Bulletin of Electrical Engineering and Informatics
Vol. 13, No. 1, February 2024, pp. 453~464
ISSN: 2302-9285, DOI: 10.11591/eei.v13i1.5841

The scheduling techniques in the Hadoop and Spark of smart cities environment: a systematic review

Nada Masood Mirza[1,2], Adnan Ali[1], Mohamad Khairi Ishak[1,3]
[1]School of Electrical and Electronic Engineering, Universiti Sains Malaysia (USM), Pulau Pinang, Malaysia
[2]College of Engineering, United Arab Emirates University, Al Ain, United Arab Emirates
[3]Department of Electrical and Computer Engineering, Ajman University, Ajman, United Arab Emirates

Article Info

Article history:

Received Jan 23, 2023
Revised Jul 13, 2023
Accepted Aug 30, 2023

Keywords:

Big data
Hadoop
Scheduling
Smart city
Spark

ABSTRACT

Processing extensive and diverse data in real-time is a significant challenge in the context of smart cities. Timely access to information and efficient analytics is essential for smart city services to make data-driven decisions and enhance urban living. Scheduling algorithms play a crucial role in ensuring the prompt delivery of services and efficient task completion. This paper explores various scheduling techniques, including static, dynamic, and hybrid schedulers, and compares their objectives and performance. Additionally, the study examines two prominent data processing frameworks, Hadoop and Spark, and compares their capabilities in handling big data in smart cities. With its ability to process large amounts of data quickly and efficiently, Spark has shown superiority over Hadoop in real-time data processing and performance optimization. The paper concludes by highlighting the strengths and limitations of each framework. It discusses the need for further research in optimizing scheduling techniques and exploring hybrid artificial intelligence scheduling for Spark. Overall, the findings contribute to a better understanding of data processing in real-time and provide insights for researchers and practitioners in smart cities.

Corresponding Author:

Mohamad Khairi Ishak
School of Electrical and Electronic Engineering, Universiti Sains Malaysia (USM)
Nibong Tebal, Pulau Pinang, 14300, Malaysia
Email: khairiishak@usm.my

1. INTRODUCTION

In today's world, data growth is reaching new highs based on the recent statistics that by the year 2050, almost 70% of the world population will live in cities. Due to this very reason development of smart cities is extremely necessary. It will be possible to provide smart, efficient, and enhanced solutions by building these smart cities. This all can be done by a smart structure built up. As the towns are getting converted into smart domain form and there is advent in other forms of modern technology, that leads to the rise of the smart city (SC) is gaining much attention; it is now being seen as a new paradigm of intelligent city development and sustainable socio-economic growth [1], [2]. To enhance the quality of life, smart city proposes a novel approach to the design and operation of urban infrastructure, including infrastructure for housing, transportation, public services, utilities, health care, and more.

Smart cities are those in which human capital and information and communication technology investments lead to long-term economic development and good quality of life [3]. Cities are therefore necessary for tackling significant public and financial challenges, such as low carbon expansion, emission reduction, energy efficiency, shared energy resources, economic development, and more [4]. The reason behind moving to smart cities is that they can provide services on a citizen-demand basis. In this way, their

needs are responded to in a better way by the organizations and businesses. Two basic requirements to attain these personalized services include the ability to understand the user's current needs and to adapt later on concerning changes in the user's behavior. To serve this purpose, data analysis is needed. However, the precise timing in the placement and occurrence of this analysis is highly crucial. This smart setup must become a reality by appropriately utilizing internet-embedded devices. These devices include sensors and electronics capable of communicating with each other via a network. However, these devices generate a massive amount of heterogeneous data named big data [5]–[7].

The number of devices that are data-producing in smart cities has expanded dramatically throughout the globe, and it will not be wrong to mention that smart cities are one of the primary sources [8]. Since then, the world's information output has skyrocketed, leading to a new phenomenon known as big data. Big data is a term used to describe very massive and complicated datasets that cannot be processed using conventional methods [9]. Such an extensive data set is a significant barrier to traditional data processing methods. Google launched one of the practical frameworks for processing massive data, MapReduce, in 2004 [10], [11]. It is scalable, dependable, and has excellent fault tolerance. In addition, Apache Hadoop is a free, open-source software framework. This framework has dominated big data analysis due to its popularity in many areas, such as the utilization of all the possible hardware resources available regardless of the computing resource from a single server to thousands of serves, a Huge amount of data processed in parallel, fault tolerance, and network load balancing. Companies such as Google, Facebook, and Amazon, have a vast amount of data that require processing to filter out valuable data. Handling this massive amount of data from smart cities is a byword in the current computing area. Since conventional boundaries of the smart city have expanded, allowing for predicting emergency events and real-time management using new technology in an innovative city system, both of which were previously impossible to achieve. Because competent resources are so crucial in the aftermath of an incident, the effectiveness with which they are allocated and scheduled is a critical indicator of any response capability [12], [13]. Many researchers are working to find ways and means to handle this big data efficiently.

This paper's significant contribution is to examine scheduling techniques in Hadoop and Spark that may be applied in a Smart Cities Environment. This review will fulfill the following objectives: i) provide an overview of smart cities, including their significance and benefits; ii) discuss and analyze the challenges of processing the massive amounts of data smart cities generate; iii) a detailed comparison of Hadoop and Spark scheduling techniques for big data analysis; iv) identify research gaps in current data processing techniques, future research directions and open research issues in real-time big data processing scheduling techniques. The scientific significance of this review paper is that it will help the researchers understand the need to develop algorithms and techniques that can help in the prosperity of smart cities and similar systems, eventually leading to the betterment of humanity.

The division for the rest of the paper is as follows; section 2 explains the smart city and its few characteristics. Section 3 presents the processing of real-time data techniques. The Spark and its comparison with the Hadoop are discussed in section 4. Finally, the paper is concluded along with future recommendations in section 5.

2. SMART CITY

A smart city should be able to optimize the utilization of all of its assets, both the material (such as transportation systems, energy distribution networks, and natural resources) and immaterial (such as human capital, the intellectual capital of companies, and organizational capital in public administration bodies) in real-time Flood, fire, earthquake emergency rescue and disaster relief, anti-terrorism, remote control of hazardous areas, and so on are some of the many potential uses [14]. In contrast to renewable resources (such as solar, wind, and geothermal energy), nonrenewable resources (such as petroleum) will finish over time because of the concept of depletion. In recent decades, experts have promoted the ideas of smart energy [15], green energy [16], and sustainable energy [17] to raise awareness of challenges and develop the best energy usage practices. A smart city has several characteristics, including the transfer of technological, infrastructural, and managerial procedures from rural to urban settings.

2.1. Characteristics of smart cities

Specific characteristics, keynotes, and organizational frameworks characterize smart cities; the idea behind this theme is the foundation of a modern, technologically advanced metropolis. A few of the smart city services are given in Figure 1. The figure highlights various features of a smart city, including the education system, health system, daily utility management, smart transportation, government sector, and public sector. The explanations below elaborate on these features, showcasing how technology and data-driven solutions enhance urban life.

