# An Experimental Based Study to Evaluate the Efficiency among Stream Processing Tools

Akshay Mudgal
Faculty of Computer Applications,
Manav Rachna International Institute of Research and Studies, India
toakshaymudgal@gmail.com

Shaveta Bhatia
Faculty of Computer Applications,
Manav Rachna International Institute of Research and Studies, India
dydir.oe@mriu.edu.in

**Abstract:** *With the advancement in internet technology, augmentation in regular data generation has been amplified at a drastic level. Several different industries, for instance hospitality, defense, railways, health care, social media, education, etc., are creating and crafting different and several types of raw and processed data at a significant level, whereas, each of them has their own unique reason to shelter and call their data imperative and crucial. Such large and huge amount of data needs some space to get saved and secured, this is what Big Data is. A Data Stream Processing Technology (DSPT) is the significant mechanism and the mainstay for compiling and computing the large amount of data as well as the way to collect and process the raw data to call it information. There are varieties of DSPT like Apache Spark, Flink, Kafka, Storm, Samza, Hadoop, Atlas.ti, Cassandra, etc. This paper aims at comparing the five well- known and widely used open source big data DSPT (i.e., Apache Spark, Flink, Kafka, Storm, and Samza). An extensive comparison will be performed based on 12 different yet interconnected standards. A matrix has been designed through which five different experiments were executed, based on which the juxtaposition will be prepared. This paper summarizes an extensive study of open source big data DPST with a practical experimental approach in a well-controlled and sophisticated environment.*

**Keywords:** *Big data, data streaming, real time stream processing.*

*Received February 5, 2023; accepted May 23, 2023*
*https://doi.org/10.34028/iajit/20/6/11*

## 1. Introduction

Since 1991, the date when the internet was publicly available, the growth in data production has reached its extent [1, 2]. To process and analyze such a giant amount of data, Big Data Analytics has been introduced, which has become a significant and key method to analyze, scrutinize, examine, and produce such enormous raw data into valuable information [3]. The term "Big Data" exactly utilize to imprints the volume, variety, velocity, and veracity of data which is grim to process using traditional data processing mechanisms [4, 5].

Processing of data is a critical task [6]. There are several reasons that could lead to data loss and tampering, which leads to conferring about data security, which is alternatively a bulging factor that needs immense concern [7, 8]. In Big Data, data processing has been achieved by two major processing methods/frameworks, that is, native streaming and micro-batching [9]. In simple terms, native streaming is the data processing mechanism that processes the data as soon as it arrives without waiting for others [10]. However, in the micro-batching mechanism (another name Fast Batching), the incoming records are batched together and then processed as a diminutive batch with a delay of a few seconds [11]. The considered open-source Data Stream Processing Technology (DSPT) (i.e., Apache Spark, Flink, Kafka, Storm, Samza) somehow uses one of those data processing mechanisms in order to produce the relevant outcome [12].

Real-Time Distributed Stream Processing Models (RT-DSPM) can benefit traffic observing applications for digital protection and danger identification [13]. Current interruption discovery and avoidance frameworks are not compelling, on the grounds that 85% of dangers require a long time to be identified and as long as 123 hours for a response after identification to be performed. Advanced RT-DSPM for security basic applications is required and in the future with the progression of the world wide web [14]. To meet these necessities, processing systems have been projected to perform distributed processing. These open-source systems can characterize custom stream processing applications for explicit cases [15]. These general purpose platforms offer an Application Programming Interface (API), adaptation to internal failure, and versatility for stream processing [16, 17].

This paper works on the investigation of five open-source DSPT by performing practically applied experiments on the foundation of a matrix system developed on the basis of five exclusive parameters, which are:

1. Saturation level.
2. Scalability.
3. Processor/resources consumption.
4. Processing rate/throughput.

5. Fault tolerance.

In the end, an in-depth comparison comprises twelve parameters; a juxtaposition has been prepared in order to satisfy an individual in a one-shot view that which DSPT would be compatible and capable enough to be utilized in his data processing needs [18].

The major research question behind this study is to practically corroborate the most efficient open source DSPT, as now a day in alpha generation such tools are widely and extensively utilized in the Internet of Things (IoT), Payment gateways, servers etc.

This paper works on the investigation of five open-source DSPT by performing practically applied experiments on the foundation of a matrix system developed on the basis of five exclusive parameters, which are:

1. Saturation level.
2. Scalability.
3. Processor/resources consumption.
4. Processing rate/throughput.
5. Fault tolerance.

In the end, an in-depth comparison comprises twelve parameters; a juxtaposition has been prepared in order to satisfy an individual in a one-shot view that which DSPT would be compatible and capable enough to be utilized in his data processing needs [19].

## 2. Problem Statement and Purpose of Study

Due to the vast expansion in data parallel to innovation in modern technology, it is vital to perform critical analysis on such tools and techniques that help in processing the vast amount of data. An enormous number of articles have been published that explain and provide vide, eclectic, and comparative information for the major open-source data stream processing frameworks. Our research imprints the state-of-the-art experimental comparative study to identify each DSPT with its properties.

