Timeline Service v.2: next milestones

Bold items are design required.

[Alpha 2]

1. UI integration
2. More types of applications
    - MapReduce flows
    - Tez migration
    - Hive

[Beta]

3. JMX: reader, collectors (logs and metrics), RM/NM JMX
4. Flow activity handling for long running applications (not necessarily services)
5. **Migration**
    - **Can we import v.1.x data?**
    - **How can frameworks cut over to v.2?**
6. Separate process / **Containers for collectors** - stability, ability to roll new code with minimal disruption
    - Impact on timeline client (limits on latency and memory usage)
7. Timeline Service working through RM failover (HA) (basic acceptability)
8. Command line support
9. **Fault tolerance - storage level**
10. (Security) authentication (kerberos)
11. **Off-application clients/collectors: different data schemas, security (use cases)**
12. Deployment ideas for a single-node HBase setup
13. Test driver setup/improvements

[Post-beta]

14. (Security) authorization - timeline domains like in v.1
15. **Offline aggregation**
16. Timeline service correctness and completeness through RM failover (HA)
17. Fault tolerance - collector-level fault tolerance
    - At 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
18. **Working with YARN federation**
19. **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
20. **Multi-DC setup**

21. Not ready for alternate storage yet, but will need to do work to make that happen
22. HDFS stats
23. Pooling readers