International Journal of Computer Science and Engineering (IJCSE)
ISSN (P): 2278–9960; ISSN (E): 2278–9979
Vol. 11, Issue 1, Jan–Jun 2022; 285–296

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

STREAM PROCESSING OPTIMIZATION USING EDGE-AWARE DATA PARTITIONING IN DISTRIBUTED SYSTEMS

Sarvesh Kumar Gupta

Consulting Member of Technical Staff, Oracle, Saint Peters, Missouri -63376, USA

ABSTRACT

Today's distributed stream processing platforms are becoming more and more vulnerable in their ability to handle large data streams being generated by IoT devices, edge sensors, industrial surveillance devices, smart cities, and various cloud applications. Conventional data partitioning methods such as hashing-based and round-robin partitioning may have trouble maintaining low latencies and evenly distributing the load in cases where data streams exhibit geographical scattering, workload imbalances, and nonuniform availability of resources. The focus of the research discussed herein is the development of optimal data partitioning approaches which will improve stream processing performance through intelligent data partitioning considering edge proximity and resource heterogeneity. Specifically, this paper will examine edge-aware partitioning as a potential solution to the challenges posed by traditional partitioning approaches. The study uses a simulation-based experimental approach implemented in Apache Flink to compare hash-based, range-based, load-aware, and proposed edge-aware partitioning strategies.The strategies are evaluated using latency, throughput, and network utilization as performance metrics. It becomes apparent from the results that the use of edge-aware data partitioning considerably improves stream processing due to reduced network communication overhead, minimized latencies, effective load balancing, and resource optimization.

KEYWORDS: *Stream Processing, Edge Computing, Edge-Aware Data Partitioning, Distributed Systems, Real-Time Data Analytics, Load Balancing, Data Locality.*

Article History

Received: *28 Apr 2022* | ***Revised:*** *30 Apr 2022* | ***Accepted:*** *30 Jun 2022*

INTRODUCTION

The emergence of new developments within the field of digital technologies, namely IoT devices, mobile apps, smart cities, automation processes, and services provided via the Internet, has led to a massive increase in data streams. Modern applications produce a huge volume of data that needs to be processed immediately for making decisions promptly. In comparison with batch-processing systems designed to handle stored data on a regular basis, distributed stream processing systems are intended to operate with the input data in real time. Thus, distributed stream processing platforms Apache Storm, Apache Flink, Apache Spark Streaming, and Kafka Streams have become very popular in recent years because they allow addressing large-scale real-time data.

With the growth of demand for real-time analytics, modern distributed stream processing systems face a range of challenges that include issues of scaling, latency, high throughput, fault tolerance, and resource management. Real-time data streams generated by millions of sensors located in various geographical locations often demonstrate such dynamism

as the changing rate of data arrivals, workload skews, bursts, and uneven data distribution. To increase the efficiency of stream processing tasks, in distributed environments such tasks are divided into pieces assigned to different computing nodes; however, inadequate partitioning of data streams may result in workload imbalances when certain nodes become overloaded while others are not utilized at all. This negatively impacts on processing speed and throughput.

To deal with workload issues, the following partitioning strategies are used today: hash-based partitioning, round-robin distribution, and key-based partitioning. These techniques are fairly simple and convenient to apply in homogenous cloud environments; nevertheless, they cannot be applied directly in distributed edge-computing environments due to their specific characteristics. The concept of edge computing implies that computing power is brought closer to data sources. Although this improves processing speed, it creates other difficulties: limited computing capacity, heterogeneity of hardware and networking conditions, as well as geographical location of computation nodes.Distributed stream processing platforms such as Apache Flink and Apache Storm have become widely adopted for real-time analytics in large-scale environments [3], [6].

Thus, the need for edge-aware partitioning arises since this approach considers data locality, network delay, workload, availability of resources, and communication costs when allocating data. Edge-aware partitioning allows improving load balancing, decreasing communication costs, and minimizing processing latency by analyzing the workload dynamically, as well as using edge computing facilities together with cloud-based resources. The development of distributed stream processing is moving toward cloud–edge collaboration; therefore, edge-aware partitioning has become one of the most important methods for optimizing distributed stream processing.Edge computing reduces communication latency by moving computation closer to data sources [1], [9].