## 3. Literature Study

Assuncao *et al*. [4] presented the study which discusses advancements and enhancement in big data techniques for stream data processing in terms of performance, data structure utilized and information state. This paper features principal target distribution channels for continuous stream handling research progressively data warehousing to produce fundamental and huge information applications. Liu and Buyya [13] Furthermore, implementation problems alongside created apparatuses and assessment confirmations for constant stream handling in completely referenced application areas are also discussed. Mehmood and Anees [16] proposes a four-layer empowering foundation involving a data stream processing framework for IoT applications. In addition, the authors assessed the performance of the five generally known, and reasonable, data stream processing systems. Supposedly, these five pieces of middleware uphold every one of the fundamental highlights of IoT applications. On the side of contention, a correlation with other existing innovations has been introduced. Ounacer *et al*. [17] presented a rigorous study based on a scientific categorization that could work with the correlation of various presented stream handling systems. In light of this scientific classification, the author presented an outline of four open-source stream handling structures. Salem [19] embraces an understanding of the elements that must be thought while choosing a stage, given a particular use case. For instance, Flink is a decent decision in the event that mind boggling stream handling is required. Nonetheless, Spark Streaming is a fuller grown project and has a greater local area. Storm is likewise a developed undertaking and can give better inertness with fewer limitations, yet can't ensure state consistency. Zubarogglu and Atalay [27] imprints the relative outcomes around three stream handling structures, including: Apache Spark, Flink, and Storm. This study changes the Yahoo streaming benchmark to make it work in a multi-hub climate and gives results about the immersion level of every system. The immersion level is basically the most extreme streaming burden that the structures could process immediately toward the completion of the interaction. Likewise, a few discoveries for tuning every one of the systems for the ideal presentation are introduced. Finally, the asset utilization and adaptability of the systems are discussed. Soumaya *et al*. [20] Amine introduce a cutting edge concerning various ideas, which prompted leading a careful correlation of information about stream handling systems. The principal objective behind this assessment is to show that enormous information design depends on batch handling, which can't handle information progressively. Storm was chosen as an instrument for information handling (as a result of this careful correlation was recommended because open source allows for constant handling with exceptionally low inactivity). Grebovic *et al*. [6] one more connexion for constant handling structures was likewise led to recommend another production wherein Artificial Intelligence (AI) and Machine Learning (ML) were utilized to work with the handling in real time.

## 4. Proposed Methodology

The experiment has been conducted on the following open source big Data Stream Processing Tools (DSPT) on the basis of 12 parameters defined in step 1. The descriptive evaluation would be on the basis of twelve different yet interconnected parameters discussed above. Subsequently, in this work, the experimental method has been utilized in order to conduct the experiments and tests, to compare the results, and to know the

performance and accuracy of the Apache Spark, Flink, Kafka, Storm and Samza as shown in Figure 1. The whole scenario has five different ways of experiments. The experiments are performed in an environment with a total of eleven different working machines that includes one master and ten slaves/nodes each has minimum configuration i.e., 12 GB of RAM and intel core i5-3230m at 2.60ghz. The latest version of Linux Ubuntu operating system and Java has been utilized connected with a 2Gbps Ethernet. This summarizes the methodology of experiments and testing of the data streaming technologies in a well-controlled and sophisticated environment. The datasets are freely download from GitHub and UNSW-NB 15. This dataset has several types of attacks and classes. The conducted research is produced on the analysis of data stream models hence such datasets has been chosen which can stress-out the systems more efficiently.

