Design of HBase schema for ATS v2

(YARN-2928)

Introduction

Proposing a hybrid schema in hbase (mix of vanilla hbase tables and phoenix driven hbase tables) for storing the ATS information.

Overview

- Schema is a collection of vanilla hbase tables and phoenix driven hbase tables
- ATS collector would write to a vanilla hbase table and Phoenix tables are on the aggregation write path.
- SQL like queries (group by, projection and filtering) that are run on flow/user/queue level stats can be efficiently executed with the help of Phoenix
- Having granular data in vanilla hbase tables helps to
    - to cluster relevant data together in columns per column family and entity level data per record
        - having separate column families helps when we are scanning for data in just one column family, for instance, a scan for metrics does not have to jump over config data in that row.
    - set different compressions and TTLs per column family.
    - have a simpler write path by allowing the statistic name to be stored as a dynamic column in hbase, instead of the overhead in manipulating it to fit in as part of row key
- Having summary data in Phoenix tables helps to
    - query with sql capabilities for analyzing the data at flow level or user level or queue level
    - does not blow up the table length since data is at a summary level
- Vanilla HBase tables in this schema proposal are:
    - entity table
    - flow table
    - flow version table
- Phoenix driven tables in this schema proposal are:

- flow summary daily and weekly tables
- user summary daily and weekly tables
- queue summary daily and weekly tables

Description of Tables

1. entity table

- each record contains that entity's info, metrics, config
- **primary key components:** UserID, ClusterID, FlowID, FlowRunID, AppID, entity type, entityID
    i. putting the UserID first helps to distribute writes across the regions in the hbase cluster.
    Pros: avoids single region hotspotting.
    Cons: connections would be open to several region servers during writes from per node ATS
- **column families** for info, metrics and config
    i. config needs to be stored as key value, not as a blob to enable efficient key based querying based on config param name
    ii. storing it in a separate column family helps to avoid scanning over config while reading metrics and vice versa
- metrics are written to with an hbase cell timestamp set to top of the minute or top of the 5 minute interval or whatever is decided. This helps in timeseries storage and retrieval in case of querying at the entity level
    i. the metrics time-series data could have TTL of X days (latest value always retained) to prevent hoarding of data (aggregate data will be available in other tables)
    ii. HBase allows us to set MIN_VERSIONS as well as TTL. If MIN_VERSIONS and TTL conflict, MIN_VERSIONS takes precedence. That means that the last MIN_VERSION number of values will always be retained
- table record example here

2. flow version table

- stores the unique version info seen per flow

- **primary key components:** ClusterID, UserID, FlowID, Version.
- Contains one column family info that has the first seen run id for that version

3. **flow table (aggregation table)**
    - contains statistics per flow run
    - could be populated via co processors triggered on each put in entity table
    - **primary key components**: ClusterID, UserID, FlowID, FlowRunID
    - **column families** for info and metrics (config not present at flow level)
    - example records shown here
4. **flow by application id table**
    - contains a mapping of application to row key in the entity table, this enables lookup of flow based on application id.
5. **flow summary daily table (aggregation table managed by Phoenix)**
    - contains the stats aggregated across all the runs of this flow on this day
    - could be triggered via co-processor with each put in flow table or a cron run once per day to aggregate for yesterday (with catchup functionality in case of backlog etc)
    - **primary key components:** UserID, ClusterID, Top_of_the_Day_Timestamp, FlowID, statistic name
    - example statistic names in the rows in this table: totalApplications, megabytemillis, job counters
    - sample record shown here
6. **flow summary weekly table (aggregation table managed by Phoenix)**
    - contains the stats aggregated across all the runs of this flow during this week
    - could be triggered via co-processor with each put in flow table or a cron run once per week to aggregate for last week (with catchup functionality in case of backlog etc)
    - **primary key components:** UserID, ClusterID, Top_of_the_Week_Timestamp, FlowID, statistic name
    - Columns in this table are exactly the same as the daily table except at the weekly granularity
    - example rows in this table: totalApplications, megabytemillis, job counters

- sample record would be very similar to the one shown here for daily table, except that the timestamp would be that of top of the week

7. **user (or queue) summary daily table (aggregation table managed by Phoenix)**
    - contains stats aggregated across all the runs of all the flows grouped per user (or queue) on that day
    - could be triggered via co-processor with each put in entity table or a cron run once per day to aggregate for yesterday (with catchup functionality in case of backlog etc)
    - **primary key components:** UserID (or queue), ClusterID, Top_of_the_Day_Timestamp, statistic name
    - sample record shown here
8. **user (or queue) summary weekly table (aggregation table managed by Phoenix)**
    - contains stats aggregated across all the runs of all the flows grouped per user (or queue) during that week
    - could be triggered via co-processor with each put in user daily table or a cron run once per week to aggregate for yesterday (with catchup functionality in case of backlog etc)
    - **primary key components:** UserID (or queue), ClusterID, Top_of_the_Week_Timestamp, statistic name
    - sample record very similar to the one shown in the daily table here except that timestamp would be that of the week.

