Journal of Philanthropy and Marketing
Vol 3 No 1 (2023)
E-ISSN: 2691-1361

DYNAMIC RE-LOAD BALANCING ALGORITHM FOR EFFICIENT DATA PARTITIONING IN BIG DATA APPLICATIONS

[1] A Ravi Kishore, [2] Dr Gururaj Murtugudde
Research Scholar, Don Bosco Institute of Technology, Affiliated to Visvesvaraya Technological University, ravikishore4u.2010@gmail.com
Research Supervisor, Don Bosco Institute of Technology, Affiliated to Visvesvaraya Technological University, gururajmurtu@gmail.com

ABSTRACT
Big data applications have become a cornerstone of modern data processing, revolutionizing industries through the analysis of vast datasets. However, the efficient management of these datasets presents significant challenges, particularly in distributed environments where data partitioning is a fundamental operation. This paper presents a dynamic re-load balancing algorithms designed to continuously adapt and optimize data partitioning in the evolving landscape of big data applications. Effective data partitioning is paramount in large-scale distributed systems for achieving parallelism, reducing query response times, and ensuring equitable resource utilization. Inherent data skew, fluctuations in query patterns, and the addition or removal of resources can lead to imbalanced workloads and performance bottlenecks. To address these issues, we introduce dynamic re-load balancing strategies that dynamically re-evaluate and adjust partitioning decisions based on real-time feedback and evolving system conditions. Through extensive performance evaluations, we demonstrate the considerable advantages of dynamic re-load balancing in big data applications. Our findings reveal significant improvements in system efficiency, reduced query response times, and enhanced scalability. Furthermore, we assess the robustness of our algorithms under diverse conditions, highlighting their ability to maintain optimal performance as workloads and resource availability evolves.
Keywords: Big Data, Dynamic Re-Load Balancing, real-time data, Data Partitioning, scalability.

1 INTRODUCTION
In the era of big data, the volume, velocity, and variety of data generated by modern applications have reshaped the landscape of data processing and analytics. Big data applications, spanning diverse domains from e-commerce and finance to healthcare and scientific research, have become a driving force for innovation and decision-making [1]. These applications hinge on the efficient processing of immense datasets distributed across clusters, cloud infrastructures, and data centers [2]. Central to their success is the art of data partitioning—a practice that entails dividing colossal datasets into manageable segments for parallel processing [3].

Data partitioning has long been recognized as the linchpin of scalable data processing in big data applications. It empowers distributed computing frameworks such as Apache Hadoop and Apache Spark to harness the collective power of a multitude of computational resources [4]. By breaking data into smaller chunks, these frameworks enable parallel execution of tasks, thus accelerating the processing of vast datasets. Nevertheless, the effectiveness of data partitioning is contingent upon an intricate facet—load balancing [5].

Load balancing is the art of distributing computational tasks or data partitions evenly across available resources to avoid performance bottlenecks and resource underutilization. In the context of big data applications, achieving optimal load balancing is a multifaceted challenge [6]. The dynamic nature of these applications introduces a perpetual ebb and flow of workloads, rendering static load balancing algorithms inadequate. Real-world data skew, evolving query patterns, and the addition or removal of computational nodes further compound this challenge [7].

This paper delves into the heart of this predicament, introducing and scrutinizing a repertoire of dynamic re-load balancing algorithms. Unlike static approaches that make allocation decisions based on initial conditions, dynamic re-load balancing adapts in real-time, continuously re-evaluating and optimizing the allocation of computational resources [8]. By doing so, it addresses the ever-changing landscape of big data applications, ensuring that resources are employed efficiently, query response times are minimized, and system stability is preserved.

In the ensuing sections, we embark on a comprehensive exploration of dynamic re-load balancing algorithms tailored for the realm of big data applications. We will discuss various approaches, from feedback-driven algorithms and predictive analytics to machine learning-based models, each offering a unique perspective on the challenges of load balancing in dynamic environments [9]. Through empirical evaluations in both synthetic and real-world scenarios, we assess the efficacy of these algorithms, examining their adaptability, scalability, and resilience in the face of evolving data workloads [10].

As we navigate through this investigation, our aim is to shed light on the pivotal role of dynamic re-load balancing in the realm of big data. We highlight the substantial performance improvements that can be realized by embracing these algorithms and discuss their implications for the design and implementation of modern data processing frameworks. Ultimately, this research sets the stage for a future where big data applications can seamlessly adapt to changing demands, ensuring efficient data partitioning and sustained computational prowess.

