Kafka-machine learning based storage benchmark kit for estimation of large file storage performance

Sanjay Kumar Naazre Vittal Rao[1], Anitha Chikkanayakanahalli Lokesh Kumar[1], Subhash Kamble[2]
[1]Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur, India
[2]Department of Information Science and Engineering, Global Academy of Technology, Bengaluru, India

Article Info

Article history:

Received Jun 13, 2024
Revised Nov 25, 2024
Accepted Dec 2, 2024

Keywords:

Benchmarking performance
Big data
Drill down feature
File storage
Payloads and storage benchmark kit

ABSTRACT

Efficient storage and maintenance of big data is important with respect to assuring accessibility and cost-friendliness to improve risk management and achieve an effective comprehension of the user requirements. Managing the extensive data volumes and optimizing storage performance poses a significant challenge. To address this challenge, this research proposes the Kafka-machine learning (ML) based storage benchmark kit (SBK) designed to evaluate the performance of the file storage system. The proposed method employs Kafka-ML and a drill-down feature to optimize storage performance and enhance throughput. Kafka-ML-based SBK has the capability to optimize storage efficiency and system performance through space requirements and enhance data handling. The drill-down search feature precisely contributes through reducing disk space usage, enabling faster data retrieval and more efficient real-time processing within the Kafka-ML framework. The SBK aims to provide transparency and ease of utilization for benchmarking purposes. The proposed method attains maximum throughput and minimum latency of 20 MBs and 70 ms, respectively on the number of data bytes is 10, as opposed to the existing method SBK Kafka.

Corresponding Author:

Sanjay Kumar Naazre Vittal Rao
Department of Computer Science and Engineering, Kalpataru Institute of Technology
Tiptur, Tumkur District, Karnataka - 572201, India
Email: sanjaynv@gmail.com

1. INTRODUCTION

The storing of data and its access is a challenge with the large amount of data in file storage systems. The evaluation of benchmarking plays a significant role in performance optimization as it supports in evaluating the storage performance in various systems [1]. The performance benchmarking allows organizations to assess the system changes during evaluation, as well as authorize for leveraging the data for further performance optimization [2]. To assess the latency, throughput and speed of the system, the benchmarking establishes a baseline for evaluating the effects of the system's changes. The storage benchmark kit (SBK) is a tool or open-source software framework utilized to estimate the read and write operations of the storage benchmarks [3]. SBK aids performance benchmarking by different implementation methods like throughput, rate limiter, and latency [4], [5]. SBK distributes the benchmarking outcomes into read or write latency and read/write throughput to the Grafana analytics for developing the performance graphs. This benchmarking allows users to estimate the maximum achievable throughput performance of their storage devices, alongside providing precise insights into the storage system's performance [6], [7]. The SBK offers widespread support for a diverse array of storage systems, encompassing local and distributed file systems, object storage systems, and key-vale storage systems [8]. This framework contributes with an efficient performance benchmarking solution through reading and writing data in the storage system [9]. The SBK accommodates a number of payload types including byte buffer, byte array and string to permit users to extend their individual payload types [10], [11]. Furthermore, the SBK helps database schemes of MySQL, SQLite, PostgreSQL, Apache Derby, and Microsoft structured query language (SQL) by Java database connectivity (JDBC) [12], [13]. The SBK executes periodic logging of the benchmarking outcomes to Grafana analytics by Prometheus examination approach [14]. During the benchmarking performance, the data is stored in a local disk of Ext4 file system, RocksDB and a LevelDB key value store. In MinIO distributed storage system, the objects are sent or receive through remote hosted MinIO server [15], [16].

The fast development of data volumes poses significant challenges in managing and optimizing the storage performance. As the data becomes enhancing uneven and distributed over different storage levels, effective location and retrieval of data is complex. Ensuring scalability, minimizing latency, and maintaining data integrity under dynamic workloads further obscures the storage management. These challenges require a complete SBK to effectively estimate and solve these complexities. To address this challenge, this research proposes the Kafka-machine learning (ML) based SBK for estimating the performance of the file storage system. In this section, some of the existing works related to storage benchmarking performance are discussed. Furthermore, this section represents the advantages and limitations of each work based on its operation functions. Munegowda and Kumar [17] introduced the SBK framework for the estimation of performance of hardware devices. This framework described the most appropriate data structures like various concurrent queues to evaluate the throughput and low latency for storage devices. The SBK framework exported the standard storage interface application programming interfaces (APIs) which then appended the storage driver to evaluate the benchmarking performance for conventional storage device. While the utilization of hardware, the benchmarking supported decision making, but the benchmarks were often personalized to particular hardware configurations, and so, the outcomes varied when utilized with various hardware setups. Gómez-Luna *et al.* [18] developed a comprehensive analysis of an open-source real-word processing-in-memory (PIM) architecture. For this comprehensive analysis, two significant aspects were considered: Initially, the experimental characterization of unified processing in memory (UPMEM) based PIM system was conducted by the utilization of microbenchmarks to perform different architecture constraints. Then, processing-in-memory benchmarks (PrIM) was presented for the estimation of 16 workloads from various application domains. The PIM minimized the latency integrated with fetched data from traditional storage devices. However, the developed PIM approach had limited memory capacity as compared to the traditional methods.

