Timeline Service v.2 : milestones

Bold items are design required.

[Beta YARN-7055] == 3.1 : supposed to be Feb 2018, no code freeze date yet

1. **Fault tolerance**
    - storage level (HBase client / buffered mutator)
        - Fundamental challenge of hbase not being up right from the beginning, since buffered mutator requires an hbase connection. Likely needs hbase code changes.
    - collector-level fault tolerance
        - At the minimum, the timeline client should not crash/keep crashing due to issues with the timeline collector. The application should be able to run irrespective of issues with collector.
        - If it crashes, we need a redo log for recovery the collector-state
    - Critical applications which cannot survive even one timeline event loss, we need fault tolerance at application level to store that each TimelineEvent goes to HDFS. This is off by default and we provide tunable knobs for frequency of writes from client side.
2. **Security / Authorization** - support timeline domains like in v.1
    - Sub app entities not working in non secure cluster
3. **Migration Path Plan** -Provide steps or plan for migrating from ATSv1/1.5 to ATSv2. Consider supporting both ATSv1 and ATSv2 for helping upgrades and checks.
4. **Support multiple HBase versions** - HBase 1.2.x as well as 2.x / conditional compilation profiles
    - Consider using shaded jars

[Alpha 2 - YARN-5355] - GA **DONE (hadoop 3.0 and hadoop 2.9)**

5. Li/Varun - UI integration (pending YARN-3368) - **DONE**
6. More types of applications: learning
    - MapReduce flows - **DONE**
    - Li/Rohith/Varun - Tez migration - **In-progress**
        - Sub application/long running - **DONE**
        - Decision making to store it in sub application
    - Vrushali - Hive, cascading: plumbing/connection to tez/MR
7. **Varun - (Security) authentication (kerberos)** - **DONE**
8. Li - Timeline Service working through RM failover (HA) (basic acceptability) - **DONE**

[Post-beta]

9. Deployment ideas for a single-node HBase setup
10. Chargeback
11. **Off-application clients/collectors: different data schemas, security (use cases): see YARN-3981**
    - **Hive use case**
12. Flow activity handling for long running applications (not necessarily services)

13. Separate daemon / **Containers for collectors** - stability, ability to roll new code with minimal disruption
    - Impact on timeline client (limits on latency and memory usage)
14. Test driver setup/improvement
15. **Migration**
    - **Can we import v.1.x data?**
    - **How can frameworks cut over from v1/v1.5 to v.2?**
16. Command line support
17. JMX: reader, collectors (logs and metrics), RM/NM JMX
18. **Offline aggregation**
19. Timeline service correctness and completeness through RM failover (HA)
20. **Working with YARN federation**
21. **Better support for services** as opposed to short-running applications
    - Timeline information from first class services
    - NodeManagers sending its own stats directly
    - RM writing cluster level metrics YARN-3881
22. **Multi-DC setup**
23. Not ready for alternate storage yet, but will need to do work to make that happen
24. HDFS stats
25. Pooling readers
26. YARN first-class flow API with lifecycles
27. Provide streaming reader API to fetch entities YARN-5627
28. Retire/combine Application History Server and MR Job History Server

---------- Extra notes ----------

- Tez application can run multiple queries from different end-users in single application master.
- Tez UI is hosted in different machines. UI component need not know about application id rather they know about their entity type. Now, all the REST queries for entity retrieval is made via application-id only. As of now , there is no support for querying entities without app-id. Support should come for this also.
-