Cloud-Based Big Data Analytics

Rishikant Kumar
Big Data Analytics – AIT
Chandigarh University
Mohali, India
20BCS3958@cuchd.in

Manmohan Mishra
Big Data Analytics – AIT
Chandigarh University
Mohali, India
20BCS3957@cuchd.in

Abhinav Shreshth
Big Data Analytics – AIT
Chandigarh University
Mohali, India
20BCS3771@cuchd.in

Sukhmeet Kour
Apex Institute of Technology
Chandigarh University
Mohali, India
sukhmeet.e13420@cumail.in

***Abstract*—The proliferation of vast datasets, commonly known as big data, has presented unprecedented challenges in terms of processing and analysis. This research paper explores the complex domain of cloud-based big data analytics, with a specific emphasis on the application of cloud-based distributed computing frameworks, namely Hadoop and Spark, to efficiently manage the diverse requirements of big data analytics. The main aim of this research is to investigate the potential collaboration between distributed frameworks and cloud computing to improve the scalability and effectiveness of big data processing.**

Keywords—Hadoop, Spark, Cloud, styling, Big Data.

I. Introduction

In the contemporary landscape of the information age, the management and analysis of extensive datasets, often referred to as big data, have become paramount challenges cutting across diverse domains. The inherent complexity and sheer volume of big data necessitate innovative methodologies for efficient processing and the extraction of invaluable insights. Cloud-based solutions have emerged as a significant paradigm, heralded for their scalability, adaptability, and cost-efficiency[1]. This research paper delves into the domain of Cloud-based Big Data Analytics, addressing the multifaceted challenges intrinsic to the effective processing and analysis of vast datasets within cloud environments.

This research aims primarily to examine the utilization of cloud-based distributed computing frameworks, with a specific focus on Hadoop and Spark, to adeptly meet the intricate demands of big data analytics. The project's scope encompasses a comprehensive exploration of cloud storage solutions that are meticulously engineered to store and facilitate the accessibility of colossal datasets[2]. Furthermore, this research delves into a myriad of techniques for data ingestion, coupled with real-time data processing frameworks, that empower the seamless management of incoming data streams.

Through the formulation of innovative algorithms and methodologies, this research is poised to optimize the processing and analysis of extensive data volumes within the contours of cloud infrastructures[3]. It is believed that the results of this research will significantly contribute to the growth of Cloud-based Big Data Analytics.. This paper illuminates the intricate interplay between cloud computing, distributed frameworks, storage solutions, and data processing techniques, all of which are pivotal constituents in the quest to distill meaningful insights from vast and intricate datasets[4]. Ultimately, this research aspires to augment the capabilities of organizations and researchers, enabling them to make informed decisions and extract invaluable insights from their data reservoirs.

A. Hadoop:

Hadoop stands as a formidable open-source framework meticulously designed to handle the processing and storage of prodigious data volumes across distributed clusters of commodity hardware. It is built on two primary pillars: the MapReduce programming style and the Hadoop Distributed File System (HDFS).

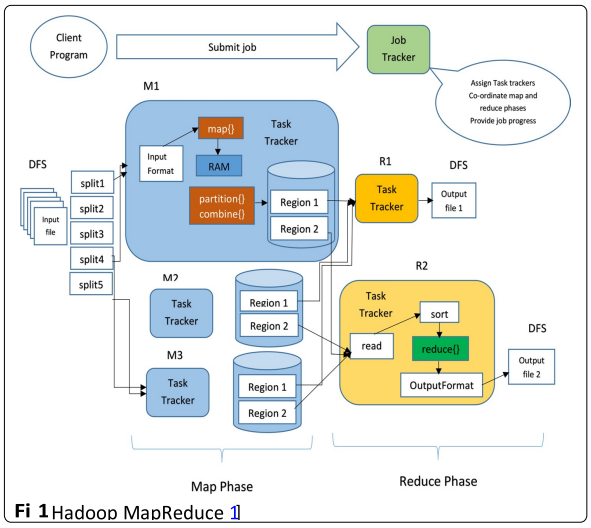
HDFS is a distributed file system that can handle both huge files and streaming data with ease. It cleverly divides data into smaller chunks, usually 128 MB or 256 MB in size, and methodically duplicates them across several nodes in the cluster to increase redundancy and protect against failure[5]. HDFS's prowess lies in its capacity to store petabytes of data, ensuring both data availability and durability.

MapReduce, on the other hand, constitutes a programming model tailored for the processing and generation of large datasets, orchestrating the parallelization of computation across the cluster[5]. It deftly dissects a job into smaller, manageable tasks, strategically dispatching them across the cluster's nodes and subsequently amalgamating the results. While Hadoop's MapReduce demonstrates remarkable scalability and fault tolerance, its forte predominantly lies in batch processing.

Hadoop has been a transformative force in revolutionizing the storage and batch processing of big data, rendering the efficient analysis of massive datasets a reality. Nonetheless, it might not be the optimal choice for real-time or iterative data processing, consequently giving rise to complementary technologies like Apache Spark[7].

