Committing and fetching consumer offsets in Kafka

In Kafka releases through 0.8.1.1, consumers commit their offsets to ZooKeeper. ZooKeeper does not scale extremely well (especially for writes) when there are a large number of offsets (i.e., `consumer-count * partition-count`). Fortunately, Kafka now provides an ideal mechanism for storing consumer offsets. Consumers can commit their offsets in Kafka by writing them to a durable (replicated) and highly available topic. Consumers can fetch offsets by reading from this topic (although we provide an in-memory offsets cache for faster access). i.e., offset commits are regular producer requests (which are inexpensive) and offset fetches are fast memory look ups.

The official Kafka documentation describes how the feature works and how to migrate offsets from ZooKeeper to Kafka. This wiki provides sample code that shows how to use the new Kafka-based offset storage mechanism.

Step 1: Discover and connect to the offset manager for a consumer group by issuing a consumer metadata request to any broker

```
import kafka.api.*;
import kafka.cluster.Broker;
import kafka.common.OffsetAndMetadata;
import kafka.common.OffsetMetadataAndError;
import kafka.common.TopicAndPartition;
import kafka.javaapi.ConsumerMetadataResponse;
import kafka.javaapi.OffsetCommitRequest;
import kafka.javaapi.OffsetCommitResponse;
import kafka.javaapi.OffsetFetchRequest;
import kafka.javaapi.OffsetFetchResponse;
import kafka.network.BlockingChannel;
import java.util.*;

...

    try {
        BlockingChannel channel = new BlockingChannel("localhost", 9092,
                BlockingChannel.UseDefaultBufferSize(),
                BlockingChannel.UseDefaultBufferSize(),
                5000 /* read timeout in millis */);
        channel.connect();
        final String MY_GROUP = "demoGroup";
        final String MY_CLIENTID = "demoClientId";
        int correlationId = 0;
        final TopicAndPartition testPartition0 = new TopicAndPartition("demoTopic", 0);
        final TopicAndPartition testPartition1 = new TopicAndPartition("demoTopic", 1);
        channel.send(new ConsumerMetadataRequest(MY_GROUP, ConsumerMetadataRequest.CurrentVersion(),
correlationId++, MY_CLIENTID));
        ConsumerMetadataResponse metadataResponse = ConsumerMetadataResponse.readFrom(channel.receive().
buffer());

        if (metadataResponse.errorCode() == ErrorMapping.NoError()) {
            Broker offsetManager = metadataResponse.coordinator();
            // if the coordinator is different, from the above channel's host then reconnect
            channel.disconnect();
            channel = new BlockingChannel(offsetManager.host(), offsetManager.port(),
                                          BlockingChannel.UseDefaultBufferSize(),
                                          BlockingChannel.UseDefaultBufferSize(),
                                          5000 /* read timeout in millis */);
            channel.connect();
        } else {
            // retry (after backoff)
        }
    }
    catch (IOException e) {
        // retry the query (after backoff)
    }
```

Step 2: Issue the OffsetCommitRequest or OffsetFetchRequest to the offset manager

```
// How to commit offsets

        long now = System.currentTimeMillis();
        Map<TopicAndPartition, OffsetAndMetadata> offsets = new LinkedHashMap<TopicAndPartition,
OffsetAndMetadata>();
        offsets.put(testPartition0, new OffsetAndMetadata(100L, "associated metadata", now));
        offsets.put(testPartition1, new OffsetAndMetadata(200L, "more metadata", now));
        OffsetCommitRequest commitRequest = new OffsetCommitRequest(
                MY_GROUP,
                offsets,
                correlationId++,
                MY_CLIENTID,
                (short) 1 /* version */); // version 1 and above commit to Kafka, version 0 commits to ZooKeeper
        try {
            channel.send(commitRequest.underlying());
            OffsetCommitResponse commitResponse = OffsetCommitResponse.readFrom(channel.receive().buffer());
            if (commitResponse.hasError()) {
                                for (partitionErrorCode: commitResponse.errors().values()) {
                                    if (partitionErrorCode == ErrorMapping.OffsetMetadataTooLargeCode()) {
                                          // You must reduce the size of the metadata if you wish to retry
                                    } else if (partitionErrorCode == ErrorMapping.
NotCoordinatorForConsumerCode() || partitionErrorCode == ErrorMapping.ConsumerCoordinatorNotAvailableCode()) {
                                      channel.disconnect();
                     // Go to step 1 (offset manager has moved) and then retry the commit to the new offset
manager
                                     } else {
                                        // log and retry the commit
                                }
                        }
                           }
                }
                catch (IOException ioe) {
                channel.disconnect();
                    // Go to step 1 and then retry the commit
            }

// How to fetch offsets

                List<TopicAndPartition> partitions = new ArrayList<TopicAndPartition>();
        partitions.add(testPartition0);
        OffsetFetchRequest fetchRequest = new OffsetFetchRequest(
                MY_GROUP,
                partitions,
                (short) 1 /* version */, // version 1 and above fetch from Kafka, version 0 fetches from
ZooKeeper
                correlationId,
                MY_CLIENTID);
                try {
                channel.send(fetchRequest.underlying());
                OffsetFetchResponse fetchResponse = OffsetFetchResponse.readFrom(channel.receive().buffer());
                OffsetMetadataAndError result = fetchResponse.offsets().get(testPartition0);
                short offsetFetchErrorCode = result.error();
            if (offsetFetchErrorCode == ErrorMapping.NotCoordinatorForConsumerCode()) {
                                channel.disconnect();
                                // Go to step 1 and retry the offset fetch
                        } else if (errorCode == ErrorMapping.OffsetsLoadInProgress()) {
                            // retry the offset fetch (after backoff)
                          } else {
                        long retrievedOffset = result.offset();
                        String retrievedMetadata = result.metadata();
                        }
                }
                catch (IOException e) {
                        channel.disconnect();
                        // Go to step 1 and then retry offset fetch after backoff
                }
```