REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

IMPLEMENTATION OF DESCRIPTIVE AND EXPLORATORY STATISTICS FUNCTIONS USING THE MAP-REDUCE MODEL

[redacted] — Amirkia RAFIEI OSKOOEI

COMPUTER PROJECT

Advisor
Prof. Mehmet Sıddık AKTAŞ

January, 2024

ACKNOWLEDGEMENTS

I sincerely thank Professor Mehmet Sıddık AKTAŞ for giving me the opportunity to work on this project and for guiding me throughout the process. His consistent support and weekly meetings played a crucial role in successfully completing the project.

I also express my gratitude to Yildiz Technical University for providing the necessary environment and facilities for project deployment and research, creating a conducive atmosphere for academic growth.

Lastly and most importantly, I want to deeply thank my family for their unwavering support at every stage of my life, especially during my time at university. Their encouragement has been vital in keeping me motivated, even in challenging situations.

Amirkia RAFIEI OSKOOEI

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Implementation of Descriptive and Exploratory Statistics Functions using the Map-Reduce Model

Amirkia RAFIEI OSKOOEI

Department of Computer Engineering
Computer Project

Advisor: Prof. Mehmet Sıddık AKTAŞ

This report outlines a university project focused on implementing and evaluating descriptive and exploratory statistical functions using the Map-Reduce model in Apache Spark. Developed in Python with the PySpark library, the project aims to enhance performance by transitioning from a single node to a cluster of nodes. The development environment includes PyCharm, Anaconda, and Apache Superset on both Windows and Linux Ubuntu, providing a comprehensive ecosystem for function implementation and visualization. The experimental study, conducted on both the original and an enlarged Iris dataset, highlights significant performance improvements in the latter case. The system demonstrates robustness, consistency, and scalability, showcasing its adaptability to various computational environments. While limitations arise with datasets much larger than the original, the project's contributions to big data analytics and statistical processing are evident. The report concludes by emphasizing the project's real-world applicability and suggesting future enhancements, such as the development of a custom library to expand its functionality.

Keywords: Big Data Analytics, Apache Spark, Apache Superset, Distributed Data Processing, Performance Improvement

ÖZET

Eşle-iİndirge Modeliyle Betimsel ve Çıkarımsal İstatistik Fonksiyonlarının Gerçeklenmesi

Amirkia RAFIEI OSKOOEI

Bilgisayar Mühendisliği Bölümü
Bilgisayar Projesi

Danışman: Prof.Dr. Mehmet Sıddık AKTAŞ

Bu rapor, Apache Spark'ta Map-Reduce modelini kullanarak tanımlayıcı ve keşifsel istatistiksel işlevleri uygulama ve değerlendirme odaklı bir üniversite projesini detaylandırmaktadır. Python dilinde PySpark kütüphanesi ile geliştirilen proje, performansı artırmayı hedefleyerek tek düğümden çoklu düğüme geçiş yapar. Geliştirme ortamı, işlev uygulama ve görselleştirme için kapsamlı bir ekosistem sağlayan Windows ve Linux Ubuntu'da PyCharm, Anaconda ve Apache Superset'i içerir. İki farklı boyuttaki Iris veri seti üzerinde gerçekleştirilen deneysel çalışma, özellikle ikincil durumda önemli performans artışlarını vurgular. Sistem, çeşitli hesaplama ortamlarına uyum sağlayan sağlamlık, tutarlılık ve ölçeklenebilirlik sergiler. Ancak, orijinalden çok daha büyük veri setleriyle sınırlamalar ortaya çıkar. Projenin büyük veri analitiği ve istatistik işleme alanındaki katkıları açıkça görülmektedir. Rapor, projenin gerçek dünya uygulanabilirliğini vurgulayarak gelecekteki geliştirmeleri önerir, örneğin işlevselliğini genişletmek için özel bir kütüphane geliştirmeyi içerir.