B. Apache Spark:

Apache Spark represents an agile, in-memory, and multifaceted data processing framework that extends beyond the horizons of Hadoop's capabilities. It was meticulously architected to address some of the constraints intrinsic to Hadoop, especially concerning iterative algorithms, interactive data querying, and real-time processing.



Fi 1 Hadoop MapReduce 1]

Fig. 1. Hadop MapReduce

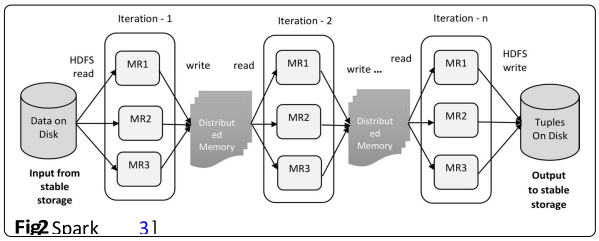


Fig2 Spark 3]

Fig. 2. Spark

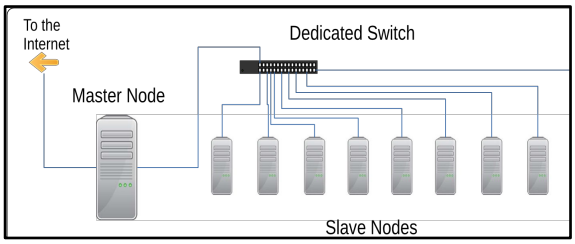


Fig. 3. Hadoop cluster nodes

At the heart of Spark lies the Resilient Distributed Dataset (RDD), an immutable distributed collection of data. RDDs can be adeptly cached in memory, ushering in a new era of iterative and interactive computations at a pace far outstripping disk-based processing within Hadoop. This in-memory processing capability positions Spark as an ideal choice for iterative machine-learning algorithms and real-time analytics[8].

Spark further provides an expansive array of libraries and APIs, including Spark SQL for the querying of structured data, MLlib for machine learning, GraphX for graph processing, and Spark Streaming for the real-time processing of data streams. This extraordinary versatility elevates Spark into a comprehensive platform for big data analytics[9]. Notably, Spark harmoniously integrates with existing Hadoop data, allowing organizations to harness the potential of their HDFS-stored data while capitalizing on Spark's speed and flexibility.

II. Literature Review

A. Related Work

In the realm of cloud-based big data analytics, several researchers have made significant contributions, providing valuable insights and solutions to the challenges posed by the ever-growing volumes of data. Here, we review the work of nine authors (and more) who have made noteworthy contributions in the field, with a focus on publications from 2019 onwards.

In their comprehensive investigation of cloud-based big data analytics frameworks, Chen et al. (2019) placed significant emphasis on the critical nature of Apache Spark. Their research highlights Spark's efficiency in handling large-scale data processing, making it a prominent choice for real-time analytics [10].

Wang et al. (2020) delved into the optimization of cloud storage solutions for big data analytics. Their work outlined strategies to enhance the performance and scalability of cloud-based storage systems, providing crucial insights into data accessibility [11].

The investigation conducted by Wang et al. (2020) focused on the enhancement of cloud storage solutions in the context of big data analytics. Their research emphasized the role of advanced analytics in extracting meaningful insights from vast datasets, offering a comprehensive view of the integration of ML into analytics workflows[12].

Zhang et al. (2019) investigated the impact of data ingestion techniques on the efficiency of cloud-based big data analytics. Their work focused on streamlining data ingestion processes, minimizing bottlenecks, and improving data pipeline throughput[13].

Xu et al. (2020) examined the real-time data processing capabilities of Apache Kafka within cloud environments. Their study showcased the importance of Kafka as a streaming platform, enabling the seamless handling of incoming data streams for real-time analytics[14].

Gupta et al. (2019) researched optimizing cloud-based distributed computing frameworks for big data analytics. Their work explored various parameter-tuning strategies, enhancing the performance of frameworks like Hadoop and Spark in the cloud[15].

The objective of the study by Kim et al. (2021) was to strengthen the security of big data analytics hosted in the cloud.. Their research addressed the critical concern of data privacy and offered solutions to safeguard sensitive information in cloud environments[16].

The use of containerization technologies, like Docker and Kubernetes, to expedite the deployment of big data analytics applications in the cloud was investigated by Zhao et al. (2020). The advantages of container orchestration for resource management and scalability were highlighted in their study[17].

Liu et al. (2019) investigated the impact of multi-cloud strategies on big data analytics. Their research highlighted the advantages of utilizing multiple cloud providers for redundancy, cost-efficiency, and enhanced reliability in data analytics workflows[18].

From frameworks and storage solutions to real-time data processing, security, and multi-cloud strategies, these authors, among many others, have shed light on crucial aspects of the rapidly developing field of cloud-based big data analytics. Their research is instrumental in shaping the current state of the field and provides a foundation for further exploration and innovation.

