Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 18 No. 17 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijim.v18i17.44555

PAPER

From Micro-benchmarks to Machine Learning: Unveiling the Efficiency and Scalability of Hadoop and Spark

Salah Eddine Hebabaze[1](✉), Mohamed El Ghmary[2], Hamid El Bouabidi[1], Sara Maftah[1], Mohamed Amnai[1]

[1]Ibn Tofaïl University, Kenitra, Morocco

[2]Sidi Mohamed Ben Abdellah University, Fez, Morocco

Salaheddine.Hebabaze@uit.ac.ma

ABSTRACT

With the exponential growth of data, the demand for efficient and scalable data processing solutions has become paramount. Hadoop and Spark, pivotal components of the open-source Big Data landscape, have been put to the test in this study. We conducted a comprehensive performance analysis of Hadoop and Spark in virtualized environments, evaluating their prowess across a suite of benchmarks. The benchmarks encompassed a spectrum of workloads, from micro-benchmarks such as Sort, WordCount, and TeraSort to web search tasks such as PageRank and machine learning endeavors including Naive Bayes and K-means. The central focus was to gauge their performance, efficiency, and resource utilization. The findings of this study underscore the benefits of Spark's in-memory processing, demonstrating its superiority over Hadoop in various scenarios. Spark excels in machine learning and web search applications, particularly when handling smaller inputs. Its efficient memory management and support for multiple iterations make it a strong choice. In resource-constrained environments or when dealing with large input files and limited memory, Hadoop may still hold an edge. The design and implementation of data processing solutions in virtualized environments should carefully consider the specific demands of each framework. This study not only presents a performance comparison of Hadoop and Spark across different benchmarks but also emphasizes the vital implications for designing and deploying data processing solutions in virtualized settings. It serves as a cornerstone for informed decision-making, paving the way for optimized algorithms and techniques in the dynamic landscape of big data processing.

KEYWORDS

big data, Hadoop, Apache Spark, MapReduce, HiBench benchmark, machine learning, memory resource limitations, data workloads

1 INTRODUCTION

Nowadays, traditional data management systems can't process the data's complexity because of its size, structure, and limited processing time [1]. Massive and

Hebabaze, S.E., El Ghmary, M., El Bouabidi, H., Maftah, S., Amnai, M. (2024). From Micro-benchmarks to Machine Learning: Unveiling the Efficiency and Scalability of Hadoop and Spark. *International Journal of Interactive Mobile Technologies (iJIM)*, 18(17), pp. 46–60. https://doi.org/10.3991/ijim.v18i17.44555

Article submitted 2023-09-08. Revision uploaded 2024-06-24. Final acceptance 2024-06-29.

complex data, known as "Big Data," requires new approaches such as Hadoop and Spark. Hadoop is batch-oriented, while Spark is in-memory and real-time, making it faster and more versatile [2]. This paper compares Hadoop and Spark, two popular data processing frameworks, highlighting their strengths and weaknesses. It emphasizes the importance of selecting the right framework for specific tasks. The paper also contributes to the ongoing discussion among professionals and organizations about improving web-based mobile learning in educational contexts by providing insights into learner behavior, identifying patterns, and predicting outcomes [3] [4].

2 RELATED WORKS

Samadi et al. [5] compared the performance of Spark and Hadoop on virtual machines and found that Spark performs better across all workloads using fewer CPU resources. Vettriselvi et al. [6] compare the performance of Spark and Hadoop. Their empirical results indicate that Spark execution time and throughput are better than Hadoop. Lagwankar et al. [7] investigated Hadoop and Spark micro-benchmarks for clustering applications. They found that the behavior of these micro-benchmarks is affected by the size, pattern, type, and source of the input datasets. Hedjazi et al. [8] compared real-time processing frameworks to batch-processing Hadoop frameworks when working on a large-scale classification project. The authors emphasized that although Hadoop is not an ideal solution when low latency is necessary, Apache Spark performs well in iterative processing. Jinbae Lee et al. [9] showed that using Hadoop and Spark schemes enhanced the precision of resource provisioning and job scheduling in large-scale data processing. Based on a wordcount job, Amritpal Singh et al. [10] concluded that Spark is the best choice for stream processing while Hadoop is better for batch processing. Safa et al. [11] found that Spark outperformed Hadoop in terms of several important performance characteristics, including processing time, CPU usage, latency, execution time, job performance, and scalability for non-real-time data. Based on their experimental analysis, Ketu et al. [12] concluded that Hadoop, an on-disk computation-based model, performs worse than Spark, an in-memory computation-based model. Tkdogan et al. [13] found that Spark is approximately five times faster in the training phase, and Hadoop consistently exhibits better classification accuracy, especially with larger input workloads. The authors of [14] suggest that the performance of Hadoop and Spark is influenced by the nature of the tasks. Spark generally outperforms Hadoop due to its in-memory processing, but choosing the right framework depends on specific healthcare applications and data characteristics. Future study should focus on scalability, energy efficiency, cloud benchmarking, and adaptability to changing workloads for a comprehensive understanding of their performance in evolving big data analytics landscapes. In this paper, we compared Hadoop and Spark performance by running six benchmarks to evaluate execution time, speedup, peak and average memory, and CPU usage on both platforms.

3 MAIN CONCEPTS

This section elaborates on the fundamental concepts central to this study, offering a more comprehensive understanding of big data, the architectural disparities between Hadoop and Spark, and the methodologies employed in related study.

3.1 5 V's concept in big data

The challenges that have evolved in the big data environment can be classified as "5V"s [15]. The five Vs must include velocity, volume, and variety, with the addition of veracity and value. Volume [16] means that the data is too large to be handled or stored on a single device. Velocity [15] comes from the need to process massive data quickly and accurately. In the world of big data, tackling real-time processing, such as swiftly handling rapidly expanding data, is a constant challenge often linked with stream-based processing. This is particularly evident in scenarios such as managing a live Twitter data stream, where the need for processing data in real-time becomes crucial [17]. Variety [18] corresponds to the fact that the data can come from various and varied sources, which are heterogeneous and will not all respect the same data format. Veracity [19] is the fact of being able to evaluate whether a piece of data is correct or not to eliminate the data that seems incorrect or inconsistent. Value is the result of extracting new knowledge and new data from existing data.