Munegowda and Kumar [19] implemented the sliding latency coverage (SLC) factors to comprehend the range and the effectiveness of percentile variation latencies in storage performance benchmarking. The SLC depicted the range of latency, median, quartiles and percentiles in an individual unit factor. The experiments were performed on Ext4 file system, LevelDB, RocksDB and MinIO storage systems. The SLC approaches facilitated a parallel access to the data and permitted various parts of a system to access the data. Nonetheless, the implemented SLC approach created overhead, leading to a poor performance. Gotz *et al.* [20] introduced deep characterization approach of the microcontrollers for the selection of appropriate device in the central pillar of smart energy policy. The introduced approach investigated the potential of different low-power microcontrollers with the benchmark with the utilization of periodic duty cycle model of the typical wireless sensor networks (WSN). But the prolonged read operations deteriorated the system’s performance when the connector was located arbitrarily from the cloud storage. Ragavan and Rubavathi [21] developed big data storage minimization of binary file system approach for category-based drill down search engine which offered the rapid multi-level filtering competence. The developed approach stored the search engine data with 5 million data in a file system. Furthermore, the binary files were introduced in crawling procedure for the drill down search, while binary file loading into significant memory took a minimum time when compared to the added file format. Still, the approach was resistant when being dealt with new data types due to the lack of effective fitness of the existing categories. From this literature survey, the few limitations that are identified that can be noted are: benchmarking outcomes were varied when utilized with various hardware setups, limited memory capacity, creation of overhead, extended read operations that caused the system performance, and a resisted approach when dealt with new data types due to the lack of effective fitness of the existing categories. To overcome these limitations, this research proposes the Kafka-ML based SBK for the effective estimation of the storage performance for a large number of data files. This ensures the storage systems meet the constraints of advanced data environments, providing significant performance and reliability. The significant contributions of this research are as follows: i) this research proposes the Kafka-ML based SBK for streaming data and estimating the storage performance for large data files. Kafka-ML has the capability to distribute workloads efficiently, further leading to maximized throughput and minimized latency. Additionally, SBK allows users to measure and analyze the throughput of storage systems under different workloads; and ii) the drill down search is developed by using the binary file system to minimize storage requirements in the data. The drill down feature minimizes the disc space that facilitates in enhancing the search performance in the stored data, as opposed to the conventional file systems.

The rest of the research paper is organized as follows: Section 2 provides the proposed methodology. Section 3 presents the data stream management and file storage performance using Kafka-ML. Section 4 shows the results and discussion. Finally, Section 5 demonstrates the conclusion.

2. PROPOSED METHOD

This research proposes the Kafka-ML based SBK for evaluating the performance of the large data file storage in hardware devices. The Kafka-ML is used for streaming data in the storage data, while SBK is used to benchmark the performance of the large file storage data. The drill down feature supports in minimizing the requirements of the storage system [21]. Figure 1 depicts the design of SBK.

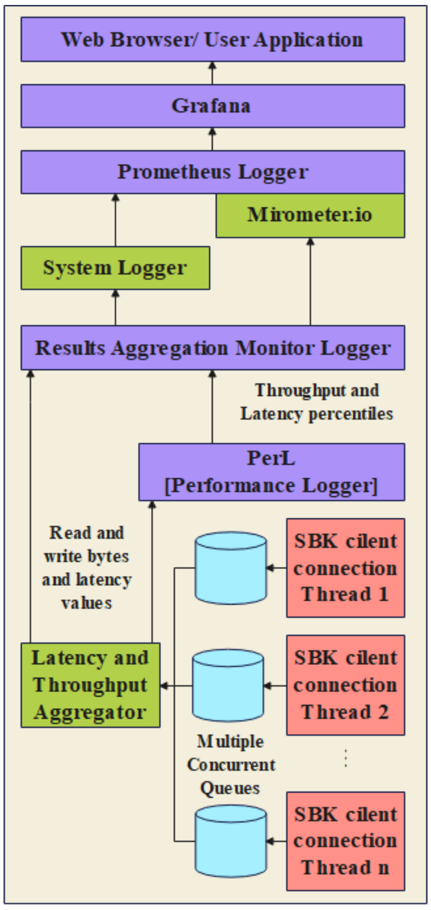
Web Browser/ User Application
Grafana
Prometheus Logger
Mirometer.io
System Logger
Results Aggregation Monitor Logger
Throughput and Latency percentiles
PerL [Performance Logger]
Read and write bytes and latency values
SBK cilent connection Thread 1
SBK cilent connection Thread 2
SBK cilent connection Thread n
Latency and Throughput Aggregator
Multiple Concurrent Queues

