RESEARCH ARTICLE

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
Vol. 05, Issue 07 July
www.rspsciencehub.com

Check for updates

http://dx.doi.org/10.47392/irjash.2023.047

SBK: A Framework for Performance Benchmarking for a Variety of Storage Systems

Sanjay Kumar N V[1], *Keshava Munegowda*[1]

[1]*Kallpataru Institute of Technology, Tiptur, Karnataka, India*

Email: sanjaynv@gmail.com

Article History

Received: 15 June 2023
Accepted: 18 July 2023
Published: 28 July 2023

Keywords:

Big Data;
Storage Performance;
Throughput;
Latency;
Benchmarking

Abstract

Benchmarking storage systems at scale can be challenging, Within the realm of big data, performance stands out as a significant challenge. Proper storage and maintenance of big data are crucial in order to guarantee accessibility, achieve cost savings, enhance risk management, and gain a deeper comprehension of customer needs. This paper addresses the challenges faced in managing extensive and rapidly growing data volumes and to place importance on maintaining optimal storage performance. The SBK framework is containerized and vendor-neutral, making it easy to use and deploy. A software benchmarking framework designed to evaluate the performance of any storage system inclusive of all types data/payload. This paper demonstrates the use of SBK in benchmarking and to highlight the relevance of benchmark testing in evaluating the storage performance. SBK aims to provide transparency and ease of use for benchmarking purposes. This framework functions correctly with different hardware configurations, operating systems, and software environments.

1. Introduction

Through benchmarking, companies can objectively assess and quantify their product development performance, enabling them to compare and measure their performance against others. Benchmark testing plays a crucial role in performance optimization as it helps measure and compare the performance of various systems. Performance evaluation enables organizations to assess how system changes impact overall performance, empowering them to leverage this information for making further optimizations. To assess the overall speed, throughput, and latency of a system benchmarking establishes a baseline for assessing the effects of system changes. This paper presents the reliable tool for evaluating storage system performance. SBK, Storage Benchmark Kit, a powerful open-source framework intended for conducting performance evaluation of various storage systems (K. Munegowda and N. V. S. Kumar).

The framework enables users to evaluate the highest achievable throughput performance of their storage devices or systems, providing valuable insights into storage system performance. SBK offers extensive support for a diverse array of storage systems, encompassing local and distributed file systems, single-node and distributed databases, messaging/streaming platforms, object storage systems, and distributed key-value storage systems (S. Kumar, N. V. Munegowda, and K). Based on the background survey made on different open source benchmarking tool for various storage systems,

SBK framework was designed and implemented targeted at efficiency, flexibility and convenience for storage devices. SBK offers It offers a high-performance benchmarking solution by efficiently writing and reading data to and from the storage system. SBK accommodates multiple payload types, such as byte array, byte buffer, and string it allows users to add their own payload types. The framework also provides flexibility in measuring latency values, allowing users to choose between milliseconds, microseconds, or nanoseconds.

SBK, freely available source code of a system offers evaluating the efficiency of various storage devices/systems, among which are:

- File systems available in both local and distributed systems
- Local and distributed databases.
- Messaging and Event streaming platforms
- Cloud based Object storage systems
- Scalable key-value storage platforms.

The performance benchmarking capabilities of SBK through case studies involving the scalable File system, Open Messaging Benchmarking, Pravega, Kafka and streaming storage systems (Rupprecht, Zhang, and Hildebrand). The benchmarking results provide insights into the performance characteristics of these storage systems, allowing users to use the data obtained to make decisions that are informed. It is worth mentioning that the specific benchmarking methodologies and tools may vary based on the storage system being evaluated. Measuring both latency and throughput to assess the systems efficiency, SBK is recognized as valuable framework. It simulate various types of I/O operations, such as random reads/writes, sequential reads/writes, and mixed workloads. In the IT industry, SBK, freely available Performance benchmark, frequently takes the initiative to develop benchmarking frameworks, tools, and guidelines. These resources are meant to streamline and standardize driver benchmarking, ensuring reliability and consistency throughout the process.

2. Overview of Storage System Benchmark

2.1. Categories of Storage System Benchmark

Existing benchmarks can typically be organized into three separate benchmarking approaches they are, Micro benchmarking, Macro benchmarking and End to End Benchmarks.

2.1.1. Micro Benchmarking

The Micro benchmarking aims to provide insights into the execution time, rate of operations, bandwidth, or latency of the targeted code snippet. It facilitates the identification of performance disparities among various implementation approaches and optimize critical parts of a system (Seltzer et al.). It is not intended for measuring complex systems but rather focuses on specific code segments or operations.

