Item 159

git_comments:

1. set job classloader if configured
2. We can't use getJobID in useLocal case because JobClient and Job point to different instances of LocalJobRunner

git_commits:

1. **summary:** MAPREDUCE-5875. Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild. (Gera Shegalov via kasha)
   message: MAPREDUCE-5875. Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild. (Gera Shegalov via kasha) (cherry picked from commit ed0e0ef9748ce5b231de677ab41c3035137bbde4)

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
2. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
3. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
4. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
5. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
6. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
7. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
8. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
9. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
10. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
11. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
12. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
13. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.

14. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
15. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
16. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
    label: code-design
17. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
18. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
19. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
20. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
21. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
22. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
23. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
    label: code-design
24. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
25. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
26. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
27. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
28. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
29. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
    label: code-design
30. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.

31. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
32. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
33. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
34. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
35. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
36. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
37. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
38. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
39. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
40. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
41. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
42. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
43. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
44. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
    label: code-design
45. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
46. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
47. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
48. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild

   description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.

49. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
50. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
    label: code-design
51. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
52. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
53. **summary:** Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild
    description: Currently, counter limits "mapreduce.job.counters.*" handled by {{org.apache.hadoop.mapreduce.counters.Limits}} are initialized asymmetrically: on the client side, and on the AM, job.xml is ignored whereas it's taken into account in YarnChild. It would be good to make the Limits job-configurable, such that max counters/groups is only increased when needed. With the current Limits implementation relying on static constants, it's going to be challenging for tools that submit jobs concurrently without resorting to class loading isolation. The patch that I am uploading is not perfect but demonstrates the issue.
    label: code-design

jira_issues_comments:

1. Is this a dupe of MAPREDUCE-5856?
2. [~sandyr], there is definitely a substantial overlap. Sorry that I missed it which would save some time. However MAPREDUCE-5856 does not seem to address uberized jobs /AM and jobs submitted without JobClient. See the unit tests. I'll link to the JIRA's for further discussion.
3. So, it sounds like both patches are needed, right? MAPREDUCE-5875 fixes this for the AM and for submitting jobs without JobClient, while MAPREDUCE-5856 fixes this for submitting and retrieving jobs via JobClient.
4. [~rkanter], MAPREDUCE-5875.v01.patch takes care of the JobClient as well because JobClient ends up calling job.submit -> JobSubmitter.submitJobInternal
5. It looks like calling {{Limits.init(conf)}} in JobSubmitter.submitJobInternal will fix the case where JobClient submits a job, but not the case where JobClient retrieves a job (i.e. {{jobClient.getJob(...)}}); that goes through a different path.
6. Thanks for pointing at {{JobClient.getJob}}, Robert. It's a wrapper around {{Cluster.getJob}} and its implementation is incorrect. It uses {{JobConf.<init>(String)}} that adds the submitted conf (job.xml) as a resource on LocalFileSystem {code} conf = new JobConf(status.getJobFile()); {code} instead of using the correct filesystem. I fixed it and added a unit test for this scenario.
7. I'm not sure if the new patch addresses the issue described in MAPREDUCE-5856. If you don't have mapred-site.xml on the classpath, {{Limits}} will use the default counters limits unless you call {{Limits.init(conf)}}, and I dont see that happening when looking through the code path at {{jobClient.getJob(...)}}; unless I'm just not seeing it?
8. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12643518/MAPREDUCE-5875.v02.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 1.3.9) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:red}-1 core tests{color}. The patch failed these unit tests in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.TestNetworkedJob org.apache.hadoop.mapred.TestJobCounters org.apache.hadoop.mapred.TestClientRedirect The following test timeouts occurred in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapreduce.v2.TestMRJobs org.apache.hadoop.mapreduce.v2.TestUberAM org.apache.hadoop.mapred.pipes.TestPipeApplication {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4585//testReport/ Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4585//console This message is automatically generated.
9. v02 in this JIRA does not assume mapred-site.xml on the class path. It covers this general scenario (modulo failing tests :)). # client 1 submits a job with conf 1. Its customizations + cluster customizations result in job.xml on the cluster. # client 2 connects to the cluster and calls getJob. Now even if it had slight different conf2, the job object returned will be the one from step 1. I don't think that MAPREDUCE-5856 correctly deals with this
10. I have rerun tests manually. $ mvn clean test -Dtest=org.apache.hadoop.mapred.TestNetworkedJob,org.apache.hadoop.mapred.TestJobCounters,org.apache.hadoop.mapred.TestClientRedirect,org.apache.hadoop.
    And none but org.apache.hadoop.mapred.pipes.TestPipeApplication has any issues. TestPipeApplication could be a red herring related to MAPREDUCE-5868. I'll re-upload the patch to have another run on Jenkins.
11. Please disregard the last comment. I ran the tests in the wrong branch.
12. v03 that - ignores FileNotFound for job conf in tests - adds job conf from history for prior to history file parsing.
13. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12643942/MAPREDUCE-5875.v03.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 1.3.9) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:red}-1 core tests{color}. The following test timeouts occurred in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.pipes.TestPipeApplication {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4592//testReport/ Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4592//console This message is automatically generated.
14. I just tried out your v3 patch for the use case in MAPREDUCE-5856 and it does fix the problem. Yours covers more cases, so let's use it and I'll close MAPREDUCE-5856 as a duplicate. thanks.
15. [~rkanter], thank you for validating the v03 patch.
16. **body:** Patch mostly looks good to me. Couple of nits: # We don't have to check if WARN is enabled for LOG level. I believe we assume INFO. # Looks like we fetch the conf, the path, the filesystem and call conf.addResource at multiple places in the patch. We should probably add a util method in MR or a helper method in Configuration itself (if it makes sense) to handle this instead of duplicating the logic.
    label: code-design