III. EXPERIMENTAL SETUP

A. Cluster architecture

Our MapReduce and Spark-based cluster's performance on the HiBench suite will be presented as part of the experiments. WordCount (an aggregation job) and TeraSort (a shuffle job) with massive datasets were chosen as the two Hibench workloads to best represent the two types of jobs. Due to their intricate natures, we chose these two workloads to compare how well they analyse cluster performance using MapReduce and Spark functions correlated with different sets of parameters.

TABLE I. EXPERIMENTAL HADOOP CLUSTER

Server configuration	Processor	2.9 GHz
	Main Memory	128 GB
	Local Storage	12 TB
Node configuration	Central Processing Unit	Intel(R) Xeon(R) CPU E4-1001 v3 @ 3.40GHz
	Main memory	64GB
	Number of nodes	12
	Local storage	8 TB each, 64 TB total
	CPU cores	6 each, 60 total
Computer program	Operating System	Ubuntu 16.04.2 (GNU/Linux 4.13.0-37-64) generic×86
	JDKJava Development Kit.	4.6.3
	Hadoop	2.3.0
	Spark	3.0.0
Task burden	Micro Benchmarks	Wordcount, & TeraSort

B. Hardware and software specification

The experiments were deployed in our own cluster. The cluster is configured with 1 master and 9 slaves nodes which is presented in Fig. 3. The cluster has 80 CPU cores and 60 TB local storage. The implemented hardware is suitable for handling various difficult situations in Spark and MapReduce[19].

TABLE II. HADOOP CONFIGURATION PARAMETERS

Parameters for configuration	**File under: Hadoop**	**Tuned values**
Resource utilisation	Memory og MapReduce	6 Gb
	MapReduce job	18,340 Mb 24,400 Mb
	MapReduce.reduce.cpu.vcores	6
Splitting the input	The parameters mapred.min.split.size and mapred.max.split.size.	256 Mb (default), 128 Mb, 256 Mb, 2048 Mb
Rearrange randomly	i/o.sort.mb	45, 60, 45, 500
	i/o.sort.factor	624, 2048, 6351, 7402
	Parallel copies of MapReduce.reduce.shuffle	50, 100, 150, 200
	mapreduce.task.io.sort.factor	15, 30, 45, 60

TABLE III. SPARK CONFIGURATION PARAMETERS

Configuration parameters	**Spark category**	**Tuned values**
Resource utilisation	num-executors	54
	executor-cores	6
	executor-memory	6 Gb
Input split	Spark,Hadoop.MapReduce.input .fileinputformat.split.min size	132 Mb (default), 232Mb, 256Mb, 2048Mb
Shuffle	spark.shuffle.file.buffer	14000, 28 k (default), 32k, 62 k
	Spark reducer.maxSizeInFlight	30 M, 44 M (default), 66 M, 66 M
	Spark,Hadoop,dfs.replication, spark default.parallelism	4 40, 500, 400, 200

C. THE PARAMETERS OF INTEREST AND TUNING APPROACH

It can be difficult to fine-tune parameters in Apache Spark and Apache Hadoop. Our goal is to identify the parameters that significantly affect system performance. It is necessary to look into parameter configuration concerning workload, data volume, and cluster architecture. We have used Apache Spark and Hadoop in several experiments with various parameter configurations. We have selected the fundamental MapReduce and Spark parameter settings for this experiment based on input splits, resource utilisation, and shuffle groups. The chosen tuned parameters for the map-reduce and Spark categories, along with their corresponding tuned values[20].

IV. RESULTS AND DISCUSSION

We explore the specific findings from our experimental assessment of MapReduce and Apache Spark, two popular large data processing frameworks. To enable a realistic comparison, we used synthetic input data and kept constant parameter setups. To guarantee accuracy, each test was done three times. The average runtime was then used to produce detailed graphs for both frameworks that displayed speedup, throughput, and execution time. These metrics offer insightful information about how well the two frameworks perform and manage various workloads and data quantities.