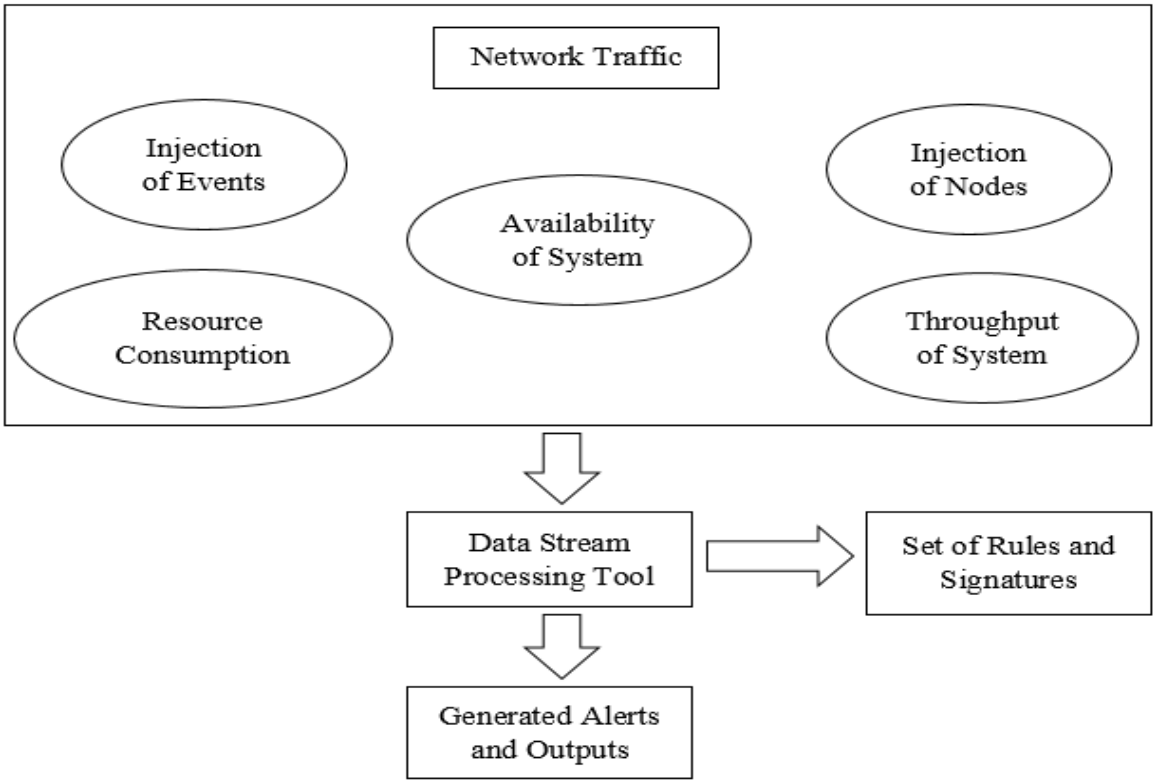


Figure 1. Overview of methodology framework.

## 5. Experiments and Results

- *Experiment* 1: Injection of the maximum number of events/packets that the system could process per second without causing any delay or deferral, which signifies the level of saturation of a system [18, 24]. This experiment represents the high cluster results of the declared and professed DSPT's. The scenario of the experiment consists of ten machines, each of which is utilized in one portion (i.e., five nodes). Whereas the master machine circulates the predefined clusters/packets/events to all the DSPT's at the same time, which are installed on apiece machine individually. Because the configuration of all the nodes is set as discussed (i.e., minimum), and the DSPT's configuration settings are kept on default, the produced output imprints the meticulous results.

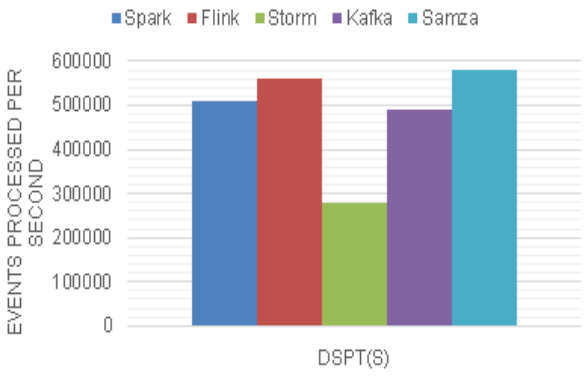


Figure 2. Graph representing processed events.

The presented graph (Figure 2) describes the high-flow projection of events to analyze the data handling and processing capabilities of a DSPT(s).

- *Results*: While the events were injected, the halts imparted by the data stream processing model are shown in the form of a graph. It can be seen that the best results are presented by Apache Flink.

Continuing on, Spark and Samza has also shown their efficiency. This is a motley outcome, as while injecting the events with no parallel load, the halts were more or less equal to zero, whereas with the increment of parallel load, the halts were increased. So, to present firm results, the executed experiment is an amalgamation of 0 parallel load, >2!>4 parallel load, and >5!>7 parallel load.

- *Experiment* 2: Injection of nodes simultaneously to identify the scalability of a system [21]. Through this experiment, the performance is judged, which would vary while changing the default configuration at the performance stage. The experimental environment remains the same as mentioned in experiment no. 1. This experiment represents the scalability of the system. This experiment corroborates the competency of a system which is analyzed by fluctuating its clusters by adding and removing nodes.

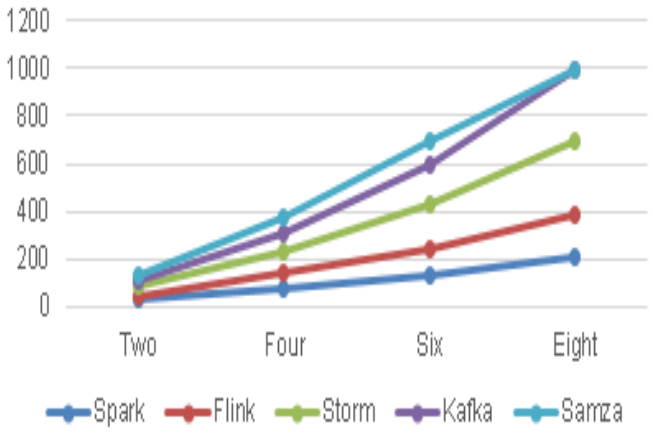


Figure 3. Identifying scalability of systems.