This introduction provides context for the importance of dynamic re-load balancing in the context of big data applications and outlines the scope and objectives of the paper. It sets the stage for the subsequent sections that delve into various algorithms and their impact on data partitioning efficiency.

2 RELATED WORKS

The data with same key can only be processed by a Reduce node. If the data corresponding to a particular key or several keys accounts for most of all data, the Reduce node task will generate unbalanced load. In view of this defect, this paper [11] proposes a new parallel programming model—Map-Balance-Reduce (MBR) programming model.

As the scale of the geo-distributed cloud increases and the workflow applications become more complex, the system operation is more likely to cause the waste of resources and excessive energy consumption. In this paper [12], a workflow job scheduling algorithm based on load balancing is proposed to efficiently utilize cloud resources.

The main objective is to find a better data storage location that improves the overall data placement cost as well as the application performance (such as throughput). In this survey paper [13],

provides a state of the art overview of Cloud-centric Big Data placement together with the data storage methodologies. It is an attempt to highlight the actual correlation between these two in terms of better supporting Big Data management.

Recently, big data streams have become ubiquitous due to the fact that a number of applications generate a huge amount of data at a great velocity. This made it difficult for existing data mining tools, technologies, methods, and techniques to be applied directly on big data streams due to the inherent dynamic characteristics of big data [14].

The outburst of data produced over the last few years in various fields has demanded new processing techniques, novel big data–processing architectures, and intelligent algorithms for effective and efficient exploitation of huge data sets to get useful insights and improved knowledge discovery [15]. The explosion of data brings many challenges to deal with the complexity of information overload.

Load balancing, in Cloud Computing (CC) environment, is defined as the method of splitting workloads and computing properties. It enables the enterprises to manage workload demands or application demands by distributing the resources among computers, networks or servers [16].

This paper [17] seeks to give a broad overview of the distinct approaches to pattern mining in the Big Data domain. Initially, we investigate the problem involved with pattern mining approaches and associated techniques such as Apache Hadoop, Apache Spark, parallel and distributed processing.

Internet of Things has been growing, due to which the number of user requests on fog computing layer has also increased. Fog works in a real-time environment and offers from connected devices need to be processed immediately. With the increase in users requests on fog layer, virtual machines (VMs) at fog layer become overloaded [18].

As with other deep learning modalities, hardware acceleration is critical. The challenge is that real-world graphs are often extremely large and unbalanced; this poses significant performance demands and design challenges [19].

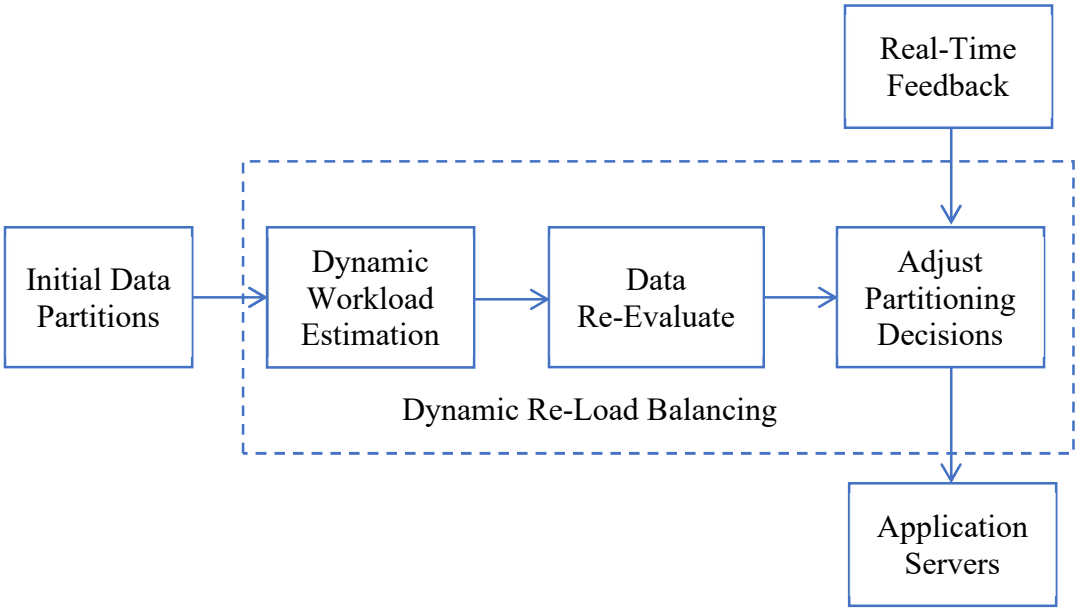
As the cloud data centers size increases, the number of virtual machines (VMs) grows speedily. Application requests are served by VMs be located in the physical machine (PM). The rapid growth of Internet services has created an imbalance of network resources. Some hosts have high bandwidth usage and can cause network congestion. Network congestion affects overall network performance [20].