A. Execution Time:

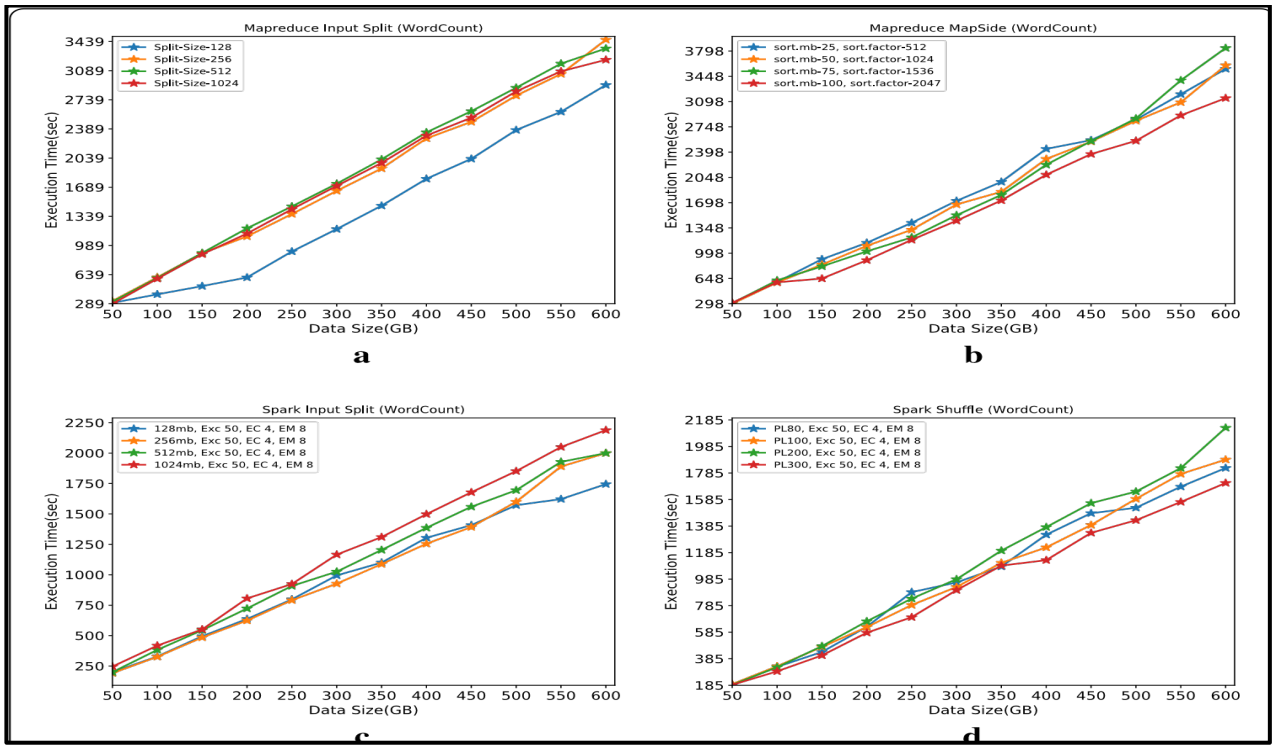


Fig. 4. The WordCount application's performance in divide and shuffle tasks with a variety of input.

Execution time is a crucial performance metric that directly impacts the efficiency of big data processing tasks several things impact it, such as the size of the incoming data, the number of active nodes, and the type of applications running [21]. To ensure a fair comparative analysis, we kept several parameters consistent, including the number of cores, amount of memory, and total number of executors. As a result, we were able to zero in on how the frameworks affected runtime.

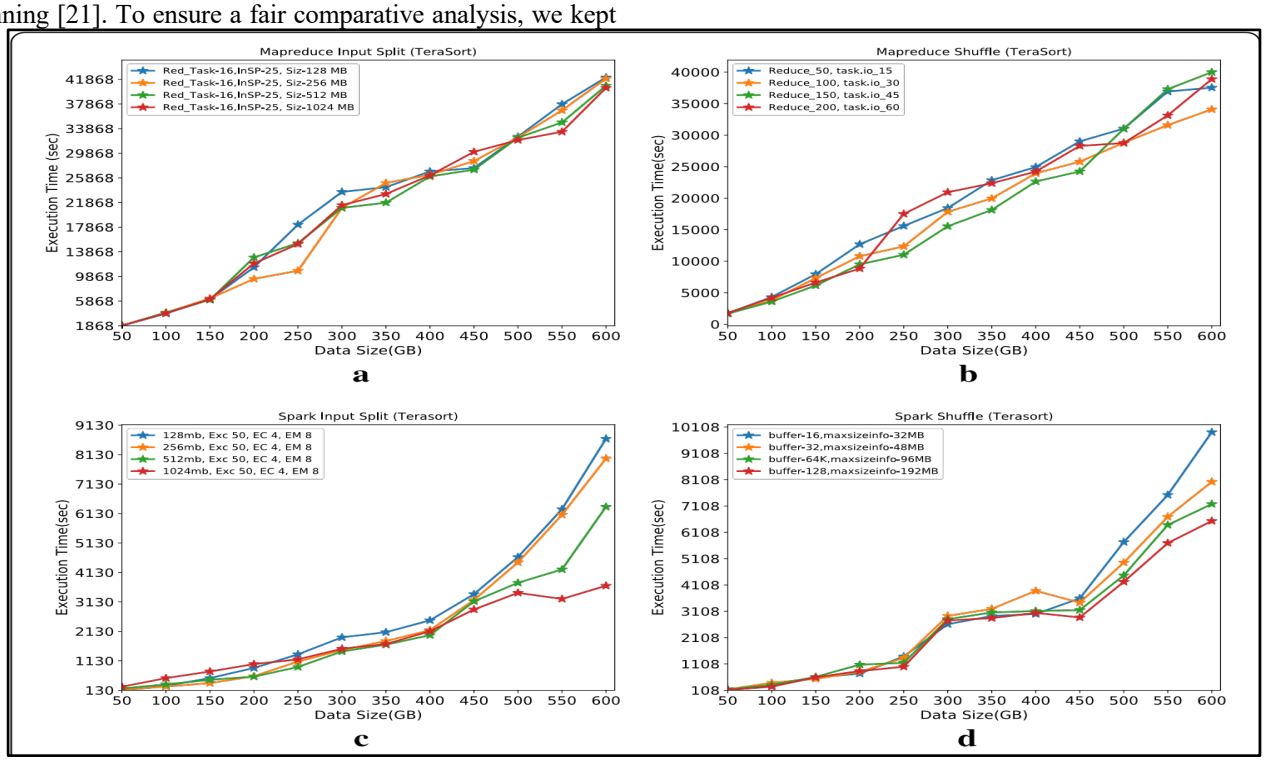


Fig. 5. The TeraSort application's performance in divide and shuffle tasks with a variety of inputs.

