Developing Scalable and Lightweight Data Stream Ingestion Framework for Stream Processing

Nwe Ni Hlaing
Faculty of Computing
University of Computer Studies,Yangon
Yangon, Myanmar
nwenihlaing@ucsy.edu.mm

Thi Thi Soe Nyunt
Faculty of Computer Science
University of Computer Studies,Yangon
Yangon, Myanmar
thithi@ucsy.edu.mm

***Abstract*—According to the development of technology, enormous amount of data are being generated as a continuous basis from Social media, IOT devices, and web etc. This lead to big data era. Many researchers are paying attention on massive amount of data stream processing coming with a rapid rate to gain valuable information in real-time or to make immediate decision .Data Ingestion Stage is an important part in data stream processing system .It is responsible for the data collection from different locations and then deliver this data for processing. The most important requirement of data ingestion is to provide low latency, high throughput, and scalability with many data producers and consumers. It can influence on entire stream processing performance. In big data stream computing, speed at which data being created and explosive growth of data lead to new challenges. One challenge is to accurately ingest different stream data into a processing platform or data storage platform.Current existing data stream ingestion systems use a combination of Apache NiFi and Kafka. Apache Nifi is used for collection and preprocessing of structured and unstructured data feeds. Kafka is used for message distribution. However, processor such as MergeRecord in Nifi can be memory , I/O CPU intensive.As a result,when processing massive data streams creation with high speed can lead to a lot of memory effort , input/output bottleneck or central processing unit (CPU) bottleneck.It leads to impact on the performance of stream processing layer and it is not appropriate for time sensitive applications. In this paper, we propose to use a combination of StreamSets Data Collector and Kafka to collect and transform from various sources of structured and unstructured feeds**.

Keywords— StreamSets Data Collector, Stream Ingestion, Apache Kafka ,Big Data

I. INTRODUCTION

According to unprecedented development of device technologies, such as Internet of Things (IoT), smartphones, smart sensors and 5G technology, enormous amount of data are generated on a daily basis[1]. The amount of data is getting bigger and bigger. In research and industrial communities, Big Data has obtained a great amount of attention .Big Data Stream Computing is one of the type of Big data which need to be processed with low latency to provide real time requirement. In Big Data applications, there has been gradually increasing moving from batch based processing to stream-based processing that enable them to gain valuable information in real-time[3]. Due to explosive growth of data, enterprise also have enormous stream data that necessary to process in real time to make immediate decision for organization profit. There is an increasing demand for E-commerce, web site analytics , intrusion detection and other practical applications.

A typical distributed stream processing systems compose of three layers including data collection , data ingestion and stream processing[1] . In data collection stage, massive amount of stream records are being generated from event sources(e.g. sensors, smart devices).Data ingestion layer is responsible for acquiring stream records, partitioned ,pre-processed to facilitate consumption and manages flow of information from incoming data sources to processing and storage engine .Finally, in stream processing phase, big data engines correspond to consume data stream buffered by data ingestion phase using a pull-based model ,obtaining consequent results and send this result to the storage layer. In storage layer, it has the responsibility for retaining data in databases for long-term persistence or storing data in-memory. The stored data can be used for further processing.

In computing big stream data in real time, it is to reduce necessary to focus on reducing end-to-end latency of the three stage pipeline because of interesting to get immediate results as fast as possible. The importance of data ingestion cannot be overlooked as it can affect on the whole stream processing system. When the speed at which new records are created is very fast , one such challenge is the ability to provide high throughput at the same time .In data ingestion stage, it is essential to acquire stream data records with high throughput from producer and also serve the consumers with high throughput in pulling data records[3]. It also not only need to reduce the producers' write latency but also consumers' read latency to guarantee low end-to-end latency.

