IPL Data Analysis

Kunal Kiran Patil
15046974
MSc. Cloud Computing
National College of Ireland

Abstract **IPL is Indian Premier League which was started in 2008. Every year there are seasons of IPL. The dataset of IPL is collected in this project in which there are all seasons from 2008 to 2016. The dataset consists of eleven columns named as season, date, team1, team2, toss winner, toss decision, winner, win by runs, win by wickets, player of the match and venue. The MapReduce is performed on the data using JAVA to analyze the result. To analyze the result three types of analysis is performed, first is the filtering the teams, in the second analysis the toss winner teams are counted and third is the distinct analysis in which match won by every team is calculated.**

Keywords ***MapReduce, JAVA, data analysis, Hadoop***

I. INTRODUCTION

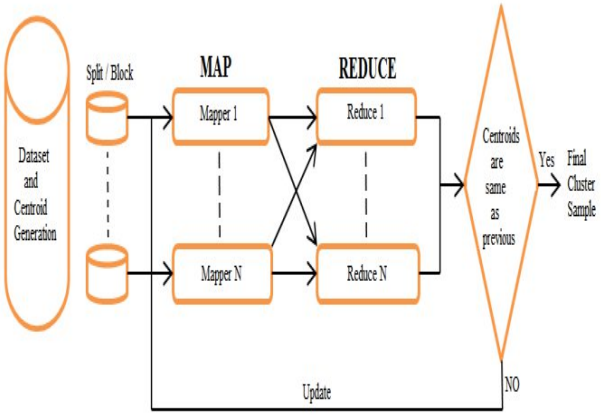
In the IPL data analysis the data of all the teams is taken that have played the seasons from 2008 to 2016. From all these teams that are put in the datasets according to the date are filtered according to the teams the matches played by each team. The number of teams is counted that have won the toss as well as won the match and the teams that have won the toss won by the team but have lost the match. Then the teams that have won the specific seasons and have won number of seasons are sorted and counted. This process of analysis will give the result to the viewer according to the teams and seasons. The hadoop mapreduce process is used in this project by using the JAVA.

A. Research Question ?

How the IPL data analysis will help us to check which team has won more matches by winning the toss and to how to sort the matches won by each team?

II. RELATED WORK

Software framework is known as hadoop mapreduce. [1]In this framework we can write the application easily. A large amount of data can be processed on large clusters in the parallel processing. In the fault-tolerant manner reliable commodity hardware is used. A dataset which is put in the mapreduce job is divided or split into number of chunks. These chunks of the dataset are put in the processing in the memory tasks which is processed in the parallel manner. The output which is given in the map is sorted in the framework. Then to reduce the task again it is inserted. The file system sorts the input and output of the job. The framework does all the process in the mapreduce like task scheduling in which the tasks are processed in a particular manner. Then the monitoring of scheduled task is the next process which is done by the framework. The tasks that are failed during the execution time are again re-executed by the framework.



The above diagram shows the execution of mapreduce. It shows the dataset which generates the centroids. Then the dataset is split in the blocks or chunks of data which is sorted in the separate data. It has various types of datasets which depends on the number of data and the site of data. Then the spited block of data is then send to mapper to map the data and after mapping of data it is reduced. The mapper and reducer is equal to every dataset and they are inter-related to each other. Once the process is completed then again the all the spitted data is merged like the process or the initial dataset , but here the merged data is in the sorted format and which gives the proper output that is been required by the user or the programmer. If the desired output is generated then the final cluster sample is shown in the output dataset. But, if the desired output is not generated then it is again send back to map the data or again process the mapper and reducer process on the desired output. For this process the JAVA mapreduce is used. In which we can filter the data and then can process it.

[2]There are two types of nodes that are compute and the storage nodes these are the same type of nodes. The mapreduce framework executes on the same node. The HDFS (hadoop-distributed-file-system) also runs on the same node. The data that must be present in the database, then the framework schedules the task on each node and that generates the bandwidth in every cluster. The configuration of the framework does the entire task. In the mapreduce there is one single master which is known as the job tracker that tracks all the types of job which scheduled, executed or to be monitored. The other one is the slave node which is task tracker that

tracks the task of execution in every cluster-node. The slave schedules all the tasks for the slave. The slave works only on the master s direction or directed by the master.

The input and output locations are specified by the application. The jobs are submitted by the job client in the hadoop (i.e. jar/executable). The job tracker configuration takes the responsibility of software distribution and configuration that is been needed by the salve. The scheduling of task is also done by the job tracker.

A. *Input and Output:-*

[3]Mapreduce framework operates on <key, value> pairs. The Input of the job is viewed by the framework in <key, value> pairs. It also produces an output in the pair of <key, value> which is of different types. The classes of key and value must be in the serial manner implemented by the framework. It also implements the writable interface of the classes that have generated the key and value. The flow of input and output job is shown below.
(input) <k1, v1> -> map -> <k2, v2> -> combine -> <k2, v2> -> reduce -> <k3, v3> (output)

B. *Mapper:-*

[4]The main task of the mapper is mapping the input pairs of key and value. It then sets the pair of intermediate key and value. Map runs on the individual task which converts all the input records into the intermediate form. The converted intermediate should be of different types considering the input once. The pair of given input may be zero to number of outputs in the pair.

