International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Lightning Fast Distributed Machine Learning Framework

Madhu Sudhan H V

Christ University Institute of Management (Department of Finance),
Malnad College of Engineering (Department of Electronics and Communication)

Abstract: ***According to IBM, Big Data can be expressed with 4 V's, namely Volume, Velocity, Variety and Veracity. Lots of companies are incorporating Big Data in their business model to derive insights from the unstructured data. Big Data is analyzed using Statistical Methods and Machine Learning. Limiting factor using traditional technologies is the incompetence to use huge amounts of data to learn or train algorithms within a practical time. This problem can be handled by using in-memory and distributed machine learning techniques with the help of distributed data sets and by allocating learning to various workstations. A distributed machine learning framework is developed with Spark, Hadoop and Python to scale the machine learning algorithm and to reduce the intensive computation.***

Keywords: Big Data, Distributed Machine Learning, Apache Spark, Python

1. Introduction

Data Science is becoming increasingly popular across various sectors. Companies are investing huge amounts of money to their R&D departments for Big Data to stay ahead of the competition. The total amount of information available from various data sources is enormous and this unrestrainable growth of data paves for new tools and architectures. New challenges have emerged with scalability, memory restrictions and efficiency of the learning algorithm. Traditional architecture train the algorithm with its entire training set in memory, which induces problems like out of memory errors and it is a time consuming, intensive computation process. Thus in order to handle this in-memory and intensive computation process, Apache Spark has come up with its distributed framework. By using this framework, machine learning algorithms can take advantage of distributed computing to manage huge volumes of data in memory or to learn over datasets that are inherently distributed across various workstations.

2. Apache Spark

Apache Spark is a distributed in-memory cluster computing system. It provides API's in Java, Scala and Python. It run programs 100 times faster than Hadoop MapReduce in-memory and 10 times faster than on disk drives. It also supports a set of higher-level tools as follows:

- Spark SQL: It allows SQL queries to be executed using Spark
- MLlib: Machine learning library for Spark
- GraphX: Spark API for graphs and graph-parallel computation
- Spark Streaming: Stream processing of live data

2.1 Spark Components

Sparks allows developer to write a driver program that implements high level control flow of their applications and runs in parallel. It has abstractions namely Resilient Distributed Datasets (RDD), Parallel Operations on the datasets and Shared Variables.

2.1.1. Resilient Distributed Datasets (RDD)

It is a distributed collection of read-only objects which can be partitioned across various machines that will be reconstructed if any node fails. Each RDD is represented by a Scala object in Spark. RDD can be constructed in following ways

- From a file system or from Hadoop file system (HDFS)
- By dividing the Scala object collection and sending it to the clusters
- By transforming the existing RDD
- By using cache

2.1.2. Parallel Operations

Spark helps to execute applications in parallel, main operations involved are reduce, collect and 'for each'. These operations mainly help in reducing the intensive computations.

2.1.3. Shared Variables

Functions (closures) when they are invoked by Programmer, shared variables are copied to the worker node. There are two types of shared variables namely Broadcast Variables and Accumulators which mainly help in reduce phase (Map Reduce) for counting and sharing the variables in multiple parallel operations.

2.2. Workflow

Spark Driver sends tasks for worker nodes to run the applications. Spark Worker node launches the executors responsible for executing the job. An application reserves certain amount of hardware resources and it's not freed until the application completes its execution.

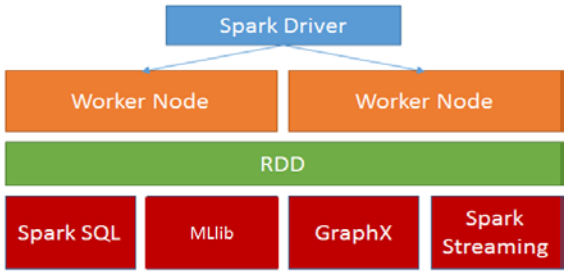

Figure 1: Spark Architecture

3. The Framework

The distributed machine learning framework helps to ingest data, process and finally visualize it in a more efficient and effortless way. This framework can be considered as n-node Hadoop cluster where all the tools and drivers are installed on the master node. This framework depicts all stages of model building lifecycle which involves data integration, data cleaning, processing, analysis, model building and finally building visualizations, dashboards and reports.

