Real Time Data Streaming & Analytics Tools: A Technical Perspective

Shlok Jivtode
20it045@charusat.edu.in
Charotar University of Science and Technology - CSPIT

Mansi Dodia
20it029@charusat.edu.in
Charotar University of Science and Technology - CSPIT

ABSTRACT

Real Time Data streaming is an increasingly popular way to process and transfer large amounts of data in real time or near-real time. This has given rise to numerous live data streaming platforms and technologies that enable the collection, analysis and distribution of data.The utilization of real-time data streaming for processing and transferring substantial volumes of data has gained significant popularity. This trend has led to the emergence of numerous live data streaming platforms and technologies, facilitating the collection, analysis, and distribution of data. This research paper evaluates the current state,trend of live data streaming, considering technological advancements and industry adoption. It highlights key challenges and limitations that must be overcome to achieve widespread adoption and maximize the effectiveness of live data streaming. This research paper thoroughly investigates various aspects of live data streaming, encompassing its benefits, challenges, and applications.

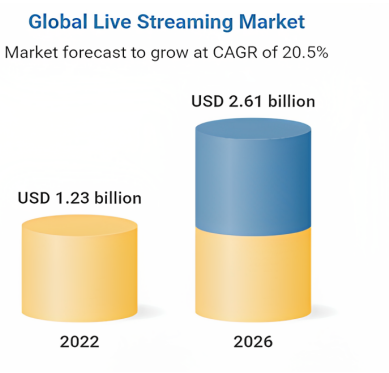
KEYWORDS

Data streaming, Data processing, IoT, Machine Learning, Real Time Analytics, Frameworks

INTRODUCTION

The increasing need for real-time data processing and analysis has led to the growth of live data streaming technologies. Live data streaming refers to the process of continuously sending and receiving data from various sources, allowing for real-time processing, analysis, and visualization. With the growth of the Internet of Things (IoT), mobile devices, and cloud computing, organizations are now able to collect vast amounts of data from various sources in real-time. This data can be used to gain insights into business operations, customer behavior, and market trends, providing organizations with a competitive advantage. According to a survey by Dresner Advisory Services, 45% of organizations are now using real-time data analytics, while another 31% plan to do so in the coming days. According to a survey by IDC, it was found that 49% of businesses have embraced real-time data analytics to gain insights and make informed decisions(1). The development of live data streaming has its roots in the early days of the internet, when data was primarily transferred

through batch processing. Over time, as technology advanced and data became more complex, the need for real-time data processing and analysis grew. This led to the development of various live data streaming technologies, including message-oriented middleware, publish/subscribe systems, and complex event processing engines. These technologies allow organizations to collect and process data in real-time, providing them with valuable insights that can be used to improve their operations and decision-making processes and overall business strategy. By 2025, it is estimated that there will be over 41.6 billion connected devices worldwide, generating 79.4 Zettabytes of data in real-time(2).

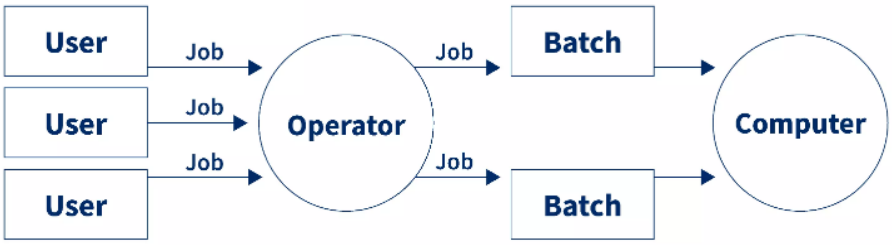


In 2022, the global market for data streaming platforms was valued at $1.23 billion, and is expected to grow at a CAGR of $2.61 billion, from 2022 to 2026(3).

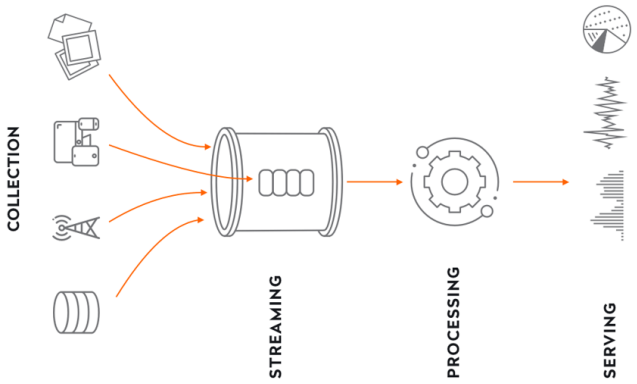
Data Processing Approaches :

Fundamentally, there exist two primary approaches to data processing: batch processing and real-time stream processing.

1. Batch Processing
   When a collection of transactions is accumulated over time, it is a time-saving approach of processing massive volumes of data. Data is gathered, inputted, and processed before batch results are generated. The basic purpose of a batch processing system is to keep all of the tasks in a batch running at the same time(4).



2. Real Time Processing
   Real-time processing systems are extremely rapid and responsive. These systems are employed in situations when a huge number of events need to be accepted and handled quickly. Real-time processing necessitates rapid transactions and is characterized by prompt responses. (5)
   Data must be downloaded in batches before it can be processed, stored, or analyzed in batch data processing methods, whereas streaming data comes in constantly and may be treated at the same time



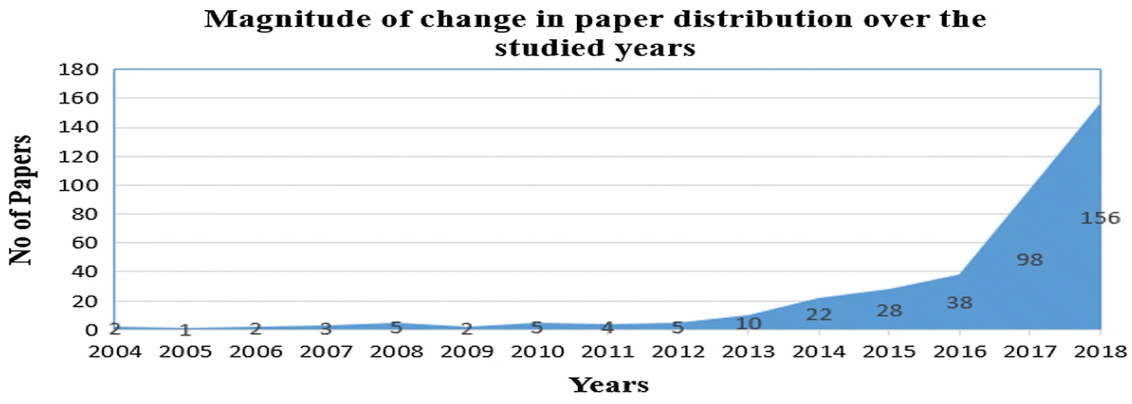
Benefits of Real Time Data Streaming(6) -

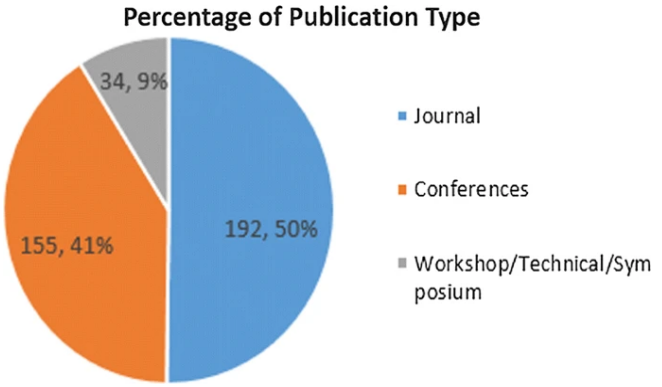
- **Alerting & Monitoring:** Live Data Streaming enables real-time data processing, enabling you to analyze and process the data as it is generated. This is especially important for applications where real-time decision-making is critical, such as: B(7). Financial transactions or real-time traffic monitoring.
- **Scalability:** Live data streaming platforms and technologies are designed to process large amounts of data in real time and to process and transmit data at scale. This is especially important for applications that generate large amounts of data, such as: B. A social media platform or Internet of Things (IoT) device(8).
- **Flexibility:** Live data streaming platforms and technologies offer a great deal of flexibility(9), allowing organizations to choose the methods and tools that best suit their needs. This allows you to adapt to changed requirements and integrate new data sources as needed.
- **Increased Visibility & Deep Insight:** In the healthcare industry, real-time data streaming is used for applications such as patient monitoring, disease surveillance, and clinical decision support(10). One hospital reported that using real-time data analytics reduced sepsis mortality rates by 53%.
- **Innovative Use Cases:** The top use cases for real-time data streaming include fraud detection (70%), real-time customer experience (59%), and predictive maintenance (54%).

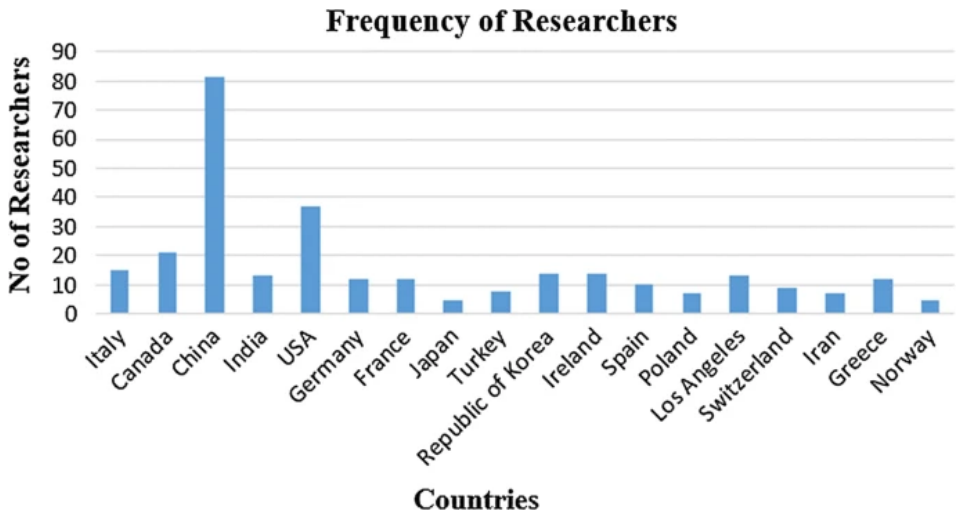
Challenges of Real Time Data Streaming(11) -

- **Latency(12):** Latency is one of the biggest challenges in live data streaming, as even small delays can have a large impact on real-time applications. This requires careful attention to minimize latency and ensure reliable data transfer when designing live data streaming systems.
- **Data Integrity:** Ensuring the integrity and accuracy of data transmitted in real-time is important, as errors and inconsistencies can have a significant impact on real-time applications(13). This requires careful attention to data validation and error checking to ensure data integrity and accuracy.
- **Privacy & Security:** Live data streaming systems are vulnerable to security threats such as hacking and invasion of privacy(14). This requires implementing robust security measures to protect sensitive data and ensure user privacy and security.

