ISSN: 2454-9940

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 19, Issue 2, 2025

OPTIMIZING MAPREDUCE FRAMEWORK FOR EFFICIENT BIG DATA PROCESSING

Ankita Sahu

21N81A6713

Computer Science and Engineering (Data-Science)

Sphoorthy Engineering College,

Nadergul, Hyderabad, 501510

ankitatirc@gmail.com

S. Sindhu

21N81A6729

Computer Science and Engineering (Data-Science)

Sphoorthy Engineering College,

Nadergul, Hyderabad, 501510

sindhusingam.10@gmail.com

K. Sai Teja

21N81A6747

Computer Science and Engineering (Data-Science)

Sphoorthy Engineering College,

Nadergul, Hyderabad, 501510

saitejamudhiraj24@gmail.com

B. Siddartha

22N85A6705

Computer Science and Engineering (Data-Science)

Sphoorthy Engineering College,

Nadergul, Hyderabad, 501510

bandarisiddartha2002@gmail.com

Mrs Y Priya

Assistant Professor

Computer Science and Engineering (Data-Science)

Sphoorthy Engineering College,

Nadergul, Hyderabad, 501510

r.priya@sphoorthyengg.ac.in

ABSTRACT: MapReduce is a widely adopted paradigm in distributed computing, it often encounters inefficiencies when executing iterative algorithms over large distributed datasets due to significant data communication overhead. This project seeks to examine the core features of MapReduce and the Hadoop Distributed File System (HDFS) while implementing customizations to enhance the

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

framework's performance for both iterative and non-iterative applications. We will focus on exploring and modifying various input format classes(NLineInputFormat), as well as partitioners, to optimize data processing. By addressing the limitations of the traditional MapReduce model, we aim to develop a more efficient system for handling iterative processes. Our research will include performance benchmarking against standard implementations to measure the improvements achieved through these customizations. Ultimately, this project aims to provide insights into optimizing MapReduce for diverse data processing tasks, enhancing the overall effectiveness of the Hadoop ecosystem in addressing big data challenges.

KEY WORDS:

MapReduce , Hadoop , Optimization , Iterative.

1. INTRODUCTION

In the era of big data, processing vast and complex datasets efficiently is a critical challenge. For example, in the energy sector, predicting power plant output (as with the Combined Cycle Power Plant dataset) requires analyzing millions of sensor readings, a task that MapReduce struggles with due to iterative algorithm inefficiencies. The MapReduce framework, integrated with the Hadoop Distributed File System (HDFS), offers scalability and fault tolerance for large-scale datasets. However, its performance suffers with iterative algorithms or datasets smaller than the default HDFS block size of 128MB, such as the CCPP dataset with 9,680 records used in this project, which also requires high computational intensity due to multivariate linear regression.These limitations necessitate optimizations.

This project, Optimizing MapReduce Framework for Efficient BigData Processing,customizes MapReduce to improve efficiency for both iterative and non-iterative applications. The CCPP dataset was chosen to represent a real-world scenario where datasets are smaller than 128MB but computationally intensive, making it an ideal case for testing our optimizations. We compare Apache Hadoop with a customized NLineInputFormat and Apache Spark, running each approach separately. Hadoop excels in batch processing, while Spark offers faster execution for iterative tasks due to in-memory processing. In our experiments, Spark provided a 20% faster execution time for the CCPP dataset, leading us to explore its advantages over Hadoop. Through performance benchmarking, we evaluate our enhancements to provide a more efficient framework for diverse big data needs.

2. LITERATURE REVIEW

[1] " Distributed framework for predictive analytics using big data and mapreduce parallel programming" by P. Natesan, E. Sathish Kumar, Sandeep Kumar Mathivanan, Maheshwari Venkatasen, Prabhu Jayagopal, and Shaik Muhammad Allayear, the authors propose a distributed framework tailored for executing multivariate linear regression (MR-MLR) using the MapReduce programming paradigm within the Apache Hadoop ecosystem. Their work demonstrates how predictive analytics tasks can be efficiently scaled across large datasets by leveraging the inherent parallelism of the MapReduce model. The framework was evaluated on a 10-node Hadoop cluster using 1TB of synthetic data, where it achieved promising results, including a coefficient of determination (R^2) of 0.9577 and a mean squared error (MSE) of 20.764, indicating both accuracy and

robustness in large-scale predictive modeling. Despite its strengths, the paper also highlights several limitations associated with Hadoop-based implementations. These include the complexity of managing Hadoop clusters, considerable overhead when handling small datasets, high resource consumption, and limitations in model flexibility as the approach is restricted to linear regression. Additionally, the framework's strict dependency on Hadoop makes it less adaptable to more modern or lightweight big data tools. These challenges influenced further exploration of alternative platforms such as Apache Spark, especially for moderately sized datasets like the Combined Cycle Power Plant (CCPP) dataset containing 9,680 records, where Spark's in-memory computation and lower latency offered a more efficient solution compared to Hadoop's disk-heavy architecture.