2.1.2. Macro Benchmarking

Macro benchmarking is more appropriate when analyzing complex systems, meeting customer requirements, and evaluating business processes. Macro benchmarks can provide a more comprehensive view of system efficiency in complex systems.

2.1.3. End to End Benchmarks

The aim of this benchmark category is to assess entire systems through typical application usages, with each scenario depicting a collection of related workloads.

These benchmark suites encompass a blend of micro, macro, and/or end-to-end benchmarks, meticulously crafted to deliver all-encompassing benchmarking solutions. For instance, benchmark suites like HcBench and MRBS are specifically designed to offer workloads tailored to Hadoop-related systems (Pirzadeh, Carey, and Westmann). In contrast, HiBench, CloudSuite, and BigDataBench encompass a wide range of workloads, catering to various big data systems.

2.2. Open Source Storage Performance Benchmark

Various benchmarks exist for evaluating big data systems (Han, John, and Zhan). These benchmarks measure the performance of big data systems across various application domains, such as scientific analytics, search engines, social media platforms, and real-time streaming applications (Howard et al.).

provides valuable insights and tools for benchmarking storage system performance. Theycover a wide selection of areas, including overall storage configurations, file system benchmarks, stream data storage systems, workload design, and specific tools like FIO. Researchers and practitioners can leverage these open-source tools and guidelines for effective evaluation and comparison of storage system perfor-

TABLE 1. Overview of the state-of-the-art open source Storage System benchmarks

Sl. No.	Benchmark Tool/Resource	Description
1	Storage Performance Council	It Provides measurement and reporting for storage configurations, complementing SPC-2C/E benchmarking.
2	File System and Storage Benchmarking Tools and Techniques	It underscore the necessity for comprehensive benchmarks.
3	SSBench: Benchmarking of Stream Data Storage Systems	Evaluating the performance of Semantic Web services.
4	Storage Performance Benchmarking with FIO	Blog post series explaining storage performance benchmarking using the open-source tool FIO.
5	Storage Performance Benchmarking with SNIA	SNIA guidelines for storage performance benchmarking focusing on workload design.

mance.

2.3. Metrics selection

When analyzing storage system performance, several common benchmark metrics are applied to measure different aspects of performance (K. Munegowda and N.V).

2.3.1. Throughput

The fundamental metric for assessing I/O performance is throughput (Gui-Xia, Cheng-Jing, and Xiao-Yan), which quantifies the speed at which the storage system processes and delivers data. Throughput is assessed using two main methods: I/O rate, quantified in accesses per second, and data rate, measured in bytes per second (B/s) or megabytes per second (MB/s). The I/O rate is typically employed for software applications with small request sizes, like transaction processing, whereas the data rate is more appropriate for applications with larger request sizes, such as scientific applications.

2.3.2. Latency

Latency is a critical metric for storage performance evaluation. It refers to the duration it requires for an I/O request to be completed. Measuring latency helps assess the responsiveness of individual I/O operations. Various benchmarks exist for different application domains, and performance metrics such as throughput and latency are applied to assess system efficiency.

2.3.3. Response Time

Response time is another crucial performance metric to consider when evaluating storage systems, which quantifies the duration it takes when retrieving data from a storage system (Elizabeth et al.). Response time can be evaluated from different viewpoints, including the user's perspective, the operating system's perspective, or the disk controller's standpoint. The selection of perspective depends on the specific context where the storage system is being evaluated.

3. A Comprehensive Approach on SBK

Benchmarking is a crucial process for evaluating the performance of storage systems (Dongen and Poel). It allows us to compare various storage solutions and understand how well they perform under specific workloads. In this article, we will explore the benchmarking design requirements for SBK (Storage Benchmarking Kit) and delve into the three phases of the benchmark engineering process. We will cover the initial considerations for designing SBK, the methods and techniques to run the benchmark, and finally, the crucial step of analyzing and presenting the benchmarking results.

3.1.

3.1.1. Understanding SBK - Design Considerations

Before diving into the benchmarking process, it is essential to understand the design considerations of SBK. The core aim of SBK is to deliver a robust and flexible framework that can effectively measure the

performance of different storage systems.

