Underlying principles

Must be fast

Must be scalable

Must be distributed

Must be persistent

Cluster is made up of multiple distributed servers, each of which is called a broker

Data

The atomic unit of data in Kafka is the Mesage.

Data is represented by a message

Messages contain a header and a byte[] payload

Data storage

Messages published to a topic are stored in a log file.

Log files are themselves partitioned

Log partitioning enables…

data storage to grow beyond what could otherwise be stored on a single broker.

increased throughput as contention for data access is reduced since data is stored in multiple places, reducing concurrent access to any single log files, instead distributing access across the log partitions.

Each log partition is controlled by a leader broker, with a configurable number of followers acting as backups in case the leader fails.

Each log partition maps to a directory

Each log directory may consist of multiple log segments

For each log segment, there will be an index file and a data file

Each log file consists of a sequence of log entries

Each log entry consists of an integer 1+4+N specifying the length of the message payload followed by the 1+4+N-byte payload

The log entry length accounts for the following:

1 byte for magic value

4 bytes crc

N bytes for the actual message bytes

Each message (log-entry) is identified by an 8-byte id representing the offset of the start of this message in the stream of all messages ever sent to the partition

Log files are named according to the offset of the first message contained

Replication

TODO

Broker discovery

Writing data

Data is written serially to the end of log files

Data is flushed to disk based on configuration

How log writes are configured determines the possible data loss in the event of a crash

The more frequently data is flushed to disk, the less amount of data can be lost.

Log files are rolled over according to configuration

Reading data

Reading consists of requesting a message to start reading at, determining which log segment the message is contained in, and then reading a chunk of bytes.

The message is indicated by the 8-byte offset id.

The chunk-size is based on configuration.

In the event a message isn’t fully contained in the chunk, the read is retried doubling the chunk-size.

Data may also be read on a latest published basis, which removes the message-id aspect.

Clients

Java client is bundled with Kafka

Clients may act either as producers or consumers

Producers

Clients specify which partition to write data to

Partitions can be decided upon based on some type of load-balancing algorithm or based on a more semantic logic.

Using a semantic partitioning strategy allows consumers to deterministically know which server to request data from.

Kafka allows clients to determine which broker is the leader for a given partition

Kafka supports batching of writes

Consumers issue fetch requests to partition leaders

Fetch request indicates log offset of first message desired.

Kafka provides a chunk back to client starting at the requested position

Push / Pull

Producer clients push messages to Kafka

Consumer clients pull messages from kafka