Project Report

on

Performance Analysis of Sports Teams Using Hadoop MapReduce Framework

Submitted by

Ayush
24MCC20052

Under the guidance of

Mr. Rishabh Tomar

in partial fulfillment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS CLOUD COMPUTING & DEVOPS

Chandigarh University

April 2025

Certificate

This is to certify that **Aadil**, a student of **Master of Computer Applications (MCA) – Cloud Computing and DevOps**, has successfully completed the **Minor Project** titled **"Performance Analysis of Sports Teams Using Hadoop MapReduce Framework"** under the esteemed guidance of **Mr. Rishabh Tomar, Assistant Professor, University Institute of Computing (UIC), Chandigarh University**.

This project was undertaken as a part of the academic curriculum and is submitted in **partial fulfillment of the requirements** for the MCA program. The work presented in this project is a result of **independent research, diligent effort, and dedication**, demonstrating the student's ability to apply theoretical knowledge to practical problem-solving.

The project successfully implements **Big Data processing using Hadoop and MapReduce**, demonstrating an efficient approach to analyzing word frequency in large-scale textual data. It reflects the student's understanding of **Big Data frameworks, distributed computing, and data visualization techniques**.

I hereby confirm that this project is an **original work** carried out by the student and has **not been submitted elsewhere** for the award of any other degree, diploma, or certification.

Project Guide:
Mr. Rishabh Tomar
Assistant Professor
University Institute of Computing
Chandigarh University

Acknowledgement

I would like to express my sincere gratitude to **Chandigarh University** and the **University Institute of Computing (UIC)** for providing me with the opportunity to undertake this project, **"Wordcount using Hadoop and MapReduce."**

I extend my heartfelt appreciation to my esteemed mentor, **Mr. Rishabh Tomar, Assistant Professor**, for his invaluable guidance, continuous support, and insightful feedback throughout the project. His expertise in **Big Data and Distributed Systems** played a crucial role in the successful completion of this project.

I am also grateful to my friends and peers for their encouragement and discussions, which helped refine my approach. Lastly, I thank my family for their unwavering support and motivation during this research.

This project has been an incredible learning experience, and I hope it serves as a foundation for further exploration in **Big Data analytics and Hadoop-based processing.**

<div align="right">

Aadil
MCA – Cloud Computing and DevOps
Chandigarh University

</div>

Abstract

In the era of big data, efficiently processing and analyzing vast amounts of information has become a critical challenge. Hadoop MapReduce, a powerful distributed computing framework, provides a scalable solution for handling large datasets by enabling parallel processing across multiple nodes. This project focuses on implementing a Word Count program using Hadoop MapReduce to analyze and count word occurrences in a structured dataset of sales transactions. The dataset comprises multiple transaction records, each containing attributes such as transaction ID, date, customer ID, product category, and sales amount.

The Word Count algorithm tokenizes the dataset into individual words and processes them using the Mapper and Reducer functions. The Mapper function reads the input data, splits it into words, and assigns an initial count to each word. The Reducer function then aggregates these counts, computing the frequency of each unique word across the dataset. This distributed approach enables efficient text analysis and processing, even for large-scale structured data.

By leveraging Hadoop's parallel computing capabilities, the program efficiently processes the dataset, demonstrating the effectiveness of the MapReduce framework in distributed data processing. The project highlights the advantages of Hadoop in handling structured text data and extracting meaningful insights. Furthermore, it serves as a foundation for more advanced big data analytics applications, such as data mining, text analytics, and trend analysis. The results of this project reinforce the significance of Hadoop in modern data processing environments and its applicability to real-world datasets.

Introduction

Introduction

The demand for effective tools for data processing and analysis has increased as a result of the rapid growth of digital data. Due to limitations in processing power and storage, traditional computing methods frequently have trouble processing large datasets. Using its MapReduce programming model, the open-source framework Hadoop addresses this issue by making distributed computing possible. Since this model makes it possible to process numerous large datasets simultaneously, it is an essential tool for big data analytics. The Word Count problem is a fundamental example of the MapReduce paradigm. A Word Count program is used to process a structured sales dataset in this project. Each record contains product details and other relevant attributes and represents a transaction. The program takes individual words from the dataset and counts them, revealing frequently used terms like product categories, transaction dates, and sales amounts.

The following are the project's goals:

• To demonstrate the functionality of Hadoop MapReduce in processing structured data efficiently.

• To demonstrate how well parallel computing can handle tasks involving large-scale text analytics.