Therefore, Big Data engines need to design having scalability with many concurrent consumers, which support processing millions of records within a second [2] . Motion data needs to manage the flow of data because many organizations have stored data that need to be migrated and processed in different locations. Within a local center traditional streaming data analytics systems are mostly limited to handle data flows. Nowadays, many organizations are operating over data centers in distinct geographic locations as the world has become more connected. One such challenge that encounter in streaming analytics system is how to collect ,connect massive data streams across the globes and how to ingest many different data sources obtaining from real-time headlines from social media live multimedia, IoT data and blogs .In data ingestion phase, one challenging task is to ensure how to accurately provide continuous streaming data ingestion and to control management of incoming data flow .So, there is a need to provide new technology and systems that can support robustness and desired scalability in solving

failures. Research study about streaming data ingestion previously has been focused on bulk data loading, deduplication ,data integration and integrity constraint maintenance.

State of the art streaming data ingestion methodologies use a combination of Apache Nifi and Kafka for accurately data ingestion and merging data streams from different location into a processing platform. Apache Nifi is used for data stream acquisition ,data stream integration and extraction.Apache kafka is used for data stream distribution. However ,our observation show that some processor such as MergeContent, MergeRecord in Nifi can be memory , I/O CPU intensive.As a result, high speed of creating new streaming data and enormous amount of data can lead to performance delay in input/output (I/O) and use a a lot of memory, disk/network , or the central processing unit (CPU) bottleneck . One such requirement in data stream processing system is low latency. It leads to impact on the performance of stream processing layer and it is not appropriate for time sensitive applications .In this paper, we propose a light weight ,high performance and scalable streaming data ingestion framework for stream data processing. We propose to use a combination of Streamsets Control Hub instead of Apache Nifi and kafka for data stream distribution to solve performance bottleneck problems in Apache Nifi.

The paper is arranged as follows .In section II, related work with data stream ingestion are reviewed. In section III, describes the propose system architecture. Section IV details our experimental and evaluation results. Finally, concluding the paper.

II. Related Work

Wei Jiang[1] proposed an improved data ingestion architecture for stream processing system in real time General architecture of stream processing system consists of a combination Apache Flume ,Apache Kafka and Storm. Flume is applied for collection the source data from clients. Kafka is deployed for message distribution .Flume is a JVM process which consume resource of producer machine .This matter impact on the performance of the producer's machine. Authors focus on data loading process of kafka.An important factor need to be consider when using kafka is speed mismatch between message producers and consumers as it can impact on the whole stream processing system. Author use Kafka cat as a non JVM process instead of flume to transfer data from the source to Kafka topic directly and this lead to reduce the network transmission.In order to avoid the performance bottleneck problem in Kafka during data loading, Kafka use memory file system instead of disk file system.

Haruna Isah [2] describes among three stages of data stream processing,the importance of data ingestion cannot be overlooked and it has great influence on whole stream processing system.This paper aim to accurately ingest various sources of data streams and integrate data streams generated from various sources into an analytic platform. Authors examines the fundamental requirements of data stream ingestion systems and propose fault-tolerant and scalable a data stream ingestion and integration framework . Authors proposed to use a combination of Apache Nifi and kafka.Apache Nifi is deployed for acquiring data streams, integration and extraction .Kafka is deployed as a message distribution.This combination can serve as a reusable component across many feeds of structured and unstructured input data in a given platform. In this paper, authors point out some nifi processors such as MergeContent can lead to performance bottleneck in input/output (I/O) , memory, disk/network usage, or the central processing Unit because Nifi run inside a java virtual machine. In order to guarantee for high volume and high-performance dataflows, NiFi configuration is far from ideal .

In this paper, authors[3] presents data ingestion system need to provide high throughput, low latency.It also needs the capability to scale many data producers and consumers. Apache Kafka use static stream partitioning and offset-based record access,. These design can impact ingestion phase ability to support high throughput and low latency. In order to alleviate Kafka limitation, authors propose KerA, a data ingestion framework that use a dynamic partitioning scheme and lightweight indexing .It can improve throughput latency and scalability of data ingestion layer.

Gobblin[4] is a kind of data ingestion framework that is developed by LinkedIn. According to an increasing amount of heterogeneous LinkedIn's data sources, authors proposed to use Gobblin as a data ingestion . It supports adaptors for accessing various data sources such as Kafka, MySQL, S3, and Salesforce. Although authors focused on improving latency , scalability and throughput, they do not taking into account provenance and extensibility issues.