Anahtar Kelimeler: Büyük Veri Analitiği, Apache Spark, Apache Superset, Dağıtılmış Veri İşleme, Performans İyileştirme

1
Introduction

In today's fast-paced world of technology, the creation and gathering of large amounts of data have become a crucial aspect of both online and offline settings. This surge in data, commonly known as big data, has significantly changed how industries function [1]. The sheer volume, speed, and variety of generated data pose both challenges and opportunities for organizations [2]. Regardless of their size or industry, companies are wrestling with the task of extracting meaningful insights from these enormous datasets to stay competitive and make well-informed decisions [3].

1.1 Background

The increase in data sources, spanning from online interactions on the web to offline transactions and sensor data, has led to an unprecedented growth in data volumes [4]. This surge requires advanced tools and techniques to extract valuable information from these massive datasets [5]. For companies, the capability to analyze big data has become a strategic necessity, offering the potential to discover hidden patterns, correlations, and trends that can inform business strategies, improve decision-making processes, and enhance overall operational efficiency.

1.2 Motivation

The motivation for this project stems from recognizing the transformative impact that big data analytics can have on various industries. By implementing descriptive and exploratory statistical functions using MapReduce-style programming in Apache Spark, the goal is to address the challenges associated with efficiently processing large datasets. The motivation is grounded in the belief that utilizing distributed systems and parallel programming paradigms can unlock significant speedups, enabling the smooth application of these functions on extensive datasets.

1.3 Objectives

The main aim of this project is to use the capabilities of Apache Spark, a distributed computing framework, to implement descriptive and exploratory statistical functions. By deploying these functions in a cluster of nodes, the intention is to take advantage of the parallel processing capabilities of distributed systems, achieving improved efficiency and reduced processing times for big data tasks. Additionally, the goal is to visualize the outcomes of these functions using the open-source software Apache Superset, providing a comprehensive and insightful representation of the datasets under consideration.

1.4 Scope of the Project

While applying the functions to the well-known Iris dataset, chosen for its prevalence in data science and AI communities [6], the exploration extends to an enlarged version of the Iris dataset. This involves duplicating rows to simulate a more extensive big data environment. The scope includes executing functions across three different configurations (1, 2, and 3 nodes) for both the original and enlarged Iris datasets. The evaluation encompasses calculating speedup metrics, comparing the performance of single-node and multi-node configurations.

In summary, this project aims to connect the theoretical foundations of distributed computing with practical implementations in big data analytics, using Apache Spark, and visualizing the results for a comprehensive understanding of the datasets at hand.

2
Preliminary Examination

2.1 Review of Literature

The existing body of work on Apache Spark emphasizes its significant role as an open-source platform for handling large-scale data processing and machine learning. MLlib, a crucial distributed machine learning library, is highlighted within this context [7]. Another scholarly article [8] provides a technical overview, emphasizing Spark's in-memory programming model and integrated APIs. It discusses Spark's applications in machine learning, graph analysis, and stream processing. A comparative analysis [9] establishes Spark's advantages over Hadoop MapReduce, specifically addressing issues like high latency and fault tolerance. The paper also underscores Spark's efficiency in real-time data processing and time-series analysis. In the realm of big data analytics, a literature review [10] compares the performance of Hadoop and Spark, revealing Spark's superiority, particularly with small datasets, and emphasizes the significance of parameter selection.

Collectively, these scholarly works contribute to our understanding of Apache Spark's role across various data processing scenarios and highlight its evolving capabilities in addressing challenges within different industries.