Figure 1. Components of storage benchmark kit

2.1. Drill down search

The drill-down feature improves the effectiveness through allowing a detailed analysis of data at different levels, which further supports in determing and solving particular effectiveness bottlenecks. It minimizes the storage necessities through optimizing the data storage and retrieval processes. Generally, one-to-one relationship of the keywords are identified in big data, and when the user searches any keyword, it is considered the search result. In this research, the keywords are stored in the page, in relation to a range of binary files, while the data is organized as a categorization of the bytes. The two keyword binary files of keyword header and data file are written in the crawling process. In this process, a crawler program [22], [23] transfers the web page content and analyses keywords in the search process. The keywords identified in the title of the page contain maximum rank values, while the keywords identified in the meta tag, body and uniform resource locator (URL) contain minimum rank values.

2.1.1. Keyword header file

The header file involves a meta data which is trivial in size and involves an offset information of the data file. Table 1 displays the format of keyword header file. In Table 1, the size of 32-bit data is depicted as the total number of keywords identified for the category search process. The n* 32-bit depicts the number of keyword identifier (ID) in the first column and n* 32-bit depicts the page's offset information about the keyword. The page's keywords represent the page ID in a data file that are quickly loaded into significant memory when the size of the header file is small.

Table 1. Format of the keyword header file

Counting of keywords	Keyword ID data	Keyword offset data
32-bit	32-bit * number of keywords	32-bit * number of keywords

2.1.2. Keyword data file

The keyword data file is reached to the gigabyte size because of a circumstance where the data is jobless. It is majorly based on the count of the header file keywords. If the data file is large, it is loaded into memory and the data offsets are obtained from the significant memory. Subsequently, in-memory data access is speedy when related to the disk-based access, and the offset of the keyword in a header file is utilized as the catalogue. Through the utilization of this offset keyword, the needed page outcomes are obtained from the data. With respect to enhancing the access speed of the data file disc, the solid-state drive (SSD) drives are chosen to accumulate the data file rather than the hard disk drive (HDD) drives.

2.2. Storage benchmark kit

Benchmarking [17] is the significant process for estimating the performance of the storage systems. The benchmarking permits users to compare the different stage solutions and understand how efficiently it performs over particular workloads. Benchmarking is important for estimating the effectiveness of storage systems for enabling users compare various solutions and assess their efficiency under different workloads. It provides a systematic way to measure performance, determing bottlenecks, alongside making informed decisions about system improvements. In this research, the necessities of the benchmarking design for SBK are discovered and investigation into the three stages of the process of benchmark engineering are carried out.

2.2.1. Understanding the consideration of SBK-design

The design consideration of SBK understanding is significant to be performed before splitting the data file for benchmarking. The significant target of the SBK is to deliver a flexible and robust framework, which effectively estimates various storage systems' performances. Few primary design considerations are mentioned below:

- Diversity of the workload: the SBK helps different workloads replicate real-world applications. It is applicable for the development of different read and write operations with arbitrary and random access for pretense in accordance to per the requirements for various applications.
- Scalability: the benchmarking framework has the capability to be scaled with the storage system over the test. It maintains large datasets for flexibility in the distributed storage setups.
- Configurability: the SBK permits users to arrange benchmark parameters for ensembling their particular use cases. This involves adjusting the sizes of the data, number of synchronized operations, and input or output (I/O) patterns.

2.2.2. Methods and techniques to run SBK benchmark

In this section, the methods and techniques necessary for running SBK benchmarking are discussed. The benchmark manufacturing process involves three significant stages of training, execution and post-processing. The detailed information of these stages is described below:

- Training stage: this is the primary step where the benchmark environment is set up. It consists of the selection of a suitable storage system by arranging hardware and installing the significant software. Furthermore, the benchmark parameters of the workload types, size of the data, and concurrency are described.
- Execution stage: after the completion of training, the benchmark is run with the selected configuration. The SBK produces workload on the storage system and estimates the complex performance metrices of latency and throughput.

– Post-processing stage: after the benchmark execution, the collected data is analyzed and processed. This stage consists of removing outliers with an average estimation and the development of comprehensive reports to outcome interpretation.

3. DATA STREAM MANAGEMENT AND FILE STORAGE PERFORMANCE USING KAFKA-ML

Kafka-ML for a SBK allows the real-time data streaming and ML-driven analysis, enabling for dynamic workload adaptation and optimized performance. This combination improves the continuous monitoring and predictive maintenance. This further makes the benchmark kit more receptive and efficient in controlling compound storage environments. The distributed log in Kafka allows users to move the log and read data streams based on their requirements. It is helpful when the system has to process data once destroyed, requiring to improve an entire data stream. In the conventional message queue systems, every message has the chance to be removed after consumption, and the datastore is required to assure the data, even in loss conditions.

