KIP-532: Broker Consumer Lag metrics in size and time

Status

Current state: *Under Discussion*

Discussion thread:

JIRA: *KAFKA-8833*

Motivation

Kafka Consumers use the FetchRequest to read topic partition data at a requested offset. The producer messages are stored as immutable log files on secondary non-volatile storage like hard disk drive (HDD). The consumers consume these messages in a sequential manner. Kafka takes advantage of the Kernel PageCache as it helps to facilitate low latency fetches for reads that get fulfilled via Page Cache. If there is enough free memory, the page is kept in the cache for an indefinite period of time and can then be reused by other processes without accessing the disk.

Therefore it becomes important order to understand the Read I/O pattern of how far away from the tail are consumers fetching the log. This will determine how much physical ram can be allocated for the page cache in order to achieve low latency high through put reads. To determine the memory size for broker, it would be useful to have stats on consumer lag in terms for bytes as well as time.

The read operation are impacted by two kinds of lag,

- time (event time provided by the producer) which is how far back in time the offset fetched is aka time lag
- size of the data that has been produced after what the consumer fetch is being requested for, aka byte lag

Note In many use cases consumers are expected to fetch at the tail end of the log.

Public Interfaces

- Metrics : The consumer lag metrics will be added to the existing metrics interface in the broker

Proposed Changes

The proposal is to have stats on consumer lag in terms for bytes as well as time. We will add broker level metrics aggregated across all partitions. Both metrics will be histogram.

- ConsumerFetchLagTimeInMs: Histogram that will measure fetch log lag using the timestamp of the messages being fetched
- ConsumerFetchLagBytes: Histogram that will measure the fetch log lag by calculating the byte lag of the message being fetched

Byte lag : The relative position of the offset is determined from the fetch info data from logsegment.read() and then newer segments size is added to determine the lag

Time lag : batch maxTimestamp is used to determine the fetched offset timestamp and activesegment.maxtimestampsofar is used.

The changes have been made at the kafka log layer and the following assumption has been made

- In order to measure the byte lag newer segments size sum requires O(N) sum operation, the number of segments are assumed to be small (or none) for the large case as the majority consumer fetch pattern is at tail end (recent)
- Event times have been used for lag measurement which has side effects of user provided timestamp info

Compatibility, Deprecation, and Migration Plan

- *No expected behavior change. The fetch code path will me used for measuring the consumer lag. The current fetch path is O(lg N), where N = number of segments. Since the consumer lag has to me measured in bytes this will have to change to an O(N) over the map that preserves the metadata of the segments at the tail end. Since the majority of the scenarios have a tail end fetch the impact is expected to be minimal.*

Rejected Alternatives

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.