Spark_Notes_Udemy

February 25, 2019

1 Spark Notes

1) What is Spark? - Apache Spark is an open source parallel processing framework for running large-scale data analytics applications across clustered computers. It can handle both batch and real-time analytics and data processing workloads.

2) Spark Vs Hadoop - Both Hadoop and Spark uses Map Reduce, but the Sparks imlementation of Map Reduce is different from Hadoop. - **How Hadoop handles MapReduce:** Hadoop Map Reduce stores the intermediate outputs from the no.of mappers to disk i.e for example lets say there are 10 mappers then there will be 10 **Physical Write** operations to HDD from these 10 mappers to store the outputs from these mappers then there will be 10 **Physical Read** Operations from Reduce Phase, this makes Hadoop Very slow when compared to Spark, Consider if there is a **Iterative Algorithm** like **Logistic Regression** and lets say there are 100 Iterations and lets say there are 10 mappers then there will be 100*10(Physical Write Operations)* + *100*10(Physical Read Operations) = 2000 Physical I/O Operations, this makes very slow execution. - **How Spark Handles MapReduce:** Spark also uses MapReduce similar to Hadoop, but the key difference here is, Spark doesn't store the Intermediate Outputs to HDD, Instead it **Caches** the Intermediate outputs in **RAM**(This is called **In-Memory Computation** in Spark Website) so there will be no Physical Read/Write Operations from Mappers Instead there will be **Logical Read/Write** Operations, Logica Read/Write is extremely Fast when compared to Physical Read/Write, because of this reason **Spark is Extremely Fast When compared to Hadoop MapReduce**. **Note:** Physical Read/Write -> Reading/Writing From/To HDD, Logical Read/Write -> Reading/Writing From/To RAM

- Spark uses **DAG(Directed Acyclic Graphs) Execution Engine** that supports Acyclic Data Flow i.e DAG engine converts Users code into Series of Tasks and These tasks are DAG in nature i.e execution flow will be one task to other and never have cyclic relation between tasks. through this way Spark has clarity on what exactly is happening as a whole so that there is a scope of optimization, this is achieved using RDD(where RDD enables Spark to keeps track of all operations and transformations on the dataset). Where as in Hadoop it consumes the Jobs given by the user and Executes accordingly and gives the output, It doesn't have Big Picture of all tasks, so there is no scope of optimization.

3) Key Features that Make Spark Faster than Hadoop - In Memory Computing - Spark Execution Engine(DAG) - RDD's - Spark Architecture

Hadoop MapReduce vs Spark How Node Failure is Handled?

Hadoop MapReduce:

All the Intermediate Outputs are stored in Hard disk i.e Out put produced by Node 1 is Stored in HardDisk and Node 6 Reades this output as input from HardDisk, if

Node1 failed, then Hadoop has 2 other Nodes replicated with same Node1 data, so we can get the data from backup nodes.

Spark:

All the Intermediate outputs are stored In-Memory so Node 6 can Directly read output from Node 1, If Node1 Failed then How Spark is going to handle, as Spark doesnt have Replication nodes. Spark Makes use of RDD(Resilient Distributed Datasets) i.e every Spark operation produces a RDD i.e Spark keeps track of all the Operations(like Aggrigations etc..) and Transformations, so that we can get other Node X with Same as that of Node 1 from RDD, Spark Handles Fault Tolerance in this way.

4)What is RDD? - RDD stands for Resilient Distributed Datasets, Resilient means Fault Tolerance - Spark makes use of RDD to keep track of all the Operations and Transformations performed on Data set i.e it Creates RDD for every Operation or Transformation performed on Data Set.

- Below is the image that Explains with an Example

- Example Illustrates a Case where if we want to check all the hdfs related errors from a text file, Then the Text file data is distributed to no.of blocks here its 5 then using filter() function we get Errors from the Log File then from that errors we are getting HDFS erros using filter() function, Lets say Partition 2 of Errors is down/lost due to Node 2 failure, then we can get Partition 2 of Error by simply running filter() function again on Partition 2 of LogFile, This is the Main use of RDD.
- **RDD** is Core and **Heart** of the Spark
- Spark Achieves **Fault Tolerance** using RDD

Misconceptions on RDD's - All RDD's are Stored in Memory: No It's not i.e if we want execute the above problem once again then it will execute as if it was executed for the first time.

5) 2 Types of Functions in Spark - Transformation Functions: Which Transforms/modifies RDD's and they dont trigger Execution, In the Above Diagram textFile(), filter() are Transformation Functions - Action Functions: Which triggers execution, In the above diagram count() is Action Function.