In continuation to this, the data streams are instead arranged to be kept in a log and are reused to train the other deployed arrangements. The ML approaches are developed to direct the whole data stream. The necessity of the data provides the respective control message to an anticipated deployment arrangement in Kafka with the recognized retention policy. Figure 2 depicts the data stream management in Kafka-ML. In Figure 2, the initial data stream is directed through control message (C1) to an arranged configuration, and C1 is resent to permit the configuration C2 to utilize a similar data stream. In the existing distributed log sate, the data stream is destroyed and is not reused longer for other deployed arrangement. A data stream integrated with C2 is directed to the deployed configuration D3 and D5 for reutilization. Eventually, a streaming of the data is utilized for training and evaluation processes, while the control messages are sent only when the data stream is completed.

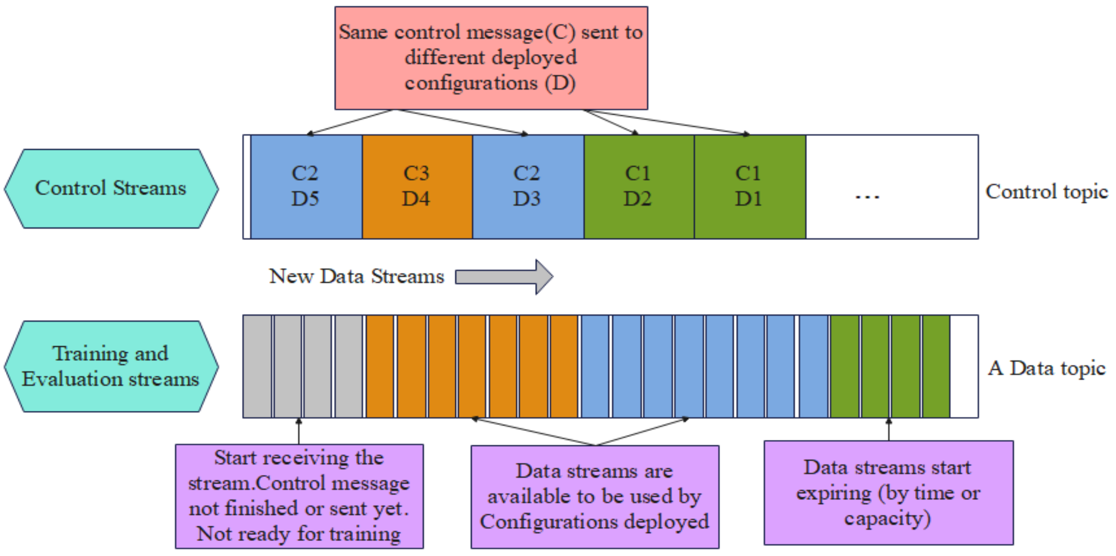
Same control message(C) sent to different deployed configurations (D)
Control Streams
C2 D5
C3 D4
C2 D3
C1 D2
C1 D1
...
Control topic
New Data Streams
Training and Evaluation streams
A Data topic
Start receiving the stream.Control message not finished or sent yet. Not ready for training
Data streams are available to be used by Configurations deployed
Data streams start expiring (by time or capacity)

Figure 2. Data stream management by Kafka-ML

To permit training and evaluation tasks with the data stream, the control message specifies both data streams and their positions in the distributed log. The Kafka-ML utilizes control messages to communicate the accurate position of the data streams to the deployed configurations. In a Kafka-ML web user interface, it is applicable where the user realizes the data stream which is then sent and reused for other system configurations. As mentioned prior, the retention policy of the Kafka determines this behavior. The Kafka removal retention policy is discussed below:

– Retention for bytes: maintains the largest size to which the partition expands before the Kafka begins to remove the old segments to free up space.
– Retention for ms: maintains maximum time for which the log is considered, prior the older segments being removed to free up the space.

3.1. Big data storage system

Big data storage systems are developed to manage large amounts of data. They scale horizontally through adding more nodes to the system, accepting the enhanced data volumes without a substantial drop in performance. The cloud-based Hadoop environment supports for large traffic from the users and data owners by the help of MapReduce [24], [25] context. The clustering, indexing and compression elements play an important part in big data storage systems. This research utilizes these elements to enhance the storage systems. Preceding to storing the data into a cloud server, the data is clustered to minimize the storage space and determine the time for users and data owners. The access control scheme for data user is controlled in the cloud server that updates the data once the ciphertext is exchanged through the data owner. The clustering is executed by the utilization of density-based spatial clustering of applications with noise (DBSCAN) [26], [27] approach. Based on the data points, it groups similar data points into an individual group by the utilization of Euclidean Distance. There are two parameters examined in a DBSCAN approach which are midpoints and 'eps'. A significant aim of this approach is to identify the structures and integration data efficiently. This approach is helpful and appropriate for identifying patterns and to predict the data points. The cluster system (CS) involves n number of domain servers and the number of clustered data partitions are applied into domain server. Every domain server handles the tree for obtainable data partitions, which is developed through the Fractal Tree Index, hence needing the minimum individual searching time and appropriate insertions for the removal of data.