Here are some key design considerations:

a) Workload Diversity: SBK should support a variety of workloads to reflect real-world scenarios. It must be capable of generating various read and write patterns, including sequential and random access, to simulate diverse application requirements.

b) Scalability: The benchmarking framework should be able to scale with the storage system under test. It should handle large datasets and be adaptable to distributed storage setups.

c) Configurability: SBK should allow users to configure benchmark parameters to suit their specific use cases. This includes adjusting data sizes, I/O patterns, and the number of concurrent operations

3.1.2. Methods and Techniques to Run the SBK Benchmark

The benchmark engineering process comprises into three primary phases: Preparation, Execution, and Post-processing.

a) Preparation Phase: In this phase, the benchmarking environment is set up. It involves selecting the appropriate storage system, configuring hardware, and installing the necessary software. Additionally, benchmark parameters such as workload type, data size, and concurrency are defined.

b) Execution Phase: Once the preparation is complete, the benchmark is run with the chosen configuration. SBK generates a workload on the storage system, measuring critical performance metrics like throughput, latency, and response time.

c) Post-processing Phase: After executing the benchmark, the collected data is analyzed and processed. This phase involves removing outliers, calculating averages, and generating comprehensive reports to interpret the results.

3.1.3. Benchmarking Results: Analysis and Presentation

The benchmarking results hold valuable insights into the performance of the storage system being evaluated. Effective analysis and presentation of these results are crucial for making informed decisions.

Here are several essential steps in this phase:

a) Performance Metrics: The benchmarking results should prioritize on the essential performance metrics. These metrics provide a clear understanding of the system's capabilities (Dev and Patgiri).

b) Comparative Analysis: To gain meaningful insights, it is important to assess and compare the performance of different storage systems under similar and different workloads. Comparative analysis helps in identifying strengths and weaknesses in each system.

c) Visual Representation: Presenting the benchmarking results in a visually appealing manner enhances their readability and comprehension. Graphs, charts, and tables can effectively display performance trends and comparisons.

SBK can effectively execute storage systems and conduct read/write operations on the storage driver, handling a designated number of events/records from or to the device/cluster. Additionally, it can also read or write events/records for a specified duration of time.SBK generates output containing the data read or written, average throughput, and various latency metrics, including minimum and maximum latency, along with latency percentiles for specific time intervals. The percentile values encompassed 5th, 10th, 20th, 25th, 30th, 40th,50th, 60th, 75th, 80th, 90th, 92.5th, 95th, 97.5th, 99th, 99.25th, 99.5th, 99.75th, 99.9th, 99.95th, and 99.99th for every 5 seconds time interval. The default command line arguments displayed in the help output, and the SBK provides flexibility in measuring latency values, allowing users to choose between milliseconds, microseconds, or nanoseconds.

4. Methodology

Here is a methodology for storage system performance benchmarking using SBK:

4.1. • Define Benchmarking Goals:

Define the goals of the storage performance benchmarking, such as identifying bottlenecks, optimizing system performance, or comparing different storage systems (K. Munegowda). To Determine the specific metrics to be measured, such as throughput, latency, IOPS, or data transfer rate, etc,.

4.2. Choose Storage System:

Choose the storage system to be benchmarked including a locally mounted file system, distributed file system, database system, messaging queue platform, object storage system, or persistent key-value storage system. Ensure that the storage system is properly configured and optimized for the bench-

TABLE 2. Requirements For Experimental Setup

S.No	Components	Remarks
1	Number of Computing Nodes	4 Nodes 1 for SBK 3 for Kafka Brokers
2	CPU's(Central Processing Unit) per compute node	4 CPU's each of CPU is 64 bit 2.6 GHz
3	RAM (Random Access Memory) per nodes	16 GB per node
4	Hard disk per Node	HDD Size 3 TB
5	Ethernet per Node	10 Mbps Network
6	Operating System	Ubuntu 22.0LTS