3.2 Hadoop

Hadoop is one of the big data systems built on the shared-nothing principle in cloud computing [12]. Hadoop is a framework and a set of tools designed to help with the storage and processing of extremely large amounts of data. Indeed, Hadoop [20] is made up of many interconnected parts working together, some of which are mandatory and others are optional. **HDFS,** or Hadoop distributed file system, stores several copies of data blocks across the nodes for improved reliability, faster calculations, and high availability. Data is fetched by MapReduce for further processing [21]. **MapReduce** is a method of processing data. It involves two functions, map and reduce [22]. The first process data by mapping input values to key-value pairs. Then, reduce combines values with the same keys into a single value. **YARN:** The Yet Another Resource Negotiator is a cluster resource manager [7] that performs both JobTracker and Application-Master functions in separate engines. It is a core feature of Hadoop 2, which increases the size and functionality of any cluster. YARN helps support non-MapReduce applications running on the same cluster as MapReduce programs.

3.3 Apache Spark

Apache Spark is a fast data processing engine designed specifically for managing massive amounts of distributed data on a large scale [23]. The program's main advantages are its speed, lazy mode, and versatility [24]. Spark is a software program that effectively solves big data problems. It's one of the most used programs thanks to its support for multiple parallel operations and scaling [25]. Spark provides the same fault tolerance and scalability as MapReduce by supporting apps that retrieve working datasets over several operations [26]. When Apache Spark operates only in memory, activities can be completed up to 100 times faster than Hadoop MapReduce. Combining disks and memory can speed up the process by 10 times. Spark is designed to work well with RDD (resilient distributed dataset) [27]. **RDD:** Spark stores information about shards to reconstruct lost fragments in case of a loss. This makes Spark more resilient, as opposed to other programming languages that would have to revert to a previous checkpoint. At each iteration, Spark workers reuse data

read from HDFS or other file systems. This avoids using up system resources by duplicating the data for a checkpoint. Reorganizing data on a read-only disk helps speed up processing. RDD partitions can't be modified [28]. To compare Hadoop and Spark, the authors in [13] used a custom benchmarking study for binary classification tasks. The study used diverse LibSVM datasets and generated sequential files for Hadoop. Execution time, accuracy, and scalability were evaluated.

4 HIBENCH BENCHMARK AND ANALYSIS

4.1 Methodology

The benchmarking methodology employed in this study involved a systematic and reproducible evaluation approach for each benchmark category. For the "WordCount" benchmark, the WordCount benchmark was executed across three distinct input sizes (37 KB, 314 MB, and 3.1 GB), generating datasets automatically using the HiBench benchmark suite. A similar systematic approach was adopted for the "Sort" benchmark, focusing on input sizes ranging from 3 MB to 3.3 GB. The "PageRank" and "K-means" benchmarks followed comparable methodologies, utilizing datasets generated by the HiBench suite with varying input sizes. For the "Naive Bayes" benchmark, the deployment involved a pseudo-distributed environment with standardized system configurations, emphasizing consistency and control in the testing environment. In our comparison, we used virtual machines, which provide many advantages over physical machines. They're easy to maintain and can be quickly deployed. We installed Spark and Hadoop with a 6 GB and 60 GB hard disk. We used Linux Ubuntu 18 as an operating system, and we placed a single workstation in this experiment. However, each daemon ran in its own Java virtual machine using the HDFS file system.

The cluster deployment setup parameters are:

- Spark and Hadoop are installed in pseudo-distributed mode
- Spark Version: 2.4.5
- Hadoop Version:2.8.4
- Hibench Version The bigdata micro-benchmark suite v7
- Spark configurations:
  - Executor Number: 2
  - Executor cores: 4
  - Executor Memory: 2 GB
  - Driver Memory: 1 GB

4.2 Benchmarks

Table 1. Benchmark categories

Category	Benchmarks
Micro-Benchmark	Sort
	WordCount
	TeraSort
Web Search	PageRank
Machine Learning	Naive Bayes
	K-means

Table 1 categorizes the benchmarks needed to assess Hadoop and Spark performance. HiBench contains workloads and generates all the input files.

Micro-benchmark: We used three benchmarks: Sort, WordCount, and TeraSort. Sort is I/O-bound and involves sorting a data source. WordCount is CPU-bound and extracts a small subset of data from a larger source. TeraSort is CPU-bound in Map and I/O-bound in Reduce and involves sorting a larger data source generated by Teragen. **Web search:** This test measures the performance of the cluster for page ranking tasks. A preparatory phase generates the graph of the data to be processed via the PageRank algorithm. Then, the processing is carried out by a series of MapReduce jobs. This test is a CPU-bound type. **Machine learning:** Naive Bayes classification performs random classification on a set of data. It involves two MapReduce jobs and is I/O-bound with high CPU usage. The K-means algorithm classifies a dataset and visualizes its representation. It is compute-bound during iteration and I/O-bound during clustering.

4.3 Performance measurement

In assessing the benchmarks, various performance metrics were systematically captured and analyzed. For the "WordCount" benchmark, the evaluation encompassed key indicators: execution time, CPU utilization, memory usage, and speedup. This analysis offered valuable insights into the efficiency of both Spark and Hadoop, highlighting their adeptness in managing diverse data scales. Similarly, the "Sort" benchmark underwent scrutiny through the monitoring of execution times, coupled with a detailed examination of memory and CPU usage patterns. In the case of the "PageRank" benchmark, a thorough assessment involved tracking execution times across three iterations, with a specific focus on memory and CPU usage under different input sizes. The performance examination of the "K-means" benchmark highlighted efficiency and scalability, demonstrating Spark's significant speedup advantage. Lastly, the analysis of the "Naive Bayes" workload centered on key metrics: execution time, speedup, memory usage, and CPU usage, offering a holistic comprehension of the efficiency and scalability of both Spark and Hadoop in tasks demanding significant memory resources.