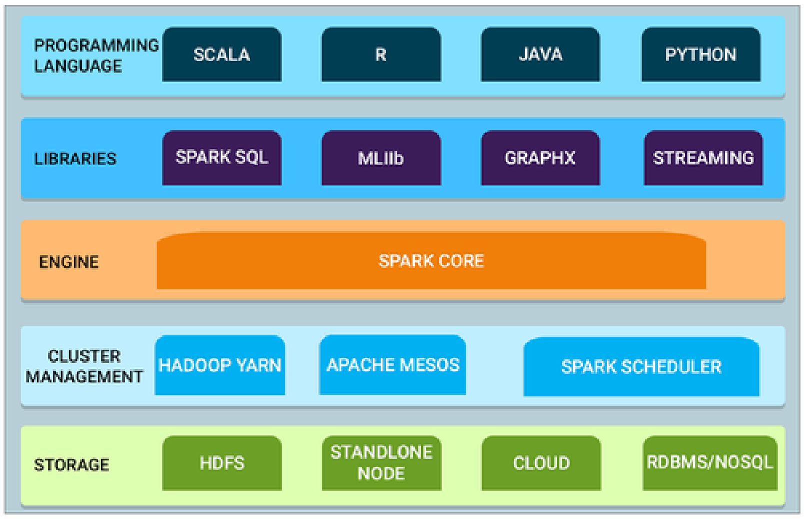
2.2 Apache Spark Overview

Apache Spark, a widely used open-source cluster computing framework, has become a key player in the world of big data processing, outshining its predecessor, Hadoop MapReduce. While maintaining the scalability and fault tolerance of MapReduce, Spark sets itself apart with impressive speed (up to 100 times faster for certain applications) and improved programming ease, offering rich APIs in Python, Java, Scala, and soon, R. At its core, Spark relies on Resilient Distributed Datasets (RDDs), a crucial concept enabling diverse processing tasks like SQL queries, streaming, machine learning, and graph processing [11].

Developed at the University of California, Berkeley, Spark's unified engine presents a seamless programming model, allowing users to create efficient and modular programs. Its versatility, supporting a wide array of tasks, brings benefits such as easy development, efficient task combination, and the facilitation of new applications, revolutionizing data science practices in both academic and commercial settings. Boasting over 1,000 contributors, Spark stands as the most active open-source project for big data processing, finding widespread adoption across diverse industries [12].

Spark's impact extends further with higher-level libraries like Spark SQL, Spark Streaming, GraphX, and MLlib, enhancing its capabilities and delivering top-notch performance on tasks ranging from relational queries to machine learning. Spark's significance in the industry lies in its ability to tackle challenges presented by large and varied datasets, solidifying its position as a crucial player in the ever-evolving landscape of big data analytics [13].

Figure 2.1 Apache Spark Ecosystem



2.3 Programming languages and Libraries

We decided to use Python as the main programming language and PySpark as the primary library for implementing statistical functions in this project because it aligns strategically with our overall goals. Python's flexibility and readability make it perfect for tasks related to data, and its extensive library ecosystem supports quick development. PySpark, an interface for Apache Spark written in Python, seamlessly integrates with the Spark ecosystem, offering a robust toolkit for distributed data processing.

By using Spark SQL and Spark MLlib within PySpark, we can efficiently implement both descriptive and exploratory statistical functions in a unified manner. These choices not only make it easy to integrate our project into the Apache Spark distributed computing framework but also allow us to tap into Spark's parallel processing capabilities for optimal performance on large-scale datasets. This aligns well with our project's objectives of scalability and efficiency in big data analytics.

3 Feasibility

3.1 Technical Feasibility

This project's practicality is clear in using a Windows 10 Pro system with an AMD Ryzen 5 2500U processor, 16GB RAM, and a 256GB SSD. While not the top-tier choice, this setup sufficiently meets the project's needs. PyCharm IDE is selected for developing Spark code using PySpark. Integrating PostgreSQL for the Iris database fits smoothly with the project's requirements. Despite Windows limitations for Apache Superset, a smart move is made to use VMware to install Ubuntu Linux for Superset, along with Docker and Portainer, ensuring a functional setup for both development and visualization.

3.2 Workforce and Time Feasibility

In terms of workforce and time, a solo development effort is the plan, drawing on a basic understanding of Python and Spark architecture. The addition of AI tools and chatbots is considered to boost programming efficiency. The proposed timeline spans a semester, outlining a well-structured development process. Initial weeks focus on understanding Apache Spark architecture and finding suitable functions, followed by a concentrated coding period using PySpark. Subsequent weeks involve setting up Linux Ubuntu, Docker, and Apache Superset, culminating in creating relevant charts for the dashboard. The final phase includes an experimental study and calculating speedup percentages, ensuring a systematic and time-efficient project completion.

