Marathwada Shikshan Prasarak Mandal's
Deogiri Institute of Engineering and Management Studies, Aurangabad

Seminar Report

On

Apache Spark

Submitted By

Nikita Dnyaneshwar Tambe (36075)

Dr. Babasaheb Ambedkar Technological University
Lonere (M.S.)

Department of Computer Science and Engineering
Deogiri Institute of Engineering and Management Studies, Aurangabad
(2019- 2020)

Seminar Report

On

Apache Spark

Submitted By

Nikita Dnyaneshwar Tambe (36075)

In partial fulfillment of

Bachelor of Technology

(Computer Science & Engineering)

Guided By

Mrs .Amruta Joshi

Department of Computer Science & Engineering

Deogiri Institute of Engineering and Management Studies,

Aurangabad

(2019- 2020)

CERTIFICATE

This is to certify that, the Seminar entitled "**Apache Spark** " submitted by **Nikita Dnyaneshwar Tambe** is a bonafide work completed under my supervision and guidance in partial fulfillment for award of Bachelor of Technology (Computer Science and Engineering) Degree of Dr. Babasaheb Ambedkar Technological University, Lonere.

Place: Aurangabad
Date:

Mrs Amruta Joshi
Guide

Mr. S.B. Kalyankar
Head

Dr. Ulhas D. Shiurkar
Director,
Deogiri Institute of Engineering and Management Studies,
Aurangabad

Abstract

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate toolsSpark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMPLby Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

Term basic abstraction in Apache Spark arises, the only name strikes in mind is.. RDD.., RDD stands for "Resilient Distributed Dataset". It is the fundamental abstraction in Apache Spark. It is the basic data structure. RDD in Apache Spark is an immutable collection of objects which computes on the different node of the cluster.

Contents

List of Graphs

1. INTRODUCTION

1.1 Introduction

Industries are using Hadoop extensively to analyze their data sets. The reason is that Hadoop framework is based on a simple programming model (MapReduce) and it enables a computing solution that is scalable, flexible, fault-tolerant and cost effective. Here, the main concern is to maintain speed in processing large datasets in terms of waiting time between queries and waiting time to run the program.

Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process.

As against a common belief, Spark is not a modified version of Hadoop and is not, really, dependent on Hadoop because it has its own cluster management. Hadoop is just one of the ways to implement Spark.

Spark uses Hadoop in two ways – one is storage and second is processing. Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only.

Apache Spark is an exceptionally a cluster computing technology, intended for quick computation. It depends on Hadoop MapReduce and it stretches out the MapReduce model to effectively utilize it for more kinds of computations, which incorporates intuitive questions and stream handling. The fundamental element of Spark is its in-memory clustering that expands the preparing pace of an application.

Apache Spark is a lightning fast cluster computing system. It provides the set of high-level API namely Java, Scala, Python, and R for application development. Apache Spark is a tool for speedily executing Spark Applications. Spark utilizes Hadoop in two different ways – one is for Storage and second is for Processhandling.

Just because Spark has its own Cluster Management, so it utilizes Hadoop for Storage objective.

Spark is intended to cover an extensive variety of remaining loads, for example, cluster applications, iterative calculations, intuitive questions, and streaming. Aside from supporting all these remaining tasks at hand in a particular framework, it decreases the administration weight of keeping up isolated apparatuses.

Spark is one of Hadoop's sub venture created in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was given to Apache programming establishment in 2013, and now Apache Spark has turned into the best level Apache venture from Feb-2014. And now the results are pretty booming.

With the development of science, a number of applications which are based on iterative algorithms appear. Hadoop MapReduce is based on an acyclic data flow model. With the output of the previous MapReduce job as the input of the next MapReduce job, the iterative programs can be accomplished. In such design, the data used in each iteration is reread and reprocessed, wasting a lot of time in operation.Spark is an open source project developed by UC Berkeley AMPLab. With the realization of RDDs, a distributed memory abstraction that lets programmers perform inmemory computations on large clusters, Spark provides RDD transformations and actions for the users to use Spark easily. Spark is implemented in Scala and described as a powerful object oriented language with ample resources. The first company to provide support for Apache storm recommends Apache Spark as Data Science tool. Spark Reliability can be judged from Intel recommendation for spark to be used in healthcare solutions.