4.4 Results and analysis

WordCount benchmark

Table 2. Workload execution time and speedup of Spark over Hadoop

Input Size		37 KB	314 MB	3.1 GB
Execution Time (s)	Hadoop	99.02	177.71	823.33
	Spark	50.09	65.31	182.44
Speedup		2.0	2.7	4.5

Table 2 illustrates the speedup difference between Spark and Hadoop when working with the same datasets and demonstrates the improvement in runtime performance when a WordCount program runs on Spark compared to one running on Hadoop.

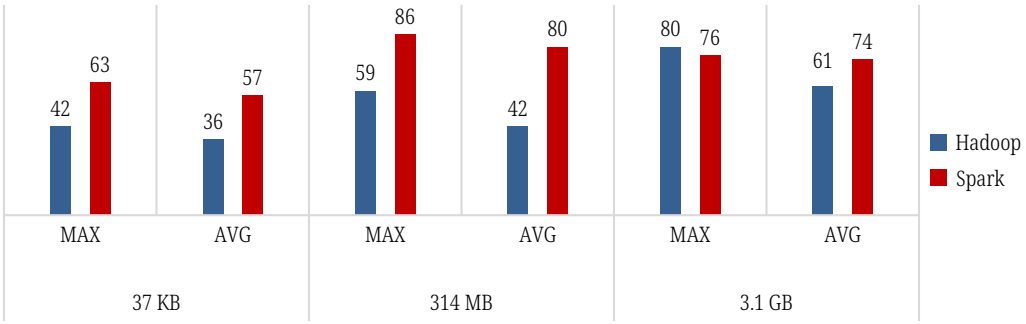

Fig. 1. Percentage of memory usage

As illustrated in Figure 1, Hadoop and Spark memory usage increases as their input size increases. But when it comes to the percentage of memory usage, Hadoop outperforms Spark. In terms of CPU usage, Spark has outstanding performance in CPU resource saving, and consumes up to 5 times less than Hadoop, as shown in Figure 2, since Hadoop counts on the hard disk while Spark stores the key-value pairs in memory.

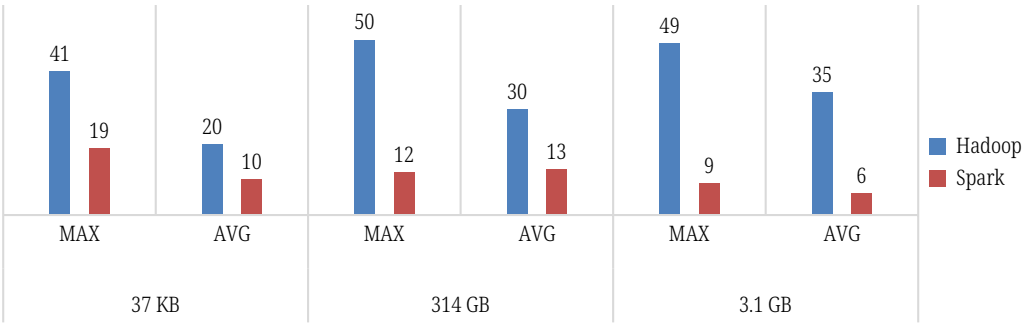

Fig. 2. Percentage of CPU usage

Performance factors and interpretation: The performance difference between Hadoop and Spark in the WordCount benchmark is mainly due to the size of the input data and the characteristics of the tasks. Spark is more efficient than Hadoop for larger datasets, achieving up to 4.5 times faster speedups. This is because Spark can store key-value pairs in memory, which reduces the need for disk access and improves CPU utilization.

Sort benchmark

Table 3. Workload execution time and speedup of Spark over Hadoop

Input Size (MB)		3	328	3285
Execution Time (s)	Hadoop	129	271	2206
	Spark	54	119	694
Speedup		2.4	2.3	3.2

Table 3 shows that Spark outperforms Hadoop when processing data of various sizes. The sizes of the inputs vary from 3 MB to 3.3 GB. The maximum acceleration is 3.2 times.

