International Journal on Recent and Innovation Trends in Computing and Communication
Volume: 5 Issue: 6

ISSN: 2321-8169
1229 – 1231

Building a GUI Application for Viewing and Searching Apache Kafka Messages

Shivkumar Goel
Deputy H.O.D,
Dept of MCA, VES Institute of Technology, Maharashtra Mumbai
shivkumar.goel@ves.ac.in

M Prabhanath Nair
Student,
Dept of MCA, VES Institute of Technology, Maharashtra Mumbai
prabhanath.nair@ves.ac.in

T Varghese John
Student,
Dept of MCA, VES Institute of Technology, Maharashtra Mumbai
varghese.john@ves.ac.in

***Abstract*-**Apache Kafka is a scalable messaging system that follows Publish-Subscribe Model as its core. Several traditional messaging system like MSMQ, RabbitMQ exist but they have limitations in terms of performance and throughput. Kafka, developed at LinkedIn is the latest messaging technology being adopted by most of the top internet companies. The purpose of this paper is to provide a GUI and search tool, to view and monitor messages insideKafka.

Keywords*-Messaging system, kafka, kafka search, GUI, disconnected system*

*****

I. INTRODUCTION

A message queue provides an asynchronous communications protocol and is a means to connect any two disconnected systems. Systems that put messages onto a message queue do not require any immediate response to continue processing. [11]

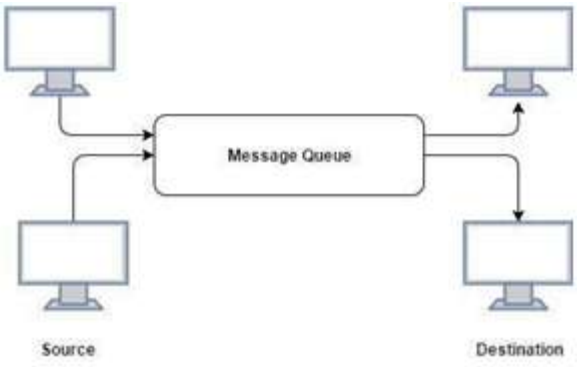


Figure-1: Message Queue Architecture

Major message queues used in the industry are:

- MSMQ
- RabbitMQ
- ActiveMQ
- ApacheKafka

- ***MSMQ***

The Microsoft Message Queuing (MSMQ) technology enablesapplications that may be temporarily offline or running at different times to communicate across heterogeneous networks.[1]

- ***RabbitMQ***

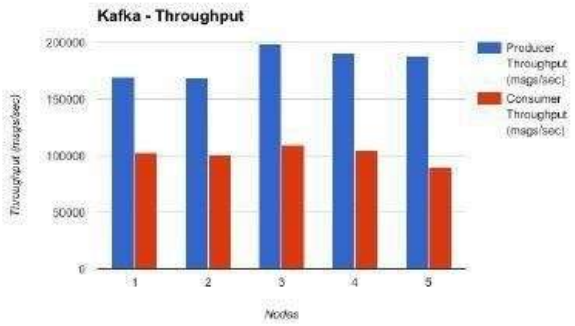
RabbitMQ is open source message broker software which implements the Advanced Message Queueing Protocol (AMQP). [9] The RabbitMQ server is built on the Open Telecom Platform framework for clustering and failover.

- ***ActiveMQ***

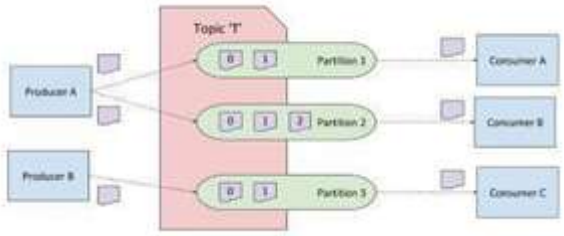
ActiveMQ provides "Enterprise Features" which in this case means fostering the communication from more than one client or server. ActiveMQ employs several modes for high availability, including both database row-level locking mechanisms and file-system, sharing of the persistence store via a shared file system, or true replication using Apache Zookeeper. [10]

- ***ApacheKafka***

Kafka among these is one of the emerging open source message queuing systems that is scalable and provides faster throughput.



1) Figure-2: Kafka Throughput



(Single -Producer & Consumer) [2]

 ISSN: 2321-8169
1229 – 1231

Kafka's architecture consists of producers, consumers, brokers, topics andpartitions.Producers or the sources generating data push messages of a particular type into topics.

These messages are then stored in a set of servers called brokers. Since Kafka is a distributed system, topics can be divided into partitions and spread across multiple brokers. This data is then pulled by consumers. Often, there is a mismatch between the rate at which producers push messages to Kafka and the rate at which consumers consume these messages. This is known as Consumer Lag.

The shortcoming with Kafka is that it is yet to provide proper visibility of messages inside the queue. As mentioned, Kafka is an open source technology, so development and improvement at every level is to beexpected.

II. APPLICATION

A GUI and search tool for tracing and monitoring messages that flow through the message queue.

Figure-4: Kafka Search Tool Deployment

The above architecture illustrates our implementation of search tool over Kafka, as it provides visibility for the set of messages inside the messaging queue. Also since Kafka is capable of storing and streaming loads of data, we need to restrict the search limit to an extent, so that the throughput for it is maximum.

