KAFKA: The Modern Platform for Data Management and Analysis in Big Data Domain

[1] Rishika Shree, [2]Tanupriya Choudhury, [3]Subhash Chand Gupta, [4]Praveen Kumar
[1,2,,3,4]Amity University Uttar Pradesh
[1]rishikashree777@gmail.com, [2] tchoudhury@amity.edu, [3]scgupta@amity.edu, [4] pkumar3@amity.edu

***Abstract*—In todays 21st century as technology is getting so much advanced, Apache Kafka emanate as one of the finest technology in the present world. Its fast, scalable, distributed stream processing platform and fault tolerant messaging system has made this technology to roar in the field of data processing and analysis. Apache Kafka is a distributed streaming platform mainly designed for low latency and high throughput. It is publish-subscribe messaging reassess as a constituent to each of number of legatee of commit log. The key notion of Apache Kafka is that it is used as a cluster on any number of servers. Server of Kafka stores record streams in classes known as topics. Every record contains a key, a value and a time stamp. It has two classes of application. Firstly, for building pipelines of real time data streams which is reliable to get the data between the systems or between the applications. Secondly, build applications streaming for real time that reacts to the record streams. A single Kafka mediator can handle hundreds of megabytes of reads and writes per second from thousands of clients. Scalable Kafka is designed to allow a single cluster to serve the central data backbone for a large organization.**

Keywords – Apache Kafka; hadoop; stream processing; low latency; flume; high throughput; publish-subscribe; producer; consumer.

I. INTRODUCTION

Apache Kafka is a tool within Hadoop eco-system that is built to handle transaction logs and other real time data feeds. Apache Kafka is an open source stream platform for processing which is mainly written in Scala and Java developed by the Apache Software Foundation. This project is for high-throughput, low-latency and unified platform to manage or handle data of real-time. The storage layer of Kafka is basically a "massively scalable pub/sub message queue architected as a distributed transaction log,"[1] which makes it very important for the infrastructures of enterprise for the processing of data streams.Kafka is used to connect with the systems externally (import or export of data) with the help of Kafka Connect and gives Kafka Streams, a Java stream library for processing. Kafka's design is developed which is actually influenced by transaction logs[2].

Kafka's major distributed approach into the level of enterprise infrastructures, performance of Kafka is monitored in every scale and it has now become a critical issue. Zookeeper is present there which is basically for consumers coordination [3].

II. LITERATURE REVIEW

Apache Kafka was actually made by LinkedIn which basically was an open source in 2011. Hadoop is dependent on framework of Java programming which is basically for the processing data which is in large amount in computing environment which is distributed. It consists of a file system which is distributed known as Hadoop Distributed File System/HDFS, whose main function is to support fast data transfer between the existing nodes. Hadoop came into picture from the Google File sytem paper[4]] which was published in October 2003. The paper introduces one more research paper from Google – MapReduce: It is actually Processing of simplified data on Large Clusters[5]. New development and research took place on the project of Apache Nutch , and a new project arrived known as Hadoop in January 2006[6]. The name 'Hadoop' was named with a small interesting story - during that time Doug Cutting was working in Yahoo![7] And named Hadoop after his son's toy which was actually a baby elephant.[8]. Hadoop Ecosystem is a framework of various types of complex and evolving tools and components which have accomplished advantage in solving problems. Hadoop Ecosystem is estranged in four different layers: data storage, data processing, data access, data management. Components inside the ecosystem of Hadoop, are the entities which are explicit and evident. The architecture of Hadoop provides eminence to Hadoop, Hadoop YARN, Hadoop Distributed File Systems (HDFS) and Hadoop MapReduce of the Hadoop Ecosystem. All Java libraries are provided by Hadoop common, abstraction at OS level,utilities. Architecture of Hadoop consist of HDFS which supports the application which is of high throughput access and one more tool MapReduce which supports parallel processing of huge amount of data.

