Apache Kafka – Your Event Stream Processing Solution

HTC global services

Introduction

Data is one among the newer ingredients in the Internet-based systems and includes user-activity events related to logins, page visits, clicks, social networking activities such as likes, sharing, and comments, and operational, and system metrics. In accordance with the new trends in Internet applications, activity data has become a part of production data and is used for real time analytics. Vast volume of data is generated on a “continuous” basis, be it in the form of click-streams, output from sensors or via sharing on popular social website. Real-time usage of these multiple sets of data collected from production systems imposes a challenge due to the volume of data collected and processed

What are Business Events?

Business events are discrete data points. In computing context, an event can be an action or occurrence that is of significance for the enterprise. Business events can be generated or triggered by the system or by the user. It can be part of a business process; for example, issuing a trade order, a sensor reading, the arrival of a transport vehicle, etc. or it can be the monitoring information about IT infrastructure, middleware, applications and business processes, etc. Business events are messages that have the event data as payload. Business events significantly influence all aspects of an enterprise. More information on events is available in Event Processing Systems- the Way to Improved and Quicker Decision Making white paper.

What is a Business Data Feed?

A business data feed is a flow of business data from an external source into persistent storage like a hard disk. It is a logical artifact that comes to life when it is connected to a dataset using connect feed constructs. On establishing connection, the feed is believed to be in the connected state.

Multiple feeds can be simultaneously connected to a dataset so that the contents of the dataset signify the union of the connected feeds. Business data feeds constitute a “plumbing” concept in principle. It is a means for creating a business data flow from external sources that continuously produce data to progressively populate data in a data store. Business data feed management is the job of maintaining the continuous flow of business data.

What is the Nature of Business Data Feed?

Business data feeds serve as the data channels for decision making. Data feeds are ingested for creating business decisions. Data ingestion of business events is normally associated with varying data arrival rates, which initiates varying demand for resources. An easy way of permitting the data being put into a Big Data management system on a constant basis is to have a single program fetch data from an external data source, parse the data, and then invoke an insert statement for every record or batch of records. This solution is restricted to a single machine's computing capability. Ingesting multiple business data feeds actually requires running and managing many individual programs / processes which are not viable. The task of continuously retrieving data from external source(s), applying pre-processing, cleansing data, filtering data, and indexing the processed data therefore requires 'gluing' together different systems. This needs to be addressed with technical solutions, as business decisions are built over multiple interrelated data streams.

Stream processing comes out as a solution to address disparate data feed ingestion requirements. It operates on data with a sliding window, also called as the time window that is usually limited by the data velocity and the memory availability. Stream processing solutions glues together different business systems by integrating their data streams.

What is a Business Feed Adaptor?

As a variety of participating system are to be glued together at their data stream levels, there is a need for feed adaptors. A feed adaptor is an implementation of an interface and its details are specific to a given data source such as SAP ERP systems. A feed adaptor may operate in a push or a pull mode depending on the data transfer requirements and associated APIs offered by the data source. The push mode involves just one initial request by the adaptor to the data source for setting up the connection. Once a connection is authorized, the data source "pushes" data to the adaptor without any subsequent requests by the adaptor. This in contrast, with the pull mode wherein the adaptor makes a separate request each time to receive data. The data streams travel to the data collector system with ease.

What are the Challenges in Designing the Data Feed Facility?

Designing a data feed facility is a challenging task. The data feed facility is built over a variety of data feed channels that are connected with the feed adaptors. The right mix of data feed adaptors will have the data moving through the data channels. The data feed facility calls for a good understanding of the business domain and associated data dependencies which impact the business results. The key challenges involved in building a data feed facility include:

- Genericity and Extensibility: A feed ingestion facility must be generic to work with a variety of data sources and high level applications. A plug-and-play model is preferred to allow extension of the offered functionality
- Fetch-Once, Compute-Many: Multiple applications may wish to consume the ingested data and may wish the arriving data to be processed and persisted differently. It is desirable to receive a single flow of data from an external source and transform it in multiple ways to drive different applications concurrently
- Scalability and Elasticity: Multiple feeds with fluctuating data arrival rates coupled with ad hoc queries over the persisted data imply a varying demand for resources. The system should offer scalability by being able to ingest increasingly large volumes of data through the addition of resources
- Fault Tolerance: Data ingestion is expected to run on a large cluster of commodity hardware that may be prone to hardware failures. It is desirable to offer the desired degree of robustness in handling failures while minimizing data loss

Stream processing has emerged as a data analysis technique to handle realtime applications. This is built over a large amount of data generated in external environments and the solutions are built around frameworks. There are many real time data stream processing frameworks available that are capable of processing large data considerably fast overcoming the single-machine limitations.