4. RESULTS AND DISCUSSION

In this research, the proposed method is implemented using SBK with certain system requirements. Table 2 represents the experimental setup of the software and hardware requirements of the proposed method. The effectiveness of the proposed method is validated on the basis of two different performance metrices, throughput and latency.

Table 2. Experimental setup of the software and hardware requirements

Components	Remarks
No. of computing nodes	4 nodes
Central processing unit (CPU)	4 CPU each of 64-bit 2.6 GHz
Random access memory (RAM)	16 GB
Hard disk per node	HDD Size 3 TB
Operating system	Windows 10 OS

4.1. Performance analysis

In this section, the proposed method's performance benchmarking is evaluated based on two performance metrices of the read and write operations. In Section 4.1.1 and 4.1.2, the performance benchmarking of read and write operations is presented. The individual frameworks like Kafka and SBK are compared with the Kafka-ML based SBK to validate the outcomes for both read and write operations.

4.1.1. Read operation

Table 3 and Figure 3 represent the read operation Kafka and SBK's throughput performance on the benchmarking task. Table 4 and Figure 4 display the read operation of Kafka and SBK latency performance benchmarking task. The different data bytes such as 10, 100, 1,000, 10,000, 100,000 and 1,000,000 are used to validate the effectiveness of the proposed method.

4.1.2. Write operation

Table 5 and Figure 5 display the write operation of Kafka and SBK throughput performance benchmarking task. Table 6 and Figure 6 exhibit the write operation of Kafka and SBK's latency performance benchmarking task. The different data bytes of 10, 100, 1,000, 10,000, 100,000 and 1,000,000 are used to validate the effectiveness of the proposed method.

Table 3. Read operation for throughput performance

Methods	Data bytes					
	10	100	1,000	10,000	100,000	1,000,000
Kafka	15	110	180	130	360	310
SBK	18	116	187	145	376	323
Kafka-ML based SBK	20	120	200	150	380	330

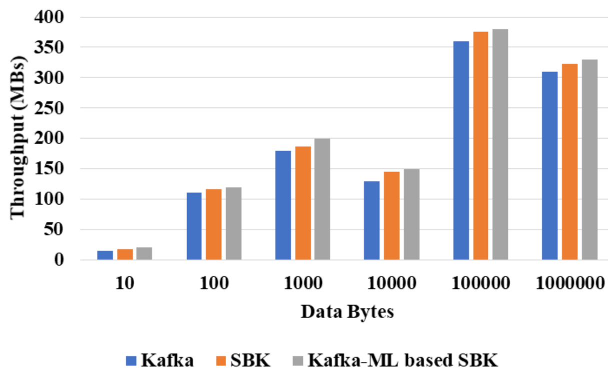

Figure 3. Graphical representation of read operation for throughput performance Kafka and SBK

Table 4. Read operation for latency performance

Methods	Data bytes					
	10	100	1,000	10,000	100,000	1,000,000
Kafka	80	90	1750	950	600	80
SBK	78	87	1578	810	580	76
Kafka-ML based SBK	70	80	1560	800	550	70

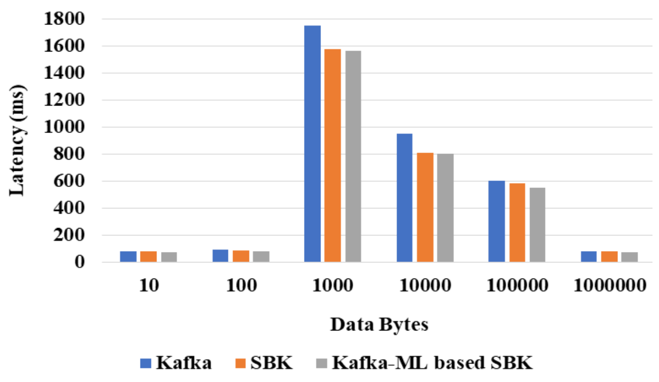

Figure 4. Read operation for latency performance

Table 5. Write operation for throughput performance Kafka and SBK

Methods	Data bytes					
	10	100	1,000	10,000	100,000	1,000,000
Kafka	10	105	174	125	350	306
SBK	13	109	185	135	360	348
Kafka-ML based SBK	15	110	191	146	366	345

