International Journal of Innovative Research in Engineering
Volume 5, Issue 2 (March-April 2024), PP: 172-174.
https://www.doi.org/10.59256/ijire.20240502023
www.theijire.com
ISSN No: 2582-8746

Goat Q: A distributed message broker

Raja Khan[1], Sourav Paul[2], Rohit Kumar[3], Tanmoy Poali[4], Soumyajit Shome[5], Sujata Ghatak[6], Kaustuv Bhattacharjee[7], Anirban Das[8]
[1,2,3,4,5,6,7,8] *Department of Computer Application, University of Engineering and Management, Kolkata, West Bengal, India.*

How to cite this paper:
Raja Khan[1], Sourav Paul[2], Rohit Kumar[3], Tanmoy Poali[4], Soumyajit Shome[5], Sujata Ghatak[6], Kaustuv Bhattacharjee[7], Anirban Das[8], "Goat Q: A distributed message broker", IJIRE-V5I02-172-174.

Abstract: ***goat Q****, a distributed message broker explores the design and implementation of a robust message broker system. Utilizing advanced distributed concepts of network and synchronization the project aims to enhance efficiency and reliability of message delivery which fits in an event driven system. It also ensures fault tolerance and scalability. It also considers the integration with containerization and on premise and cloud deployment models to use goatQ as a message broker solution in production.*

The project seeks to unravel the intricate design of message brokers, illuminating the functionality of publishers, subscribers, and message queues. A key focus lies in a rigorous performance analysis, evaluating metrics such as throughput, latency, and resource use across wide variety message broker implementations. Scalability and fault tolerance, critical considerations in the realm of distributed systems, are addressed through an exploration of how message brokers adapt to dynamic scaling requirements and recover from failures.

By synthesizing findings from diverse dimensions of message broker technology, this research project aims to equip technology professionals with valuable insights, aiding informed decision-making in the selection, optimization, and integration of message brokers within their ecosystems.

Keyword: *About four key words or phrases in alphabetical order, separated by commas. Keywords are used to retrieve documents in an information system such as an online journal or a search engine. (Mention 4-5 keywords)*

I.INTRODUCTION

In the ever-evolving landscape of information technology, efficient and reliable communication among distributed systems is paramount for the seamless operation of modern applications. The role of message brokers has become increasingly vital in facilitating this communication by serving as intermediaries that enable the exchange of messages between disparate components. As organizations continue to embrace complex, interconnected architectures, the need for robust message broker systems has grown exponentially.

Our research project, titled "Advancing Communication: A Comprehensive Exploration of Message Brokers," aims to delve into the intricacies of message broker technologies, shedding light on their functionality, performance, and the impact they have on the overall efficiency of distributed systems. In an era where real-time data processing, scalability, and fault tolerance are critical considerations, understanding the nuances of message brokers becomes indispensable.

The objectives of our research encompass:

Understanding Message Broker Architecture: We will examine the fundamental architecture of message brokers, exploring key components such as publishers, subscribers, and the message queue. By gaining insight into the underlying structure, we aim to identify the factors that contribute to the efficiency and reliability of message delivery.

Performance Analysis: Our project will conduct a comprehensive analysis of the performance metrics associated with different message broker implementations. This includes considerations such as throughput, latency, and resource utilization. By benchmarking popular message broker solutions, we aim to provide valuable insights into their comparative strengths and weaknesses.

Scalability and Fault Tolerance: As distributed systems continue to scale, the ability of message brokers to seamlessly grow with the system is of utmost importance. We will investigate the scalability features of various message brokers and evaluate their resilience in the face of failures, ensuring that our research addresses the critical aspect of fault tolerance.

Security Considerations: In an era of increasing cyber threats, the security of communication channels facilitated by message brokers cannot be overlooked. Our project will assess the security features of different message broker solutions, examining encryption, authentication, and authorization mechanisms to provide a holistic view of their security posture.

Integration with Modern Architectures: As micro services, server less computing, and containerization become standard practices, we will explore how message brokers integrate with these modern architectural paradigms. Understanding the compatibility and advantages of message brokers in diverse environments is crucial for their successful adoption.