III. Proposed System Architecture

The paper emphasize on data ingestion component When processing data streams in real time ,the phase of data ingestion is necessary to guaranetee high throughput and reduce latency as much as possible. As a current data stream ingestion methodology , a combination of apache Nifi and Kafka is deployed to process massive data streams arriving at rapid rate .But sometimes Nifi has low performance because some Nifi processors needs a lot of memory effort, CPU effort. It is not appropriate for making decision in real time as it can increase latency of the processing layer and take a lot of processing time. So, data ingestion phase must be lightweight as it cannot satisfy the deadline constraint. The purpose of this study is for developing lightweight and scalable data ingestion framework for massive amount of streaming data and to flexibly control management of data flow between systems. In order to solve problems in Apache Nifi, we proposed to use StreamSets Data Collector(SDC) instead of Apache Nifi and deploy Kafka for streaming data distribution .Fig. 1 shows framework for proposed data flow management architecture. Our framework architecture is divided into three components ,(1) Streaming Data Collection from disparate sources (2) Data Stream Preprocessing such as extraction ,enrichment and (3) Routing Data Streams to various downstream systems such as storage layer or processing layer.

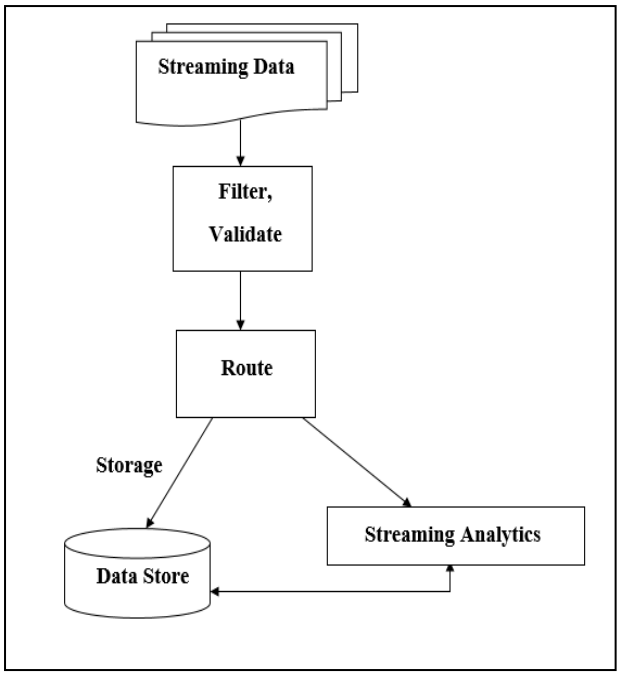

Fig. 1. Data Flow Management Architecture

A. *Streaming Data Collection*

The process of acquiring data streams generated from several sources with different formats is called data streams collection [1] .It is also the main entry point for processing data streams. Incoming data streams from social media such as facebook,twitter ,IoT devices and sensors are bringing into a processing platform .In this paper ,we are deploying StreamSets Data Collector(SDC) for stream acquisition. StreamSets Data Collector Framework provide common data access .The framework [8] supports common data access schemes such as, modern devices Representational State Transfer (REST) API, Streaming API, or custom schemes and sockets or application interaction patterns such as publish-subscribe and stream protocols .

The main reason of choosing StreamSets Data Collector is also a powerful and lightweight data collection engine that can access and collect streams data in real time We can route and process data streams using data collector. This StreamSets Data Collector [6] provides an attractive web-based user interface (UI) to configure pipelines, preview data, monitor pipelines, and review snapshots of data. We can define the flow of data for Data Collector and can configure a *pipeline*. A pipeline consists of stages that represent the origin and destination of the pipeline, and any additional processing that we want to perform. Data Collector processes data when it arrives at the origin and waits quietly when not needed.