III. METHODOLOGY

Consider the following scenario where a Pub-Sub model is illustrated that uses Kafka as an intermediate in its process of communication:

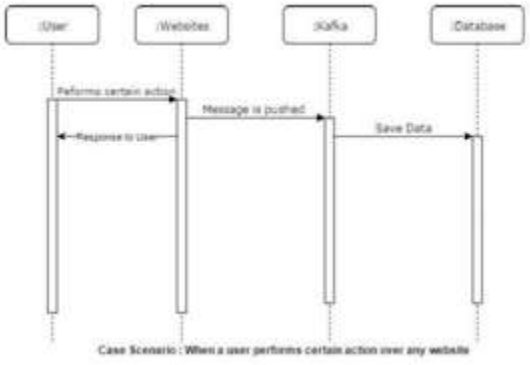


Figure-5: Sequence Diagram for Kafka

2) Problem Statement:

- Kafka comes into picture when we have a huge amount of data coming from any source (e.g. websites, applications) to be consumed by another destination system (e.g. data warehouse).
- The systems that push messages to Kafka needs to be debugged to check if data is actually being received by Kafka and forwarded to theconsumer.
- Consider the followingscenario's:
  - Producer is not able to produce propermessages.
  - Producer itself has produced baddata.
  - Consumer is not able to consume properly from the messagingqueue.
  - Serialization issues while the data flows from one system toanother.

Thus it becomes, important and vital to get to the root cause of these issues. To debug each and every flow of data, we must have a UI where viewing messages inside the queue is possible.

Kafka as mentioned earlier follows a pub-sub model where it requires a producer and a consumer to push and pull messages from the queue. In our case, we require a consumer that is capable of pulling out latest/real-time messages from Kafka. We evaluated the following libraries for .NET that would allowus to fetch messages from Kafka inourapplication:

- Rest Proxy
- Kafka .NETclient
- Confluentclient

- *RestProxy*

The Kafka REST Proxy provides a Restful interface to a Kafka cluster. It makes it easy to produce and consume messages, view the state of the cluster, and perform administrative actions without using the native Kafka protocol or clients. [5].

It works over HTTP to communicate with Kafka and but it provides limited information as Confluent Rest Proxy is not able to fully utilize the exposed client ofKafka:

- List ofTopics
- List of Consumers andProducer

- *Kafka .NETClient*

.Net implementation of the Apache Kafka Protocol that provides basic functionality through Producer/Consumer classes.

Using this library we can do the following operations:

- The producer can send one message or an entire batch to multiple topicsatonce.
- The simple consumer allows full control for retrieving data. You can instantiate a consumer directly b y providing a consumer configuration and then calling Fetch. [6]
- This library is able to pull and display messages from Kafka, but it has one major disadvantage.
- It is a polling consumer and therefore keeps waitingto fetch new messages that are pushed to thequeue.
- This wasn't suitable in our use case where we just wanted to fetch messages inside the queue and not wait to retrieve new messages that were being pushed.

- *Confluent.NETClient*

This is also a similar kind of .NET Client as above, but it fit well with our requirements. It provides classes and methods, using which we can access messages inside Kafka without continuously polling the queue. This is the library that we went with to fetch messages fromKafka.

Features:

- **High performance** – The client is a lightweight wrapper around 'librdkafka', a finely tuned Cclient.
- **Support** - Commercial support is offered byConfluent.[7]

IV. CONCLUSION AND FUTURE WORKS

In this paper, we've demonstrated how we went through various .NET libraries and finally zeroed down on the official Confluent client to fetch messages into our application from Kafka. To improve user experience when debugging applications, the ability to search for a particular message from the messages fetched was also implemented.

A graphical dashboard to display the consumer lag of every consumer is something that could improve the usefulness of this tool and is currently being worked upon.

ACKNOWLEGEMENT

This acknowledgment is a small effort to express our gratitude to all those who have assisted us during the course of preparing this paper. We are greatly indebted to express immense pleasure and sense of gratitude towards our guide andmentor Prof. Shivkumar Goel for his constant support and valuable encouragement.

REFERENCES

[1] Performance and stability testingof MSMQ in the .NET environment Jae-Kyu Chun 1, Seok-Hyung Cho 2 Oper. Support Syst. Lab., Korea Telecom, South Korea, IEEE

[2] A survey of distributed message broker queue Vineet John 1, Xia Liu 2 University of Waterloo

[3] P. Bellavista, A. Corradi, and A. Reale, "Quality of Service in Wide Scale Publish/Subscribe Systems," Communications Surveys & Tutorials, IEEE, vol. 16, no. 3, pp. 1591-1616, 2014.

[4] J. Zhao, Z. Sun, and Q. Liao, "Implementation of K- Means Based on Improved Storm Model." p. 5.

[5] http://docs.confluent.io/3.0.0/kafka-rest/docs/intro.html

[6] https://github.com/Microsoft/CSharpClient-for-Kafka

[7] https://github.com/confluentinc/confluent-kafka-dotnet

[8] Kafka and its Using in High-throughput and Reliable Message Distribution Zhenghe Wang, Wei Dai*, Feng Wang, Hui Deng, Shoulin Wei, Xiaoli Zhang and Bo Liang Yunnan Key Laboratory of Computer TechnologyApplications Kunming University of Science and TechnologyKunming, China

[9] https://en.wikipedia.org/wiki/RabbitMQ

[10] https://en.wikipedia.org/wiki/Apache_ActiveMQ

[11] https://www.cloudamqp.com/blog/2014-12-03-what-is-message-queuing.html