KIP-693: Client-side Circuit Breaker for Partition Write Errors

Status

Current state: *Voting*

Discussion thread: here

JIRA: KAFKA-12793

Motivation

When Kafka is used to build data pipeline in mission critical business scenarios, availability and throughput are the most important operational goals that need to be maintained in presence of transient or permanent local failure. One typical situation that requires Ops intervention is disk failure, some partitions have long write latency caused by extremely high disk utilization; since all partitions share the same buffer under the current producer thread model, the buffer will be filled up quickly and eventually the good partitions are impacted as well. The cluster level success rate and timeout ratio will degrade until the local infrastructure issue is resolved.

One way to mitigate this issue is to add client side mechanism to short circuit problematic partitions during transient failure. Similar approach is applied in other distributed systems and RPC frameworks.

Public Interfaces

New producer config option is added:

- *enable.mute.partition: When set to 'true', Producer will call ProducerInterceptor and Partitioner initialization ProducerMuteManager during construction*

Add a ProducerMuteManager class that manages the partition metadata related to this mechanism

```
/**
 * A class which maintains mute of TopicPartitions. Also keeps the number of TopicPartitions accumulated-
batches and in-flight requests.
 */
public class ProducerMuteManager implements Closeable {

    /**
     * Add mute of TopicPartition
     *
     * @param topicPartition
     */
    public void mute(TopicPartition topicPartition);

    /**
     * Remove muted of TopicPartition
     *
     * @param topicPartition
     */
    public void unmute(TopicPartition topicPartition);


    public boolean isMute(TopicPartition topicPartition);

    /**
     *  Return muted of TopicPartitions
     *
     * @return
     */
    public Set<TopicPartition> getMutedPartitions();

    public void close();

    /**
     *  Return the number of TopicPartition accumulated-batches requests
     *
     * @return
     */
    public Map<TopicPartition, Integer> getAccumulatedBatches();

    void setInFlightBatches(Map<TopicPartition, List<ProducerBatch>> inFlightBatches);

    /**
     * Return the number of TopicPartition in-flight requests
     *
     * @return The request count.
     */
    public Map<TopicPartition, Integer> getInFlightRequestCount();
```

Add a 'initialize' method in Partitioner class

```
public interface Partitioner extends Configurable, Closeable {

    /**
     * This method is called when the Producer is built to initialize the partition mute manager
     *
     * @param producerMuteManager
     */
    default void initialize(ProducerMuteManager producerMuteManager) {
    }

    int partition(String topic, Object key, byte[] keyBytes, Object value, byte[] valueBytes, Cluster cluster);
}
```

Add a 'initialize' method in ProducerInterceptor class

```
public interface ProducerInterceptor extends Configurable, Closeable {

    /**
     * This method is called when the Producer is built to initialize the partition mute manager
     *
     * @param producerMuteManager
     */
    default void initialize(ProducerMuteManager producerMuteManager) {
    }

    public ProducerRecord<K, V> onSend(ProducerRecord<K, V> record);

    public void onAcknowledgement(RecordMetadata metadata, Exception exception);
}
```

Proposed Changes

We propose to add a configuration driven circuit breaking mechanism that allows Kafka client to ‘mute’ partitions when certain condition is met. The mechanism adds callbacks in Sender class workflow that allows to filtering partitions based on certain policy.

The client can choose proper implementation that fits a special failure scenario, Client-side custom implementation of Partitioner and ProducerInterceptor

- Customize the implementation of ProducerInterceptor, and choose the strategy to mute partitions.
- Customize the implementation of Partitioner, and choose the strategy to filtering partitions.

Muting partitions have impact when the topic contains keyed message as messages will be written to more than one partitions during period of recovery. We believe this can be an explicit trade-off the application makes between availability and message ordering.

Compatibility, Deprecation, and Migration Plan

Rejected Alternatives

The proposed solution is only beneficial to applications with Kafka clients upgraded to the new version. Large organizations almost surely have mixed clients which will not all be protected. Similar mechanism can also be implemented on the server side and benefit all clients regardless of their version. We argue that client-side circuit breaking and server side broker high availability are complementary instead of conflicting. On one hand it is not likely (or extremely expensive) to implement broker HA in the control plane; on the other hand we have also often seen client side mechanism used to mitigate network problem between client and broker.