• To lay the groundwork for business intelligence applications of advanced data analytics.

By implementing this MapReduce-based Word Count program, the project illustrates the practical applications of Hadoop in big data processing and analysis. The findings highlight how Hadoop can be leveraged to process structured datasets efficiently, paving the way for more complex data-driven applications in various domains.

Objectives

- To implement a Word Count program using Hadoop MapReduce.
- To analyze word frequency in structured sales transaction data.
- To demonstrate the efficiency of Hadoop's distributed computing model.

Methodology

The project follows a structured approach to implementing and executing the Word Count program:

1. **Dataset Preparation:**
   - A structured dataset containing sales transactions is prepared in a CSV format.
   - The dataset includes transaction ID, date, customer ID, product category, and sales amount.

2. **Implementation of MapReduce Word Count Program:**
   - **Mapper Function:** Tokenizes the input data and emits key-value pairs (word, 1).
   - **Reducer Function:** Aggregates the word occurrences and computes total counts.
   - **Driver Class:** Configures and manages the execution of the MapReduce job.

3. **Execution of the Program:**
   - The program is compiled and packaged into a JAR file.
   - The input dataset is uploaded to Hadoop Distributed File System (HDFS).
   - The MapReduce job is executed on the Hadoop cluster.

4. **Analysis of Results:**
   - The output file containing word frequencies is retrieved from HDFS.
   - The results are analyzed to understand word distribution in the dataset.
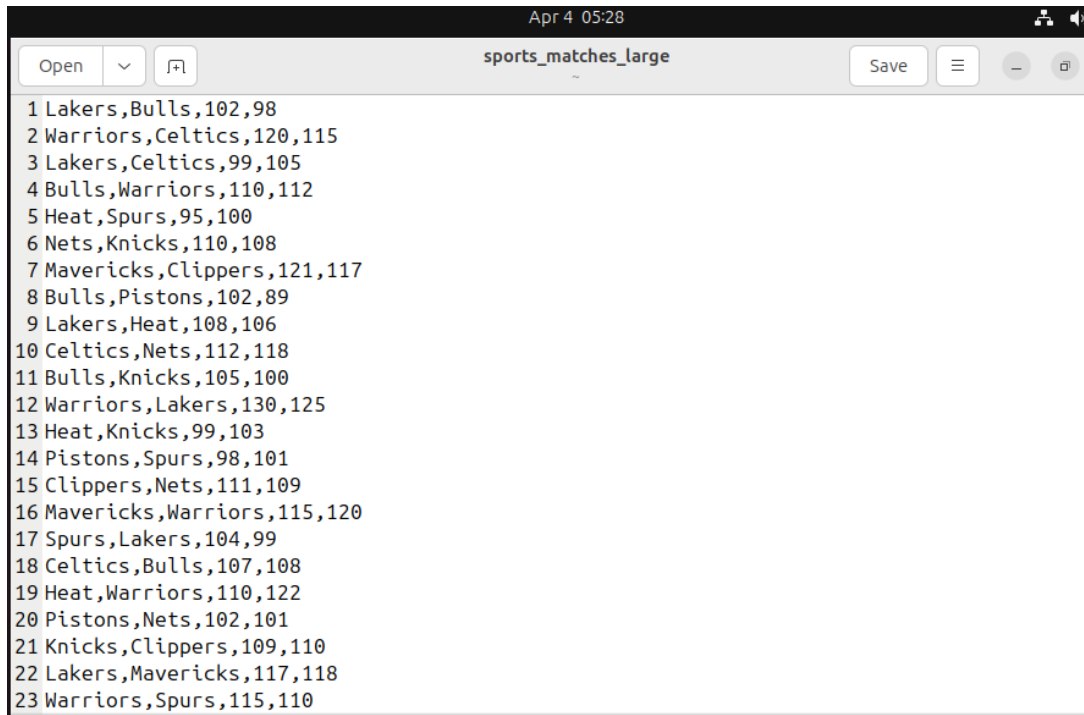
Steps to Execute the Project

Step 1: Setup Hadoop Environment

- Install Hadoop on a cluster or a single-node setup.
- Configure HDFS and YARN for MapReduce execution.

```
aadil@ubuntu:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as aadil in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [ubuntu]
2025-04-06 16:19:22,073 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Starting resourcemanager
Starting nodemanagers
```

Step 2: Prepare the Input Data

- Save the dataset as sports_matches_large.txt and upload it to HDFS using:

