KIP-562: Allow fetching a key from a single partition rather than iterating over all the stores on an instance

JIRA: ⚠ **Unable to render Jira issues macro, execution error.**

Discussion: **https://www.mail-archive.com/dev@kafka.apache.org/msg104287.html**

Motivation:

Whenever a call is made to get a particular key from a Kafka Streams instance, currently it returns a store wrapper that contains a list of the stores for all the running and restoring/replica(with KIP-535: Allow state stores to serve stale reads during rebalance) on the instance via StreamThreadStateStoreProvider#stores().

When serving queries (like `` `get(key)` `` or `` `range(from, to)` ``), the wrapper actually iterates over all the underlying stores and issues the same query on each one. This is quite inefficient, and more importantly, it disallows some capabilities that KIP-535 intended to provide.

KIP-535 introduced two discovery mechanism so that users could implement a query routing layer, the ability to find out the partition for a specific key, and the ability to find out the locations and freshness of each replica of each partition of a store. Further, it introduced one key mechanism of a resilient query fetch layer, the ability to serve queries from hot-standby replicas and not just running active ones.

What is implicit is that the query routing layer would select an instance from which to fetch each partition of a store that the query spans, and then fan out to execute sub-queries against each such partition on the selected instances. However, the current store() API disallows this last step. Callers are only able to query *all* partitions on the local instance, not one *specific* partition.

Here's an example of how this is a drawback:

Imagine we have a cluster with two instances (A and B), and a store S with two partitions (0 and 1). Imagine further that store S has one active and one standby replica configured. Say, instance A hosts (0-active and 1-standby) and instance B hosts (1-active and 0-standby). Now, suppose the query routing layer wants to query the standby replica (so as not to compete with active processing). This arrangement is currently impossible. What would happen instead is that both instance A and B would return results from both partition 0 and 1, and the query router would have to de-duplicate the results. Plus, it would not achieve the objective to avoid competing with active processing.

To fill this gap, this KIP proposes to allow querying a specific partition of a store, while still preserving the ability to query all local partitions. This would also reduce latencies while querying a particular key from an instance, as it will fetch the key only from the specific store partition where it belongs which would be very helpful in instances containing multiple partitions.

Public Interfaces:

- Adding new class StoreQueryParameters to provide user options to the `QueryableStoreProvider` layer to understand what kind of stores a user wants. It would currently include whether a user is okay with serving stale data and if user already knows what is the partition of the store a user is looking at. Since store name and partition would be a unique combination, a taskId can be generated from this information to return the store for that particular task.

StoreQueryParameters.java

```
package org.apache.kafka.streams;

// Represents all the query options that a user can provide to state what kind of stores it is expecting
public class StoreQueryParameters<T> {

    public static <T> StoreQueryParameters<T> fromNameAndType(final String storeName, final 
QueryableStoreType<T>  queryableStoreType);

    public StoreQueryParameters<T> withPartition(final Integer partition);

    public StoreQueryParameters<T> enableStaleStores();

    public Integer partition();

    public boolean staleStoresEnabled();

    public String storeName();

    public QueryableStoreType<T> queryableStoreType();
}
```

- Changing the `KafkaStreams#store(final String storeName, final QueryableStoreType<T> queryableStoreType, final boolean staleStores)` in favour of the function mentioned below as this one hasn't been released yet.

KafkaStreams.java

```
public class KafkaStreams {
  @Deprecated
  public <T> T store(final String storeName, final QueryableStoreType<T> queryableStoreType);

  // remove (was added via KIP-535 and was never released)
  public <T> T store(final String storeName, final QueryableStoreType<T> queryableStoreType, final boolean 
staleStores);

  // newly added
  public <T> T store(final StoreQueryParameters<T> storeQueryParameters);
}
```

Proposed Changes:

- Add a new public class `StoreQueryParameters` to set options for what kind of stores a user wants.
- Create a taskId from the combination of store name and partition provided by the user.
- In `StreamThreadStateStoreProvider.java` return only the stores for the task requested by the user and also check the condition to return only running stores or standby/recovering stores as well.

Compatibility, Deprecation, and Migration Plan:

- KafkaStreams#store(final String storeName, final QueryableStoreType<T> queryableStoreType, final boolean includeStaleStores) will be changed to the one mentioned in the Public Interfaces changes. Since the mentioned function is not released yet in any version, no deprecation is required.
- Deprecating store(final String storeName, final QueryableStoreType<T> queryableStoreType) method in favour of public <T> T store(final StoreQueryParameters<T> storeQueryParameters) as both store name and queryableStoreType have been added to StoreQueryParameters.

Rejected Alternatives:

- Overload the QueryableStoreProvider#getStore() and StreamThreadStateStoreProvider#stores() with new parameters to pass a list of partitions along with the currently passed flag includeStaleStores.