- **Legacy Systems and Infrastructure**(15)**:**Legacy systems and infrastructure present several challenges in real-time data streaming. These challenges include limited scalability, as legacy systems may lack the processing power and storage capabilities to handle high volumes of real-time data. Integration issues arise as legacy systems often do not support modern streaming protocols, hindering seamless data ingestion and processing. Latency and performance problems arise due to the lack of optimization for real-time processing. Compatibility issues may arise with outdated or proprietary data formats. Limited analytics capabilities, security and compliance concerns, and maintenance and support challenges further compound the difficulties. Organizations need to consider modernizing their infrastructure and adopting more scalable and compatible technologies to overcome these challenges and harness the benefits of real-time data streaming.
- **Automation and Smartness:**One of the key challenges is automating data processing and decision-making in real-time. This involves designing and implementing intelligent algorithms and models that can analyze streaming data, detect patterns, and make automated decisions or trigger actions(16). Developing such smart systems requires expertise in data science, machine learning, and artificial intelligence. Additionally, ensuring the accuracy and reliability of automated processes in real-time environments is challenging, as data streams are dynamic and can be prone to noise, outliers, or concept drift. Balancing the trade-off between automation and human intervention, as well as addressing ethical considerations and biases, are crucial factors to address in order to achieve effective and trustworthy automation in real-time data streaming.
- **Standardization & Compliance:**The lack of standardized protocols, formats, and interfaces for real-time data exchange makes it difficult to integrate and interoperate between different streaming systems and components. This hinders seamless data flow and interoperability(17), leading to complexity and inefficiency in managing real-time data streams. Compliance with data protection regulations, privacy requirements, and industry standards is also a challenge. Real-time data streaming involves handling sensitive and personal data, and ensuring data security, privacy, and compliance becomes crucial. Organizations must navigate the complexities of data governance, consent management, and auditability in real-time data streaming scenarios to meet legal and regulatory obligations while maintaining data integrity and protecting individual rights. Establishing industry-wide standards and frameworks for real-time data streaming, along with robust compliance mechanisms, is essential to overcome these challenges and enable secure and compliant real-time data processing and exchange.

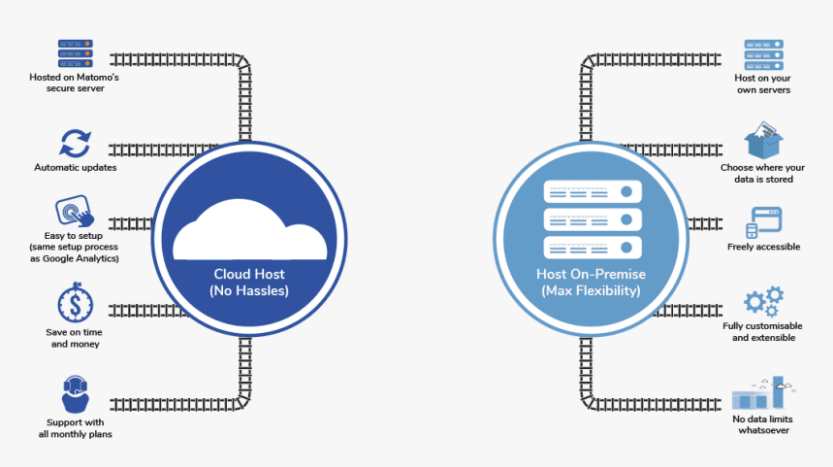






LITERATURE SURVEY

Live data streaming applications can be widely divided into 2 categories – On Premise tools and Cloud Hosted tools(18). Further in the paper we have discussed each category and which tools fall under each category. Later on we reviewed a few tools of each category and compared their working, architecture and performance.



On Premise Tools -

On-premise live data streaming applications refer to applications that are installed and run on local hardware(19), rather than being hosted on the cloud. These applications allow real-time data to be ingested from various sources, processed, and analyzed in real-time. Here are some examples of on-premise live data streaming applications:

- **Apache Kafka:** Apache Kafka is an open-source, distributed streaming platform that allows you to ingest, process, and analyze data streams in real-time. Kafka is commonly used for use cases such as log aggregation, event processing, and real-time analytics(20).
- **Apache NiFi:** Apache NiFi is an open-source data integration platform that allows you to automate the flow of data between systems(21). NiFi can ingest data from a wide range of sources, including files, databases, and IoT devices.
- **Apache Storm:** Apache Storm is a distributed real-time computation system that allows you to process streams of data in real-time(22). Storm provides a fault-tolerant, scalable, and distributed system for processing data streams.
- **Microsoft Stream Insight**: Microsoft Stream Insight is a complex event processing engine that allows you to process and analyze data in real-time. Stream Insight can handle high-speed data streams from a variety of sources, including sensors and IoT devices.
- **IBM Infosphere Streams**(23)**:** IBM Info Sphere Streams is a real-time analytics platform that allows you to process and analyze data streams in real-time. Streams can handle large volumes of data from a wide range of sources, including social media, sensors, and logs.

Overall, on-premise live data streaming applications provide organizations with the ability to process and analyze data in real-time.

1. **Amazon Kinesis**

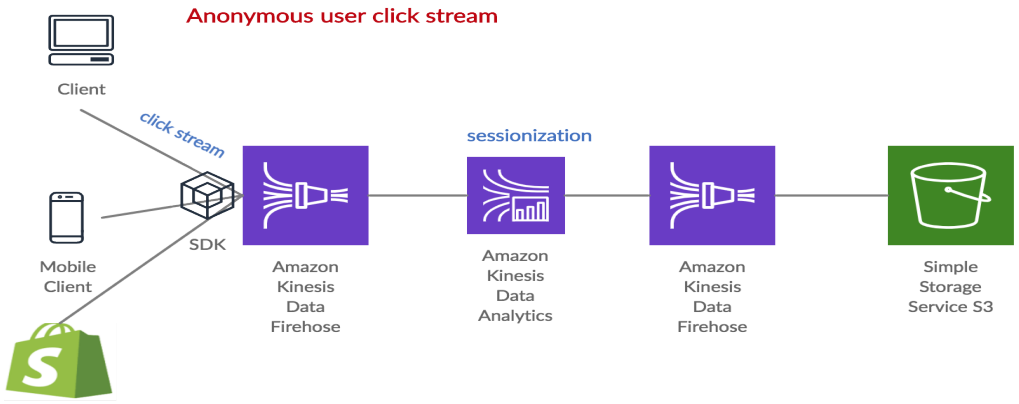
   Amazon Kinesis is an essential feature of Amazon Web Services (AWS) that makes it easy to ingest, process, and analyze video and data streams in real time(24). Key suggestion: This allows you to quickly obtain timely information as well as react immediately to new information.

   Architecture(25) –

   - **Streams:** A stream is the fundamental entity of Amazon Kinesis. It is a sequence of data records that are stored in a distributed manner across multiple shards. Each stream can have multiple producers and consumers. Data is stored in a stream for a specified period of time, and can be processed in real-time.
   - **Producers:** Producers are entities that generate data and send it to a Kinesis stream. Examples of producers include sensors, IoT devices, web applications, and log files.
   - **Consumers:** Consumers are entities that read and process data from a Kinesis stream. They can be applications, analytics tools, or storage systems.
   - **Shards:** A shard is a partition of a stream. It is a unit of capacity that can process a fixed rate of data records per second, up to a maximum size limit. Each shard is hosted

on a separate machine, and data records are evenly distributed across all the shards in a stream.

- **AWS Kinesis Client Library (KCL):** The KCL is a set of libraries that make it easy for developers to consume and process data from a Kinesis stream. It provides features such as load balancing, checkpointing, and error handling.
- **AWS Kinesis Data Analytics:** Kinesis Data Analytics is a fully managed service that enables real-time processing of streaming data using SQL. It allows you to analyze data in real-time, and output results to various destinations such as S3, Redshift, and Elasticsearch.
- **AWS Kinesis Data Firehose:** Kinesis Data Firehose is a fully managed service that enables you to load streaming data into data stores and analytics tools. It can automatically transform and compress data records before loading them into destinations such as S3, Redshift, and Elasticsearch.



Key Features –

It has a number of important features that make it a powerful and adaptable tool for processing and streaming data, such as:

- **Streaming data in real time:** Customers can collect and process streaming data in real time with Amazon Kinesis, allowing them to react quickly to events and gain insights from the data as it is generated.
- **Scalability:** Amazon Kinesis can be scaled to meet your data streaming requirements. Changes in data volume or processing requirements can be easily accommodated by increasing or decreasing the number of streams or shards(26).
- **Durability:** By default, Amazon Kinesis can store data for up to seven days, but it can be set up to store data for up to 365 days. Customers are able to replay or examine historical data as a result of this(27), even after processing has taken place.
- **Integration with other services offered by AWS:** Customers can construct complete, end-to-end data processing pipelines thanks to Amazon Kinesis's integration with other AWS services like Amazon S3, Amazon Redshift, and AWS Lambda.
- **Analytics and artificial intelligence:** Customers can analyze and gain insights from their streaming data thanks to Amazon Kinesis's integration with AWS analytics and

machine learning services like Amazon EMR, Amazon Elasticsearch, and Amazon SageMaker.

- **Security:** Amazon Kinesis integrates with AWS Identity and Access Management (IAM)(29) to control who can access data streams and offers encryption both in transit and at rest.
- **Tools for developers:** Customers can quickly and easily develop custom applications that stream and process data in real-time with the assistance of Amazon Kinesis's developer tools, which include SDKs for Java, Python, and other programming languages as well as a REST API.

Strengths -

- **Reliability:** Minimize data loss with synchronous redundancy of streaming data across all Availability Zones in an AWS Region.
- **Security:** Sensitive data can be encrypted in KDS for private data access through Amazon Virtual Private Cloud (VPC).
- **Easy to use and low cost:** Components such as connectors, agents, and the Kinesis Client Library (KLC) help you build streaming applications quickly and efficiently. There are no upfront fees for Kinesis Data Streams. You only pay for the resources you use.
- **Elasticity and real-time performance(30):** According to SNDK Corp, applications can easily and dynamically scale from gigabytes per hour to terabytes per hour by throttling throughput. Real-time analytics applications can receive real-time streaming data in a very short time after the data is collected.

Weaknesses –

While Amazon Kinesis has many advantages, it may also have some disadvantages, such as:

- **Complexity:** To properly configure and manage Amazon Kinesis, which is a complicated service, significant technical expertise is required. Customers must be familiar with AWS infrastructure and tools and have a thorough understanding of real-time data processing and streaming.
- **Cost:** Amazon Kinesis can be expensive, especially for customers who need to stream a lot of data at once. To fully utilize Kinesis, customers may need to pay for additional AWS services like AWS Lambda, which can raise costs.
- **Insufficient integrations:** Even though Amazon Kinesis is capable of integrating with other AWS services, it only has limited integrations with third-party platforms, making it challenging to utilize in multi-cloud environments.
- **Restricted adaptability:** Amazon Kinesis is capable of handling large-scale data streaming requirements, but its scalability is limited. To handle very high data volumes, customers may need to use multiple Kinesis streams, which can make the configuration more complicated.
- **Conformity and safety:** While Amazon Kinesis offers data security features like encryption and access control, customers are still responsible for adhering to HIPAA

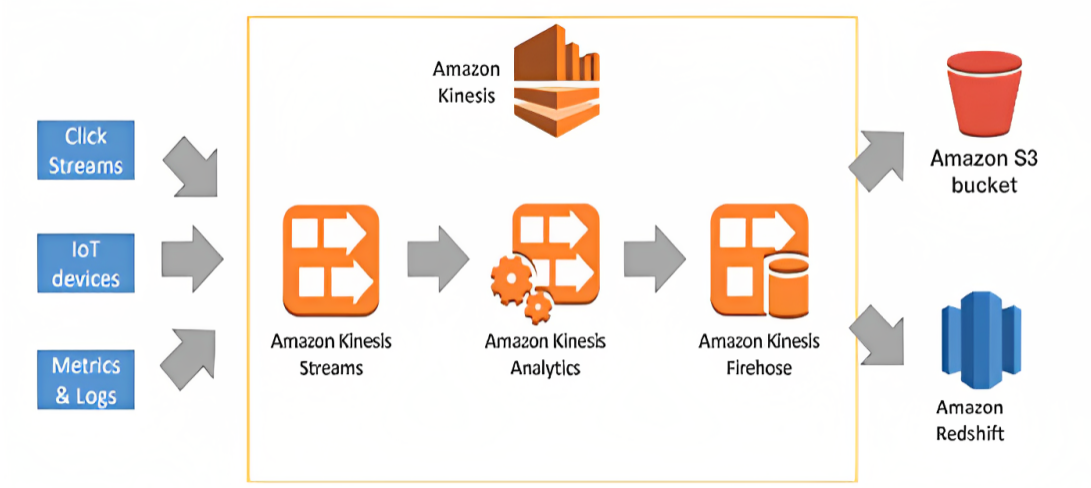
and GDPR regulations when using the service. When dealing with sensitive data streams, compliance can be particularly challenging.

