International Journal of Leading Research Publication (IJLRP)
E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

Migrating Spark Jobs from On-Premises to GCP Cloud Dataproc

Suhas Hanumanthaiah

suhas.h@hotmail.com

Abstract:
The growing demand for scalable, cost-efficient, and agile data processing solutions has driven organizations to migrate big data workloads from on-premises environments to cloud platforms. Apache Spark, a widely adopted distributed computing framework, plays a pivotal role in processing large-scale datasets, and its migration to the cloud has become a strategic imperative. This research paper provides a comprehensive exploration of migrating Apache Spark jobs to Google Cloud Platform (GCP) using Dataproc—a fully managed, scalable, and cost-effective service for Spark and Hadoop workloads. The study evaluates various migration strategies, including lift-and-shift, cloud-native re-architecting, and hybrid approaches, while analyzing critical factors such as resource management, job scheduling, storage integration, and configuration optimization. Emphasis is placed on performance tuning through intelligent frameworks, zero-execution configuration techniques, and reinforcement learning-based optimization, all of which significantly enhance Spark performance in the cloud. Real-world case studies from domains such as healthcare, bioinformatics, and real-time analytics illustrate practical benefits including performance gains, operational efficiency, and improved scalability. The paper concludes by offering best practices for successful migration, recommendations for production readiness, and insights into future trends such as serverless computing, AI integration, and edge convergence. These findings provide a robust foundation for enterprises planning to modernize their big data infrastructure through cloud migration.

Keywords: Apache Spark, Google Cloud Platform (GCP), Dataproc, Cloud Migration.

1. INTRODUCTION
1.1 Background
The Cloud computing paradigm has revolutionized the computer science horizon during past decade and enabled emergence of computing as fifth utility [1]. Organizations are increasingly recognizing the need to modernize their big data infrastructure to remain competitive and scalable. Apache Spark is one of the most widely used open-source processing framework for big data, it allows to process large datasets in parallel using number of node [3].
Traditional on-premises Spark deployments face several limitations including fixed infrastructure costs, limited scalability, and maintenance overhead. Cloud computing services eliminate various requirements, such as dedicated space and maintenance expensive hardware software [4]. The migration to cloud platforms like GCP Dataproc offers organizations the opportunity to leverage elastic computing resources, managed services, and pay-as-you-go pricing models.

1.2 Research Objectives
This research aims to:
- Analyze the current landscape of Spark migration from on-premises to cloud environments
- Examine GCP Dataproc's capabilities and architecture
- Identify migration strategies, challenges, and best practices
- Evaluate performance considerations and optimization techniques

- Provide practical recommendations for successful migration

2. LITERATURE REVIEW

2.1 Cloud Computing Evolution and Apache Spark

Cloud computing has instigated (1) shorter establishment times for start-ups, (2) creation scalable global enterprise applications, (3) better cost-to-value associativity scientific high-performance computing, and (4) different invocation/execution models pervasive ubiquitous applications [1]. The evolution of cloud computing has created new opportunities for big data processing frameworks.

Apache Spark is one of the most well-known, widely used, and open-source distributed processing framework, which is proven to quickly handle enormous volumes of remotely sensed data [5]. Research has demonstrated Spark's effectiveness across various domains, from healthcare data analysis where correct analysis of such data will improve the quality of care and reduce costs [6] to medical imaging applications using Apache Spark on high-performance computing and Google Cloud Platform [7].

2.2 Cloud Migration Drivers

Organizations are increasingly harnessing advanced management technologies such as artificial intelligence and cloud computing to overcome challenges in managing modern transactions and vast data volumes [8]. While these technologies bring benefits like faster processing and reduced operational costs, they also pose security integration complexities. [8] Looking ahead, organizations envision more integrated, responsive, secure ecosystem that leverages convergence of AI, computing, and storage.