We can view real-time statistics about our data, inspect data as it passes through the pipeline, or take a close look at a snapshot of data. We will check each processor inside your dataflow to make sure all processors are correctly configured before we can run .Processors in StreamSets can exchange record. The ingested data stream is automatically converted into standard record-oriented format. There are no queues in between processors in StreamSets Data Collector and they are not represented visually different with Apache NiFi. The advantage of using StreamSets Data Collector is that it can process binary data. Some sources, such as Kafka Consumer, can read messages from the Kafka topic and pass them to other processors or external systems without parsing the structure of the binary message into the record format. This allows us to forward the efficient data to some other destination with minimum overhead.

Some of the StreamSets processors may generate events, including errors. You should use special processors called Executors to handle that. For example, there is Email Executor, which can send emails when an error has occurred.Similar with Nifi, StreamSets Data Collector is used as a technology for motion data and it supports flow-based processing. Moreover it can also be deployed for acquiring data ,processing basic event and to distribute data streams to various downstream components. It can accommodate diverse dataflows being generated from the connected world.

B. *Data Stream Preprocessing*

The ingested data stream is necessary to transform based on intended application and the incoming data stream nature. Preprocessing data streams include several tasks such as noise ,language, content parsing, data type transformations and duplicate detection

- Data Stream Extraction

The incoming data stream can be raw format.We need to convert ingested stream data into different schemas depending on the different kind of downstream analytics. As an example ,detection of duplicate data streams and filtering deduplicate data is a extraction task in stream data ingestion.StreamSets Data Collector can support processors such as Record Deduplicator to detect duplicate record in dataflow.

- Data Stream Enrichment

Data Stream Enrichment is the process of getting data arriving from external sources(such as file, API or database) and adding them with ingested data in order to get more details information. By using join operation, enrichment can be performed. StreamSets Data Collector provides processors such as Static Lookup

C. *Routing Stream Distrtibution*

Distribution unstructured or structured data from multiple producers or multiple sources into storage processing component(such as Cassandra or HDFS) or a processing system (such as Flink , Spark Streaming or Storm) from RabbitMQ, Kafka or MQTT message queue [3] . StreamSets Data Collector(SDC) can be directly used for connecting big data stream computing engines such as Spark Streaming by using custom processors. However, StreamSets Data Collector is not a good choice for using to deliver enormous amount and high speed continuous data streams to processing systems beacause of complex structure pipeline in analytics. Moreover, the process of streaming data ingestion is necessary to extend and scale to multiple producers and multiple consumers . If we add a new data consumers such as Apache Flink job in StreamSets Data Collector, we need to change the flow.

In this paper, we choose to use Apache Kafka for routing data streams to different processing engines or storage engines as it can allow to connect multiple

consumers and multiple producers without changing the flow of data pipeline. The main reason of choosing Apache Kafka as a data steam distribution is the ability of durable ,scalable and fast publish-subscribe messaging system. Apache Kafka is developed by LinkedIn and it has unique advantages comparing with current distributed messaging systems .Apache Kafka has many features including low latency, multi-language support, high reliability, horizontal scalability , asynchronous communications and high throughput. Kafka can also be used for both offline and real-time message processing .It can perform collection and submitting massive log data in a low-latency environment .Apache Kafka is composed of three components such as Kafka producers ,Kafka brokers,Kafka consumers. Kafka Producers are responsible for publishing or writing messages into a Kafka message queue or Kafka topic. A topic can also be viewed as a feed name or message category in which messages are published. A Kafka broker can perform multiple reads ranging from hundreds of megabytes and multiple writes per second from multiple client. The creation of Kafka topics are performed on Kafka broker. Kafka broker is working as a Kafka server and it can also store the required messages. Kafka Consumers is allowed to subscribe one or more Kafka topic to retrieve messages based on messages pulling from Kafka brokers. Kafka Consumers can be divided into offline consumers and real time consumers. Most common use of offline consumers are Hadoop and traditional data warehouse. Offline Consumers are used for consuming and storing messages for offline analysis. NoSQL databases like HBase or Cassandra and Apache Storm are real-time consumers and it can be used for consuming messages and store in unstructured query language database .Especially,Apache Storm is used for data stream processing in real time.

