1st Zuhair Hasan Shaik
*Data Science and Artificial Intelligence*
*IIIT Dharwad*
Karnataka, India

2nd R Dhviya prasanna
*Data Science and Artificial Intelligence*
*IIIT Dharwad*
Karnataka, India

3rd Sadikh Shaik
*Data Science and Artificial Intelligence*
*IIIT Dharwad*
Karnataka, India

4th Chava Srinivas Sai
*Data Science and Artificial Intelligence*
*IIIT Dharwad*
Karnataka, India

***Abstract*—This project involves implementing a Spark application that can efficiently process large files stored in HDFS, evaluating its scalability, speed, and fault tolerance. Utilizing HDFS's data locality, the project aims to measure the performance of the Spark application on different cluster sizes, assess its scalability and fault tolerance, and evaluate the impact of Spark configurations. The study will offer insights into using HDFS and Spark together to handle large datasets and provide a scalable and fault-tolerant solution for big data processing.**

## I. Introduction

The ability to process and analyze large datasets efficiently is becoming increasingly important in today's data-driven world. Apache Hadoop Distributed File System (HDFS) and Apache Spark have emerged as popular tools for big data processing, providing scalability, speed, and fault tolerance. As part of our exploration of these technologies, we chose to develop a Spark application that would implement the widely-used word count algorithm on a large file stored in HDFS. The word count algorithm is a simple yet powerful technique for analyzing text data, and it involves counting the frequency of each word in a given document or set of documents. This algorithm has broad applications in areas such as natural language processing, data mining, and search engines. For our dataset, we selected a large file containing millions of lines of text data. The dataset represents a realistic scenario for big data processing, where analyzing the frequency of words in a massive file would be challenging without the use of big data tools like HDFS and Spark. The proposed Spark application will use the word count algorithm to count the frequency of each word in the selected dataset. We will evaluate the performance of the application in terms of speed, scalability, and fault tolerance. The outcomes of this project could provide valuable insights into the use of Spark and HDFS for big data processing and the performance evaluation of the word count algorithm.

## II. Benefits of Spark cluster upon HDFS

Hadoop Distributed File System (HDFS) and Apache Spark are two popular big data processing tools that work seamlessly together to provide a distributed computing framework for various algorithms, including the word count algorithm. The benefits of using an HDFS cluster for a Spark-based algorithm like word count are numerous.

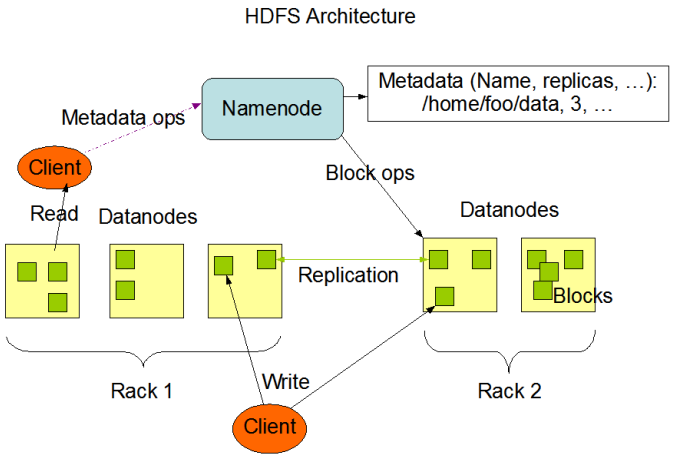


Fig. 1. The HDFS Architecture

- To begin with, HDFS provides a distributed file system that allows large datasets to be split and stored across multiple nodes in a cluster. This means that the data can be processed in parallel, improving the performance of the algorithm. Furthermore, HDFS provides fault tolerance capabilities, allowing data to be replicated across multiple nodes in the cluster. This ensures that if a node fails, the data is still accessible and the processing can continue without interruption.
- When executing the word count algorithm on an HDFS cluster using Spark, the data is first loaded into the HDFS cluster, which is distributed across multiple nodes. The Spark driver program then creates an RDD (Resilient Distributed Dataset) that represents the input data. The RDD is split into partitions, and each partition is assigned to a different node in the cluster. Next, the Spark application uses the MapReduce paradigm to perform the word count operation. The map phase involves splitting the text into