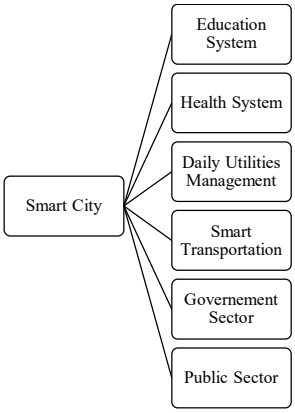

Figure 1. Smart city services

2.1.1. Smart energy (sustainable resources)

The concept of sustainability has maintained its prominence throughout the development of the smart city [18]–[20]. Preserving energy and natural resources is critical for a smart city to function sustainably [21]–[23]. In the early days of the smart city movement, enhancing residents' comfort was a primary focus. To address these issues, various cities throughout the world ran trials. Intelligent lighting has been the attention of specific studies. Citizens may adjust the brightness of the ten thousand sensor-equipped streetlights to suit their needs. The goal is to reduce power consumption by approximately 70% [24].

Smart energy is appealing more since it promotes an all-encompassing approach to coordinating environmentally friendly power, maintainable energy, and a sustainable power source. The goal of eco-friendly energy is to use fuel with minimal environmental impact and the least negative natural consequences. An alternative energy source that does not deplete the planet's resources over time is the best option for meeting the world's energy needs. Increased focus on energy needs has led to a rise in the popularity of renewable energy sources. Much research is going on to integrate renewable energy sources into intelligent buildings. Smart buildings may use renewable energy, or the existing infrastructure can incorporate renewable energy plants. There is a proposal for a microgrid control framework that integrates a photovoltaic (PV) power source with a significant energy storage unit [25]. Similarly, Jia *et al.* [26] propose combining solar and wind power to decrease the dependency on critical energy resources.

2.1.2. Smart transportation

Accessibility at regional and international levels and the availability of cutting-edge, environmentally friendly transportation technologies all fall under the term smart transportation [27], [28]. The need for reliable modes of transportation dates back to the dawn of civilization. As technology has progressed, all modes of transportation, including land, sea, rail, and air, must follow the same stipulation. Neither the world's traditional transportation strategy nor its components were linked or interlinked. A cutting-edge linked system has replaced the conventional transportation system due to the concept of everyday interfacing devices. Therefore, modern automobiles are part of various communication and route frameworks. All the automobiles that participate in a particular transmission are linked together. Several standalone transporters are connected to form a global transportation system by increasing the connections inside a single transporter. Intelligent transportation systems (ITS) have given much thought to the ad hoc vehicle network (VANET) [29]. VANET has widely used vehicle-to-vehicle (VV) and VV-to-infrastructure (VI) communication capabilities to manage rural traffic. Using the new transportation framework metrics to ensure the metropolitan area's viability comes at the expense of the residents' happiness [30].

2.1.3. Smart healthcare

The present healthcare system is struggling to keep up with the demands of a rapidly expanding population. Furthermore, the issue worsens because medical staff numbers have not increased with population growth. As a result, the healthcare expectations and the delivery gap widen due to a lack of resources and high demand. To meet the need and improve the quality of administration, current innovative well-being administrations use sensor organizations, ICT, distributed computing, computer fog, cell phone applications, and incredible information handling systems [31]. Integrating electronic clinical records (ECRs) further allows for timely decisions with the most up-to-date information [32]. Another method of achieving satisfactory portable well-being in metropolitan areas was given by [33].

2.1.4. Waste management

Rapid urbanization and increased manufacturing have contributed to a rise in waste production. Effective waste management is possible via the cooperation of the workforce, municipal authorities, and private businesses [34]. There are four main phases of waste management, and they are as follows: waste collection, waste removal, waste reuse, and waste recovery. Poor and unmanaged waste management generates challenges in human health and the environment [35], making trash management essential for the economic development of smart urban areas.

3. THE PROCESSING OF DATA IN REAL-TIME

The problem of processing extensive data becomes increasingly difficult as data volume and diversity both rise. For efficient analytics, it is necessary to have access to the information within this time frame. For instance, real-time data processing is essential in a traffic monitoring system that constantly tracks millions of cars. This processing helps in locating alternative routes and calculating arrival times. Timeliness is of the utmost significance in this context since a mistake or delay might result in the misrouting of an ambulance, putting lives at risk. With more and more people needing access to decision-making tools in real-time, timeliness has emerged as a crucial indicator of data quality. Therefore, having enough time to handle massive amounts of data in real-time is vital. As a bonus, the timely nature of big data might aid in analyzing event streams to enable real-time decision-making. Therefore, the diverse data sets provided by many data sources must be integrated into a unified analytical platform to minimize potential delays in real-time processing [36]–[39]. The flowchart of data processing in real-time is given in Figure 2. It begins with real-time data collection from various sources in the city. A framework is then selected to handle the collected data efficiently. Big data analysis is conducted to derive insights and identify patterns. Finally, optimized smart city services, such as smart transportation, energy management, waste management, and more, are implemented based on the analysis. This systematic approach leverages technology and data to improve urban living.

Real-time data processing is essential for maximizing the effectiveness of smart city services. However, effective scheduling becomes crucial to guarantee the prompt delivery of services and the effective completion of tasks. Tasks are prioritized and efficient timetables for various processes are created using scheduling algorithms and policies. In complex real-time operations, scheduling is especially important for ensuring punctuality and meeting requirements. A schedule that meets most requirements for a particular set of processes is considered optimal in this context.

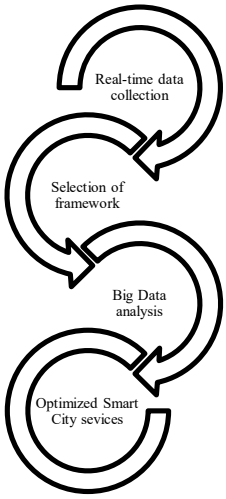

Figure 2. Real-time data processing flow chart

3.1. Scheduling

A scheduler will prioritize the tasks using an algorithm or policy. The job of a scheduler is to create a timetable for a group of processes. A process set is realistic if it can timetable itself to meet specific requirements. Complex real-time periodic operations often need a guarantee of punctuality. An optimum

schedule is a schedule that meets most of the specified requirements for a given set of processes. In most cases, a scheduler is optimum if it can schedule every possible collection of operations [40]. Static and dynamic [41] are two ways to categorize scheduling algorithms.

3.1.1. Static scheduler

Static scheduling, in which a schedule is generated offline. All scheduling decisions, such as when to execute each operation or send each message, are contained in the program. During runtime, a simple dispatcher distributes jobs based on the schedule. Static scheduling is sometimes known as time-triggered scheduling [42]. All scheduling choices are stored in a table for usage at runtime. It is only possible to do this with previous information on how the process works. Therefore, this plan can only function if all operations are genuinely periodic. Although it demands insight into a process's traits beforehand, the overhead it imposes during execution is negligible. Real-time shortest job first (SJF) and rate monotonic (RM) are appropriate algorithms for static process scheduling. In both algorithms, priority is allocated depending on the deadline and time required to finish the task [43].