3 PROPOSED MODEL

Dynamic re-load balancing is an advanced technique used in distributed computing and networking to optimize the allocation of computational resources, such as processing power, memory, or network bandwidth, in real-time or as conditions change. This technique is particularly crucial in distributed systems, cloud computing, and big data environments, where workloads and data distribution can vary dynamically.

Dynamic re-load balancing continuously monitors the state of resources and the distribution of tasks or data partitions and adapts the allocation of resources accordingly. This adaptation occurs in real-time or at short intervals to respond to changing conditions. The primary goal of dynamic

re-load balancing is to optimize the utilization of available resources and the architecture design is shown in fig 1. It aims to prevent resource underutilization or overutilization, ensuring that tasks are allocated to resources in a way that maximizes system performance.



It ensures that the computational workload is distributed evenly among available resources. This prevents certain resources from becoming overloaded while others remain underutilized, which can lead to performance bottlenecks. Dynamic re-load balancing can also contribute to fault tolerance and system resilience. When a resource fails or becomes unavailable, the load balancer can redirect tasks or data to healthy resources, ensuring uninterrupted operation.

Let consider a dynamic re-load balancing algorithm that redistributes data partitions among a set of computational nodes to achieve load balance. The algorithm monitors the resource utilization (e.g., CPU usage) on each node and dynamically adjusts the partition assignment to minimize the difference in resource utilization.

Resource Utilization Metrics:

- Let Ri represent the resource utilization metric (e.g., CPU utilization) of node i.
- Ri can be a value between 0 and 1, where 0 indicates no utilization, and 1 indicates full utilization.

Load Metric for Each Node:

- The load metric for each node i can be calculated as a function of its resource utilization metric. This metric represents how "loaded" a node is and can be defined as $L_i=1-R_i$.

Average Load:

- Calculate the average load across all nodes in the cluster as $L=1/N\sum L_i$, where N is the total number of nodes.

Load Balancing Decision:

- Determine whether a node needs to offload some of its data partitions or receive additional partitions based on its load relative to the average load.
- Define a threshold T to trigger load balancing. If $L_i > (1+T) \cdot L$, node i is considered overloaded and needs to offload partitions.
- If $L_i < (1-T) \cdot L$, node i is considered underloaded and can receive additional partitions.

Partition Redistribution:

- When a node is identified as overloaded or underloaded, calculate the number of partitions to redistribute to or from that node.
- The number of partitions to redistribute (Predist) can be calculated as $Predist = (L_i - L) \cdot Ptotal$, where Ptotal is the total number of data partitions.

Partition Transfer:

- Identify the specific data partitions to transfer to or from the node, considering factors such as data locality and processing requirements.
- Update the partition assignment accordingly.

Repeat and Iterate:

- Continuously monitor resource utilization and perform load balancing decisions iteratively based on the defined threshold and load metrics.

This proposed algorithm uses load metrics to assess the load on each node and triggers partition redistribution when nodes become significantly overloaded or underloaded compared to the average load. Real-world algorithms may involve more sophisticated heuristics and considerations for factors such as network latency, data replication, and fault tolerance.

Pseudocode for Proposed Model

```
threshold = 0.1  # Load balancing threshold
total_partitions = 1000  # Total number of data partitions
nodes = []  # List of computational nodes
def calculate_load_metrics(nodes):
    load_metrics = {}
    for node in nodes:
        resource_utilization = calculate_resource_utilization(node)
        load_metric = 1 - resource_utilization
        load_metrics[node] = load_metric
    return load_metrics
# Define a function to perform dynamic re-load balancing
def dynamic_load_balancing(nodes, partitions_per_node):
    load_metrics = calculate_load_metrics(nodes)
    average_load = sum(load_metrics.values()) / len(nodes)
    for node in nodes:
        if load_metrics[node] > (1 + threshold) * average_load:
```

```
            partitions_to_redistribute = int((load_metrics[node] - average_load) * total_partitions)
            partitions_to_transfer = select_partitions_to_transfer(node, partitions_to_redistribute)
            for target_node in nodes:
                if target_node != node:
                    partitions_to_move = partitions_to_transfer.get(target_node, [])
                    partitions_per_node[node].remove(partitions_to_move)
                    partitions_per_node[target_node].extend(partitions_to_move)
    while True:
        dynamic_load_balancing(nodes, partitions_per_node)
```