Use Cases –

Use Case	Description	Data Source	Data Sink	Benefit
Real-time data streaming	Kinesis can stream and process data in real-time, making it an ideal solution for use cases such as real-time analytics, fraud detection, and monitoring.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards or alerts, databases, or data warehouses.	Real-time insights, faster decision making, and reduced response time.
Clickstream analysis	Kinesis can process clickstream data in real-time, enabling businesses to extract insights and personalize user experiences.	Clickstream data from websites or mobile apps.	Real-time dashboards, databases, or data warehouses.	Personalized user experiences, targeted advertising, and improved customer retention.
Media ingestion and processing	Kinesis can be used to ingest and process media files such as videos, audio, and images, making it an ideal solution for use cases such as media transcoding, analysis, and storage.	Media files from various sources.	Storage, databases, or other media processing services.	Scalability, faster processing times, and cost-effectiveness.

Real-time inventory management	Kinesis can be used to monitor inventory levels in real-time, enabling businesses to optimize inventory levels, reduce waste, and improve customer satisfaction.	Inventory data from sensors or other sources.	Real-time dashboards or alerts, inventory management systems.	Reduced inventory costs, improved customer satisfaction, and better decision-making.
IoT Analytics	Kinesis can be used to process data from IoT devices, extract insights, and trigger actions in real-time.	Streaming data from IoT devices.	Real-time dashboards, alerts, or trigger actions in	Real-time insights, proactive maintenance, and reduced downtime.

Tool	Salient features	Integration with	Real time application
Apache Kafka	Concurrent processing Can move large data quickly	Apache Hive	Uber Netflix
Amazon Kinesis	Scalable customizable solution	Amazon Firehose	Log Monitoring Web Analytics

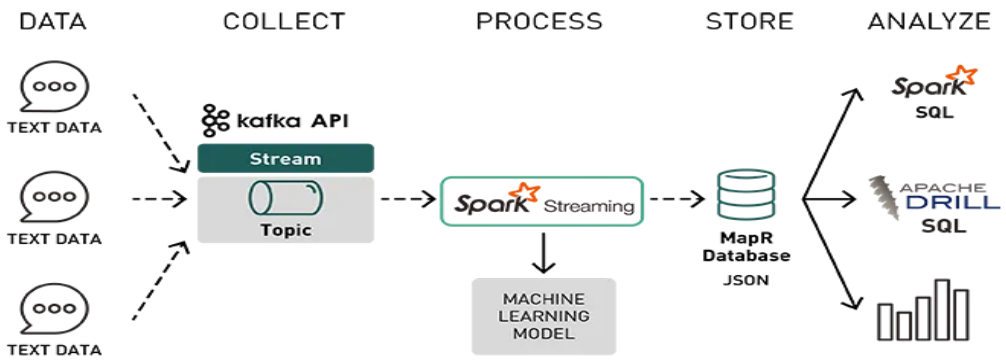


2. **Apache Kafka**

Apache Kafka is a distributed streaming platform that is designed to handle real-time data feeds(31). It was developed by the Apache Software Foundation and is written in Scala and Java.Kafka allows applications to publish and subscribe to streams of records, which can be data or events. These records are stored in a distributed and fault-tolerant way, making it possible to build real-time data pipelines and stream processing applications(32).Kafka is often used for building real-time data processing applications, including streaming ETL (33)(extract, transform, load) pipelines, real-time analytics, and event-driven architectures. It can handle large volumes of data and is known for its scalability, reliability, and performance(34).

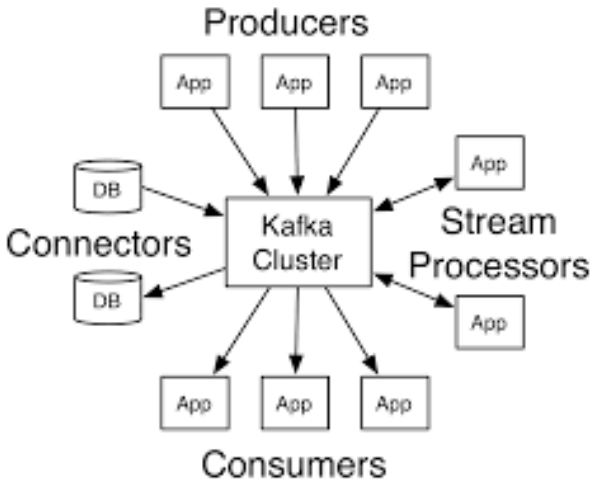
Architecture(35)-

- **Topics:** A topic is a category or feed name to which messages are published by producers. A topic in Kafka can have multiple partitions that are spread across different brokers.
- **Partitions:** A partition is a unit of parallelism in Kafka. Each partition is an ordered and immutable sequence of messages that is assigned to a single broker.
- **Brokers:** A broker is a Kafka server that runs in a cluster. Brokers are responsible for storing and managing partitions.
- **Producers:** A producer is an application that publishes messages to a Kafka topic. Producers can write to a specific partition or let Kafka choose a partition based on a partitioning strategy.
- **Consumers:** A consumer is an application that subscribes to a Kafka topic and reads messages from it. Consumers can read from a specific partition or consume from all partitions in a topic.
- **Consumer Groups:** A consumer group is a set of consumers that work together to consume messages from one or more partitions of a topic. Each partition is consumed by only one consumer within a group.
- **ZooKeeper:** ZooKeeper is a coordination service that is used by Kafka to manage(36).



Features(37)-

- **High-throughput and Low-latency:** Kafka is designed to handle high volumes of data with low latency. It can handle millions of messages per second and offers consistent performance even under high loads.
- **Distributed and Scalable:** Kafka is a distributed platform that can scale horizontally by adding more brokers to the cluster. It can handle large volumes of data while maintaining high performance and reliability.
- **Fault-tolerant:** Kafka is fault-tolerant and can automatically recover from failures by replicating data across multiple brokers in the cluster.
- **Real-time Streaming:** Kafka offers real-time streaming of data, allowing for instant processing of data as it is generated.
- **Durability:** Kafka stores data for a configurable retention period, allowing consumers to replay messages for a certain period of time.
- **Multiple Client APIs:** Kafka supports multiple client APIs, including Java, Python, and C++, allowing developers to choose the language and API that best fits their needs.
- **Security:** Kafka supports encryption and authentication, allowing for secure communication between producers, consumers, and brokers.
- **Compatibility:** Kafka is compatible with other big data tools and technologies, such as Apache Spark and Apache Storm, allowing for seamless integration with existing data processing workflows.



Strengths:

- **Scalability:** Kafka's distributed architecture allows it to scale horizontally by adding more brokers to the cluster, making it easy to handle large volumes of data and traffic.
- **Fault-tolerance:** Kafka is designed to be highly fault-tolerant and can automatically recover from failures by replicating data across multiple brokers, ensuring data availability and durability.
- **High-throughput and low-latency:** Kafka can handle millions of messages per second with low latency, making it ideal for real-time data processing and analytics.

Weaknesses:

- **Complexity:** Setting up and configuring a Kafka cluster can be complex and time-consuming, requiring expertise in distributed systems and network architecture.
- **Monitoring and management:** Kafka requires a robust monitoring and management system to ensure high availability, reliability, and performance, which can be challenging to set up and maintain.
- **Limited tooling for data processing:** Kafka is primarily a data transport platform and does not offer extensive data processing capabilities. While it integrates with other data processing tools, such as Apache Spark and Flink, developers may need to write custom code to build more complex data processing pipelines

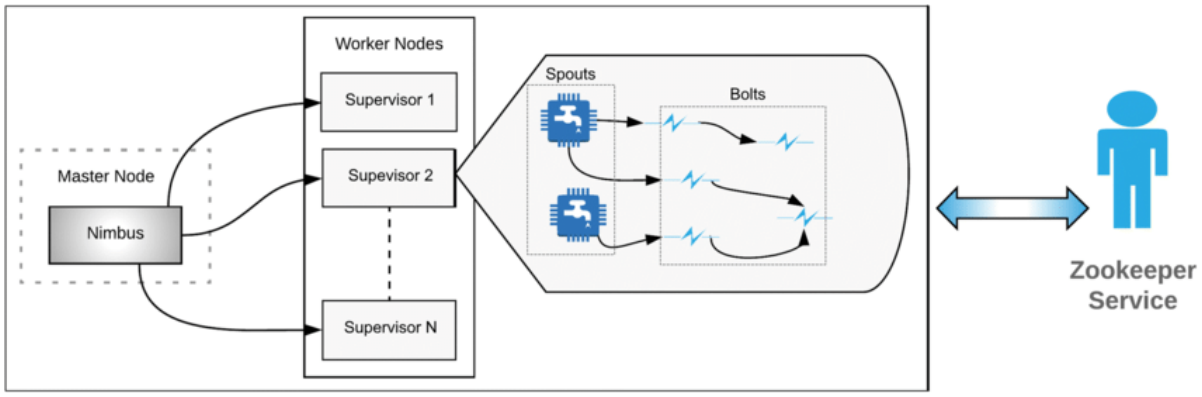
Use Cases:-

- **Messaging:** Kafka is often used as a messaging system(38) for real-time data transfer between applications, services, and systems. It provides a scalable and reliable way to transmit messages and data between different parts of a distributed system.
- **Log Aggregation:** Kafka is frequently used for log aggregation, collecting data from various systems and applications and storing it in a central location for analysis and monitoring. It can store log data for extended periods and allow analysts to replay the data to understand historical trends and patterns.
- **Stream Processing:** Kafka is ideal for stream processing, allowing developers to process data in real-time(39) as it arrives in the system. It can integrate with other stream processing tools such as Apache Flink and Spark Streaming to enable real-time analytics and machine learning.
- **IoT Data Processing:** Kafka is used for processing and analyzing data generated by IoT devices(40), including sensor data, machine data, and telemetry data. Its ability to handle high volumes of data and low latency make it ideal for IoT applications.
- **Metrics Monitoring:** Kafka can be used for metrics monitoring, collecting and aggregating data from different systems and applications to provide real-time visibility into system performance and health.
- **Clickstream Data Processing:** Kafka can handle high volumes of clickstream data generated by websites and applications, allowing for real-time analysis and insights into user behavior and engagement.