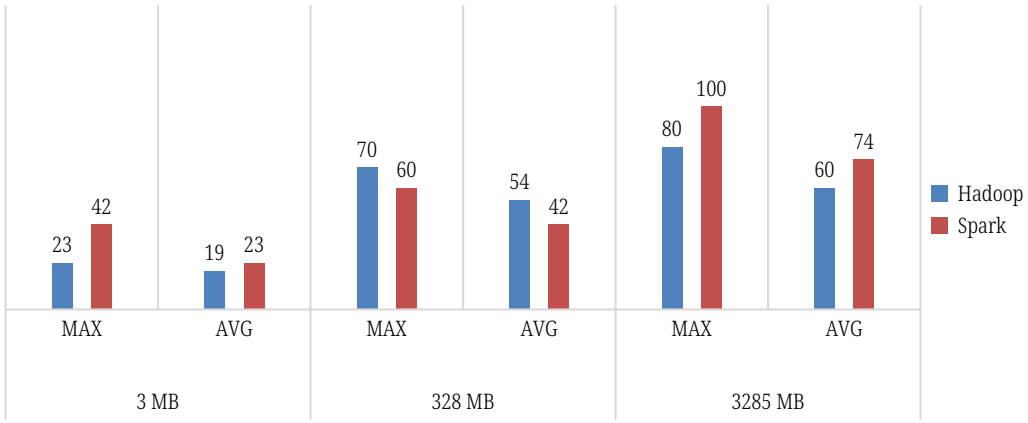

Fig. 3. Percentage of memory usage

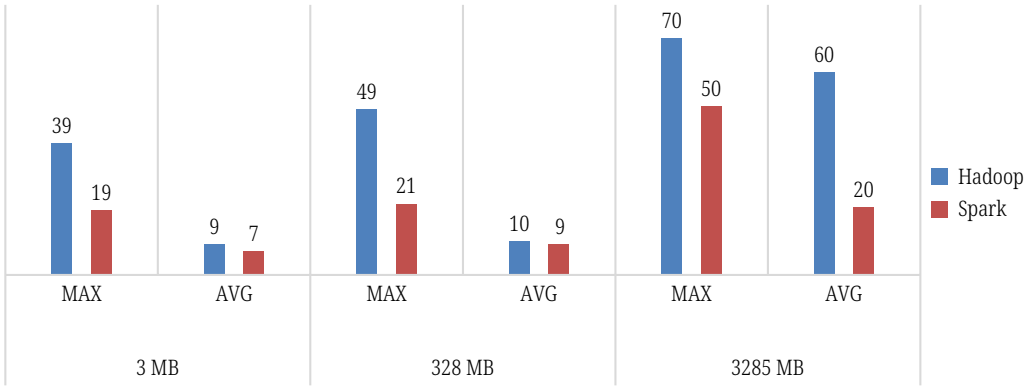

Fig. 4. Percentage of CPU usage

Figures 3 and 4, show memory and CPU usage for Hadoop and Spark. To implement Sort Workloads, Spark requires more memory resources but costs less CPU resources than Hadoop. Hadoop has an extremely high maximum CPU utilization, up to three times higher than Spark.

Performance Factors and Interpretation: A closer look at Table 3 reveals that Spark consistently outperforms Hadoop across varying data sizes, indicating its adaptability to different workloads. Spark's sustained superiority in sort workloads highlights its efficiency in data sorting and manipulation. System configuration analysis shows that Spark demands more memory but efficiently utilizes CPU resources, contrasting with Hadoop's high CPU utilization, suggesting inefficiencies in its resource management.

TeraSort benchmark

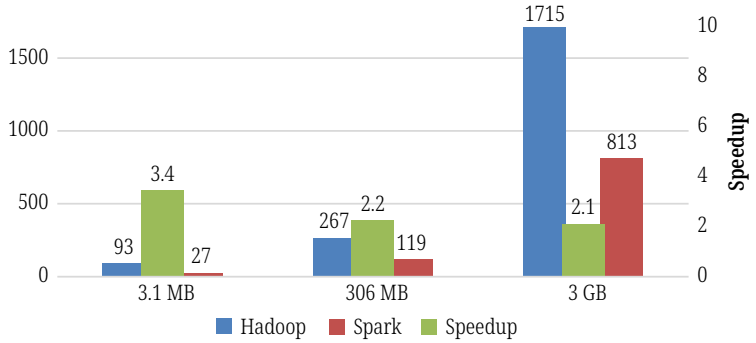

Fig. 5. Execution time comparison for TeraSort in seconds

Figure 5 shows a comparison of runtime performance for TeraSort running on Hadoop and Spark, illustrating the speedup of Spark over Hadoop on the same datasets. Input sets vary from three million to 3000 million records in size, with each record being 100 bytes. When the input is tiny, Spark and Hadoop perform equally well. When the input is greater than three million records, Spark outperforms Hadoop. At a large input size, the difference is more visible.

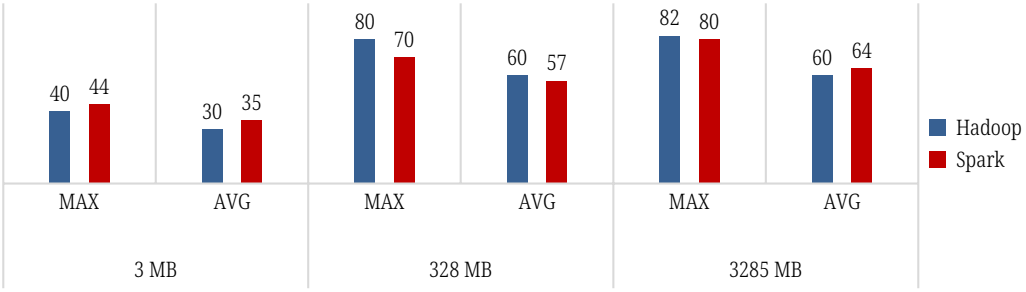

Fig. 6. Percentage of memory usage

Figures 6 and 7 present the memory and CPU usage, respectively. The memory usage is comparable for both frameworks, indicating efficient resource utilization. However, Figure 7 illustrates that Spark consumes fewer CPU resources, particularly evident at a data size of 3 GB, where Hadoop's maximum CPU usage is six times higher than Spark.

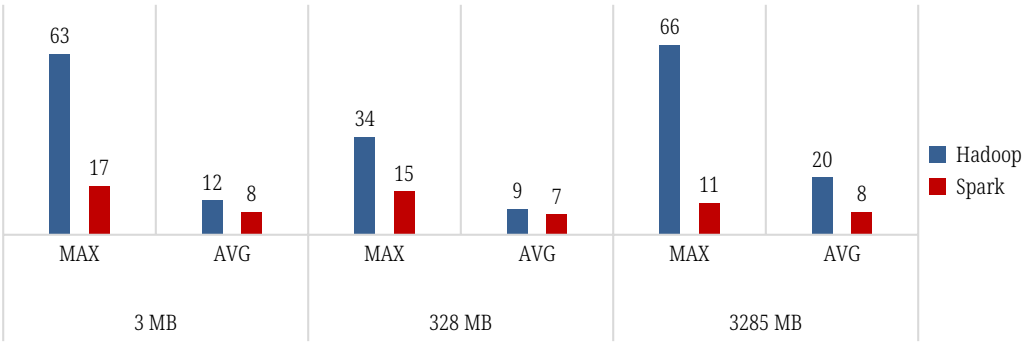

Fig. 7. Percentage of CPU usage

Performance factors and interpretation: Spark consistently outperforms Hadoop across data sizes and workloads in the TeraSort benchmark, demonstrating its efficiency in handling diverse datasets. Spark's superior performance benefits from its efficient CPU utilization, while Hadoop's high CPU usage suggests inefficiencies. Spark's ability to achieve competitive performance with lower CPU utilization reaffirms its prowess in handling diverse workloads compared to Hadoop.