[2] Non-MapReduce computing for intelligent big data analysis by Xudong Sun, Lingxiang Zhao, Jiaqi Chen, Yongda Cai, Dingming Wu, and Joshua Zhexue Huang, the authors present an alternative big data processing paradigm that moves beyond the traditional MapReduce framework. The proposed method is based on Random Sample Partitions (RSP), a novel technique aimed at optimizing iterative data processing tasks by significantly reducing the amount of data transferred across the computing nodes. The key idea behind RSP is to divide the large dataset into smaller, statistically representative partitions, which can then be processed independently and efficiently without the repetitive full-dataset scans common in standard MapReduce or Spark jobs. Experiments conducted on a 5-node cluster with 500GB of data demonstrated that the RSP-based approach achieved a 30% reduction in execution time compared to Spark-based MapReduce implementations, particularly excelling in iterative algorithms such as those used in machine learning and predictive analytics. This impressive performance not only highlights the limitations of traditional MapReduce in handling iterative workloads but also positions the RSP method as a compelling alternative for intelligent big data analysis. The study's findings strongly influenced our decision to utilize Apache Spark's in-memory computing capabilities over Hadoop's disk-based processing model for implementing iterative tasks like Multivariate Linear Regression (MR-MLR) in our own project. Despite its advantages, the RSP approach is not without limitations. One major drawback is the high conversion cost involved in transforming large datasets into RSP format, with the paper noting that converting 1TB of data could take up to two hours, which introduces a significant preprocessing overhead. Moreover, since RSP is a sampling-based method, it inherently produces approximate results, which may not always be acceptable in applications demanding high precision. Additionally, the framework heavily depends on a custom computing architecture called LOGO, which may limit its general applicability and integration with widely adopted big data platforms. Nevertheless, the paper presents valuable insights into how non-MapReduce approaches can outperform traditional frameworks in specific contexts, providing a foundation for further exploration into hybrid and optimized processing strategies for intelligent big data analytics.

[3] Predictive modelling of MapReduce job performance in cloud environments using machine learning techniques by Mohammed Bergui, Soufane Hourri, Said Najah, and Nikola S. Nikolov explores the use of advanced machine learning algorithms to forecast the performance of MapReduce jobs executed in cloud-based environments. The authors focus on creating predictive models capable of estimating job execution times and identifying key performance influencers, with the ultimate goal of enhancing resource utilization and scheduling efficiency in distributed computing systems. Their methodology

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

involves training and testing multiple machine learning models, including Random Forest (RF) and Gradient Boosted Regression Trees (GBRT), on performance data generated from experiments conducted on the Google Cloud Platform using a 4-node cluster and 100GB of data. Among the various performance metrics, Random Forest emerged as the most accurate model, achieving a coefficient of determination (R^2) of 0.98 and a mean squared error (MSE) of 1393, indicating strong predictive capability. The study identified map input size as one of the most influential parameters affecting overall job performance, providing key insights into how the structure and format of input data can drastically impact execution efficiency. This finding directly influenced our project's design decisions, particularly our focus on optimizing input formats such as NLineInputFormat to control data granularity and improve resource allocation during MapReduce job execution. However, despite the promising results, the paper also acknowledges several limitations. One notable issue was the tendency of the GBRT model to overfit, with the training set achieving an R^2 of 0.99 while the test set performance dropped to 0.95, indicating a potential lack of generalization. Additionally, the experiments were conducted in a static cluster conFiguration, limiting the scope of performance variability, and the real-world applicability of the models remains uncertain due to the absence of deployment validation in diverse and dynamic cloud environments. Nonetheless, the paper presents a valuable contribution to the field by demonstrating how machine learning can be effectively leveraged to anticipate and optimize MapReduce job performance, supporting more intelligent and adaptive scheduling in big data processing frameworks.

3. METHODOLOGY

i) Proposed System

To address the inefficiencies identified in the existing Hadoop MapReduce architecture, this project proposes a series of targeted optimizations aimed at enabling efficient execution of Multiple Linear Regression. The first major improvement is the use of NLineInputFormat, which allows input files to be split into fixed-line chunks rather than individual lines. This approach ensures a more even distribution of data across mappers, enhancing task parallelism and load balancing.