Only following two scenarios, can hinder the suitability of Apache spark are Low Tolerance to Latency requirements and shortage of Memory resources.

Dataset is a data structure in SparkSQL which is strongly typed and is a map to a relational schema. It represents structured queries with encoders. It is an extension to dataframe API. Spark Dataset provides both type safety and object-oriented programming interface. We encounter the release of the dataset in Spark 1.6. The encoder is primary concept in serialization and deserialization (SerDe) framework in Spark SQL. Encoders translate between JVM objects and Spark's internal binary format. Spark has built-in encoders which are very advanced. They generate bytecode to interact with off-heap data.

An encoder provides on-demand access to individual attributes without having to de-serialize an entire object. To make input output time and space efficient, Spark SQL uses SerDe framework. Since encoder knows the schema of record, it can achieve serialization and deserialization.

Spark Dataset is structured and lazy query expression that triggers on the action. Internally dataset represents logical plan. The logical plan tells the computational query that we need to produce the data. the logical plan is a base catalyst query plan for the logical operator to form a logical query plan. When we analyze this and resolve we can form a physical query plan.

RDD stands for "Resilient Distributed Dataset". It is the fundamental data structure of Apache Spark. RDD in Apache Spark is an immutable collection of objects which computes on the different node of the cluster.

Decomposing the name RDD:

- **Resilient**, i.e. fault-tolerant with the help of RDD lineage graph(DAG) and so able to recompute missing or damaged partitions due to node failures.
- **Distributed,** since Data resides on multiple nodes.
- **Dataset** represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or database via JDBC with no specific data structure.

Hence, each and every dataset in RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster. RDDs are fault tolerant i.e. It posses self-recovery in the case of failure.

There are three ways to create RDDs in Spark such as – Data in stable storage, other RDDs, and parallelizing already existing collection in driver program. One can also operate Spark RDDs in parallel with a low-level API that offers transformations and actions. We will study these Spark RDD Operations later in this section.

Spark RDD can also be cached and manually partitioned. Caching is beneficial when we use RDD several times. And manual partitioning is important to correctly balance partitions. Generally, smaller partitions allow distributing RDD data more equally, among more executors. Hence, fewer partitions make the work easy.

Programmers can also call a persist method to indicate which RDDs they want to reuse in future operations. Spark keeps persistent RDDs in memory by default, but it can spill them to disk if there is not enough RAM.

2. LITERATURE SURVEY

2.1 Literature Survey

Spark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

The main abstraction in Spark is that of a resilient distributed dataset (RDD), which represents a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. Users can explicitly cache an RDD in memory across machines and reuse it in multiple MapReduce-like parallel operations. RDDs achieve fault tolerance through a notion of lineage: if a partition of an RDD is lost, the RDD has enough information about how it was derived from other RDDs to be able to rebuild just that partition. Although RDDs are not a general shared memory abstraction, they represent a sweet-spot between expressivity on the one hand and scalability and reliability on the other hand, and we have found them well-suited for a variety of applications

Spark is implemented in Scala , a statically typed high-level programming language for the Java VM, and exposes a functional programming interface similar to DryadLINQ. In addition, Spark can be used interactively from a modified version of the Scala interpreter, which allows the user to define RDDs, functions, variables and classes and use them in parallel operations on a cluster. We believe that Spark is the first system to allow an efficient, general-purpose programming language to be used interactively to process large datasets on a cluster.

Although our implementation of Spark is still a prototype, early experience with the system is encouraging. We show that Spark can outperform Hadoop by 10x in iterative machine learning workloads and can be used interactively to scan a 39 GB dataset with sub-second latency.

2.2 Features of Apache Spark

Apache Spark has following features.

