Design of Topics and Partitions

SciSpike

Topic Design

- Name
- Schema
- Payload (Data)
- Key
- Number of partitions
- Number of replicas

The Short Story

- DevOps concerns
  - Bandwidth consumption → Size of messages, `serdes`, etc.
  - Fault tolerance and availability → Size of cluster, replication factor, etc.
  - Performance → Partitions, message size, cost of serialization, etc.
- Producer concerns
  - Ease of production → Clear schema, cost of serialization, delivery guarantees, etc.
- Consumer concerns
  - Can I subscribe to only what I need → Topics and partitions
  - Latency → Cluster size, performance, etc.

How Topics and Partitions Influence Concerns?

- Topic topology
  - Schema (structure, format, etc.)
  - Temporal constraints (frequency, triggers, etc.)
  - Do you use topics to allow for fine grain subscriptions?
- Partitions
  - Determines throughput (but not without cost)
  - Can be used for fine-grained subscription (requires use of low level API)
- Recommendation
  - Use topics to convey semantic
  - Use partition to control throughput

Name

- Descriptive name
- Don't hardcode the name all over your application!
- Rule of thumb:
  - Use a longer name that is easy to understand

Schema

- JSON
  - Common choice
  - Not the most efficient

- Apache Avro
  - Binary format
  - Compression
  - Schema evolution
  - Dynamic typing (no code generation needed for serialization)

Partitions and Throughput

- Unit of parallelism: topic partition
- Writes to different partitions done in parallel
- Consumer: one thread get a single partition's data
- Consumer parallelism: bounded by the number of consumed partitions
- Throughput on a producer is a function of:
  - Batching size
  - Compression codec
  - Acknowledgement type
  - Replication factor
- Consumer throughput is a function of the message processing logic

Overpartitioning

- Problem: Increasing the number of partition and message ordering
  - If messages have keys, increasing the number of partitions may cause problems
  - Kafka maps a message to a partition based on the hash of the key
  - Messages with the same key go to the same partition
  - If we increase the number of partitions, this does not hold
    - Messages with the same key may for the retention period appear in multiple paritions
- Therefore:
  - Overpartition for a situation you expect in future

Too Many Partitions

- Each partition maps to a directory in the broker
  - 2 files: index, actual data
- You may need to configure the open file handle limit
  - Configuration
  - Seen in production > 30K open file handles / broker

Partitions and Availability

- Intra-cluster replication
- A partition can have multiple replicas, each on a different broker
- Clean broker shutdown: the controller moves the leaders off the broker that is shutting down
  - Takes a couple of ms
- Unclean shutdown: the loss of availability is dependent on the number of partitions
  - All replicas become unavailable at the same time
  - A new leader must be elected
  - Potential unavailability in seconds

Partitions and End-to-End Latency

- Consumers can consume a message after it has been committed
  - Requires replication to all in-sync replicas
- Commit time can be significant
  - Too long for some real-time systems

- A rule of thumb:
  - Limit the number of partitions per broker to:
    100 x (number of brokers) x (replication factor)

Based on Confluent Blog

Partitions and Client Memory

- A producer can set the amount of memory for buffering messages
  - Messages are buffered per partition
  - When buffer is full, messages are sent to the broker
- More partitions: more message buffering in the producer
- If out of memory, producer will block or drop new messages
  - Reconfigure
- Allocate at least a few tens of KB per partition

Summary

- Design topics based on your application semantics
  - Message types
  - Consumer concerns
  - Subscription granularity
- Decide on the number of partitions based on throughput
  - The more partitions, the higher theoretical throughput
  - Not without penalty
    - File handlers, consumer memory, latency, etc
  - Evaluate to overpartition to accommodate future growth