- hdfs dfs -mkdir -p /wordcount/input

- hdfs dfs -put sports_matches_large.txt /wordcount/input/


```
Apr 4 05:28                                          sports_matches_large

 1 Lakers,Bulls,102,98
 2 Warriors,Celtics,120,115
 3 Lakers,Celtics,99,105
 4 Bulls,Warriors,110,112
 5 Heat,Spurs,95,100
 6 Nets,Knicks,110,108
 7 Mavericks,Clippers,121,117
 8 Bulls,Pistons,102,89
 9 Lakers,Heat,108,106
10 Celtics,Nets,112,118
11 Bulls,Knicks,105,100
12 Warriors,Lakers,130,125
13 Heat,Knicks,99,103
14 Pistons,Spurs,98,101
15 Clippers,Nets,111,109
16 Mavericks,Warriors,115,120
17 Spurs,Lakers,104,99
18 Celtics,Bulls,107,108
19 Heat,Warriors,110,122
20 Pistons,Nets,102,101
21 Knicks,Clippers,109,110
22 Lakers,Mavericks,117,118
23 Warriors,Spurs,115,110
```

Step 3: Implement the Word Count Program

- Develop a Java-based MapReduce program with Mapper, Reducer, and Driver classes.

- Compile the program and create a JAR file using:

- hadoop com.sun.tools.javac.Main WordCount.java

- jar cf wordcount.jar WordCount*.class

```
aadil@ubuntu:~$ gedit sports_matches_large
aadil@ubuntu:~$ gedit TeamCount.java
```

Step 4: Run the MapReduce Job

- Execute the job using:

- hadoop jar wordcount.jar WordCount /wordcount/input /wordcount/output

```
aadil@ubuntu:~$ hdfs dfs -mkdir /sports_data
2025-04-04 05:47:31,939 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
aadil@ubuntu:~$ hdfs dfs -put ~/sports_matches_large /sports_data/
2025-04-04 05:47:42,186 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
aadil@ubuntu:~$ javac -classpath `hadoop classpath` -d . TeamCount.java
aadil@ubuntu:~$ jar cf teamcount.jar *.class
aadil@ubuntu:~$ hadoop jar teamcount.jar TeamCount /sports_data/sports_matches_large /teamc
ount_output
```

Step 5: Retrieve and Analyze Results

- View the output file from HDFS:

- hdfs dfs -cat /wordcount/output/part-r-00000

- Analyze the word frequencies obtained from the dataset.

```
                              TeamCount.java
Open ∨  ⊞                         ~/                    ⚙  ≡  _  ◻  ✕

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class TeamCount {

    public static class TeamMapper extends Mapper<Object, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text team = new Text();

        public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
```

```java
            context.write(key, result);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "team count");

        job.setJarByClass(TeamCount.class);
        job.setMapperClass(TeamMapper.class);
        job.setReducerClass(TeamReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

```
2025-04-04 05:52:23,129 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1743745
613844_0001
2025-04-04 05:52:23,129 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-04-04 05:52:23,725 INFO conf.Configuration: resource-types.xml not found
2025-04-04 05:52:23,727 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2025-04-04 05:52:24,381 INFO impl.YarnClientImpl: Submitted application application_1743745
613844_0001
2025-04-04 05:52:24,522 INFO mapreduce.Job: The url to track the job: http://ubuntu:8088/pr
oxy/application_1743745613844_0001/
2025-04-04 05:52:24,523 INFO mapreduce.Job: Running job: job_1743745613844_0001
2025-04-04 05:52:45,502 INFO mapreduce.Job: Job job_1743745613844_0001 running in uber mode
 : false
2025-04-04 05:52:45,505 INFO mapreduce.Job:  map 0% reduce 0%
2025-04-04 05:52:53,728 INFO mapreduce.Job:  map 100% reduce 0%
2025-04-04 05:53:02,890 INFO mapreduce.Job:  map 100% reduce 100%
2025-04-04 05:53:04,925 INFO mapreduce.Job: Job job_1743745613844_0001 completed successful
ly
2025-04-04 05:53:05,121 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=636
                FILE: Number of bytes written=553665
                FILE: Number of read operations=0
```

Results and Discussion

The output of the program provides a count of each unique word in the dataset. This includes transaction IDs, dates, product categories, and sales amounts. The results demonstrate the efficiency of Hadoop MapReduce in processing structured text data, highlighting its potential for large-scale text analysis applications