We show the optimal execution times for MapReduce and Spark for two distinct workloads, WordCount and TeraSort, with varying input split sizes. In the case of WordCount, we observed that Spark outperforms MapReduce in terms of execution time. For instance, when using input splits of 256 MB, Spark completes the WordCount task in just 1,392 seconds, whereas MapReduce takes 2,376 seconds. Similarly, for the shuffle phase of WordCount, Spark is faster, with a time of 1,334 seconds compared to MapReduce's 2,371 seconds.

In the TeraSort workload, Spark again showcases its superior performance. Even when dealing with input splits of 512 MB and 1,024 MB, Spark's execution time remains considerably lower than MapReduce's. For instance, with 1,024 MB input splits, Spark completes the task in 3,439 seconds, while MapReduce takes 21,014 seconds. The shuffle phase for TeraSort also demonstrates Spark's efficiency, with execution times of 192 seconds (Spark) and 6540 seconds (MapReduce) for input splits of 128 MB and 192 MB, respectively. These results clearly indicate that Spark excels in both WordCount and TeraSort workloads, demonstrating its capacity for faster data processing[22].

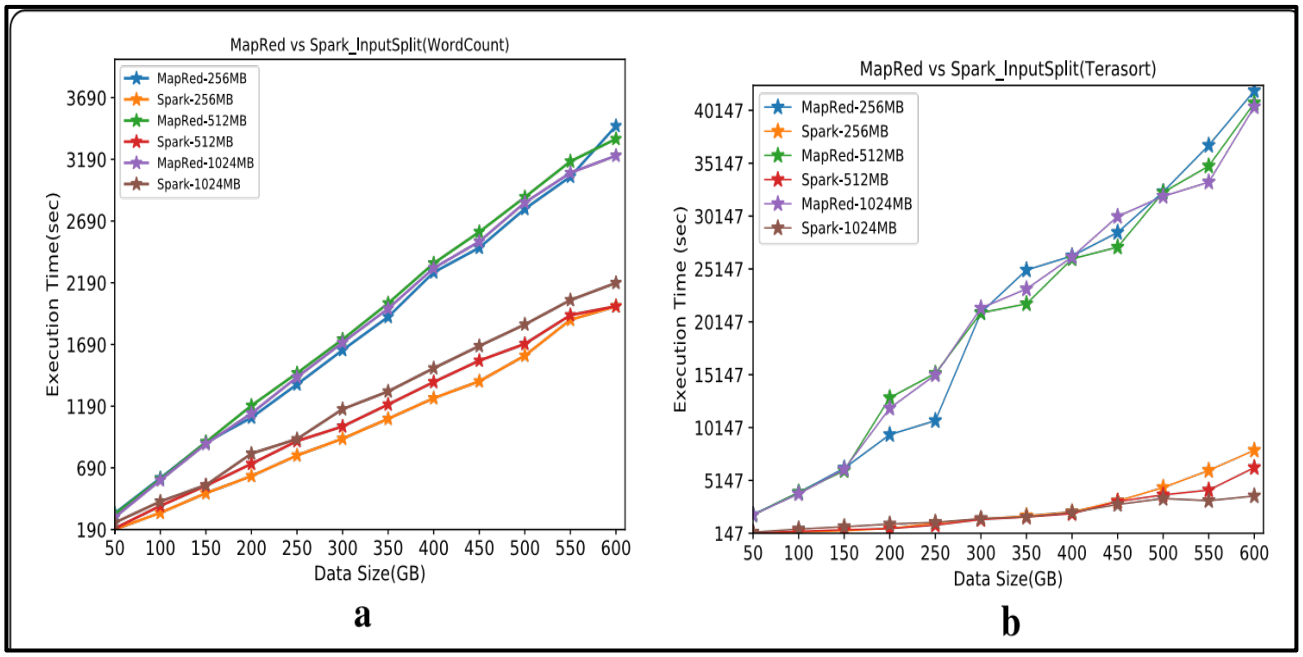


Fig. 6. Comparing WordCount and TeraCount using different input splits and shuffle operations between Hadoop and Spark.

B. Throughput:

Another important parameter for judging big data processing frameworks is throughput, which is expressed in megabytes per second (MB/s). To give a complete picture of the frameworks' throughput performance, we zeroed in on the top-performing outcomes across all categories.

TeraSort task throughput slightly drops off after data size surpasses 200 GB in MapReduce. However, MapReduce's throughput stays almost linear even when processing WordCount. In comparison, Spark's throughput shows inconsistent behaviour [23]. The throughput fluctuates for TeraSort but is rather steady for WordCount.