What are Data Streams?

Business depends on data. The view of data changes when one thinks about what a business actually does with the data generated in large systems. For example, the retail segment generates orders that lead to sales, shipments, and so on, a financial institution generates orders that have an impact on a stock price. A social network platform generates clicks, impressions, and searches that are used to create a sort of intelligent analysis to further display personalized data to its users. These types of data can be considered as streams of events. The abstraction for data flow is called stream, which is a continuous and unbounded sequence of tuples.

More information on data stream ingestion and complex event processing systems is available in Data Stream Ingestion & Complex Event Processing Systems for Data Driven Decisions white paper.

Stream processing computational paradigm consists of assimilating data readings through business data feeds from collections of software or hardware sensors in stream form, and aggregating them in a data feed facility to analyze the data, and produce actionable results. HTC is working on cutting edge data stream solutions through its research and development division from its global delivery center. There are two types of stream computing mechanisms - Data Stream Computing and Event Stream Computing.

Data and Event Stream Computing

Stream computing is addressed through the new generation solutions. Data stream computing is able to analyze and process data in real time to gain an immediate insight. It is typically applied to the analysis of vast amount of data in real time and to process them at high speeds. Many application scenarios require big data stream computing.

More information on this is available in Data Stream Ingestion & Complex Event Processing Systems for Data Driven Decisions white paper.

Complex Event Processing (CEP) is an emerging network technology that creates actionable, situational knowledge from distributed message-based systems, databases and applications in real time or near real time. CEP can provide organizations the capability to define, manage, and predict events, situations, exceptional conditions, opportunities, and threats in complex, heterogeneous networks. CEP delivers high-speed processing of many events

across all the layers of the organization, identifying the most meaningful events within the event cloud, analyzing their impact, and taking subsequent action in real time. Esper is an Event Stream Processing (ESP) and Event Correlation Engine.

More information on Real-time Data Streaming, refer to Managing Real-time Data Streaming Effectively with CEP Solutions white paper.

What is Latency in Data Streams?

Every business system has a defined latency. End-to-end latency of a streaming system is formally expressed as the duration between the time a data record enters the system and the time the results corresponding to it is generated. Based on the system model, the end to end latency is calculated as the sum of the following terms.

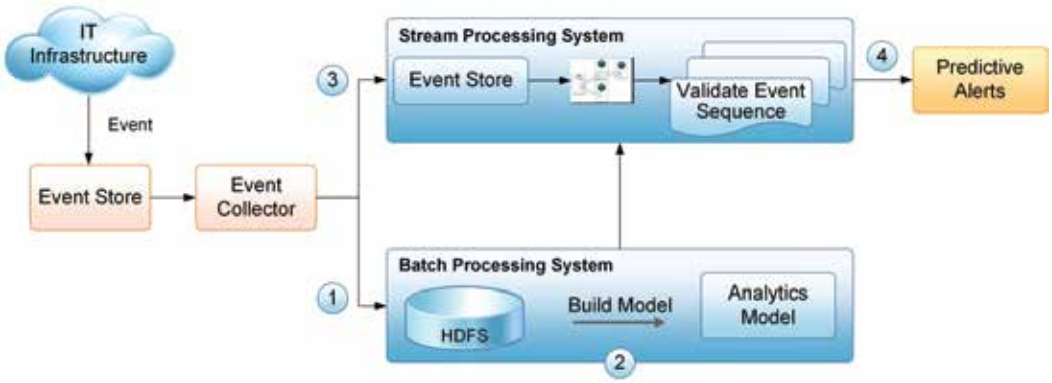
- Batching delay : The duration between the time a data record is received by the system and the time it is sent to the batch queue – this quantity is upper bounded by the corresponding batch's interval
- Queuing delay: The time a batch spends waiting in the batch queue
- Processing time: The processing time of a batch, which is also dependent on the corresponding batch's interval. It is clear that the latency directly depends on a batch interval – lower batch interval leads to lower latency. It is therefore desirable to keep the batch interval as low as possible

Latency can directly impact the end results. Intuitively, a streaming workload can be stable only if the system can process data as fast as the data is being received. When batch processing is considered, the processing time monotonically increases with the batch interval. Longer intervals imply more data to process, and therefore higher processing times. The exact relationship between the intervals and the processing times can be non-linear, as it depends on the characteristics of the processing system, the available resources, the nature of the workload, and the data ingestion rates. Therefore, it is essential to understand that the choice of data ingestion systems are more a design issue.