LITERATURE REVIEW

The concept of stream processing has emerged as the most essential computation paradigm in various applications where the continuous processing of high-throughput streaming data is required, including IoT surveillance systems, smart city systems, industrial automation, fraud detection systems, traffic monitoring, and real-time recommendation engines. While traditional batch processing systems process large datasets once they arrive, the stream processing systems process data continuously as it comes in, which raises issues such as latency, throughput, load balancing, fault tolerance, and resource utilization. In distributed stream processing systems, it is vital to have proper stream data partitioning because tuples coming to the system must be appropriately partitioned and distributed across several parallel operators and nodes.

One of the early investigations of distributed stream processing focused on parallelism and scalability in a cluster-based setting. StreamCloud, an elastic and scalable system for processing stream data, proposed by Gulisano et al. (2012), demonstrated that throughput can be increased by partitioning operators across multiple machines. Nonetheless, most of the existing clustering techniques for stream partitioning assume relatively stable network conditions and homogenous processing infrastructure. In contrast, cloud-, fog-, and edge-based stream partitioning solutions need to account for network conditions, heterogeneity, and varying loads.

Nasir et al. (2015) offered an important breakthrough in the realm of load-balancing techniques by introducing Partial Key Grouping (PKG). Traditional key grouping preserves per-key state consistency but suffers from poor performance in case of skew because it may result in hot keys overloading some particular worker processes. The PKG technique utilizes the power of two choices concept: each key will be assigned to two candidate workers, and the less

loaded worker will process the key. It helps to considerably improve the quality of load balancing while preserving most of the key advantages of key-based partitioning. The paper is important for edge-aware partitioning because the problem of skew is quite common for IoT edge-generated streams.

Katsipoulakis et al. (2017) continued discussing the topic of distributed stream partitioning by highlighting its cost dimensions. They argue that besides the classic metric of load balancing, a good partitioning scheme needs to take into consideration such factors as communication cost, state management, synchronization overhead, and downstream aggregation cost. All the mentioned aspects are especially critical in an edge context because of the limited bandwidth and high latencies in such environments.

Pacaci and Özsu (2018) presented the concept of distribution-aware stream partitioning as a novel alternative to uniform data partitioning in distributed stream processing systems. Distribution awareness in partitioning allows taking into account input data distribution and dynamic nature of stream data workloads. In many cases, stream data originating from edge environments is unevenly distributed because they come from different geographical regions. Therefore, distribution-awareness can be especially useful in edge partitioning.

The problem of operator placement in a distributed stream environment has attracted considerable attention as well. Specifically, the concept of network-aware operator placement, proposed by Pietzuch et al. (2006), was one of the first to acknowledge the necessity of placing stream processing operators in a way that minimizes the communication cost and takes into account network topology. It is important to mention this piece of work because the current edge placement is similar in many respects.

Due to the rising popularity of edge computing, the problem of operator placement received new attention. For instance, Sajjad et al. (2016) proposed SpanEdge, a framework intended for unified processing of stream data in central data centers and near-edge environments. It showed that the proximity of processing and storage reduces latency and required bandwidth. Such a finding is in line with the basic idea of edge-aware stream processing: data doesn't necessarily need to go to a cloud because partitioning and processing can be conducted based on local and proximal resources.

In a similar spirit, De Assunção, Veith, and Buyya (2018) provide a review of distributed stream processing and edge computing. They emphasize that the problem of resource elasticity is becoming increasingly difficult due to geographic distribution, heterogeneity, and latency sensitivity. As they noted, stream processing systems of the future should incorporate intelligent scheduling and resource-aware decision-making. Thus, edge-aware partitioning serves as a solution to this challenge.

Moreover, Hiessl et al. (2019) investigated the problem of optimal operator placement in fog environments. The main objective of their study was to determine how operators should be distributed among fog and cloud infrastructures to achieve the best results in terms of reduced response time and minimum cost. This work is highly related to edge-aware stream partitioning since the problem of operator placement and stream partitioning are highly interconnected. For instance, partitioning by geographic location will yield the best results if the operators are also deployed close to such regions. Hence, the problem of optimization should involve both aspects.