2.3 Distributed Computing Architectures

Research offers comparative panorama of Central Cloud, Distributed Cloud Architectures, and Collaborative Computing Strategies. [9][9] For evaluation, researchers compare architectures with 8 criteria (User Proximity, Latency & Jitter, Network stability, high throughput, Reliability, Scalability, Cost Effectiveness, Maintainability), analyzing advantages and disadvantages of each.

3. GOOGLE CLOUD DATAPROC OVERVIEW

3.1 Service Architecture and Capabilities

Dataproc is a fast, easy-to-use, low-cost and fully managed service that lets you run the Apache Spark and Apache Hadoop ecosystem on Google Cloud Platform [10]. Dataproc automation helps you create clusters quickly, manage them easily, and save money by turning clusters off when you don't need them. [2] With less time and money spent on administration, jobs and data can be focused on.

Dataproc integrates with several other Google Cloud Platform services and migrates Hadoop processing jobs to the cloud [11]. The service provides seamless integration with other GCP services including:

- Google Cloud Storage for data persistence
- BigQuery for data warehousing
- Cloud Monitoring and Logging for operations
- Identity and Access Management (IAM) for security

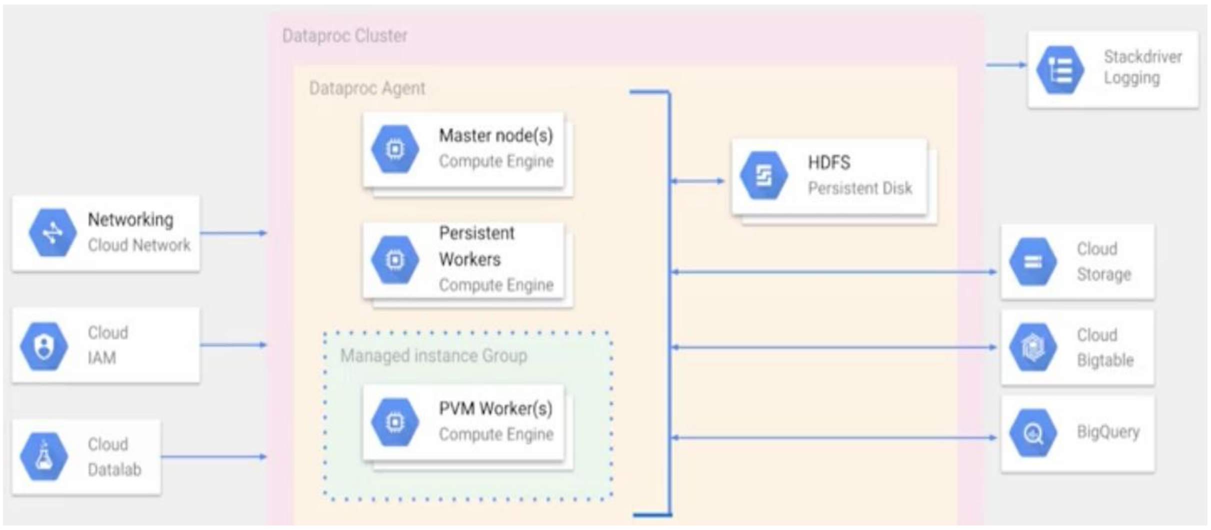

Figure 1: Cloud Dataproc Architecture [10]

3.2 Performance Characteristics

Research has demonstrated significant performance improvements when using Dataproc. Studies have shown speedup factors of 7.12, 13.17 and more in standalone Spark clusters and Google cloud Dataproc clusters respectively, with an increase in the number of cores [5]. Performance evaluations on Google Cloud Platform show average speed increases of 28.56 times on four homogeneous computing nodes [7].