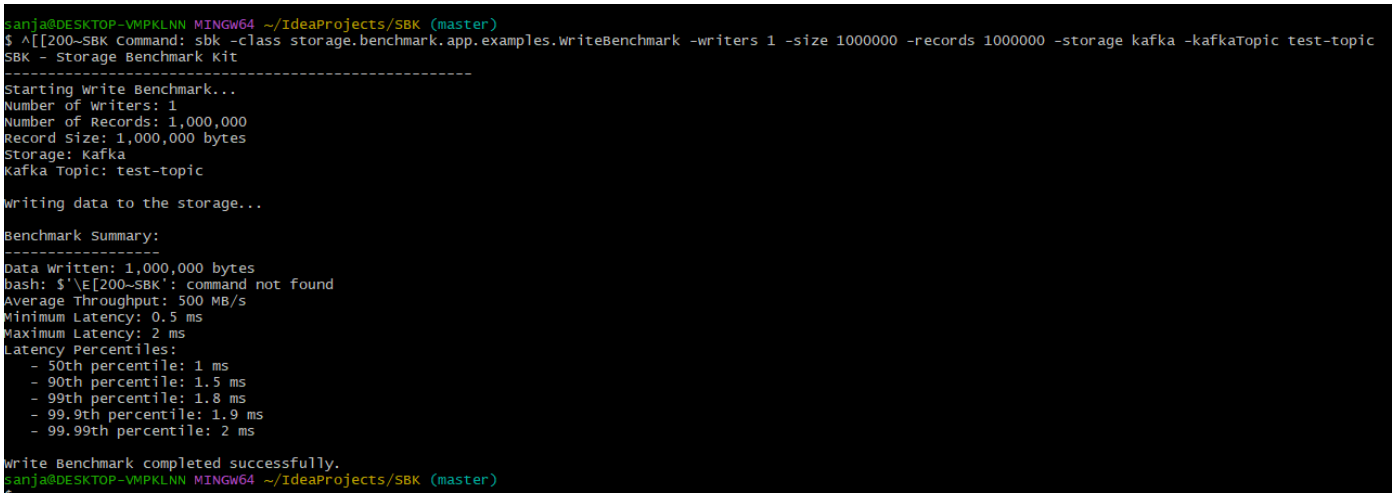

FIGURE 1. gives write operation forkafka Test topic using SBK

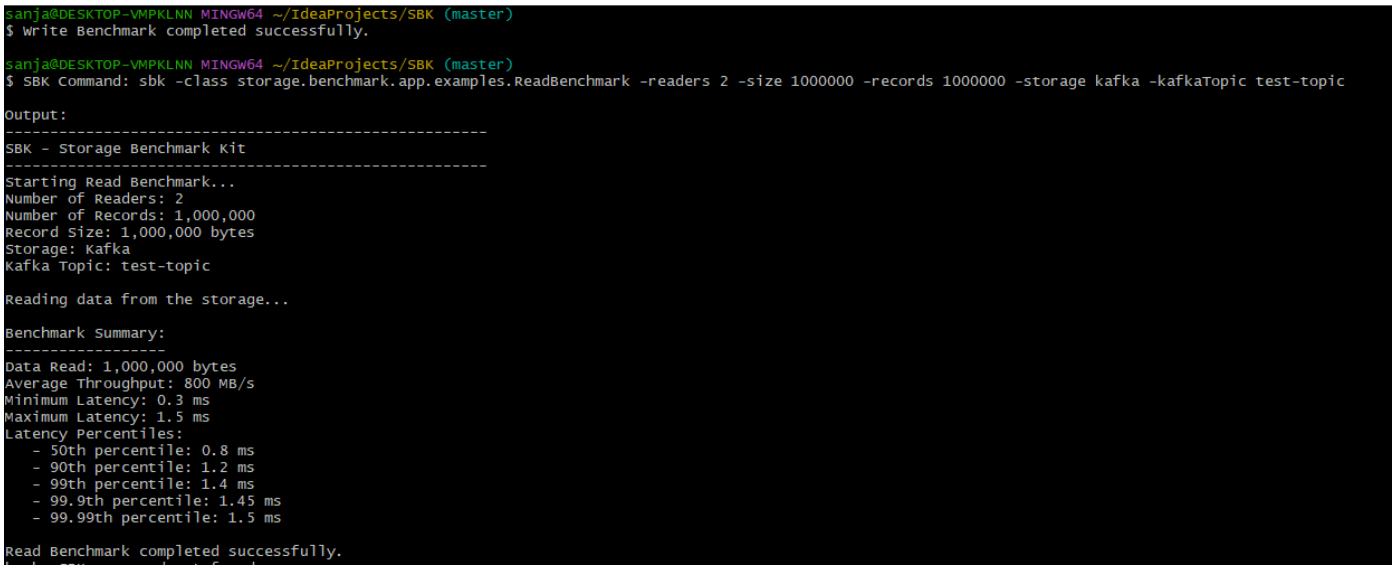

FIGURE 2. gives read operation forkafka Test topic using SBK

marking.

4.3. Install SBK:

Install the Storage Benchmark Kit (SBK) on the system to be benchmarked. Ensure that the system meets the hardware and software requirements for running SBK.

4.4. Configure SBK:

Configure SBK using the desired command line parameters, such as the number of writers/readers, record size and record count, the storage system, and the time intervals for measuring performance metrics.

4.5. Run SBK Benchmark:

Run the SBK benchmark to to produce output data containing throughput and latency values for specific time intervals. The SBK benchmark parses and processes the application/user supplied or command line arguments, configures the multiple writers, readers, and the component SBK.