C. *Reducer:-*

Reducer is used to reduce a set of all the intermediate values that shares a key for the smaller set of all the values. There are basically 3 types of primary phases in reducer.

1. *Shuffle:-*

The input given to the reducer is in the sorted form of output of the generated mappers. In, this phase, through HTTP the framework fetches all the relevant partition of all the output which is in the mapper.

2. *Sort:-*

In this stage the framework groups all the reducer input by the key format. The sort phase and the shuffle phase works simultaneously. The outputs in the map are fetched and then they are merged together.

3. *Reduce:-*

In this phase the method calls for each pair <key, (list of Values)> which is in the group format of the inputs. The reduce tasks output is directly written in the file system through output collector. Collect (writable comparable, writable). The output given by the reducer is in the unsorted format.

D. *Partitinor:-*

It is mainly used to portion the key space. [5]The partitioning of the keys is controlled by the partitinor for all the intermediate map-outputs. To desire the partition format the key or the sub-key is used, which typically uses the hash function. The numbers of partition are similar to the number of tasks that are reduced for the job. The controls of all the intermediate key is sent for the reduction purpose. Hash-partition is also co default partitioner.

E. *Reporter:-*

[6] Reporter is a facility given for the mapreduce application to give the progress of the report. It sets the states messages of application level and also updates the counters. The mapper and reducer implementations use the reporter for reporting the progress of the tasks. It also indicates that the process is in working or alive. The application takes a proper time for processing the key/value of the pair. This process becomes crucial as the working framework assumes that the task that is running has reached out of the time and it may kill the task. To avoid this we can set a configuration parameter (mapred.task.timeout) for the high level of value. We can also set to zero so that there will be not timeout. Using reporter the counters are updated by the applications.

F. *Output Collector:-*

[7]It is the generalized format for the facility provider which provides the facility for the mapreduce framework, which is used to collect the output of the data through the mapper or the reducer. The collected output may be in the intermediate form or the jobs of output. The Hadoop mapreduce is in the bundled format which consists of library of all the useful mappers, reducer and the partitioners that are used for the generation of the output.

III. IMPLEMENTATION

The Indian Premier League (IPL) dataset is considered which consists of eleven columns. The data is in numeric and in alphabetical format. The named columns are given below.

1. Season (2008-2016)
2. Date (18/04/2008 to 29/05/2016)
3. Team1 (first playing team)
4. Team2 (second playing team)
5. Toss winner (team that has won the toss)
6. Toss decision (field-bat)
7. Winner (team winner)
8. Win by runs (won by number of runs)
9. Win by wickets (won by number of wickets)
10. Player of the match (best player played in the match)
11. Venue (name of the stadium)

The dataset was chosen because it has all the information of the matches and the seasons from the start of IPL to the last played seasons. It contains all the basic information of all the matches with the name of the top players and the venue of the match.

- *Data Preparation for the analysis*

The dataset is considered in which all the data is put in the SQL which is needed by the mapper to map all the data. The mapper is the first component that is used to fetch the data from the database. First the table was created by using CREATE DATABASE. The IPL database is created which consists of all the eleven columns that are to be used. The data of the database was not having any primary key; an auto generation of primary key is taken that have generated the date of the data as a primary key. This data is now put in the HDFS for the next processing.

In the analysis of the project the three patterns are used to generate the desired output, the patterns are filtering pattern, the distinct pattern and the summarization pattern.

A. Filtering Pattern:-

The filtering pattern is used to filter all the data that is not to be used. The filter mapper is used to retrieve all the input data from IPL dataset then filter it and then it is been extracted into the output file. The names of all teams is taken and which teams has won the maximum limits are sorted by using map(key, record):
By using this mapping function we can decide which value should be and want to remove.
The filter pattern used in the form of WHERE clause for executing the filter pattern.
SELECT * FROM IPL WHERE value < 50.

