International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I1P102

Volume 72 Issue 1, 8-11, January 2024

Original Article

Asynchronous Data Stream Ingestion in Distributed Cloud Infrastructure

Vinay Gupta

Microsoft Corporation, Senior Software Engineer, Redmond, WA, USA.

Corresponding Author : vinaygupta@microsoft.com

Received: 14 November 2023 Revised: 23 December 2023 Accepted: 11 January 2024 Published: 24 January 2024

Abstract *- Businesses use cloud infrastructure to boost performance and cut costs, making it a crucial engine for today's agile software ecosystem. Given the staggering amount of data generated daily, businesses have started offloading most of their data onto the cloud. However, cloud providers need to catch up in meeting the exponential volume of incoming data, pushing cloud providers to come up with innovative solutions to scale ingestion effectively. This paper discusses one innovative solution for scaling the ingestion of asynchronous data streams using distributed Kafka-based ingestion. I will highlight the relevant components of a Kafka cluster, what is a bottleneck in naïve data ingestion, and how Kafka can scale ingested data streams to billions per day. I will also discuss how Kafka clusters are used in distributed cloud infrastructure and why asynchronous data is a good candidate for Kafka-based ingestion. Kafka has significantly increased the scale of data streams a public cloud provider can ingest.*

Keywords *- Apache Kafka, Asynchronous data streams, Cloud infrastructure, Distributed messaging queues.*

1. Introduction

Astounding, 329 million terabytes of data are generated every day. [1] Additionally, the data must be stored, processed, and converted to meaningful information for customers in microseconds. Can the monolithic services the world used in the early 2000s process this amount of data efficiently?

This is where the business need for public cloud providers came into existence. The need for fast, centralized storage of these data streams was becoming increasingly obvious, but how to change our monolithic architecture to handle the increasing data requirements was an open problem. This is where the concept of microservices was introduced in 2011. [2]

But how can the infrastructure be updated to decouple the actions performed by individual services and accommodate seamless communication between these services?. The current infrastructure used in public cloud providers is better suited for synchronous data streams. This paper will advocate distributed messaging queues for efficient asynchronous communication between software components of a cloud infrastructure.

2. Background

When the data is sent to be ingested by the cloud provider, the sender expects three critical assurances:

- Data Persistence
- Data Security
- Data Processing

While one can argue that monolithic service architecture is good enough for data persistence and data security, fast data processing can only be done through microservices-oriented architecture. Furthermore, these data streams can be both synchronous and asynchronous and be huge. [3] That is why using a messaging queue in a microservices architecture is a well-suited solution to this open problem.

2.1. Distributed Cloud Infrastructure to Store Data

To comply with the data privacy laws of different countries, cloud providers need to ensure that data centers in each country have a specific data flow structure. This requires cloud providers to offer services globally and have local data centers in each country. This also benefits the customers, as having data centers nearby reduces latency and improves cloud performance.

2.2. Asynchronous vs Synchronous Data Streams

Customers who send a request to the cloud services and wait for the processed information are sending synchronous data streams. However, there are asynchronous data streams when a customer sends a request to the cloud services and can do something else after receiving an acknowledgment from the providers. There is no set clock in asynchronous

data streams, and requests of any size can be sent anytime. Cloud providers need to ingest both asynchronous and synchronous data streams.

2.3. Distributed Messaging Queue

When billions of asynchronous data streams enter the cloud provider ecosystem waiting to be processed, they are added to a messaging queue, which can be considered a long list. Publishers add data/requests to the queues, and subscribers fetch this data from the queue to start processing data to generate meaningful information. [5] Advanced messaging queues use persistent databases to store data.

3. Apache Kafka as a Distributed Messaging Queue

Messaging queues can be overloaded with a million requests if the queue lives on a single server. That is why, to scale cloud infrastructure, it is essential to distribute a messaging queue across servers. Kafka is an excellent open-source distributed messaging queue. It provides fault tolerance and durable data ingestion. The reason is that Kafka replicates the queue using a replication factor and stores the data on the persistent database. [6]

3.1. Partition

When a single queue is distributed into multiple similar queues, each is called a Partition. The total number of partitions is called a Partition count. Dividing one queue into multiple partitions helps scale the ingestion exponentially.