3.3 Economic Feasibility

The economic feasibility is apparent in the careful use of both hardware and software resources. The provided laptop, valued at around 800 USD in 2023, is the primary hardware investment. Notably, all software components, including PySpark, PostgreSQL, Docker, Portainer, Apache Superset, VMware, and Linux (Ubuntu distro), are freely available, reducing software-related costs. Emphasizing open-source solutions aligns with an economically prudent approach, ensuring the project remains budget-friendly without sacrificing functionality or performance.

3.4 Legal Feasibility

The robust legal feasibility of the project is evident, as all tools, technologies, and datasets are open-source and available for public use. Licensing under open-source licenses for tools like Apache Spark, PySpark, PostgreSQL, Docker, and Portainer ensures unrestricted utilization. The Iris dataset, a key project component, is publicly accessible, removing any legal constraints. The project's commitment to open-source practices ensures compliance with licensing terms, creating a legally sound project environment.

4
System Analyze

4.1 Functional Requirements

The system's functional requirements are centered around implementing a comprehensive set of descriptive and exploratory statistical functions using Apache Spark and PySpark. Descriptive functions, including mean, median, minimum, maximum, variance, percentile, correlation, cumulative sum, skewness, and kurtosis, aim to provide valuable insights into the dataset. Exploratory functions, such as histogram, z-test, t-test, linear regression, and chi-square test, enhance the user's understanding of the data. The expected outcomes are not only displayed in the system console but are also integrated into visualizations within the Apache Superset dashboard. This integration enhances the end user's experience by offering a multifaceted understanding of the dataset through both numerical outputs and graphical representations.

4.2 Non-functional Requirements

Key non-functional requirements include performance, scalability, and usability. Performance, especially for multi-node configurations, is crucial, emphasizing the need for efficient parallel processing. The system must demonstrate adaptability to different datasets, ensuring versatility for both individual and organizational users. Scalability is paramount, particularly when transitioning from a single-node setup to multiple nodes, aiming for improved efficiency and reduced processing times. Usability considerations involve ease of use, allowing users to seamlessly apply statistical functions, configure Spark clusters, and visualize outcomes through Apache Superset.

4.3 Data Flow

The data flow within the system initiates with PySpark reading data from a .csv file and converting it to a schema if possible. Simultaneously, the .csv file undergoes conversion to a PostgreSQL database, facilitating seamless integration with Apache Superset. This connection allows for visualizations in the form of charts, tables, and graphs, enhancing the interpretability of statistical results. Furthermore, the Apache Superset dashboard provides the flexibility for end users to download visualizations as image or PDF files, enabling additional offline analysis and reporting.