B. Summarization Pattern:-

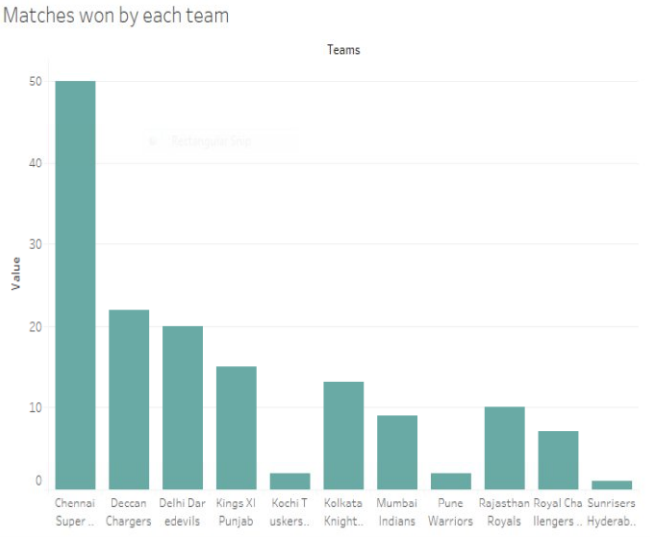
The summarization pattern is the next pattern used in this IPL analysis in which the counting of the match winner. The first count that is run in this analysis is the Toss Winner with the Match Winner, in which the number of teams that have won the toss and also have won the matches is calculated. In the second phase of the count process the number of that has won the toss but has lost the match is counted.
Steps processed in the summarization patterns:-
Step 1: All the cleaned IPL data is send to the mapper to map it.
Step 2: The combiner works in the data combination to combine the related data.
Step 3: Then the partitioner is used that partitions all the combined data for the reducer.
Step 4: The shuffle and sorting of algorithm is used to sort out the data that has been combined,
Step 5: In the end the reducer is used to create the output data that is sorted for the analysis.
Step 6: The analyzed output is shown in the new database.

C. Distinct Pattern:-

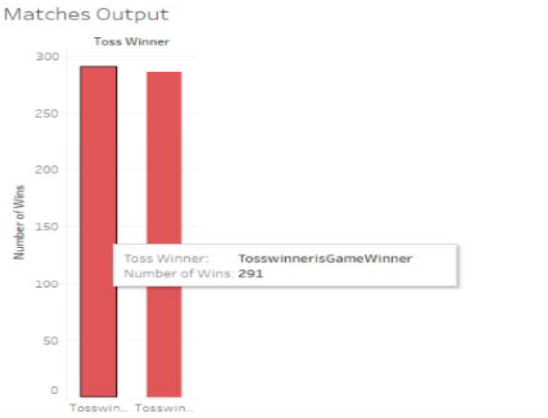
The distinct dataset is used to analyze the match won by every team. In this pattern every teams winning data is analyzed and matches won by every team is analyzed. Mapper is responsible for fetching each records from database table. Teams played the match is extracted from each data row and used as a key. Team who won the match is used as a value for mapper task. Later in reducer, No of matches won by each team is counted based on the value.

IV. RESULT

The following graph shows the number of teams that have won the matches. The number of matches is also shown with the numeric values. In this result the maximum matches won by Chennai Super kings with the value of 50 matches is shown in the graph.



In the second analysis the number of toss wins and the number of matches won by the teams is shown. The teams that have won toss have won the matches are 291 teams. In the second bar graph it shows that the number of teams that have won the toss but have lost the matches are 286.



V. CONCLUSION

In the above implemented project it shows that the numbers of teams that have won the matches by winning the toss are 291 times. The teams that have won the toss but did not win the match are 286 times, throughout the seasons from 2008 to 2016. In another result it shows the output of all the individual teams that have won the matches in all the seasons. The record shows that the Chennai Super kings are on the top as they have won 50 matches through-out the seasons. The least are the Sunrisers Hyderabad that has won only one match in all the seasons. The rest of the teams are in between these two teams. So, the obtained result becomes easy for the cricket fans to judge the teams in the future seasons as well as the players that are playing in these teams.

REFERENCES

[1] A. Pal, K. Jain, P. Agrawal and S. Agrawal, "A Performance Analysis of MapReduce Task with Large Number of Files Dataset in Big Data Using Hadoop," *2014 Fourth International Conference on Communication Systems and Network Technologies*, Bhopal, 2014, pp. 587-591. doi: 10.1109/CSNT.2014.124J.

[2] A. Boukhdhir, O. Lachiheb and M. S. Gouider, "An improved mapReduce design of kmeans for clustering very large datasets," *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, Marrakech, 2015, pp. 1-6. doi: 10.1109/AICCSA.2015.7507226.

[3] K. C. Wee and M. S. M. Zahid, "Auto-tuned Hadoop MapReduce for ECG analysis," *2015 IEEE Student Conference on Research and Development (SCOReD)*, Kuala Lumpur, 2015, pp. 329-334. doi: 10.1109/SCORED.2015.7449350.

[4] C. Verma and R. Pandey, "Big Data representation for grade analysis through Hadoop framework," *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, Noida, 2016, pp. 312-315.
doi: 10.1109/CONFLUENCE.2016.7508134

[5] J. Zhu, J. Li, E. Hardesty, H. Jiang and K. C. Li, "GPU-in-Hadoop: Enabling MapReduce across distributed heterogeneous platforms," *2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS)*, Taiyuan, 2014, pp. 321-326. doi: 10.1109/ICIS.2014.6912154

[6] V. V. Pol and S. M. Patil, "Implementation of on-process aggregation for efficient big data processing in Hadoop MapReduce environment," *2016 International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, 2016, pp. 1-5. doi: 10.1109/INVENTIVE.2016.7830105

[7] D. Garg, P. Gohil and K. Trivedi, "Modified Fuzzy K-mean clustering using MapReduce in Hadoop and cloud," *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, 2015, pp. 1-5. doi: 10.1109/ICECCT.2015.7226046