3.1.2. Dynamic scheduler

On the other hand, a dynamic approach establishes schedules during execution, providing a more adaptable system capable of handling unanticipated occurrences. It is plausible to claim that in safety-critical systems, all events should be predictable, and stimulability should be the primary concern before any action; this means it needs a scheduling method that is entirely unchanging across time. Online schedulers make scheduling choices while the system is actively running. It can be both static and active. These choices are grounded in the process context's past and present state—the current systemic condition. The term clairvoyant refers to a planner or scheduler. Two commonly used dynamic schedulers in real-time systems are the least slack time first (LST) and the earliest deadline first (EDF). In these algorithms, priority is decided based on slack time and deadlines of the given processes. These both are considered more suitable for soft real-time operating systems [43]. The objectives for the few static and dynamic scheduler algorithms are discussed in Table 1.

Table 1. Static and dynamic scheduler algorithms

Category	Algorithm	Objectives achieved
Static	Highest level first with estimated	Minimized running time
	Time [44]	It simplified the list scheduling algorithm
	Critical path on a processor [44]	They limited the cost of computation and time consumed
	Constrained earliest finish time [45]	Reduction in implementation time
	Multipriority queueing genetic [46]	It decreased the execution time for subtasks
	Parallelism-based earliest finish time [47]	It reduced the finish time
Dynamic	Dynamic level scheduling [48]	It decreased scheduled time
	Dynamic task scheduling [49]	Less complicated, and less time is taken to finish the tasks
	Dynamic load balancing using genetic algorithms [50]	Optimized load balancing and processor consumption along with high speed
	New response time bounds for fixed priority [51]	Better response time
	Load-based schedulability [52]	Scheduling based on priority

3.1.3. Hybrid scheduler

Schedulers may be either preemptive or non-preemptive. In most cases, pre-emption happens when a process with a higher priority becomes executable. As a result of pre-emption, a procedure might go on hold without the participant's consent. It is not the practice of non-preemptive schedulers to temporarily suspend running tasks; however, it can manage concurrency for processes running inside a resource with mutually exclusive access [53].

It's also feasible to use a hybrid system. A scheduler can have a pre-emptive design while allowing processes to work in less time and then put them on hold; it may define an immutable block of code that another method cannot bypass. For instance, the program may poll the system clock for the current time, use that to determine how much of a delay is required, and then implement that delay. If the process could pause between reading the clock and performing the hold, it would be impossible to write such code. Using caution while implementing code that uses delayed pre-emption primitives is essential. The ensuing blocking must be limited and minor-often of the same order of magnitude as the overhead of context switching. The computer's scheduler uses this strategy to enable a rapid context switch; the switch operates up to 50 processor cycles to postpone itself; as a result, the context to be moved is short, and only ten additional cycles can accommodate the modified context [54].

3.2. Previous works

Numerous studies have been conducted on task scheduling, exploring various algorithms and models. One notable study by Liu and Layland in 1973 [55] focused on the earliest-deadline-first (EDF) scheduling algorithm and fixed priority (FP) scheduling. They investigated these algorithms using the ordinary periodic job model, without self-suspensions and demonstrated that EDF is an optimum approach to meeting commitments. Additionally, they established the superiority of the rate-monotonic (RM) scheduling algorithm among FP techniques.

Another study in [56], [57] centered around configuring and scheduling emergency resources during fire catastrophes. They developed a dynamic model to analyze and address this critical aspect. Similarly, constructed emergency resource scheduling models, considering factors such as arbitrary initial time for rescue operations and a fixed number of rescuers [57], [58]. In 2010 Sandholm and Lai [59] proposed a dynamic proportional share scheduler. This scheduler is an enhancement to Hadoop schedulers that gives the volume quality of service (QoS) to diverse users based on priority. This process allows the handler to choose tasks and schedule them according to their preference. Change in the allocated resources based on the work requirements is also doable. This scheduler becomes fair in case of no users and resource requirements.

In 2016, Zacheilas and Kalogeraki [60] introduced a cost-effective scheduling technique. This strategy aims to meet financial constraints while also improving task completion time. This method implements the Pareto approach. This scheduler aids in decreasing completion time and giving better throughput. One aspect that influences a cluster's overall performance is Job response time. This aspect inspires Zaharia *et al.* [61] to suggest a longest approximate time to end (LATE) scheduling algorithm to improve response time. This method processes the backup task of a slow task on a separate node. Various factors, including increased CPU usage and the sluggishness of background tasks, are the reason behind the task's slow progress.

Locality-aware reduced task scheduling (LARTS) [62]. This algorithm aims to enhance data localization, and as a result, there is minimum network traffic. This study also addressed premature shuffle concerns. Although early shuffle improves performance and reduces turnaround time, it also burdens the network. Therefore, LARTS requested that the shuffle begins once the specific addressing processing is done; the sweet spot is the name for the beginning point of the shuffle. In 2012, Guo *et al.* [63] proposed delay scheduling, which addresses the disadvantage of the fair scheduler by attempting to remove the difficulties of locating the tasks. When a request for a new task enters delay scheduling, it finds the job that meets the equality constraints and does not assign the job if conditions are not fulfilled.

Table 2 presents a comprehensive comparison of the discussed techniques with other approaches. The table provides a detailed evaluation of various factors, such as performance metrics, scalability, resource utilization, and adaptability. By comparing the discussed techniques with alternative methods, this analysis offers insights into the strengths and limitations of each approach, aiding researchers and practitioners in selecting the most suitable scheduling technique for their specific requirements.

Table 2. Scheduler techniques comparison table

	Resolved issues	Throughput	Response time	Execution time	Energy efficient
Dynamic proportional share scheduler [59]	Fairness	✗	✗	✓	N/A
Longest approximate time to end (LATE) [61]	Speculative execution	✗	✓	✗	N/A
Delay scheduling [63]	Data locality and fairness	✓	✓	✗	N/A
Cost-effective scheduling technique [60]	Data locality	✓	✗	✗	N/A
Locality-aware reduces task scheduling (LARTS) [62]	Data locality	✗	✓	✗	N/A
Parental prioritization-based task scheduling algorithm [64]	Fairness	N/A	N/A	✓	✗
Modified particle swarm optimization algorithm [63]	N/A	N/A	N/A	✓	✗
A hybrid of genetic and particle swarm optimization [65]	N/A	✗	N/A	✓	✓

3.3. Computable and decidable

The computational cost and complexity of scheduling for intricate systems are a genuine concern. Online scheduling methods should avoid using scheduling algorithms with exponential complexity because of their severe influence on the amount of time spent on application software. Furthermore, some scheduling considerations are computationally intractable, making them inappropriate for offline scheduling. Therefore, computability and decidability must be considered two aspects of computational complexity. The computability of a schedule determines if a given schedule is feasible. At the same time, decidability helps to assess whether a possible schedule exists [40].