Use Case	Description	Data Source	Data Sink	Benefit
Real-time data streaming	Kafka can stream and process data in real-time, making it an ideal solution for use cases such as real-time analytics, fraud detection, and monitoring.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards or alerts, databases, or data warehouses.	Real-time insights, faster decision-making, and reduced response time.
Log aggregation	Kafka can be used for centralized log aggregation, enabling businesses to store, process, and analyze logs from multiple applications and systems.	Application logs, system logs, or network logs.	Analytics tools, dashboards, or data warehouses.	Improved visibility, faster troubleshooting, and reduced downtime.
Message queuing	Kafka can be used for message queuing, enabling applications to send and receive messages asynchronously.	Messages from various sources.	Other applications, databases, or message brokers.	Scalability, fault-tolerance, and faster processing times.
Event sourcing	Kafka can be used for event sourcing, enabling businesses to store all changes to their data as a sequence of events.	Events from various sources.	Databases or other systems that consume event data.	Improved data integrity, auditability, and easier data recovery.
Machine learning	Kafka can be used for machine learning pipelines such as data pre-processing, feature extraction, and model training.	Large data sets stored in Kafka topics.	Machine learning models, data warehouses, or other systems.	Faster model training, more accurate predictions, and reduced costs.

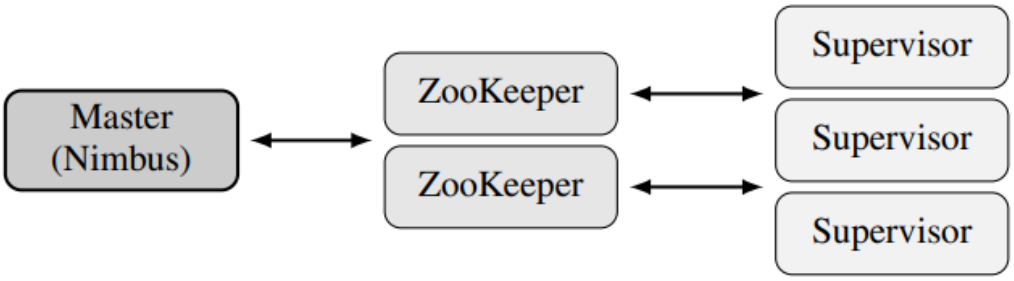
3. Apache Storm

Apache Storm is a real-time stream processing system ,Storm is a master-slave architecture(41). There is a master server called Nimbus that runs on a single node called the master node. There is a subservice called Supervisor that runs on each worker node. Supervisors run one or more worker processes, called workers, that run in parallel to process input. Workflows save their output to a file system or database(42). Storm uses Zookeeper to orchestrate the distributed process. The diagram shows a Storm architecture with one master node and five worker nodes. A Nimbus process is running on the master node. Each worker node runs one supervisor process. Each worker node runs multiple worker processes. A worker receives input from a file system or database and also stores output from a file system or database.



Apache Storm features(43) include -

- This is open source and part of the Apache project.
- Helps with big data processing and is a fast and reliable processing system.
- It can receive large volume and high-speed data.
- Has a high degree of parallelism, scalability and fault tolerance



Strengths –

- Apache Storm is a distributed real-time computation system designed to process large streams of data in parallel. Some of its key strengths include:
- **Scalability:** Storm is highly scalable and can be easily scaled horizontally to handle large amounts of data and processing requirements.

- **Fault-tolerant:** Storm is designed to be fault-tolerant and can recover from node failures, network partitions, and other issues without losing any data.
- **Real-time processing:** Storm processes data in real-time, making it ideal for applications that require immediate processing and quick response times.
- **Flexibility:** Storm is a flexible system that can be integrated with a variety of programming languages, data stores, and messaging systems.
- **Easy to use:** Storm is relatively easy to use and has a simple API, making it accessible to developers with different skill levels.
- **High performance:** Storm is designed to process large volumes of data with low latency and high throughput, making it suitable for high-performance applications.

Weaknesses –

- **Complexity:** Although Storm is relatively easy to use compared to other distributed computing frameworks, it can still be complex and challenging to set up and configure.
- **Resource-intensive:** Storm can be resource-intensive, requiring large amounts of memory and processing power, which can be a challenge for smaller organizations or those with limited resources.
- **Limited windowing support:** Storm has limited support for time-based windowing, which can be a challenge for applications that require time-based aggregation of data.
- **Lack of built-in data persistence:** Storm does not have built-in data persistence, which means that users need to rely on external data stores or other tools to store and analyze data over time.
- **Limited support for machine learning:** Storm does not have built-in support for machine learning algorithms, which can be a limitation for applications that require advanced analytics.
- **Steep learning curve:** While Storm has a simple API, it can still have a steep learning curve for developers who are new to distributed computing systems.

Use Cases -

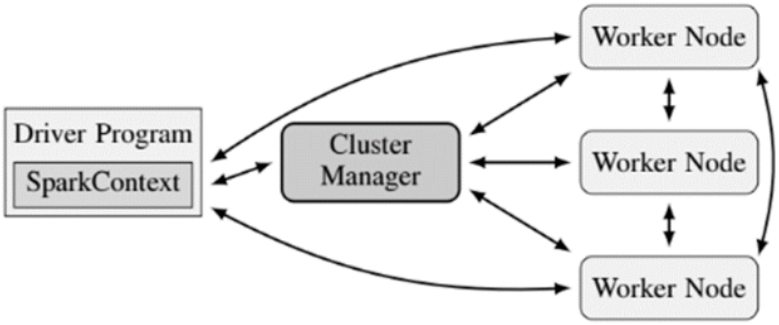
- Credit card companies can use it for fraud detection on swipe.
- Investment banks can use it for trade pattern analysis in real time.
- Retail stores can use it for dynamic pricing.
- Transportation providers can use it for route suggestions based on traffic data.
- Healthcare providers can use it for the monitoring of ICU sensors.
- Telecom organizations can use it for processing switch data.

Use Case	Description	Data Source	Data Sink	Benefit
Real-time data processing	Storm can process real-time data streams, making it an ideal solution for use cases such as real-time analytics, monitoring, and fraud detection.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards or alerts, databases, or data warehouses.	Real-time insights, faster decision-making, and reduced response time.
Distributed data processing	Storm can be used for distributed data processing, enabling businesses to process large amounts of data in parallel across a cluster of nodes.	Data from various sources.	Other systems, databases, or data warehouses.	Scalability, fault-tolerance, and faster processing times.
Complex event processing	Storm can be used for complex event processing, enabling businesses to detect patterns, anomalies, and trends in real-time data streams.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards, alerts, or other systems.	Improved situational awareness, proactive decision-making, and reduced risk.
Fraud detection	Storm can be used for real-time fraud detection, enabling businesses to detect and prevent fraud in real-time data streams.	Data from various sources, such as financial transactions or user behavior.	Real-time alerts or other systems.	Reduced financial losses, improved customer satisfaction, and increased trust.

Real-time inventory management	Storm can be used for real-time inventory management, enabling businesses to monitor inventory levels in real-time, optimize inventory levels, reduce waste, and improve customer satisfaction.	Inventory data from sensors or other sources.	Real-time dashboards or alerts, inventory management systems.	Reduced inventory costs, improved customer satisfaction, and better decision-making .

4. **Spark Streaming**

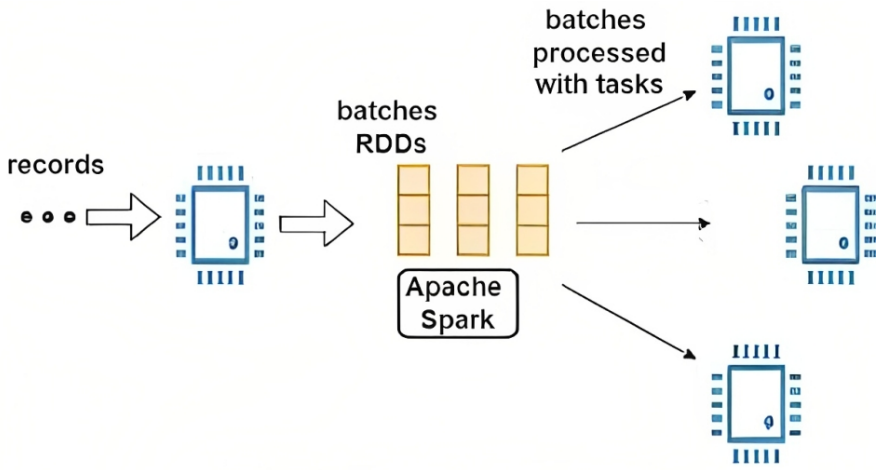
Spark Streaming is an extension of the core Spark API(44)that enables scalable, fault-tolerant, high-throughput stream processing of live streams. It can ingest data from many sources such as Kafka, Kinesis, TCP sockets, and process it using complex algorithms expressed using high-level functions such as map, reduce, join, and window. Finally, processed data can be transferred to file systems, databases, and live dashboards(45). In fact, you can apply Spark's machine learning and graph processing algorithms to your data streams.



Architecture(46)-

- **Input DStreams:** Spark Streaming processes data in the form of continuous streams of data, called DStreams. Input DStreams can be created from various sources, such as Kafka, Kinesis, or Flume, or by transforming existing DStreams.
- **Processing Engine:** The processing engine of Spark Streaming is responsible for processing the incoming data and executing the transformations defined in the application. The processing engine uses Spark's core engine to execute transformations in parallel across a cluster of machines, providing high-level fault tolerance and scalability.
- **Receivers:** Receivers are the components of Spark Streaming that receive the data from the input sources and convert them into DStreams. Each receiver runs as a separate task in the cluster, allowing Spark Streaming to scale horizontally to accommodate large data streams.

- **Batch Interval:** Spark Streaming processes data in batches, with a configurable batch interval that determines how often the processing engine processes the incoming data. The batch interval can be adjusted to achieve a balance between latency and processing overhead.
- **State Management:** Spark Streaming provides a mechanism for maintaining the state of the application between batches, allowing for the processing of stateful operations such as windowed aggregations or sessionization. The state is maintained in memory, with periodic checkpoints to a fault-tolerant storage system for durability.
- **Output Operations:** The final results of the processing can be output to various sinks, such as databases, file systems, or message queues. Spark Streaming provides a set of built-in output operations, such as write to HDFS or publish to a Kafka topic.



Records processed in batches with short tasks
Each batch is an RDD (partitioned dataset)

Features -

For processing and analyzing large amounts of real-time streaming data, Spark Streaming is a powerful tool. Among its most important features are:

- **Advanced API:** Using the same programming model as Apache Spark's batch processing jobs, developers can create streaming jobs with Spark Streaming's high-level API.
- **Fault-tolerance:** Spark Streaming is fault-tolerant, which means that it can handle network partitions and node failures without losing any data.
- **Processing in real time:** Spark Streaming is ideal for applications that require immediate processing and quick response times because it processes data in real time.
- **Processing in microbatch:** Spark Streaming makes use of micro-batch processing, which enables it to process data in small batches rather than individually for each event. As a result, performance and scalability are enhanced.
- **Compatibility with other tools:** Advanced analytics on streaming data can be carried out by integrating Spark Streaming with other tools in the Apache Spark ecosystem, such as Spark SQL, MLlib, and GraphX.

- **Processing with windows:** Spark Streaming lets developers work with data in a sliding window by supporting window-based processing.
- **Multiple sources of input:** Spark Streaming can read data from Kafka, Flume, and the Hadoop Distributed File System (HDFS) as input sources.
- **Processing in a state:** Spark Streaming makes it possible for developers to carry out intricate calculations on streaming data by allowing them to maintain state across multiple batches of data.

