Streaming

Streaming data surce that of event that has happened already

It analyse the event in real time

Use cases

Messaging : latency a user maybe waiting on that message or the availability if the system is down ,durability the data may get lost

Streaming :throughput there are lot of event which we need to put in efficient manner

Ordering and satefull processing and batching the data might be in a batch

Storage message and proccess

In order to have complete data platform we need to have a 3 component

Storage:Need to store data and durability and flexibility

Message:Need to have flexible messaging system to have inout and process the data out of it

Compute:We analyze the data perform the function the data and modify it

Messaging model

Producer sends message to the pulsar topics . suscription determine how the topic is going to get consumed. Suscription is a durable resource that will keep position on the topic

There are different types of suscription

Architecture view

It is two layer. Here broker is a stateless service it does not store the data locally. Everything is stored inside bookie and bookkeeper. In this we can add brooker and bookeeper independently based on our need

Segment centric

Pulsar brokers can send data to consumers and accept data from producers individually. In other words, if the load gets too high, you can simply solve this by adding another broker.

Segment vs partition

In its essence, Kafka doesn’t support stateless brokers as each broker uses the complete log for each of its partitions. If any of those brokers fails and the workload gets too high, the user won’t be able to simply solve this by adding another broker.

Tiered storage

Example : I we are in cloud environment then we might need to keep the few segment in bookeeper for some hours and rest of the segment can be pushed to AWS S3 and this is possible due to segment.

Schema registry

Like if we think about stream then We need to know the type of data is there so we have added a schema registery which is based on bookkeeper

**geo-replication**

Geo-replication is the replication of persistently stored message data across multiple clusters of a Pulsar instance.

In this diagram, whenever **P1**, **P2**, and **P3** producers publish messages to the **T1** topic on **Cluster-A**, **Cluster-B**, and **Cluster-C** clusters respectively, those messages are instantly replicated across clusters. Once the messages are replicated, **C1** and **C2** consumers can consume those messages from their respective clusters.

Without geo-replication, **C1** and **C2** consumers are not able to consume messages that **P3** producer publishes.

**Pulsar Functions overview**

**Pulsar Functions** are lightweight compute processes that

* consume messages from one or more Pulsar topics,
* apply a user-supplied processing logic to each message,
* publish the results of the computation to another topic.

Pulsor IO

It shows how we get data in and out of pulsor

It is framework based on pulsor function on top of which we can write connectors very quickly

**Sources** feed data into Pulsar topic from other systems.

**Sinks** are fed data from Pulsar topic.