4. DATA PROCESSING FRAMEWORKS

In big data analytics, efficient data processing frameworks serve as the backbone of handling and analyzing vast amounts of information, which includes the data generated by smart cities. These frameworks provide the necessary tools and infrastructure to extract valuable insights from diverse data sources, enabling cities to make data-driven decisions and optimize urban life. Hadoop and Spark are two prominent data processing frameworks that have revolutionized the field and found extensive applications in smart cities.

4.1. Spark framework

Spark is an open-source framework for processing large amounts of data quickly and easily. This approach debuted in 2009 at Berkeley and was officially adopted by Apache the following year. Iterative algorithms in machine learning, interactive data analysis tools, and graph algorithms are all examples of recursive systems that benefit from repetition [66]. As a result, the programmers developed the Spark framework [67] to accommodate these programs while providing scalability and fault tolerance in the MapReduce framework. Parallel operations on these datasets (referring to providing a function to utilize a dataset) and resilient distributed datasets [68] are Spark's two primary abstractions for parallel scheduling. Resilient distributed datasets were first made possible by Spark (RDDs). Distributed read-only datasets (RDDs) are groups of read-only items kept on many computers but can quickly reassemble in case of partition removal. It allows the user to store the RDD in the machines' memory and run the parallel process, such as MapReduce, many times. As a result, Spark excels in processing recursive algorithms on datasets [69], [70].

4.2. Hadoop vs Spark

Aziz *et al.* [71] analyzed Twitter data using the Spark platform in 2018. It took one second to explore all the tweets on Spark. This research has centered on the author's examination of the actual execution and completion of the standard Hadoop MapReduce framework, as well as the implementation of the Apache Spark framework. Experiment simulations are also run to determine actual-time data utilizing Spark and Hadoop. In addition, there is a discussion of Hadoop's constraints and benefits when it comes to its implementation in the real-time process. Finally, there is a simulation comparison regarding speed for both frameworks. All that shows that Spark is a powerful tool for processing real-time data streams.

In 2017 Hazarika *et al.* [72] evaluated the theoretical and practical differences between the Spark and Hadoop systems. From what they've seen in their studies, Spark's cache benefits from repeated queries like logistic regression and makes them significantly quicker. On the other hand, Spark's performance is weak for nonrepetitive queries because of the small cache size. Small iterations, however, benefit considerably from Hadoop's speed.

In 2015, Gopalani and Arora [73] examined two large data processing frameworks, Hadoop and Spark. To put it another way, they used Hadoop and Spark to apply the K-means algorithm, a fundamental machine learning technique, using a dataset comprised of sensor data and then comparing the two platforms' respective execution times. Data showed that Spark performed better than Hadoop in real-world scenarios. Furthermore, Gu and Li [74] conducted another comparison of memory needs and processing times for the Hadoop and Spark systems. The PageRank algorithm was implemented in several network datasets in the same study. According to the findings, Spark used more memory while simultaneously taking less time to execute, as impressive is the fact that Spark is 73% faster than Hadoop when dealing with massive datasets.

In 2013 Zaharia *et al.* [75] used logistic regression to examine the Hadoop and Spark frameworks. The author of this study focused on a subset of software programs that recycle data from an active, dynamical database using a multi-threaded, parallel architecture. These include many iterative machine-learning algorithms and interactive data analysis tools. Spark introduces an abstraction known as resilient distributed datasets (RDDs) to help achieve these objectives. Spark can beat Hadoop by a factor of ten in repeated machine learning tasks, and it can be used interactively on a 39 GB query dataset with a response time of less than one second. According to the findings of this article, Spark is the preferable option.

Liang *et al.* [76] compared Hadoop, Spark, and big dataMPI in terms of execution speed, memory footprint, and central processing unit consumption in 2014. The author uses Big Data Bench, a benchmark suite for large data sets, to conduct in-depth analyses of Spark, DataMPI, and Hadoop's resource use characterizations and performance. In these investigations, DataMPI delivered a 57% improvement over Spark. Furthermore, it has improved Hadoop by 50% regarding job implementation time. DataMPI's main advantages were its efficient communication mechanisms and its high throughput. In addition, DataMPI makes better use of its resources (disc, CPU, network I/O, and memory) than the other two structures and frameworks. As a result, the MPI platform outperformed both Spark and Hadoop, and Spark even surpassed Hadoop.

Mavridis and Karatza [77] assessed the performance of log file analysis using both Hadoop and Spark. They have looked at log file analysis using the cloud computing frameworks Apache Hadoop® and

Apache Sparks. The authors have enhanced the log file analysis in both frameworks so that they can handle real-world data from the Apache Web Server. They have also conducted other tests with varied parameters to evaluate and contrast the two frameworks and structures. Im and Moseley [78] used MapReduce to examine conditional lower bounds on graph connectedness. This research discovered the possible problems that don't allow efficient external algorithms to integrate into MapReduce. This study also answers a fundamental research question: how to tell whether a graph has a closed cycle. In particular, they examine the issue of designing an algorithm to determine whether or not two unconnected processes exist in a given network. This challenge aims to verify the graph's global structure so that all local graph parts are equivalent. They identify the natural class of algorithms that can only transfer/store/process data and information in paths, proving that no random algorithm can answer the question in a sublogarithmic number of rounds. Kodali *et al.* [79] work on a k-NN-based method using MapReduce for meta-path categorization in heterogeneous information networks. The authors of this study used the Passim similarity measure in a Heterogeneous Information Network to classify the meta-paths uncovered by applying the well-known MapReduce paradigm to the problem of locating k-nearest neighbors. Moreover, they figured out the classification technique to deal with the massive data found in HINs using MapReduce.

Wang *et al.* [80] conducted a study on MapReduce task programming with excessive energy consumption in heterogeneous clusters; as a result, there was a task programming framework for heterogeneous groups that considered resource utilization, deadlines, and data locality to keep energy costs to a minimum. The framework includes updates to the slot list, new task lists, and scheduling. In addition, a proposal for a novel job sequence to create a rational list of jobs and tasks based on factors like expected work processing times, available job slots, and due dates. Wei *et al.* [81] introduced a MapReduce-centric clustering method for handling large datasets. Their study compared and contrasted the MapReduce implementation of the Canopy method with the widely used K-means algorithm. By evaluating their performance and effectiveness in clustering large datasets, Wei *et al.* [81] shed light on the advantages and limitations of these approaches.

In a related study, Roger *et al.* [82] proposed a preemptive fair scheduler strategy for the disco MapReduce architecture. They explored how the Preemptive Fair Scheduler Policy impacted job execution times in both experimental production and research settings. While the strategy proved beneficial in reducing execution times for production jobs, it had a negative impact on research jobs. The author provided insights into the trade-offs and considerations of implementing the Preemptive Fair Scheduler Policy.