PageRank benchmark

Table 4. Speedup of Spark over Hadoop

Input Size		10.6 KB	1.73 MB	1.1 GB
Number of Pages		5E + 01	5E + 03	5E + 06
Execution Time (s)	Hadoop (s)	200	600	5988
	Spark (s)	40	70	4232
Speedup		5.0	8.6	1.4

Table 4 shows a comparison of runtime performance for PageRank running on Hadoop and Spark. Table 4 summarizes the acceleration of Spark over Hadoop on the same datasets. Input datasets range from 50 pages to five million pages, and input sizes range from 10.6 KB to 1.1 GB.

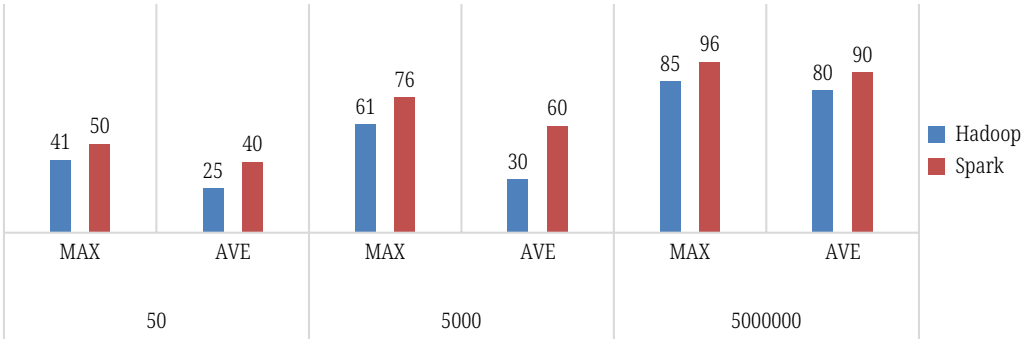

Fig. 8. Percentage of memory usage

The workload is evaluated through three iterations. Spark outperforms Hadoop by up to 8.6 times when the input size is between 50 and 5000 pages, whereas the input size increases and Hadoop's execution time approaches Spark's time. PageRank works with iterations that require more memory resources. As shown in Figure 8, Spark consumes up to twice as much RAM as Hadoop. Whenever the input data size is too large, Spark becomes limited by memory for newly generated RDDs and must initiate its replacement policy, affecting Spark's performance and increasing CPU use, as seen in Figure 9. Hadoop's CPU utilization is about the same regardless of the input size. Compared to Hadoop, Spark saves CPU resources with minimal inputs.

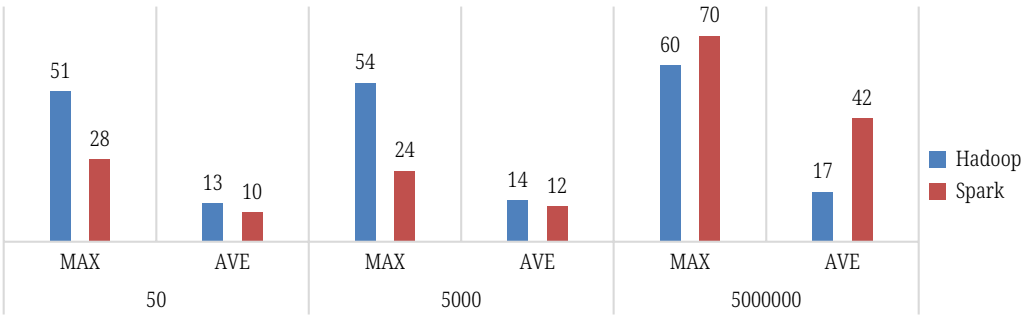

Fig. 9. Percentage of CPU usage

Performance factors and interpretation: Spark outperforms Hadoop for smaller datasets, with the gap narrowing as the size of the data increases. Spark's higher memory consumption is evident in PageRank tasks, while Hadoop maintains consistent CPU utilization. Data scale and task type influence the performance of Spark and Hadoop.

K-means benchmark

Table 5. Workload execution time and speedup of Spark over Hadoop

Input Size (MB)		1.33	603	4096
Execution Time (s)	Hadoop	467	881	3462
	Spark	71	127	956
Speedup		6.6	6.9	3.6

Table 5 highlights Spark's performance advantage over Hadoop on the same data sets. Furthermore, compare the execution time performance of K-means. The input size is between 1 MB and 4 GB. We ran the K-means algorithm with three clusters. The results presented in the table clearly show that Spark has a speedup of up to 6.9 times better than Hadoop. However, the benefit is limited by memory. Acceleration decreases when input is greater than 603 MB at equal speedup 3.6 times when input size is 4 GB.