4.4 Use-Case Scenarios

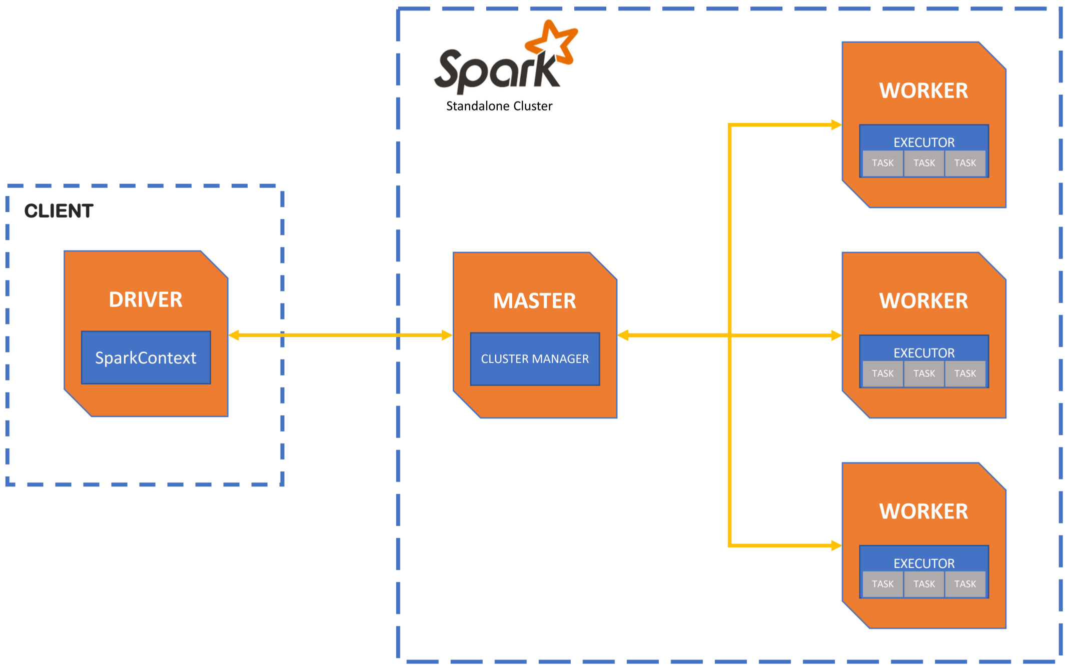
End-users interact with the system by executing statistical functions either through a text editor, IDE, or the command line. The functions can be applied to any .csv file with a similar dataset suitable for statistical analysis, with the user having the flexibility to modify schema or perform data preprocessing if necessary. Each function is isolated within its folder, utilizing separate Spark sessions for enhanced stability. Users can view function outputs in the command line or console. For Apache Superset, end-users can log in to their personalized dashboards, accessing visualizations and adding new charts and tables for an enriched analytical experience. This user-friendly interaction model ensures accessibility and versatility for a range of end-users.

5
System Design

5.1 Architectural Design

The system's architectural design centers around Apache Spark, featuring a master node housing the driver program containing PySpark codes and libraries. Optionally, worker nodes may be included, managed by a cluster manager responsible for distributing data and tasks [14]. In this project, the default standalone scheduler serves as the cluster manager for simplicity and effectiveness. The Spark architecture enables parallel processing and scalable computation. While other cluster managers like Apache Mesos and Hadoop YARN are available, the project opts for the default standalone scheduler.

Figure 5.1 Architecture of Apache Spark



5.2 Database Design

The database design focuses on PostgreSQL, chosen for its compatibility with Apache Superset and seamless integration. The Iris database mirrors the structure of the Iris dataset, including features like sepal width and length, petal width and length, and the species column. Apache Superset directly connects to this live database, treating it as a dataset. Superset utilizes this dataset for creating real-time, dynamic representations of statistical outcomes, enhancing interactivity and responsiveness in the Superset dashboard.

5.3 Algorithmic Design

The algorithmic design follows a consistent framework for implementing descriptive and exploratory statistical functions. Each function starts with configuring and creating a Spark session, initiating a timer. The dataset is then read from a .csv file, and the corresponding Spark library executes the desired function. After completion, results are presented, and the timer is stopped for accurate performance analysis timing. This meticulous approach ensures a standardized structure across all implemented functions.

5.4 Cluster Configuration Design

The Spark cluster is configured with three settings: 1 worker node, 2 worker nodes, and 3 worker nodes. Each worker node is allocated 2GB of memory, with an additional 1GB overhead memory for potential runtime requirements. The master node, housing the driver program, defaults to 1GB of memory. These configurations align with the experimental design, allowing for evaluating the system's performance under varying cluster sizes and memory allocations, balancing resource utilization and scalability.

5.5 Error Handling and Recovery

Apache Spark integrates robust error handling and recovery mechanisms for reliable and fault-tolerant data processing. Features include automatic retries, lineage tracking, checkpointing, accumulators, event logging, fault-tolerant shuffle service, task blacklisting, and dynamic resource allocation. These mechanisms enhance the system's resilience, ensuring consistent and accurate execution of statistical functions despite unexpected errors or failures.

