Service binding/recover/discovery for application aggregator of ATS

Key Assumptions and Requirements:

We need a per-application logical aggregator for ATS which provides aggregator service in form of REST API to: AM, NMs and RM, and all counterparts need to discover service provided by aggregator:

- Physically, we have a per NM aggregator which could be an auxiliary service or a separate daemon process or even containers with interface of REST service (refer to diagram in YARN-3033).
- One application only have one aggregator to aggregate all information related to this application (not include info from RM side).
- As providing RESTful service, one physical aggregator can serve multiple logical app aggregators service. However, an explicitly bind effort is needed as YARN-3030 shows. In our first auxiliary service implementation, a new app aggregator service (for application) get registered/bind to a physical aggregator (NM's web server) when AM container get launched on particular NM.
- We always try to bind logical aggregator service to local node of AM in launching AM container for data locality reason. However, AM container could be reschedule to other node for some reason (container failure, etc.), so we cannot guarantee the two are always together.
- NM can be failed, so logical app aggregator service can be re-bind to other physical aggregator when this failure happens, existing service consumers (AM, NMs with app's containers) could have way to detect/discover this change and update.
- NM handle the detailed life cycle management of logical app aggregator, include: register/unregister aggregator for an application. However, RM is the center part to exchange information of aggregator, include: address, healthy status, etc., with related counterparts. If NM get failed, RM will notify other NMs to take care related application aggregators.
- NM restart with work preserving will handle the recovery of registered aggregators info and known aggregators info (getting from RM).
- RM restart with work preserving could recover aggregator list from RMStateStore or asking NM to reconnect with registered aggregators.
- RM could have a separated aggregators which could be a separated topic.
- Security topic will be discussed in a separated document/JIRA.

Several options:

1. By leverage YARN-913, use “Yarn Service Registry”:

   AM register a new service to RM via YARN service registry framework. RM and related NMs can get this information from RM registered services with service_type and application_id.

   Pros: more generic solution

   Cons: may not meet particular requirement, like:

   - Timeline service will serve as build-in service, not necessary for application to register service explicitly
   - NM also need this aggregators info
   - we have preference to bind service to local node of AM container
   - launching of NM aggregators is not in way of YARN service container (YARN-3033)

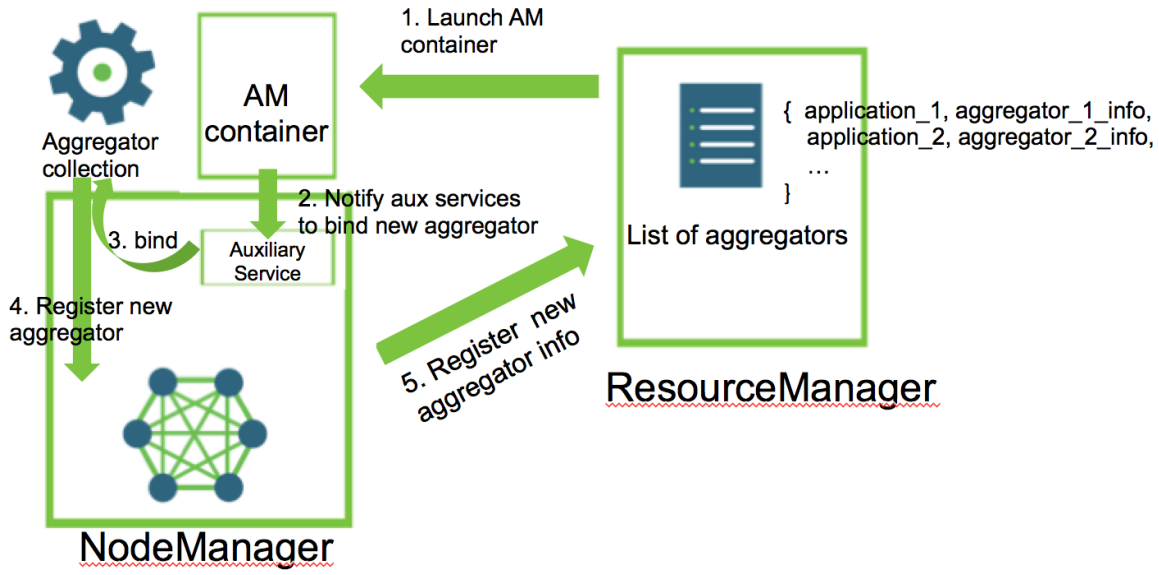
   Also, it depends on another big feature’s progress (YARN-913)