III. HADOOP ECOSYSTEM

In the file system which is distributed storage of data takes place in the Storage layer. The major component of Hadoop known as Hadoop distributed file system or HDFS which is distributed and also scalable is written in Java for the framework of Hadoop whose range is typically in gigabytes or terabytes[9] which can be across any number of machines. HDFS is reliable by having the capability to replicate data across any number of hosts and consist of Hbase as well as

ColomnDB Storage. Hbase is only for the storage of structured data.

Hadoop consist of layer for processing which is for Scheduling management of resource and cluster. This layer has tools like YARN which is for job scheduling and cluster resource management which is with Map Reduce. In the ecosystem of hadoop the layer which is above the file system is the MapReduce Engine which has a *JobTracker*, where submission of applications take place of client jobs of MapReduce. After this JobTracker gives work to *TaskTracker which is available in the* nodes inside the cluster.

There is one data access layer, which is a request from Management layer that is to be sent to the Data Processing Layer. There are certain projects that have been introduced for the above layer, such as: Hive[11], A data warehouse infrastructure that provides data summarization and ad hoc querying; Pig[12], A high-level data-flow language and execution framework for parallel computation.

Data management layer consist of components like apache zookeeper and apache flume

Apache Zookeeper[9] is a coordination service for distributed application that enables synchronization across a cluster. Zookeeper inside Hadoop which can be visualized as a repository that is centralized where applications that are distributed can store data and extract data from it. It makes the distributed system to function together and form a single unit, it is used for coordination serialization and synchronization goals.

Apache Flume [10] which is a distributed and reliable service helps in collecting and aggregating and also moving huge amount of data like log data efficiently and effectively. It has a flexible and simple architecture based on streaming data flows. It is a fault tolerant with mechanisms of tunable reliability and many and also recovery mechanisms. It supports a simple data model which is extensible and supports application of online analytics.

Apache Kafka is a tool inside hadoop eco-system that is built to handle ingesting transaction logs and other real time data feeds.

IV. APACHE FLUME

Flume is basically a distributed tool of Hadoop for nicely picking up, storing, and transferring huge log data. Flume consists of simple architecture based on data streams flow. It is a fault tolerant with a great reliability features and and error management mechanisms. It is used for online application analytic.

A. Advantages of flume:

a. It can absorb data of log inside a central storage from multiple web servers(HDFS, HBase) effectively.
b. Flume can be used to extract data from multiple servers to Hadoop.
c. Flume can also be used for importing large volumes of data event generated by the social networks.
d. Flume is used for flows of multi-hop , fan-in and fan out , contextual routing etc.

V. HOW KAFKA CAME INTO PICTURE?

A. In 2010, LinkedIn wanted to move to something different to handle event log processing because the existing system was fragile, not very scalable, and used XML(!) as the format. Flume was still very much in its early days. Cloudera was invited over to talk about Flume to see if it would meet our needs. Two big points came out of that discussion:

B. Flume was extremely hard to manage because it did not multiplex connections. It was essentially one socket per log type. This was thought to be sort of ridiculous, especially given that syslog and other demigod had been doing this for a very long time. To add to the overhead, Flume required that there be a different configuration file per socket that was opened.

C. A client is needed to pull rather than a push to client for various reasons, many of the Kafka team have documented elsewhere. It was asked if Flume could be modified to reverse the polarity a bit. The answer was (mostly) no.

It was pretty clear that the team was going to need to build something different to handle our needs. Thus Kafka was born.

VI. APACHE KAFKA:

Kafka is a streaming platform. It is capable to publish and subscribe the records which are in the form of streams. It is like a messaging system of an enterprise. It is used in storing streams of records without any fault. It helps in processing record streams as they come.Kafka is run like cluster which is compatible for more than one host.The Kafka cluster categorizes *record streams* in *topics*. The record consists of key, timestamp and a value. A topic is a subdivision where records get published. Topics in Kafka can always subscribed by more than one subscriber; that is, a topic can have any number of consumers, it can be zero also, that are the subscribers to the data written to it.. The partitioned log is maintained for each Kafka topic.

