KIP-326: Schedulable KTable as Graph source

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *here [NOT CREATED YET]*

Motivation

We have faced the following scenario/problem in a lot of situations with KStreams:
- Huge incoming data being processed by numerous application instances
- Need to aggregate different fields whose records span all topic partitions (something like “total amount spent by people aged > 30 yrs” when processing a topic partitioned by userid).

The challenge here is to manage this kind of situation without any bottlenecks. We don't need the “global aggregation” to be processed at each incoming message. On a scenario of 500 instances, each handling 1k messages/s, any single point of aggregation (single partitioned topics, global tables or external databases) would create a bottleneck of 500k messages/s for single threaded/CPU elements.

For this scenario, it is possible to store the partial aggregations on local stores and, from time to time, query those states and aggregate them as a single value, avoiding bottlenecks. This is a way to create a "timed aggregation barrier”.

If we leverage this kind of built-in feature we could greatly enhance the ability of KStreams to better handle the CAP Theorem characteristics, so that one could choose to have Consistency over Availability when needed.

We started this discussion with Matthias J. Sax here: https://issues.apache.org/jira/browse/KAFKA-6953

Public Interfaces

[existing class] org.apache.kafka.streams.StreamBuilder

[added function] KTable<K, V> scheduledTable(String storeName, String scheduleExpression, boolean allInstances)

Proposed Changes

Create a new DSL Source Type with the following characteristics:

- Source parameters: “time schedule", "state store name", bool “global”, where

  - Time Schedule: A crontab like scheduling expression that defines when this source will be triggered

  - state store name: The name of the state store whose K,V tuples will be streamed when the time schedule is triggered

  - all instances: if true, all instances of this state store will be fetched, concatenated and streamed to the underlying Graph. Else, just the local state store instance will be used. When true, only one Graph instance of the application will be trigged (the others will be on stand-by).

Differently from current DSL Source Types, it is not a new incoming message from a Kafka Topic that sources a Graph. The Graph is sourced when time is elapsed, and the messages are generated based on an existing KTable. If the KTable is empty, no messages are sourced. If it has 100 elements, for example, everytime time is elapsed, the Graph is sourced with 100 elements, even if nothing has changed to the KTable.

Compatibility, Deprecation, and Migration Plan

- There won’t be any compatibility issues because this is a brand new feature/interface

Rejected Alternatives

1- Sink all instances' local aggregation result to a Topic with a single partition so that we could have another Graph with a single instance that could aggregate all results

- In this case, if I had 500 instances processing 1000 messages/s each (with no bottlenecks), I would have a single partition topic with 500k messages/s for my single aggregating instance to process that much messages (great bottleneck)

2- Expose a REST endpoint on each application instance so that I can query each instance's local state storage values (through Interactive Queries) and aggregate the results externally (using a database or something else).

3- Create a scheduled Punctuate at the end of the Graph so that we can query (using getAllMetadata) all other instances's locally aggregated values and them aggregate them all and publish to another Kafka Topic from time to time.

- For this to work we created a way so that only one application instance's Punctuate algorithm would perform the calculations (something like a master election through instance ids and metadata) and we had to create a REST proxy on each instance. We have to implement a bunch of things for each transversally aggregated view we need. In a web analytics case, for example, this might span dozens of views (and a lot of code) in a userid partitioned stream: global views per page, views per browser/device, average time in page, average user age, max time in page and so on...

- It would be great if we could solve this problem without the need to add complexity to the system by adding an external database or tool. I would have to publish the aggregated values back to a Kafka topic if I have another Graph that needs that data too (what is common). Having KStreams to INPUT /OUTPUT its state through Kafka Topics is great because of the ecosystem that is already built.