- **Speed** − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- **Supports multiple languages** − Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
- **Advanced Analytics** − Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

2.3 Apache Spark Core

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

2.3.1 Spark SQL0

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

2.3.2 Spark Streaming

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

Spark became a top-level project of the Apache Software Foundation in February 2014, and version 1.0 of Apache Spark was released in May 2014. Spark version 2.0 was released in July 2016.

The technology was initially designed in 2009 by researchers at the University of California, Berkeley as a way to speed up processing jobs in Hadoop systems.

2.4 Prerequisites

.Basic knowledge of python

· scala

· sql

· Linux

2.5 How Apache Spark works:

Apache Spark can process data from a variety of data repositories, including the Hadoop Distributed File System (HDFS), NoSQL databases and relational data stores, such as Apache Hive. Spark supports in-memory processing to boost the performance of big data analytics applications, but it can also perform conventional disk-based processing when data sets are too large to fit into the available system memory.

The Spark Core engine uses the resilient distributed data set, or RDD, as its basic data type. The RDD is designed in such a way so as to hide much of the computational complexity from users. It aggregates data and partitions it across a server cluster, where it can then be computed and either moved to a different data store or run through an analytic model. The user doesn't have to define where specific files are sent or what computational resources are used to store or retrieve files.

In addition, Spark can handle more than the batch processing applications that MapReduce is limited to running.

2.6 Uses of Spark:

· Data processing applications

· Batch applications

· SQL

· Machine Learning

· Streaming data processing

· Graph data processing

3. BRIEF ON SYSTEM

3.1 Working of apache Spark [1]:

Apache Spark is an open source, general-purpose distributed computing engine used for processing and analyzing a large amount of data. Just like Hadoop MapReduce, it also works with the system to distribute data across the cluster and process the data in parallel. Spark uses master/slave architecture i.e. one central coordinator and many distributed workers. Here, the central coordinator is called the driver.

The driver runs in its own Java process. These drivers communicate with a potentially large number of distributed workers called executors. Each executor is a separate java process. A Spark Application is a combination of driver and its own executors. With the help of cluster manager, a Spark Application is launched on a set of machines. Standalone Cluster Manager is the default built in cluster manager of Spark. Apart from its built-in cluster manager, Spark also works with some open source cluster manager like Hadoop Yarn, Apache Mesos etc.

3.1.1 Components of Spark Run-time Architecture

1. **Apache Spark Driver**

The main() method of the program runs in the driver. The driver is the process that runs the user code that creates RDDs, and performs transformation and action, and also creates SparkContext. When the Spark Shell is launched, this signifies that we have created a driver program. On the termination of the driver, the application is finished.

The driver program splits the Spark application into the task and schedules them to run on the executor. The task scheduler resides in the driver and distributes task among workers.

The two main key roles of drivers are:

- Converting user program into the task.
- Scheduling task on the executor.

The structure of Spark program at a higher level is: RDDs consist of some input data, derive new RDD from existing using various transformations, and then after it performs an action to compute data. In Spark Program, the DAG (directed acyclic graph) of operations create implicitly. And when the driver runs, it converts that Spark DAG into a physical execution plan.

2. Apache Spark Cluster Manager

Spark relies on cluster manager to launch executors and in some cases, even the drivers launch through it. It is a pluggable component in Spark. On the cluster manager, jobs and action within a spark application scheduled by Spark Scheduler in a FIFO fashion. Alternatively, the scheduling can also be done in Round Robin fashion. The resources used by a Spark application can dynamically adjust based on the workload. Thus, the application can free unused resources and request them again when there is a demand. This is available on all coarse-grained cluster managers, i.e. standalone mode, YARN mode, and Mesos coarse-grained mode. Learn: Spark RDD – Introduction, Features & Operations of RDD.

Apache Spark Executors

The individual task in the given Spark job runs in the Spark executors. Executors launch once in the beginning of Spark Application and then they run for the entire lifetime of an application. Even if the Spark executor fails, the Spark application can continue with ease.