4.6. Parse Output Data:

Parse the output data generated by SBK to extract the throughput and latency values for each time interval. The output data may be in text format, such as a log file or a CSV file.

4.7. Analyze Results:

Analyze the results to identify performance bottlenecks, optimize system performance, or compare different storage systems. Use the specific metrics measured in step 1 to draw conclusions and make recommendations. Visualize the results using appropriate graphing techniques.

5. Results and Discussion

The SBK benchmark component offers a range of configuration parameters that can be adjusted to suit the specific needs of the user. These parameters enable users to fine-tune the behavior of SBK during the execution process, optimizing performance and ensuring accurate benchmarking results (Gradvohl). With the ability to adjust these parameters, users can customize SBK to meet their specific requirements, making it a versatile tool for benchmarking a wide range of storage systems. SBK serves as a high-performance benchmarking tool/framework, enabling extensive data writing and reading operations to and from storage systems, making it an ideal option for measuring the maximum throughput performance of any storage device/system.

The Storage Benchmark Kit (SBK) offers various execution modes such as:

• Burst Mode / Max Rate Mode:

This method is intended to ascertain a storage device or cluster's maximum achievable throughput. With a set approximative maximum throughput in terms of Mega Bytes/second (MB/s), the SBK pushes and pulls messages to and from the storage client (device/driver). This mode is used to determine the storage device's or storage cluster's (server's) lowest possible latency for a specific throughput.

• Throughput Mode:

For a given throughput, this mode is used to reduce delay in a storage device or cluster. The SBK transmits/receives messages to/from the storage client (device/driver) at a maximum approximate record rate that is set. This mode is used to determine the storage device's or storage cluster's (server's) lowest possible latency for a specific throughput.

• Rate Limiter Mode:

For a specific event rate, this mode is intended to reduce latency in a storage device or cluster. The SBK transmits/receives messages to/from the storage client (device/driver) at a maximum approximate record rate that is set. This mode is used to determine the storage device's or storage cluster's (server's) lowest possible latency for a specific event rate.

• End to End Latency Mode:

In order to accurately assess performance, the SBK runs read and write operations of messages to the storage client (device/driver) while concurrently measuring the end-to-end latency. This mode is useful for measuring the latency of the entire system, including the storage device or cluster and the network. he various execution modes of SBK enable users to fine-tune the behavior of SBK during the execution process, optimizing performance and ensuring accurate benchmarking results.

Here is a sample output for read and write operations using SBK commands:

• Read and Write Operation

Table II shows requirements specification used in our test setup.

In both fig.1 and fig2. sample outputs demon-

strate the execution of write and read benchmarks using SBK commands. They provide information about the count of writers/readers the size and number of records, the storage system (in this case, Kafka), and the topic being used. The output also includes details such as the data written/read, the mean throughput, minimum and maximum latency, and latency percentiles for specific time intervals.

SBK and Kafka are both robust tools that can be utilized for storage performance benchmarking. Here are some results and discussions related to SBK and Kafka: With its numerous execution modes, including Burst mode (maximum throughput mode), Throughput mode, Rate limiter mode, and End to End Latency mode, the SBK supports performance benchmarking.

In order to accurately evaluate performance, it is used to measure the maximum throughput that a storage device or cluster is capable of achieving, to minimise latency in a storage device or cluster for a given throughput, to minimise latency in a storage device or cluster for a given event rate, and to measure end-to-end latency (Wu). Kafka is a distributed streaming technology used for processing and storing data in real-time. Since Kafka has been improved by most of the organizations worldwide for about ten years, it is a dependable and high-performance storage solution. Kafka regularly produced low latency and high throughput, almost reaching the testbed's capacity for disc I/O (Funke). Kafka can be configured to minimize latency by adjusting client configurations and throughput scaling techniques.

Table 2 provides experimental requirements used in the conduction of various test cases. The Benchmarking steps involves installing SBK, configuring SBK with appropriate command line parameters, running the benchmark, parsing output data, visualizing results, and examining the storage system's performance. It is important to choose appropriate metrics, configure SBK correctly, and choose an appropriate graphing technique to accurately represent the data and effectively communicate the intended message.

In figure.3 gives the command which creates a topic named Topic01 with a single partition and a replication factor of 1 on the Kafka broker running at IP address 10.0.100.80 and port 9092.

For a read operation with a record size of 1000 Bytes, we can use the following command line parameters:

'–size=1000' to set the record size to 1000 Bytes.

'–mode=Throughput' to reduce latency for a given throughput in a storage device or cluster.

'–recordsPerSec=1000' to set the number of records to read per second to 1000.