Strengths -

- **High Efficiency:** Spark Streaming is suitable for a variety of use cases, including real-time analytics, fraud detection, and more, because it can process a large amount of data in real time with low latency.
- **Simple to Use:** Spark Streaming is a straightforward programming model that is simple to understand and use, especially for Spark aficionados.
- **Incorporation with Flash Environment:** The rest of the Spark ecosystem, which provides access to numerous data sources and processing tools, is seamlessly integrated with Spark Streaming.
- **Tolerance for Faults:** Spark Streaming is a dependable option for mission-critical applications because it is designed to gracefully handle failures and can recover from driver and worker node failures.

Weaknesses –

- **Complexity:** Spark Streaming can be hard to set up and keep running, especially if you don't know much about distributed computing.
- **Processing Costs:** In particular for small batch sizes, Spark Streaming introduces some processing overhead that can affect performance and latency.
- **Sizes of limited processing windows:** Spark Streaming can only handle processing windows of a fixed size, which can be a problem in some situations.
- **Few Streaming Options:** Out of the box, Spark Streaming only supports a small number of streaming sources, which may necessitate custom integration for some applications.

Use Cases(47) –

- **Social Media Analysis:** Social media generates a massive amount of data in real-time. Spark Streaming can be used to analyze this data in real-time and extract useful insights, such as sentiment analysis, trending topics, and user behavior patterns.
- **Fraud Detection:** Spark Streaming can be used to detect fraud in financial transactions by analyzing large volumes of transactional data in real-time. It can identify patterns and anomalies in the data and raise alerts if it detects any suspicious activity.
- **IoT Sensor Data Processing:** IoT devices generate a huge volume of data that needs to be processed in real-time. Spark Streaming can be used to process this data and extract valuable insights, such as temperature, humidity, and other sensor readings.

- **Log Analysis:** Log files contain a wealth of information that can be analyzed to identify issues and trends. Spark Streaming can be used to analyze log data in real-time and provide alerts when issues arise.
- **Recommendation Systems:** Spark Streaming can be used to process large volumes of user data in real-time to generate recommendations for products or services. This can be useful for e-commerce platforms or media companies.

Use Case	Description	Data Source	Data Sink	Benefit
Real-time data processing	Spark Streaming can process real-time data streams, making it an ideal solution for use cases such as real-time analytics, monitoring, and fraud detection.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards or alerts, databases, or data warehouses.	Real-time insights, faster decision-making, and reduced response time.
Distributed data processing	Spark Streaming can be used for distributed data processing, enabling businesses to process large amounts of data in parallel across a cluster of nodes.	Data from various sources.	Other systems, databases, or data warehouses.	Scalability, fault-tolerance, and faster processing times.
Complex event processing	Spark Streaming can be used for complex event processing, enabling businesses to detect patterns, anomalies, and trends in real-time data streams.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards, alerts, or other systems.	Improved situational awareness, proactive decision-making, and reduced risk.

Fraud detection	Spark Streaming can be used for real-time fraud detection, enabling businesses to detect and prevent fraud in real-time data streams.	Data from various sources, such as financial transactions or user behavior.	Real-time alerts or other systems.	Reduced financial losses, improved customer satisfaction, and increased trust.
Real-time inventory management	Spark Streaming can be used for real-time inventory management, enabling businesses to monitor inventory levels in real-time, optimize inventory levels, reduce waste, and improve customer satisfaction.	Inventory data from sensors or other sources.	Real-time dashboards or alerts, inventory management systems.	Reduced inventory costs, improved customer satisfaction, and better decision-making.

Cloud Hosted Tools -

Cloud-hosted live data streaming applications refer to applications that are hosted and run on cloud-based infrastructure(48). These applications allow real-time data to be ingested from various sources, processed, and analyzed in real-time. Here are some examples of cloud-hosted live data streaming applications:

- **Google Cloud Dataflow:** Google Cloud Dataflow is a fully-managed data processing service that allows you to build batch and streaming data processing pipelines. Dataflow provides a unified programming model for both batch and streaming data processing, allowing you to write your pipelines once and run them on either mode.
- **Amazon Kinesis:** Amazon Kinesis is a fully-managed streaming data platform that allows you to ingest, process, and analyze data streams in real-time. Kinesis provides pre-built connectors for popular data sources such as Amazon S3, Amazon DynamoDB, and Amazon Redshift.
- **Microsoft Azure Stream Analytics:** Microsoft Azure Stream Analytics is a fully-managed event processing engine that allows you to process and analyze real-time data from a wide range of sources. Stream Analytics integrates with many Azure services, such as Azure Event Hubs and Azure IoT Hub.
- **IBM Streaming Analytics:** IBM Streaming Analytics is a cloud-based real-time analytics platform that allows you to process and analyze data streams from a variety of sources. Streaming Analytics can handle large volumes of data from sources such as IoT devices and social media.

- **Confluent Cloud:** Confluent Cloud is a fully-managed streaming platform based on Apache Kafka. It provides a scalable, reliable, and secure platform for streaming data and provides pre-built connectors for many popular data sources.

Overall, cloud-hosted live data streaming applications provide organizations with a scalable, cost-effective, and fully-managed solution for processing and analyzing data in real-time. These applications can be easily integrated with other cloud-based services, allowing organizations to build end-to-end data processing pipelines.

Aspect	**Batch Processing**	**Stream Processing**
Data Processing	Processes data in fixed-size batches	Processes data continuously in real-time
Latency	High latency	Low latency
Input Data	Static, finite datasets	Continuous, potentially infinite data streams
Data Ordering	Not time-sensitive	Time-sensitive
Output Generation	Generates output after processing	Generates output in near real-time
Fault Tolerance	Relatively simple	Requires complex fault-tolerant mechanisms

Scalability	Scales well for large datasets	Scales well for high data ingestion rates
Use Cases	Periodic reporting, data analytics	Real-time analytics, monitoring, alerting
Examples	Monthly sales reports, batch jobs	Real-time stock market analysis, IoT data

5. **Google Cloud Dataflow**

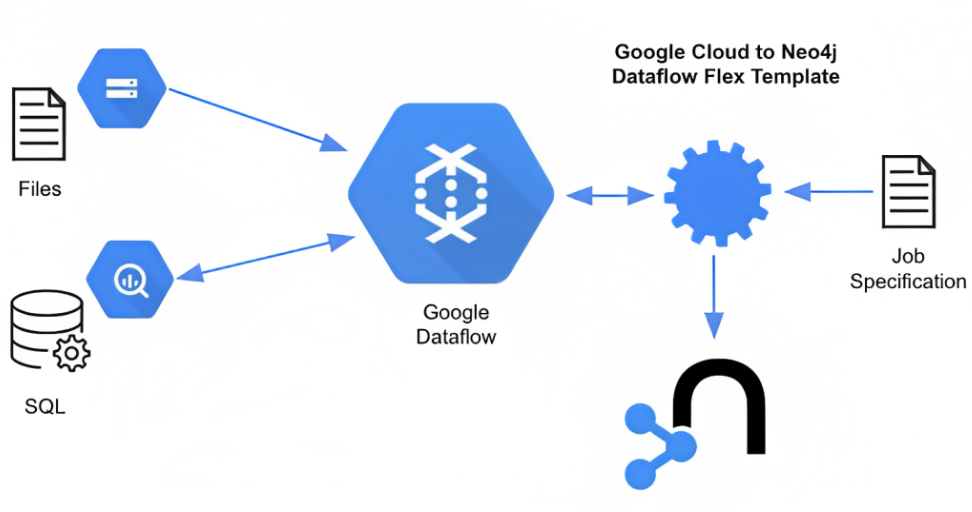
   Google Cloud Dataflow is a completely made-up, cloud-based information handling administration that permits you to deal with a lot of information equally, making it a reasonable answer for huge information handling. You can create and run intricate data processing pipelines with this unified programming model and execution engine.

   Architecture –

   Because Dataflow has a distributed architecture, your data processing jobs can run on multiple Google Cloud-managed virtual machines (VMs) or nodes. The Apache Beam programming model, which is an open-source API for defining and carrying out data processing workflows, serves as the foundation for the Dataflow architecture.

   - **Pipelines:** The Dataflow components known as pipelines enable you to define your data processing workflows. A data processing pipeline is made up of a series of processing steps or transforms that can be chained together.
   - **Transforms:** The processing steps that are applied to your data as it moves through the pipeline are called transforms. Transformations can be straightforward operations like filtering or mapping data, or they can be more complicated operations like aggregating data or joining multiple datasets.
   - **Runners:** Your Dataflow pipeline is run by runners, which are the components. Two runners are supported by Google Cloud Dataflow: the Apache Flink Runner and the Dataflow Runner. The default runner for Dataflow is the Google Cloud-optimized Dataflow Runner.
   - **Workers:** Laborers are the virtual machines (VMs) that run your information handling position. Google Cloud manages and automatically provides workers, which can be scaled up or down depending on the amount of data being processed.
   - **Storage:** Dataflow pipelines are able to read and write data from a number of different types of storage, such as Google Cloud Storage, Google Cloud Bigtable, Google Cloud Pub/Sub, and storage systems provided by third parties.

- With Dataflow, you can concentrate on defining your workflows for processing data while letting Google Cloud manage the infrastructure underneath.



Features –

- **Unified Batch and Streaming:** Google Cloud Dataflow provides a unified programming model for both batch and streaming data processing, allowing you to write your pipelines once and run them in either mode.
- **Automatic Resource Management:** Google Cloud Dataflow automatically manages the computing resources required to process your data. It can dynamically allocate and deallocate resources based on the processing needs of your pipeline.
- **Serverless Computing:** Google Cloud Dataflow provides serverless computing, which means that you don't have to manage any infrastructure. Google Cloud Dataflow automatically provisions and scales computing resources as needed, allowing you to focus on your data processing pipeline.
- **Flexible Windowing:** Google Cloud Dataflow provides flexible windowing for stream processing, allowing you to define windows of data based on time, session, or custom criteria. This allows you to aggregate data and perform calculations on sliding windows of data.
- **Integration with Google Cloud Platform:** Google Cloud Dataflow integrates with other services in the Google Cloud Platform, such as Google Cloud Storage, Google BigQuery, and Google Cloud Pub/Sub, allowing you to easily build end-to-end data processing pipelines.
- **Monitoring and Debugging:** Google Cloud Dataflow provides monitoring and debugging tools that allow you to visualize your pipeline's progress, identify bottlenecks, and troubleshoot issues.

Strengths –

- **Brought together programming model:** You can write data processing code in a single language using Dataflow's unified programming model, which is based on Apache Beam. You can then run that code on a variety of execution engines.

- **Scalability:** Dataflow can be scaled up or down as needed to handle large datasets. It can also parallelize processing across multiple nodes and automatically partition data to improve performance.
- **Fully-managed:** Since Dataflow is a fully managed service, Google Cloud is in charge of the network, storage, virtual machines, and underlying infrastructure. Instead of managing infrastructure, you can concentrate on developing your data processing logic because of this.
- **Fault-tolerance:** Dataflow is built to be fault-tolerant, which means that it can automatically recover from processing node failures or errors, preventing data loss and allowing processing to continue without interruption.
- **Sinks and sources of data:** Dataflow is able to read and write data from a variety of sinks and sources, including BigQuery, Pub/Sub, Google Cloud Storage, and others.
- **Logging and monitoring:** You can monitor the progress of your data processing jobs, spot errors, and resolve problems with Dataflow's real-time monitoring and logging.
- **Cost-effective:** Pricing for Dataflow is based on the resources used by your job, so it is designed to be affordable. Google Cloud handles scaling and management automatically to optimize cost, so you only pay for the resources you use.