There are two main roles of the executors:

- Runs the task that makes up the application and returns the result to the driver.
- Provide in-memory storage for RDDs that are cached by the user.

3.1.2 How to launch a Program in Spark?

Despite using any cluster manager, Spark comes with the facility of a single script that can use to submit a program, called as spark-submit**.** It launches the application on the cluster. There are various options through which spark-submit can connect to different cluster manager and control how many resources our application gets. For some cluster managers, spark-submit can run the driver within the cluster (e.g., on a YARN worker node), while for others, it can run only on your local machine.

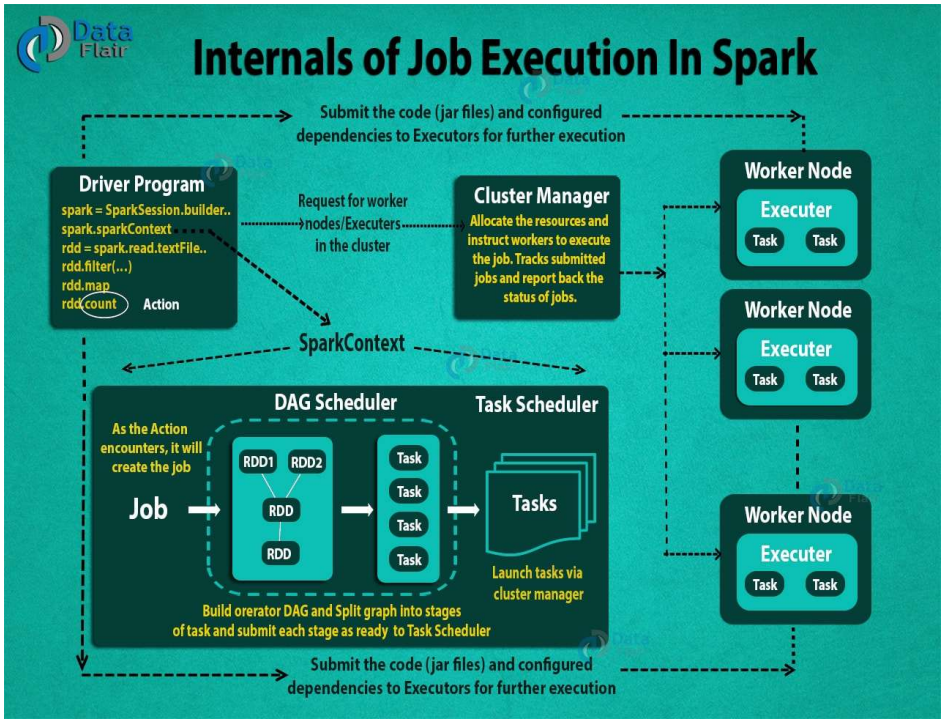

Fig : 3.1

3.2 Spark Architecture [2]

Apache Spark has a well-defined layered architecture where all the spark components and layers are loosely coupled. This architecture is further integrated with various extensions and libraries. Apache Spark Architecture is based on two main abstractions:

- Resilient Distributed Dataset (RDD)
- Directed Acyclic Graph (DAG)