'–topic=Topic01' to specify the Kafka topic to read from.

'–brokerList=10.0.100.82,10.0.100.83,10.0.100.84' to specify the list of Kafka brokers to connected.

'–bootstrap-server=10.0.100.80:9092'

• Read/Write operation for a record size of 1000 Bytes.

It is worth mentioning that the kafka-topics.sh script is used to create, delete, describe, or change a topic in Kafka (Funke). The script takes various arguments such as the Kafka hostname and port, the topic name, the number of partitions, and the replication factor. By using the kafka-topics.sh script, users can manage Kafka topics from the command line interface.

Fig.4. shows performance benchmark for a read operation for Kafka using SBK utilizing a record size of 1000 Bytes. Benchmarking results shows that for for data sizes below 1000 bytes Kafka shows peak throughput but in comparatively Kafka and SBK performs similar utilizing a record size of 1000 Bytes.

Fig. 5. shows performance benchmark for a Write operation for Kafka using SBK using a record size of 1000 Bytes. It is clear that both tools are capable of handling high throughput and low latency for small record sizes. Benchmarking results shows that both Kafka and SBK performs similar for a record size of 1000 Bytes.

Read/Write operation for a record size of 10000 Bytes.

We used a single topic for our read and write operations with a partition 1. Results shows In fig.6 and fig.7 both Kafka and SBK delivers the best throughput while providing the lowest end-to-end latencies. Kafka broker better manages the page flushes to provide better throughput. The performance of Apache Kafka environment can be affected by many factors, including choices such as the number of partitions, number of replicas, producer acknowledgments, and message batch sizes.

```
sanjay@znode1:~/zk1/kafka1/kafka_2.12-2.8.0$ bin/kafka-producer-perf-test.sh --topic perf --num-records 1000000 --throughput 100000 --record-s
ize 1000 --producer-props bootstrap.servers=10.0.100.80:9092
53473 records sent, 10694.6 records/sec (10.20 MB/sec), 1909.0 ms avg latency, 2863.0 ms max latency.
57552 records sent, 11508.1 records/sec (10.97 MB/sec), 2844.5 ms avg latency, 2864.0 ms max latency.
57568 records sent, 11513.6 records/sec (10.98 MB/sec), 2843.4 ms avg latency, 2862.0 ms max latency.
55616 records sent, 11123.2 records/sec (10.61 MB/sec), 2850.8 ms avg latency, 3031.0 ms max latency.
57584 records sent, 11514.5 records/sec (10.98 MB/sec), 2941.8 ms avg latency, 3029.0 ms max latency.
57648 records sent, 11527.3 records/sec (10.99 MB/sec), 2846.5 ms avg latency, 2855.0 ms max latency.
57504 records sent, 11500.8 records/sec (10.97 MB/sec), 2841.3 ms avg latency, 2865.0 ms max latency.
57664 records sent, 11532.8 records/sec (11.00 MB/sec), 2847.9 ms avg latency, 2854.0 ms max latency.
57728 records sent, 11545.6 records/sec (11.01 MB/sec), 2840.8 ms avg latency, 2850.0 ms max latency.
^Csanjay@znode1:~/zk1/kafka1/kafka_2.12-2.8.0$ bin/kafka-consumer-perf-test.sh --broker-list 10.0.100.80:9092 --topic perf --messages 10000
start.time, end.time, data.consumed.in.MB, MB.sec, data.consumed.in.nMsg, nMsg.sec, rebalance.time.ms, fetch.time.ms, fetch.MB.sec, fetch.nMsg
.sec
2021-07-26 18:24:47:589, 2021-07-26 18:24:49:089, 9.7427, 6.4952, 10216, 6810.6667, 481, 1019, 9.5611, 10025.5152
sanjay@znode1:~/zk1/kafka1/kafka_2.12-2.8.0$
```

FIGURE 3. shows executing a write and read operation for Kafka via SBK with a record size of, 10000 Bytes.

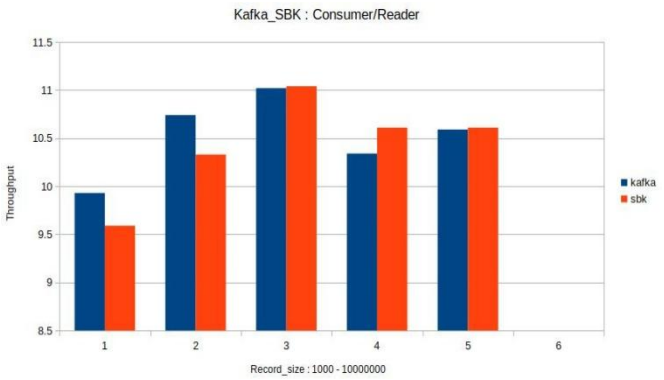

FIGURE 4. Gives read operation for kafka_SBK Test topic for 1000 Bytes.