3.2. Topic

Similar data being ingested is consumed by a group of partitions. This grouping is called a topic.

3.3. Broker

The partitions live on production servers. Each such server holding one or more Partitions is called a Broker.

3.4. Record

Data is stored in the partitions as data items called a Record.

3.5. Producer

A Producer is the infrastructure component that supplies the data to these partitions.

3.6. Consumer

The data in the partitions is picked up by individual services called a Consumer.

3.7. Pub-Sub Model

Pub-Sub model is a short form of the Publisher – Subscriber model. This means the queue will have a publisher to produce the data to be added. And a Subscriber that will consume the data to do intended processing on the ingested data streams. [7]

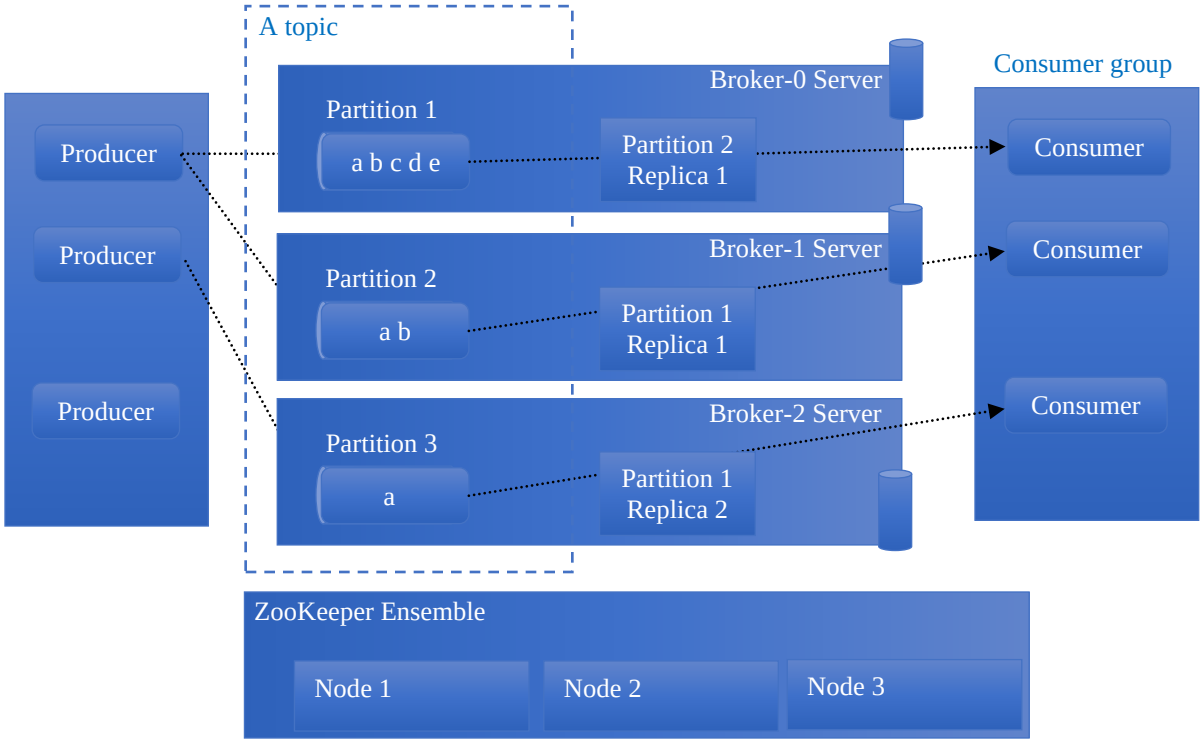

Fig. 1 Kafka architecture [8]

4. Enhancing Cloud Capabilities with Kafka

The need for public cloud providers stems from the vast amount of data that needs processing, mainly in near real-time. For this, multiple distributed messaging queues are required to work together seamlessly.