II. BACKGROUND STUDY

"The relevance of using message broker in robust enterprise applications'' an article published by **IEEE** covers the design choices of modern enterprise applications, classic software engineering issues like tight coupling of micro services which might lead to cascaded failure. Synchronous operation between micro services also degrades the performance when significantly high volumes of data are processed synchronously.

Apache kafka architecture and internals, - an engineering blog by jun Rao at confluent.io encompasses the design choices of apache kafka to fit for all sorts of use cases. It covers how highly available brokers are hosted on a production kafka cluster. Kafka keeps the computational layer and storage layer separate to scale them independently as per the requirements. They persist messages with persistent flags on an append only log File. The messages can be partitioned by hashing the key of the event record.

Performance Evaluation and Comparison of Distributed Messaging Using Message Oriented Middle ware: A study by Naveen Muparaju, University of North Florida who conducted a comprehensive performance evaluation of popular message broker solutions, comparing factors such as throughput, latency, and resource utilization. Their findings contribute valuable benchmarks for assessing the efficiency of different message broker implementations.

Scalability and Fault Tolerance: Scalability and fault tolerance are critical considerations in distributed systems. By exploring the scalability features of message brokers, emphasizing their ability to handle increasing workloads. Additionally, the article published titled as **Evaluation of highly available and fault-tolerant middleware clustered architectures using Rabbit MQ** investigates fault tolerance mechanisms in message brokers, offering insights into their resilience and recovery strategies.

Security considerations in message broker systems are addressed by an open stack blog. This research examines encryption, authentication, and authorization mechanisms employed by message brokers, highlighting the importance of securing communication channels in distributed environments.

Integration with Modern Architectures: The integration of message brokers with contemporary architectural paradigms is explored in studies like the one by Christian Antonio. This work delves into how message brokers align with micro services, serverless computing, and containerization, offering insights into their compatibility and advantages in diverse technological ecosystems.

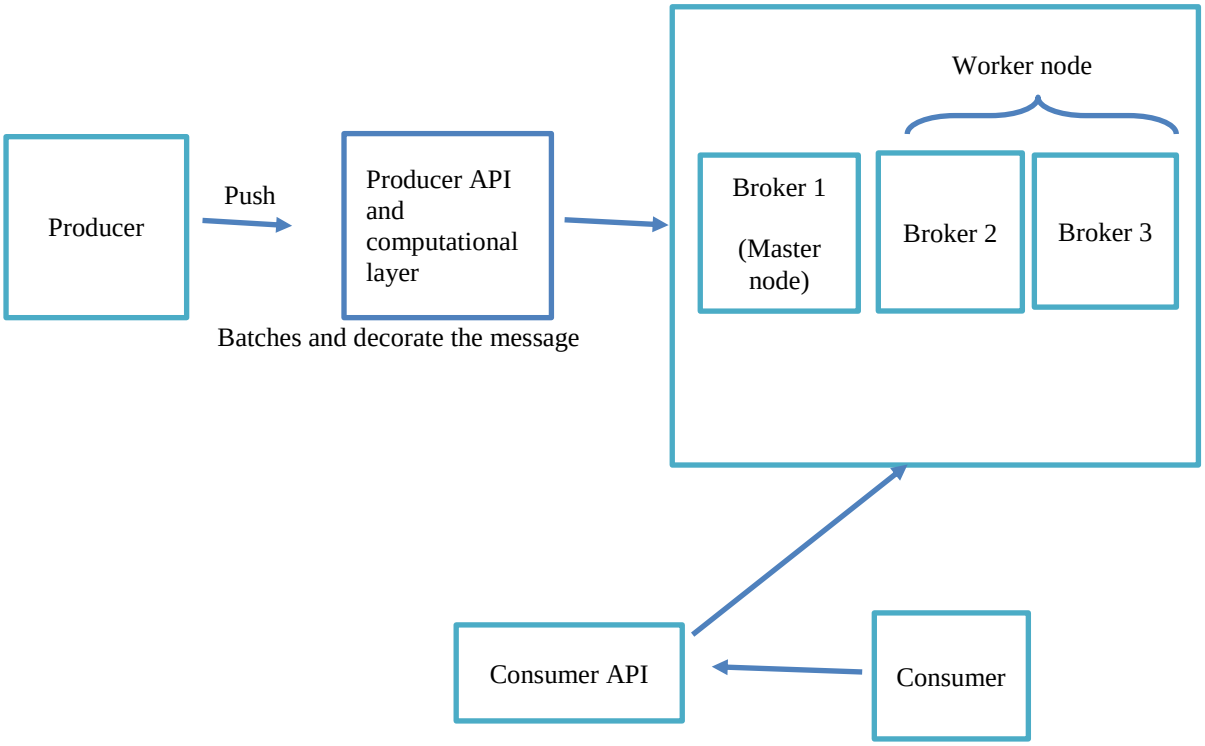
Industry Use Cases and Best Practices: Case studies and practical insights from industry applications are discussed by Sharma and Patel (2019). This literature sheds light on real-world implementations of message brokers, showcasing best practices and lessons learned from organizations that have successfully integrated message broker technology into their systems.