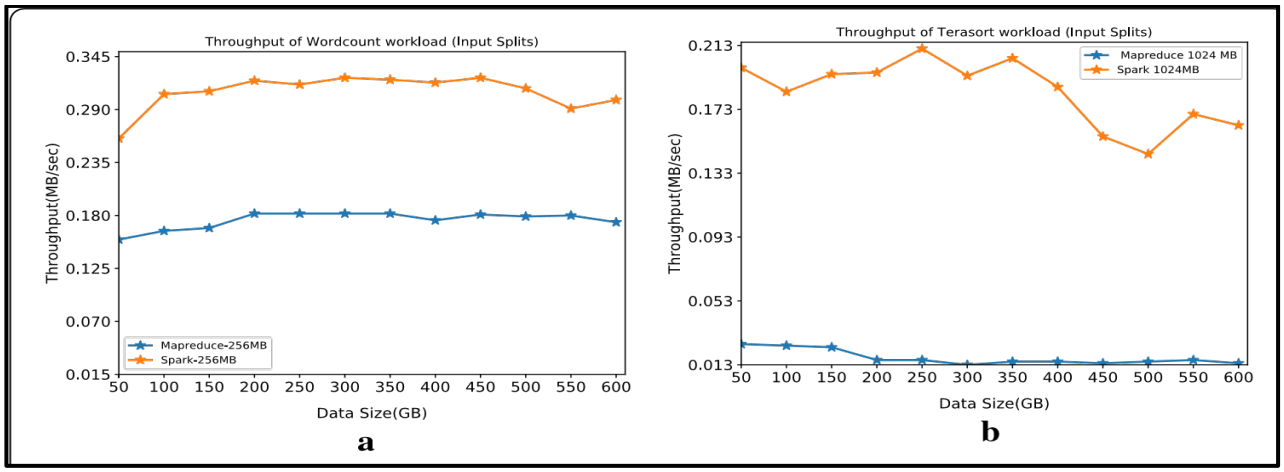
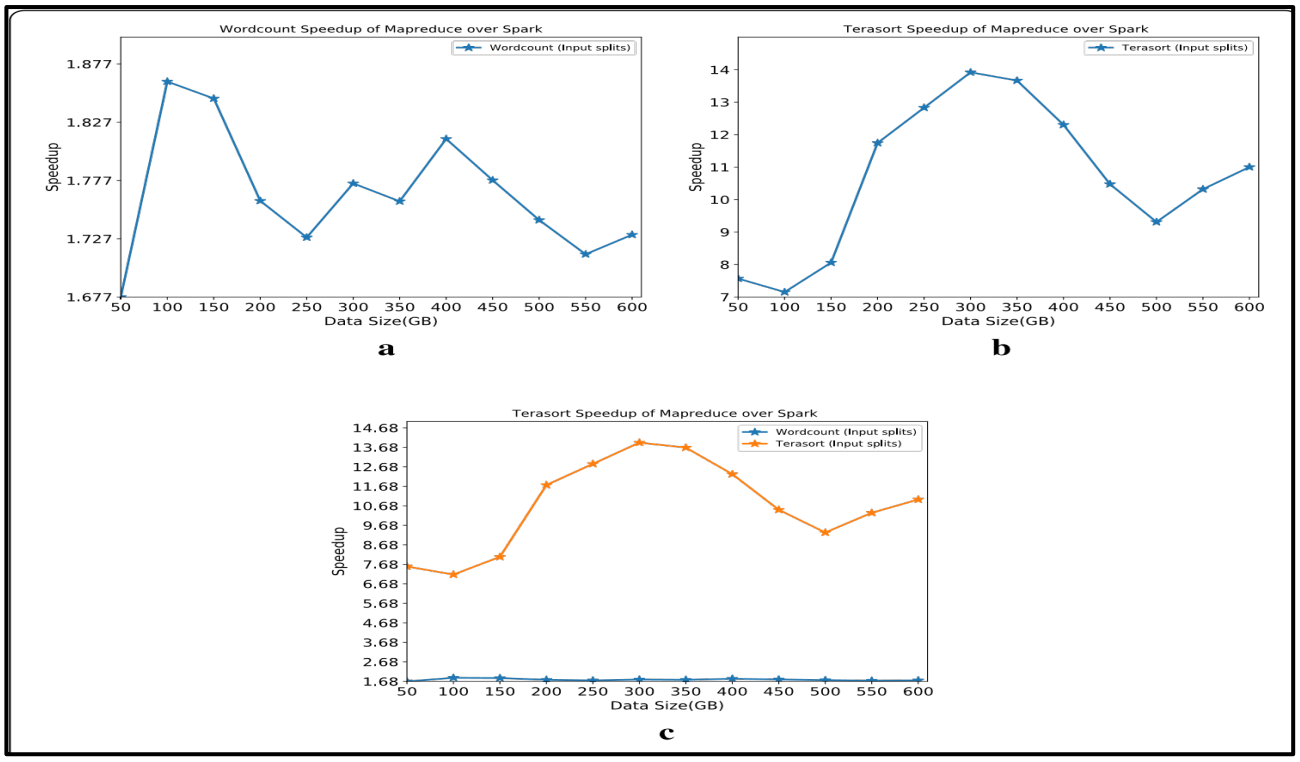


