Analysis of Call Data Record (CDR) using Hadoop Cluster

Authors: Toufiqur Rahman Chowdhury, Arun Sai Prakash Arumugam; Supervisor: Jeongkyu Lee
Department of Computer Science and Engineering
University of Bridgeport, Bridgeport, CT - 06604

Abstract

The management of big data is the most important issue for this decade since the real world applications are generating very large scale of data in petabytes and zetabytes scale. Most popular solution of big data management is a system based on Hadoop Distributed File System.

However, implementing enterprise level solution is a challenge because of the production of such huge data. In this project, we employ Hadoop cluster to the telecommunication data since it produces a huge amount of log data regarding to customer calls as well as network equipment. To emphasize more realistic solution we ponder on call data details for our Big Data application.

In this project, we have acquired real-time Call Data Record (CDR) data for our implementation from telco operator named Banglalink who is operating 30 million users in Bangladesh. To narrow down the scope, CDR data analytics using Hadoop cluster can result top callers to promote customer experience. This implement can also help Banglalink to implement similar application for backup data warehouse using Hadoop cluster for CDR analytics.

Hadoop Architecture

Hadoop Distributed File System (HDFS) can handle large scale of data-set in zetabytes using sequential read and write operation. To process files are divided into chunks and stored into three data nodes. A Name Node and thousands of Data node can be employed to serve for processing such large scale of dataset. In Name node, Job Tracker at first, gets job from Client node and assigns task to Task-trackers in Data nodes. Second, it co-ordinates Map and Reduce Phases and provide job progress info. So, MapReduce plays a very important role and is employed to process input format of splits or chunks using user defined 'Map' function, shuffle and then sort and finally merge using to achieve output data.

MapReduce is the restricted programming model to process and generate large scale data set over HDFS. MapReduce programs are automatically parallelized as well as executed in HDFS. In Map function, user specifies statements to process a key/value pairs for generating intermediate key/value pairs. In Reduce function, all intermediate values get merged for the same intermediate keys.

Since the input data set can be very large the computation is distributed across thousands of cluster the complexity of such systems is the process of parallelism of computation. MapReduce provides the encapsulation of details of parallelization, fault tolerance, load balancing and data distribution using its restricted functional programming model offering simple and powerful interface.

CDR Data Format

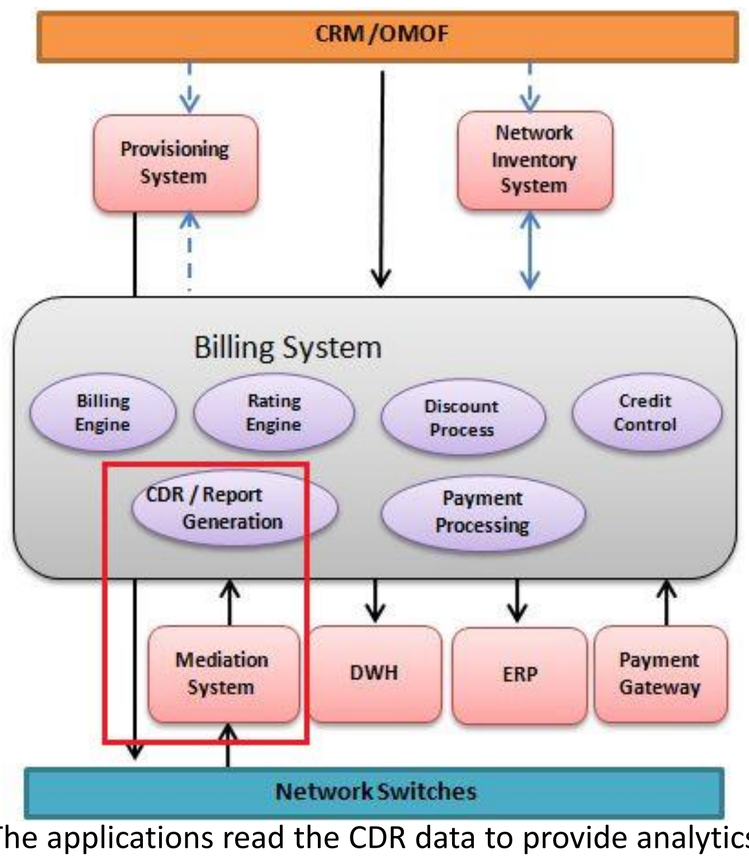
1	000030	470036500840510	1912858068	L	358395041433190		
2	000002	470037700377798	1928042391	L	355115057893670		02018801900712062
3	000030	470031203484191	1967867117	L	355701053326490		
4	000001	470036500199526	1926198622	L	355454055622040		02018801900894991
5	000030	470031202835401	1966992799	L	352841053676240		
6	000002	470034200118055	1914890674	L	011982000435630		02018801900717064
7	000030	470035000195206	1915298468	L	355047009107730		
8	000002	470 IMSI 08059296	198 SUBNO 8	L	867961010159610		02018801900715982
9	000001	470031408294861	1950490386	L	359384055412590		02018801900715982
10	000002	470032700865356	1977196529	L	352685056883160		02018801900717545

