KIP-79 - ListOffsetRequest/ListOffsetResponse v1 and add timestamp search methods to the new consumer

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: *KAFKA-4148*

Released: 0.10.1.0

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

This KIP includes two parts:

Part 1: Introduce ListOffsetRequest v1 to support accurate search based on timestamp.

With KIP-33, the brokers can now search messages by timestamp accurately. To maintain the backwards compatibility, we did not change the behavior of ListOffsetRequest v0. In this KIP, we will introduce ListOffsetRequest v1 to provide more accurate search based on timestamp.

Part 2: Add a search message by timestamps method to o.a.k.c.consumer.Consumer

In SimpleConsumer, users used to be able to search the offsets by timestamp. This method no longer exists in the KafkaConsumer. We want to introduce the method to allow user to search the offsets by timestamp. This is useful in a few cases, for example:

1. From time to time, applications may need to reconsume the messages for a cerntain period, so they will need to rewind the offsets back to, say 6 hours ago, and reconsume all the messages after that.
2. In a multi-datacenter enviroment, users may have different Kafka clusters in each datacenter for disater recovery. If one of the datacenter failed, the applications may need to switch the consumption from one data center to another datacenter. Because the offset between two different Kafka clusters are independent, users cannot use the offsets from the failed datacenter to consume from the DR datacenter. In this case, searching by timestamp will help because the messages should have same timestamp if users are using CreateTime. Even if users are using LogAppendTime, a more granular search based on timestamp can still help reduce the amount of messages to be reconsumed.

Another related feature missing in KafkaConsumer is the access of partitions' high watermark. Typically, users only need the high watermark in order to get the per partition lag. This seems more suitable to be exposed through the metrics.

Although the above two parts can be two separate KIPs, but since part 2 will be heavily depending on part 1 and may also potentially impact the semantic for the ListOffsetRequest v1, it might be better to discuss them together to make sure the wire protocol change can satisfy the various use cases on the consumer side.

Public Interfaces

Add ListOffsetRequest(ListOffsetRequest) v1

```
// ListOffsetRequest v1
ListOffsetRequest => ReplicaId [TopicName [Partition Time ~~MaxNumberOfOffsets~~]]
  ReplicaId => int32
  TopicName => string
  Partition => int32
  Time => int64
  ~~MaxNumberOfOffsets => int32~~
```

```
// ListOffsetResponse v1
ListOffsetResponse => [TopicName [PartitionOffsets]]
  PartitionOffsets => Partition ErrorCode Timestamp [Offset]
  Partition => int32
  ErrorCode => int16
  Timestamp => int64
  Offset => int64
```

Add new Error Code 43 - UnsupportedForMessageFormat

This error code is added to indicate that the requested operation is not supported by the message format version. In this KIP it means the timestamp search operation is not supported by the message format before 0.10.0 when ListOffsetRequest is v1. This error code could also be used in the future when message format evolves again. (e.g. KIP-82).

Add a new method to o.a.k.c.consumer.Consumer to allow user search offsets by timestamp.

offsetForTime() method

```
/**
 * Look up the offsets for the given partitions by timestamp. The returned offset for each partition is the
 * earliest offset whose timestamp is greater than or equals to the given timestamp in the corresponding
partition.
 *
 * This is a blocking call. The consumer does not have to be assigned the partitions.
 *
 * @param timestampsToSearch the mapping from partition to the timestamp to look up.
 * @return For each partition, returns the timestamp and offset of the first message with timestamp greater
 *         than or equal to the target timestamp.
 */
Map<TopicPartition, OffsetAndTimestamp> offsetsForTimes(Map<TopicPartition, Long> timestampsToSearch);

/**
 * Get the earliest available offsets for the given partitions.
 *
 * @param partitions the partitions to get the earliest offsets.
 * @return The earliest available offsets for the given partitions
 */
Map<TopicPartition, Long> beginningOffsets(Collection<TopicPartition> partitions);

/**
 * Get the latest offsets for the given partitions.
 *
 * @param partitions the partitions to get the latest offsets.
 * @return The latest available offsets for the given partitions.
 */
Map<TopicPartition, Long> endOffsets(Collection<TopicPartition> partitions);

public class OffsetAndTimestamp {
        private final long timestamp;
        private final long offset;

        public OffsetAndTimestamp(long offset, long timestamp) {
                this.timestamp = timestamp;
                this.offset = offset;
        }

        public long timestamp() { return timestamp; }
        public long offset() { return offset; }
}
```