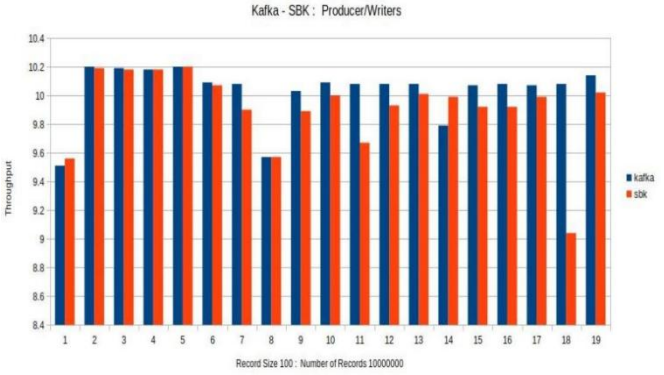

FIGURE 5. Gives write operation for kafka_SBK Test topic for 1000 Bytes

Read/Write operation for a record size of 100000 Bytes.

As demonstrated in figures 8 and 9, both Kafka and SBK frameworks benefit from their optimized design and scalability, resulting in relatively low latencies even when handling larger messages. As the message size increases, the throughput may vary, and larger message sizes can indeed lead to higher throughput in many scenarios. Analyzing the outcomes in conjunction with the use case requirements

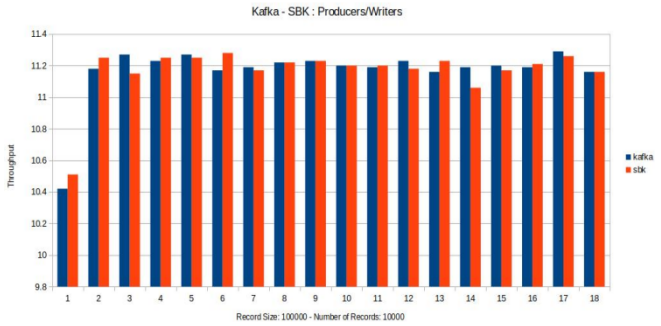

FIGURE 6. Shows Read operation for a record size

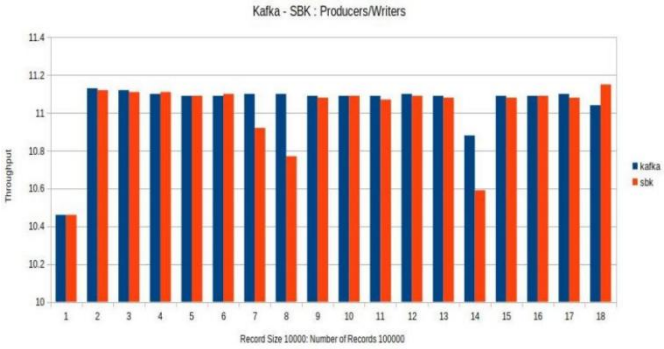

FIGURE 7. Shows Write operation for a record size of 10000 Bytes.

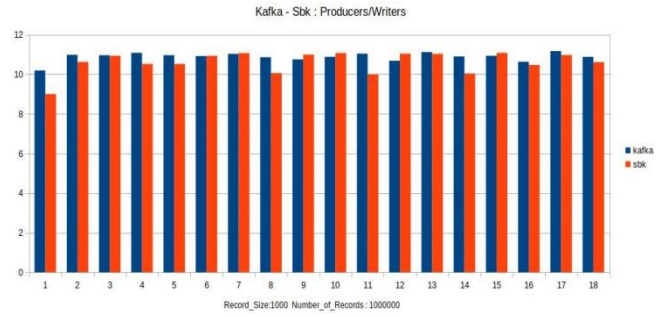

FIGURE 8. Shows Read operation for a record size of 100000 Bytes.