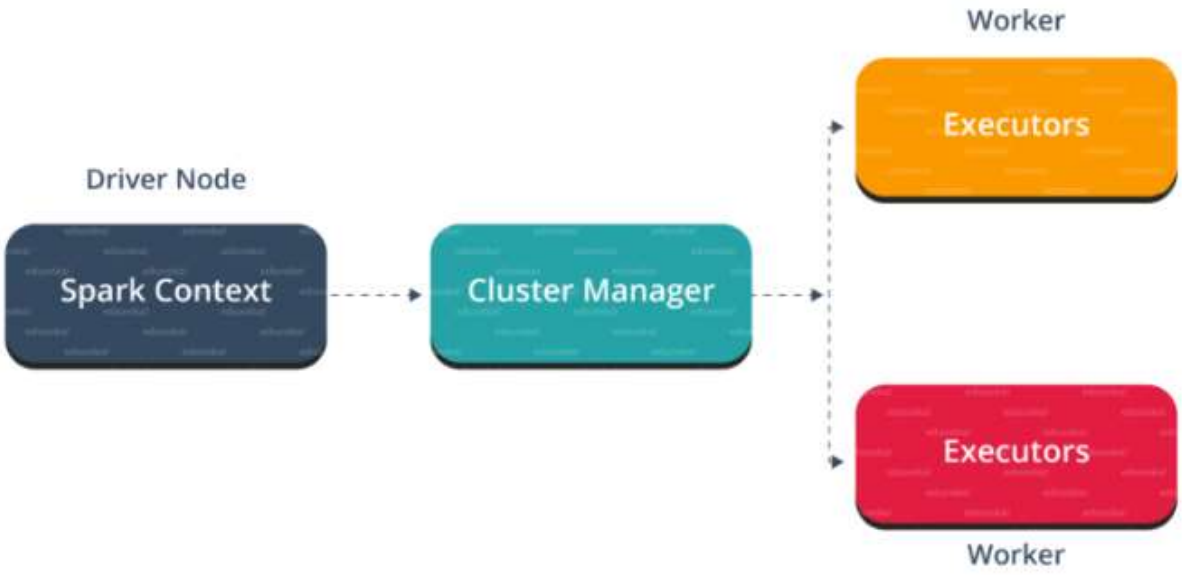

Fig : 3.2

4.Conclusion

4.1 Conclusion

Spark SQL, a new module in Apache Spark providing rich integration with relational processing. Spark SQL extends Spark with a declarative DataFrame API to allow relational processing, offering benefits such as automatic optimization, and letting users write complex pipelines that mix relational and complex analytics. It supports a wide range of features tailored to large-scale data analysis, including semi-structured data, query federation, and data types for machine learning. To enable these features, Spark SQL is based on an extensible optimizer called Catalyst that makes it easy to add optimization rules, data sources and data types by embedding into the Scala programming language. User feedback and benchmarks show that Spark SQL makes it significantly simpler and more efficient to write data pipelines that mix relational and procedural processing, while offering substantial speedups over previous SQL-on-Spark engines.

4.2 Applications [4] :

4.2.1 Machine Learning – Apache Spark is equipped with a scalable Machine Learning Library called as MLlib that can perform advanced analytics such as clustering, classification, dimensionality reduction, etc. Some of the prominent analytics jobs like predictive analysis, customer segmentation, sentiment analysis, etc., make Spark an intelligent technology.

4.2.2 Fog computing – With the influx of big data concepts, IoT has acquired a prominent space for the invention of more advanced technologies. Based on the theory of connecting digital devices with the help of small sensors this technology deals with a humongous amount of data emanating from numerous mediums. This requires parallel processing which is certainly not possible on cloud computing. Therefore Fog computing which decentralizes the data and storage uses Spark streaming as a solution to this problem.

4.2.3 Event detection – The feature of Spark streaming allows the organization to keep track of rare and unusual behaviors for protecting the system. Institutions like financial institutions, security organizations, and health organizations use triggers to detect the potential risk.

5 REFERENCES

[1] Working of apache spark

https://data-flair.training/blogs/how-apache-spark-works/

[2] Apache spark Arechitecture

https://www.edureka.co/blog/spark-architecture/

[3] Spark SQL: Relational Data Processing in Spark. Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, Matei Zaharia. *SIGMOD 2015*. June 2015.

[4] Spark applications

https://intellipaat.com/blog/tutorial/spark-tutorial/apache-spark-applications/

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to Prof. S.B.Kalyankar, HOD-Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Aurangabad, for his generous guidance, help and useful suggestions.

I express my sincere gratitude to Prof. Amruta Joshi, Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Aurangabad, for her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

I am extremely thankful to Dr.Ulhas Shiurkar, Director, Deogiri Institute of Engineering and management Studies Aurangabad, for providing me infrastructural facilities to work in, without which this work would not have been possible.

Signature of Student

Nikita Dnyaneshwar Tambe