3.3 Multi-tenancy and Scalability

Google Dataproc clusters can be configured to allow autoscaling based on workload. But Google recommends not to use the default auto scaling on Spark Structured Streaming. gcloud command line can be used to scale up and down without any down time.[12]

4. MIGRATION STRATEGIES AND METHODOLOGIES

4.1 Lift-and-Shift Approach

The lift-and-shift approach involves moving existing Spark applications to Dataproc with minimal code changes. Organizations can take advantage of Cloud Dataproc to read and process data from numerous IoT devices and then analyze and predict business opportunities. [11] Dataproc image versions are an important part about how the service works, as Cloud Dataproc uses images to merge Google Cloud Platform connectors with Apache Spark.

4.2 Re-architecting for Cloud-Native Benefits

Cloud computing services provide a powerful environment to store large volumes of data. [4][4] Handling big data is a time-consuming task that requires computational clusters to ensure successful storage processing. Cloud-native re-architecting involves:

- Leveraging managed services for enhanced reliability
- Implementing auto-scaling capabilities
- Optimizing for cloud storage integration
- Adopting serverless computing patterns where appropriate

4.3 Hybrid Migration Strategies

Research addresses questions about which storage processing architectures are best suited for applications and whether generic architectures can meet specific application cases [9]. Hybrid approaches allow organizations to maintain critical workloads on-premises while migrating suitable workloads to the cloud.

5. TECHNICAL IMPLEMENTATION CONSIDERATIONS

5.1 Data Storage and Integration

Google Cloud Storage provides object storage for any amount of data with unlimited storage, low latency and worldwide accessibility and availability. [11] Google Cloud Storage is overtaking the on-premises Hadoop Distributed File System (HDFS) deployments at a fast pace, and should be used as the primary data source.

5.2 Resource Management and Optimization

Applications often use resource management systems like YARN, which provide jobs specific amounts of resources for their execution. [3][3] Studies implement different machine learning algorithms, then manage resources (CPU, memory, and Disk) to assess performance of Spark.
Research shows that tuning resource allocation is crucial - not only allocating all available resources gets better performance, but it depends on how you tune the allocation. [3] Proposed managed parameters show they give better total execution time than default settings.

5.3 Job Scheduling Optimization

Research comparing FIFO and Fair schedulers on Spark MLlib tasks reveals that the Fair scheduler consistently outperforms the FIFO scheduler, particularly as cloud clusters scale from small to moderate sizes. [13] The Fair scheduler's ability to dynamically allocate resources and manage concurrent jobs more efficiently leads to reduced execution times and better overall cloud cluster utilization.

6. PERFORMANCE ANALYSIS AND OPTIMIZATION

6.1 Scalability Considerations

Performance analysis includes scalability evaluation analyzing speedup and execution time. [3] Studies deduce that beyond certain numbers of nodes in a cluster, it's no longer necessary to add additional nodes to improve execution time.

6.2 Storage Optimization

Efficient data placement is challenging since I/O density is usually unknown at the time data needs to be placed for I/O-intensive temporary intermediate data on SSD and HDD [14]. Analysis of production logs from Google's data centers for data processing pipelines shows that I/O density may be predictable. [14] This suggests that learning-based strategies could extract predictive features for I/O density of temporary files, which could be used to improve storage management efficiency.

6.3 Memory Management

Studies of Persistence in Resilient Distributed Datasets (RDDs) with different algorithms show that specific storage levels give the best execution performance among tested levels [3].

7. AUTOMATED CONFIGURATION PARAMETER OPTIMIZATION

7.1 Intelligent Configuration Framework

Optimal configuration of memory and computational resources is critical for maximizing Apache Spark applications' performance and resource efficiency. [22] An intelligent framework can dynamically generate tailored configuration parameters—including memory allocation, the number of executors, and cores per executor—based on specific characteristics of a Spark job such as data volume and execution plan complexity. The framework offers precise, data-driven recommendations that significantly enhance Spark job performance by leveraging historical job performance data, machine learning models, and heuristic-driven analysis.