Finally, the concept of latency-aware stream processing placement on edge computing infrastructure was explored by Veith, de Assunção, and Lefèvre (2018). They conclude that decisions about operator placement should consider topology and latency, and that resource constraints need to be taken into consideration as well. The authors highlight that,

like in other types of optimization, stream partitioning in an edge environment should also match the physical properties of the infrastructure in terms of available bandwidth, node capacity, etc.

There were other developments in self-regulating stream processing frameworks as well. Floratou et al. (2017) proposed a self-regulating system Dhalion for the popular distributed stream engine called Twitter Heron. The idea behind it consists in automatic detection of performance issues, identification of the cause, and correction. It is not specifically an edge-aware partitioning technique, yet it is interesting for developing adaptive approaches to edge-aware stream processing. For instance, the system can be used to detect imbalances in partitioning and overloaded edge nodes, hot spots, and network congestion.

RESEARCH METHODOLOGY

In this study, the efficiency of edge-aware data partitioning was evaluated as a strategy for improving the performance of distributed stream processing systems. An experimental comparative approach was used to examine how different partitioning techniques affect latency, throughput, and network utilization under simulated edge–cloud stream processing conditions.

The main research questions focused on identifying the limitations of traditional stream partitioning methods, developing an edge-aware partitioning strategy, evaluating its performance, and comparing it with hash-based, range-based, and load-aware partitioning approaches. The study aimed to determine whether data locality, network proximity, current workload, and resource availability could improve stream processing performance in distributed edge–cloud environments.

A simulated distributed stream processing test environment was created using Apache Flink. The environment consisted of 10 edge nodes and 4 cloud nodes connected through an edge–cloud hybrid network. A simulated IoT data stream containing 300,000,000 events was generated over a 60-minute processing duration. Each event had an average size of 1 KB. The workload was designed to represent geographically distributed IoT data sources with varying arrival rates and workload intensity.

Four partitioning strategies were evaluated under the same workload conditions: hash-based partitioning, range-based partitioning, load-aware partitioning, and the proposed edge-aware dynamic partitioning strategy. The proposed strategy assigned incoming events to processing nodes based on data locality, available node capacity, current node load, and network latency. When local edge nodes exceeded the defined load threshold, events were redirected to alternative edge nodes or cloud nodes.

The effectiveness of each partitioning strategy was evaluated using three main performance metrics: average processing latency, throughput, and network utilization. Latency was measured as the average time required to process an event from arrival to completion. Throughput was measured as the number of processed events per second. Network utilization was measured as the total volume of data transferred between edge and cloud layers during the experiment.

All partitioning strategies were executed under identical simulated workload conditions to ensure fair comparison. The collected results were analyzed using comparative performance evaluation to determine whether the proposed edge-aware partitioning strategy improved processing efficiency, reduced communication overhead, and enhanced overall system responsiveness.

Table 1: Experimental Environment

Parameter	Value
Number of Edge Nodes	10
Number of Cloud Nodes	4
Stream Volume	300,000,000 Events
Framework Used	Apache Flink
Average Event Size	1 KB
Processing Duration	60 Minutes
Network Type	Edge–Cloud Hybrid Network
Partitioning Strategy	Edge-Aware Dynamic Partitioning
Baseline Comparison	Hash-Based, Range-Based, Load-Aware Partitioning
Evaluation Metrics	Latency, Throughput, Network Utilization

STREAM PROCESSING AND EDGE COMPUTING ARCHITECTURE

Modern stream processing solutions have been designed to analyze continuous flow of data coming from sensors, IoT devices, smartphones, manufacturing machinery, social media sites, and enterprise applications. Unlike batch processing systems, stream processing solutions operate on continuously arriving data, which helps in making timely decisions based on evolving situations. Distributed stream processing systems like Apache Flink, Apache Storm, and Apache Spark Streaming break down streams of data into manageable partitions to run them in parallel on distributed computing nodes.

Data processing using distributed stream processing consists of various stages. Firstly, data is produced by edge devices and sent to computing nodes. Then, the stream is divided into partitions and distributed to operators for filtering, aggregation, transformations, and analytics. Finally, processed output is delivered to other applications, stored to cloud databases, displayed on dashboards, or used in decision-making processes. Proper distribution of workload is crucial for achieving balanced resource usage and reducing processing delay.