In this paper, StreamSets Data Collector is utilized in the role of Kafka producer. Streamsets Data Collector is bringing data from multiple sources and send data to appropriate Kafka topic .Our proposed ingestion framework ,a combination of StreamSets Data Collector and Kafka can provide the power of adding or removing consumers at any time without modifying data ingestion pipeline. Next section, we will evaluate the performance of data ingestion framework by using global news feeds as a case study of monitoring news articles.

IV. EXPERIMENTAL EVALUATION

We will evaluate performance of our proposed design compared with Nifi –Kafka using data streams that obtained from global news API as a case study of breaking news headlines monitoring in this part.We can obtain worldwide breaking news headlines . Moreover, we can also search articles from blogs all over the web and news sources by using our news API.

A. *Use Case Scenario*

In order to show management of dataflow utility in Fig. 2, we evaluate breaking news articles monitoring as a case study with following requirements.

- Ingest streaming breaking news headlines from News API
- Remove noise such as duplicates and fake news in the data streams
- Distribute breaking news feed to appropriate analytics engines or permanent data store

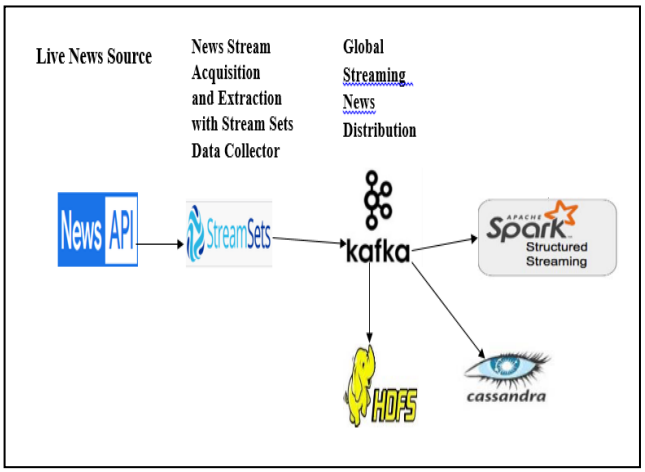

Fig. 2. Worldwide live news articles processing architecture with a lightweight and scalable dataflow management

B. *Performance Evaluation*

We compare data loading performance efficiency of StreamSets Data Collector and Apache Nifi using data streams from News API under the same condition. In Fig. 3 we can see the obtained experimental outcome. Under same condition, we can view data loading efficiency of Streamsets Data Collector based producer are higher throughput than Apache Nifi based producer machine.

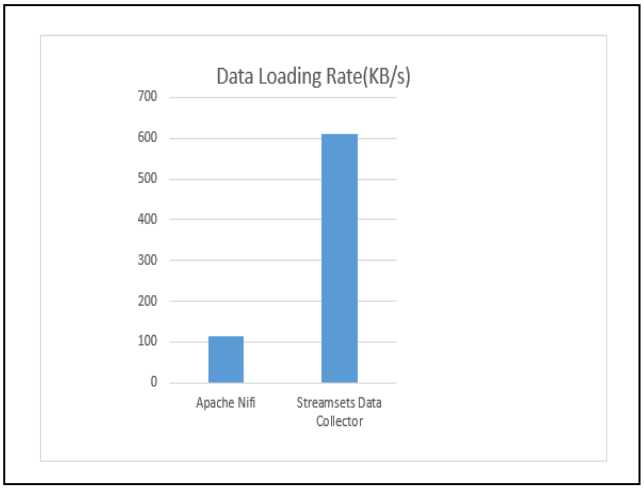

Fig. 3. Performance comparison of loading performance between Apache Nifi and StreamSets Data Collector

As a second experiment, we also test the performance impact of processors such as FlattenJson in Apache Nifi and StreamSets Data Collector .In Fig. 4,we illustrate screenshots of processed world wide live news feeds in Apache Nifi . In StreamSets Data Collector ,live monitoring is the single feature different from Apache Nifi. Monitor panel display statistics for the processing pipeline by default. In Fig. 5,Fig. 6,Fig. 7, we can show summary statistics of processed live news feeds in StreamSets Data

Collector. Fig. 5 shows processed record counts without error during a minute. In Fig. 6,no of records consumed per second is shown and runtime statistics for pipeline can be viewed in Fig. 7.We also find that performance of operators in Streamsets Data Collector is lightweight and faster than Apache Nifi 's operators performance. As a future work, we need to do more detail experiments by exploring deployment and making evaluation for improving the system performance.