In general, Google Cloud Dataflow is a powerful and adaptable platform for processing large amounts of data. It offers a scalable, fault-tolerant, and fully managed service that can assist you in streamlining your data processing workflows and extracting insights from your data.

Weaknesses –

- **Support in limited languages:** The programming model for Dataflow is based on Apache Beam, which supports fewer languages than other data processing frameworks. Dataflow currently only supports Python and Java, which may be a drawback for some users.
- **Inadequate integration:** Although Dataflow is compatible with a number of Google Cloud services, it may not be as compatible as other data processing frameworks with other third-party services.
- **Expectation to learn and adapt:** Due to its distinctive approach to data processing, the Dataflow programming model may have a steeper learning curve than other data processing frameworks.
- **Support for machine learning is limited:** Even though Dataflow is able to process a great deal of data, it might not be the best option for machine learning workloads that call for more complex models or hardware that is tailored to a specific purpose.

Use Cases –

Use Case	Description	Data Source	Data Sink	Benefit
Real-time data processing	Dataflow can process data in real-time, making it an ideal solution for use cases such as streaming analytics, fraud detection, and monitoring.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards or alerts, databases, or data warehouses.	Real-time insights, faster decision-making, and reduced response time.
Batch data processing	Dataflow can handle large amounts of data and process it in parallel, making it an ideal solution for batch processing use cases such as ETL, data cleansing, and data integration.	Large data sets stored in cloud storage, databases, or other data sources.	Cloud storage, databases, or data warehouses.	Scalability, cost-effectiveness, and faster processing times.
Machine learning	Dataflow can be used for machine learning pipelines such as data pre-processing, feature extraction, and model training.	Large data sets stored in cloud storage, databases, or other data sources.	Machine learning models, cloud storage, or databases.	Faster model training, more accurate predictions, and reduced costs.

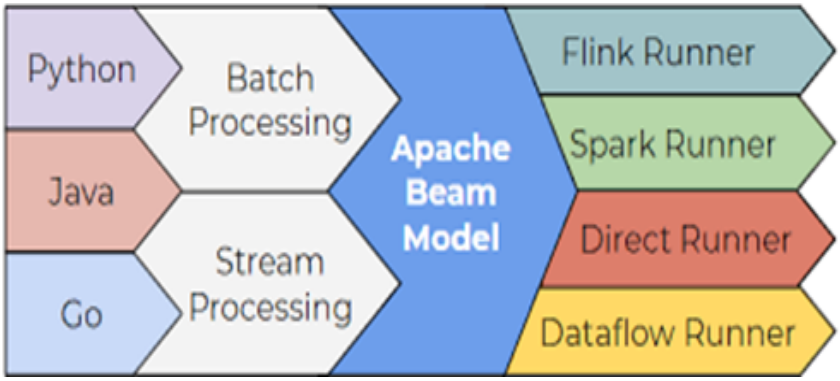
IoT Analytics	Dataflow can be used to process data from IoT devices, extract insights, and trigger actions in real-time.	Streaming data from IoT devices.	Real-time dashboards, alerts, or trigger actions in other cloud services.	Real-time insights, proactive maintenance, and reduced downtime.
Clickstream analysis	Dataflow can be used to process large amounts of clickstream data, extract insights, and personalize user experiences.	Clickstream data from websites or mobile apps.	Real-time dashboards, databases, or data warehouses.	Personalized user experiences, targeted advertising, and improved customer retention.

6. **Apache Beam**

   Apache Beam is an open-source unified programming model for data processing pipelines(49). It provides a set of language-agnostic APIs and a portable execution framework for building batch and streaming data processing pipelines that can run on various execution engines such as Apache Flink, Apache Spark, and Google Cloud Dataflow. Beam allows developers to write data processing pipelines in a variety of programming languages, including Java, Python, Go, and others. Since Beam provides a portable execution framework, developers can write pipelines once and run them on multiple execution engines without modifying the code.

   Architecture –

   - **Pipeline:** A set of data processing steps that are carried out in a predetermined order is referred to as a pipeline.
   - **Transform:** A pipeline operation or processing step is a PTransform. It produces one or more outputs from one or more inputs.
   - **Runner:** In a specific execution environment, like Apache Flink, Apache Spark, or Google Cloud Dataflow, a runner is in charge of running a pipeline.
   - **Sinks and sources of data:** The points at which data enters and exits a pipeline are referred to as data sources and sinks. Files, databases, message queues, and other data sources can all be examples of these.
   - In general, the architecture of Apache Beam is intended to offer a framework that is adaptable and scalable for creating and running data processing pipelines that are compatible with a wide range of execution environments.



Features(50) –

- **Unified Programming Model:** Apache Beam provides a single programming model for both batch and stream processing. This allows developers to write their pipelines once and run them on multiple execution engines, without having to rewrite the code.
- **Portable:** Apache Beam is designed to be portable across different execution engines, making it easy to switch between them. This also makes it possible to run the same pipeline on different cloud providers.
- **Scalable:** Apache Beam is built to handle large-scale data processing, and can scale to handle very large data sets. It does this by partitioning the data and distributing it across multiple nodes in a cluster.
- **Extensible:** Apache Beam is highly extensible, and allows developers to easily add new data sources, data sinks, and transformations to their pipelines.
- **Language independent:** Apache Beam supports multiple programming languages, including Java, Python, and Go. This allows developers to use the language they are most comfortable with.

Strengths –

- **Portability:** Beam pipelines can be run on a variety of execution engines, including Google Cloud Dataflow, Apache Spark, and others.
- **Efficacy of Use:** Beam makes it simpler for developers to construct and maintain pipelines by providing a high-level API that abstracts away many of the low-level details of distributed data processing.
- **Extensibility:** Beam has a flexible programming model that makes it simple for developers to integrate with other systems and tools and add their own custom functionality.

Weaknesses –

- **Curve of learning:** Even though the Beam API is meant to be simple to use, developers who are new to distributed data processing and the Beam programming model still need to learn how to use it.
- **Performance:** Even though Beam is very adaptable and portable, it might not always perform the best for some use cases. Particularly, Beam may not support some data processing tasks because they are better suited to particular execution engines.

- **Asset necessities:** Beam pipelines can use a lot of computing power, especially when processing a lot of data. This can prompt higher framework expenses and longer handling times, which might be a worry for certain associations.

Use Cases –

- **Concentrate, Change, and Burden, or ETL:** Data can be loaded into a target system, transformed into the desired format, and extracted from a variety of sources using Beam.
- **Streams of Processing:** Real-time data streams from financial markets, social media platforms, and Internet of Things (IoT) devices can all be processed by Beam.
- **Bulk processing:** In bunch mode, Shaft is able to handle massive amounts of data, such as those generated by AI applications, information mining, or warehousing.
- **Data Fuse:** Beam can integrate data from a variety of systems and sources, including but not limited to message queues, web services, file systems, and databases.

Use Case	Description	Data Source	Data Sink	Benefit
Data pipeline orchestration	Beam can be used for orchestrating data pipelines, enabling businesses to easily design, implement, and manage data processing pipelines.	Data from various sources.	Other systems, databases, or data warehouses.	Improved data processing efficiency, better scalability, and faster time-to-market.
Batch processing	Beam can be used for batch processing, enabling businesses to process large amounts of data in batch mode.	Data from various sources.	Other systems, databases, or data warehouses.	Improved data processing efficiency, better scalability, and reduced costs.

Stream processing	Beam can be used for stream processing, enabling businesses to process real-time data streams.	Streaming data from IoT devices, social media, or sensors.	Real-time dashboards or alerts, databases, or data warehouses.	Real-time insights, faster decision-making, and reduced response time.
ETL processing	Beam can be used for ETL processing, enabling businesses to extract data from various sources, transform it, and load it into other systems.	Data from various sources.	Other systems, databases, or data warehouses.	Improved data quality, better integration, and reduced costs.
Machine learning	Beam can be used for machine learning pipelines such as data pre-processing, feature extraction, and model training.	Large data sets stored in Beam-compatible formats.	Machine learning models, data warehouses, or other systems.	Faster model training, more accurate predictions, and reduced costs.

OTHER REAL-TIME DATA ANALYSIS TOOLS

1. Apache Flume: It is a distributed, reliable, and available software for efficiently collecting log data present in log files from web servers, aggregating it in HDFS for analy-sis, and moving large amounts of log data into HDFS. It has a simple and flexible architecture based on streaming data flows(51). It is robust and fault tolerant with tunable reli-ability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

2. Apache Kafka: It's an open-source distributed stream-processing software platform that is used to publish and subscribe to streams of records. It is horizontally scalable,fault-tolerant, wicked fast, and runs in production in thousands of companies. Kafka is designed to allow your apps to process records as they occur(52).

3. Apache S4: It's a general-purpose, near real-time,distributed, decentralized,scalable,event-driven, modular plat-form that allows programmers to easily implement applications for processing continuous unbounded streams of data. S4 has a decentralized and symmetric architecture in which all the nodes in a cluster are identical, different to the classic master-nodes architecture. S4 employs ZooKeeper as the communication layer to coordinate the nodes within the cluster(53).

4. Apache NiFi:is an open-source software project from the Apache Software Foundation designed to automate the flow of data between software systems(54). It is based on the "Niagara Files" software previously developed by the NSA. Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic.

5. Apache Hive(55): It's a data warehouse software project built on top of Apache Hadoop for providing data query and analysis. It facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.

6. Apache Hadoop: It's a command-line interface application designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases(56).

7. Apache HBase: is the Hadoop database, a distributed,scalable, big data store. It is used when you need random,realtime read/write access to your Big Data. This project's goal is the hosting of very large tables -- billions of rows X millions of columns -- atop clusters of commodity hardware(57). HBase is an open-source non-relational distributed database modeled after Google's Bigtable and written in Java.

8. HStreaming: It's an analytics platform built on top of Hadoop and MapReduce. The architecture of HStreaming consists of two components: data acquisition and data analytics(58). The data acquisition component is able to collect data in near real-time, and has ETL capabilities, while the analytics component allows to analyze unstructured and structured data on HDFS in a real-time fashion.

9. Apache Impala: It's a modern, open source massively parallel processing(59), distributed SQL query engine for data stored in a computer cluster running Apache Hadoop.

