Streaming Window Aggregates for Samza SQL

Milinda Pathirage

06-26-2015

Introduction

Due to inherently unbounded nature of data streams, windowing is used to reduce the context (devide streams into analyzable segments) of a streaming query so that we can compute meaningful results in a bounded set of elements. There are two ways to perform windowing over a data stream:

- Tuple based windows: Window size and advance is based on the number of tuples
- Time based windows: Window size and advance is based on event time or system time

And there are three main types of windows:

- Tumbling window
- Hopping window
- Sliding window

Implementing time based windows are hard mainly because in real world events arrive out-of-order with respect to the event time. So window implementation should focus on handling out of order events. Even under out-of-order arrivals, handling those are easy if the window is explicit. But if windowing (tumbling and sliding) is done as part of the GROUP BY clause as encouraged in Calcite, aggregate logic should handle complexities inherent in data streams in addition to taking care of aggregations.

This document discuss and describe the design of streaming aggregate operator which handles window queries specified as general aggregate queries over data streams. Please note that current extensions to standard SQL only supports tumbling and hopping windows and requires messages with embedded timestamps as a field in the message (or this should be handle during creation of streams via DDL).

Tumbling Windows

Expressed using date arithmetic

```
SELECT STREAM FLOOR(rowtime TO HOUR) AS rowtime,
  productId,
  COUNT(*) AS c,
  SUM(units) AS units
FROM Orders
GROUP BY FLOOR(rowtime TO HOUR), productId;
```

Query Plan

```
LogicalDelta
  StreamingAggregate(group=[{0, 1}], c=[COUNT()], units=[SUM($2)])
    LogicalProject(rowtime=[FLOOR($3, FLAG(HOUR))], productId=[$1], units=[$2])
      StreamScan(table=[[KAFKA, ORDERS]])
```

Expressed using TUMBLE function

```
SELECT STREAM START(rowtime),
  COUNT(*)
FROM Orders
GROUP BY TUMBLE(rowtime, INTERVAL '1' HOUR)
```

Expressed using HOP function

```
SELECT STREAM START(rowtime),
  COUNT(*)
FROM Orders
GROUP BY HOP(rowtime, INTERVAL '1' HOUR, INTERVAL '1' HOUR)
```

Aligned Tumbling Window

TUMBLE does not have an align argument, so you need to use HOP.

```
SELECT STREAM START(rowtime),
  COUNT(*)
FROM Orders
GROUP BY HOP(rowtime,
    INTERVAL '1' HOUR,
    INTERVAL '1' HOUR,
    TIME '0:30')
```

Hopping Windows

Sample Query

```
SELECT STREAM START(rowtime),
  COUNT(*)
FROM Orders
GROUP BY HOP(rowtime,
    INTERVAL '30' MINUTE,
    INTERVAL '1:45' HOUR TO MINUTE)
```