Jang *et al.* [83] proposed investigating k-nearest neighbor input initialization for neural network inversion. This study reveals a fresh way of initializing the input variables of neural networks, centered on the k-nearest neighbor technique (k-NN). The proposed method finds inputs that generate an outcome near a target output within a training dataset and combines them to form the starting input variables. Chen *et al.* [84] performed quick peak density clustering for large-scale data emphasizing kNN. The proposed methodology, computed using a fast kNN algorithm like a cover tree, significantly improves over the previous method of computing density using kNN-density. It uses kNN-density and a quick form to differentiate between local and nonlocal density peaks.

Janardhan and Samuel [85] investigated the optimal parallelism in the Spark architecture on Hadoop yet another resource negotiator (YARN) to get the most out of the cluster's resources. This research suggests the best parallelism conformation and configuration for an Apache Spark architecture deployed on a Hadoop YARN cluster. However, the concepts depend on the studies' findings that examine the reliance on parallelism at each level of Spark application performance. A zone-based resource allocation technique called Zebras enhances Spark's efficiency in a heterogeneous cluster and has also been proposed; by proposing and implementing this technique, optimizing resource utilization and allocation within the Spark cluster ultimately improves its overall performance.

According to Hussain and Surendran [86], efficient content-based fast-response picture retrieval is explored using the MapReduce and Spark model framework. The authors leverage the MapReduce model structure to sign efficiently and index massive volumes of photos, enabling fast retrieval based on content. Furthermore, in 2021, Mostafaeipour *et al.* [87] adopted Spark as a proportional method for recovering the index, operating on the upper layer of the MapReduce framework and utilizing the Hadoop distributed file system (HDFS). Their work focuses on efficient index recovery using Spark's capabilities within the MapReduce ecosystem.

In addition to the insights provided, Table 3 further reinforces the key differences between Spark and Hadoop. The table highlights specific research gaps and indicates whether each gap is present in Spark or Hadoop. This comprehensive comparison aids in understanding the unique strengths and limitations of each framework, enabling researchers and practitioners to make informed decisions regarding their data processing needs in the context of smart cities.

Table 3. Data processing aspects comparison table

Aspect	Spark	Hadoop
Performance optimization	✓	✗
Cache and query optimization	✓	✗
Memory usage and processing time	✓	✗
Machine learning performance	✓	✗
Resource utilization	✓	✗
Log file analysis performance	✓	✗
Mapreduce efficiency	✗	✓
Energy efficiency	✓	✗

5. CONCLUSION

As the world merges toward the era of smart cities highly dependent on IoT and Web Apps. Smart cities are gaining popularity as they positively impact a country's economy. Intelligent and rapid decision-making are critical requisites of a sophisticated smart city system. At the same time, this system generates multiple files known as big data that revolve around the characteristics of the 3 V's, which has led to the recognition of a great problem. New ideologies, strategies, and frameworks must be introduced to constructively overcome the issue of handling and scheduling big data. This article provides an overview of a thorough study of work done for scheduling techniques in the Hadoop and Spark environments. Dynamic Scheduling is crucial to achieving high performance in extensive data processing. Data volume, diversity, data velocity, security and privacy, cost, connectivity, and data sharing are just a few of the difficulties with big data. From the conducted review, it can be easily said that the baseline is adequate for processing if the data is static, and it is possible to wait until batch processing is finished. However, Spark has had an advantage regarding real-time data processing in parallelism. It still needs extensive research to conclude that Spark is the only solution for analyzing real-time streaming data.

Additionally, as demonstrated in the study, Spark could evaluate data quickly. Spark is a top-notch memory processing technology that enables real-time streaming data processing on massive amounts of data. Compared to Hadoop, Apache Spark is far more sophisticated. It supports several needs, including batch, streaming, and real-time processing. In the future, schedule optimization can be done for Hadoop. For Spark, it can be done by modifying various default parameter configuration settings, introducing new scheduling techniques, and hybrid artificial intelligence scheduling.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the School of Electrical and Electronic Engineering, Universiti Sains Malaysia (USM).

REFERENCES

[1] J. M. Shapiro, "Smart Cities: Quality of Life, Productivity, and the Growth Effects of Human Capital," *Review of Economics and Statistics*, vol. 88, no. 2, pp. 324–335, May 2006, doi: 10.1162/rest.88.2.324.

[2] A. Luberg, T. Tammet, and P. Järv, "Smart City: A Rule-based Tourist Recommendation System," in *Information and Communication Technologies in Tourism 2011*, Vienna: Springer Vienna, 2011, pp. 51–62, doi: 10.1007/978-3-7091-0503-0_5.

[3] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart cities in Europe," *Journal of Urban Technology*, vol. 18, no. 2, pp. 65–82, 2013, doi: 10.4324/9780203076224.

[4] X. Gong, F. Dong, M. A. Mohamed, O. M. Abdalla, and Z. M. Ali, "A Secured Energy Management Architecture for Smart Hybrid Microgrids Considering PEM-Fuel Cell and Electric Vehicles," *IEEE Access*, vol. 8, pp. 47807–47823, 2020, doi: 10.1109/ACCESS.2020.2978789.

[5] J. Chin, V. Callaghan, and I. Lam, "Understanding and personalising smart city services using machine learning, The Internet-of-Things and Big Data," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, IEEE, Jun. 2017, pp. 2050–2055, doi: 10.1109/ISIE.2017.8001570.

[6] M. M. Rathore, A. Ahmad, and A. Paul, "IoT-based smart city development using big data analytical approach," in *2016 IEEE International Conference on Automatica (ICA-ACCA)*, IEEE, Oct. 2016, pp. 1–8, doi: 10.1109/ICA-ACCA.2016.7778510.

[7] A. Z. Abualkishik, "Hadoop and big data challenges," *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 12, pp. 3488–3500, 2019.

[8] Q. Duan *et al.*, "Optimal Scheduling and Management of a Smart City Within the Safe Framework," *IEEE Access*, vol. 8, pp. 161847–161861, 2020, doi: 10.1109/ACCESS.2020.3021196.

[9] M. Chen, S. Mao, and Y. Liu, "Big Data: A Survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, Apr. 2014, doi: 10.1007/s11036-013-0489-0.

[10] J. Dean and S. Ghemawat, "MapReduce," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.

[11] A. Ali, N. M. Mirza, and M. K. Ishak, "A New Merging Numerous Small Files Approach for Hadoop Distributed File System," in *2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, IEEE, May 2022, pp. 1–4, doi: 10.1109/ECTI-CON54298.2022.9795369.

