Spark (Lightning-fast Cluster Computing) Application in Telecommunication Sector to Prevent Customer Churn out

Rajkumar Khatri, Anil Kumar (Senior Member IEEE)
Mody University of Science and Technology,
Lakshmangarh, Sikar, INDIA

ABSTRACT
Apache Spark™ is a fast and general engine for large-scale data processing. Apache Spark is current buzzword which is gaining enormous attention because of its lightning-fast in-memory cluster computation of Big data. Spark programs are 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. Many organizations are using Spark for the processing of large datasets, Spark is igniting the world of big data analytics which requires fast performance, such as, interactive querying, iterative processing, large-scale batch computations as well as streaming, and graph computations. In this paper, we are finding the Call Details Records of customers facing frequent call drops in Roaming. This helps telecom companies to prevent customer churn out, to improve the connectivity issues in specific areas.

Keywords: Apache, Spark, in-memory, Cluster Computation, Hadoop, Big data

1. INTRODUCTION
Apache Spark is an open source big data processing framework which is best suited because of its in-memory speed, ease of use, and sophisticated analytics. Spark was originally developed in 2009 in UC Berkeley’s AMPLab, and open sourced in 2010 as an Apache project. Spark has many advantages compared to MapReduce, Storm and other hadoop technologies. Spark provides a comprehensive and unified framework to manage big data processing requirements with diverse datasets whether they are batch or real-time streaming data. Spark applications in Hadoop clusters run up to 100 times faster in memory and 10 times faster on disk. We can quickly write Spark applications in Java, Scala, or Python. It has built-in set of over 80 high-level operators. And we can interactively use it to query data within the shell. It supports Map and Reduce operations as well as SQL queries, streaming data, machine learning and graph data processing.

In this paper, we have a CDR (Call Details Record) file; and find out top 20 customers facing frequent call drops in Roaming. This is a very important report which telecom companies use to prevent customer churn out, by calling them back and at the same time contacting their roaming partners to improve the connectivity issues in specific areas.

2. BRIEF HISTORY OF SPARK
Spark was started by Matei Zaharia in 2009 as a research project in the UC Berkeley RAD Lab, later to become the AMPLab. Spark was first open sourced in March 2010 and transferred to the Apache Software Foundation in June 2013. Spark had over 465 contributors in 2014, making it the most active project in the Apache Software Foundation and among Big Data open source projects. The recent release of Spark is Spark 1.4.0, which was released on June 11, 2015.

3. Spark cluster computing architecture
Spark has similarities to Hadoop, but it represents a new cluster computing framework with useful differences. At beginning, Spark was designed for a specific type of workload in cluster computing which reuse a working
set of data across parallel operations. For the optimization of these workloads, Spark introduces the concept of in-memory cluster computing, where datasets are cached in memory to reduce their latency of access.2

Apache Spark introduces an abstraction called resilient distributed datasets (RDDs). RDDs are read-only collection of objects which are distributed across nodes. RDDs are resilient or fault tolerant, because they can be rebuilt if a part of the dataset is lost. The process of rebuilding a portion of the dataset relies on a fault-tolerance mechanism that maintains lineage (or information that allows the portion of the dataset to be re-created based on the process from which the data was derived).6

An RDD is represented as a Scala object and can be created from a file; as a parallelized slice (spread across nodes); as a transformation of another RDD; and finally through changing the persistence of an existing RDD, such as requesting that it be cached in memory.

Applications in Spark are called drivers, and these drivers operations are performed on single or a set of nodes. Hadoop as well as Spark supports a single-node or a multi-node cluster. Spark relies on the Mesos cluster manager for multi node cluster. Mesos provides a platform for resource sharing and isolation for distributed applications. So it allows Spark to work with Hadoop in a single shared pool of nodes.