Furthermore, edge computing brings computational power near the source of the stream. Instead of sending raw data to cloud servers, initial processing is done on local edge nodes, where data is filtered and aggregated. This process saves precious bandwidth and lowers latency, while providing timely results. It becomes important when dealing with applications like autonomous vehicles, automated industrial facilities, health monitoring, and smart city infrastructure.

However, there are a number of data distribution problems associated with distributed edge-cloud architecture. The stream usually consists of data blocks with a varying degree of processing complexity. This causes workload imbalances between different nodes and leads to the emergence of hotspots, which are nodes receiving more processing requests. Besides, data streams from geographically distributed devices arrive in irregular intervals and create workload variations. Another issue is network latency, which affects the choice of partitioning methods, especially for resource-heterogeneous environments and unreliable nodes. Existing partitioning algorithms usually do not take into account network conditions and data locality properties, and result in additional costs associated with network communications.

To minimize these issues, it is possible to design an edge-aware partitioning algorithm, which takes into account network location, current workload levels, and resource conditions of computing nodes. These algorithms significantly save bandwidth, balance processing loads and increase overall efficiency. That is why, integration of edge computing with intelligent partitioning algorithms became necessary for modern distributed systems.

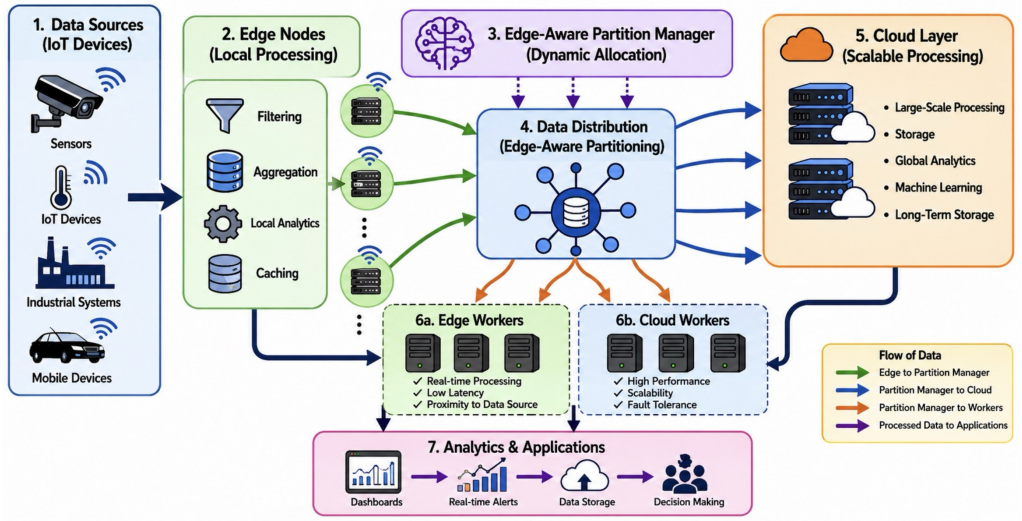


Figure 1: Edge-Cloud Stream Processing Architecture.

Table 2: Edge vs Cloud Processing Characteristics

Metric	Edge	Cloud
Processing Location	Near data source	Centralized data center
Latency	Very Low	Moderate to High
Bandwidth Usage	Low	High
Response Time	Real-Time	Near Real-Time
Resource Capacity	Limited	Very High
Scalability	Moderate	High
Network Dependency	Low	High
Data Locality	Excellent	Limited
Fault Tolerance	Moderate	High
Operational Cost	Lower communication cost	Higher communication cost
Suitable Applications	IoT, Smart Cities, Autonomous Systems	Large-Scale Analytics, Historical Processing
Communication Overhead	Low	High

EDGE-AWARE DATA PARTITIONING TECHNIQUES

One of the essential stages in the process of distributing stream processing in the system is data partitioning. This procedure influences the performance of the system, including latency, throughput, scalability, and resource usage. Classical partitioning algorithms have been designed for use in centralized cloud systems. However, with the emergence of edge computing, there arose a need for developing new methods of partitioning that take into account many parameters, such as proximity of nodes and resource usage.