Fig. 4. Screenshot of processed worldwide live news feeds in Apache Nifi

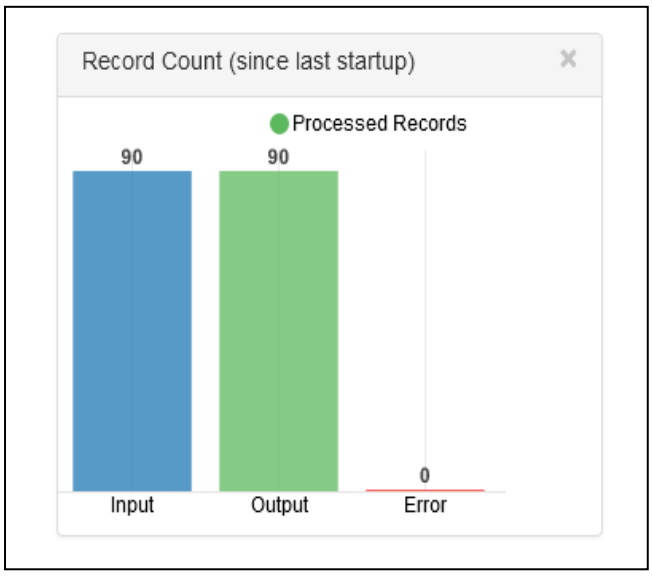

Fig. 5. Record count in StreamSets Data Collector

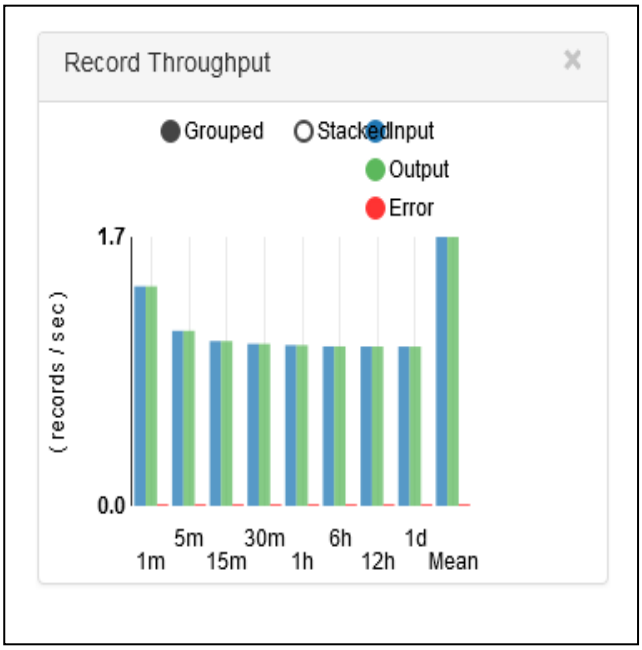

Fig. 6. Record throughput in StreamSets Data Collector

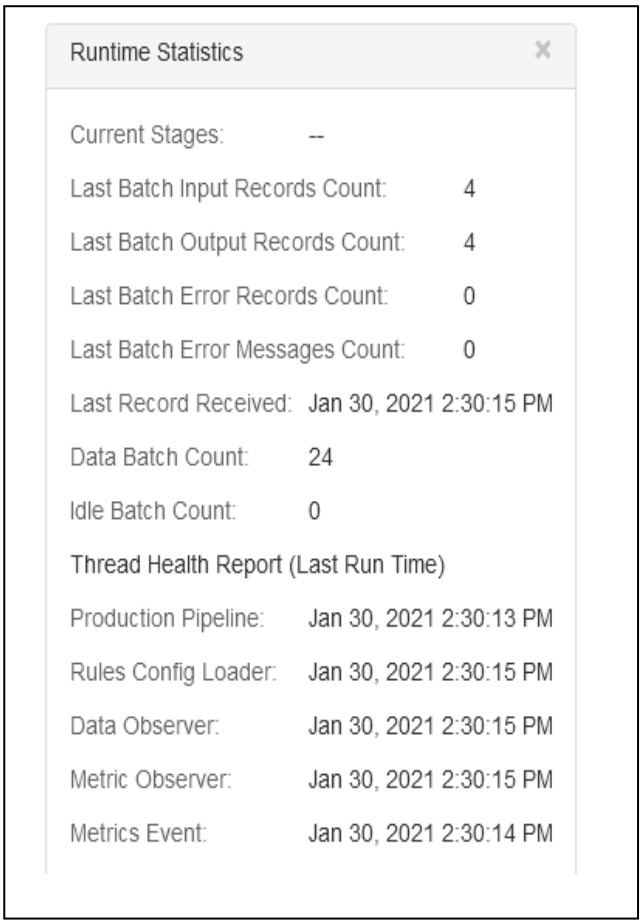

Fig 7. Screenshots of runtime statistics processed live news feeds in StreamSets Data Collector

V. CONCLUSION

Nowadays, massive amount of data stream are generated from various sources such as social media ,sensor and IOT devices. Most organizations collect data and analyze on this data for business profit in real time.The most important requirement of processing streaming data is to provide high throughput and reduce latency. The importance of data stream ingestion cannot be ignored because it can impact on the whole data stream processing system.Data stream Ingestion component should be lightweight, high throughput scalable .Moreover,It need to support the flow of data between many producers and many consumers. As a current data stream ingestion methodology , a combination of apache Nifi and Kafka is deployed to process massive data streams arriving at rapid rate. Apache Nifi is deployed for data stream collection and data stream preproessing. Kafka is used to distribute motion data to processing layer or storage layer. However, Apache Nifi is low performance as processors in Nifi are IO intensive ,memory intensive and CPU intensive. It can lead to undesirable effect such as high latency in processing stream data in real time. So, it cannot be guaranteed to support immediate decision as it miss deadline in data stream processing.

