UPPSALA
UNIVERSITET

Data Engineering II

Assignment 1

Jinglin Gao

April 13, 2023

1 Task 2

For Task 2 Part A we need to create a consumer and a producer on a single machine. Basically, just follow the instruction and create new 'tmux' window to run consumer and producer while the pulsar is running background. We use this command to connect them to pulsar:

```
client = pulsar.Client('pulsar://localhost:6650')
```

For Part B since we need to set up two virtual machines and let Apache pulsar and producer runs on virtual machine A and consumer runs on virtual machine B, we need to connect to pulsar with its actural IP address.

2 Task 3

2.1 Q1

Apache Pulsar is an all-in-one messaging and streaming platform. Its layered architecture allows rapid scaling across hundreds of nodes, without data reshuffling. This architecture is more flexible compared to Kafka. In addition, Apache Pulsar offers multi-tenancy support, which allows you to share resources across different teams or applications. Kafka does not offer this feature, which can lead to resource contention and slower performance. Also, Apache Pulsar offers several advantages including seamless horizontal scalability, consistent performance (even under heavy loads), and the ability to handle both streaming and queuing workloads compared to other data stream framework.[1]

2.2 Q2

Batch processing is the processing of transactions in a group or batch. No user interaction is required once batch processing is underway.

Unlike traditional batch processing which involves the processing of static data, streaming or online processing involves processing dynamic or continuous data. Extracting meaningful and timely insights from unbounded data is very challenging. Batch processing systems suffer from latency problems due to the need to collect input data into batches before it can be processed. This is the main problem with batch processing cause the ability to handle and process continuous streams of data under very short delays is becoming an essential part of building today's data-driven organizations.

Modern Data Stream Processing Systems (DSPS) try to combine batch and stream processing capabilities into single or multiple parallel data processing pipelines. The goal of DSPS is to overcome the latency by processing big data volumes and providing useful insights into the data prior to saving it to long-term storage. It processes the live, raw data immediately as it arrives and meets the challenges of incremental processing, scalability, and fault tolerance.[2]

2.3 Q3

Pulsar is built on the publish-subscribe pattern where producers publish messages to topics and consumers subscribe to those topics, process incoming messages, and send acknowledgments to the broker when processing is finished.[3]

The advantage of this messaging pattern is that it allows for scalability and flexibility in message delivery. Apache Pulsar supports four different subscription types: exclusive, failover, shared and key shared. It also supports multiple subscriptions on a single topic. Additionally, this pattern allows for asynchronous communication between producers and consumers, which can help improve performance and reduce latency.

2.4 Q4

The subscription mode indicates the cursor type. There are two kinds of subscription modes: Durable and NonDurable. Durable is the default subscription mode. If durable is chosen, the cursor is durable, which retains messages and persists the current position. Reader's subscription mode is NonDurable in nature and it does not prevent data in a topic from being deleted.

A subscription is a named configuration rule that determines how messages are delivered to consumers. Four subscription types are available in Pulsar: exclusive, shared, failover, and key_shared.

Exclusive: In the Exclusive type, only a single consumer is allowed to attach to the subscription. If multiple consumers subscribe to a topic using the same subscription, an error occurs. If the topic is partitioned, all partitions will be consumed by the single consumer allowed to be connected to the subscription.

Failover: In the Failover type, multiple consumers can attach to the same subscription. A master consumer is picked for a non-partitioned topic or each partition of a partitioned topic and receives messages. When the master consumer disconnects, all messages are delivered to the next consumer in line.

Shared: In shared or round-robin type, multiple consumers can attach to the same subscription. Messages are delivered in a round-robin distribution across consumers, and any given message is delivered to only one consumer. When a consumer disconnects, all the messages that were sent to it and not acknowledged will be rescheduled for sending to the remaining consumers.

Key_Shared: In the Key_Shared type, multiple consumers can attach to the same subscription. Messages are delivered in distribution across consumers and messages with the same key or same ordering key are delivered to only one consumer. No matter how many times the message is re-delivered, it is delivered to the same consumer.

2.5 Q5

Zookeeper is a top-level software developed by Apache that acts as a centralized service and is used to maintain naming and configuration data and to provide flexible and robust synchronization within distributed systems. For Pulsar, Zookeeper is responsible for a wide variety of configuration-related and coordination-related tasks.

It is possible to ensure reliable without Zookeeper in streaming frameworks such as Pulsar or Kafka. There are some plans to simplify Apache Pulsar deployments - Pulsar Improvement Plan - to eliminate the Zookeeper dependency and replace it with a pluggable framework.[4]

2.6 Q6

At the highest level, a Pulsar instance is composed of one or more Pulsar clusters.