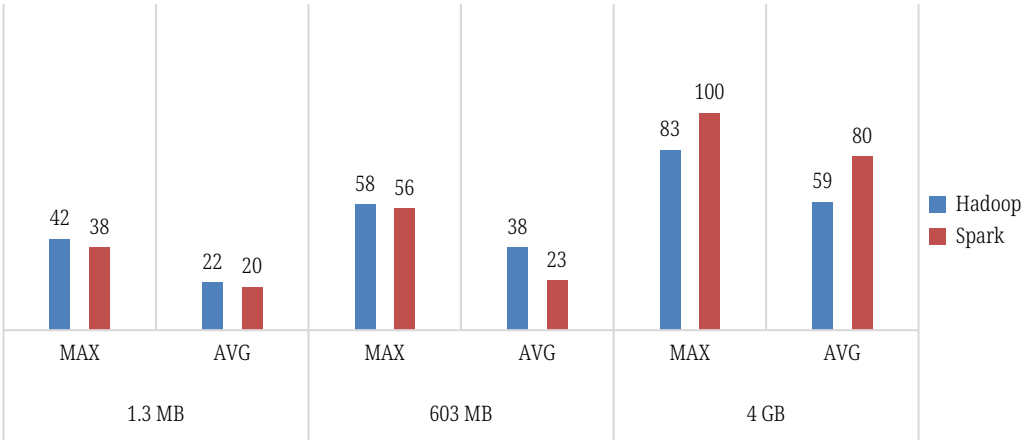

Fig. 10. Percentage of memory usage

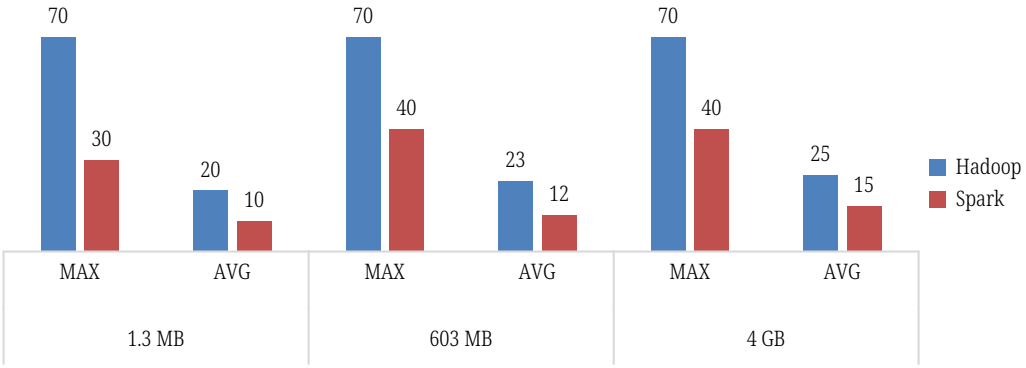

Fig. 11. Percentage of CPU usage

As shown in Figure 10, the maximum memory use for Spark is 100% with a 4 GB input. Spark cannot produce any more RDDs at this time, but it saves more CPU resources than Hadoop, as shown in Figure 11, particularly with tiny input. Its maximum CPU usage is only half that of Hadoop, with inputs of 1.3 MB and 603 MB.

Performance factors and interpretation: The performance gap between Hadoop and Spark is influenced by key factors. Spark is faster with smaller datasets, but this advantage decreases as data scales up, reaching 3.6 times speedup at 4 GB. Spark's in-memory storage benefits are evident in the efficiency of the K-means algorithm compared to Hadoop's disk-based approach. System configuration, including executor parameters, also impacts performance, emphasizing the relationship between data characteristics, algorithm requirements, and system setups.

Naive Bayes benchmark

Table 6. Workload execution time and speedup of Spark over Hadoop

Input Size (MB)		88.50	106.23	358.37
Pages Numbers		25000	30000	500000
Class Numbers		10.00	100.00	100.00
Execution Time (s)	Hadoop	789.00	1506.00	4660.00
	Spark	98.00	102.00	146.00
Speedup		8.1	14.8	31.9

Table 6 summarizes the speedup of Spark over Hadoop on the same datasets. In addition, compare the execution time performance of Naive Bayes. Input sizes range from 88.5 to 359 MB. Spark is intended for jobs that require multiple iterations using the same dataset to improve a parameter's performance by assuming it has better results than Hadoop on Naive Bayes, which is a machine learning benchmark. Table 6 shows that Spark has a significant advantage in performing Naive Bayes workloads, with a speedup up to 31 times more than Hadoop, particularly with large data sizes.

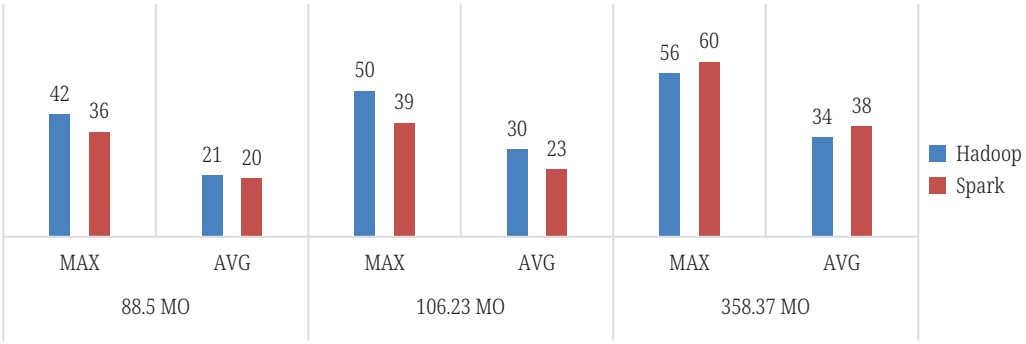

Fig. 12. Percentage of memory usage

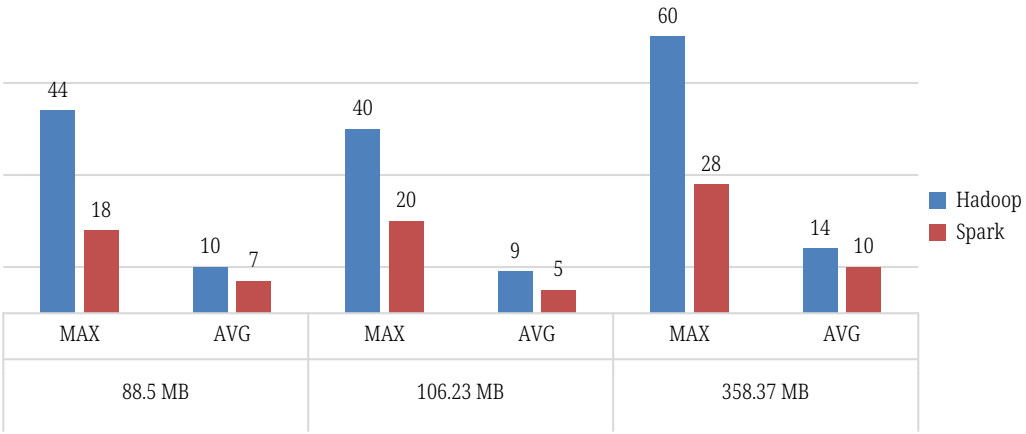

Fig. 13. Percentage of CPU usage

Figure 12 indicates that Hadoop slightly exceeds Spark in memory utilization for the Naive Bayes benchmark with a large input dataset. Yet, in terms of CPU usage, Figure 13 shows that Spark outperforms Hadoop.

Performance factors and interpretation: Spark outperforms Hadoop in the Naive Bayes benchmark due to its architecture, which is optimized for iterative processing in machine learning tasks. Spark's efficient use of memory and CPU resources also contributes to its performance advantage. The benchmark results show that Spark can be up to 31.9 times faster than Hadoop.