They need to serve hundreds of microservices on the producer and consumer sides. Also, data needs to be persistent, and processing needs to be fault-tolerant. A single messaging queue cannot achieve all this. Hence, I advocate using distributed Kafka to enhance the performance of public cloud infrastructures. [9]

4.1. Why using Kafka Scales Cloud Infrastructure

Kafka is a distributed messaging queue. It provides the capabilities of topics and partitions. Topics help the producers organize similar data and send it to a defined set of partitions inside the topic.

This way, the consumers do not need to search for relevant data. The consumer can only communicate with the issues of interest and focus on processing the data faster. [10]

4.2. How Data is Published to Kafka

Customers use REST APIs to send data to the front-end microservices. This front-end service will act as the producer for the Kafka partitions. It will have the logic to determine the Kafka partition key and send the incoming data to respective partitions in the topic. [10] There can be multiple front-end services and many issues that talk to services.

4.3. How Data Lives in Kafka

Kafka is a fault-tolerant and durable messaging queue. It replicates the data across multiple partitions using a replication factor. Another strength of Kafka is that the data is stored on persistent databases. So, the data is recovered when a Kafka server restarts.

4.4. How Data is Consumed from Kafka

The partitions in Kafka are subscribed by microservices interested in the data for those partitions. Each subscriber has the location of the data record it needs to read using the concept of Offsets. [11] Subscriber reads the records sequentially using offset. Zero or more microservices can subscribe to each partition.

5. Distributed Kafka in Cloud Infrastructure

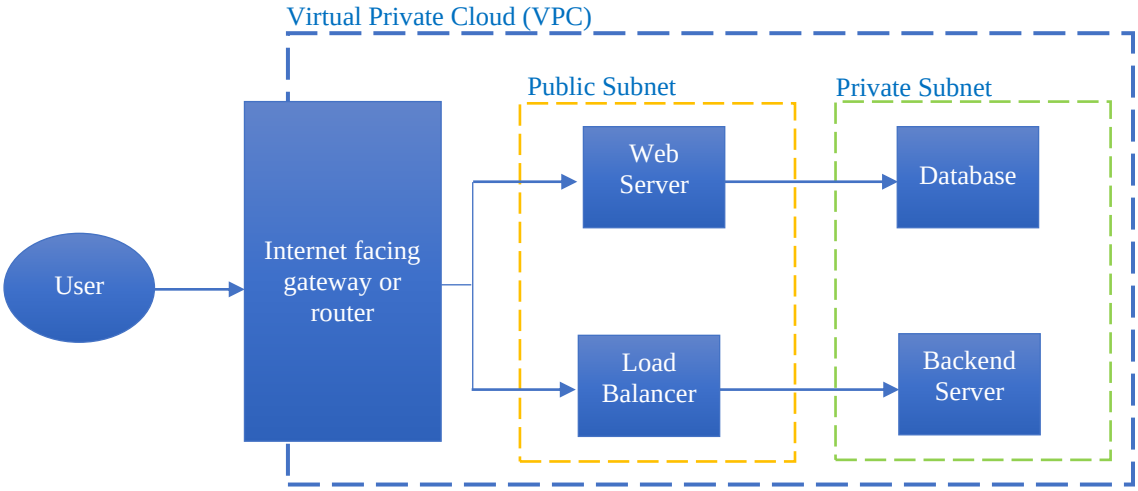

Fig. 2 Cloud computing architecture [12]