5.6 Performance Optimization

To optimize performance, the project utilizes Apache Spark's built-in strategies, focusing on memory management, partitioning, caching, parallelism, and SQL query optimization. These strategies contribute to enhanced computational efficiency and reduced processing times, aligning with the project's objective of achieving optimal performance in big data analytics.

6 Implementation

6.1 Programming Langugaes and Libraries

The programming language chosen for this project is Python due to its popularity and readability [15], especially within the AI and data science communities [16]. Python's extensive libraries, particularly in data analytics and machine learning, make it ideal for creating statistical functions. The primary library used is PySpark, which seamlessly integrates Python with Apache Spark for distributed computing. This choice aligns with the project's goals of accessibility and ease of use for data science and AI practitioners.

6.2 Development Environment

The development environment revolves around PyCharm and Anaconda for code development and package management in Windows, while in a Linux Ubuntu virtual machine, Apache Superset runs within Docker containers managed by Docker Compose. Portainer is used as a graphical interface to monitor Docker containers and images. This setup ensures smooth integration between the development environment and the Linux-based execution environment.

6.3 Usage of Spark Libraries

The project extensively utilizes the PySpark SQL library for deploying and configuring Spark sessions across all functions. Additionally, certain functions make use of the scipy library alongside PySpark SQL for specific statistical analyses. For more advanced statistical analyses, PySpark MLlib is used in conjunction with PySpark SQL. Basic descriptive functions rely solely on the PySpark SQL library.

6.4 Cluster Deployment

Cluster deployment is achieved through a flexible code snippet that allows users to configure the number of cluster nodes, memory allocation, overhead memory, and application name. This adaptability caters to different experimental configurations, enabling scaling of the Spark cluster as per project requirements.

```
# Create a Spark session
spark = SparkSession.builder.appName("CorrelationCalculator") \
    .master("local[3]") \
    .config("spark.executor.memory", "2g") \
    .config("spark.executor.memoryOverhead", "1g") \
    .getOrCreate()
```

6.5 Monitoring

Monitoring and debugging are facilitated through the Spark dashboard on localhost:4040, providing insights into various aspects of the Spark session, aiding in efficient debugging and code optimization.

6.6 Documentation

The project utilizes AI chatbots for code documentation, ensuring thorough and consistent documentation throughout the codebase. This approach optimizes the use of time and resources, resulting in a well-documented and user-friendly codebase for developers and potential contributors.

7 Experimental Results

The evaluation of the implemented descriptive and exploratory statistical functions was conducted in three configurations, varying the number of nodes in the Spark cluster (1, 2, and 3 nodes). The primary metric for analysis was the elapsed time, emphasizing the project's core goal of reducing computation time for big data processing. Each function underwent three consecutive runs, and the average elapsed time was computed, serving as the representative runtime for performance analysis.

The original Iris dataset, consisting of 150 rows, was used as a benchmark, while an enlarged version, with 10 thousand times more rows, simulated a big data environment. The size difference between the original and enlarged datasets was significant, with the original iris.csv file measuring 5KB on a Windows system, and the enlarged version reaching approximately 44,345KB. This substantial increase in dataset size allowed for a comprehensive examination of the system's scalability and efficiency in handling larger data volumes.

The recorded elapsed time values for each function across varying configurations and dataset sizes are summarized in the presented table in seconds (s). The outcomes not only showcase the efficiency of the implemented functions in reducing computation time but also provide insights into their scalability concerning the number of nodes in the Spark cluster. The experimentally derived results form the basis for assessing the project's success in achieving its overarching goal of enhancing the performance of statistical functions on big data.