5 CONCLUSION

In conclusion, our study underscores critical implications for the development of efficient and scalable data processing solutions within the realm of big data. The consistent outperformance of Spark, particularly in memory-intensive tasks, emphasizes its role as a powerful engine for applications requiring iterative processing, such as machine learning and web searching. The specific scenarios where Spark excels, such as with K-means and Naive Bayes on small inputs, provide actionable insights for decision-makers. For tasks demanding multiple iterations and substantial memory resources, Spark emerges as the preferred choice, while Hadoop may find suitability in environments dealing with large input files and limited memory. These recommendations offer practical guidance in aligning framework choices with task requirements. Additionally, our study aligns with and contributes to the ongoing discourse on the performance of Hadoop and Spark in the dynamic landscape of big data. Recent studies, such as those by Samadi et al. [5] and Safa et al. [11], have consistently highlighted Spark's superior performance, particularly in scenarios with smaller inputs and memory-intensive tasks. These findings reinforce larger trends in big data study. that highlight the role of memory efficiency and real-time processing capabilities in shaping the choice between Hadoop and Spark. Future study directions may explore scalability with larger datasets, encompassing various data mining tasks beyond classification, and address additional aspects such as energy efficiency and adaptability to changing workloads in cloud environments.

6 REFERENCES

[1] M. Naeem *et al.*, "Trends and future perspective challenges in big data," in *Advances in Intelligent Data Analysis and Applications. Smart Innovation, Systems and Technologies*, J. S. Pan, V. E. Balas, and C. M. Chen, Eds., Springer, Singapore, vol. 253, 2022, pp. 309–325. https://doi.org/10.1007/978-981-16-5036-9_30

[2] A. Mostafaeipour, A. Jahangard Rafsanjani, M. Ahmadi, and J. Arockia Dhanraj, "Investigating the performance of Hadoop and Spark platforms on machine learning algorithms," *The Journal of Supercomputing*, vol. 77, pp. 1273–1300, 2021. https://doi.org/10.1007/s11227-020-03328-5

[3] M. Hakiki *et al.*, "Enhancing practicality of web-based mobile learning in operating system course: A developmental study," *International Journal of Interactive Mobile Technologies (iJIM),* vol. 17, no. 19, pp. 4–19, 2023. https://doi.org/10.3991/ijim.v17i19.42389

[4] A. D. Samala *et al.*, "Top 10 most-cited articles concerning blended learning for introductory algorithms and programming: A bibliometric analysis and overview," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 17, no. 5, pp. 57–70, 2023. https://doi.org/10.3991/ijim.v17i05.36503

[5] Y. Samadi, M. Zbakh, and C. Tadonki, "Performance comparison between Hadoop and Spark frameworks using HiBench benchmarks," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 12, 2018. https://doi.org/10.1002/cpe.4367

[6] A. Vettriselvi, N. Gnanambigai, P. Dinadayalan, and S. Sutha, "A comparative study of machine learning algorithms using RDD based regression and classification methods," *Annals of the Romanian Society for Cell Biology*, vol. 25, no. 4, pp. 15168–15199, 2021. http://annalsofrscb.ro/index.php/journal/article/view/4887/3915

[7] I. Lagwankar, A. N. Sankaranarayanan, and S. Kalambur, "Impact of map-reduce framework on hadoop and spark MR application performance," in *2020 IEEE International Conference on Big Data (Big Data),* Atlanta, GA, USA, 2020, pp. 2763–2772. https://doi.org/10.1109/BigData50022.2020.9378269

[8] M. A. Hedjazi, I. Kourbane, Y. Genc, and B. Ali, "A comparison of Hadoop, Spark and storm for the task of large-scale image classification," in *2018 26th Signal Processing and Communications Applications Conference (SIU),* Izmir, Turkey, 2018, pp. 1–4. https://doi.org/10.1109/SIU.2018.8404688

[9] J. Lee, B. Kim, and J. M. Chung, "Time estimation and resource minimization scheme for Apache spark and hadoop big data systems with failures," *IEEE Access*, vol. 7, pp. 9658–9666, 2019. https://doi.org/10.1109/ACCESS.2019.2891001

[10] A. Singh, A. Khamparia, and A. K. Luhach, "Performance comparison of Apache hadoop and Apache spark," in *Proceedings of the Third International Conference on Advanced Informatics for Computing Research (ICAICR '19)*, 2019, pp. 1–5. https://doi.org/10.1145/3339311.3339329

[11] S. Alkatheri, S. A. Abbas, and M. A. Siddiqui, "A comparative study of big data frameworks," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 1, pp. 66–73, 2019.

[12] S. Ketu, P. K. Mishra, and S. Agarwal, "Performance analysis of distributed computing frameworks for big data analytics: Hadoop vs Spark," *Computación y Sistemas*, vol. 24, no. 2, pp. 669–686, 2020. https://doi.org/10.13053/cys-24-2-3401

[13] T. Tekdogan and A. Cakmak, "Benchmarking Apache Spark and Hadoop MapReduce on big data classification," in *Proceedings of the 2021 5th International Conference on Cloud and Big Data Computing (ICCBDC),* 2021, pp. 15–20. https://doi.org/10.1145/3481646.3481649

[14] S. Ibtisum, E. Bazgir, S. M. A. Rahman, and S. M. S. Hossain, "A comparative analysis of big data processing paradigms: Mapreduce vs. apache spark," *World Journal of Advanced Research and Reviews*, vol. 20, no. 1, pp. 1089–1098. https://doi.org/10.30574/wjarr.2023.20.1.2174

[15] M. Younas, "Research challenges of big data," *Service Oriented Computing and Applications*, vol. 13, pp. 105–107, 2019. https://doi.org/10.1007/s11761-019-00265-x

[16] F. Cappa, R. Oriani, E. Peruffo, and I. McCarthy, "Big data for creating and capturing value in the digitalized environment: Unpacking the effects of volume, variety, and veracity on firm performance," *Journal of Product Innovation Management*, vol. 38, no. 1, pp. 49–67, 2021. https://doi.org/10.1111/jpim.12545

