HBaseStorage Timestamp Extensions

Version: 0.02, 2012-09-05
Status: proposal
Authors: PROMETHEUS development team, Deutsche Post Direkt GmbH

Overview

HBaseStorage as part of release 0.9.1 of PIG doesn't support timestamp treatment in the form of

- writing a value with a specific timestamp
- getting timestamp information for a value (or a tuple of values)
- retrieving the value (or a tuple of values) which is valid at or before a specific timestamp (snapshot view)

This document specifies extensions to HBaseStorage for appropriate timestamp treatment in respect to these requirements, designed with focus on the ease of subsequent processing with standard PIG functionality. These extensions are especially useful when operating on time series, event based, or incremental data.

API Extensions

The HBaseStorage API is extended by additional options which comply to the following terms:

- The sequence of options doesn't matter.
- There will be no changes to the current behaviour[1] if not using these additional options.
- Additional options can be combined with existing options and don't interfere with the behaviour induced by the existing options.
- An error is raised if a loader option is used with store, and vice versa.

Store Extensions

`-timestamp` *ts* (ts is of type long)

All values are written with timestamp ts.

Load Extensions

`-snapshot`

Per rowkey: For every distinct timestamp used in any cell for that rowkey one tuple is produced which contains the values which are most recent at that timestamp. Values which didn't exist at that timestamp will be returned as *null*[2] (if used in a fixed schema – it isn't part of the resulting map if it is used in a variable schema defined with *cf*:*).

[1] where *current* or *existent* means "in comparison to the functionality of HBaseStorage as delivered in PIG 0.9.1"
[2] as it is defined for HBaseStorage

The timestamp will be returned as field "ts" at the first – or, if `-loadKey true` is specified – at the second position.

Notice that it usually won't make sense to use aggregate functions when operating on data retrieved with `-snapshot` because of multiple data for the same rowkey.

`-snapshotFrom` *ts* (ts is of type long)

This option can only be used in combination with `-snaphot`. It sets the lower timestamp bound for the snapshot data to be retrieved.

The timestamp of the first snapshot per rowkey is set to *ts* even if *ts* is never used as a version information in the data of that rowkey. It is guaranteed that there is at least this one row with timestamp *ts* per rowkey in the result set.

Notice that even whether this option is provided, the whole version history before *ts* of each involved cell has to be examined to build the appropriate snapshot at *ts*.

`-snapshotTo` *ts* (ts is of type long)

This option can only be used in combination with `-snaphot`. It sets the upper timestamp bound for the snapshot data to be retrieved. Values with a timestamp later than *ts* will be ignored when building the output data.

Examples

Suppose we have an HBase table with one rowkey (for simplicity) and

- all columns included in the HBaseStorage loader result set
- -loadKey true
- timestamp: ts*n* is after ts*m* if n > m, i.e. ts30 is after ts20

RowKey	Column Qual. cqA	Column Qual. cqB	Column Qual. cqC
rk	ts10: valA1 ts20: valA2	ts20: valB1 ts30: null (deleted)	ts30: valC1

The following table shows the expected results for

- ts05 (same for all ts < ts10)
- ts15 (same for all ts10 <= ts < ts20)
- ts25 (same for all ts20 <= ts < ts30)
- ts35 (same for all ts >= ts30)

and some combinations of the new options.

-snapshot

RowKey	ts	cqA	cqB	cqC
rk	ts10	valA1	*null*	*null*
rk	ts20	valA2	valB1	*null*
rk	ts30	valA2	*null*	valC1

(result set size per rowkey is identical to number of distinct timestamps used with this row)

-snapshot -snapshotFrom ts05

RowKey	ts	cqA	cqB	cqC
rk	ts05	*null*	*null*	*null*

-snapshot -snapshotFrom ts15

RowKey	ts	cqA	cqB	cqC
rk	ts15	valA1	*null*	*null*
rk	ts20	valA2	valB1	*null*
rk	ts30	valA2	*null*	valC1

-snapshot -snapshotFrom ts25

RowKey	ts	cqA	cqB	cqC
rk	ts25	valA2	valB1	null
rk	ts30	valA2	*null*	valC1

-snapshot -snapshotFrom ts35

RowKey	ts	cqA	cqB	cqC
rk	ts35	valA2	null	valC1

-snapshot –snapshotTo ts05

RowKey	ts	cqA	cqB	cqC

(empty result set)

-snapshot -snapshotTo ts15

RowKey	ts	cqA	cqB	cqC
rk	ts10	valA1	*null*	*null*

-snapshot -snapshotTo ts25

RowKey	ts	cqA	cqB	cqC
rk	ts10	valA1	*null*	*null*
rk	ts20	valA2	valB1	*null*

-snapshot -snapshotTo ts35

RowKey	ts	cqA	cqB	cqC
rk	ts10	valA1	*null*	*null*
rk	ts20	valA2	valB1	*null*
rk	ts30	valA2	*null*	valC1

-snapshot –snapshotFrom ts05 –snapshotTo ts05

RowKey	**ts**	**cqA**	***cqB***	**cqC**
rk	ts05	*null*	*null*	null

-snapshot –snapshotFrom ts05 –snapshotTo ts15

RowKey	**ts**	**cqA**	***cqB***	**cqC**
rk	ts05	*null*	*null*	null
rk	ts10	valA1	*null*	*null*

-snapshot –snapshotFrom ts15 –snapshotTo ts15[3]

RowKey	**ts**	**cqA**	***cqB***	**cqC**
rk	ts15	valA1	*null*	*null*

Final remark

The enhancements described in this document should cover the key issue pictured in PIG-1832 (specify timestamp when storing data). Other improvements are imaginable like providing a time slice based raster in the form of "snapshot at the first day of each month", for example.

[3] This combination of options provides the typical – and most restrictive – snapshot view at a distinct timestamp. The result is one row per rowkey with the most recent data at the given timestamp.