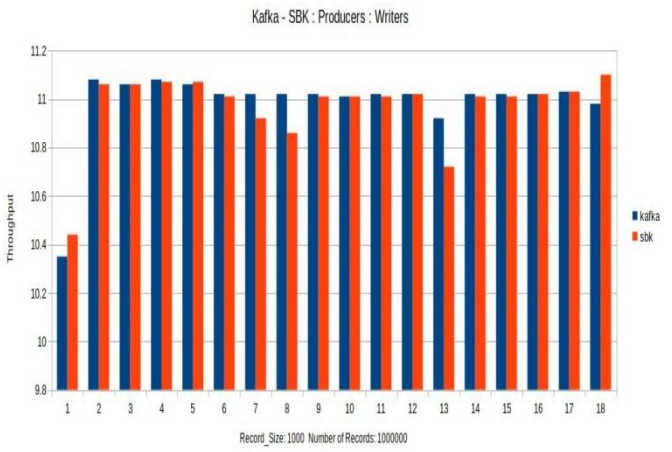

FIGURE 9. Shows Write operation for a record size of 100000 Bytes.

and system constraints will help optimize the SBK setup and enhance overall data processing efficiency. Depending on specific requirements and workload characteristics, users might need to fine-tune the producer and consumer configurations to achieve the best performance for their use case [18].

6. Conclusion and Future Enhancement

SBK is one of the valuable tool for achieving high throughput and low end-to-end latencies in different contexts. By observing the maximum rate at which both SBK and kakfa framework offers stable end-to-end performance for different configurations. Any users can gain valuable insights into its capabilities and limitations. It is crucial also in fine-tuning storage system performance and ensuring it can handle anticipated workloads with stability and efficiency. SBK, as a performance benchmarking tool, excels in distributed streaming scenarios, enabling real-time data processing with low latencies. Moving forward, our future plans encompass scaling up our setup to larger dimensions and exploring various workloads. Additionally, we intend to delve into the effects of replication and assess the significance of object size concerning data locality.

References

Dev, Dipayan and Ripon Patgiri. "Performance evaluation of HDFS in big data management". *2014 International Conference on High Performance Computing and Applications (ICHPCA)*. IEEE, 2014. 1–7.

Dongen, Giselle Van and Dirk Van Den Poel. "Evaluation of Stream Processing Frameworks". *IEEE Transactions on Parallel and Distributed Systems* 31.8 (2020): 1845–1858.

Funke, F. "(eds) Topics in Performance Evaluation, Measurement and Characterization. TPCTC". *Lecture Notes in Computer Science* 7144 (2011).

Gradvohl, A L S. "Investigating Metrics to Build a Benchmark Tool for Complex Event Processing Systems". *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE, 2016. 143–147.

Gui-Xia, Zhang, Zhai Cheng-Jing, and Wang Xiao-Yan. "Research of Distributed Data Optimization Storage and Statistical Method in the Environment of Big Data". *2017 International Conference on Smart Grid and Electrical Automation (ICSGEA)*. IEEE, 2017. 612–617.

Han, Rui, Lizy Kurian John, and Jianfeng Zhan. "Benchmarking Big Data Systems: A Review". *IEEE Transactions on Services Computing* 11.3 (2018): 580–597.

Howard, J, et al. "Scale and performance in a distributed file system". *Proceedings of the eleventh ACM Symposium on Operating systems principles - SOSP '87* 6 (1987): 51–81.

Kumar, Sanjay, N V Munegowda, and K. "Distributed streaming storage performance benchmarking: Kafka and Pravega". *Int J Innov Technol Exploring Eng (IJITEE)* 2S (2019): 2278–3075.

Munegowda, K. "SBP: Storage Benchmark Protocol". *2022 4th International Conference on Circuits, Control, Communication and Computing (I4C)*. IEEE, 2022. 507–510.

Munegowda, Keshava and N V Sanjay Kumar. "Design and Implementation of Storage Benchmark Kit". *Emerging Research in Computing, Information, Communication and Applications*. Ed. Nalini and N. Springer Singapore, 2022. 45–62.

Munegowda, Keshava and Sanjay Kumar N.V. "SLC: Sliding Latency Coverage Factors for Optimal Performance Benchmarking of Storage Systems". *2022 3rd International Conference for Emerging Technology (INCET)*. IEEE, 2022. 1–8.

Pirzadeh, Pouria, Michael Carey, and Till Westmann. "A performance study of big data analytics platforms". *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017. 2911–2920.

Seltzer, M, et al. "The case for application-specific benchmarking". *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*. IEEE Comput. Soc, 1999. 102–107.

Wu, H. "Research Proposal: Reliability Evaluation of the Apache Kafka Streaming System". *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2019. 112–113.

Embargo period: The article has no embargo period.

To cite this Article: Kumar N V, Sanjay, and Keshava Munegowda. "**SBK: A Framework for Performance Benchmarking for a Variety of Storage Systems** ." International Research Journal on Advanced Science Hub 05.07 July (2023): 222–231. http://dx.doi.org/10.47392/irjash.2023.047