Spark: Unified Analytics Engine

GFS{google file system}- HDFS

MapRedue - Map reduce

Bigtable: HBase

MapReduce: functional programming based programming model.{code to data}

**Short Cummings With Hadoop:**

Hard to administer, batch processing {boiler plate: reputative code}, large i\o for intermediate result. Not interactive.

HDFS 
HDFS 
HDFS 
HDFS 
Reads 
Reads 
Writes 
Writes 
Figure 1-1. Intermittent iteration of reads and writes between map and reduce 
computations 

**Spark:** fast, simple and easy.

**Apache Spark** is a unified analytics engine designed for large-scale distributed data processing, on premises in data centers or in the cloud.

Spark built its computation of transformation and action as DAG.{direct acyclic graph}

RDD: System level abstraction- higher level abstraction is Dataset and data frame.

* Easy programming model of transformation and action.
* Modularity: unified API based on workload in any language.
* Extensibility: parallel computation from any data source . DataFrame Reader and DataFrame Writer can be extended to read and write from any source. Eg: S3, Hive, RDBMS, Cassandra et.

**Spark Components:**

Spark SQL and 
Data Frames + 
Datasets 
Scala 
Spark Streaming 
Machine Learning 
(Structured 
MLlib 
Streaming) 
Spark Core and Spark SQL Engine 
Graph 
Processing 
Graph X 
SQL 
Python 
Java 
Figure 1-3. Apache Spark components and API stack 

**Spark SQL:** for structured Data ANSI standard, functions as pure SQL engine. We can create permanent or temporary tables and perform direct SQL to generate DataFrames.

**Spark Streaming:**  Spark treats stream as continuous growing table- new rows are getting appended to the existing table.

We can query them as normal Static table.

# Spark Architecture:

Spark Application 
Spark Driver 
SparkSession 
Cluster Manager 
O 
Core 
oo 
oo 
Spark Executor 
oo 
oo 
Spark Executor 
Figure 1-4. Apache Spark components and architecture 

Spark Driver orchestrate the parallel operation on cluster using SparkSession.

**Spark Driver:**

* Communicates with cluster manager before resource allocation and after directly executer .
* Request resources
* Transforms operations into DAG computation.
* Schedules them
* Distributes them.

**SparkSession:**

* Single entry point for all spark functionality. i.e no need to have Spark different context object. Eg: SQL context.
* Using SparkSession we can read data, create context, create DataFrame and DataSet and query.
* Context Objects is available for backward compatibility we can directly read file and query it.

Which config parameter can be provided in Spark Session for large files?

Ans: number of Partitions

import org.apache.spark.sql.SparkSession

// Build SparkSession

val spark = SparkSession

.builder

.appName("LearnSpark")

.config("spark.sql.shuffle.partitions", 6)

.getOrCreate()

...

// Use the session to read JSON

val people = spark.read.json("...")

...

// Use the session to issue a SQL query

val resultsDF = spark.sql("SELECT city, pop, state, zip FROM table\_name")

**Cluster Manager:**

* responsible for managing and allocating resources.
* Four cluster managers are supported

1. Standalone
2. Yarn
3. Mesos
4. Kubernetes

**Spark Executor:**

* Responsible for executing task on worker nodes
* Communicates with driver

**Q: What are spark deployment modes.**

**Ans:**Local, standalone, YARN, Kubernetes.

**Partitions:**

* spark treats partitions as high level logical data abstraction.{DataFrame}
* Partition is distributed scheme of breaking up data into chunks

on 
on 
on 
O 
Spark Executor 
Data Partitions 
on 
on 
on 
O 
Spark Executor 
Data Partitions 
Data Partitions 
on 
on 
on 
O 
Spark Executor 
Data Partitions 
O Core [1 Partition 
Figure 1-6. Each executor's core gets a partition of data to work on 