Kafka

Kafka is used to decouple pipelines, where a bunch of producers use Kafka as a broker to pass data onto a set of consumers. The producers are called publishers, and the consumers are called subscribers, with Kafka as the communication infrastructure in the middle.

A publisher publishes messages to the communication infrastructure, and a subscriber subscribes to a category of messages.

Kafka does message passing between publishers and subscribers with

1. Topic based systems
    a. Publishers send messages with topic labels
    b. Subscribers subscribe to relevant topics
2. Content based systems
    a. Subscribers define a matching criteria
    b. They receive all matching messages

Pros of Kafka

1. No hard wired connection between publishers and subscribers
2. Easy to scale up on both sides

Cons of Kafka

1. Design and maintenance of topics is high, increasing the performance overhead by the communication infrastructure

A Kafka message contains

1. CRC
2. Magic
3. Attributes
4. Key Length
5. Key Message
6. Message Length
7. Message content

Usually, a kafka message is retained for a fixed duration.

Scaling in Kafka happens through partitions, where partitions allow logs greater than disk size, increasing throughput. These partitions are distributed over servers, and a publisher can balance the load based on either round robin or key hashing techniques, with each message having an offset.

Partitions can be replicated, where the leader performs all the reads and writes and the follower replicates. If the leader fails, a replica takes its place.

Synchronization in a kafka cluster happens after a vote is taken among all the nodes in the system, called a quorum.

A consumer group is typically multiple instances of an application, and partitions deliver messages to one of the group members, providing load balancing.