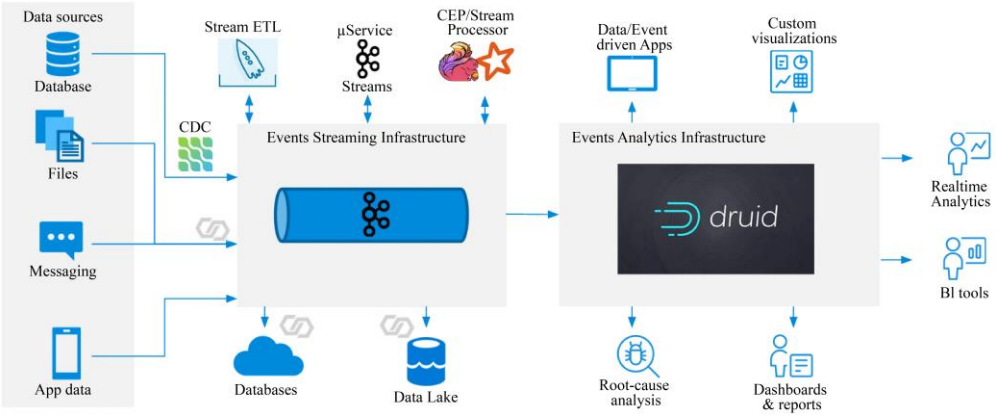

Fig. 3 Kafka in the Cloud Architecture [13]

6. Conclusion

Asynchronous data streams are much of the data being processed by public cloud infrastructure. To efficiently handle data of this massive scale, creative solutions that go multiple steps beyond a simple monolithic architecture or a single messaging queue are needed. Being a distributed, fault-tolerant, and durable messaging queue, Kafka can help scale the public cloud providers' capacity exponentially while keeping the data persistent. Also, given that Kafka is open-source, public cloud providers can easily make a wrapper service around Kafka to cater to their individual customers' needs.

References

[1] Fabio Duarte, Amount of Data Created Daily (2024), The Exploding Topics website, 2023. [Online]. Available: https://explodingtopics.com/blog/data-generated-per-day

[2] Keith D. Foote, A Brief History of Microservices, The Dataversity website, 2021. [Online]. Available: https://www.dataversity.net/a-brief-history-of-microservices/

[3] Eiman Alothali, Hany Alashwal, and Saad Harous, "Data Stream Mining Techniques: A Review," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 2, pp. 728-737, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Ali T. Atieh, "The Next Generation Cloud Technologies: A Review on Distributed Cloud, Fog and Edge Computing and their Opportunities and Challenges," *ResearchBerg Review of Science and Technology*, vol. 1, no. 1, pp. 1-15, 2021. [Google Scholar] [Publisher Link]

[5] Anh-Tuan H. Bui et al., "A Comprehensive Distributed Queue-Based Random Access Framework for mMTC in LTE/LTE-A Networks with Mixed-Type Traffic," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12107-121, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Han Wu, Zhihao Shang, and Katinka Wolter, "Performance Prediction for the Apache Kafka Messaging System," *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Zhangjiajie, China, pp. 154-161, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Jonathan Hasenburg, and David Bermbach, "DisGB: Using Geo-Context Information for Efficient Routing in Geo-Distributed Pub/Sub Systems," 2020 *IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, Leicester, UK, pp. 67-78, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Kafka Overview, IBM Automation - Event-driven Solution - Sharing knowledge, The IBM cloud website, 2022. [Online]. Available: https://ibm-cloud-architecture.github.io/refarch-eda/technology/kafka-overview/.

[9] Mohamed Ouhssini et al., "Distributed Intrusion Detection System in the Cloud Environment Based on Apache Kafka and Apache Spark," *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, Fez, Morocco, pp. 1-6, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] The Apache Kafka Documentation, 2021. [Online]. Available: https://kafka.apache.org/documentation/.

[11] Sean Rooney et al., "Kafka: The Database Inverted, but Not Garbled or Compromised," *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA, pp. 3874-3880, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] What Is Cloud Computing? Definition, Benefits, Types, and Trends, The Spiceworks Website, 2022. [Online]. Available: https://www.spiceworks.com/tech/cloud/articles/what-is-cloud-computing/.

[13] Reading Avro Streams from Confluent Cloud into Apache Druid, The Hellmar Becker Website, 2021. [Online]. Available: https://blog.hellmar-becker.de/2021/10/19/reading-avro-streams-from-confluent-cloud-into-druid/.