KIP-807: Refactor KafkaStreams exposed metadata hierarchy

Status

Current state: "Under Discussion"

Discussion thread: *here*

JIRA: KAFKA-12370

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

Motivation

Currently, there are different methods in `KafkaStreams` that expose different type of Metadata classes. Several use cases exists in which a user would need to retrieve all, or several, of the different Metadata classes.

At the moment of this KIP, we have the following Metadata classes:

- `StreamsMetadata`
- `TaskMetadata`
- `ThreadMetadata`
- `KeyQueryMetadata`

And the `KafkaStreams` API has the following methods:

- `metadataForLocalThreads`, which returns a set of `ThreadMetadata`
- `queryMetadataForKey`, which returns a `KeyQueryMetadata`
- `streamsMetadataForStore`, which returns a set of `StreamsMetadata`
- `metadataForAllStreamsClients`, which returns a set of `StreamsMetadata`

The motivation of this KIP is to simplify the API, by creating a hierarchy of the Metadata classes and consistently returning the one in the top level.

Reasoning for the new hierarchy of the Metadata classes (taken from the Jira issue):

- `StreamsMetadata` represent the metadata for the client, which includes the set of ThreadMetadata for its existing thread and the set of TaskMetadata for active and standby tasks assigned to this client, plus client metadata including hostInfo, embedded client ids.
- `ThreadMetadata` includes name, state, the set of TaskMetadata for currently assigned tasks.
- `TaskMetadata` includes the name (including the sub-topology id and the partition id), the state, the corresponding sub-topology description (including the state store names, source topic names).
- `KeyQueryMetadata` could be deprecated and instead use a combination of the previous ones.

As described in the Jira task:

> To illustrate as an example, to find out who are the current active host / standby hosts of a specific store, we would call streamsMetadataForStore, and for each returned StreamsMetadata we loop over their contained TaskMetadata for active / standby, and filter by its corresponding sub-topology's description's contained store name.

Public Interfaces

Deprecate methods in `KafkaStreams` returning metadata classes that are not `StreamsMetadata`, and create new ones with similar semantics that would return `StreamsMetadata` .

org.apache.kafka.streams.KafkaStreams

```
public class KafkaStreams implements AutoCloseable {

        ...
        /**
     * Finds the metadata containing the active hosts and standby hosts where the key being queried would
reside.
     *
     * @param storeName     the {@code storeName} to find metadata for
     * @param key           the key to find metadata for
     * @param keySerializer serializer for the key
     * @param <K>           key type
     * Returns {@link KeyQueryMetadata} containing all metadata about hosting the given key for the given store,
     * or {@code null} if no matching metadata could be found.
     * @deprecated since 3.2.0. Use {@link #metadataForKey(String, Object, Serializer)} instead.
     */
    @Deprecated
    public <K> KeyQueryMetadata queryMetadataForKey(final String storeName,
                                                    final K key,
                                                    final Serializer<K> keySerializer) {
                ...
    }

    /**
     * Finds the metadata containing the active hosts and standby hosts where the key being queried would
reside.
     *
     * @param storeName     the {@code storeName} to find metadata for
     * @param key           the key to find metadata for
     * @param partitioner the partitioner to be use to locate the host for the key
     * @param <K>           key type
     * Returns {@link KeyQueryMetadata} containing all metadata about hosting the given key for the given
store, using the
     * the supplied partitioner, or {@code null} if no matching metadata could be found.
     * @deprecated since 3.2.0. Use {@link #metadataForKey(String, Object, StreamPartitioner)} instead.
     */
    @Deprecated
    public <K> KeyQueryMetadata queryMetadataForKey(final String storeName,
                                                    final K key,
                                                    final StreamPartitioner<? super K, ?> partitioner) {
                ...
        }


    /**
     * Finds the metadata containing the active hosts and standby hosts where the key being queried would
reside.
     *
     * @param storeName     the {@code storeName} to find metadata for
     * @param key           the key to find metadata for
     * @param keySerializer serializer for the key
     * @param <K>           key type
     * Returns a collection of {@link StreamsMetadata} containing all metadata about hosting the given key for
the given
     * store, or {@code null} if no matching metadata could be found.
     */
    public <K> Collection<StreamsMetadata> metadataForKey(final String storeName,
                                                          final K key,
                                                          final Serializer<K> keySerializer) {
                ...
    }

    /**
     * Finds the metadata containing the active hosts and standby hosts where the key being queried would
reside.
     *
     * @param storeName     the {@code storeName} to find metadata for
     * @param key           the key to find metadata for
     * @param partitioner the partitioner to be use to locate the host for the key
     * @param <K>           key type
     * Returns a collection {@link StreamsMetadata} containing all metadata about hosting the given key for the
given
     * store, using the the supplied partitioner, or {@code null} if no matching metadata could be found.
     */
    public <K> Collection<StreamsMetadata> metadataForKey(final String storeName,
                                                          final K key,
                                                          final StreamPartitioner<? super K, ?> partitioner) {
        ...
    }

        /**
     * Returns runtime information about the local threads of this {@link KafkaStreams} instance.
     *
     * @return the set of {@link ThreadMetadata}.
     * @deprecated since 3.2.0. Use {@link #localMetadata()} instead.
     */
    @Deprecated
    public Set<ThreadMetadata> metadataForLocalThreads() {
        ...
    }

    /**
     * Returns metadata about the local {@code KafkaStreams} instance.
     * Note: this is a point in time view and it may change due to partition reassignment.
     *
     * @return {@link StreamsMetadata} for this local {@code KafkaStreams} instance.
     */
    public StreamsMetadata localMetadata() {
        ...
    }

        ...

}
```