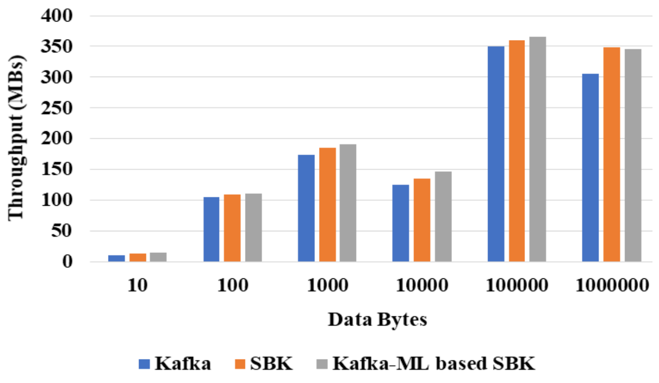

Figure 5. Graphical representation of write operation for throughput performance

Table 6. Write operation for latency performance

Methods	Data bytes					
	10	100	1,000	10,000	100,000	1,000,000
Kafka	70	82	1600	928	590	75
SBK	68	80	1580	918	567	74
Kafka-ML based SBK	65	76	1550	900	570	73

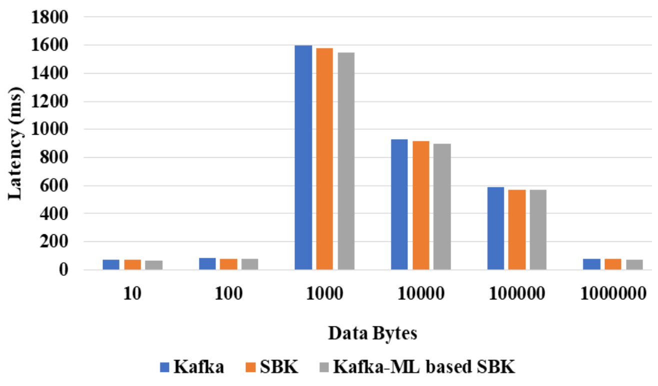

Figure 6. Write operation for latency performance

4.2. Comparative analysis

Table 7 exhibits the comparative analysis of the proposed method with the existing storage performance methods such as SBK_Kafka [16] and drill down [20]. This section discusses the analysis of the proposed method based on the writing operation on different number of data bytes from 10 to 100,000. The effectiveness of the proposed method is validated on two performance metrics of throughput and latency with different number of bytes. Table 8 represents the comparative analysis of the proposed method based on a search time with different data levels.

Table 7. Comparative analysis of the proposed method using throughput and latency

Performance metric	Methods	Data bytes				
		10	100	1,000	10,000	100,000
Throughput (MBs)	SBK_Kafka [16]	15	110	180	130	360
	Proposed Kafka-ML based SBK	20	120	200	150	380
Latency (ms)	SBK_Kafka [16]	80	90	1750	950	600
	Proposed Kafka-ML based SBK	70	80	1560	800	550

Table 8. Comparative analysis of the proposed method using search time with different data levels

Performance metric	Method	Data level			
		Main	1	2	3
Search time (ms)	Drill down approach [20]	55	60	65	82
	Proposed Kafka-ML based SBK	50	53	62	79

4.3. Discussion

In this section, the achievement of the proposed Kafka-ML based SBK framework is discussed along with the limitations of the existing methods. The existing works have the limitations of varied benchmarking outcomes when utilized with various hardware setups, limited memory capacity, creation of overhead, extended read operations that hampered the system performance, and resisted approach when dealt with new data types due to the lack of effective fitness of the existing categories. In order to overcome these limitations, this research proposes the Kafka-ML based SBK for estimating the storage performance for a large number of data files effectively. Through influencing the Kafka's robust streaming capabilities combined with ML, the SBK dynamically adjusts to the mutable workloads, making sure effective data processing and storage management. The drill down approach is used to minimize the storage requirements by the utilization of two binary streams, keyword header file and keyword data file. The proposed method attains a maximum throughput and minimum latency of 20 MBs and 70 ms, respectively. The combination of Kafka-ML into SBK effectively improves real-time data processing and performance optimization. The

Kafka-ML has the capability to maintain the continuous data streams which enables for dynamic workload adaptation, resulting in the most effective storage management.

As compared with SBK_Kafka [16], the Kafka-ML-based SBK performs effectively and exhibits superior outcomes better due to its real-time flexibility and scalability. The proposed Kafka-ML-based SBK approach enables for effective analysis, making it appropriate for advanced and data-intensive environments. However, the Kafka-ML-based SBK introduces challenges such as the requirement for expertise in data streaming and ML, which constraints its availability for some users. This research aims to estimate the integration of Kafka-ML into a SBK to improve real-time data processing and performance optimization. Kafka-ML-based SBK demonstrates an effective tool for optimizing storage performance in real-time, providing significant advantages over the existing approaches. The proposed method solves the disadvantages of existing methods through integrating the strength of Kafka-ML and SBK. The significance of the research lies in its latent to enhance the reliability and throughput of the SBK.