In this study, we propose a lightweight and scalable data ingestion framework for streaming data ingestion. We propose to use a combination of StreamSets Data Collector and kafka. As a case study, we compare data loading efficiency performance of Streamsets Data Collector and Apache Nifi using data streams from News API under the same condition. We find that the data loading efficiency of StreamSets Data Collector is higher than Apache Nifi StreamSets Data Collector is scalable and lightweight. Moreover, it can also manage data flow and it can solve the problem of processors performance bottleneck in Apache Nifi.

REFERENCES

[1] J W. Jiang, L-G.Xu. H-B,Hu, and Y.Mae, “Improvement Design for Distributed Real-Time Stream Processing Systemss,” JOURNAL OF ELECTRONIC SCIENCE AND TECHNOLOGY, VOL. 17, NO. 1, MARCH 2019

[2] H.Isah,F. Zulkernine,” A Scalable and Robust Framework for Data Stream Ingestion”,2018 IEEE International Conference on Big Data (Big Data),2018

[3] O.-C. Marcu *et al.*, "KerA: Scalable Data Ingestion for StreamProcessing," in ICDCS 2018-38th IEEE International Conference on Distributed Computing Systems, pp. 1-6, 2018

[4] L. Qiao et al., "Gobblin: Unifying data ingestion for Hadoop," Proceedings of the VLDB Endowment, vol. 8, no. 12, pp.1764 - 1769,2015

[5] Get breaking news headlines, and search for articles from news sources and blogs all over the web Avaliable : https://newsapi.org/

[6] https://streamsets.com/documentation/datacollector/3.5.0/help/datacollector/UserGuide/Tutorial/BasicTutorial.html