[Draft] Table design for YTS Next Gen--Phoenix Phase I

Drafted 04/02/15

Timeline entity data to be stored

(I = Immutable, M = Mutable, Q = supports query, A = supports aggregation)

- Entity ID (value): I
- Entity Type (value): I
- Metrics (List: Tuple<id, Map: info key -> info, singleData, Map: time -> info, start&end time>): M, Q, A
- Configs (Map: config key -> config info): I, Q
- Children (Map: type->children list): M
- Parent (Tuple<ID, Type>): I
- Events (Set: Tuple<id, Map: info key -> info, timestamp>): M
- CreatedTime (value): I
- ModifiedTime (value): M
- Info (Map: info key -> info): M, Q

- Context: ClusterID, UserID, FlowID, FlowRunID, AppID, entity type, entityID (PK)

In this evaluation phase we focus on basic timeline entity info, metrics, events and configs but temporarily ignore info

Table Schema

Entity data table

Primary key

- Cluster ID
- User ID
- Flow ID
- Flow Run ID
- YARN Application ID
- Entity Type

- Entity ID (we need this to be in the PK so that different containers will have different PK)

Other columns

- Entity Creation Time
- Parent Entity PK (limited to the same cluster, user, flow, flow run)
- Queue Name
- Available config keys
- Available metrics
- Modified time

Parent Entity PK column is used to query all children entity of a timeline entity. In this way we can build the DAG among timeline entities.

Latest Metrics table

Used to list all latest metrics for one PK

Primary key

- logical PK: matches the row from the entity data table

Other columns

- each metric name as a column, we store metrics data in the cell.
- Sample metric name: cpu_user_minute (per-minute values), one per epoch

Metrics precision table

Used to query on precise metrics data for timeline series, indexed by PK+epoch time, **for each application or flow**.

Primary key

- logical PK: matches the row from the entity data table, on **Application level**
- epoch time: server side time for a given metric epoch to start, link to hbase timestamp (Note from Siddharth Wagle: We do this to avoid a full table scan for time range queries. This speeds up reads by utilizing the HFile timestamp. This was done as an optimization to scale the queries on the tall table. Note: Since timestamp appears further down in the row key and we do not have secondary indexes, this speeds up the Scan. PHOENIX-914: Was created to make a feature in Phoenix.)

Other columns

- each metric name as a column, we store metrics data in the cell: raw data for each metric time point, in the form of {t1: v1; t2: v2, ...}.
- Sample metric name: cpu_user_minute (per-minute values), one per epoch

How our current data schema works for metrics:
For now it's only seems to be feasible to store timeline series data on App level, rather than containers level. That's why we have separate tables for "Latest metrics" and "precision" data. The latter is designed for app/flow level time series data.

Available timeline metrics will be stored on a per-PK basis, in the entity data table. Right now we're planning to keep that data to be immutable to reduce metric update traffic.

Config table

Primary key

- logical PK: matches the row from the entity data table

Other columns

- each config info name maps to a value in a cell

The current data schema works for queryable configs directly through the config table. It is also possible to read all available configs for one PK.

Event table

Primary key

- logical PK: matches the row from the entity data table
- timestamp
- Event key

Other columns

- Event info: detailed event info.