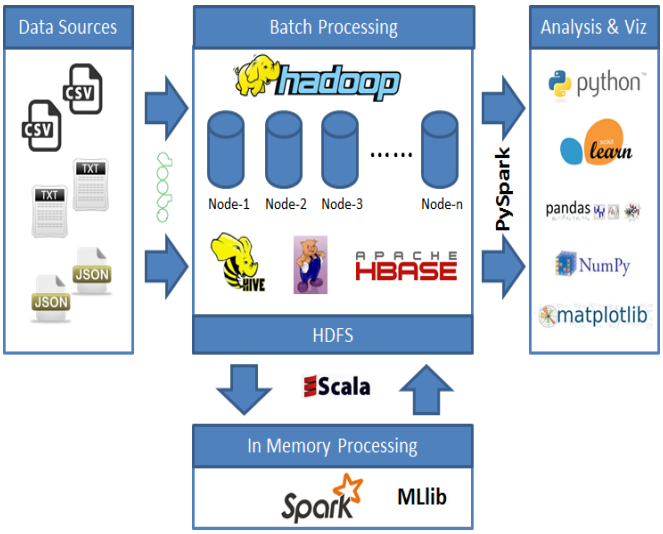

Figure 2: Lightning Fast Distributed Machine Learning Framework

3.1 Data Sources

Big data sources involve different types of data like text, csv files, json, image, audio, video data etc. These data can be imported in to Hadoop ecosystem using various tools. For example as shown in the framework, Sqoop can be used to import data from relational databases like mysql, sql server, oracle etc. Data imported will be stored in Hadoop File System (HDFS) within Hadoop environment.

3.2 Batch Processing

This is a Hadoop ecosystem which mainly involves processing and cleaning the data using various tools. Before building any model, it's necessary to treat data and drop unnecessary features. Data residing in HDFS is accessed using tools like hive and pig to create features and to pre-process the data. Data is stored in database like HBase. Spark has more robust in-memory computing Map Reduce when compared to Hadoop Map Reduce. Spark map-reduce jobs are launched to clean the data and to create required variables. Spark mixed with hive called Shark is another feature of Spark where we can write Hive queries using Spark.

3.3 In Memory Processing

Cleaned data resulting from the above batch processing is used for building the model. Model is trained by using the cleansed data residing on HDFS. Spark has Machine Learning Library (MLlib) for various statistical and algorithmic computations. The in-memory computations are very fast when compared to traditional tools like R, Python, Hadoop Map Reduce etc. Required MLlib is called using Scala API and required parameters are passed to the function for training the model. Resulting output is stored back in HDFS which is further analyzed and visualized.

3.4 Analysis & Viz

PySpark is another feature of Spark where Python can be used as an API for writing in Spark. Python and its required libraries are installed on the master node. Descriptive Statistics are performed on the data residing on HDFS before the model development using Python libraries like Pandas, Scikit Learn, Matplotlib. The model output is further plotted, analysed and reported using Python libraries. Python helps to visualize the data during initial and final phase of the model development.

4. Results

For Experimentation purpose, below configurations are used:

- Framework is implemented on a commodity hardware
- Processor: i3, RAM: 3GB each for node
- Two Node Hadoop Cluster configured on Centos 6
- Hive, Pig and HBase are configured for the master
- Latest Apache Spark 1.1.0 configured for both the nodes
- Anaconda Ipython Notebook configured for the master node, modules namely 'pyspark', 'pandas', 'numpy', 'matplotlib' are installed

4.1 Performance

CPU and RAM are used more while executing the machine learning algorithm as shown in Figure 3. Red color represents the percentage of processor time for both the nodes, green and blue represents the page-reads/sec and pages/sec respectively. We can see that there is almost a linear trend between CPU and RAM.