Table 7.1 Recorded elapsed time values for each function across varying configurations and datasets

Function	Nodes	iris.csv (s)	SpeedUp	iris10Kx.csv (s)	SpeedUp
Mean	1	6.33	0.0%	20.34	0%
	2	6.32	0.2%	14.51	29%
	3	6.37	-0.6%	12.96	36%
Median	1	6.43	0.0%	25.19	0%
	2	6.36	1.1%	18.25	28%
	3	6.31	1.9%	16.33	35%
Min-Max	1	6.41	0.0%	19.93	0%
	2	6.56	-2.3%	14.66	26%
	3	6.24	2.7%	12.94	35%
Histogram	1	7.7	0.0%	14.16	0%
	2	8	-3.9%	11.66	18%
	3	7.75	-0.6%	11.05	22%
Cummulative Sum	1	6.51	0.0%	19.04	0%
	2	6.65	-2.2%	14.26	25%
	3	6.59	-1.2%	12.69	33%
Percentile	1	7.3	0.0%	30.21	0%
	2	4.98	31.8%	19.43	36%
	3	5.41	25.9%	17.36	43%
Correlation	1	8.31	0.0%	48.6	0%
	2	8.2	1.3%	32.73	33%
	3	8.18	1.6%	26.43	46%
Skewness and Kurtosis	1	8.15	0.0%	23.44	0%
	2	8.13	0.2%	17.41	26%
	3	8.2	-0.6%	15.52	34%
Variance	1	6.92	0.0%	34.46	0%
	2	7	-1.2%	22.88	34%
	3	6.9	0.3%	19.75	43%
Z-test	1	5.35	0.0%	12.2	0%
	2	5.99	-12.0%	12.18	0%
	3	5.44	-1.7%	11.78	3%
T-test	1	11.04	0.0%	23.42	0%
	2	10.82	2.0%	22.79	3%
	3	10.52	4.7%	27.07	-16%
Chi-Square test	1	8.58	0.0%	16.47	0%
	2	8.55	0.3%	12.44	24%
	3	8.56	0.2%	11.66	29%
Linear Regression	1	7.97	0.0%	21.94	0%
	2	8.12	-1.9%	16.57	24%
	3	8.08	-1.4%	14.9	32%

8 Performance Analyze

The main aim of this project was to improve the performance of descriptive and exploratory statistical functions by employing a cluster of nodes in Apache Spark. Initially focusing on the original Iris dataset with 150 rows, the transition from a single node to a cluster of nodes did not yield significant performance improvements and, in some cases, even showed signs of degradation. This was attributed to inherent overhead related to job scheduling, data distribution, synchronization, and collection. The small scale of the original Iris dataset hindered the exploitation of parallel processing benefits, resulting in limited potential for speedup.

However, a crucial shift occurred when assessing performance on the 10 thousand times enlarged Iris dataset. In this scenario, the transition from a single node to a cluster of 2 or 3 nodes demonstrated a notable improvement, leading to a substantial reduction in runtime. The enlarged dataset, characterized by increased volume and complexity, effectively utilized the parallel processing capabilities of the Spark cluster. The observed performance gains affirmed the project's success in enhancing computational efficiency for big data scenarios.

It's important to highlight an exception regarding the t-test and z-test functions. Unlike other functions implemented with PySpark libraries, these specific statistical tests were implemented using the scipy library, which lacks adherence to map-reduce style programming and parallel processing features inherent to PySpark. Consequently, these functions did not exhibit any speedup, serving as a notable exception in the broader performance analysis. Overall, the project's success in enhancing performance on larger datasets emphasizes the significance of scalability considerations in implementing distributed computing solutions.

9 Results

The primary aim of achieving speedups on large datasets by transitioning from a single node to a cluster of nodes has been successfully accomplished in this project. With the exception of the t-test and z-test, implemented using the scipy library, all descriptive and exploratory statistical functions exhibited substantial performance improvements within a cluster environment. These outcomes highlight the project's success in optimizing computational efficiency for scenarios involving substantial data volumes.