8. ADVANCED PARAMETER CONFIGURATION STRATEGIES

8.1 Reinforcement Learning-Based Optimization

Apache Spark provides more than 180 configuration parameters for users to manually select appropriate parameter values. [24] However, due to the large number of parameters and the inherent correlation between them, manual tuning is very tedious.

Reinforcement Learning Implementation: A reinforcement-learning-based Spark configuration parameter optimizer first trains a Spark application performance prediction model with deep neural networks, verifying the accuracy and effectiveness from multiple perspectives. [24] To improve search efficiency for better configuration parameters, an improved Q-learning algorithm automatically sets start and end states in each iteration of training, effectively improving the agent's performance in exploring better configuration parameters.

Performance Results: Experimental results show that optimized configurations gained average performance improvements of 47%, 43%, 31%, and 45% for four different types of Spark applications, indicating that the Spark configuration parameter optimizer could efficiently find better configuration parameters and improve performance of various Spark applications [24].

8.2 Multi-Objective Configuration Optimization

Energy-Efficient Tuning: Apache Spark is suitable for multi-objective tuning methods. [25] Different from preceding studies, a new energy-focused optimization algorithm for Spark (EFOS) can be evaluated on the MLlib library of Spark, devised by considering the data processing abilities of Spark.

Multi-Objective Reinforcement Learning: Model-free reinforcement learning for multi-objective optimization of Apache Spark leverages reinforcement learning agents to uncover internal dynamics in hyperparameter optimization. [26] The method runs iterations to update each cell of the RL grid, achieving tradeoffs between three objective functions comprising time, memory, and accuracy. Optimal hyperparameter values are obtained via ensemble techniques analyzing individual results from each objective function.

Optimization Method	Performance Improvement	Implementation Complexity	Computational Overhead
Fair Scheduling	Reduced execution times and better overall cloud cluster utilization	Medium	Low
Reinforcement Learning	47%, 43%, 31%, and 45% for four different types of Spark applications [24]	High	Medium
Multi-Objective RL	37% better speedup than competitors [26]	Very High	Medium

Table 1: Comparative Analysis of Spark Optimization Techniques

9. BEST PRACTICES AND RECOMMENDATIONS

9.1 Migration Planning

Assessment Phase: Evaluate current on-premises workloads and dependencies

Proof of Concept: Deploy pilot projects following established instructions within 20 minutes to start analyzing datasets using free credits provided by cloud platforms. [16] This approach is suitable for pilot testing and preliminary study while reserving flexibility and scalability for large-scale projects

Gradual Migration: Implement phased migration approaches

9.2 Technical Best Practices

Cluster Configuration: Be specific about Dataproc cluster image versions as they are important for how the service works [11]

Storage Strategy: Leverage Cloud Storage as primary data source instead of HDFS

Resource Management: Implement proper resource allocation and scheduling policies

Monitoring: Establish comprehensive monitoring and alerting systems

9.3 Operational Excellence

Initialization actions allow automatic script running or component installation when creating Dataproc clusters. [17] When using cluster initialization actions in production environments, copy initialization scripts to Cloud Storage rather than using external sources.

9.4 Production-Ready Configuration Templates

9.4.1. Large-Scale Data Processing Configuration

```
# Optimized configuration for large datasets spark.executor.memory=12g spark.executor.cores=5
spark.executor.instances=20 spark.driver.memory=4g spark.driver.cores=2 spark.sql.adaptive.enabled=true
spark.sql.adaptive.coalescePartitions.enabled=true
spark.serializer=org.apache.spark.serializer.KryoSerializer spark.sql.execution.arrow.pyspark.enabled=true
```

9.4.2. Multi-GPU Support Configuration

DeepVariant-on-Spark optimizes resource allocation and enables multi-GPU support to accelerate processing pipelines [16].

```
# GPU-accelerated configuration
spark.rapids.sql.enabled=true
spark.plugins=com.nvidia.spark.SQLPlugin
spark.executor.resource.gpu.amount=1
spark.task.resource.gpu.amount=0.25
```

9.5. Automated Configuration Management

9.5.1 CI/CD Integration for Configuration

Automating the traditionally manual and time-consuming tuning process improves resource utilization and job efficiency. [22] It empowers users to achieve optimal configurations without requiring deep expertise in Spark tuning.

