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: RM, AM and NMs, 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 with interface of web service. One application only have one aggregator to aggregate all information related to this application.
- As providing RESTful service, one physical aggregator can serve multiple logical app aggregators service.
- A new app aggregator service (for application) get registered/bind to a physical aggregator (NM) 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, RM, NMs with app's containers) could have way to detect/discover the change and update.
- RM handle the life cycle management of logical app aggregator especially check the healthy and relaunch somewhere if necessary.
- 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)

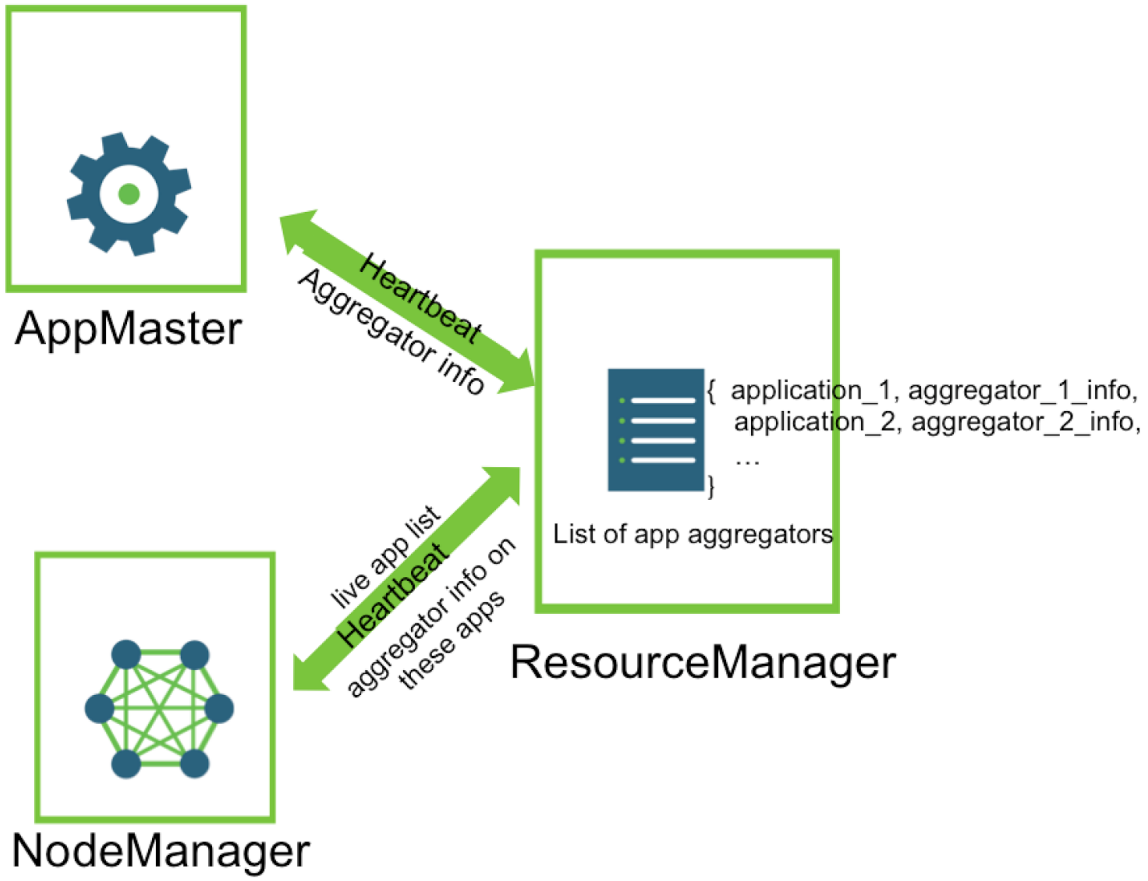
   Also, it depends on another big feature's progress (YARN-913)

2. Maintain an aggregator list in RM:

- When RM launch the AM container, RM can inform NM to start/update the logical aggregator service to provide a new service entry for new application.
- 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 logical app aggregator to physical world.
- RM keep track of app aggregator status and relaunch (rebinding) it to other NMs if necessary in case of aggregator/NM failure.
- AM and NMs can get/update related app aggregator info from RM in heartbeat protocol
- We have some retry logic in AM, NM, RM for consuming of app appgregator service, so it could tolerant some time to allow RM to refresh aggregators get failed.

Base on above discussion, we can start with option 2 in initiative implementation.

Architecture for Option 2



Failure Cases

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

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 (e.g. REST ping with an interval). If an aggregator failure get detected, RM will inform a new healthy NM to bind a new service entry for this application. AM and NMs will leverage some retry logic to handle REST api call failed (most are 404) with renewing the aggregator service address in 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: RM can check each logical aggregator's healthy status by ping in an interval.

Option 2: detect-on-fly: if AM or NMs cannot visit aggregator service (like 404 error), it will notify RM in next heartbeat. When receiving a report of aggregator service broken, it will try to connect: if failed, it notify other NM to rebind app aggregator and get back to related AM and NMs in following heartbeats.