5. CONCLUSION

The Kafka-ML-based SBK represents significant advantages in attaining maximum throughput and minimum latency over different configurations. Through using the drill-down feature with a binary file system, the framework effectively minimizes the disk space necessities and improves the search effectiveness compared to conventional file systems. This approach not only improves the efficiency of data retrieval but also optimizes storage management, making it a valuable tool for fine-tuning storage systems and managing anticipated workloads. The SBK and Drill down feature is important in fine-tuning the storage system outcomes for guaranteeing the maintenance of the anticipated workloads efficiently. Because of this, obtaining data from the binary file system is faster as it performs as an effective storage minimization model in the drill down search. The proposed Kafka-ML-based SBK attains the maximum throughput of 20 MBs, 120 MBs, 200 MBs, 150 MBs and 380 MBs at the data bytes of 10, 100, 1,000, 10,000 and 100,000 respectively, as compared to the existing method, SBK_Kafka. The future work will focus on expanding the Kafka-ML based SBK to further enhance its data storage capabilities, aiming to address emerging challenges and contribute to a more efficient storage solutions in the field.

REFERENCES

[1] M. Bin Saif, S. Migliorini, and F. Spoto, “Efficient and secure distributed data storage and retrieval using interplanetary file system and blockchain,” *Future Internet*, vol. 16, no. 3, pp. 1–13, Mar. 2024, doi: 10.3390/fi16030098.

[2] D. Blum *et al.*, “Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings,” *Journal of Building Performance Simulation*, vol. 14, no. 5, pp. 586–610, 2021, doi: 10.1080/19401493.2021.1986574.

[3] R. Hui and Y. Kang, “Design of high performance distributed storage system,” in *International Conference on Computer Network Security and Software Engineering (CNSSE 2023)*, Jun. 2023, vol. 12714, pp. 291–296, doi: 10.1117/12.2683549.

[4] S. Ahmad and S. Mehfuz, “Efficient time-oriented latency-based secure data encryption for cloud storage,” *Cyber Security and Applications*, vol. 2, pp. 1–10, 2024, doi: 10.1016/j.csa.2023.100027.

[5] A. Manni, A. Caroppo, G. Rescio, P. Siciliano, and A. Leone, “Benchmarking of contactless heart rate measurement systems in ARM-based embedded platforms,” *Sensors*, vol. 23, no. 7, pp. 1–15, Mar. 2023, doi: 10.3390/s23073507.

[6] V. Kjorveziroski and S. Filiposka, “Kubernetes distributions for the edge: serverless performance evaluation,” *The Journal of Supercomputing*, vol. 78, no. 11, pp. 13728–13755, Jul. 2022, doi: 10.1007/s11227-022-04430-6.

[7] E. Gamess and S. Hernandez, “Performance evaluation of different raspberry Pi models for a broad spectrum of interests,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, pp. 819–829, 2022, doi: 10.14569/IJACSA.2022.0130295.

[8] I. U. Akgun, A. S. Aydin, A. Shaikh, L. Velikov, and E. Zadok, “A machine learning framework to improve storage system performance,” in *Proceedings of the 13th ACM Workshop on Hot Topics in Storage and File Systems*, Jul. 2021, pp. 94–102, doi: 10.1145/3465332.3470875.

[9] A. Lazidis, K. Tsakos, and E. G. M. Petrakis, “Publish–subscribe approaches for the IoT and the cloud: functional and performance evaluation of open-source systems,” *Internet of Things*, vol. 19, pp. 1–47, Aug. 2022, doi: 10.1016/j.iot.2022.100538.

[10] L. M. Lim, J.-W. Park, and K. Hadinoto, “Benchmarking the solubility enhancement and storage stability of amorphous drug–polyelectrolyte nanoplex against co-amorphous formulation of the same drug,” *Pharmaceutics*, vol. 14, no. 5, pp. 1–24, May 2022, doi: 10.3390/pharmaceutics14050979.

[11] C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, and B. Rubio, “Kafka-ML: connecting the data stream with ML/AI frameworks,” *Future Generation Computer Systems*, vol. 126, pp. 15–33, Jan. 2022, doi: 10.1016/j.future.2021.07.037.

[12] H. B. S. Reddy, R. R. S. Reddy, R. Jonnalagadda, P. Singh, and A. Gogineni, “Analysis of the unexplored security issues common to all types of NoSQL databases,” *Asian Journal of Research in Computer Science*, vol. 14, no. 1, pp. 1–12, May 2022, doi: 10.9734/ajrcos/2022/v14i130323.

[13] M. A. Mhana, A. Khalifeh, and S. Alouneh, “Performance comparison of big data processing utilizing SciDB and Apache Accumulo databases,” in *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*, Dec. 2022, pp. 1–5, doi: 10.1109/FMEC57183.2022.10062513.

[14] N. V. Patil, C. R. Krishna, and K. Kumar, “KS-DDoS: Kafka streams-based classification approach for DDoS attacks,” *The Journal of Supercomputing*, vol. 78, no. 6, pp. 8946–8976, Apr. 2022, doi: 10.1007/s11227-021-04241-1.