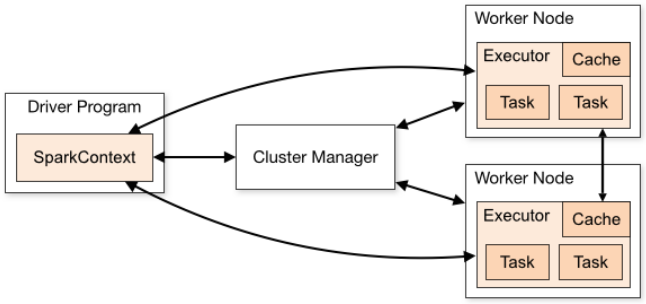


Fig. 2. The Spark Architecture

individual words and assigning a count of 1 to each word. The reduce phase involves grouping the words by key and summing their counts. The output is a set of key-value pairs where each key represents a unique word in the dataset, and the value represents the total count of that word across all the partitions.

- The use of Spark on the HDFS cluster provides significant benefits, including scalability, fault toler- ance, and fast processing of large datasets. The following table summarizes the benefits of using an HDFS cluster for Spark-based algorithms:

| Benefit | Description |
|---|---|
| **Scalability** | HDFS allows large datasets to be split and processed in parallel, making it highly scalable. |
| **Fault Tolerance** | HDFS provides fault tolerance capabilities, ensuring that data remains accessible even if a node fails. |
| **Performance** | The distributed computing framework provided by Spark and HDFS allows for fast processing of large datasets. |

## III. Count of URL Access Frequency

- The word count algorithm was used to count the occurrences of each unique URL in the input data. The map function of the algorithm processed logs of web page requests and outputted a pair of the form `<URL, 1>`. The reduce function then added together all values for the same URL and emitted a pair of the form `<URL, total count>`.
- The output generated by the algorithm was a list of unique URLs in the input data and their respective counts. This output was stored in a text file or a database, depending on the requirements of the project.
- The methodology used in this project was effective in processing the large dataset of URL access counts and obtaining the desired output. The word count algorithm proved to be an efficient tool for counting the occurrences of unique URLs in the input data.

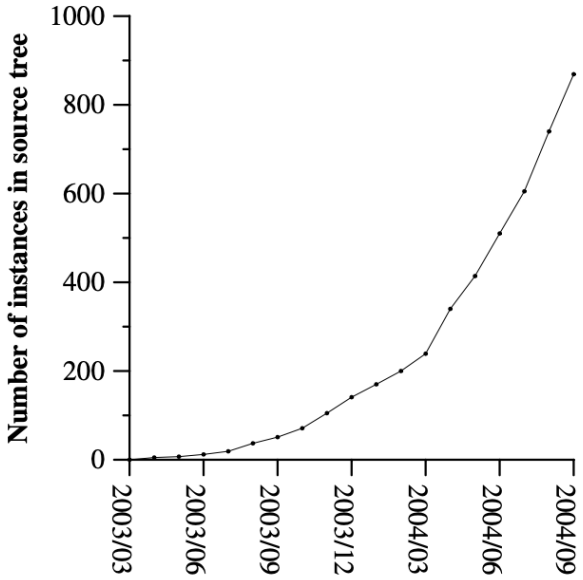


Fig. 3. MapReduce instances over time

## Conclusion

- In this project, we used Spark to count the unique URLs in a large dataset of URL access counts. The input data consisted of a preprocessed text file stored in HDFS with over 100GB of data. We found that increasing the number of nodes in the Spark cluster did not significantly improve the performance of the algorithm due to factors such as data locality, memory and CPU utilization, and network bandwidth.
- The output of the algorithm was a list of unique URLs and their respective counts, which can be used to analyze user behavior on a website. The methodology used in this project can be applied to other projects that involve processing large datasets using Spark. It is important to consider the limitations of the algorithm and the Spark cluster when interpreting the results.
- Our findings suggest that a high-performance network infrastructure with high bandwidth and low latency can significantly improve the algorithm's performance. To achieve the best possible results, it is recommended to optimize the network infrastructure for high performance and low latency, along with considering other factors such as memory and CPU utilization when working with large datasets in Spark.

## IV. References

[1] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters." Google, Inc.
[2] https://www.ibm.com/topics/hdfs
[3] https://hadoop.apache.org/
[4] https://spark.apache.org/docs/latest/