[Design] Application Level Aggregation of Timeline data

Introduction

We need application level aggregation of Timeline data

- To present end user aggregated states for each application, include: resource (CPU, Memory) consumption across all containers, number of containers launched/completed/failed, etc. We need this for apps while they are running as well as when they are done.
- Also, framework specific metrics, e.g. HDFS_BYTES_READ, should be aggregated to show details of states in framework level.
- Other level (Flow/User/Queue) aggregation can be more efficient to be based on Application-level aggregations rather than raw entity-level data as much less raws need to scan (with filter out non-aggregated entities, like: events, configurations, etc.).

Related Tables

The ApplicationState table stores time series data for application metrics as well as final states. It can be split into two tables by aggregated from RMTimelineCollector or AppLevelTimelineCollector.

ApplicationState Table (aggregated from RMTimelineCollector*)*

PKs: User_ID + Cluster_ID + Flow_ID + FlowRun_ID + Application ID	**Number of Attempts**	**Container Aggregate Metrics**	**Resource Metrics**	**Time: start, end, last_modification**
… + application_1431113219200_0001	1	allocated: 0 preempted:0 failed: 0 reuse: 0	mem-allocated: 0 cpu-allocated: 0 resource-preempted: (0, 0)	start: 1431113213456 end: last_modification: 1431113213456

ApplicationState Table (aggregated from AppLevelTimelineCollector*)*

PKs: User_ID + Cluster_ID + Flow_ID + FlowRun_ID + Application ID	**Container Aggregate metrics**	**Resource Metrics**	**Framework Metrics**	**Time: start, end, last_modification**
… + application_1431113219200_0001	allocated: 0 preempted:0 failed: 0 reuse: 0 (Note: belows are currently not available yet) TotalContainerRequest: 0 ContainerRequestByLocalityNode: 0 ContainerRequestByLocalityRack: 0 ContainerRequestByPriority: 0	pMem-consumption: 0 vMem-consumption: 0 CPU-consumption: 0 Resource-consumption: 0	MR: *HDFS_BYTES_READ: 107374182*, *HDFS_BYTES_WRITTEN*: 268435456	start: 1431113219200, end: last_modification: 1431113219200

Control Flow in Details

Aggregation from AppLevelTimelineCollector

AppLevelTimelineCollector will do aggregation since the begin of its lifecycle, the normal flow is:

1. aggregate metrics data received from NM and AM with locally cached aggregated states data.
2. post aggregated data to Application States table (with HBase or Phoenix writer) in a given (configurable) time interval.
3. cache aggregated summary data locally for next time aggregation.

In case AppLevelTimelineCollector get failed over (together with AM get launched somewhere else), the new launched AppLevelTimelineCollector will recover previous aggregated states by query from ApplicationStates tables as initial step before continue the aggregation on new received metrics data.

Aggregation from RMTimelineCollector

For metrics data sourcing from RM (resource allocated, etc.), RMTimelineCollector will aggregate these metrics, cache summary locally and push to storage backend in a configurable interval which is similar to AppLevelTimelineCollector. The failed over case for RM is similiar, new launched RMTimelineCollector will try to recover previous aggregated states from ApplicationStates table and start aggregation base on that.

We need to consider NM/RM Timeline collector failure cases (or rolling upgrades) that the cached aggregated results could be lost. Put them to NM/RM state store is overkill and not suitable for all cases (like: NM failed without restart, AppTimelineCollector will get launched in somewhere else), another way is to make sure AppLevelTimelineCollector check things in entities table when it get restarted.

-