1	809316	131127175142
2	801747797991	13112717511230
3	80492	131127175142
4	801927112537	1311271174807216
5	801616340571	131127175142
6	801715041438	13112717513210
7	80492	131127175142
8	80 B SUBNO 86	1 CALLDURATION
9	801989665108	131127175113 30
10	801853943110	13112717511528
11	801973396220	13112717504558
12	801943825668	13112717511627
13	801963880375	1311271174340483
14	801620141097	13112717512320
15	801948337280	13112717511627

Though the highlighted data are IMSI and SUBNO, B-SUBNO and CALL DURATION, we have also considered time start for call in our algorithm

Methodology

Considering the Telecommunication Billing architecture presented below, we gathered the data set from Mediation System. The system generates data from usage data of end user from network switches and builds a call detail record (CDR) after the call is made. The CDR is a standard call data record that contains A party number (Subscriber Number) and B party number (Destination Number) as well as start and end date and times for any call made using the Telco operator network.



The applications read the CDR data to provide analytics operation for a given subscriber number. We have targeted to prepare two applications from CDR data purposefully with advise from the Telecom Company we are working with.

1. The first application provides total call duration for each user. From which we can conclude the Top N caller
2. Another application provides the destinations for each caller with duration and start time for call

The Problem statement

The CDR data set needs to be parsed to necessary fields like subscriber number, duration and destinations. For a given subscriber number, the application will find out key and value pair as subscriber number and call duration. Finally, we can get the total call duration that can be sorted to get Top N callers using automatic Key sort and also Call destination with timestamp and duration made by the subscriber number.

Solution Algorithm

Application 1:

Step1: At first we put the test Call Data Record (CDR) CDR data set to HDFS cluster.

Step2: Secondly, we will create a 'Map' function to parse the Call Data Record (CDR) data. Our algorithm for 'Map' function perform line reading and then parse the Subscriber Number and the Call Duration using *substring (int beginIndex, int endIndex)* function for given length for different fields.

Step 3: The analytics application1 combiner then sums up all the call durations for the given subscriber number from all the mapper output. This will result with key/value pair as Subscriber Number and total call duration.

Step 4: The reducer has sum up functionality and does the toggling of key and value pair.

It results with key/value pair as and total call duration Subscriber Number.

Step 5: We use the following command to find out the Top 10 callers,

$ hadoop fs cat output/part | sort –rg | head –n 10*

Application 2:

Step1: At first we put the test Call Data Record (CDR) CDR data set to HDFS cluster.

Step2: Secondly, the algorithm for 'Map' function perform line reading and then parse the Subscriber Number, Call Duration, Destination number and Start time using *substring (int beginIndex, int endIndex)* function for given length for different fields. Finally, the algorithm pass key/value pairs as <Text , Text > where the Value contains JSON format text data that can be highly usable by Web-Service to feed any application.

CDR App Hadoop Cluster Architecture

For our project we have implemented a Hadoop cluster with 1 master node and 3 salve node in UB Big Data Lab. We have used Cloudera hadoop version 2.0.

We have checked carefully the behavior or our master and slave nodes while processing of medium large scale amount of CDR data, approximately 702 MB reak time test data. We have followed the following steps during our test,

1. We have run Master node and Slave nodes daemons accordingly.
2. We have started monitoring our server using port 50070 ad 50030.
3. We have uploaded our data from Client System (in our case which is master node itself)
4. We checked the data occupancy in data nodes from the monitoring system.
5. We have run hadoop jar command to run our executable application written in Java that was build in our development system.
6. We followed up the system behavior in the cluster for all data nodes with the master node.
7. We have successfully run our applications in our own cluster in Big Data Lab.

Data Throughput Performance: an Overview

Though the cluster size is small but this highly scalable. We can plug-in more data node anytime according to our need of data processing. In our case, the cluster Map job capacity is 8 and also Reduce jobs capacity is 8. In future, we intend to upgrade our cluster as well as increasing the CDR data volume to find more insight of the analytics.

Conclusion

For implementation, we will use 4 node Hadoop clusters to test around 700 MB of data to measure the performance throughput for Call Data Record (CDR) data analytics applications. For which Application 1 takes 120000 ~ 130000 ms and for Application 2 , it takes 140000 ~ 150000 ms whare as in pseudo-cluster it takes more than 180000 ms.

This application should help understanding to realize deeper insights of customer behavior patterns to find out revenue per user as well as offering promotion to leverage customer experience. The applications are peoduction solution for global telecom section since the Call Data Record (CDR) data set is the standard network data from Mediation System of State of Art Telecom Infrastructure.

References:

1. MapReduce: Simplified Data Processing on Large Clusters by Jeffrey Dean and Sanjay Ghemawat jeff@google.com, sanjay@google.com Google, Inc.
2. A Comparison of Approaches to Large-Scale Data Analysis Andrew Pavlo Erik Paulson Alexander Rasin