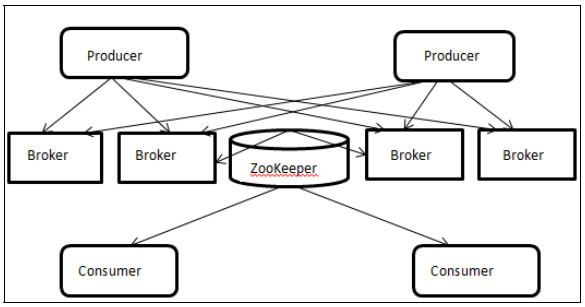

Figure 1: Kafka architrcture

This figure shows the architecture of Kafka containing producers that publish message or data with different brokers

in which ZooKeeper helps in coordination and lastly consumer subscribes the message or data that is published by the producers.

A. Kafka's capabilities:

a. It helps in making real-time pipelines of data stream.

b. It is used in creating real-time application streams.

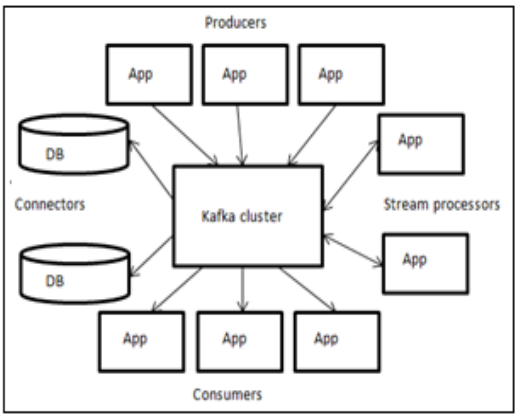

Figure 2: Kafka APIs

B. Kafka has four broad Application Programming Interfaces:

a. Producer API:- In this API, record stream gets publish in more than one topics of Kafka.

b. Consumer API:-In this API, processing of record streams takes place and also the subscription of any number of topics of Kafka.

c. Streams API:- In this API effectively processing of input streams takes place which ultimately produces output stream to any number of output topics.

d. Connector API:- In this API, running and making use of consumers and producers which can be reused again which helps in connecting Kafka topics to the application or data systems which is already available. For example, if we are making any change in a relational database table then through connector each and every change might get capture.

C. Advantages of Kafka:

a. If a producer sends messages to a partition topic particularly, proper ordering of messages take place. The instance of a consumer views the record in the same order as they are placed inside the log.

b. The topic having the factor N as replication, it can bear till N-1 failures of server with no data loss as—it has the property of scale processing and also supports any number of subscribers—it is not required to choose any of them.

c. Kafka has excellent guarantee of order as compared to traditional or other system for messaging.

D. Applications of Kafka:

a. Kafka for Messaging:

Messaging historically consist of models which can be categorize as:publish subscribe and queuing. In which queue is a pool, which consists of customers that can read from server in which records are send.The group representing consumer concept in Kafka emphasize on these two basic key points. In a queue the group of consumer gives access to process collection of processes. By using publish-subscribe, Kafka provides flexibility to disseminate messages to various groups of consumer.

A conventional queue put records in a particular order on the host, and when there are more than one users use the queue and the records are in the same sequence as the storage is done. Though the host provides the record in sequence, the delivery of records are synchronous to users, so sometimes it happens that the arrival is not in order to each consumers. This indicates that the order of the logs is lost whenever parallel consumption takes place.