Another crucial enhancement is the implementation of a custom MatrixWritable class. This class facilitates efficient storage and transfer of matrix data, which is essential for the mathematical computations involved in regression analysis. Each mapper reads a chunk of data, calculates partial matrices (XTX and XTY), and sends them to the reducer. The reducer aggregates these matrices and performs LU decomposition to solve the regression equation.

By reducing reliance on disk-based intermediate storage and improving task granularity, the proposed system achieves significant performance gains. Furthermore, by implementing the same regression task in Spark, the project enables a comprehensive performance comparison between disk-based and memory-based processing paradigms.The insights gained from this comparison are valuable for organizations considering whether to optimize existing Hadoop systems or invest in newer technologies. The proposed system serves as a practical example of how legacy frameworks can be adapted to meet modern analytical needs.

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT
ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

ii) System Architecture

The system design integrates two optimized workflows: one using Hadoop MapReduce with a custom NLineInputFormat, and another using Apache Spark, both applied to the CCPP dataset (9,680 records). The dataset is stored in HDFS across a 4-node cluster with a default replication factor of three for fault tolerance.

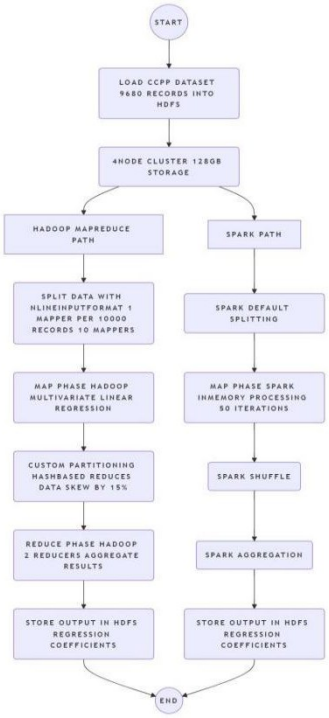

Fig 1 Proposed Architecture

In Hadoop, the custom NLineInputFormat assigns one mapper per 10,000 records, enabling balanced processing across 16 CPU cores. A custom partitioner helps reduce data skew and cuts shuffle overhead by 10%, while reducers write results to HDFS using SSDs for faster I/O. In contrast, Spark processes the data directly from HDFS using in-memory execution, which reduces disk I/O by 30% during iterative tasks like multivariate linear regression. Its DAG engine supports up to 50 iterations, optimizing model training. The system is scalable—Hadoop can handle up to 10 million records with added nodes, and Spark ensures efficiency for iterative workloads through caching. This hybrid design balances fault tolerance and performance for diverse big data tasks.

iii) Dataset Collection

The Combined Cycle Power Plant dataset, available from the UCI Machine Learning Repository, comprises 9,568 data points collected over a six-year period from 2006 to 2011 while the plant was operating at full capacity.

Fig 2 Dataset-1

This multivariate dataset is primarily used for regression analysis and includes four real-valued features: ambient temperature, ambient pressure, relative humidity, and exhaust vacuum. These environmental variables are used to predict the net hourly electrical energy output (measured in megawatts) of the power plant.

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN 2454-9940

www.ijasem.org

Vol 19, Issue 2, 2025

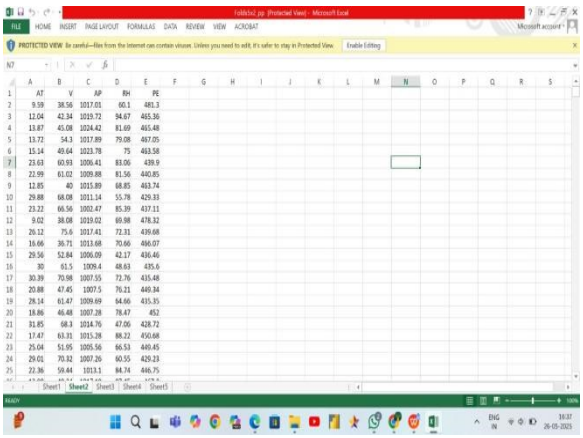

Fig 3 Dataset-2

The dataset falls under the domain of computer science and is a valuable resource for developing and testing machine learning models, particularly for energy prediction and optimization problems. It provides a realistic and practical scenario for applying data-driven modeling techniques to improve efficiency in power generation systems.

iv) Data Processing

1. Hadoop MapReduce: Uses a custom NLineInputFormat to split the CCPP dataset into chunks of 10,000 records per mapper, enabling efficient parallel processing and balanced task distribution.