The presented line chart (Figure 3) describes the projection of events with varying cluster sizes to analyze the range of compatibility of a DSPT.

- *Result*: This study identifies the increase in the number of events processed per second against a verified number of cluster sizes. There is a vast, comparable variation in the systems. Significantly, the highest scale-up ratio is obtained by Storm, whereas the least one is scored by Samza.
- *Experiment* 3: Consumptions of Central Processing Unit (CPU) and other fertile resources in a machine is a vital aspect to be tartan in a system [20]. This experiment signifies the consumption of resources (like CPU, memory, and network) whilst the DSPT's are kept on to achieve optimal performance as shown in Table 1. The testing environment is established with a four test matrix approach, whereby the beginning, i.e., 0, is the base matrix and 6 is the highest one. After each test, an addition of two nodes would be made to enhance the work load [21, 22]. The DSPT's are set at their best-default configuration to sustain impartiality and promote fair test outcomes.

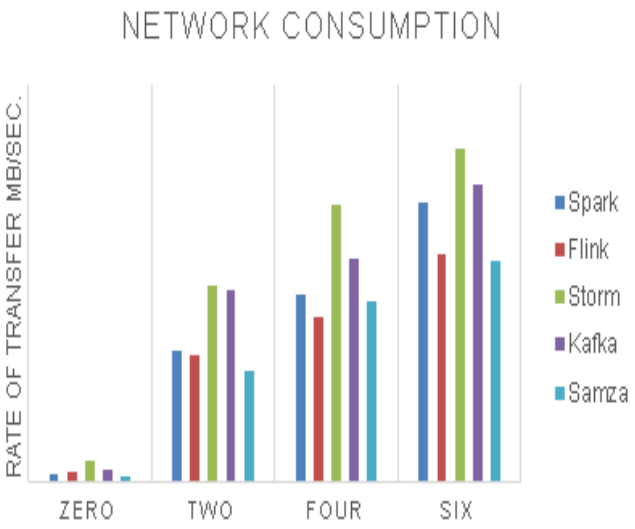


Figure 4. Determining network resources consumption.

The data transfer while the systems were kept at their optimal conformation is measured in Megabytes per second.

Table 1. CPU Usage at different packet rates.

| Traffic | CPU | | | | |
|---|---|---|---|---|---|
| | Spark | Flink | Storm | Kafka | Samza |
| 1000 | 16 | 12 | 28 | 26 | 25 |
| 1400 | 19 | 16~17 | 31 | 29 | 29~31 |
| 1800 | 23~25 | 20 | 36~39 | 34 | 36 |
| 2200 | 28 | 26 | 43 | 38 | 36~38 |
| 2600 | 31 | 28~30 | 49 | 42 | 44 |
| 3000 | 36 | 32-37 | 52~55 | 47~51 | 49 |
| 3400 | 40 | 39~42 | 63 | 55 | 52~56 |
| 3800 | 45~49 | 44 | 66~68 | 59~62 | 60 |
| 4200 | 54 | 49~52 | 71 | 68 | 64 |
| 4600 | 59~61 | 58 | 76 | 71~74 | 71 |
| 5000 | 74+ | 64+ | 84+ | 75~80+ | 80+ |

Analysing the CPU consumption is projected in percentage, whilst the systems were kept at their optimal configuration as demonstrated in Table 1.

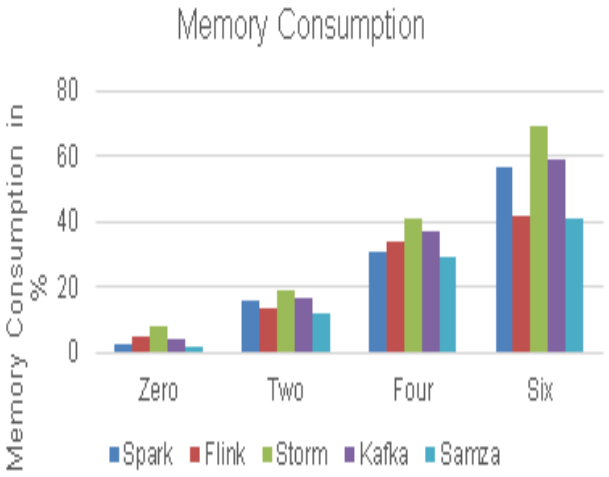


Figure 5. Determining memory consumption.

Investigating the memory consumption in percentage, whilst the systems were kept at their optimal configuration.

- *Result*: The experiment no. 2 shepherded entails of three graphs, concluding with the best-in- class recital projected by Samza, Flink, and sustained by Spark, Kafka, and Storm. The tests were carried out using applied-practical approach, where each DSPT was installed on a single node machine with the default configuration. Cumulatively in Figure 4, 5, and 6 the test results signify the Network consumption, CPU consumption, and Memory consumption where a single node machine was utilized to operate with 0, 2, 4, 6 work load respectively. The network consumption plays a vital role to analyses a systems performance, the Figure 4 signifies a rational and lucid intake of network bandwidth by Samza, relatively CPU and memory consumption also dictates a system decency. The whole experimental scenario was occupied by the state-of-the-art data streaming model of apache.