In a Pulsar cluster:
1. One or more brokers handles and load balances incoming messages from producers, dispatches messages to consumers, communicates with the Pulsar configuration store to handle various coordination tasks, stores messages in BookKeeper instances, relies on a cluster-specific ZooKeeper cluster for certain tasks, and more.
2. A BookKeeper cluster consisting of one or more bookies handles persistent storage of messages.
3. A ZooKeeper cluster specific to that cluster handles coordination tasks between Pulsar clusters.

Brokers: The Pulsar message broker is a stateless component.
Clusters: A Pulsar instance consistes of one ore more Pulsar clusters.
Metadata store: The Pulsar metadata store maintains all the metadata of a Pulsar cluster, such as topic metadata, schema, broker load data, and so on.
Configuration store: The configuration store maintains all the configurations of a Pulsar instance, such as clusters, tenants, namespaces, partitioned topic-related configurations, and so on.
Persistent storage: Pulsar provides guaranteed message delivery for applications. If a message successfully reaches a Pulsar broker, it will be delivered to its intended target.
Apache BookKeeper: Pulsar uses a system called Apache BookKeeper for persistent message storage.
Ledgers: A ledger is an append-only data structure with a single writer that is assigned to multiple BookKeeper storage nodes, or bookies. Ledger entries are replicated to multiple bookies.
Replicators: To support geo-replication on global topics, the broker manages replicators that tail the entries published in the local region and republish them to the remote region using the Pulsar Java client library1.

2.7 Q7

The Pulsar message broker is a stateless component that's primarily responsible for running two other components: One is an HTTP server that exposes a REST API for both administrative tasks and topic lookup for producers and consumers; the other is a dispatcher, which is an asynchronous TCP server over a custom binary protocol used for all data transfers.

2.8 Q8

Tenants can be spread across clusters and can each have their own authentication and authorization scheme applied to them. They are also the administrative unit at which storage quotas, message TTL, and isolation policies can be managed. Tenants are used to isolate resources such as topics, namespaces, and policies. A tenant can have multiple namespaces and topics.

2.9 Q9

Pulsar offers topic compaction. When you run compaction on a topic, Pulsar goes through a topic's backlog and removes messages that are obscured by later messages. If there is no log compaction in Apache Pulsar, then the topic will continue to store all messages with the same key. This can lead to an increase in storage usage and slower performance.

3 Task 4

3.1 Q1

In this mode, if we try to process a large amount of data, if the connection is interrupted during the transfer, we will lose all the information. Another problem is that huge data may exceed the memory capacity and cause the program to crash.

3.2 Q2

By redesigning the program structure, we decided to use the producer to process the input data and split the information into individual words based on whitespace before sending to the broker.

The consumer will receive information from the broker, and process and print the information. We define the conversion function in the consumer file, decode the information obtained by the consumer into an editable format, and then process and output the result.

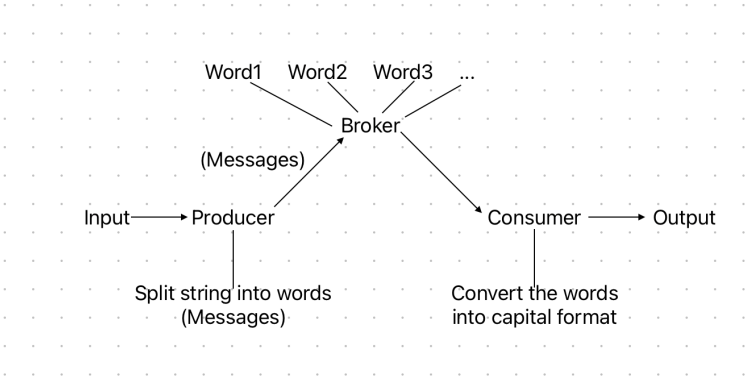

Figure 1: Demonstrate of the architecture

References

[1] DataStax. *Apache Pulsar*. 2023. URL: https://www.datastax.com/guides/apache-pulsar.

[2] Haruna Isah **andothers**. "A Survey of Distributed Data Stream Processing Frameworks". **in***IEEE Access*: 7 (2019), **pages** 154300–154316. DOI: 10.1109/ACCESS.2019.2946884.

[3] Apache Pulsar. *Apache Pulsar Documentation*. 2023. URL: https://pulsar.apache.org/docs/2.11.x/concepts-messaging/.

[4] StreamNative. *Moving Toward ZooKeeper-less Apache Pulsar*. https://streamnative.io/blog/moving-toward-zookeeper-less-apache-pulsar. Accessed on 6 April 2023. 2022.