4. SPARK VS. HADOOP

Hadoop as a big data processing technology exists for 10 years is a solution for processing large data sets. Map Reduce is a good solution for one-pass computations, but not efficient for multi-pass computations and algorithms. Each step in the data processing workflow has one Map phase and one Reduce phase and you'll need to convert any use case into MapReduce pattern to leverage this solution.

The Job output data between each step has to be stored in the distributed file system before the next step can begin. So Map Reduce is slow due to replication & disk storage. Hadoop solutions include clusters that are tough to set up and manage. This also requires the integration of several tools for Mahout and Storm.

MapReduce jobs have high-latency, and none could start until the previous job had finished completely. Spark enables programmers to develop complex and multi-step data pipelines using directed acyclic graph pattern. Spark also supports in-memory data sharing across DAGs, so that different jobs can work with the same data.

Spark runs on top of Hadoop Distributed File System (HDFS) infrastructure and additional functionality. It provides support for deploying Spark applications in an existing Hadoop. So Spark is an alternative to Hadoop MapReduce rather than a replacement to Hadoop. Spark is not developed to replace Hadoop but to provide a comprehensive/ unified solution to manage different big data use cases and requirements.

5. SPARK PLATFORM

![Spark SQL](#) ![Spark Streaming](#) ![MLLib (Machine Learning)](#) ![GraphX (graph)](#)

APACHESPARK CORE ENGINE

![Figure 1: Apache Spark Core Engine](#)
5.1 APACHE SPARK CORE ENGINE
Spark Core is the underlying general execution engine for the Spark platform that all other functionality is built on top of. It provides in-memory computing capabilities to deliver speed, a generalized execution model to support a wide variety of applications, and Java, Scala, and Python APIs for ease of development.

5.2 SPARK SQL
Many data scientists, analysts, and general business intelligence users rely on interactive SQL queries for exploring data. Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrames and can also act as distributed SQL query engine. It enables unmodified Hadoop Hive queries to run up to 100x faster on existing deployments and data. It also provides powerful integration with the rest of the Spark ecosystem (e.g., integrating SQL query processing with machine learning).

5.3 STREAMING
Many applications need the ability to process and analyze not only batch data, but also streams of new data in real-time. Running on top of Spark, Spark Streaming enables powerful interactive and analytical applications across both streaming and historical data, while inheriting Spark’s ease of use and fault tolerance characteristics. It readily integrates with a wide variety of popular data sources, including HDFS, Flume, Kafka, and Twitter.

5.4 MLLIB
Machine learning has quickly emerged as a critical piece in mining Big Data for actionable insights. Built on top of Spark, MLlib is a scalable machine learning library that delivers both high-quality algorithms (e.g., multiple iterations to increase accuracy) and blazing speed (up to 100x faster than MapReduce). The library is usable in Java, Scala, and Python as part of Spark applications, so that you can include it in complete workflows.

5.5 GRAPH-X
GraphX is a graph computation engine built on top of Spark that enables users to interactively build, transform and reason about graph structured data at scale. It comes complete with a library of common algorithms.

6.0 IMPLEMENTATION
We have generated 1-TBdata of customer having the following field first visitor location, second call drop, third phone number and error code.

Using Spark, we are able calculate call drop within 5.6 seconds; however it will take nearly 200 sec on-disk data.

7.0 CONCLUSION
We have presented Apache Spark, lightning-fast in-memory cluster computation, an efficient, fault-tolerant engine for processing bigdata. The survey results indicate that 13% are already using Spark in production environments with 20% of the respondents with plans to deploy Spark in production environments in 2015, and 31% are currently in the process of evaluating it. In total, the survey covers over 500 enterprises that are using or planning to use Spark in production environments ranging from on-premise Hadoop clusters to public clouds, with data sources including key-value stores, relational databases, streaming data and file systems. Applications range from batch workloads to SQL queries, stream processing and machine learning.

We have found the application of Spark in telecommunication to prevent customer churn out.

6. REFERENCES
1. HTTPS://SPARK.APACHE.ORG/
3. https://databricks.com/spark/about