2. Maintain an aggregator list in RM:
   - When RM launch the AM container on a NM, a new logical aggregator service entry will be binded to per node aggregator (by auxiliary service now). If binding work get successfully, NM will notify RM on the new aggregator details (include binding address and port).
   - Getting back this app aggregator info, RM add this info to a list of application aggregators (a hashmap, from applicationID to service address) that can map from applicationID to access address for aggregator service.
   - AM and other NMs will get related app aggregator details in heartbeat with RM.
   - RM keep track of app aggregator status (actively or reactively) and relaunch (rebinding) it to other NMs if necessary in case of aggregator/NM failure.
   - Some retry logic in TimelineClient for tolerating unaccessibility during failure cases. During retry, it will wait and listen any updates on aggregators info within AM or NM.

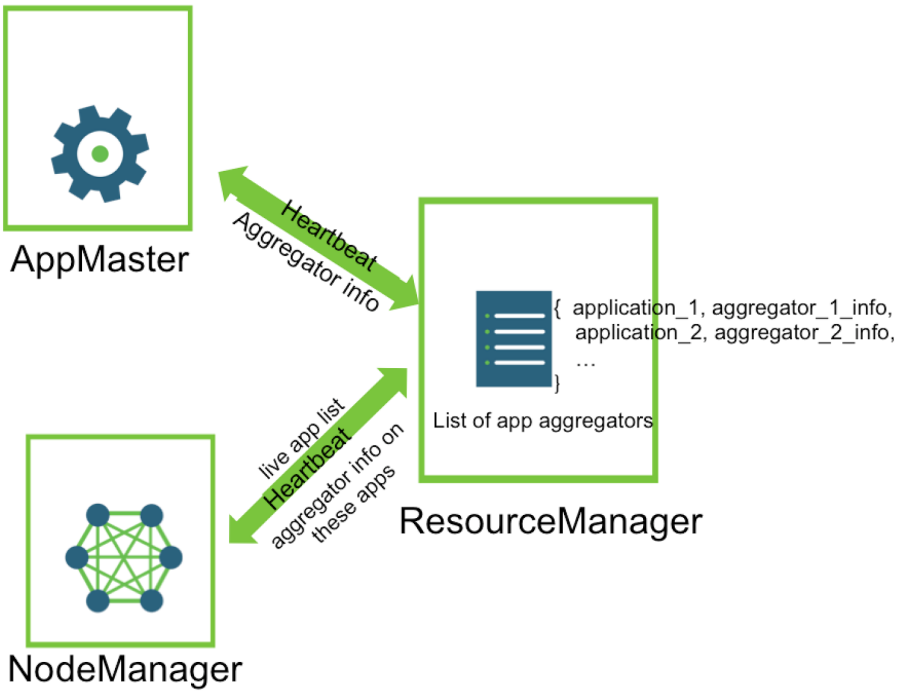
Based on pros & cons above and discussions in JIRA of YARN-3039, we will go with option 2 as an initiative implementation.

Architecture for Option 2

1. aggregator registered into RM



2. aggregators' info broadcast to related AM and NMs from RM



Failure Cases

1. RM failures:

   Aggregator list get stored in RMStateStore, so RM failed over with HA will recover this list automatically.

   Or an alternative way: NM keep the registered aggregator list, and notify this to RM during reconnection.

2. AM container failures:

   AM will request aggregator service info from RM again (in heartbeat) when get relaunched somewhere.

3. Aggregator failed (only):

   RM will track the status (healthy) of logical app aggregator (active or reactive). If an aggregator failure get detected, RM will inform a new healthy NM to bind a new service entry for this application. TimelineClient has some retry logic to handle REST api call failed (most are 404) with waitting for renewing the aggregator service address in AM/NM next rounds of heartbeat with RM.

4. NM failed:

   AM + Aggregator failed at the same time. RM launch new AM and logical app aggregator (in the same location of AM), and do the same thing as case 3.

Open Questions

How to identify the failure of application aggregator?
Option 1: Actively - RM can check each logical aggregator's healthy status by ping in an interval.

Option 2: Reactively (detect-on-fly): if AM or NMs cannot visit aggregator service (like 404 error), it will notify RM in next heartbeat or call some new api in rpc. When receiving a report of aggregator service broken, it will check if new aggregator info get updated recently; if not, it will try to connect and get failed then notify other NM to rebind app aggregator and get back to related AM and NMs in following heartbeats.