- *Experiment* 4: The availability of a system is a significant and essential factor [23]. The stream processing system should have the capability to recover rapidly from any hazardous and unwanted failure without effecting the overall fiasco [24]. This experiment indicates and handles the fault tolerance factor, allowing us to investigate the capability of DSPT's as displayed in Table 2. There could be a large number of reasons leading to failure, such as network failure, node failure, software catastrophe, etc. The paper consists of five different types of systems, where one uses micro-batch processing and the other uses stream data processing to practice the data. This experiment uses a message broker subscription service of Apache Kafka (latest version) [25]. These messages will be injected to analyze the fault tolerance capability of DSPT's.

Table 2. Categorized packet loss at different traffic rate.

| Traffic Rate | TCP | | | | | UDP | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Spark | Flink | Storm | Kafka | Samza | Spark | Flink | Storm | Kafka | Samza |
| 1000 | 4 | <2 | 9 | 6 | 5 | 4 | 2 | 12 | 7 | 4 |
| 1400 | 5-7 | 3 | 12 | 11-13 | 9 | 7 | 4 | 15 | 14 | 12 |
| 1800 | 12 | 9 | 17 | 14 | <9 | 9 | 6 | 17 | 11 | <15 |
| 2200 | 11-14 | 12 | 21 | <24 | <19!<14 | 11 | 9 | 22-25 | 16 | 19 |
| 2600 | 18 | 16 | 35 | 31 | 29 | 14 | 15 | 26 | 18 | 18 |
| 3000 | 21 | 18-20 | 41 | 36 | 31-33 | 16 | 18 | 29-31 | 21 | 23 |
| 3400 | 33 | 25 | 52-56 | 39-41 | 39 | 19-22 | 21 | 39-41 | 25 | 28 |
| 3800 | 38 | 29-33 | 59-61 | 49 | 44 | 24 | 22-24 | 44 | 29-31 | 30 |
| 4200 | 43 | 37 | 65 | 52 | 48-50 | 26 | 28 | 52 | 37 | 39 |
| 4600 | 44-46 | 41 | 68-71 | 55 | 53 | 28 | 33 | 66 | 39-41 | 44 |
| 5000 | 49 | 44 | 74 | 58 | 52 | 31 | 36-38 | 72 | 49 | 48-51 |

The above graph (Figure 6) represents the calculated percentage of data loss happened while transmitting the events/message stream into the system.

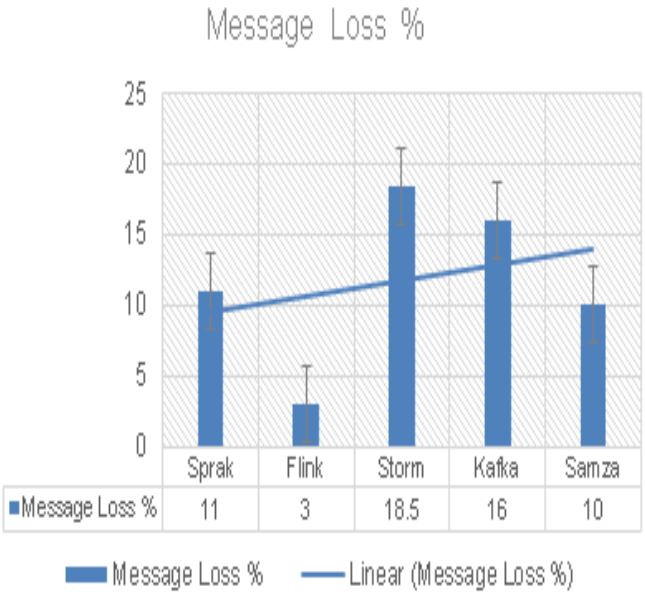


Figure 6. Representation of transmitted data loss.

- *Result*: The experiment shows the system behavior when the transmission of events is projected and a predefined fault occurs, that is, fluctuation in network bandwidth and node failure occur. The presented Figure 7 signifies the minimum loss of data handled by the system. We obtained the result with a 98% confidence interval.

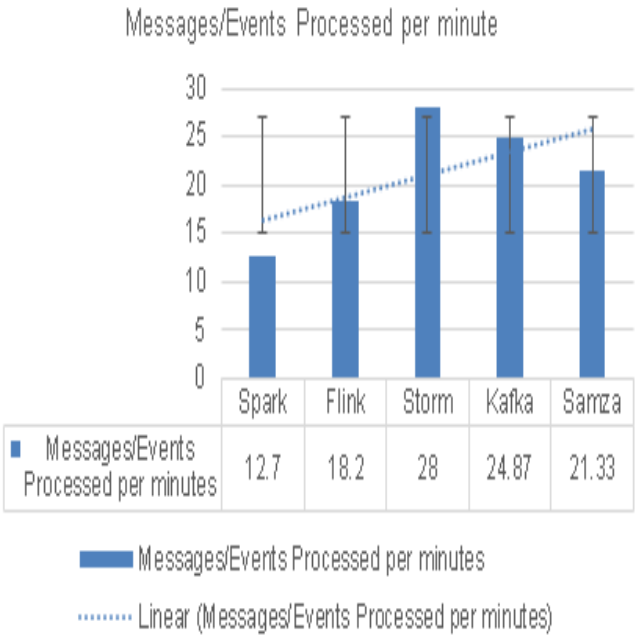


Figure 7. Throughput result when messages injected per minute.