Hash-Based Partitioning

A common algorithm for data partitioning used in many distributed streaming frameworks, such as Apache Flink or Storm, is hashing-based. In this case, one chooses a specific field or key to calculate a hash value. All records having the same key value are directed to one particular processing node. The main strengths of the algorithm include simplicity and even data distribution in uniform conditions. Moreover, state maintenance becomes easier since all related records are processed by the same worker node.

However, the problem with this type of partitioning is that it fails to deal with skewed data effectively. With popular keys generating too much traffic, some processing nodes might be overloaded, and other nodes will be idle. Also, there will be no consideration for network latency and location of nodes, leading to excessive data transfer.

Range Partitioning

In the case of range partitioning, each record of the input stream falls into a certain range of key values. Each node will process data within a certain range of values. Range partitioning works well when data has something in common, such as temporal or spatial data, making query execution faster.

The problem with range partitioning is that there will always be cases when the incoming stream of data is not balanced over range partitions. As a result, some partitions will have a greater workload compared to others. Also, constant rebalancing of partitions is needed to keep the balance of the workload, adding to communication and administrative costs.

Load-Aware Partitioning

Load-aware partitioning was created as an improvement to static partitioning algorithms. Load-aware mechanism will consider system state, such as current workload, available resources, and other factors. Incoming data will be redistributed to other nodes in order to maintain optimal performance of nodes.

Such an algorithm allows for scaling and adaptation to changing loads. In other words, it is suitable for processing variable-rate streams. At the same time, the problem with load balancing is that it requires continuous monitoring, migrating and syncing of states between nodes. Thus, there will be considerable communication overhead for such operations.

Proposed Edge-Aware Partitioning Algorithm

The proposed edge-aware dynamic partitioning algorithm assigns each incoming stream event to the most suitable processing node by considering data locality, node capacity, current workload, and network latency. Unlike hash-based or range-based partitioning, which mainly distribute events according to keys or fixed ranges, the proposed algorithm evaluates the current condition of available edge and cloud nodes before making a partitioning decision.

Algorithm 1: Edge-Aware Dynamic Partitioning

Input

Incoming event E, edge nodes N, cloud nodes C, available capacity AC, current node load CL, network latency NL, load threshold T

Output

Optimal processing node for event E

Steps

- Receive incoming stream event E from an edge data source.
- Identify candidate edge nodes located nearest to the source of E.
- For each candidate node, collect:
    - available processing capacity,
    - current node load,
    - network latency from the data source,

- bandwidth availability.

- Compute the node selection score:

  $Score(i) = \alpha(AC_i) - \beta(CL_i) - \gamma(NL_i)$

  Where α, β, and γ represent the weighting factors for capacity, load, and latency respectively.

- Select the edge node with the highest score.
- If the selected edge node has load below threshold T, assign event E to that node.
- If the selected edge node exceeds threshold T, evaluate the next highest-scoring edge node.
- If all candidate edge nodes exceed threshold T, forward event E to the nearest available cloud node.
- Update node load statistics after assignment.
- Repeat the process for each incoming stream event.

The algorithm prioritizes nearby edge nodes to reduce communication delay and network traffic. At the same time, it avoids overloading individual nodes by incorporating current workload and capacity into the decision process. The weighting parameters α, β, and γ were adjusted experimentally to balance locality, load distribution, and latency reduction during the simulation.

Table 3: Comparison of Partitioning Techniques

Technique	Advantages	Limitations
Hash-Based Partitioning	Simple implementation, efficient key grouping, good scalability under uniform workloads	Sensitive to workload skew, ignores data locality, may create hotspots
Range Partitioning	Preserves data ordering, efficient for time-series and range queries	Uneven distribution may cause imbalance, requires repartitioning
Load-Aware Partitioning	Dynamic workload balancing, improved resource utilization, adapts to workload changes	Monitoring and synchronization overhead, possible state migration costs
Edge-Aware Partitioning	Considers locality, latency, bandwidth, and workload simultaneously; reduces communication overhead and improves responsiveness	Higher implementation complexity, requires continuous resource monitoring
Hybrid Edge-Cloud Partitioning	Balances edge and cloud resources effectively, supports scalability and resilience	Requires sophisticated orchestration and decision mechanisms
Static Round-Robin Partitioning	Easy deployment, uniform assignment under stable workloads	Ignores workload characteristics and network conditions, poor adaptability

PERFORMANCE EVALUATION AND BENCHMARKING

To measure the efficacy of the edge-aware approach in partitioning data, it was compared to the existing methodologies, namely, hash-based, range, and load-aware data partitioning. Three main criteria were used for the assessment: processing latency, throughput, and network utilization.