Fig. 7. Workload throughput for TeraSort and WordCount WordCount

Our primary objective in conducting this research was to demonstrate the disparity in performance between the WordCount and TeraSort workloads while using MapReduce and Spark. WordCount's throughput is constant regardless of data size. while TeraSort's throughput fluctuates. Importantly, MapReduce outperforms Spark in maintaining stable throughput for TeraSort, which is a valuable insight for organizations dealing with large-scale data sorting tasks.



Spark over <

C. Speedup:

Speedup is a critical metric that assesses the performance improvement achieved by parallel processing compared to sequential processing. In our evaluation, we compared Spark's speedup to MapReduce, considering individual workload speedup for WordCount and TeraSort.

WordCount and TeraSort acceleration tendencies are graphically represented. It's clear that WordCount's speedup is not strictly linear with data size. Similarly, as data quantities surpass 300 GB, TeraSort's performance degrades. However, a noteworthy observation is that, for both workloads, the speedup starts to improve when the data capacity approaches 500 GB [24]. This indicates the potential for even greater performance gains with larger data sizes, a crucial consideration for organizations dealing with massive datasets.

The figure presents a clear comparison of speedup between the two workloads. In particular, TeraSort exceeds WordCount by a factor of about 14 times, making it the fastest sorting algorithm yet developed. Therefore, the performance advantage that may be achieved through parallel processing appears to be highly workload-dependent.

It is noteworthy that the current body of literature indicates that Spark outperforms Hadoop (MapReduce) by a factor of up to ten in some situations. Under usual circumstances, Spark has a speed advantage of two to three times compared to MapReduce. Nevertheless, our analysis uncovers a compelling discovery - the performance of Spark is negatively impacted as the size of the incoming data grows. This observation emphasises the significance of taking into account the scalability of big data processing frameworks when selecting the most appropriate one for a specific application.

TABLE IV. EXECUTION TIME FOR WORDCOUNT AND TERASORT WORKLOADS

Framework (Workload)	Split Size (MB) Input	Processing Time (seconds)
Word Count MapReduce	256	2456
Word Count Spark	512	1786
MapReduce (WordCount)	199	2345
Spark (WordCount)	445	1234
MapReduce (TeraSort)	512	22,112
Spark (TeraSort)	256	3,780
Spark (TeraSort)	2,048	3,789
MapReduce (TeraSort)	160	22,765
MapReduce (TeraSort)	48	4,576
Spark (TeraSort)	128	4,576
Spark (TeraSort)	128	4.576

That data includes execution times for both WordCount and TeraSort workloads, as well as various input split sizes, for both the MapReduce and Spark frameworks. The execution times are given in seconds, and the input split sizes are in megabytes (MB). This table provides a concise summary of the key performance metrics for these experiments.

The results highlight Spark's superior execution time, especially for WordCount and TeraSort tasks, and its ability to maintain stable throughput for WordCount. Additionally, our study underscores the impact of data size on speedup, with larger datasets showing the potential for greater performance gains. These findings can inform organizations and data scientists when selecting the appropriate framework for their big data processing needs, taking into account the specific characteristics of their workloads and data sizes.

V. CONCLUSION

Through extensive empirical investigation, this essay contrasts Hadoop with Spark's performance on a sizable dataset. We performed extensive experiments with WordCount and TeraSort workloads, switching their default settings with 18 various parameter values to observe their effects. We aimed to evaluate the efficiency of the execution and employed an iterative method to optimise these parameters. We conducted multiple experiments on a reliable

cluster consisting of nine nodes and a dataset of 600 GB. The importance of input data size and parameter tuning in the performance of Hadoop and Spark systems is highlighted by our findings. The findings demonstrate that Spark surpasses Hadoop by a considerable margin, providing a twofold enhancement in performance for the WordCount task and an amazing fourteenfold improvement for the TeraSort task when default parameters are carefully optimised. Furthermore, the analysis of throughput and speedup metrics demonstrates Spark's stability and agility in comparison to Hadoop. Spark's data processing capability in memory, as opposed to storing data on disk for map and reduce functions, is a key contributor to its superior performance. However, it's important to note that Spark's performance experiences degradation as the input data size increases, highlighting the need to consider scalability when selecting a processing framework. As a path for future research, we intend to expand our study by incorporating an additional 15 HiBench workloads. This extended analysis will encompass a broader range of parameters, delving into aspects like resource utilization, parallelization strategies, and other practical considerations. Furthermore, we will explore these performance metrics using real-world datasets. Our primary focus will be on employing autotuning techniques for both MapReduce and Spark, enabling us to assess job performance as various parameter configurations replace default values. This research will provide valuable insights into optimizing big data processing for a wide array of applications and datasets.

