KIP-241 KTable repartition with compacted Topics

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: *"Under Discussion"*

Discussion thread: *here [Change the link from the KIP proposal email archive to your own email thread]*

JIRA: *here [Change the link from KAFKA-1 to your own ticket]*

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

Motivation

When aggregating a KTable we need to repartition the messages by aggregation key to make all of them be processed by the same Task. Instead of using Change<V> serialized into an uncompacted topic we can publish the events into a compacted topic as they are and use the aggregation key to decide on the partition. This has some benefits users might find very helpful.

1. small tables with many update can be much smaller in the repartition topics.
2. debugging repartitioned topics becomes easier because its not required to unserialize a Change<V> but only V itsself
3. One can finally present the aggregator with the key of the record that has to be added or removed from the aggregate

Public Interfaces

Briefly list any new interfaces that will be introduced as part of this proposal or any existing interfaces that will be removed or changed. The purpose of this section is to concisely call out the public contract that will come along with this feature.

A public interface is any change to the following:

- *Binary log format*
- *The network protocol and api behavior*
- Any class in the public packages under clientsConfiguration, especially client configuration
    - org/apache/kafka/common/serialization
    - org/apache/kafka/common
    - org/apache/kafka/common/errors
    - org/apache/kafka/clients/producer
    - org/apache/kafka/clients/consumer (eventually, once stable)
- *Monitoring*
- *Command line tools and arguments*
- *Anything else that will likely break existing users in some way when they upgrade*

Proposed Changes

Describe the new thing you want to do in appropriate detail. This may be fairly extensive and have large subsections of its own. Or it may be a few sentences. Use judgement based on the scope of the change.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
- *If we are changing behavior how will we phase out the older behavior?*
- *If we need special migration tools, describe them here.*
- *When will we remove the existing behavior?*

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.