17. Hi [~kasha], thanks for the review. I addressed 2. As for 1, I prefer to consistently check whether log level is enabled regardless of the actual level.
18. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12647856/MAPREDUCE-5875.v04.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 1.3.9) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:red}-1 core tests{color}. The following test timeouts occurred in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.pipes.TestPipeApplication {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4638//testReport/ Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4638//console This message is automatically generated.
19. We were trying out v3 of the patch some more and ran into a problem if you configure your history files (i.e. the "done" directory) to go under /user instead of /tmp: {noformat} wypoon@p0110 ~]$ mapred job -status job_1401722172470_0018 14/06/02 10:43:40 INFO mapred.ClientServiceDelegate: Application state is completed. FinalApplicationStatus=SUCCEEDED. Redirecting to job history server Exception in thread "main" org.apache.hadoop.security.AccessControlException: Permission denied: user=wypoon, access=EXECUTE, inode="/user/history2/done":mapred:hadoop:drwxrwx--- at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkFsPermission(FSPermissionChecker.java:271) at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:257) at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkTraverse(FSPermissionChecker.java:208) at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:171) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:5499) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:5481) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPathAccess(FSNamesystem.java:5443) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocationsUpdateTimes(FSNamesystem.java:1714) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocationsInt(FSNamesystem.java:1666) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations(FSNamesystem.java:1646) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations(FSNamesystem.java:1620) at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getBlockLocations(NameNodeRpcServer.java:482) at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:322 at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodeProtocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java) at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:585) at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1026) at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:1986) at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:1982) at java.security.AccessController.doPrivileged(Native Method) at javax.security.auth.Subject.doAs(Subject.java:415) at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1555) at org.apache.hadoop.ipc.Server$Handler.run(Server.java:1980) at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method) at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:57) at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45) at java.lang.reflect.Constructor.newInstance(Constructor.java:526) at org.apache.hadoop.ipc.RemoteException.instantiateException(RemoteException.java:106) at org.apache.hadoop.ipc.RemoteException.unwrapRemoteException(RemoteException.java:73) at org.apache.hadoop.hdfs.DFSClient.callGetBlockLocations(DFSClient.java:1139) at org.apache.hadoop.hdfs.DFSClient.getLocatedBlocks(DFSClient.java:1127) at org.apache.hadoop.hdfs.DFSClient.getLocatedBlocks(DFSClient.java:1117) at org.apache.hadoop.hdfs.DFSInputStream.fetchLocatedBlocksAndGetLastBlockLength(DFSInputStream.java:264) at org.apache.hadoop.hdfs.DFSInputStream.openInfo(DFSInputStream.java:231) at org.apache.hadoop.hdfs.DFSInputStream.