COMPARATIVE ANALYSIS OF ON PREMISE TOOLS

Factor	Apache Beam	Kafka Streams	Apache Storm	Apache Spark
Architecture	Unified	Stream-based	Real-time	Distributed
Latency	Medium	Low	Low	Low
Throughput	High	High	High	High
Scalability	High	High	Medium	High
Fault Tolerance	Yes	Yes	Yes	Yes
APIs/SDKs	Java, Python, Go	Java	Java	Scala, Java, Python
Processing Model	Batch and Streaming	Publish-Subscribe	Stream Processing	Batch and Streaming
Developer Launched	2016	2011	2011	2014
Language	Java	Java	Java	Scala, Java, Python, R
Free Version	Yes	Yes	Yes	Yes

Paid Version	No	Yes	No	No
Max Concurrent Connections	N/A	High	N/A	High
Max Data Ingestion Rate	N/A	High	N/A	High
Recovery	Fault-tolerant	Resilient	Robust	Reliable
License	Apache License	Apache License	Apache License	Apache License

Name of system	Architecture	Catalog	Integration/ Query	In-memory	Recovery	License
Hadoop Online	Master/slaves	No	Integration	Yes	No	Apache License 2.0
Storm	Peer	No	Integration	Yes	Manual	Eclipse Public License
Flume	Peer	No	Integration	Yes/ backed by files	Manual	Apache License 2.0
Spark Streaming	Master/slaves	No	Both	Yes	Parallel	Apache License 2.0
Kafka	publish/subscribe	No	Integration	Yes/ backed by files	Yes	Apache License 2.0
Scribe	Master/ slaves	No	Integration	Yes/ backed by files	Yes	MIT License

S4	Peer	No	Integration	Yes	Manual	Apache License 2.0
HStreaming	Master/slaves	No	Both	Semi-in-memory	Yes	Community
All-RiTE	Peer	No	Both	In-memory	Yes	GPL 2.0
Impala	Peer	Yes	Analytics query	Yes	Yes	Apache License 2.0

CHALLENGES AND RESEARCH SCOPE

Throughout the process of comparing and employing different live data streaming tools one can face many challenges as there are a wide range of architectures,features, tools, applications, models and plenty other factors that need to be carefully reviewed,analyzed and evaluated in all possible circumstances. Below are the few challenges –

- **Selection of tools:** There are numerous live data streaming tools available in the market, each with its own set of features and capabilities. Choosing the right set of tools to compare was a bit challenging.
- **Comparison criteria:** Defining the criteria for comparison is also a challenging task. Our goal was that the criteria should be relevant, meaningful, and cover all the important aspects of live data streaming tools.
- **Performance evaluation:** Measuring the performance of live data streaming tools can be complex, and often depends on various factors such as data volume, data velocity, and hardware configurations.
- **Benchmarking:** There is a lack of standard benchmarking methodologies for evaluating the performance of live data streaming tools, which can make it difficult to compare different tools accurately.
- **Smartness:**Applications for live data streaming must be intelligent enough to analyze the incoming data and make snap judgements in real-time. Effective algorithms and machine learning models that can learn from the data and adjust to changing circumstances are needed for this.As an illustration, Apache Flink enables users to create intelligent and adaptable streaming applications by supporting stateful stream processing and machine learning libraries like FlinkML and TensorFlow.
- **Scalable:**Applications for live data streaming must be scalable in order to manage enormous data volumes and support numerous concurrent users. A distributed architecture that can grow horizontally as necessary is necessary for this.By adding additional brokers to the cluster, Apache Kafka's distributed architecture may extend horizontally to accommodate more users and data.

- **Troubleshooting:**Applications that broadcast live data may encounter a number of challenges, including data loss, network connection issues, and device failures. Real-time troubleshooting of these problems can be difficult and calls for powerful monitoring and debugging tools.Users may track the state of their streaming topologies and conduct real-time problem-solving with Apache Storm's Storm UI, a UI-based monitoring and debugging tool.
- **Monitoring:**Applications that transmit live data must be closely watched to make sure everything is operating as it should and to spot any problems before they get out of hand. A monitoring system that can deliver measurements and alarms in real-time is necessary for this. Users may keep tabs on the efficiency of their stream processing processes with Apache Samza's monitoring system, Samza Metrics, and set up alarms for certain occurrences.
- **Integration:**Applications that stream live data must interface with many different data sources and systems, such as databases, data warehouses, and other programmes. Support for a range of data types and protocols is necessary for this.Users may combine their streaming applications with a variety of data sources thanks to Apache Spark Streaming's support for a number of data sources, including Kafka, Flume, HDFS, and Amazon Kinesis.
- **Security:**Applications used to broadcast live data must be safe and guard against hacking or unauthorized access to the data. Support for secure data transfer and authentication is necessary for this.As an illustration, Apache Flink supports Kerberos-based authentication and SSL/TLS-based secure data transport.
- **Master Dashboard:**A centralized dashboard that offers a real-time view of the complete system, including metrics, logs, and alarms, is necessary for live data streaming applications. A dashboard that can combine data from several sources and present an in-depth picture of the system is necessary for this.An illustration of this is the web-based dashboard offered by Apache NiFi, which enables customers to track the progress of their data flows and observe real-time metrics and alarms.
- **Programming Model:**Applications for live data streaming must offer effective and scalable programming paradigms that let users create clear and succinct complicated data processing algorithms. Support for declarative programming, pipeline processing, and pattern matching are necessary for this.Using stateful processing, declarative programming, pipeline processing, and complex pattern matching are all supported by the unified batch and stream processing paradigm offered by Apache Beam.
- **Declarative Programming:**Declarative programming can make writing complicated data processing logic simpler, but it might be difficult to apply in live data streaming systems. Creating a declarative programming language or API that is easy to learn and use for developers of all skill levels is a problem. Another difficulty is ensuring that the declarative logic is executed effectively and does not cause performance snags or resource use problems. Last but not least, enabling complicated data types and data manipulations that may not be easily expressed in a declarative language is a further challenge.
- **Pipeline Processing:**Greater scalability, fault tolerance, and parallelism are made possible by pipeline processing, but it can also provide a unique set of difficulties. Making sure that every stage of the pipeline is executed effectively and does not cause performance bottlenecks or resource utilization problems is one difficulty. Another difficulty is making sure the pipeline is fault-tolerant and can manage faults at any point in the processing. The last problem is

supporting complicated data relationships and data transformations that may need bespoke logic or extensions.

- **Extending the comparison criteria:** The criteria for evaluating live data streaming tools could be extended to include additional aspects such as security, ease of use, and integration with other tools and systems.
- **Exploring hybrid solutions:** Hybrid solutions that combine on-premise and cloud-based live data streaming technologies could be evaluated to determine their effectiveness and potential benefits.
- **Investigating the impact of machine learning:** As machine learning becomes increasingly popular in data processing, there is a need to investigate its impact on live data streaming and how it can be integrated into existing live data streaming tools.
- **Examining the impact of emerging technologies:** Emerging technologies such as edge computing, 5G networks, and quantum computing could be investigated to determine their impact on live data streaming and how they can be integrated into live data streaming tools.

CONCLUSION

Live data streaming is an essential component of modern data processing and analysis, and there are several tools available to support this capability. In this research article, we have compared six popular live data streaming tools, namely Amazon Kinesis, Apache Storm, Apache Beam, Google Cloud Dataflow, Apache Flink, and Spark Streaming. Our comparison reveals that each tool has its own unique strengths and weaknesses, and no single tool is the best fit for all use cases.

Selecting the most suitable and efficient architecture and configuration is a challenging task that requires careful consideration of the use case and technology. Furthermore, the verification and validation process is complex and necessitates extensive knowledge about the technology and use case. However, time constraints and resource availability often make this task difficult or unachievable. Thus, the cost-efficiency,systemwide-complexity and other quality-constraint matrix trade-offs play a vital role in the overall success of the use case.

Our analysis reveals that Amazon Kinesis is a powerful and scalable tool for real-time data processing and streaming, while Apache Storm offers low latency and high throughput for big data processing. Apache Beam provides a unified programming model that enables developers to write batch and streaming jobs with ease. Google Cloud Dataflow offers auto-scaling and managed services for seamless integration with other Google Cloud services. Spark Streaming provides integration with the popular Spark ecosystem and supports both batch and streaming processing.

By carefully evaluating these tools and understanding their strengths and weaknesses, organizations and users can make informed decisions and implement the best tool or combination of tools for their needs. In conclusion, selecting the right live data streaming tool is critical for the success of the data processing and analysis project, and organizations should carefully evaluate the available options to make the best informed decision.

REFERENCES

1.Gupta, A., Singh, R. K., & Mangla, S. K. (2022). Evaluation of logistics providers for sustainable service quality: Analytics based decision making framework. *Annals of Operations Research*, *315*(2), 1617–1664. https://doi.org/10.1007/s10479-020-03913-0

2.S. Tripathi and B. K. Chaurasia, "Broker Clustering Enabled Lightweight Communication in IoT using MQTT," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023, pp. 1-6, doi: 10.1109/ISCON57294.2023.10112105.

3.Yang Wang, Yutong Li, Ting Wang, Gang Liu, Towards an energy-efficient Data Center Network based on deep reinforcement learning,Computer Networks,Volume 210,2022,108939,ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2022.108939.

4.Kim, S. H., Lee, C., & Youn, C. H. (2020). An accelerated edge cloud system for energy data stream processing based on adaptive incremental deep learning scheme. *IEEE Access*, *8*, 195341–195358. https://doi.org/10.1109/ACCESS.2020.3033771

5.Zhang L, Stoffel A, Behrisch M, Mittelstadt S, Schreck T, Pompl R, et
al. Visual analytics for the big data era—A comparative review of stateof-the-art commercial systems. In2012 IEEE Conference on Visual Analytics Science and Technology (VAST) 2012: 173-182

6.Cao, M. D., Ganesamoorthy, D., Elliott, A. G., Zhang, H., Cooper, M. A., & Coin, L. J. M. (2016). Streaming algorithms for identification of pathogens and antibiotic resistance potential from real-time MinION™ sequencing. *GigaScience*, *5*(1). https://doi.org/10.1186/s13742-016-0137-2

7.Ulak, M. B., Ozguven, E. E., Moses, R., Sando, T., Boot, W., AbdelRazig, Y., & Sobanjo, J. O. (2019). Assessment of traffic performance measures and safety based on driver age and experience: A microsimulation based analysis for an unsignalized T-intersection. *Journal of Traffic and Transportation Engineering (English Edition)*, *6*(5), 455–469. https://doi.org/10.1016/j.jtte.2018.05.004

8.Lai, C.-H. (2007). Understanding the Design of Mobile Social Networking. *M/C Journal*, *10*(1). https://doi.org/10.5204/mcj.2607

9.Lee, A. (2021). In the Shadow of Platforms. *M/C Journal*, *24*(2). https://doi.org/10.5204/mcj.2750

10.Shehhi, S. S., & Al Maflahi, M. A. (2021). Eyes on Air. Society of Petroleum Engineers (SPE). https://doi.org/10.2118/207455-ms

11.Mehmood, E., & Anees, T. (2020). Challenges and Solutions for Processing Real-Time Big Data Stream: A Systematic Literature Review. *IEEE Access*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2020.3005268

12.Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F., & Khan, S. (2019). A survey of distributed data stream processing frameworks. *IEEE Access*, *7*, 154300–154316. https://doi.org/10.1109/ACCESS.2019.2946884

13.Wei, J., Miao, M., Tian, G., Shen, J., Chen, X., & Susilo, W. (2023). Optimal Verifiable Data Streaming Under Concurrent Queries. *IEEE Transactions on Mobile Computing*. https://doi.org/10.1109/TMC.2023.3309270

14.Bhegade, N., Kanase, S., Kashetwar, S., & Chavan, V. (2021). Survey Paper on Various Techniques of Privacy Preservation in Data Stream Mining. *Academia.Edu*, *10*(01), 692–696. Retrieved from https://www.academia.edu/download/65994399/survey_paper_on_various_techniques_of_IJERTV10IS010122.pdf