The workload consisted of approximately 300 million streaming events (1 KB average size) generated during a 60-minute execution period. The resulting traffic volume ranged from 310 GB to 620 GB depending on the partitioning strategy employed. Each algorithm was run under the same conditions in order to make sure the performance is properly

compared. The edge-aware data partitioning was designed to assign incoming streams to processing nodes according to several parameters, such as data locality, resource availability, network connectivity, and workload distribution.

Processing latency was defined as the time elapsed from receiving an event to its completion. According to the experiments, the hash-based method demonstrated the worst latency performance, mainly due to imbalance in workload and communication overhead. As for range partitioning, it provided slightly faster processing compared to hash-based partitioning. Load-aware partitioning improved the results due to load balancing, yet still did not provide high-performance processing. In its turn, the edge-aware approach showed the best result in terms of latency because of processing close to the data sources and minimizing network communication overhead.

In regard to throughput, a similar trend can be observed. While hash-based partitioning processes the smallest number of events at a time under heavy workloads because of node overloads, the range partitioning demonstrated relatively good performance, yet was still sensitive to workload distribution. Load-aware data partitioning achieved better throughput due to load balancing. The edge-aware approach delivered even better performance because of workload balancing and locality-based processing.

The criterion of network utilization was measured by the amount of data transferred between the edge and cloud layers during processing. As opposed to traditional techniques which require transferring all raw data to the cloud layer and thus generate a considerable amount of network traffic, the edge-aware algorithm reduces network utilization by processing data near sources and transmitting processed information only to the cloud layer.

Thus, benchmarking proves that using edge-awareness in partitioning algorithms provides significant advantages. The edge-aware technique significantly outperforms other approaches in processing latency, throughput, and network utilization. It can be stated that edge-aware data partitioning is an efficient optimization tool.

Table 4: Processing Latency Comparison

Technique	Average Latency (ms)
Hash-Based Partitioning	145
Range Partitioning	118
Load-Aware Partitioning	92
Proposed Edge-Aware Partitioning	58

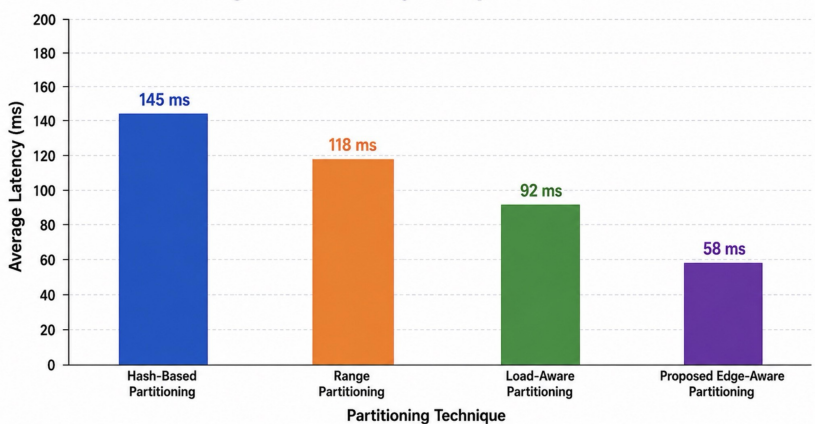
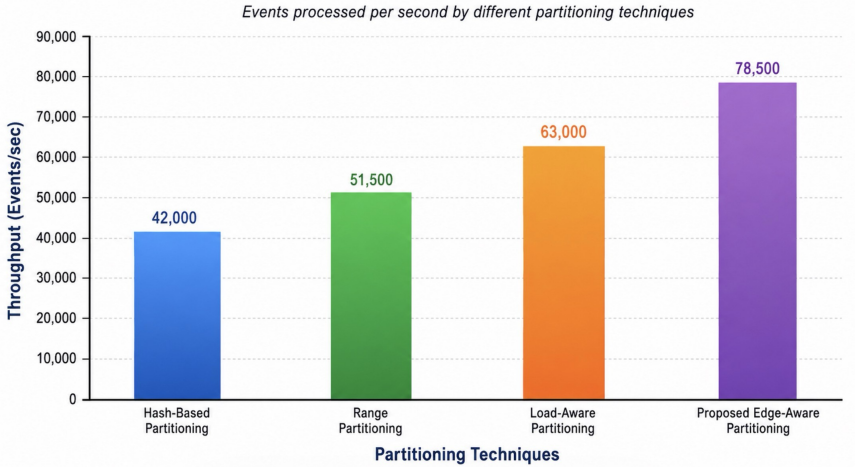