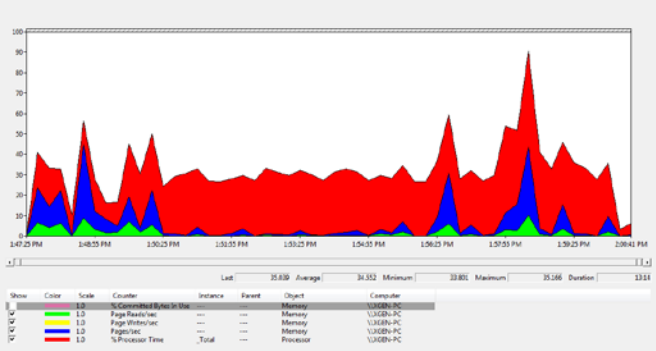

Figure 3: Performance Results

4.2 Machine Learning Algorithm

For Experimental purpose, Logistic Regression from Spark MLlib is used to run on a dataset with 5 Million rows and 4 independent variables. Figure 4 represents final output after training the model.

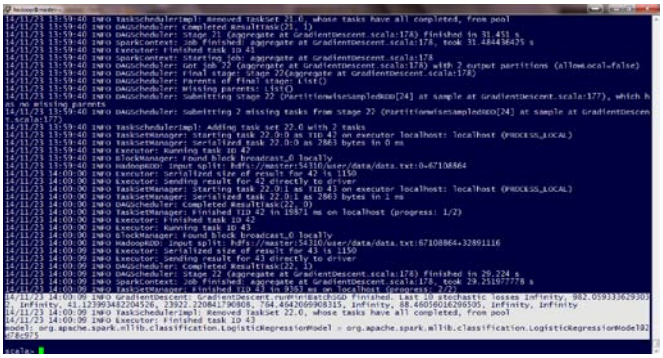

Figure 4: Spark Shell

Figure 5 represents Spark stages for Logistic Regression. Logistic Regression library takes 17 minutes to train the algorithm for dataset with 5 Million rows and 4 independent variables.

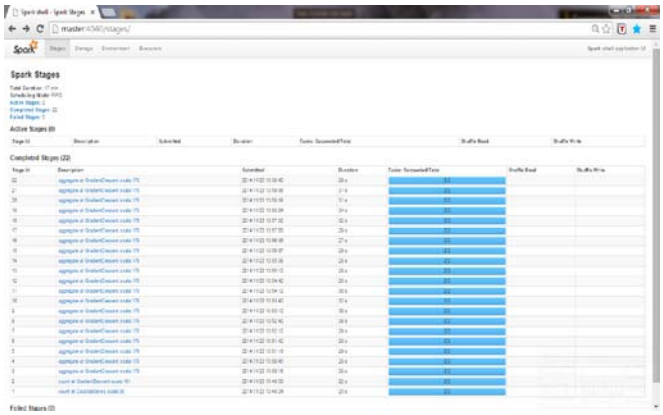

Figure 5: Spark Stages for Logistic Regression

5. Conclusion

Framework is very efficient for large datasets while running Machine Learning algorithms; Because of in-memory distributed computing, time spent on training algorithm using Spark is considerably less when compared to traditional tools. Spark can be easily deployed in the production environment with the other tools as mentioned in the framework.

References

[1] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica, “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing”, University of California, Berkeley.

[2] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin. Incoop: MapReduce for incremental computations. In ACM SOCC ’11, 2011.

[3] R. Bose and J. Frew. Lineage retrieval for scientific data processing: a survey. ACM Computing Surveys, 37:1–28, 2005.

[4] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst. HaLoop: efficient iterative data processing on large clusters. Proc. VLDB Endow., 3:285–296, September 2010.

[5] J. Cheney, L. Chiticariu, and W.-C. Tan. Provenance in databases: Why, how, and where. Foundations and Trends in Databases, 1(4):379–474, 2009.

[6] Scala programming language. http://www.scala-lang.org.

[7] Spark programming language. https://spark.apache.org.

Author Profile

Madhu Sudhan H V received B.E. in Electronics and Communication from Malnad College of Engineering and M.B.A in Finance from Christ University. During 2008-2010, he was working as Assistant Systems Engineer at Tata Consultancy Services. Presently he is working for AXA as a Data Scientist.