Extend the `StreamsMetadata` API to include `ThreadMetadata` and `TasksMetadata`.

org.apache.kafka.streams.StreamsMetadata

```
/**
 * Metadata of a Kafka Streams client.
 */
public interface StreamsMetadata {

        ...

        /**
     *
     * @return metadata of this client threads
     */
    Set<ThreadMetadata> threadMetadata();

    /**
     * Metadata of all active tasks assigned to this client.
     *
     * @return metadata of the active tasks
     */
    Set<TaskMetadata> activeTasks();

    /**
     * Metadata of all standby tasks assigned to this client.
     *
     * @return metadata of the standby tasks

     */
    Set<TaskMetadata> standbyTasks();

        ...

}
```

Extend `TaskMetadata` API to include the state and the store of given task.

org.apache.kafka.streams.TaskMetadata

```
public interface TaskMetadata {

    ...

    /**
     * State of the given task
     *
     * @return a String representing the task state
     */
    String state();

    /**
     * Names of the state stores assigned to the given task
     *
     * @return names of the state stores assigned to the given task
     */
    Set<String> stateStoreNames();

        ...
}
```

Deprecate `KeyQueryMetadata` class in favour of using `StreamsMetadata` and `TaskMetadata`.

org.apache.kafka.streams.KeyQueryMetadata

```
/**
 * Represents all the metadata related to a key, where a particular key resides in a {@link KafkaStreams}
application.
 * It contains the active {@link HostInfo} and a set of standby {@link HostInfo}s, denoting the instances where
the key resides.
 * It also contains the partition number where the key belongs, which could be useful when used in conjunction
with other APIs.
 * e.g: Relating with lags for that store partition.
 * NOTE: This is a point in time view. It may change as rebalances happen.
 * @deprecated since 3.2.0. Use {@link StreamsMetadata instead}
 */
@Deprecated
public class KeyQueryMetadata {
        ...
}
```

Proposed Changes

`StreamsMetadata` will become the central and principal class in when it comes to retrieving metadata for Streams. Through this class, one will be able to access all relevant metadata (streams, tasks and threads ones). As described before, changes for this class include adding sets for its `ThreadMetadata`, and `TaskMetadata` sets for its active and standby tasks.

`TaskMetadta` class will be extended to include the task's state and the stores assigned to the given task.

To keep compatibility, old methods in `KafkaStreams returning Set<ThreadMeadata> and KeyQueryMetadata` will be deprecated (and deleted in subsequent releases), while new methods returning `Set<StreamsMetadata>` will be introduced.

The deprecated methods are only used, at the moment, within the test classes or internal classes which can be safely migrated and rewritten to use the newly introduced methods.

The internal StreamsMetadataState can be safely cleaned by removing usages of `KeyQueryMetadata`.

Compatibility, Deprecation, and Migration Plan

Changes are source compatible as old methods and classes are only deprecated and not deleted. Deprecated methods will be deleted in subsequent releases.

List of actions users would need to take to migrate to this version:

- Users of `Streams#keyQueryMetadataForKey` should migrate to `Streams#metadataForKey`
    - Subsequently, using `StreamsMetadta` and its full API instead of the deprecated `KeyQueryMetadata`
- Users of `Streams#metadataForLocalThreads` should migrate to `Streams#localMetadata`

Rejected Alternatives

No rejected alternative at the moment.