[12] N. Komninos, H. Schaffers, and M. Pallot, "Developing a Policy Roadmap for Smart Cities and the Future Internet," *eChallenges e2011*, pp. 1–8, 2011.
[13] A. Jones and A. Kovacich, *Emergency Management*, 2nd Edition. New York: Routledge, 2012, doi: 10.1201/b11887.
[14] T. Nam and T. A. Pardo, "Smart city as urban innovation," in *Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance*, New York, NY, USA: ACM, Sep. 2011, pp. 185–194, doi: 10.1145/2072069.2072100.
[15] H. Lund, B. Vad Mathiesen, D. Connolly, and P. A. Østergaarda, *Renewable energy systems - A smart energy systems approach to the choice and modelling of 100 % renewable solutions*, vol. 39, no. Special Issue. 2014, doi: 10.3303/CET1439001.
[16] O. B. Mora-Sanchez, E. Lopez-Neri, E. J. Cedillo-Elias, E. Aceves-Martinez, and V. M. Larios, "Validation of IoT Infrastructure for the Construction of Smart Cities Solutions on Living Lab Platform," *IEEE Transactions on Engineering Management*, vol. 68, no. 3, pp. 899–908, Jun. 2021, doi: 10.1109/TEM.2020.3002250.
[17] S. Chu and A. Majumdar, "Opportunities and challenges for a sustainable energy future," *Nature*, vol. 488, no. 7411, pp. 294–303, Aug. 2012, doi: 10.1038/nature11475.
[18] G. C. Lazaroiu and M. Roscia, "Definition methodology for the smart cities model," *Energy*, vol. 47, no. 1, pp. 326–332, Nov. 2012, doi: 10.1016/j.energy.2012.09.028.
[19] L. Guan, "Smart steps to a battery city," *Government News*, vol. 32, no. 2, pp. 24–27, 2012.
[20] J. Yang, Y. Kwon, and D. Kim, "Regional Smart City Development Focus: The South Korean National Strategic Smart City Program," *IEEE Access*, vol. 9, pp. 7193–7210, 2021, doi: 10.1109/ACCESS.2020.3047139.
[21] S. P. Mohanty, U. Choppali, and E. Kougianos, "Everything you wanted to know about smart cities: The Internet of things is the backbone," *IEEE Consumer Electronics Magazine*, vol. 5, no. 3, pp. 60–70, Jul. 2016, doi: 10.1109/MCE.2016.2556879.
[22] F. Herraiz Faixó, F. J. Arroyo-Cañada, M. P. López-Jurado, and A. M. Lauroba Pérez, "Digital assets Horizon in smart cities: Urban congestion management by IoT, Blockchain/DLT and human reinforcement," *Advances in Intelligent Systems and Computing*, vol. 894, pp. 63–82, 2020, doi: 10.1007/978-3-030-15413-4_6.
[23] R. Kitchin, "The real-time city? Big data and smart urbanism," *GeoJournal*, vol. 79, no. 1, pp. 1–14, Feb. 2014, doi: 10.1007/s10708-013-9516-8.
[24] L. Sanchez *et al.*, "SmartSantander: Experimentation and service provision in the smart city," *International Symposium on Wireless Personal Multimedia Communications, WPMC*, pp. 1–6, 2013.
[25] D. Puiu *et al.*, "CityPulse: Large Scale Data Analytics Framework for Smart Cities," *IEEE Access*, vol. 4, pp. 1086–1108, 2016, doi: 10.1109/ACCESS.2016.2541999.
[26] G. Jia, G. Han, J. Jiang, N. Sun, and K. Wang, "Dynamic Resource Partitioning for Heterogeneous Multi-Core-Based Cloud Computing in Smart Cities," *IEEE Access*, vol. 4, pp. 108–118, 2016, doi: 10.1109/ACCESS.2015.2507576.
[27] J. Ha, M. Kambe, and J. Pe, *Data Mining: Concepts and Techniques*, 3rd editio. Amsterdam: Morgan Kaufmann is an imprint of Elsevier, 2011, doi: 10.1016/C2009-0-61819-5.
[28] J. Han, C. Choi, W. Park, I. Lee, and S. Kim, "Smart home energy management system including renewable energy based on ZigBee and PLC," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 2, pp. 198–202, May 2014, doi: 10.1109/TCE.2014.6851994.
[29] V. Fernandez-Anez, J. M. Fernández-Güell, and R. Giffinger, "Smart City implementation and discourses: An integrated conceptual model. The case of Vienna," *Cities*, vol. 78, pp. 4–16, Aug. 2018, doi: 10.1016/j.cities.2017.12.004.
[30] I. Ahmed, H. Karvonen, T. Kumpuniemi, and M. Katz, "Wireless Communications for the Hospital of the Future: Requirements, Challenges and Solutions," *International Journal of Wireless Information Networks*, vol. 27, no. 1, pp. 4–17, Mar. 2020, doi: 10.1007/s10776-019-00468-1.
[31] A. Stratigea, C.-A. Papadopoulou, and M. Panagiotopoulou, "Tools and Technologies for Planning the Development of Smart Cities," *Journal of Urban Technology*, vol. 22, no. 2, pp. 43–62, Apr. 2015, doi: 10.1080/10630732.2015.1018725.
[32] A. Ojo, E. Curry, T. Janowski, and Z. Dzhusupova, "Designing Next Generation Smart City Initiatives: The SCID Framework," 2015, pp. 43–67, doi: 10.1007/978-3-319-03167-5_4.
[33] V. Stepaniuk, J. Pillai, B. Bak-Jensen, and S. Padmanaban, "Estimation of Energy Activity and Flexibility Range in Smart Active Residential Building," *Smart Cities*, vol. 2, no. 4, pp. 471–495, Nov. 2019, doi: 10.3390/smartcities2040029.
[34] J. Engelbert, L. van Zoonen, and F. Hirzalla, "Excluding citizens from the European smart city: The discourse practices of pursuing and granting smartness," *Technological Forecasting and Social Change*, vol. 142, pp. 347–353, May 2019, doi: 10.1016/j.techfore.2018.08.020.
[35] D. Wang, B. Bai, K. Lei, W. Zhao, Y. Yang, and Z. Han, "Enhancing Information Security via Physical Layer Approaches in Heterogeneous IoT With Multiple Access Mobile Edge Computing in Smart City," *IEEE Access*, vol. 7, pp. 54508–54521, 2019, doi: 10.1109/ACCESS.2019.2913438.
[36] C. L. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347, Aug. 2014, doi: 10.1016/j.ins.2014.01.015.
[37] Z. Zheng, P. Wang, J. Liu, and S. Sun, "Real-time big data processing framework: Challenges and solutions," *Applied Mathematics and Information Sciences*, vol. 9, no. 6, pp. 3169–3190, 2015, doi: 10.12785/amis/090646.
[38] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and Good practices," in *2013 Sixth International Conference on Contemporary Computing (IC3)*, IEEE, Aug. 2013, pp. 404–409, doi: 10.1109/IC3.2013.6612229.
[39] N. Khan *et al.*, "Big Data: Survey, Technologies, Opportunities, and Challenges," *The Scientific World Journal*, vol. 2014, pp. 1–18, 2014, doi: 10.1155/2014/712826.
[40] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Real-Time System Scheduling," in *Predictably Dependable Computing Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 41–52, doi: 10.1007/978-3-642-79789-7_3.
[41] S.-C. Cheng and J. A. Stankovic, "Scheduling Algorithms for Hard Real-Time Systems--A Brief Survey," *IEEE Transaction on Software Engineering*, pp. 150–173, 1988.
[42] J. Mäki-Turja, K. Hänninen, and M. Nolin, "Efficient development of real-time systems using hybrid scheduling," *Proceedings of the 2005 International Conference on Embedded Systems and Applications, ESA'05*, pp. 53–59, 2005.
[43] J. Teraiya and A. Shah, "Analysis of Dynamic and Static Scheduling Algorithms in Soft Real-Time System with Its Implementation," in *Soft Computing: Theories and Applications*, Singapore: SpringerLink, 2020, pp. 757–768, doi: 10.1007/978-981-15-0751-9_69.
[44] H. Topcuoglu, S. Hariri, and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, Mar. 2002, doi: 10.1109/71.993206.
[45] M. A. Khan, "Scheduling for heterogeneous Systems using constrained critical paths," *Parallel Computing*, vol. 38, no. 4–5, pp. 175–193, Apr. 2012, doi: 10.1016/j.parco.2012.01.001.