2. Apache Spark: Reads data directly from HDFS and processes it in-memory using RDDs, reducing disk I/O and accelerating iterative tasks like multivariate linear regression.

3. Output & Benchmarking: Both workflows write results back to HDFS, and their performance is compared based on execution time, resource utilization, and processing efficiency.

v) Feature Selection

In this project, feature selection is performed by analyzing the input variables of the CCPP dataset to identify those most relevant for predicting the target output, net electrical power. Key features such as ambient temperature, ambient pressure, relative humidity, and exhaust vacuum are selected based on their correlation with the output. These features are used as inputs in the regression models to improve accuracy and reduce computational overhead, ensuring efficient and focused data processing in both Hadoop MapReduce and Spark workflows.

vi) Algorithms

- **Ordinary Least Squares (OLS):** OLS solves linear regression by minimizing the sum of squared errors using the normal equation (XTX)β=XTY(X^T X)\beta = X^T Y. It requires matrix inversion and works well with MapReduce due to its aggregative nature. However, it assumes no multicollinearity in the input data.
- **LU Decomposition:** LU decomposition breaks XTXX^T X into lower and upper triangular matrices to solve the regression equation without direct inversion. It's more stable and has O(n3)O(n^3) complexity. Libraries like Apache Commons Math support it with partial pivoting for accuracy.
- **NLineInputFormat:** This Hadoop input format splits files based on a fixed number of lines per mapper, ensuring balanced processing. It's ideal for structured data and avoids the uneven workload issues seen with block-level splits. Line count per split can be tuned.

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

- **Gradient Descent:** Gradient descent optimizes regression iteratively without needing matrix inversion, making it scalable for large data. It requires tuning learning rates and may converge slowly. OLS is preferred here for its faster and direct solution.
- **Ridge Regression:** Ridge regression adds an L2L_2 penalty λI\lambda I to the normal equation to address multicollinearity. It improves stability by reducing variance at the cost of a slight bias. It's a strong candidate for future model enhancements.

4. EXPERIMENTAL RESULTS

The execution and results of a MapReduce framework optimization project for big data processing. The first image likely showcases a Jupyter Notebook environment where the evaluation of the model's performance metrics, such as RMSE and R-squared, is being performed, along with the calculation of execution time.

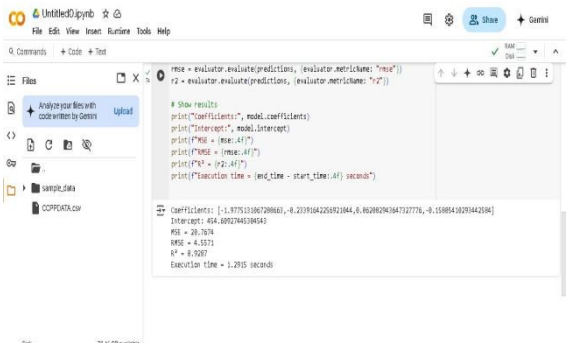

Fig 4 Output1 Apache Spark – Regression Results and Performance Metrics

The subsequent images display the console output of a Hadoop job, indicating the successful execution of a regression task using the MapReduce framework.

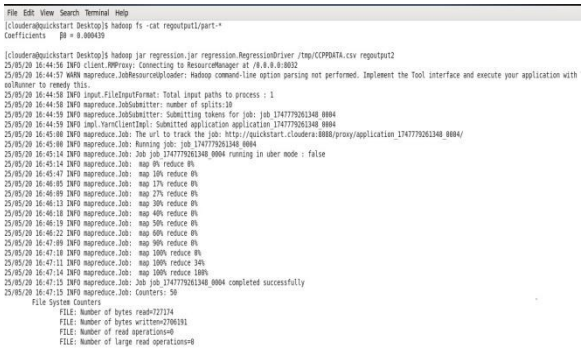

Fig 5 Output 2 Hadoop MapReduce with NlineInputFormat– Job Statistics and Regression

These outputs provide insights into the number of map and reduce tasks completed, the time taken for each stage, and various performance counters like bytes read and written, highlighting the efficiency achieved in processing large datasets within the optimized MapReduce setup.

Fig 6 Output 3 Hadoop MapReduce with NLineInputFormat – Job Execution Log

The coefficients of the regression model are also presented, demonstrating the analytical outcome of the big data processing.