III. METHODOLOGY

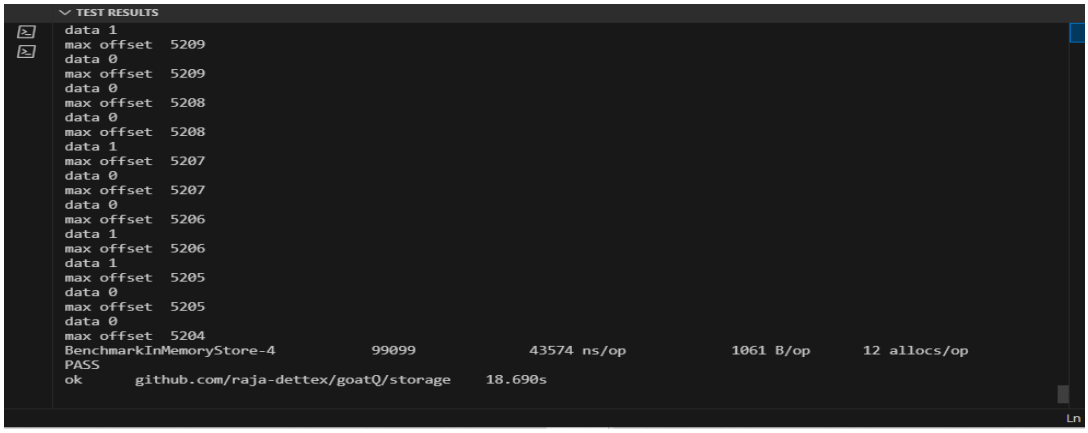
Goat Q offers a message broker solution, with easy integration of cross platform. And easily deployable in most of the modern infrastructural setup.

A typical end to end workflow of goat Q is: producer produces messages and events those have to be processed asynchronously. Goat Q brokers are single handedly responsible for handling the stream of events. Goat Q brokers are essentially the storage nodes in the data layer. In a typical Master replica setup, the message goes to the master node. Master node is responsible for routing and queueing and replicating messages.

Goat Q provides mechanisms for acknowledging successful receipt and processing of messages, ensuring reliable message delivery. It Supports transactions to ensure atomicity and consistency of message processing, especially in scenarios where multiple messages need to be processed together. Goat Q allows you to spin up multiple nodes to replicate messages. It supports the deployment of redundant instances to ensure that if one instance fails, another can seamlessly take over. goatQ allows it to scale horizontally by adding more nodes when the load is comparatively high and also scale down when the spike is over. The key design choice of goat Q is keeping the computational layer and data layer separate from each other. So that they can be independently scaled.



IV.RESULTS AND DISCUSSION



The above benchmarking results shows that allocations per op is really low and bytes allocated per op is also decent.

References

[1] *The Relevance of Using Message Brokers in Robust Enterprise Applications : Publisher - IEEE, Authors : Vladyslav Apukhtin; Mariya Shirokopetleva; Victoria Skovorodnikova*

[2] *How Apache Kafka Works: An Introduction to Kafka's Internals : Author - Jun Rao*