[46] Y. Xu, K. Li, T. T. Khac, and M. Qiu, "A Multiple Priority Queueing Genetic Algorithm for Task Scheduling on Heterogeneous Computing Systems," in *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, IEEE, Jun. 2012, pp. 639–646, doi: 10.1109/HPCC.2012.91.
[47] X. Wu, S. Cheng, S. Yuan, and Z. Wang, "A Parallelism-Based Earliest Finish Time (PBEFT) Algorithm for Workflow Scheduling in Clouds," in *2022 7th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, IEEE, Apr. 2022, pp. 22–26, doi: 10.1109/ICCCBDA55098.2022.9778917.
[48] G. C. Sih and E. A. Lee, "Dynamic-level scheduling for heterogeneous processor networks," in *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing 1990*, IEEE Comput. Soc. Press, pp. 42–49, doi: 10.1109/SPDP.1990.143505.
[49] D. J. Lilja, L. Y. Kit, and B. Hamidzadeh, "Dynamic task scheduling using online optimization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 11, pp. 1151–1163, 2000, doi: 10.1109/71.888636.
[50] Seong-hoon Lee and Chong-sun Hwang, "A dynamic load balancing approach using genetic algorithm in distributed systems," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, IEEE, 1998, pp. 639–644, doi: 10.1109/ICEC.1998.700103.
[51] N. Guan, M. Stigge, W. Yi, and G. Yu, "New Response Time Bounds for Fixed Priority Multiprocessor Scheduling," in *2009 30th IEEE Real-Time Systems Symposium*, IEEE, Dec. 2009, pp. 387–397, doi: 10.1109/RTSS.2009.11.
[52] H. Li and S. Baruah, "Load-based schedulability analysis of certifiable mixed-criticality systems," *Embedded Systems Week 2010 - Proceedings of the 10th ACM International Conference on Compilers, Architecture and Synthesis for Embedded Systems, EMSOFT'10*, pp. 99–107, 2010, doi: 10.1145/1879021.1879035.
[53] H. Tokuda, *'Real-Time Critical Section: Not Preempt, Preempt or Restart?* 1989.
[54] A. Burns and A. Wellings, *Real–Time Systems and their programming languages*, 4 edition. United States: Addison-Wesley, 1989.
[55] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973, doi: 10.1145/321738.321743.
[56] Z. Guangliang and S. Lian, "Research on Scheduling Models of Emergency Resource," in *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, IEEE, Mar. 2011, pp. 1110–1113, doi: 10.1109/ICICTA.2011.279.
[57] S. Shan, L. Wang, and L. Li, "Modeling of emergency response decision-making process using stochastic Petri net: an e-service perspective," *Information Technology and Management*, vol. 13, no. 4, pp. 363–376, Dec. 2012, doi: 10.1007/s10799-012-0128-7.
[58] A. M. Caunhye, X. Nie, and S. Pokharel, "Optimization models in emergency logistics: A literature review," *Socio-Economic Planning Sciences*, vol. 46, no. 1, pp. 4–13, Mar. 2012, doi: 10.1016/j.seps.2011.04.004.
[59] T. Sandholm and K. Lai, "Dynamic Proportional Share Scheduling in Hadoop," in *Part of the Lecture Notes in Computer Science*, Heidelberg: Springer, 2010, pp. 110–131, doi: 10.1007/978-3-642-16505-4_7.
[60] N. Zacheilas and V. Kalogeraki, "ChEsS: Cost-Effective Scheduling Across Multiple Heterogeneous Mapreduce Clusters," in *2016 IEEE International Conference on Autonomic Computing (ICAC)*, IEEE, Jul. 2016, pp. 65–74, doi: 10.1109/ICAC.2016.58.
[61] M. Zaharia, A. Konwinsk, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments," *8th USENIX Symposium on Operating Systems Design and Implementation*, pp. 29–42, 2008.
[62] M. Hammoud and M. F. Sakr, "Locality-Aware Reduce Task Scheduling for MapReduce," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, IEEE, Nov. 2011, pp. 570–576, doi: 10.1109/CloudCom.2011.87.
[63] Z. Guo and G. Fox, "Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, IEEE, May 2012, pp. 714–716, doi: 10.1109/CCGrid.2012.12.
[64] M. S. Arif, Z. Iqbal, R. Tariq, F. Aadil, and M. Awais, "Parental Prioritization-Based Task Scheduling in Heterogeneous Systems," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3943–3952, Apr. 2019, doi: 10.1007/s13369-018-03698-2.
[65] V. D. Reddy, G. R. Gangadharan, G. S. V. R. K. Rao, and M. Aiello, "Energy-Efficient Resource Allocation in Data Centers Using a Hybrid Evolutionary Algorithm," in *Machine Learning for Intelligent Decision Science*, Singapore: Springer, 2020, pp. 71–92, doi: 10.1007/978-981-15-3689-2_4.
[66] M. Kang and J.-G. Lee, "Effect of garbage collection in iterative algorithms on Spark: an experimental analysis," *The Journal of Supercomputing*, vol. 76, no. 9, pp. 7204–7218, Sep. 2020, doi: 10.1007/s11227-020-03150-z.
[67] Z. Tang, K. Liu, J. Xiao, L. Yang, and Z. Xiao, "A parallel k -means clustering algorithm based on redundance elimination and extreme points optimization employing MapReduce," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 20, pp. 1–18, Oct. 2017, doi: 10.1002/cpe.4109.
[68] W. Xiao and J. Hu, "SWEclat: a frequent itemset mining algorithm over streaming data using Spark Streaming," *The Journal of Supercomputing*, vol. 76, no. 10, pp. 7619–7634, Oct. 2020, doi: 10.1007/s11227-020-03190-5.
[69] M. Massie, B. Li, B. Nicholes, and V. Vuksan, *Monitoring with Ganglia*, 1 edition. USA: O'Reilly Media, 2012.
[70] I. S. Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2009, pp. 96–108.
[71] K. Aziz, D. Zaidouni, and M. Bellafkih, "Real-time data analysis using Spark and Hadoop," in *2018 4th International Conference on Optimization and Applications (ICOA)*, IEEE, Apr. 2018, pp. 1–6, doi: 10.1109/ICOA.2018.8370593.
[72] A. V. Hazarika, G. J. S. R. Ram, and E. Jain, "Performance comparision of Hadoop and spark engine," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, IEEE, Feb. 2017, pp. 671–674, doi: 10.1109/I-SMAC.2017.8058263.
[73] S. Gopalani and R. Arora, "Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means," *International Journal of Computer Applications*, vol. 113, no. 1, pp. 8–11, 2015, doi: 10.5120/19788-0531.
[74] L. Gu and H. Li, "Memory or Time: Performance Evaluation for Iterative Operation on Hadoop and Spark," in *2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing*, IEEE, Nov. 2013, pp. 721–727, doi: 10.1109/HPCC.and.EUC.2013.106.
[75] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark : Cluster Computing with Working Sets," in *HotCloud'10: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2013, pp. 1–7.
[76] F. Liang, C. Feng, X. Lu, and Z. Xu, "Performance Benefits of DataMPI: A Case Study with BigDataBench," in *BPOE 2014: Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, New York: Springer, Cham, 2014, pp. 111–123, doi: 10.1007/978-3-319-13021-7_9.

