# KIP-923: Add A Grace Period to Stream Table Join



## Status

**Current state**: Accepted

**Discussion thread**: *here*

**JIRA**: *here*

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

## Motivation

The Stream Table join is inflexible how it handles out of order data in its current state. We recently added versioned tables which allow the table side of the a join to be processed in a timestamp aware method. Right now we only have the option to process the stream side in the offset order as the records arrive. However a record that is a better fit might show up in the table later. This semantic gap leads to incorrect output in some cases as the stream can only process data as it comes in and then is joined with whatever is the latest version in the table.

If the table side uses a materialized version store, it can store multiple versions of each record within its defined table history retention period. This proposal would bring the stream side into alignment with that flexibility. By adding a grace period buffer to the stream side it can wait until it is certain that the record in the table side is the right one for its timestamp.

In summary, by buffering the stream side the join will be able to find the correct version of the key in the table for the timestamp.

Small example.

Say we have a versioned table like this and a table history retention time of 10:

| Key | Value at timestamp 1 | val@TS2 | val@TS3 |
|---|---|---|---|
| 1 | a | a | a |
| 2 | b | x | x |
| 3 | c | c | y |

and a stream joining with it like this:

| Key | Value | TS |
|---|---|---|
| 1 | d | 4 |
| 2 | e | 1 |
| 3 | f | 2 |
| 2 | g | 2 |
| 3 | h | 3 |

With these changes and a large enough grace period we would buffer the stream and always have the same join output:

| Key | Join result |
|---|---|
| 2 | eb |
| 3 | fc |
| 2 | gx |
| 3 | hy |
| 1 | da |

Without buffering the stream it could look like this:

| Key | Join result |
|---|---|
| 1 | da |
| 2 | eb |
| 3 | fc |
| 2 | gb |
| 3 | hc |

# Public Interfaces

**Joined.java**

```
public class Joined {
   /**
        * Set the grace period on the stream side of the join. Records will enter a buffer before being
processed. Out of order records in the grace period will be processed in timestamp order. Late records, out of
the grace period, will be executed right as they come in, if it is past the table history retention this could
result in joins on the wrong version or a null join. Long gaps in stream side arriving records will cause
records to be delayed in processing, even resulting in be processed out of the grace period window.
        *
        *
        * @param gracePeriod the duration of the grace period. Must be less than the joining table's history
retention.
        * @return new {@code Joined} instance configured with the gracePeriod
        */
        public Joined withGracePeriod(Duration gracePeriod);

   /**
     * Create an instance of {@code Joined} with key, value, and otherValue {@link Serde} instances.
     * {@code null} values are accepted and will be replaced by the default serdes as defined in
     * config.
     *
     * @param keySerde the key serde to use. If {@code null} the default key serde from config will be
     * used
     * @param valueSerde the value serde to use. If {@code null} the default value serde from config
     * will be used
     * @param otherValueSerde the otherValue serde to use. If {@code null} the default value serde
     * from config will be used
     * @param name the name used as the base for naming components of the join including any
     * repartition topics
         * @param gracePeriod Duration of the stream side buffer
     * @param <K> key type
     * @param <V> value type
     * @param <VO> other value type
     * @return new {@code Joined} instance with the provided serdes
     */
    public static <K, V, VO> Joined<K, V, VO> with(final Serde<K> keySerde,
                                                   final Serde<V> valueSerde,
                                                   final Serde<VO> otherValueSerde,
                                                   final String name,
                                                   final Duration gracePeriod);

    public Duration gracePeriod() {
        return gracePeriod;
    }
}
```

# Proposed Changes

To allow users to configure the buffertime we will add two new apis. The grace period will be set on the Joined object using a duration. This Joined object is currently only used for the stream table join and will be optional to set. The grace period will only affect the stream buffer and the table history retention, will remain unchanged.

If a grace period is not set the join will execute as before, using the same logic in the stream table join node. If a grace period of zero is set the join will execute as a normal join as each record comes it will try to join to the point of time in the versioned table. If the grace period is non zero, the record will enter a stream buffer and will dequeue when the record timestamp is lees than or equal to than stream time minus the grace period.  Late records, out of the grace period, will be executed right as they come in.

The buffer will be checked whenever stream time is advanced. Stream time will be advanced whenever a new record enters the node from the stream. From then the records will be evicted and sent down to the next processor, the join node. The buffer will use an on disk implementation of `TimeOrderedKeyValueBuffer`.

Setting the grace period will only be allowed for already materialized tables with a table history retention that is greater than the stream buffer's grace period. All other configurations will result in an Exception.

When a failure occurs the buffer will try to recover from an OffsetCheckpoint if possible. If not it will reload the buffer from a compacted change-log topic.

Adding a buffer to the stream will increase the latency of the join by at least the grace period. It will also take the time to serialize and store the record as well as retrieve and deserialize.

# Compatibility, Deprecation, and Migration Plan

Old joins will not be affected, in order to use the new feature users will need to set a grace period. Nothing needs to be deprecated.

Changing grace period might cause some capability issues. Decreasing the grace period would result in a lot of records being ejected. Increasing it might cause records to go past the retention time of the table and miss joins it should have made. When changing the grace period it makes sense to update the tables grace period and maybe reprocess data with the new grace period. To keep things simple for the zero duration buffer we will create a dummy store that will never have anything inserted.

# Test Plan

The testing should be covered with unit test and integration test. The only other tests that would be necessary, would be benchmarking to assess the  performance impact on the stream buffer on the join.

# Rejected Alternatives

- Allowing grace period to be set for non versioned tabled. This maybe something we add later, but for now it is not clear if it is desired.
- Auto materializing tables, choosing the table retention is a complicated process and is best controlled by the user.
- Offset order buffer. It might make sense to do this, but it should be part of a follow up kip. The point of adding a buffer to the stream in joins is to process records based on timestamp rather the order they come out of the topic. Adding offset processing seems counterintuitive and out of scope.