Figure 2: Processing Latency Comparison.

Table 5: Throughput Comparison

Technique	Events/sec
Hash-Based Partitioning	42,000
Range Partitioning	51,500
Load-Aware Partitioning	63,000
Proposed Edge-Aware Partitioning	78,500



Figure

Table 6: Network Traffic Reduction

Technique	Data Transferred (GB)
Hash-Based Partitioning	620
Range Partitioning	540
Load-Aware Partitioning	470
Proposed Edge-Aware Partitioning	310

FINDINGS AND DISCUSSION

Concluding from the results of the performance evaluation, it can be noted that edge-aware partitioning of data provides significant improvements compared to traditional approaches used to implement distributed processing tasks. The simulation experiment showed that taking into account the aspects of data locality, network connections, and available resources when implementing partitioning resulted in noticeable advantages in the field of processing efficiency, reduced communication overheads, and improved system responsiveness.

The first important result is the reduction of processing latency achieved with the help of the proposed partitioning technique. Classical hashing and range partitioning techniques are not concerned with the locations of data sources or the places where the data is processed. That is why in these approaches, the majority of data should pass through the network before processing starts. On the contrary, edge-aware partitioning takes advantage of the proximity of computing devices to process events in real time, which is a useful feature for such use cases as smart manufacturing, autonomous transport, health care systems, and industrial Internet-of-things (IoT) applications.

Another aspect that needs to be highlighted is related to the improvement of computing resource utilization. Standard approaches to partitioning lead to workload imbalance since data is divided among servers unevenly; moreover, some parts of data could create a bottleneck at one processing node while other nodes stay idle, making it hard to use computing resources efficiently. The proposed approach to edge-aware partitioning includes workload analysis and more efficient distribution of incoming streams across different resources available for use.

Scalability issues can also be addressed successfully with the help of edge-aware partitioning. While traditional methods were experiencing growing latency and inefficiency with increasing stream volume because of excessive inter-node communication and workload imbalance, The proposed approach maintained stable performance throughout the simulated workload.

To conclude, this paper managed to demonstrate how edge-aware partitioning could improve distributed stream processing by combining two beneficial ideas of workload balancing and data locality. Not only did the method provide lower latencies and higher throughput; it could also help to utilize computing resources more efficiently and enhance system scalability. All these properties make edge-aware partitioning an appropriate solution for next-generation distributed systems.

Table 7: Summary of Key Findings

Parameter	Traditional Method	Proposed Method (Edge-Aware)
Average Processing Latency	Higher (92–145 ms)	Lower (58 ms)
Throughput	Moderate (42,000–63,000 Events/sec)	High (78,500 Events/sec)
Network Traffic	High communication overhead	Reduced communication overhead
Resource Utilization	Uneven workload distribution	Balanced resource allocation
Data Locality Awareness	Not considered	Explicitly considered
Scalability	Performance degrades under heavy workloads	Maintains stable performance
Load Balancing	Limited effectiveness	Dynamic workload balancing
Edge Resource Usage	Minimal	Optimized utilization
Cloud Dependency	High	Reduced through local processing
Overall System Efficiency	Moderate	Significantly Improved

CONCLUSION AND FUTURE WORK

The emergence of the large amount of data in real-time through the Internet of Things devices, smart infrastructure, industries, and cloud services makes stream processing crucial for distributed systems' effective operation. This paper examines the importance of data partitioning with consideration of edge awareness as a possible solution for enhancing distributed systems' performance. Reviewing various partitioning techniques and their comparative performance analysis, it was found that current approaches, namely hash-based and range-based partitioning, face several disadvantages due to workload imbalance, network congestion, and increased latency in the edge-cloud environment.