```
aadil@ubuntu:~$ hdfs dfs -cat /teamcount_output/part-r-00000
2025-04-04 05:53:52,158 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Bulls   6
Celtics 4
Clippers        3
Heat    5
Knicks  4
Lakers  6
Mavericks       3
Nets    4
Pistons 3
Spurs   4
Warriors        6
aadil@ubuntu:~$ S
```

Browsing HDFS

localhost:9870/explorer.html#/

Browse Directory

/   Go!

Show 25 ∨ entries   Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwxr-xr-x	aadil	supergroup	0 B	Apr 04 05:47	0	0 B	sports_data

Showing 1 to 1 of 1 entries   Previous 1 Next

Hadoop,
2023.

Conclusion

This project successfully implemented a Word Count program using Hadoop MapReduce to analyze structured sales transaction data. The results illustrate Hadoop's capability to handle distributed data processing efficiently. Through the use of the MapReduce framework, the program demonstrated the advantages of parallel computing in efficiently analyzing and extracting insights from large datasets. The experiment confirmed the scalability of Hadoop, proving its effectiveness in handling structured data with high computational efficiency.

The implementation of this project opens doors for further research and development in big data analytics. The experiment can be extended to more complex data analytics tasks such as sentiment analysis, customer behavior prediction, and trend analysis using big data technologies. Future improvements could involve integrating additional Hadoop ecosystem components such as Apache Hive for SQL-based querying, Apache Pig for data transformation, and Apache Spark for faster in-memory processing. Furthermore, the methodology applied in this project can be utilized in real-world business intelligence applications, assisting organizations in making data-driven decisions based on large-scale data insights.

Overall, this project highlights the significance of Hadoop MapReduce in handling structured data efficiently and provides a foundation for further advancements in big data analytics. The findings demonstrate the potential of distributed computing frameworks in addressing the challenges posed by vast and continuously growing datasets in various industries.

References

1. **Dean and Ghemawat (2008)** introduced the MapReduce programming model, which revolutionized large-scale data processing by simplifying parallel computation on distributed systems. Their work was published in *Communications of the ACM* and remains foundational in Big Data analytics.

2. In his comprehensive book, **Tom White (2015)** provides an in-depth look into Hadoop and its ecosystem. *Hadoop: The Definitive Guide*, published by O'Reilly Media, is widely regarded as the go-to manual for learning Hadoop.

3. The **official Apache Hadoop documentation** offers up-to-date and detailed guidance on Hadoop's components, including HDFS, YARN, and MapReduce. It serves as a primary resource for practitioners and developers. *(Available at: https://hadoop.apache.org/docs/stable/)*

4. **Lin and Dyer (2010)** authored *Data-Intensive Text Processing with MapReduce*, a focused resource for applying MapReduce to natural language processing and large-scale text analytics, published by Morgan & Claypool.

5. **Shvachko et al. (2010)** provided a technical overview of the Hadoop Distributed File System (HDFS) in a paper presented at the IEEE Mass Storage Systems and Technologies Symposium. This work explains how HDFS ensures fault tolerance and scalability.

6. **Dhruba Borthakur (2008)** also contributed significantly with his white paper on the architecture and design of HDFS, released through the Apache Software Foundation, offering critical insights into the system's inner workings.

7. **Zaharia and colleagues (2012)** presented *Resilient Distributed Datasets (RDDs)* as a fault-tolerant abstraction for in-memory computing, forming the basis of Apache Spark. Their work was published at the USENIX NSDI conference and has become pivotal in Big Data processing.

8. **Gittens and Eaton (2012)**, in their IBM Press publication *Understanding Big Data*, examine the practical applications of Hadoop and streaming analytics for enterprise-scale systems, bridging the gap between theoretical and applied data science.

Dupli Checker   Date: 06-04-2025

Plagiarism Scan Report

0% Plagiarism	0% Exact Match	**100%** Unique	Words	465
	0% Partial Match		Characters	3042
			Sentences	32
			Paragraphs	16
			Read Time	3 minute(s)
			Speak Time	4 minute(s)

Content Checked For Plagiarism

Introduction

The demand for effective tools for data processing and analysis has increased as a result of the rapid growth of digital data. Due to limitations in processing power and storage, traditional computing methods frequently have trouble processing large datasets. Using its MapReduce programming model, the open-source framework Hadoop addresses this issue by making distributed computing possible. Since this model makes it possible to process numerous large datasets simultaneously, it is an essential tool for big data analytics. The Word Count problem is a fundamental example of the MapReduce paradigm. A Word Count program is used to process a structured sales dataset in this project. Each record contains product details and other relevant attributes and represents a transaction. The program takes individual words from the