[15] H. Jin, W. G. Choi, J. Choi, H. Sung, and S. Park, “Improvement of RocksDB performance via large-scale parameter analysis and optimization,” *Journal of Information Processing Systems*, vol. 18, no. 3, pp. 374–388, Jun. 2022.

[16] K. D. Hartomo, A. F. Daru, and H. D. Purnomo, "A new approach of scalable traffic capture model with Pi cluster," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 2, pp. 2186–2196, Apr. 2023, doi: 10.11591/ijece.v13i2.pp2186-2196.

[17] K. Munegowda and N. V. Sanjay Kumar, "Design and implementation of storage benchmark kit," in *Emerging Research in Computing, Information, Communication and Applications: ERCICA 2020*, vol. 2, Springer Singapore, 2022, pp. 45–62, doi: 10.1007/978-981-16-1342-5_5.

[18] J. Gomez-Luna, I. El Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a new paradigm: Experimental analysis and characterization of a real processing-in-memory system," *IEEE Access*, vol. 10, pp. 52565–52608, May 2022, doi: 10.1109/ACCESS.2022.3174101.

[19] K. Munegowda and N. V. Sanjay Kumar, "SLC: sliding latency coverage factors for optimal performance benchmarking of storage systems," in *2022 3rd International Conference for Emerging Technology (INCET)*, May 2022, pp. 1–8, doi: 10.1109/INCET54531.2022.9825170.

[20] M. Gotz, S. Khriji, R. Cheour, W. Arief, and O. Kanoun, "Benchmarking-based investigation on energy efficiency of low-power microcontrollers," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 7505–7512, Oct. 2020, doi: 10.1109/TIM.2020.2982810.

[21] N. Ragavan and C. Y. Rubavathi, "A novel big data storage reduction model for drill down search," *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 373–387, 2022, doi: 10.32604/csse.2022.020452.

[22] J. Chou and S. Hsu, "Automated prediction system of household energy consumption in cities using web crawler and optimized artificial intelligence," *International Journal of Energy Research*, vol. 46, no. 1, pp. 319–339, Jan. 2022, doi: 10.1002/er.6742.

[23] S.-H. Chae, S.-M. Baek, J. Lee, and K.-J. Cho, "Agile and energy-efficient jumping–crawling robot through rapid transition of locomotion and enhanced jumping height adjustment," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 5890–5901, Dec. 2022, doi: 10.1109/TMECH.2022.3190673.

[24] J. Mary Arockiam and A. C. Seraphim Pushpanathan, "MapReduce-iterative support vector machine classifier: novel fraud detection systems in healthcare insurance industry," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, pp. 756–769, Feb. 2023, doi: 10.11591/ijece.v13i1.pp756-769.

[25] M. B. Al-Masadeh, M. S. Azmi, and S. S. Syed Ahmad, "Tiny datablock in saving Hadoop distributed file system wasted memory," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 2, pp. 1757–1772, Apr. 2023, doi: 10.11591/ijece.v13i2.pp1757-1772.

[26] V. D. Babu and K. Malathi, "Large dataset partitioning using ensemble partition-based clustering with majority voting technique," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 29, no. 2, pp. 838–844, Feb. 2023, doi: 10.11591/ijeecs.v29.i2.pp838-844.

[27] N. M. Mahfuz, M. Yusoff, and Z. Idrus, "Clustering heterogeneous categorical data using enhanced mini batch K-means with entropy distance measure," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, pp. 1048–1059, Feb. 2023, doi: 10.11591/ijece.v13i1.pp1048-1059.

BIOGRAPHIES OF AUTHORS

Sanjay Kumar Naazre Vittal Rao is professor in the Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur. He Pursued his Ph.D. degree in computer science and engineering at Visvesvaraya Technological University, Belagavi, Karnataka, India. He Holds a M.Tech. and B.E. degree in computer science and engineering at VTU, Belagavi. His research areas are: big data and storage system benchmarking. His research interests include storage systems, performance analysis, analytics, and distributed systems. He can be contacted at email: sanjaynv@gmail.com.

Anitha Chikkanayakanahalli Lokesh Kumar is professor in the Department of Computer Science and Engineering, Kalpataru Institute of Technology, Tiptur. She pursued her Ph.D. degree in computer science and engineering at Visvesvaraya Technological University, Belagavi, Karnataka, India. She has published several research papers, international conference papers since 2011, She is an active member of ISTE. She can be contacted at email: clanitha@gmail.com or kitanitha1@gmail.com.

Subhash Kamble received the B.E. degree in computer science and engineering from Visvesvaraya Technological University, Karnataka, India, in 2006 and the M.E. degree in computer science and engineering from UVCE, Bengaluru, Bangalore University, Bengaluru, Karnataka, in 2009. He is currently pursuing a Ph.D. degree in computer science and engineering at Bangalore University, Bengaluru, India. He has published 5 articles in refereed international journals and conferences. His research interest includes data mining, machine learning, and big data analytics. He can be contacted at email: subhashkamble@gat.ac.in.