Configuration Management Pipeline:

```
# dataproc-config.yaml
cluster_config:
  software_config:
    properties:
      "spark:spark.executor.memory": "8g"
      "spark:spark.executor.cores": "4"
      "spark:spark.dynamicAllocation.enabled": "true"
      "spark:spark.sql.adaptive.enabled": "true"
```

9.5.2 Performance-Based Auto-tuning

Model-free reinforcement learning approaches produce speedup that is 37% better than competitors on average. [26] The approach is very competitive in terms of converging to high accuracy when optimizing Convolutional Neural Networks.

9.6. Multi-Tenant Security Implementation

Use cloud-native technologies like Docker and Kubernetes to solve problems of performance isolation, security, and scaling in multi-tenant environments while enabling secure data sharing and data privacy for collaboration [27].

```
# Security configuration
spark.authenticate=true
spark.network.crypto.enabled=true
spark.io.encryption.enabled=true
spark.sql.execution.arrow.pyspark.enabled=true
```

10. CASE STUDIES AND REAL-WORLD APPLICATIONS

10.1 Bioinformatics Applications

Research demonstrates big data processing solutions using Apache Spark with Hadoop Distributed File System as data storage on clusters. [18] Studies compare string matching process time between stand-alone systems and Apache Spark deployments with 3, 5, 11, and 16 nodes using Hadoop Distributed File System storage on Google Cloud Platform clusters.

10.2 Decision-Making Systems

Experiments carried out within GCP's Dataproc service, which allows execution of Apache Hadoop and Spark tasks on Google Cloud, obtained very significant and promising results [19].

10.3 Healthcare and Scientific Computing

Research presents optimized processing pipelines to enable multi-GPU support and accelerate processing. [16] These solutions are deployed on Cloud Platform (GCP) to make processing more accessible to everyone.

10.4 Healthcare Data Processing

Parallel microwave image reconstruction algorithms based on Apache Spark on high-performance computing and Google Cloud Platform show average speed increases of 28.56 times on four homogeneous computing nodes, with input data converted to resilient distributed datasets and distributed to multiple cluster nodes [7].

10.5 Stream Processing Applications

Apache Spark-based Big Data Streaming processing systems can be deployed over Open Platform for Functions Virtualization, providing accurate real-time threat detection services [23].

11. FUTURE DIRECTIONS AND EMERGING TRENDS

11.1 Serverless Computing Integration

Recent technological developments and paradigms such as serverless computing, software-defined networking, Internet of Things, and processing at network edge are creating new opportunities for cloud computing [1].

11.2 AI and Machine Learning Integration

Real-time data analysis involving new tools such as Apache Kafka and Spark Streaming coupled with emerging AI techniques including Graph Neural Networks, NLP, reinforcement learning, transfer learning, and quantum computing. [20] Comparative analysis with Google Cloud AI Platform and its AI tools shows efficiency in real-world applications.

11.3 Edge Computing Convergence

Edge computing is a revolutionary extension of traditional cloud computing, expanding central infrastructure with execution environments close to users in terms of latency to enable new generation applications [21].

12. CONCLUSION

The migration of Apache Spark jobs from on-premises infrastructure to GCP Dataproc represents a significant opportunity for organizations to achieve enhanced scalability, cost optimization, and operational efficiency. This comprehensive analysis reveals that successful migration requires careful planning, appropriate technical strategies, and adherence to established best practices.

