KIP-416: Notify SourceTask of ACK'd offsets, metadata

Status

Current state: *replaced by KIP-382: MirrorMaker 2.0*

Discussion thread: here and and here

JIRA: Unable to render Jira issues macro, execution error.

Pull Request: ~~https://github.com/apache/kafka/pull/6171~~ replaced with: KIP-382: MirrorMaker 2.0

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

Motivation

MirrorMaker 2.0 (KIP-382) needs to know the downstream offsets of replicated records in order to provide cross-cluster offset translation. Currently, WorkerSourceTask receives this information from KafkaProducer but throws it away. It's possible that other Connectors may benefit from this change, e.g. see KIP-381, which also proposes to notify SourceTasks of ACK'd records. In particular, this proposal makes it possible to distinguish when records have been durably stored vs when they have been skipped altogether by a SourceConnector.

Public Interfaces

The callback commitRecord() will be overloaded with an extra parameter:

```
public abstract class SourceTask implements Task {
---%<---


// existing method
public void commitRecord(SourceRecord sourceRecord) {
  // nop
}


// new method calls old one by default
public void commitRecord(SourceRecord sourceRecord, RecordMetadata recordMetadata) {
  commitRecord(sourceRecord);
}
---%<---
```

Proposed Changes

Currently, SourceTask includes a commitRecord() callback, which is invoked under these conditions:

- record is ACK'd from producer
- SourceRecord is filtered out from transformation chain, and thus is never sent to Kafka
- SourceTask gives up after retries and skips the SourceRecord.

The new overloaded version will be invoked instead, with recordMetadata null when there is no ACK. To preserve backwards compatibility, the default implementation will call the old method.

Compatibility, Deprecation, and Migration Plan

This is a new callback and won't affect existing code beyond an additional function call.

Rejected Alternatives

- We could introduce a new method name, but overloading an existing method is a lighter touch.
- We could invoke both old and new methods, but this yields a confusing interface, where it is unclear which method to implement.
- SinkRecord has offset and partition fields, so we could potentially use commitRecord(SinkRecord), but this would be a confusing abuse of SinkRecord's semantics. In particular, it would be weird to have SinkRecord in the SourceTask interface.
- We could add kafkaOffset and kafkaPartition fields to SourceRecord, which would be null prior to ACK and then filled in by WorkerSourceTask after ACK. However, this breaks the "value class" semantics of SourceRecord.
- We could extend SourceRecord with AckedSourceRecord or LoggedSourceRecord, but this seems overkill.
- We could avoid using RecordMetadata in the API, and instead include kafkaOffset, kafkaPartition, kafkaTopic, and kafkaTimestamp as parameters to commitRecord(). But this is a lot of parameters, and RecordMatadata is already in the clients API.
- We could pass just the kafkaOffset to commitRecord(), but an offset is mostly meaningless without an associated partition. We could include both kafkaOffset and kafkaPartition, but this doesn't account for transformations that may change the downstream topic name. We could include kafkaTopic as well, but then we might as well include the entire RecordMetadata.