Kafka can do this more nicely, by providing the concept of parallelism—the partition which takes place inside the topics, Kafka is capable to give both guarantee of proper order and also to balance the load over a pool of processes of consumer. This can be done by providing partitions to the consumer inside the group of consumer such that every partition is used byone consumer in the group of consumer. Through this we can become sure that the reader of that particular partition is no one other than the consumer which uses the data in order. It is not possible to have consumer instances more than the partitions.

b. Kafka for Storage:

Message queue that gives the permission to publish messages uncoupled from using them is basically like a system for storage which is for the in-flight messages. Kafka is an excellent system for storage. The data jot down to Kafka is inscribed in the disk and then replication takes place for fault tolerance. The disk structures Kafka uses scale well—Kafka gives the same result in both the situation of providing whether 100 KB or 100 TB of data which is persistent that the host contains Kafka is a special distributed file system giving out high-performance, commit log storage of low-latency and replication.

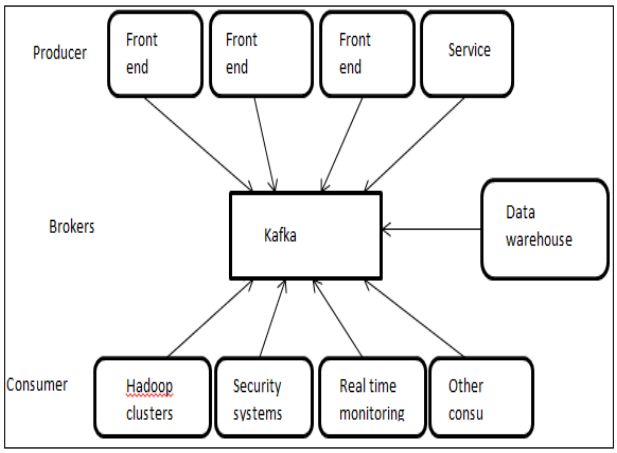

Figure 3: Kafka decouples Data Pipeline

c. Kafka as processing of Streams:

It is actually not much to have just read and store or to write data streams. The main motive is to process the data streams. Kafka takes the streams in continuous manner from the topics as input and after processing, it produces continuous data streams to output topics. But Kafka for complex or difficult processing, it provides an API of integrated Streams. This helps in making applications that is capable for processing which is non trivial and can perform aggregations of streams or joining of streams together. This can solve tough or difficult issues if faced by the application.

d. Combination of above three:

By combining all the above that is messaging, storage, and processing of streams may looks not familiar but yes it is necessary or important for Kafka as a platform for streaming. File system like HDFS use for storing files which is static for the batch processing. Basically this type of system is used for storage and processing data which is *historical*. A conventional enterprise for messaging does the processing of messages in future that arrives when it gets subscribed. Kafka can do both the things that is as it has the capability to do both. With the property of low latency and excellent storage it is capable to process the historical data and instead of ending the process when it reaches the end, it keeps on processing the future data which is arriving at that time.

VII. COMPARISION OF APACHE KAFKA AND APACHE FLUME:

a) Apache Kafka is a General purpose distributed publish-subscribe messaging system whereas Apache Flume is built around Hadoop ecosystem for the primary purpose of sending messages to HDFS & HBase.
b) With Kafka you pull data, so each consumer has and manages it's own read pointerWith this, you could deliver your event streams to HBase, Cassandra, Storm, Hadoop, RDBMS all in parallel whereas Aapache Flume pushes data, you have to do some interesting work to sink data to two data stores. which requires copying your channels, one for each sink.
c) With Kafka 0.8+ you get replication of your event data. If you lose a broker node, others will take up the limp to delivery your events without loss whereas with Flume & FlumeNG, and a File channel, if you lose a broker node you will lose access to those events until you recover that disk.
d) Apache Kafka is used for any system to connect to other systems that requires enterprise level messaging (website activity tracking, operational metrics, stream processing etc) whereas the database channel with Flume is reported too slow for any production use cases at volume.