The developed edge-aware method of partitioning incorporates the principles of local workload distribution, network state, availability of resources, and data location during the process. As a result, this approach is able to reduce the amount of communication and improve performance, as well as reduce latency in processing tasks by moving data closer to where it is created. Edge-aware data partitioning also shows better scalability compared to conventional approaches since it maintains performance while increasing the workload on the system.

This simulation-based study demonstrates the potential valueof locality-aware workload management in today's edge–cloud architecture and offering an efficient partitioning approach that provides improved performance compared to existing solutions. Edge-aware data partitioning can become a beneficial technique in numerous applications, including smart cities, autonomous vehicles, healthcare monitoring, industry automation, and analytics platforms.

Further research might include creating partitioning algorithms based on machine learning which would be able to predict workload and adjust partitioning according to the prediction. Energy efficient partitioning, multiple-edge collaborative models, and fault tolerance might also be investigated as promising future directions. Furthermore,

application of artificial intelligence, resource prediction, and autonomous systems orchestration might contribute to development of the next generation distributed systems architecture.

REFERENCES

1. *de Assunção, M. D., Veith, A. S., & Buyya, R. (2018). Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. Journal of Network and Computer Applications, 103, 1–17. https://doi.org/10.1016/j.jnca.2017.12.001*
2. *Floratou, A., Agrawal, A., Graham, B., Rao, S., & Ramasamy, K. (2017). Dhalion: Self-regulating stream processing in Heron. Proceedings of the VLDB Endowment, 10(12), 1825–1836. https://doi.org/10.14778/3137765.3137786*
3. *Gulisano, V., Jiménez-Peris, R., Patiño-Martínez, M., Soriente, C., & Valduriez, P. (2012). StreamCloud: An elastic and scalable data streaming system. IEEE Transactions on Parallel and Distributed Systems, 23(12), 2351–2365. https://doi.org/10.1109/TPDS.2012.24*
4. *Hiessl, T., Karagiannis, V., Hochreiner, C., Schulte, S., & Nardelli, M. (2019). Optimal placement of stream processing operators in the fog. In 2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC), 1–10. https://doi.org/10.1109/CFEC.2019.8733147*
5. *Katsipoulakis, N. R., Labrinidis, A., & Chrysanthis, P. K. (2017). A holistic view of stream partitioning costs. Proceedings of the VLDB Endowment, 10(11), 1286–1297. https://doi.org/10.14778/3137628.3137639*
6. *Nasir, M. A. U., Morales, G. D. F., García-Soriano, D., Kourtellis, N., & Serafini, M. (2015). The power of both choices: Practical load balancing for distributed stream processing engines. 2015 IEEE 31st International Conference on Data Engineering, 137–148. https://doi.org/10.1109/ICDE.2015.7113279*
7. *Pacaci, A., & Özsu, M. T. (2018). Distribution-aware stream partitioning for distributed stream processing systems. In Proceedings of the 5th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond, 1–10. https://doi.org/10.1145/3206333.3206338*
8. *Pietzuch, P. R., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M., & Seltzer, M. I. (2006). Network-aware operator placement for stream-processing systems. 22nd International Conference on Data Engineering. https://doi.org/10.1109/ICDE.2006.105*
9. *Sajjad, H. P., Danniswara, K., Al-Shishtawy, A., & Vlassov, V. (2016). SpanEdge: Towards unifying stream processing over central and near-the-edge data centers. 2016 IEEE/ACM Symposium on Edge Computing, 168–178.*
10. *Veith, A. S., de Assunção, M. D., & Lefèvre, L. (2018). Latency-aware placement of data stream analytics on edge computing. In Service-Oriented Computing: ICSOC 2018, 215–229. https://doi.org/10.1007/978-3-030-03596-9_14*