- *Experiment* 5: The performance of the system in terms of recital throughput is another determining

factor [27]. This experiment was conducted with the same data set applied in experiment number 4. The scenario of the experiment is as follows: the data sets are injected into the system in totality and will replicate the same as and when necessary [26]. The replication of data sets plays a vivacious role as the throughput can be judged more accurately when the processing rate and delivery rate are calculated with the utmost stress and load [7, 9].

This experiment will be performed on five different yet LAN connected machines with the discussed basic configurations.

- *Result*: The executed experiment declares the significant outcomes, which helps to understand the data stream processing systems' behavior and processing clout under various circumstances. Figure 8 demonstrates the processing power of a system with and under normal circumstances, whereas when the working environment significantly vicissitudes, a vast change in data processing occurs. In both situations, the performance of Apache Flink has its own denotations and dominance. As discussed above, the genuine performance of any system resides in its patience level. The same methodology has been applied here by adding two stress factors, i.e., replicating the events and fluctuating the systems by turning off or restarting or by affecting the bandwidth of the network.

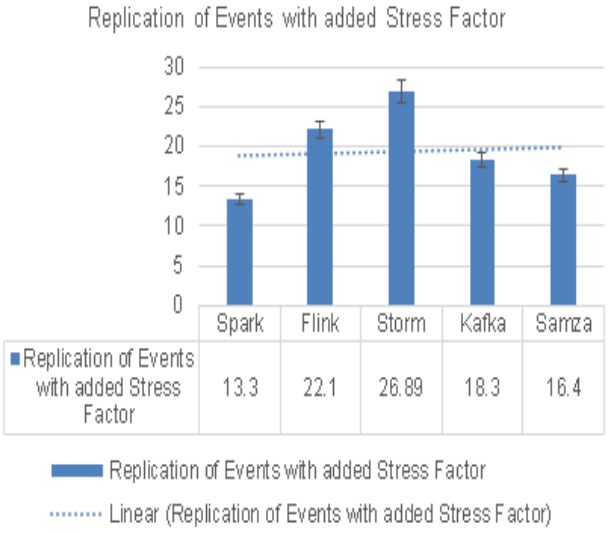


Figure 8. Throughput result with replication messages and added stress load.

Table 3. Demonstrating parameterized comparison among DSPT's after analysis.

| | | Open-source DSPT | | | | |
|---|---|---|---|---|---|---|
| **Sr. No.** | **Parameters** | **Spark** | **Flink** | **Storm** | **Kafka** | **Samza** |
| 01 | Architecture | Master-Slave | Kappa [Four layered architecture] | Kappa [Four layered architecture] | Cluster | Compatible [Replicate every change] |
| 02 | Processing | Micro-Batch | Hybrid | Hybrid | Streaming | High |
| 03 | Fault Tolerance | Compatible | Compatible [Receives programs] | Compatible [Receives programs] | Compatible [Replica] | Fourth |
| 04 | Latency | <1 second | High | High | Good | Compatible |
| 05 | Throughput | Fifth | Second | Second | Third | 1024 MB |
| 06 | Storage | Compatible | Compatible | Compatible | Compatible | Java and Scala |
| 07 | Inbuilt memory | 300 MB | 1000 MB | 1000 MB | Depends upon system configuration | Compatible |
| 08 | API programming | Java, Python and Scala | Java and Scala | Java and Scala | Java andScala | Highly |
| 09 | Data mobility | Compatible | Compatible | Compatible | Compatible | True |
| 10 | Flexibility | Good | Fair | Fair | Less | Open |
| 11 | Platform compatibility | True | True | True | True | Compatible [Replicate every change] |
| 12 | Source model | Open | Open | Open | Open | High |

## 6. Research Findings and Closure Discussion

The conducted experiments explicitly imprints comparison of several different data stream processing systems out of which each has its own capability and limitation. The conducted experiments prove and display the working and performing scenarios of the open source data stream processing systems that has been demonstrated in Table 3 as a whole. The projected results and outcomes are produced at a 98% confidence interval. When the data events were projected with replication, the performance objects modelled by different frameworks explicitly display a variation of processing in comparison to when the replication was not performed. Being Storm, the most mature data streaming tool, produces the best in class outcome, as shown in Figures 7 and 8. The resource consumption is another critical aspect, which needs crucial undertaking. The executed test no. 3 imprints this facet by demonstrating three separate graphs, to display the network, memory, and CPU core consumption in percentage. The result displays Flink, most suitable for resource utility. The inclusive experimentation methodology dictates an overall appropriate performance by Samza. However, when maturity and market is defined and debated, the most suitable open source data streaming framework anticipated would be Apache Spark and Apache Flink.

