[YARN-3981] Support timeline-clients not associated with an application

Problem Statement

In Timeline Service v2 design so far, all the timeline clients are associated with an ApplicationId. This forces user timeline clients to know ApplicationId mandatorily. Sometimes entities need to be published by the workflows about its status or general configuration used or progress of the flow-run etc. These general entities can not be associated with any of ApplicationId.

Ex : Hive publishes its entity to ATS such as entity type as `HIVE_QUERY_ID` and its corresponding data. But ATSv2 does not allow Hive to publish its data.

Requirement

1. Any workflows should be able to publish entities outside the context of ApplicationId and within the boundary of flow and flow-runs.
2. User should able to get these entities without knowledge of ApplicationId.

Proposal

1. Support for flow-run collectors.
2. Allow ATSv2 TimelineClient to discover flow-run collector address.

Currently, App-Collector service is distributed across the cluster. It is started along with AM containers in NodeManager as auxiliary service. The lifetime of App-Collector is same as ApplicationMaster lifetime.

For timeline client which are not associated with ApplicationId, the below are new proposals

Distributed Flow-Collector Service

Where to run: Logical design

Flow Collectors will run in NodeManager. Flow collectors are configured as system service so that flow collectors are started during YARN cluster startup.

Life-cycle

Container Based Solution using System Services.

- START
    - During YARN cluster startup start, flow collector service is started.
    - One collector service, any timeline client are allowed to publish entities. Many clients publishes into one collector service.
    - Any arbitrary timeline client can discover the collector address and able to publish entities at any point of time.

- STOP
    - Flow collector is long running service, there is no end for this service. Only admin can control over this service.

Handling Denial of Service

// TODO, Is it really required to control? In secured environment, users are authenticated which should be allowed to publish entities. Bad user vs Bad app. This is current problem even in app collectors where bad app end up in writing plenty of entities. This need to brain strorm more on general solution for app collectors and flow collectors.

Physical optimizations

- **Container Based solution :** The containers are managed by system native services. This make sures that availability and reliability of flow-collectors. Discovering flow collectors will be easier and secured.

Discovery

Flow collectors are started via native services as system services. Each system service registers their service details with service-registry. And these are discovered by service-registry-clients.

Using the above feature from native services, Timeline Client is embedded with service-registry-client. During init/start of timeline client, service registry will get flow-collector address from service registry which will be used for publishing entities.

Discovery steps are

1. Timeline Client#start contact service registry for system service flow-collector details.
2. Timeline Client can publish entities using flow-collector address.
3. If flow-collector killed in running node, the collector will be launched in any other available node. In such cases, timeline client rediscover the collector address.

Fault tolerance

Since native services manages life cycle of flow collectors, and collector address are discovered through service registry, service failure cases are handled by native services where in these service are relaunched on high priority. The TimelineClient need to retry on connect exception.

Security

TBD, Token also should be distributed via service-registry?

Storage Implementation

Existing table is sufficient for storing off line client entities with constant application name. There is no change in the table schema.

1. Able to accept entities with */ws/v2/timeline/entities?appid=$appid&async=true&flowid=$flowid&flowrunid=$flowrunid*
2. Every entities are written into flow_table, flow_run_table and entity_table. Note that not for application_table.
3. User will be able to query flows and flow_run details.

Timeline Client

1. User can create timeline client using new constructor.
   TimelineClient(String flowName, long flowRunId)
   Or
   TimelineClient() // default constructor at which default flowName and flowRunId are taken.

Timeline Reader

1. New REST endpoint is required to expose i.e

/ws/v2/timeline/entities/ENTITY_TYPE?userid=$userid&flowid=$flowid&flowrunid=$flowrunid

2. If user do not specify any query parameter then reader search for default flowid and flowrunid.

Challenges

1. Aggregation : Since there is no application attached, app level aggregation is not possible.
2. How do we optimize connection to HBase.

Dependency

1. Flow collectors are planned to launch as System Service. It is dependent on https://issues.apache.org/jira/browse/YARN-1593.