[77] I. Mavridis and H. Karatza, "Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark," *Journal of Systems and Software*, vol. 125, pp. 133–151, Mar. 2017, doi: 10.1016/j.jss.2016.11.037.
[78] S. Im and B. Moseley, "A Conditional Lower Bound on Graph Connectivity in MapReduce," in *arxiv Computer Science*, 2019, doi: 10.48550/arXiv.1904.08954.
[79] S. Kodali, M. Dabbiru, B. T. Rao, and U. Kartheek Chandra Patnaik, "A k-NN-Based Approach Using MapReduce for Meta-path Classification in Heterogeneous Information Networks," in *Soft Computing in Data Analytics*, Singapore: Springer, 2019, pp. 277–284, doi: 10.1007/978-981-13-0514-6_28.
[80] J. Wang, X. Li, R. Ruiz, J. Yang, and D. Chu, "Energy Utilization Task Scheduling for MapReduce in Heterogeneous Clusters," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 931–944, Mar. 2022, doi: 10.1109/TSC.2020.2966697.
[81] P. Wei, F. He, L. Li, C. Shang, and J. Li, "Research on large data set clustering method based on MapReduce," *Neural Computing and Applications*, vol. 32, no. 1, pp. 93–99, Jan. 2020, doi: 10.1007/s00521-018-3780-y.
[82] D. Garcia-Roger *et al.*, "5G Functional Architecture and Signaling Enhancements to Support Path Management for eV2X," *IEEE Access*, vol. 7, pp. 20484–20498, 2019, doi: 10.1109/ACCESS.2019.2897843.
[83] S. Jang, Y.-E. Jang, Y.-J. Kim, and H. Yu, "Input initialization for inversion of neural networks using k-nearest neighbor approach," *Information Sciences*, vol. 519, pp. 229–242, May 2020, doi: 10.1016/j.ins.2020.01.041.
[84] Y. Chen *et al.*, "Fast density peak clustering for large scale data based on kNN," *Knowledge-Based Systems*, vol. 187, Jan. 2020, doi: 10.1016/j.knosys.2019.06.032.
[85] P. S. Janardhanan and P. Samuel, "Optimum Parallelism in Spark Framework on Hadoop YARN for Maximum Cluster Resource Utilization," in *First International Conference on Sustainable Technologies for Computational Intelligence*, Singapore: Springer, 2020, pp. 351–363, doi: 10.1007/978-981-15-0029-9_28.
[86] D. M. Hussain and D. Surendran, "RETRACTED ARTICLE: The efficient fast-response content-based image retrieval using spark and MapReduce model framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 4049–4056, Mar. 2021, doi: 10.1007/s12652-020-01775-9.
[87] A. Mostafaeipour, A. J. Rafsanjani, M. Ahmadi, and J. A. Dhanraj, "Investigating the performance of Hadoop and Spark platforms on machine learning algorithms," *The Journal of Supercomputing*, vol. 77, no. 2, pp. 1273–1300, Feb. 2021, doi: 10.1007/s11227-020-03328-5.

BIOGRAPHIES OF AUTHORS

Nada Masood Mirza currently serves as an Instructor in the College of Engineering at United Arab Emirates University (UAEU), United Arab Emirates. She received her BE and MS degrees in Mechatronics Engineering from the College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Pakistan. Her post-graduate research was mostly focused on applying artificial intelligence, robotics, and wireless monitoring of renewable energy systems. She worked for a few years in academia in Pakistan, where her research focused mostly on autonomous wireless intelligent robotic systems. Since 2014 she has been involved in academic activities related to control and electronics engineering in United Arab Emirates. She is currently a Ph.D. student at Universiti Sains Malaysia (USM). Her research focuses on robotics, artificial intelligence, IoT, and control systems. She can be contacted at email: nada.mirza@student.usm.my.

Adnan Ali is a highly skilled system developer and web developer based in the United Arab Emirates. Holding a B. Eng. degree in Software Engineering from Al Ain University (AAU), he currently works as a web developer, leveraging expertise in programming languages such as PHP, MySQL, JavaScript, CSS, and HTML. Simultaneously, he is also pursuing an MS Engineering degree at Universiti Sains Malaysia (USM). With a keen interest in cutting-edge technologies, his research focuses on big data, IoT, embedded systems, simulation, and virtual reality. Notably, he has contributed to the academic community by publishing multiple papers, showcasing his dedication to expanding knowledge and innovation in his field. He can be contacted at email: adnan14711@gmail.com.

Mohamad Khairi Ishak received a B. Eng degree in Electrical and Electronics Engineering from the International Islamic University Malaysia (IIUM), Malaysia, an MSc. in Embedded Systems from the University of Essex, United Kingdom, and a Ph.D. from the University of Bristol, United Kingdom. He is a member of IEEE and a registered graduate engineer with the Board of Engineers Malaysia (BEM). He is an Associate Professor, Lecturer in Mechatronics Engineering at the School of Electrical and Electronic Engineering, Universiti Sains Malaysia (USM) and Ajman University, United Arab Emirates. His research interests are embedded systems, real-time control communications, and the internet of things (IoT). Emphasis is given to developing theoretical and practical methods that can be practically validated. Recently, significant research has been directed toward important industrial issues of embedded networked control systems and IoT. He can be contacted at email: khairiishak@usm.my and m.ishak@ajman.ac.ae.