## 7. Contribution towards Society

This research presents an open source data streaming system with an experimental approach to signifies the best operable and suitable data processing tool for

several platforms (like ML, Intrusion Detection System (IDS), Honeypot, etc.,) to impart and improve the processing speed. The conducted experiments signify Apache Flink and Spark the most advanced and suitable tools to satisfy today's generation need of data processing.

## 8. Conclusions

There are many existing and arising applications that demand ongoing handling of high-volume heterogeneous information streams. There are also many open-source and exclusive frameworks for information stream handling. Removing significant and opportune bits of knowledge from unbounded information is extremely difficult. The enormous number of accessible frameworks is great but represents a significant test as far as choosing the right parts or handling systems for various use cases. Understanding the necessary capacities of stream structures is fundamental in settling on the right plan or utilized on decision. Information that is accumulated progressively can turn out to be excessively important at the time it shows up and upholds significant navigation. The discussed frameworks arose to empower dispersed handling of surges of huge information. Components utilized by unmistakable systems to confront the difficulties presented by stream handling with regard to huge amounts of information were discussed in this work. We likewise depicted a scientific categorization that could work with the examination of various highlights presented by stream handling systems. In light of this scientific classification, we conducted descriptive justified experiments of five open-source stream handling systems. Our review gives an understanding of the elements that must be thought about while choosing a platform. This paper reports our commitments towards moderating these difficulties. We present an experiment based writing review and an investigation of the DSPT.

## 9. Future Direction

The future direction considers the results of this study (data streaming technology) to be utilized in order to improve the efficiency of data processing efficiency of online video conferencing portals.

## References

[1] Bahri M., Bifet A., Gama J., Gomes H., and Maniu S., "Data Stream Analysis: Foundations, Major Tasks and Tools," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 3, pp. 1-17, 2021. DOI:10.1002/widm.1405

[2] Carbone P., Gévay G., Hermann G., Katsifodimos A., Soto J., Markl V., and Haridi S., *Handbook of Big Data Technologies*, Springer, 2017. https://doi.org/10.1007/978-3-319-49340-4_7

[3] Cardellini V., Lo Presti F., Nardelli M., and Russ G., "Run-Time Adaptation of Data Stream Processing Systems: The State of the Art," *ACM Computing Surveys*, vol. 54, no. 11s, pp. 1-36, 2022. https://doi.org/10.1145/3514496

[4] De Assuncao M., Da Silva Veith A., and Buyya R., "Distributed Data Stream Processing and Edge Computing: A Survey on Resource Elasticity and Future Directions," *Journal of Network and Computer Applications*, vol. 103, pp.1-17, 2018. https://doi.org/10.1016/j.jnca.2017.12.001

[5] Fragkoulis M., Carbone P., Kalavri V., and Katsifodimos A., "A Survey on the Evolution of Stream Processing Systems," *arXiv Preprint, arXiv:2008.00842v2*, 2023. https://arxiv.org/pdf/2008.00842.pdf

[6] Grebovic M., Filipovic L., Katnic I., Vukotic M., Popovic T., "Machine Learning Models for Statistical Analysis," *The International Arab Journal of Information Technology*, vol. 20, no. 3A, pp. 505-514, 2023. DOI: 10.34028/iajit/20/3A/8.

[7] Hesse G. and Lorenz M., "Conceptual Survey on Data Stream Processing Systems," *in Proceedings of the IEEE 21st International Conference on Parallel and Distributed Systems*, Melbourne, pp. 797-802, 2015. DOI:10.1109/ICPADS.2015.106

[8] Hirzel M., Soulé R., Schneider S., Gedik B., and Grimm R., "A Catalog of Stream Processing Optimizations," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1-34, 2014. https://doi.org/10.1145/2528412

[9] Isah H., Abughofa T., Mahfuz S., Ajerla D., Zulkernine F., and Khan S., "A Survey of Distributed Data Stream Processing Frameworks," *IEEE Access*, vol. 7, pp. 154300-154316, 2019. DOI: 10.1109/ACCESS.2019.2946884.

[10] Javed M., Lu X., and Panda D., "Characterization of Big Data Stream Processing Pipeline: A Case Study Using Flink and Kafka," *in Proceedings of the 4th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, Texas, pp. 1-10, 2017. https://doi.org/10.1145/3148055.3148068

[11] Kamburugamuve S. and Fox G., "Survey of Distributed Stream Processing," Technical Report, Bloomington: Indiana University, 2016. DOI:10.13140/RG.2.1.3856.2968

[12] Karakaya Z., Yazici A., and Alayyoub M., "A Comparison of Stream Processing Frameworks," *in Proceedings of the International Conference on Computer and* Applications, Doha, pp. 1-12, 2017. DOI:10.1109/COMAPP.2017.8079733

[13] Liu X. and Buyya R., "Resource Management and Scheduling in Distributed Stream Processing Systems: A Taxonomy, Review, and Future Directions," *ACM Computing Surveys*, vol. 53, no.