What is Stream Processing?

Stream processing refers to integration and processing of data before storing the data to a storage medium. A stream processing system is built out of multiple units called Stream Processing Elements (SPE). Each SPE receives input from its input business data queue, performs computation on the input

using its local business rules, and generates output to its output business queue. The success of stream processing depends on the dynamics of the interaction between the SPEs.



What are Stream Processing Solutions?

Data streams consisting of events are not valuable but can be taken advantage of a system that processes these events, produces result, and provides it to other services. Such stream processing elements have been designed as a solution and are made available in the market for easy consumption. Some key systems available are Twitter's Storm, Yahoo's S4 Apache Spark, Google's MillWheel, Apache Samza, Photon, etc.

More information on this is available in Data Stream Ingestion & Complex Event Processing Systems white paper.

What is Apache Kafka?

Apache Kafka is a Stream Processing Element (SPE) taking care of the needs of event processing. Apache Kafka was initially developed at LinkedIn and subsequently released as an open source project with the Apache Software Foundation. It is a fast, scalable system, and distributed in nature by its design. It is used to buffer messages between data stream producing systems, where message is just a simple and independent business data which is like an internal office memo for understanding. Messages are classified into topics which are similar to various files in an office. The generated message streams are committed to specific topics, which are like filing a document in a specific file. Within each topic, the filed messages are guaranteed to be linearly ordered in time, with Kafka providing an index for each message.

Why Apache Kafka and how it is Designed?

Apache Kafka is a distributed publish/subscribe messaging system.

To know more about publish subscribe systems, refer the Publish/Subscribe Solutions for Responding to Events in Real-time white paper.

Apache Kafka is designed with the following characteristics:

- Persistent messaging: To derive the real value from big data, any kind of information loss cannot be afforded. Apache Kafka is designed to provide constant-time performance even with very large volumes of stored messages, which is in order of TB
- High throughput: With big data in mind, Kafka has been designed to work on commodity hardware and to support millions of messages per second
- Distributed: Apache Kafka explicitly supports messages partitioning over Kafka servers and distributing consumption over a cluster of consumer machines
- Multiple client support: Apache Kafka system supports easy integration of clients from different platforms such as Java, .NET, PHP, Ruby, and Python
- Real time: Messages produced by the producer threads should be immediately visible to consumer threads. This feature is critical to event-based systems such as CEP systems

Kafka provides a real-time publish-subscribe solution, which overcomes the challenges of the conventional solution issues like real-time data management issues. Kafka supports parallel data loading in Hadoop systems. It is therefore meaningful to fall back on Apache Kafka for designing big data solutions, address the needs of streaming business events, and CEP.

Why Kafka is called as Publish Subscribe System?

Messaging is the art of moving message across the producer and consumer group. It integrates distributed applications to an overall system by providing message queues for communication between them. One application can publish its messages to the queue and the other application can asynchronously read it from the queue at any time. Message Broker systems persist incoming messages in an enhanced type of a message queue, named topic. Sending messages to the broker in the form of publishing to a specific topic and on the other hand receiving messages only for the specified topic, is called publish / subscribe and this therefore classifies Kafka as a publish subscribe system.

What is a Topic in Kafka?

Kafka provides a high-level abstraction called Topic. Users define a new Topic (file) for each new category of messages (documents) and the messages are published to a category or stream name. A topic allows the message broker to deliver messages to multiple independent consumers.

Kafka has a very simple storage layout. Each partition of a topic (file) corresponds to a logical log. Each time a producer publishes a message to a topic, the broker appends the message to the last segment file. The message is exposed to the consumers after it is flushed.

Who are Producers in Kafka?

Producers are the clients publishing messages to a Topic. Producers are diverse in nature and publish varied messages to different topics. A producer can publish data to a topic of its choice and is responsible for choosing which message will be assigned to which partition within a topic. Kafka producer client is configured to send messages in either a synchronous or asynchronous fashion to the topics. The asynchronous mode allows the client to batch small messages into larger data chunks before sending them over the network.

What is a Message Broker?

A Message Broker is a key element of the solution. It is a dedicated component which decouples the source and target systems by assuming full responsibility for coordinating communication between all the connected nodes. The published messages are stored at a set of servers called Brokers. Each Kafka cluster consists of one or more Brokers. The partitions of a Topic are distributed over the Brokers of the Kafka cluster with each Broker handling data and requests for a share of the partitions. Each partition is replicated across a configurable number of Brokers for fault tolerance. The main tasks of a message broker are the dynamic registration of endpoints, determining the location of a target system, and performing the communication as well as the transformation of a message from one format to another.