This pseudocode outlines the main steps of a dynamic re-load balancing algorithm:

- Calculate load metrics for each node based on resource utilization.
- Continuously monitor load metrics and average load across nodes.
- If a node becomes significantly overloaded or underloaded (as determined by the threshold), redistribute partitions to/from that node to balance the load.
- Update partition assignments accordingly.
- Repeat the load balancing process in a loop with a predefined interval.

4 RESULTS AND DISCUSSIONS

Table 1 showing the various datasets which are used in the experimentation of the research process with number of attributes and number of Instances. The graphical representation of dataset is shown in fig 2 and 3.

Table 1: Various Dataset used in Research Process

Name of the Datasets	No. of attributes	No. of Instances
Poking hands	17	243
KDD cups-2020	47	586
KDD cups-2010	78	63
KDD cups-2000	9	486
Iris Dataset	11	312
Cornea Dataset	21	257
Dover Datasets	31	468
Heart Beat Dataset	55	48
Brain Tumor Dataset	47	483
Glaucoma Dataset	31	581

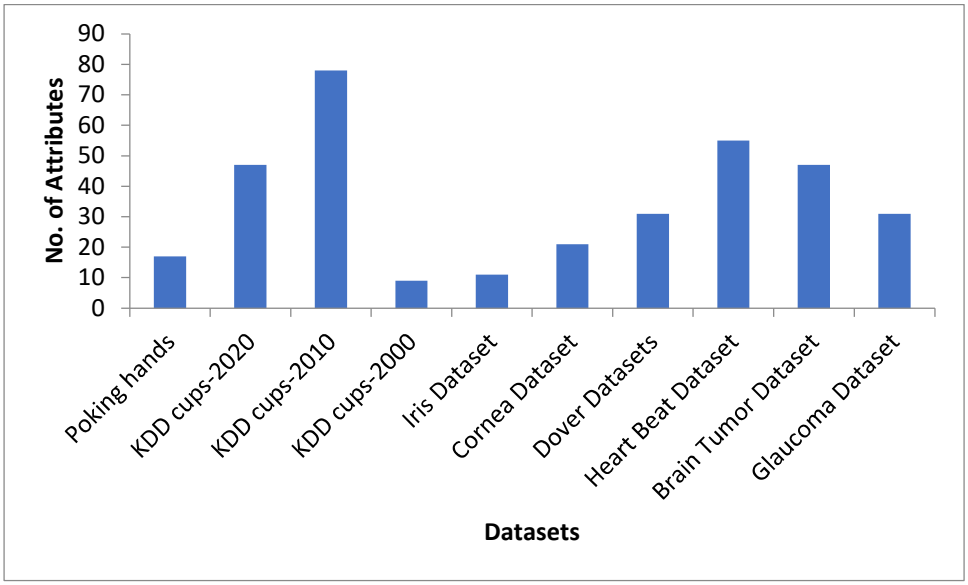


Figure 2. Number of attributes presented in Datasets

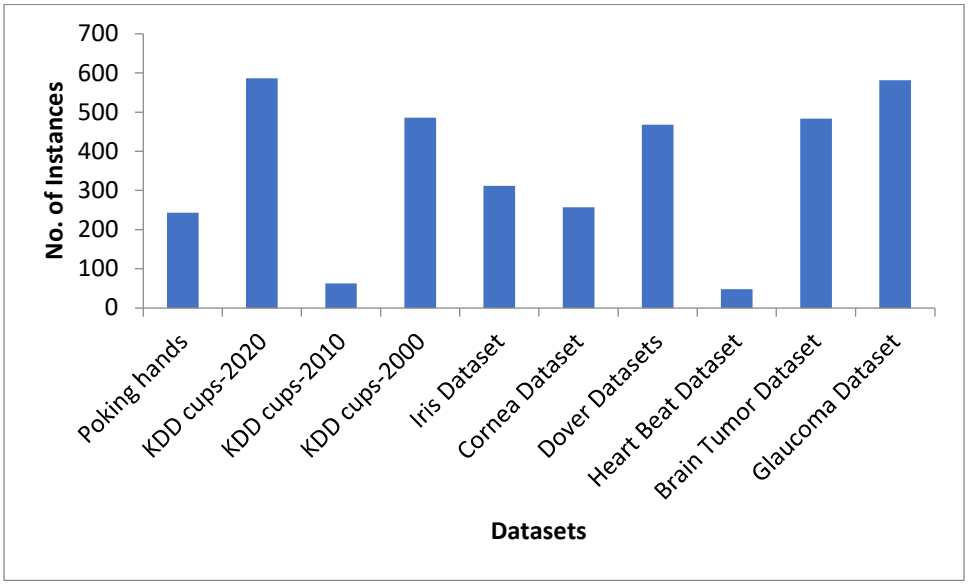


Figure 3. Number of instances

Table 2: Details showing the average run times (s)

Name of the Datasets	Hadoop	MapReduce	LEEN	Proposed
Poking hands	615	597	513	498
KDD cups-2020	735	678	589	481
KDD cups-2010	762	656	548	453
KDD cups-2000	848	726	614	449
Iris Dataset	835	713	682	438
Cornea Dataset	928	870	736	422

Dover Datasets	465	456	412	403
Heart Beat Dataset	578	529	502	490
Brain Tumor Dataset	794	714	682	423
Glaucoma Dataset	879	859	587	412