Concerning system robustness, the observed speedups remained consistent and noteworthy across multiple runs under different configurations. The system exhibited stability and predictability, without instances of inconsistency or unexpected behavior. Despite the existing CPU and RAM limitations (4 cores and 16GB of RAM, respectively), the project demonstrated scalability, suggesting potential efficiency improvements on systems with higher core counts and greater RAM capacity, showcasing adaptability to varying computational environments.

However, a limitation was identified regarding the system's capability to handle datasets significantly larger than the original. Challenges, particularly in terms of RAM and JVM Heap overflow, arose when scaling the dataset to one million times larger than the original Iris dataset. The proposed solution involves leveraging industry-level computation systems or cloud computing, aligning with real-world scalability requirements for more extensive datasets.

This project contributes significantly to big data analytics and statistical processing. Its applicability extends beyond the Iris dataset, and with minimal code modifications, the functions can be adapted for real-world problems. The demonstrated scalability and efficiency emphasize the project's potential impact on a diverse range of applications. Looking ahead, enhancing the project could involve developing a custom library for implementing functions not available within the PySpark library, such as the t-test and z-test, further extending its utility and adaptability to various statistical processing tasks in big data analytics.

References

[1] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih, "Big data technologies: A survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 4, pp. 431–448, 2018.

[2] J. Fan, F. Han, and H. Liu, "Challenges of big data analysis," *National science review*, vol. 1, no. 2, pp. 293–314, 2014.

[3] T. H. Davenport and J. Dyché, "Big data in big companies," *International Institute for Analytics*, vol. 3, no. 1-31, 2013.

[4] K. Coffman and A. Odlyzko, "The size and growth rate of the internet," 1998.

[5] J. Zakir, T. Seymour, and K. Berg, "Big data analytics.," *Issues in Information Systems*, vol. 16, no. 2, 2015.

[6] T. M. Fahrudin, P. A. Riyantoko, and K. M. Hindrayani, "Implementation of big data analytics for machine learning model using hadoop and spark environment on resizing iris dataset," in *2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, IEEE, 2022, pp. 429–434.

[7] X. Meng *et al.*, "Mllib: Machine learning in apache spark," *The journal of machine learning research*, vol. 17, no. 1, pp. 1235–1241, 2016.

[8] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *International Journal of Data Science and Analytics*, vol. 1, pp. 145–164, 2016.

[9] N. Ahmed, A. L. Barczak, T. Susnjak, and M. A. Rashid, "A comprehensive performance analysis of apache hadoop and apache spark for large scale data sets using hibench," *Journal of Big Data*, vol. 7, no. 1, pp. 1–18, 2020.

[10] R. C. Maheshwar and D. Haritha, "Survey on high performance analytics of bigdata with apache spark," in *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, IEEE, 2016, pp. 721–725.

[11] J. G. Shanahan and L. Dai, "Large scale distributed data science using apache spark," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 2323–2324.

[12] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache spark: A big data processing engine," in *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, IEEE, 2019, pp. 1–6.

[13] M. Zaharia *et al.*, "Apache spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[14] A. J. Awan, M. Brorsson, V. Vlassov, and E. Ayguade, “Architectural impact on performance of in-memory data analytics: Apache spark case study,” *arXiv preprint arXiv:1604.08484*, 2016.

[15] Z. Ahmed, F. J. Kinjol, and I. J. Ananya, “Comparative analysis of six programming languages based on readability, writability, and reliability,” in *2021 24th International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2021, pp. 1–6.

[16] D. Cielen and A. Meysman, *Introducing data science: big data, machine learning, and more, using Python tools*. Simon and Schuster, 2016.

Curriculum Vitae

FIRST MEMBER

Name-Surname: Amirkia RAFIEI OSKOOEI
Birthdate and Place of Birth: [redacted]
E-mail: [redacted]
Phone: [redacted]
Practical Training:

Project System Informations

System and Software: Windows OS, Ubuntu Linux OS, VMWare Virtual Machine, Pycharm IDE, Python, Docker, Portainer, Apache Spark, Apache Superset, PostgreSQL
Required RAM: 16GB
Required Disk: 100GB