VIII. FIELDS WHERE KAFKA CAN BE USED:

a) Tracking of website activity:
Kafka is capable to make a pipeline for activities, it is like a set or group for publish/subscribe which is actually real time which consequently means that whatever activities going on the site is published to topics which is central that includes for each activity there is one topic. (Pageviews of search result pages etc that is done by users) Tracking of activity which is actually in high volume because messages for each user view page are generated..

b. Aggregation of log:
Kafka is used by many people for aggregation of log solution which is used to replace the solution for aggregation of log. Aggregation of log is basically used to collect log files that are physical from the server and store it to a central place (HDFS) mainly for processing. Kafka hides the file details and provides a clear abstraction of log data as a messaging streams. This results in processing with low latency. By comparing it to systems like log-centric that is Flume, Scribe whereas Kafka provides nice performance, stronger and higher guaranty for durability because of the replication feature along with very low latency .

c. Commit-Log:
Kafka is capable to serve and can be used for external commit-log for a system which is distributed. Basically the log supportss replication of data between the nodes and it is like a method for re-syncing or technique for the nodes which are failed to backup or restore the data.

IX. CONCLUSION

Kafka can be used when you particularly need a highly reliable and expandable enterprise messaging system to connect many multiple systems like Hadoop. Kafka is the way to go if you want high repitition /custom-made producer (easily customization for strong delivery/ordering guarantees) Kafka offers partitioning and high throughput message delivery. Kafka can handle hundred thousands of message per second per mediator. If you are looking for high velocity data, go with this one. Kafka has a lot of options for guarantee levels on both producing and consuming, with their tradeoffs. our reasoning for choosing Kafka and some of the advantages of Kafka. The Flume head start on HDFS integration has been really closed on by Kafka via the Confluent Kafka connectors which are professional integration components with the Hadoop ecosystem.

The one big advantage is that Kafka is being a pull system, i.e. Kafka provides back pressure to prevent overflowing consumers, by persistently storing the incoming messages until they “expire” configurable days later, so that late consumers can pick the messages up even rewind a few times, at their own pace - while Flume is a push system which implies data loss when consumers can’t keep up.

Thus ensures that the future of Kafka is really bright and it can process huge amount of data in a very small span of time

which can help in solving critical and heap of problems in a very easy manner.

REFERENCES

[1] "Monitoring Kafka performance metrics", Datadog Engineering Blog, accessed 23 May 2016

[2] The Log: What every software engineer should know about real time data's unifying abstraction. LinkedIn Engineering Blog, accessed 5 May 2014

[3] "collecting Kafka performance metrics- Datado"*2016-04-06. Retrieved 2016-10-05.*

[4]. *Ghemawat, Sanjay; Gobioff, Howard; Leung, Shun-Tak. "The Google File System"*.

[5] "Big data: A survey," Mobile Networks and Applications, M. Chen, S. Mao, and Y. Liu, vol. 19, no. 2, pp. 171–209, 2014.

[6] M. D. Assunc ̧ao, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," Journal of Parallel and Distributed Computing, M. D. Assunc ̧ao, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, vol. 79, pp. 3–15,2015.

[7] "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads,"Proceedings of the VLDB Endowment, Y. Chen, S. Alspaugh, and R. Katz, vol. 5, no. 12, pp. 1802–1813,2012.

[8] "Scarlett: coping with skewed content popularity in mapreduce clusters," in Proceedings of the sixth conference on Computer systems. ACM, 2011, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris pp. 287–300.Datadog Engineering Blog, accessed 23 May 2016/

[9] LinkedIn Engineering Blog, accessed 5 May 2014

[10] "Index-based join operations in hive," in Big Data, M. Mofidpoor, N. Shiri, and T. Radhakrishnan, 2013 IEEE International Conferenceon. IEEE, 2013, pp. 26–33.

[11] Programming hive. "O'Reilly Media, Inc.", E. Capriolo, D. Wampler, and J. Rutherglen, Programming hive. 2012.

[12]" O'Reilly Media, Inc.", E. Sammer, Hadoop operations. 2012..