Key findings include:

1. Performance Benefits: Research demonstrates significant performance improvements with speedup factors exceeding 13x in Dataproc environments and average speed increases of 28.56 times on optimized configurations [5][7].
2. Architectural Considerations: Google Cloud Storage provides superior performance characteristics compared to traditional HDFS deployments, offering unlimited storage, low latency, and worldwide accessibility [11].

3. Operational Excellence: Dataproc's automation capabilities enable quick cluster creation, easy management, and cost savings through intelligent resource management [2].
4. Scalability and Resource Management: Fair scheduling algorithms and proper resource allocation strategies are critical for optimizing performance in cloud environments [13][13].

Organizations planning migration should adopt a phased approach, beginning with pilot projects and gradually scaling to full production workloads. The integration of emerging technologies such as AI, edge computing, and serverless architectures will continue to shape the evolution of cloud-based big data processing platforms. Future research should focus on automated migration tools, advanced cost optimization algorithms, and integration patterns for hybrid cloud architectures. Strategies for future-proofing database management with robust data governance, continuous learning, stakeholder collaboration, and adaptability to evolving technologies [20] will be essential for long-term success.

REFERENCES:

[1] R. Buyya et al., "A Manifesto for Future Generation Cloud Computing," Association for Computing Machinery, 2018. https://doi.org/10.1145/3241737
[2] G. Cloud, "Dataproc overview | Dataproc Documentation | Google Cloud," internet, n.d..
[3] K. Aziz, D. Zaidouni, M. Bellafkih, "Leveraging resource management for efficient performance of Apache Spark," Springer Science+Business Media, 2019. https://doi.org/10.1186/s40537-019-0240-1
[4] A. K. Sandhu, "Big data with cloud computing: Discussions and challenges," Tsinghua University Press, 2022. https://doi.org/10.26599/bdma.2021.9020016
[5] B. Rupa, A. Negi, "Scalable SubXPCA with Extended Morphological Profiles Applied to Hyperspectral Image Classification on a Distributed Platform," BigData Congress [Services Society], 2024. https://doi.org/10.1109/BigData62323.2024.10825513
[6] E. Nazari, M. Shahriari, H. Tabesh, "BigData Analysis in Healthcare: Apache Hadoop , Apache spark and Apache Flink," Frontiers in Health Informatics, 2019. https://doi.org/10.30699/FHI.V8I1.180
[7] R. Ullah, T. Arslan, "PySpark-Based Optimization of Microwave Image Reconstruction Algorithm for Head Imaging Big Data on High-Performance Computing and Google Cloud Platform," Applied Sciences, 2020. https://doi.org/10.3390/app10103382
[8] S. Ionescu, V. Dacona, "Transforming Financial Decision-Making: The Interplay of AI, Cloud Computing and Advanced Data Management Technologies," Agora University, 2023. https://doi.org/10.15837/ijccc.2023.6.5735
[9] O. Debauche, S. Mahmoudi, P. Manneback, F. Lebeau, "Cloud and distributed architectures for data management in agriculture 4.0 : Review and future trends," Elsevier BV, 2021. https://doi.org/10.1016/j.jksuci.2021.09.015
[10] G. Cloud, "Dataproc FAQ | Dataproc Documentation | Google Cloud," internet, n.d..
[11] NetApp, "Google Cloud Dataproc Best Practices," internet, n.d..
[12] Google, “Scale Dataproc clusters” internet, n.d.. https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/scaling-clusters
[13] K. Sai et al., "Comparison of Fair and FIFO Schedulers on Cloud Cluster of Apache Spark," None, 2024. https://doi.org/10.1109/ICEI64305.2024.10912108
[14] U. U. Hafeez, M. Maas, M. Uysal, R. McDougall, "Rethinking Storage Management for Data Processing Pipelines in Cloud Data Centers," arXiv.org, 2022. https://doi.org/10.48550/arXiv.2211.02286
[15] M. Ouhssini, K. Afdel, M. Idhammad, E. Agherrabi, "Distributed intrusion detection system in the cloud environment based on Apache Kafka and Apache Spark," International Conference on the Digital Society, 2021. https://doi.org/10.1109/ICDS53782.2021.9626721
[16] P. Huang et al., "DeepVariant-on-Spark: Small-Scale Genome Analysis Using a Cloud-Based Computing Framework," Hindawi Publishing Corporation, 2020. https://doi.org/10.1155/2020/7231205

