine multiple workloads seamlessly.

What are Apache Spark Workloads?

The Spark framework includes:

- Spark Core as the foundation for the platform
- Spark SQL for interactive queries
- Spark Streaming for real-time analytics
- Spark MLlib for machine learning
- Spark GraphX for graph processing



Spark Core

Spark Core is the foundation of the platform. It is responsible for memory management, fault recovery, scheduling, distributing & monitoring jobs, and interacting with storage systems. Spark Core is exposed through an application programming interface (APIs) built for Java, Scala, Python and R. These APIs hide the complexity of distributed processing behind simple, high-level operators.

MLlib

Machine Learning

Spark includes MLlib, a library of algorithms to do machine learning on data at scale. Machine Learning models can be trained by data scientists with R or Python on any Hadoop data source, saved using MLlib, and imported into a Java or Scala-based pipeline. Spark was designed for fast, interactive computation that runs in memory, enabling machine learning to run quickly. The algorithms include the ability to do classification, regression, clustering, collaborative filtering, and pattern mining.

Spark Streaming

Real-time

Spark Streaming is a real-time solution that leverages Spark Core's fast scheduling capability to do streaming analytics. It ingests data in mini-batches, and enables analytics on that data with the same application code written for batch analytics. This improves developer productivity, because they can use the same code for batch processing, and for real-time streaming applications. Spark Streaming supports data from Twitter, Kafka, Flume, HDFS, and ZeroMQ, and many others found from the Spark Packages ecosystem.

Spark SQL

Interactive Queries

Spark SQL is a distributed query engine that provides low-latency, interactive queries up to 100x faster than MapReduce. It includes a cost-based optimizer, columnar storage, and code generation for fast queries, while scaling to thousands of nodes. Business analysts can use standard SQL or the Hive Query Language for querying data. Developers can use APIs, available in Scala, Java, Python, and R. It supports various data sources out-of-the-box including JDBC, ODBC, JSON, HDFS, Hive, ORC, and Parquet. Other popular stores—Amazon Redshift, Amazon S3, Couchbase, Cassandra, MongoDB, Salesforce.com, Elasticsearch, and many others can be found from the Spark Packages ecosystem.

GraphX

Graph Processing

Spark GraphX is a distributed graph processing framework built on top of Spark. GraphX provides ETL, exploratory analysis, and iterative graph computation to enable users to interactively build, and transform a graph data structure at scale. It comes with a highly flexible API, and a selection of distributed Graph algorithms.

What are the use cases of Apache Spark?

Spark is a general-purpose distributed processing system used for big data workloads. It has been deployed in every type of big data use case to detect patterns, and provide real-time insight. Example use cases include:

Financial Services

Spark is used in banking to predict customer churn, and recommend new financial products. In investment banking, Spark is used to analyze stock prices to predict future trends.

Healthcare

Spark is used to build comprehensive patient care, by making data available to front-line health workers for every patient interaction. Spark can also be used to predict/recommend patient treatment.

Manufacturing

Spark is used to eliminate downtime of internet-connected equipment, by recommending when to do preventive maintenance.

Retail

Spark is used to attract, and keep customers through personalized services and offers.

How deploying Apache Spark in the cloud works?

Spark is an ideal workload in the cloud, because the cloud provides performance, scalability, reliability, availability, and massive economies of scale. ESG research found 43% of respondents considering cloud as their primary deployment for Spark. The top reasons customers perceived the cloud as an advantage for Spark are faster time to deployment, better availability, more frequent feature/functionality updates, more elasticity, more geographic coverage, and costs linked to actual utilization.

What are the AWS offerings for Apache Spark?

Amazon EMR is the best place to deploy Apache Spark in the cloud, because it combines the integration and testing rigor of commercial Hadoop & Spark distributions with the scale, simplicity, and cost effectiveness of the cloud. It allows you to launch Spark clusters in minutes without needing to do node provisioning, cluster setup, Spark configuration, or cluster tuning. EMR enables you to provision one, hundreds, or thousands of compute instances in minutes. You can use Auto Scaling to have EMR automatically scale up your Spark clusters to process data of any size, and back down when your job is complete to avoid paying for unused capacity. You can lower your bill by committing to a set term, and saving up to 75% using Amazon EC2 Reserved Instances, or running your clusters on spare AWS compute capacity and saving up to 90% using EC2 Spot.