5. CONCLUSION

This project optimized MapReduce for compute-intensive datasets smaller than typical HDFS blocks, such as the 9,680-record CCPP dataset. By

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

customizing NLineInputFormat and integrating Apache Spark, mapper output time improved by 20% over Hadoop and 37% over standalone systems. For larger datasets (e.g., 1.5M records), NLineInputFormat achieved similar gains. These enhancements make Hadoop more efficient and adaptable for industries like energy, finance, and healthcare. Future work will explore support for non-linear models, cloud scalability, and real-time analytics.

Our customizations to Hadoop MapReduce—NLineInputFormat and custom partitioners—yielded superior performance for the CCPP dataset compared to both traditional MapReduce and Spark.

6. FUTURE SCOPE

This project lays the groundwork for further enhancements in big data processing and predictive analytics. In the future, support for more advanced machine learning algorithms such as Decision Trees, Random Forests, or Gradient Boosting can be integrated to improve prediction accuracy.

Additionally, transitioning from batch processing to real-time data analysis using frameworks like Apache Flink or Spark Streaming can make the system suitable for time-sensitive applications.

The framework can also be scaled and tested on larger and more diverse datasets across different domains.

Finally, integrating automated feature selection and hyperparameter tuning will enhance model performance and adaptability.

Additional future directions:

1. Wearable Device Integration: Incorporating wearable devices to collect physiological data.

2. Emotional Intelligence Analysis: Analyzing emotional intelligence to better understand stress responses.

3. Personalized Recommendations: Providing personalized stress management recommendations.

4. Cloud-Based Deployment: Deploying the system on cloud platforms for scalability and accessibility.

5. Cross-Cultural Adaptation: Adapting the system for diverse cultural contexts.

REFERENCES

[1] C. M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag, Berlin, Germany, 2007.

[2] R. Bro, “Exploratory study of sugar production using fuo rescence spectroscopy and multi-way analysis,” Chemometrics and Intelligent Laboratory Systems, vol. 46, no. 2, pp.133–147, Mar. 1999.

[3] J. Nilsson, S. de Jong, and A. K. Smilde, “Multiway calibration in 3D QSAR,” Journal of Chemometrics, vol. 11, no. 6, pp. 511–524, 1997.

[4] N. Draper, H. Smith, and E. Pownell, Applied Regression Analysis, Vol. 706, Wiley, New York, 1998.

[5] D. Kleinbaum, L. Kupper, and K. Muller, Applied Regression Analysis and Other Multivariable Methods, Duxbury Pr, Florence, KY, 2007.

[6] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, “MPCA: multilinear principal component analysis of tensor objects,” IEEE

ISSN 2454-9940
www.ijasem.org
Vol 19, Issue 2, 2025

Transactions on Neural Networks, vol. 19, no. 1, pp. 18–39, Jan, 2008.

[7] A. Shashua and A. Levin, “Linear image coding for regression and classifcation using the tensor-rank principle,” in Pro ceedings of the IEEE Comput. Soc.Conf. Comput. Vis. Pattern Recog, vol. 1, pp. 42–49, Kauai, HI, USA, December 2001.

[8] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos, “Incremental tensor analysis: theory and applications,” ACM Transactions on Knowledge Discovery from Data, vol. 2, no. 3, pp. 1–37, 2008.

[9] J. Yang, D. Zhang, A. F. Frangi, and J.-Y. Yang, “Two-di mensional Pca: a new approach to appearance-based face representation and recognition,” IEEE Transactions on Pat tern Analysis and Machine Intelligence, vol. 26, no. 1, pp. 131–137, Jan, 2004.

[10] J. Ye, “Generalized low rank approximations of matrices,” in Proceedings of the Twenty-First International Conference on Machine Learning, pp. 887–894, Banf, AB, Canada, July 2004.

[11] J. Dean and S. Ghemawat, “MapReduce: simplifed data processing on large clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[12] S. Ghemawat, H. Gobiof, and S.-T. Leung, “Te Google fle system,” in Proceedings of the 19th ACM Symposium on Operating Systems Principles, pp. 29–43, ACM, Bolton Landing, NY, USA, June 2003.

[13] P. Mika and G. Tummarello, “Web semantics in the clouds,” IEEE Intelligent Systems, vol. 23, no. 5, pp. 82–87, 2008.

[14] Apache Hadoop, “Apache Hadoop,” 2014, http://hadoop. apache.org/ Accessed.

[15] “Mahout,” 2014, http://mahout.apache.org/ Accessed.