Queries (all answerable by current hRaven)

- UI Based queries:
    - *See UI description below*
    - List all the flows for a given cluster
        - allow for filtering on user, queue, name of flow
        - allow for custom date ranges
    - list all flows which have an application in state ‘running’

        - any state for that matter (For Flows ‘Completed’ state is not deterministic via ATS)
    - For a given application, what counters did it emit? (hits the entity table, counters are column names, row key is well known, it’s a get)
    - Which were the top applications in Queue X last week? (hits the queue weekly table)
    - How did the megabytemillis (any metric) for application X change over time?
    - How was the resource usage in user X’s Hadoop queue over time?
    - How many applications do we run per cluster per day (per user per queue) ?
- Analytical queries:
    - Reducer estimation queries: Given a flow with a version and a user, return x,y,z metrics and a,b,c config params from the last 4 runs
    - Which users wrote more than X TB yesterday? (hits the user daily table)
    - How many applications do we run per cluster per day (per user per queue) ?
    - How many of those are Scalding applications and how many are Pig?
    - Which new applications showed up yesterday?
    - What percentage of jobs have more than X containers?
    - How many jobs have a higher than desirable garbage collection time to cpu time ratio?
    - Which jobs are asking for more memory but actually using very little?

UI description

- Main page shows paginated, tabular list of all flows
    - allows for filtering based on user, queue, name of flow, state.
- Clicking on any row takes you to the next page which shows the latest N runs of that flow along with some statistics of that run
- Clicking on any particular flow run takes you to a page that draws the execution flow of that pig job and shows below it a tabular display of all applications that ran as part of that flow and their statistics
    - allows for filtering based on app id or other criteria
- Clicking on any particular application in that table takes you to the applications detail page which is like the RM or history server page for a yarn application details.

- It has links for details on config and metrics and list of tasks

Example entity table records:

Row key	*column family* **info**	*column family* **metrics**	*column family* **config**
user_A ! cluster_B ! Some_Pig_Script_name ! 1427224700180 1! Application_ID1234 ! entity type ! entityID	*key: value* entity_start_time: 1392993084018 m!megabytemillis: 11777980983440 total_containers: 100	*key: value* gc_millis: 10240124 "org.apache.hadoop.mapreduce.TaskCounter!MAP_OUTPUT_MATERIALIZED_BYTES": 1157231144494 version: 3AK50927836L	*key: value* mapreduce.jobtracker.jobhistory.lru.cache.size: "5" mapreduce.jobtracker.address: "local"

Example Flow table records

rowkey	*column family* **info**	*column family* **metrics**
Cluster_A ! User_X ! Some_Scalding_Script_Name ! 1427224102000	*key : value* flow_name: some_scalding_script_name submit_time: 1427224102000 (submit time of the first job in the flow == smallest submit time seen amongst apps in this flow) finish_time: 1427224699000 (finish time of the last job in the flow == biggest finish	*key : value* gc_millis: 20940124 "org.apache.hadoop.mapreduce.TaskCounter!MAP_OUTPUT_MATERIALIZED_BYTES": 91257931144494

	time seen amongst apps in this flow) version: 3AK50927836L	

Example Flow Summary Daily Table records

row key	*column family* info
User_A ! Cluster_B ! 1427155200000 ! Some_Pig_Script_Name ! megabytemillis	24533697592832
User_ A ! Cluster_B ! 1427155200000 ! Some_Pig_Script_Name ! totalApplications	22
User_A ! Cluster_X ! 1427155200000 ! Some_Pig_Script_Name ! queue	“root.queue1.subqueue2”
User_A ! Cluster_X ! 1424736000000 ! Some_Scalding_Application_Name ! org.apache.hadoop.mapreduce.TaskCounter ! MAP_OUTPUT_MATERIALIZED_BYTES	912517931144494
User_B ! Cluster_B ! 1424736000000 ! Some_Scalding_Application_Name ! totalApplications	16
User_B ! Cluster_X ! 1424736000000 ! Some_Scalding_Application_Name ! queue	“root.queueX”

Example User Summary daily table record

row key	*column family* info
User_12 ! Cluster_ABC ! 1424736000000 ! megabytemillis	120193720027461

User_12 ! Cluster_ABC ! 1424736000000 ! totalApplications	84
User_1266 ! Cluster_XYZ ! 1424736000000 ! org.apache.hadoop.mapreduce.TaskCounter ! MAP_OUTPUT_MATERIALIZED_BYTES	1226999177100261
User_1266 ! Cluster_XYZ ! 1424736000000 ! totalApplications	96