A consumer can subscribe to one or more topics from the brokers and consume the subscribed messages by pulling data from the Brokers. The Broker locates the requested message by searching the list and sends the data back to the consumer. After a consumer receives a message it computes the location of the next message to consume and uses it in the next pull request.

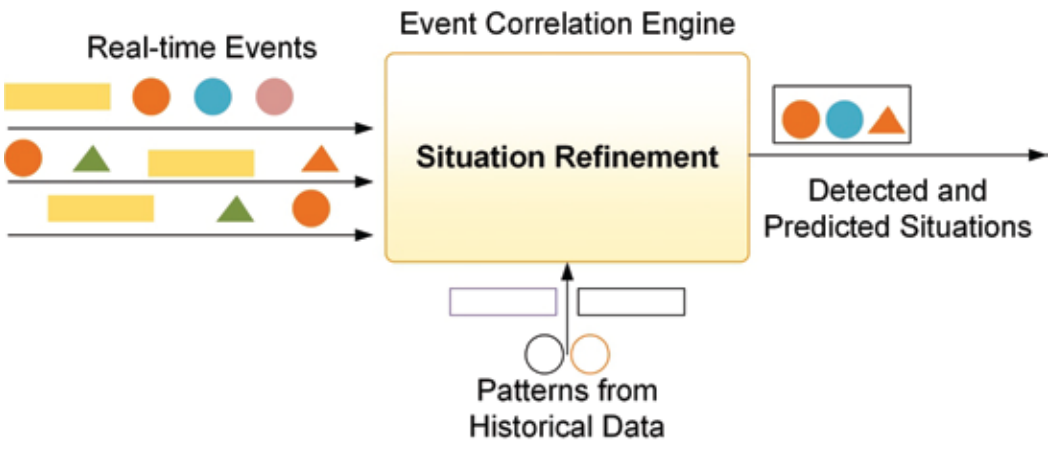
Who are Consumers in Kafka?

The term consumer describes the clients that consume messages from a Topic. Producers and Consumers can simultaneously write to and read from multiple topics. A consumer always consumes messages from a particular partition sequentially. If the consumer acknowledges a particular message it implies that the consumer has received all the messages prior to that message. This leads to the fact that Kafka relies on the pull-model to reach maximum performance on the consumption side.

Apache Kafka does not keep track of what messages have been consumed on the Broker and therefore does not rely on any acknowledgments from the consumer. Instead, the position of a consumer is just a single integer on a partition which defines the offset of the next message to consume. As a side benefit, it permits the consumer to rewind back to an old offset and re-consume data. Each consumer is represented as a process and these processes are organized within groups called consumer groups as Kafka supports the concept of consumer groups. Each consumer group consists of one or more consumers that jointly consume a set of subscribed topics.

Kafka as a Business Solution

Streaming solutions can ingest fast-moving data feeds. An architecture comprising a Kafka cluster as a high-speed data ingestor and an Apache Storm cluster for processing / analytics works well in practice, with a Kafka spout reading data from the Kafka cluster at high speed. The Kafka cluster stores the events in the queue. It is essential to outline the business solutions of Kafka and the value it brings to the table.



How is Kafka Different from Conventional Solutions?

Apache Kafka differs from traditional messaging system.

- It is designed as a distributed system which is very easy to scale out
- Offers high throughput for both publishing and subscribing
- Supports multi-subscribers and automatically balances the consumers during failure
- Persist messages on disk and thus can be used for batched consumption such as ETL, in addition to real time applications

What is a Kafka Business Use Case?

Kafka can be used in any architecture in a number of ways. Some of the popular use cases for Apache Kafka and the well-known companies that have adopted it are given below.