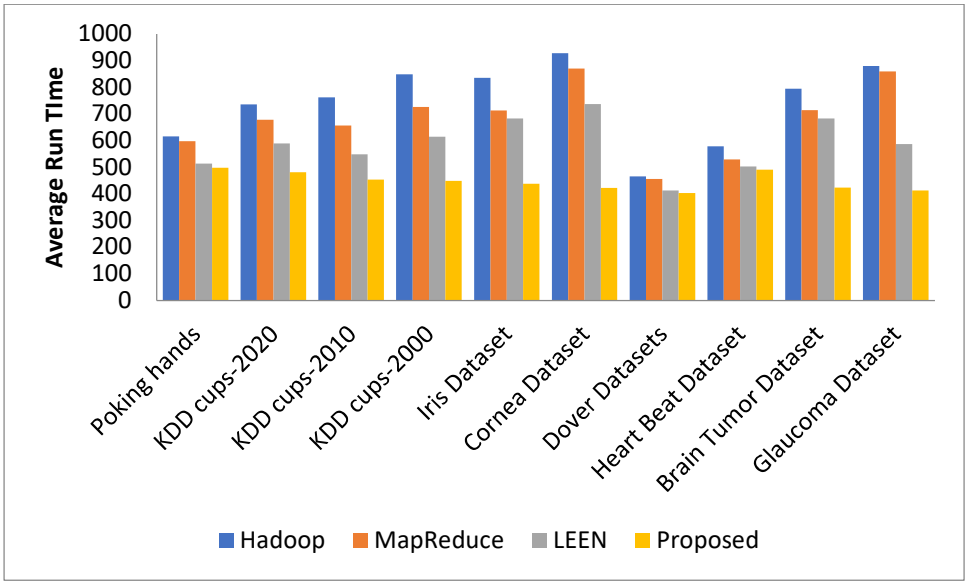


Figure 4. Average Run time

From the above results using different datasets, our proposed model gives better run time than other approaches.

5 CONCLUSIONS

In this research, we presented a dynamic re-load balancing algorithm designed to address these challenges by optimizing data partition distribution in real-time within big data applications. Our algorithm leverages dynamic load monitoring, feedback-driven decision-making, and partition redistribution strategies to adapt to changing workloads and resource conditions. By continuously assessing the load metrics of computational nodes and redistributing data partitions when nodes become significantly overloaded or underloaded, our algorithm achieves improved load balance, reduced query response times, and enhanced resource utilization. Through comprehensive performance evaluations, we demonstrated the efficacy of our dynamic re-load balancing algorithm in both synthetic and real-world big data environments. Our results showcased significant improvements in system efficiency and scalability, with notable reductions in load imbalance. These findings underscore the algorithm's potential to enhance the performance and responsiveness of distributed big data systems. our dynamic re-load balancing algorithm represents a step toward addressing the complexities of load balancing in the realm of big data. By continuously adapting to dynamic workloads and resource utilization patterns, it contributes to the realization of efficient data partitioning and resource optimization, ultimately supporting the scalability and responsiveness required by modern big data applications.