3, pp. 1-41, 2020. https://doi.org/10.1145/3355399

[14] Lobato A., Lopez M., Cardenas A., Duarte O., and Pujolle G., "A Fast and Accurate Threat Detection and Prevention Architecture Using Stream Processing," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 3, pp. 1-17, 2022. DOI: 10.1002/cpe.6561

[15] Lopez M., Lobato A., and Duarte O., "A Performance Comparison of Open-Source Stream Processing Platforms," *in Proceedings of the IEEE Global Communications Conference* (GLOBECOM), Washington (DC), pp. 1-6, 2016. DOI:10.1109/GLOCOM.2016.7841533

[16] Mehmood E. and Anees T., "Challenges and Solutions for Processing Real-Time Big Data Stream: A Systematic Literature Review," *IEEE Access*, vol. 8, pp. 119123-119143, 2020. DOI: 10.1109/ACCESS.2020.3005268

[17] Vikash., Mishra L., and Varma S., "Performance Evaluation of Real-Time Stream Processing Systems for Internet of Things Applications," *Future Generation Computer Systems*, vol. 113, pp. 207-217, 2020. https://doi.org/10.1016/j.future.2020.07.012

[18] Ounacer S., Talhaoui M., Ardchir S., Daif A., and Azouazi M., "A New Architecture for Real Time Data Stream Processing," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, 2017. DOI:10.14569/IJACSA.2017.081106

[19] Ramírez-Gallego S., Krawczyk B., García S., Woźniak M., and Herrera F., "A Survey on Data Preprocessing for Data Stream Mining: Current Status and Future Directions," *Neurocomputing*, vol. 239, pp. 39-57, 2017. https://doi.org/10.1016/j.neucom.2017.01.078

[20] Salem F., Comparative Analysis of Big Data Stream Processing Systems, Master's Thesis, Aalto University, 2016. https://aaltodoc.aalto.fi/handle/123456789/21577

[21] Soumaya O., Amine T., Soufiane A., Abderrahmane D., and Mohamed A., "Real-Time Data Stream Processing Challenges and Perspectives," *International Journal of Computer Science Issues*, vol. 14, no. 5, pp. 6-12, 2017. https://doi.org/10.20943/01201705.612

[22] Tantalaki N., Souravlas S., and Roumeliotis M., "A Review on Big Data Real-Time Stream Processing and its Scheduling Techniques," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 35, no. 5, pp. 571-601, 2020. DOI:10.1080/17445760.2019.1585848

[23] Vakilinia S., Zhang X., and Qiu D., "December. Analysis and Optimization of Big-Data Stream Processing," *in Proceedings of the IEEE Global Communications Conference* (GLOBECOM), Washington (DC), pp. 1-6, 2016. DOI:10.1109/GLOCOM.2016.7841598

[24] Zhang S., He B., Dahlmeier D., Zhou A., and Heinze T., "Revisiting the Design of Data Stream Processing Systems on Multi-Core Processors," *in Proceedings of the IEEE 33rd International Conference on Data Engineering*, San Diego, pp. 659-670, 2017. DOI:10.1109/ICDE.2017.119

[25] Zhang S., Zhang F., Wu Y., He B., and Johns P., "Hardware-Conscious Stream Processing: A Survey," *ACM SIGMOD Record*, vol. 48, no. 4, pp. 18-29, 2020. https://doi.org/10.1145/3385658.3385662

[26] Zhao X., Garg S., Queiroz C., and Buyya R., "A Taxonomy and Survey of Stream Processing Systems," *Software Architecture for Big Data and the Cloud*, pp. 183-206. 2017. https://doi.org/10.1016/B978-0-12-805467-3.00011-9

[27] Zubaroğlu A. and Atalay V., "Data Stream Clustering: A Review," *Artificial Intelligence Review*, vol. 54, no. 2, pp. 1201-1236, 2021. https://doi.org/10.1007/s10462-020-09874-x

**Akshay Mudgal** is a Research Scholar at Faculty of Computer Applications, Manav Rachna International Institute of Research and Studies, Faridabad, India. He has done Master's in Computer Applications, Master's in Business Administration (Information System Management), CCNA (Network Security), and DOEACC O'. His prioritized area of research includes Network Security, Information Security, Big Data, and Blockchain.

**Shaveta Bhatia** has been awarded her Ph.D. degree in Computer Applications and she has completed her Master in Computer Applications (MCA) from Kurukshetra University. She has 17 years of academic and research experience and is a member of various professional bodies like ACM, IAENG and CSI. Her specialized domains include Mobile Computing, Web Applications, Data Mining and Software Engineering and guiding research scholars in these areas.