[17] B. Amen, S. Faiz, and T. T. Do, "Big data directed acyclic graph model for real-time COVID-19 twitter stream detection," *Pattern Recognition*, vol. 123, p. 108404, 2022. https://doi.org/10.1016/j.patcog.2021.108404

[18] G. Vranopoulos, N. Clarke, and S. Atkinson, "Addressing big data variety using an automated approach for data characterization," *Journal of Big Data*, vol. 9, 2022. https://doi.org/10.1186/s40537-021-00554-3

[19] A. P. Reimer and E. A. Madigan, "Veracity in big data: How good is good enough," *Health Informatics Journal*, vol. 25, no. 4, pp. 1290–1298, 2019. https://doi.org/10.1177/1460458217744369

[20] Samir Abou El-Seoud, Hosam F. El-Sofany, Mohamed Ashraf Fouad Abdelfattah, and Reham Mohamed, "Big data and cloud computing: Trends and challenges," *International Journal of Interactive Mobile Technologies (iJIM),* vol. 11, no. 2, pp. 34–52, 2017. https://doi.org/10.3991/ijim.v11i2.6561

[21] K. Kalia and N. Gupta, "Analysis of Hadoop MapReduce scheduling in heterogeneous environment," *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 1101–1110, 2021. https://doi.org/10.1016/j.asej.2020.06.009

[22] Z. Lu, N. Wang, J. Wu, and M. Qiu, "IoTDeM: An IoT big data-oriented MapReduce performance prediction extended model in multiple edge clouds," *Journal of Parallel and Distributed Computing*, vol. 118, pp. 316–327, 2018. https://doi.org/10.1016/j.jpdc.2017.11.001

[23] J. Liu, S. Tang, G. Xu, C. Ma, and M. Lin, "A novel configuration tuning method based on feature selection for Hadoop MapReduce," *IEEE Access*, vol. 8, pp. 63862–63871, 2020. https://doi.org/10.1109/ACCESS.2020.2984778

[24] R. J. Erizka, L. A. Romadhona, R. Febrianti, A. Winata, and C. P. Amanda, "Hadoop-MapReduce pada YARN framework," *Journal of Network and Computer Applications,* vol. 1, no. 2, pp. 40–47, 2022.

[25] H. Ahmadvand, M. Goudarzi, and F. Foroutan, "Gapprox: Using Gallup approach for approximation in big data processing," *Journal of Big Data*, vol. 6, 2019. https://doi.org/10.1186/s40537-019-0185-4

[26] E. S. Hamza, A. Abou El Kalam, A. Outchakoucht, and S. Benhadou, "Machine learning enhanced access control for big data," *Int. J. Comput. Sci. Netw. Secur.*, vol. 20, no. 3, pp. 83–91, 2020.

[27] Z. Ruihong and H. Zhihua, "WITHDRAWN: Comparative research on active learning of big data based on mapreduce and Spark," *Microprocessors and Microsystems*, p. 103425, 2020. https://doi.org/10.1016/j.micpro.2020.103425

[28] S. Badrinarayanan *et al.,* "A Plug-n-Play framework for scaling private set intersection to billion-sized sets," in *Cryptology and Network Security (CANS 2023),* in Lecture Notes in Computer Science, J. Deng, V. Kolesnikov, and A. A. Schwarzmann, Eds., Springer, Singapore, vol. 14342, 2022, pp. 443–467. https://doi.org/10.1007/978-981-99-7563-1_20

7 AUTHORS

Salah Eddine Hebabaze is a PhD student at the Faculty of Sciences, Ibn Tofaïl University, Kenitra, Morocco. Affiliated with the Research Laboratory in Computer Science and Telecommunications (LARIT), Team Networks and Telecommunications. She obtained a Research Master's degree in Cloud and High-Performance Computing from the National School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University (UM5), Rabat, Morocco (E-mail: salaheddine.hebabaze@uit.ac.ma).

Pr. Mohamed El Ghmary is a Professor of Computer Science at the Faculty of Science Dhar El Mahraz (FSDM), Sidi Mohamed Ben Abdellah University, Fez, Morocco. He is an associate member of the Intelligent Processing and Security of Systems (IPSS) team of the Computer Science Department at the Faculty of Science, Mohamed V University (UM5), Rabat, Morocco. He is also an associate member of the Research Laboratory in Computer Science and Telecommunications (LARIT), Team Networks and Telecommunications, Faculty of Science, Ibn Tofail University, Kenitra, Morocco. His research interests are mobile edge computing (MEC), cloud computing, machine learning, deep learning, intelligent systems, and optimization (E-mail: mohamed.elghmary@usmba.ac.ma).

Hamid El Bouabidi is a PhD student at the Faculty of Sciences, Ibn Tofaïl University, Kenitra, Morocco. Affiliated with the Research Laboratory in Computer Science and Telecommunications (LARIT), Team Networks and Telecommunications. He obtained a Research Master's degree in Cloud and High-Performance Computing

from the National School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University (UM5), Rabat, Morocco (E-mail: elbouabidihamid2015@gmail.com).

Sara Maftah is a PhD student at the Faculty of Sciences, Ibn Tofaïl University, Kenitra, Morocco. Affiliated with the Research Laboratory in Computer Science and Telecommunications (LARIT), Team Networks and Telecommunications. She obtained a Research Master's degree in Cloud and High-Performance Computing from the National School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University (UM5), Rabat, Morocco (E-mail: sara.maftah@uit.ac.ma).

Pr. Mohamed Amnai is an assistant at the National School of Applied Sciences in Khouribga, Settat University, Morocco. He joined the Department of Computer Science and Mathematics, Faculty of Sciences, Tofaïl University, Kenitra, Morocco, as an Associate Professor in 2018. He is also an associate member of the Research Laboratory in Computer Science and Telecommunications (LARIT), Team Networks and Telecommunications Faculty of Science, Kenitra, Morocco. He is also an associate member of the laboratory at the IPOSI National School of Applied Sciences, Hassan 1 University, Khouribga, Morocco (E-mail: mohamed.amnai@uit.ac.ma).