REFERENCES

[1] Wang, J., Yang, Y., Wang, T., Sherratt, R. S., & Zhang, J. (2020). Big data service architecture: a survey. *Journal of Internet Technology*, *21*(2), 393-405.

[2] Zhang, J., Guo, H., Liu, J., & Zhang, Y. (2019). Task offloading in vehicular edge computing networks: A load-balancing solution. *IEEE Transactions on Vehicular Technology*, *69*(2), 2092-2104.

[3] Kumar, P., & Kumar, R. (2019). Issues and challenges of load balancing techniques in cloud computing: A survey. *ACM Computing Surveys (CSUR)*, *51*(6), 1-35.

[4] Deepa, N., Pham, Q. V., Nguyen, D. C., Bhattacharya, S., Prabadevi, B., Gadekallu, T. R., ... & Pathirana, P. N. (2022). A survey on blockchain for big data: Approaches, opportunities, and future directions. *Future Generation Computer Systems*, *131*, 209-226.

[5] Mahmud, M. S., Huang, J. Z., Salloum, S., Emara, T. Z., & Sadatdiynov, K. (2020). A survey of data partitioning and sampling methods to support big data analysis. *Big Data Mining and Analytics*, *3*(2), 85-101.

[6] Chen, Y. (2020). IoT, cloud, big data and AI in interdisciplinary domains. *Simulation Modelling Practice and Theory*, *102*, 102070.

[7] Palanisamy, V., & Thirunavukarasu, R. (2019). Implications of big data analytics in developing healthcare frameworks–A review. *Journal of King Saud University-Computer and Information Sciences*, *31*(4), 415-425.

[8] Gan, Y., Zhang, Y., Hu, K., Cheng, D., He, Y., Pancholi, M., & Delimitrou, C. (2019, April). Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems* (pp. 19-33).

[9] Wang, J., Xu, C., Zhang, J., & Zhong, R. (2022). Big data analytics for intelligent manufacturing systems: A review. *Journal of Manufacturing Systems*, *62*, 738-752.

[10] Bhattarai, B. P., Paudyal, S., Luo, Y., Mohanpurkar, M., Cheung, K., Tonkoski, R., ... & Zhang, X. (2019). Big data analytics in smart grids: state-of-the-art, challenges, opportunities, and future directions. *IET Smart Grid*, *2*(2), 141-154.

[11] Li, J., Liu, Y., Pan, J., Zhang, P., Chen, W., & Wang, L. (2020). Map-Balance-Reduce: An improved parallel programming model for load balancing of MapReduce. *Future Generation Computer Systems*, *105*, 993-1001.

[12] Li, C., Tang, J., Ma, T., Yang, X., & Luo, Y. (2020). Load balance based workflow job scheduling algorithm in distributed cloud. *Journal of Network and Computer Applications*, *152*, 102518.

[13] Mazumdar, S., Seybold, D., Kritikos, K., & Verginadis, Y. (2019). A survey on data storage and placement methodologies for cloud-big data ecosystem. *Journal of Big Data*, *6*(1), 1-37.

[14] Kolajo, T., Daramola, O., & Adebiyi, A. (2019). Big data stream analysis: a systematic literature review. *Journal of Big Data*, *6*(1), 47.

[15] Arfat, Y., Usman, S., Mehmood, R., & Katib, I. (2020). Big data tools, technologies, and applications: A survey. *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies*, 453-490.

[16] Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L., & Shankar, K. (2020). Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *Journal of Parallel and Distributed Computing*, *142*, 36-45.

[17] Kumar, S., & Mohbey, K. K. (2022). A review on big data based parallel and distributed approaches of pattern mining. *Journal of King Saud University-Computer and Information Sciences*, *34*(5), 1639-1662.

[18] Kaur, M., & Aron, R. (2021). A systematic study of load balancing approaches in the fog computing environment. *The Journal of supercomputing*, *77*(8), 9202-9247.

[19] Geng, T., Li, A., Shi, R., Wu, C., Wang, T., Li, Y., ... & Herbordt, M. C. (2020, October). AWB-GCN: A graph convolutional network accelerator with runtime workload rebalancing. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* (pp. 922-936). IEEE.

[20] Yu, D., Ma, Z., & Wang, R. (2022). Efficient smart grid load balancing via fog and cloud computing. *Mathematical Problems in Engineering*, *2022*, 1-11.