KIP-8 - Add a flush method to the producer API

Status

Current state: Accepted

Discussion thread: here

Original motivation here.

JIRA: KAFKA-1865

Released: 0.8.3

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

Motivation

Currently there is no way to force the sending of all buffered messages in the new Java producer.

Currently if you want to send a batch of messages and don't care about the error or offset you can do that like this:

Current Usage

```
List<ProducerMetadata> results = new ArrayList<ProducerMetadata>();
for(String messageToSend: batch) {
    ProducerMetadata result = producer.send(new ProducerRecord("my-topic", messageToSend));
    results.add(result);
}
for(ProducerMetadata result: results)
    result.get();
```

There are two problems with this usage, first it is sort of annoying to iterate through all the futures to wait until they are all sent.

Second, if you set linger.ms > 0 to encourage batching of messages, which is likely a good idea for this kind of use case, then the second for loop will block for a ms as we will not immediately send the records but wait for more to arrive. Since your code isn't sending any more this waiting is kind of silly.

Public Interfaces

The proposal is to add a new method to the Producer interface:

Flush method

```
    /**
     * Immediately flush any records currently buffered in the producer. This will cause any unsent records to
immediately
     * be available to send regardless of the configured linger.ms setting. The method will block until all
previously
     * sent records have completed sending (either successfully or with an error).
     */
    public void flush();
```

One nuance of this interface is what does it mean when flush is called in a multithreaded use case? Does it block sends in other threads? Is it possible for two threads to call flush at the same time, and if so, what does that mean?

Here is the proposed multi-threaded semantics: flush doesn't block additional send calls from other threads. The semantics will be that any send from any thread that completes before flush initiates will be completed when the flush call completes (flush flushes *all* buffered messages, not just the ones sent by the current thread). Other threads can initiate sends while flush is in progress and these sends won't be blocked and may well be included in the requests resulting from the flush however there is no guarantee one way or another.

There can be multiple flush calls occurring at the same time, each will just block until all the record batches that were in the accumulator at the time that flush call was invoked have been completed.

Currently there is no hard request timeout except what the server enforces, however when a client side timeout is added we can bound the time flush() will take by this timeout since after that time expires the request will be considered failed and hence completed.

Proposed Changes

The full description is pretty much covered in the api description. There is a first pass on implementation here.

Compatibility, Deprecation, and Migration Plan

This is a new producer API and changes the Producer.java interface. We have said that that interface was really for our usage and we make no promise to other implementors of it avoid changes.

No existing users of KafkaProducer should be broken, though.

Rejected Alternative

1. Do nothing and just instruct people to use the futures.
2. Making the signature flush(long timeout, TimeUnit), we decided it is good enough to allow setting a request timeout and having that be the implicit timeout for flush() (since then the requests will be considered failed and hence completed).