[17] G. Cloud, "Dataproc best practices for production | Dataproc Documentation | Google Cloud," internet, n.d..
[18] F. Zhafirah, T. Hidayat, L. Riza, "Apache Spark Implementation on Algorithms Boyer-Moore Horspool for Case Studies Internal Transcribed Spacer and Restriction Enzyme," None, 2024. https://doi.org/10.17509/jcs.v5i1.70790
[19] L. Lamrini, M. C. Abounaima, M. T. Alaoui, "New distributed-topsis approach for multi-criteria decision-making problems in a big data context," Springer Science+Business Media, 2023. https://doi.org/10.1186/s40537-023-00788-3
[20] M. Godly, C. Mathew, M. S. Santhosh, M. Anandhrosh, M. S. Thomas, "Database and Modern Database Technology," None, 2024. https://doi.org/10.47392/irjaem.2024.0546
[21] B. Sonkoly, J. Czentye, M. Szalay, B. Nmeth, L. Toka, "Survey on Placement Methods in the Edge and Beyond," Institute of Electrical and Electronics Engineers, 2020. https://doi.org/10.1109/comst.2021.3101460
[22] V. R. Sarabu, P. Udayaraju, C. Ravulu, S. Kumar, J. Sushma, K. T. V. Narayana, "Developing an Intelligent Framework for Optimizing Apache Spark Tasks to Improve the Efficiency of Memory Configuration," None, 2024. https://doi.org/10.1109/OCIT65031.2024.00010
[23] M. A. Lopez, D. M. F. Mattos, O. C. M. B. Duarte, G. Pujolle, "Toward a monitoring and threat detection system based on stream processing as a virtual network function for big data," Wiley, 2019. https://doi.org/10.1002/cpe.5344
[24] X. Huang, H. Zhang, X. Zhai, "A Novel Reinforcement Learning Approach for Spark Configuration Parameter Optimization," Italian National Conference on Sensors, 2022. https://doi.org/10.3390/s22155930
[25] M. M. Ozturk, V. Nejkovic, "Energy-consumption focused multi-objective tuning of Apache Spark," None, 2024. https://doi.org/10.69994/11ic24034
[26] M. ztrk, "Mfrlmo: Model-Free Reinforcement Learning for Multi-Objective Optimization of Apache Spark," Social Science Research Network, 2024. https://doi.org/10.2139/ssrn.4358735
[27] G. Chikafa, S. Sheikholeslami, S. Niazi, J. Dowling, V. Vlassov, "Cloud-native RStudio on Kubernetes for Hopsworks," arXiv.org, 2023. https://doi.org/10.48550/arXiv.2307.09132

14. Abbreviations

Abbreviation	Full Form
GCP	Google Cloud Platform
HDFS	Hadoop Distributed File System
IAM	Identity and Access Management
RDD	Resilient Distributed Dataset
CPU	Central Processing Unit
GPU	Graphics Processing Unit
MLlib	Machine Learning Library (in Spark)
CI/CD	Continuous Integration / Continuous Deployment
YAML	Yet Another Markup Language
NLP	Natural Language Processing
AI	Artificial Intelligence
RL	Reinforcement Learning
EFOS	Energy-Focused Optimization Strategy
MFRLMO	Model-Free Reinforcement Learning for Multi-Objective Optimization
SSD	Solid-State Drive

HDD	Hard Disk Drive
FIFO	First In, First Out (Scheduling)
Q-Learning	Quality Learning (a type of reinforcement learning algorithm)