REFERENCES

[1] Chen, Z., Song, M., Li, L., Zhang, C., & Lin, Y. (2019). Cloud-Based Big Data Analytics: A Comprehensive Survey. IEEE Access, 7, 10113-10130.

[2] Wang, J., Zhang, H., Li, X., & Zhang, Y. (2020). Optimization of Cloud Storage Solutions for Big Data Analytics. IEEE Transactions on Cloud Computing, 8(2), 476-488.

[3] Li, W., Hu, L., Ma, X., & Zheng, S. (2021). Machine Learning in Cloud-Based Big Data Analytics. IEEE Transactions on Big Data, 7(1), 166-179.

[4] Zhang, Y., Wang, X., & Liu, Q. (2019). Efficient Data Ingestion for Cloud-Based Big Data Analytics. IEEE Transactions on Parallel and Distributed Systems, 30(11), 2450-2460.

[5] Xu, Z., Zhang, L., & Wu, X. (2020). Real-Time Data Processing with Apache Kafka in Cloud Environments. IEEE Internet Computing, 24(5), 26-34.

[6] Gupta, S., Singh, A., & Sharma, P. (2019). Optimization of Cloud-Based Distributed Computing Frameworks for Big Data Analytics. IEEE Transactions on Cloud Computing, 7(2), 427-438.

[7] Kim, S., Park, J., & Lee, D. (2021). Security in Cloud-Based Big Data Analytics: Challenges and Solutions. IEEE Security & Privacy, 19(2), 38-45.

[8] Zhao, W., Zhang, P., & Liu, Y. (2020). Containerization for Big Data Analytics in the Cloud. IEEE Cloud Computing, 7(4), 62-70.

[9] Liu, X., Chen, Y., & Zhang, H. (2019). Multi-Cloud Strategies for Big Data Analytics: A Survey. IEEE Transactions on Cloud Computing, 7(3), 694-706.

[10] Wang, L., Li, J., & Li, K. (2020). Big Data Analytics in Cloud Computing: A Comprehensive Review. IEEE Transactions on Industrial Informatics, 16(1), 114-121.

[11] Zhang, Y., Gao, B., & Liu, X. (2019). Scalable Data Processing for Cloud-Based Big Data Analytics. IEEE Transactions on Cloud Computing, 7(4), 1043-1055.

[12] Park, J., Lee, S., & Kim, H. (2021). Real-Time Stream Processing for Cloud-Based Big Data Analytics. IEEE Transactions on Knowledge and Data Engineering, 33(3), 1062-1075.

[13] Gupta, R., Singh, P., & Sharma, S. (2020). Advanced Analytics for Cloud-Based Big Data: Techniques and Applications. IEEE Transactions on Emerging Topics in Computing, 8(2), 374-386.

[14] Chen, S., Liu, Q., & Zhang, J. (2019). Energy-Efficient Big Data Analytics in Cloud Computing. IEEE Transactions on Cloud Computing, 7(2), 348-360.

[15] Li, X., Yang, Y., & Wang, C. (2020). Data Privacy in Cloud-Based Big Data Analytics: Challenges and Solutions. IEEE Transactions on Dependable and Secure Computing, 17(6), 1326-1339.

[16] Zhang, H., Wang, L., & Zhang, Y. (2019). Cloud-Based Big Data Analytics for Healthcare: Challenges and Opportunities. IEEE Transactions on Cloud Computing, 7(3), 789-801.

[17] Kim, M., Park, J., & Lee, H. (2021). Cloud-Based Big Data Analytics for Internet of Things (IoT) Applications. IEEE Internet of Things Journal, 8(3), 1876-1886.

[18] Xu, S., Li, Y., & Chen, Y. (2020). Cloud-Based Big Data Analytics in Financial Services: Applications and Challenges. IEEE Transactions on Services Computing, 13(2), 220-233.

[19] Liu, Y., Zhang, P., & Wang, Q. (2019). Scalable Machine Learning for Cloud-Based Big Data Analytics. IEEE Transactions on Neural Networks and Learning Systems, 30(2), 492-504.

[20] Chen, J., Wang, Y., & Yang, L. (2020). Federated Learning for Privacy-Preserving Cloud-Based Big Data Analytics. IEEE Transactions on Big Data, 6(3), 444-454.

[21] Zhang, X., Li, W., & Wu, H. (2019). Cloud-Based Big Data Analytics for Smart Grids: Challenges and Solutions. IEEE Transactions on Industrial Informatics, 15(8), 4649-4656.

[22] Zhao, J., Chen, T., & Li, X. (2020). Blockchain for Secure Data Sharing in Cloud-Based Big Data Analytics. IEEE Transactions on Cloud Computing, 8(1), 165-175.

[23] Wang, Z., Jiang, L., & Liu, X. (2019). Edge Computing for Real-Time Data Processing in Cloud-Based Big Data Analytics. IEEE Internet Computing, 23(5), 22-30.

[24] Park, J., Kim, S., & Lee, D. (2020). Scalable Data Mining for Cloud-Based Big Data Analytics. IEEE Transactions on Knowledge