Proposed Changes

ListOffsetRequest/ListOffsetResponse v1

Add ListOffsetRequest/ListOffsetResponse v1 with the following changes compared with ListOffsetRequest/ListOffsetResponse v0

1. In ListOffsetRequest, removed MaxNumberOfOffsets field.

2. In ListOffsetResponse, only return timestamp(newly added) and offset of the first message whose timestamp is greater than or equals to the target time.

In ListOffsetRequest/ListOffsetResponse v0, we return a list of offsets which is smaller than or equals to the target time. This was because the timestamp search is based on the segment create time. In order to make sure not to miss any messages, users may have to consume from some earlier segments. After KIP-33, we can accurately find the messages based on the timestamps, so there is no need to return a list of offsets to the users anymore. Arguably we can change the request class name to `OffsetRequest` as well because it no longer "list" the offsets, but it seems not worth breaking the backwards compatibility in this case.

Also, since the messages in format 0.10 and above has timestamp with them, it would be natual to also return the timestamp together with the offset of the messages as the search result.

The semantic of getting the latest offset (target time = -1) and the earliest offset (target time = -2) will be preserved in ListOffsetRequest /ListOffsetResponse v1.

If message format on the broker side is before 0.10, i.e. messages do not have timestamps. ListOffsetRequest v1 will only work if target time = -1 or -2. Any other target time will result in an UnsupportedForMessageFormat (error code 43).

Implementation wise, we will migrate to o.a.k.common.requests.ListOffsetRequest class on the broker side.

Because Kafka request schema uses an `Array` to represent a `Map`, it is possible that users form a `ListOffsetRequest` that contains duplicate partitions with differen timestamps. In this case, the broker will return an InvalidRequestException(error code 42) for that partition in the `ListOffsetResponse.`

offsetsForTimes() method in Consumer

Add a new method to allow users to search messages by timestamp. The input of the method is a mapping from partitions to the target times. The output of the method is a mapping from partitions to the timestamps and offsets of the messages whose timestamps are greater than or equal to the given target time of each partition.

The most important use case for `offsetsForTimes()` method is to consume from the returned offsets. The following code gives an example to do that:

consume from timestamp

```
long offset = consumer.offsetsForTimes(Collections.singletonMap(topicPartition, targetTime)).get
(topicPartition).offset;
consumer.seek(topicPartition, offset);
ConsumerRecords records = consumer.poll();
```

The target timestamp have to be non-negative. Otherwise an `IllegalArgumentException` will be thrown. If no message has a timestamp that is greater than or equals to the target time, a `null` will be returned.

beginningOffsets() and endOffsets()

`earlistOffsets()` and `endOffsets()` return the first and last offset for the given partitions.

Compatibility, Deprecation, and Migration Plan

This KIP is a pure addition, so there is no backward compatibility concern.

The new ListOffsetRequest v1 will only be used by new consumer. The old high level consumer will still be using ListOffsetRequest v0. We will not update SimpleConsumer either as we are already in the progress of deprecating it. OffsetRequest(ListOffsetRequest) v0 will be deprecated together with old high level consumer and SimpleConsumer.

Test Plan

Test the new consumer against broker with message format before and after 0.10.

Test `beginningOffsets()`, `endOffsets()`, `offsetsForTimes()` with CreateTime and LogAppendTime

Rejected Alternatives

Using seekToTimestamp() instead of offsetsForTimes()

1. Although in most cases users search offset by time to simply consume from that offset. But there are some cases that users just want to get the offsets for checking and not want to move the offsets. (e.g. people checkpointing offsets outside of Kafka)
2. It is easy for the users to seek to the offset returned by `offsetsForTimes()`
3. A separate `offsetsForTimes()` method allows user to query any topic partition without first get the partition assigned.