Popular Kafka Use Cases

- Log aggregation: the process of collecting physical log files from servers and putting them in a central place for processing. Kafka provides clean abstraction of log or event data as a stream of messages, thus taking away any dependency over file details. This also gives lower-latency processing and support for multiple data sources and distributed data consumption
- Stream processing: Kafka can be used for the use case where the collected data undergoes processing at multiple stages. An example is raw data consumed from topics and enriched or transformed into new Kafka topics for further consumption. Hence, such processing is also called stream processing
- Commit logs: Kafka can be used to represent external commit logs for any large scale distributed system. Replicated logs over Kafka cluster help failed nodes to recover their states
- Click stream tracking: Another very important use case for Kafka is to capture user click stream data such as page views, searches, and so on as real-time publish/subscribe feeds. This data is published to central topics with one topic per activity type as the volume of the data is very high. These topics are available for subscription, by many consumers for a wide range of applications including real-time processing and monitoring
- Messaging: Message brokers are used for decoupling data processing from data producers. Kafka can replace many popular message brokers as itoffers better throughput, built-in partitioning, replication, and fault-tolerance

Use Cases of Well-known Companies Using Apache Kafka

Some well-known companies that are using Apache Kafka in their respective use cases are given below.

- LinkedIn (www.linkedin.com): Apache Kafka is used at LinkedIn for the streaming of activity data and operational metrics. This data powers various products such as LinkedIn News Feed and LinkedIn Today, in addition to offline analytics systems such as Hadoop
- DataSift (www.datasift.com): At DataSift, Kafka is used as a collector to monitor events and as a tracker of users' consumption of data streams in real time
- Twitter (www.twitter.com): Twitter uses Kafka as a part of its Storm - a stream processing infrastructure
- Foursquare (www.foursquare.com): Kafka powers online-to-online and online to-offline messaging at Foursquare.It is used to integrate Foursquare monitoring and production systems with Foursquare-and Hadoop-based offline infrastructures
- Square (www.squareup.com): Square uses Kafka as a bus to move all system events through Square's various datacenters. This includes metrics, logs, custom events, and so on. On the consumer side, it outputs into Splunk, Graphite, or Esper-like real-time alerting

What are Kafka's Strengths?

Kafka's strengths are:

- High-Throughput and Low Latency: even with very modest hardware, Kafka can support hundreds of thousands of messages per second, with latencies as low as a few milliseconds
- Scalability: a Kafka cluster can be elastically and transparently expanded without downtime
- Durability and Reliability: messages are persisted on disk and replicated within the cluster to prevent data loss
- Fault-Tolerance: immune to machine failure in the Kafka cluster
- High Concurrency: ability to simultaneously handle a large number (thousands) of diverse clients, simultaneously writing to and reading from Kafka

HTC has been spending time to understand the design and implications of Kafka driven business solutions.

HTC custom developed and implemented a variant of event processing solution for the banking domain. Our solution helps to process high volumes of export data. The processed data is used to maintain export logs for reporting to India's central bank. The system brought in measurable benefits by enabling the banks to retrieve the applicable information on demand.

"Using the appropriate tools confers enormous business benefits and helps enterprises to take adequate preventive action immediately and avert catastrophes."

- Practice Head - Big Data, HTC Global Services Inc.

Conclusion

The future is built around high speed streaming data. The emergence of IoT and the flood of new generation wearable gadgets necessitates the lookout for alternative business models. The new business models which are emerging worldwide are data centric and data products driven. Industries need to have awareness of the emerging trends and realign their business solutions to meet the dynamic business challenges.

Acronyms

The acronyms used in this white paper and their expansion are provided below:

Acronym	Expansion
CEP	Complex Event Processing
ESP	Event Stream Processing
IOT	Internet of Things
PE	Processing Elements
SPE	Stream Processing Element
SPL	Streams Processing Language

References

1. Learning Apache Kafka - Second Edition - Nishant Garg, Packt Publishing: www.packtpub.com
2. Powered By Apache KAFKA
   https://cwiki.apache.org/confluence/display/KAFKA/
   https://cwiki.apache.org/confluence/display/KAFKA/Powered+By
3. Apache Kafka: Next Generation Distributed Messaging System
   http://www.infoq.com/articles/apache-kafka
4. Kafka Design Fundamentals
   https://www.safaribooksonline.com/library/view/apachekafka/9781782167938/ch04lvl1sec18.html
5. Kafka in a Nutshell
   http://sookocheff.com/post/kafka/kafka-in-a-nutshell/

About HTC's Big Data CoE

HTC's Center of Excellence for Big Data Management and Analytics brings in mature technologies and thought leadership. Our dedicated R&D team develops highly customized and cost-effective cutting edge solutions to enable clients manage and understand big data for improved and quicker decision making.

This white paper was developed by HTC's Big Data CoE.

HTC global services

Reaching out… through IT ®

World Headquarters

3270 West Big Beaver Road
Troy, MI 48084, U.S.A
Phone: 248.786.2500
Fax: 248.786.2515
www.htcinc.com