15.Shipman, G., Campbell, S., Dillow, D., Doucet, M., Kohl, J., Granroth, G., … Taylor, R. (2014). Accelerating data acquisition, reduction, and analysis at the spallation neutron source. In *Proceedings - 2014 IEEE 10th International Conference on eScience, eScience 2014* (Vol. 1, pp. 223–230). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/eScience.2014.31
16.Nizam, H., Zafar, S., Lv, Z., Wang, F., & Hu, X. (2022). Real-Time Deep Anomaly Detection Framework for Multivariate Time-Series Data in Industrial IoT. *IEEE Sensors Journal*, *22*(23), 22836–22849. https://doi.org/10.1109/JSEN.2022.3211874
17.Mekuria, R. (2017). Network streaming and compression for mixed reality tele-immersion. *ACM SIGMultimedia Records*, *8*(4), 1. https://doi.org/10.1145/3129151.3129159
18.Tao, X., Yang, Y., Xu, M., Duan, Y., Huang, D., & Liu, W. (2021). Quality of experience oriented multimedia computing communications. *Journal of Image and Graphics*, *26*(6), 1201–1215. https://doi.org/10.11834/jig.200864
19. Ara, A., & Ara, A. (2018). Case study: Integrating IoT, streaming analytics and machine learning to improve intelligent diabetes management system. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing, ICECDS 2017* (pp. 3179–3182). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICECDS.2017.8390043
20.Taranov, K., Byan, S., Marathe, V., & Hoefler, T. (2022). KafkaDirect: Zero-copy Data Access for Apache Kafka over RDMA Networks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 2191–2204). Association for Computing Machinery. https://doi.org/10.1145/3514221.3526056
21.Pandya, A., Kostakos, P., Mehmood, H., Cortes, M., Gilman, E., Oussalah, M., & Pirttikangas, S. (2019). Privacy preserving sentiment analysis on multiple edge data streams with Apache NiFi. In *Proceedings of the 2019 European Intelligence and Security Informatics Conference, EISIC 2019* (pp. 130–133). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/EISIC49498.2019.9108851
22.Asaithambi, S. P. R., Venkatraman, S., & Venkatraman, R. (2021). Big data and personalisation for non-intrusive smart home automation. *Big Data and Cognitive Computing*, *5*(1), 1–21. https://doi.org/10.3390/bdcc5010006
23.Biem, A., Bouillet, E., Feng, H., Ranganathan, A., Riabov, A., Verscheure, O., … Moran, C. (2010). IBM InfoSphere Streams for scalable, real-time, intelligent transportation services. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 1093–1103). https://doi.org/10.1145/1807167.1807291
24.Evermann, J., Rehse, J. R., & Fettke, P. (2016). Process discovery from event stream data in the cloud - A scalable, distributed implementation of the flexible heuristics miner on the amazon kinesis cloud infrastructure. In *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom* (Vol. 0, pp. 645–652). IEEE Computer Society. https://doi.org/10.1109/CloudCom.2016.0111
25.Zdravevski, E., Lameski, P., Apanowicz, C., & Ślęzak, D. (2020). From Big Data to business analytics: The case study of churn prediction. *Applied Soft Computing Journal*, *90*. https://doi.org/10.1016/j.asoc.2020.106164
26.Zdravevski, E., Lameski, P., Apanowicz, C., & Ślęzak, D. (2020). From Big Data to business analytics: The case study of churn prediction. *Applied Soft Computing Journal*, *90*. https://doi.org/10.1016/j.asoc.2020.106164
27.Gupta, A., Dhanda, N., & Gupta, K. K. (2023). Ingest and Visualize CSV Files using AWS Platform for Transition from Unstructured to Structured Data. In *International Conference on Emerging Trends in Engineering and Technology, ICETET* (Vol. 2023-April). IEEE Computer Society. https://doi.org/10.1109/ICETET-SIP58143.2023.10151634
28.Spiegelberg, L., Yesantharao, R., Schwarzkopf, M., & Kraska, T. (2021). Tuplex: Data Science in Python at Native Code Speed. In *Proceedings of the ACM SIGMOD International Conference on*

Management of Data (pp. 1718–1731). Association for Computing Machinery. https://doi.org/10.1145/3448016.3457244
29.Boomija, M. D., & Raja, S. V. K. (2023). Securing medical data by role-based user policy with partially homomorphic encryption in AWS cloud. *Soft Computing*, *27*(1), 559–568. https://doi.org/10.1007/s00500-022-06950-y
30.Salah, K., Calyam, P., & Boutaba, R. (2017). Analytical Model for Elastic Scaling of Cloud-Based Firewalls. *IEEE Transactions on Network and Service Management*, *14*(1), 136–146. https://doi.org/10.1109/TNSM.2016.2640297
31.Shree, R., Choudhury, T., Gupta, S. C., & Kumar, P. (2018). KAFKA: The modern platform for data management and analysis in big data domain. In *2nd International Conference on Telecommunication and Networks, TEL-NET 2017* (Vol. 2018-January, pp. 1–5). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/TEL-NET.2017.8343593
32.Shaheen, J. A. (2017). Apache Kafka: Real Time Implementation with Kafka Architecture Review. *International Journal of Advanced Science and Technology*, *109*, 35–42. https://doi.org/10.14257/ijast.2017.109.04
33.Riadsolh, A., Lasri, I., & Elbelkacemi, M. (2020). Cloud-based sentiment analysis for measuring customer satisfaction in the moroccan banking sector using naïve bayes and stanford nlp. *Journal of Automation, Mobile Robotics and Intelligent Systems*, *14*(4), 64–71. https://doi.org/10.14313/JAMRIS/4-2020/47
34.Isah, H., & Zulkernine, F. (2019). A Scalable and Robust Framework for Data Stream Ingestion. In *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018* (pp. 2900–2905). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/BigData.2018.8622360
35. Shaheen, J. A. (2017). Apache Kafka: Real Time Implementation with Kafka Architecture Review. *International Journal of Advanced Science and Technology*, *109*, 35–42. https://doi.org/10.14257/ijast.2017.109.04
36.Bimal Patel, Parth Shah,Operating system support, protocol stack with key concerns and testbed facilities for IoT: A case study perspective,Journal of King Saud University - Computer and Information Sciences,Volume 34, Issue 8, Part A,2022,Pages 5420-5434,ISSN 1319-1578,https://doi.org/10.1016/j.jksuci.2021.01.002
37.Komisarek, M., Pawlicki, M., Kozik, R., & Choraś, M. (2021). Machine learning based approach to anomaly and cyberattack detection in streamed network traffic data. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, *12*(1), 3–19. https://doi.org/10.22667/JOWUA.2021.03.31.003
38.Wu, H., Shang, Z., & Wolter, K. (2020). Learning to Reliably Deliver Streaming Data with Apache Kafka. In *Proceedings - 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020* (pp. 564–571). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/DSN48063.2020.00068
39.Wu, H., Shang, Z., & Wolter, K. (2019). Trak: A testing tool for studying the reliability of data delivery in apache kafka. In *Proceedings - 2019 IEEE 30th International Symposium on Software Reliability Engineering Workshops, ISSREW 2019* (pp. 394–397). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ISSREW.2019.00101
40.Iyengar, A., & Portilla, I. (2020). Analytics at the Edge. Retrieved from https://www.ibm.com/cloud/blog/analytics-at-the-edge
41.Kumar, Manish., & Singh, Chanchal. (2017). *Building Data Streaming Applications with Apache Kafka*. *Packt* (p. 269). Packt Publishing.
42.Lampa, S., Dahlö, M., Alvarsson, J., & Spjuth, O. (2019). Scipipe: A workflow library for agile development of complex and dynamic bioinformatics pipelines. *GigaScience*, *8*(5). https://doi.org/10.1093/gigascience/giz044

43.Kim, Y., Son, S., & Moon, Y. S. (2019). SPMgr: Dynamic workflow manager for sampling and filtering data streams over Apache Storm. *International Journal of Distributed Sensor Networks*, *15*(7). https://doi.org/10.1177/1550147719862206
44.Bifet, A., Maniu, S., Qian, J., Tian, G., He, C., & Fan, W. (2016). StreamDM: Advanced Data Mining in Spark Streaming. In *Proceedings - 15th IEEE International Conference on Data Mining Workshop, ICDMW 2015* (pp. 1608–1611). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICDMW.2015.140
45.Zheng, L. H., Li, M. Z., & Sun, H. (2009). Development of an analyzing system for soil parameters based on NIR spectroscopy. *Guang Pu Xue Yu Guang Pu Fen Xi/Spectroscopy and Spectral Analysis*, *29*(10), 2633–2636. https://doi.org/10.3964/j.issn.1000-0593(2009)10-2633-04
46.Ed-daoudy, A., & Maalmi, K. (2019). A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0271-7
47. Isah, H., Abughofa, T., Mahfuz, S., Ajerla, D., Zulkernine, F., & Khan, S. (2019). A survey of distributed data stream processing frameworks. *IEEE Access*, *7*, 154300–154316. https://doi.org/10.1109/ACCESS.2019.2946884
48.Zhang, W., He, Z., Du, B., Luo, M., & Zheng, Q. (2019). Deploying external bandwidth guaranteed media server clusters for real-time live streaming in media cloud. *PLoS ONE*, *14*(4). https://doi.org/10.1371/journal.pone.0214809
49.The Apache Software Foundation. (2018). Design Your Pipeline. *Apache Beam Documentation*.
50.Begoli, E., Hyde, J., Akidau, T., Knight, K., Hueske, F., & Knowles, K. (2019). One SQL to rule them all - An efficient and syntactically idiomatic approach to management of streams and tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 1757–1772). Association for Computing Machinery. https://doi.org/10.1145/3299869.3314040
51.Birjali, M., Beni-Hssane, A., & Erritali, M. (2017). Analyzing Social Media through Big Data using InfoSphere BigInsights and Apache Flume. In *Procedia Computer Science* (Vol. 113, pp. 280–285). Elsevier B.V. https://doi.org/10.1016/j.procs.2017.08.299
52.Akanbi, A., & Masinde, M. (2020). A distributed stream processing middleware framework for real-time analysis of heterogeneous data on big data platform: Case of environmental monitoring. *Sensors (Switzerland)*, *20*(11), 1–25. https://doi.org/10.3390/s20113166
53.Simoncelli, D., Gringoli, F., Dusi, M., & Niccolini, S. (2013). Stream-monitoring with blockmon: Convergence of network measurements and data analytics platforms? *Computer Communication Review*, *43*(2), 30–35.
54.Kim, S. S., Lee, W. R., & Go, J. H. (2019). A Study on Utilization of Spatial Information in Heterogeneous System Based on Apache NiFi. In *ICTC 2019 - 10th International Conference on ICT Convergence: ICT Convergence Leading the Autonomous Future* (pp. 1117–1119). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICTC46691.2019.8939734
55.Liu, H., Tang, B., Zhang, J., Deng, Y., Zheng, X., Shen, Q., … Luo, Z. (2022). GHive: A Demonstration of GPU-Accelerated Query Processing in Apache Hive. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 2417–2420). Association for Computing Machinery. https://doi.org/10.1145/3514221.3520166
56.The Apache Software Foundation. (2009). *Sqoop. The Apache Software Foundation*.
57.Tsai, C. P., Chang, C. W., Hsiao, H. C., & Shen, H. (2022). The Time Machine in Columnar NoSQL Databases: The Case of Apache HBase. *Future Internet*, *14*(3). https://doi.org/10.3390/fi14030092
58.Li, F., Özsu, M. T., Chen, G., & Ooi, B. C. (2014). R-Store: A scalable distributed system for supporting real-time analytics. In *Proceedings - International Conference on Data Engineering* (pp. 40–51). IEEE Computer Society. https://doi.org/10.1109/ICDE.2014.6816638
59.Yang, C. T., Chen, T. Y., Kristiani, E., & Wu, S. F. (2021). The implementation of data storage and analytics platform for big data lake of electricity usage with spark. *Journal of Supercomputing*, *77*(6), 5934–5959. https://doi.org/10.1007/s11227-020-03505-6