<init>(DFSInputStream.java:224) at org.apache.hadoop.hdfs.DFSClient.open(DFSClient.java:1290) at org.apache.hadoop.hdfs.DistributedFileSystem$3.doCall(DistributedFileSystem.java:300) at org.apache.hadoop.hdfs.DistributedFileSystem$3.doCall(DistributedFileSystem.java:296) at org.apache.hadoop.fs.FileSystemLinkResolver.resolve(FileSystemLinkResolver.java:81) at org.apache.hadoop.hdfs.DistributedFileSystem.open(DistributedFileSystem.java:296) at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:764) at org.apache.hadoop.mapreduce.Cluster.getJob(Cluster.java:190) at org.apache.hadoop.mapreduce.tools.CLI.run(CLI.java:264) at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70) at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84) at org.apache.hadoop.mapred.JobClient.main(JobClient.java:1237) Caused by: org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.security.AccessControlException): Permission denied: user=wypoon, access=EXECUTE, inode="/user/history2/done":mapred:hadoop:drwxrwx--- at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkFsPermission(FSPermissionChecker.java:271) at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:257) at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkTraverse(FSPermissionChecker.java:208) at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:171) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:5499) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPermission(FSNamesystem.java:5481) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.checkPathAccess(FSNamesystem.java:5443) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocationsUpdateTimes(FSNamesystem.java:1714) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocationsInt(FSNamesystem.java:1666) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations(FSNamesystem.java:1646) at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations(FSNamesystem.java:1620) at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getBlockLocations(NameNodeRpcServer.java:482) at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTranslatorPB.getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:322 at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$ClientNamenodeProtocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java) at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.call(ProtobufRpcEngine.java:585) at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1026) at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:1986) at org.apache.hadoop.ipc.Server$Handler$1.run(Server.java:1982) at java.security.AccessController.doPrivileged(Native Method) at javax.security.auth.Subject.doAs(Subject.java:415) at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1555) at org.apache.hadoop.ipc.Server$Handler.run(Server.java:1980) at org.apache.hadoop.ipc.Client.call(Client.java:1409) at org.apache.hadoop.ipc.Client.call(Client.java:1362) at org.apache.hadoop.ipc.ProtobufRpcEngine$Invoker.invoke(ProtobufRpcEngine.java:206) at com.sun.proxy.$Proxy11.getBlockLocations(Unknown Source) at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolTranslatorPB.getBlockLocations(ClientNamenodeProtocolTranslatorPB.java:219) at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57) at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) at java.lang.reflect.Method.invoke(Method.java:606) at org.apache.hadoop.io.retry.RetryInvocationHandler.invokeMethod(RetryInvocationHandler.java:186) at org.apache.hadoop.io.retry.RetryInvocationHandler.invoke(RetryInvocationHandler.java:102) at com.sun.proxy.$Proxy12.getBlockLocations(Unknown Source) at org.apache.hadoop.hdfs.DFSClient.callGetBlockLocations(DFSClient.java:1137) ... 16 more {noformat} This happens when calling {{at org.apache.hadoop.mapreduce.Cluster.getJob(Cluster.java:190)}} which is where v3 was doing {{conf.addResource(fs.open(jobPath), jobPath.toString());}} I haven't had a chance to try this with v4, but from looking at the patch, I believe it will run into the same problem. I think you may need to add a {{doAs}} somewhere?
20. I will look at it later this week, Robert. In your scenario, the user issuing getJob is not identical with the owner of the job, correct?
21. The user calling getJob is the same as the user who submitted the job. Looking at the permissions in the error, {{mapred:hadoop:drwxrwx---}} it's only going to allow either user "mapred" or anyone from the "hadoop" group access. Also, this used to work before, presumably because Cluster wasn't trying to access the FileSystem there before.

22. Hi Robert, bq. We were trying out v3 of the patch some more and ran into a problem if you configure your history files (i.e. the "done" directory) to go under /user instead of /tmp This should be unrelated to the root location /tmp vs /user. It's related to the permissions/umask used to create the history done dir: {code} public static final FsPermission HISTORY_DONE_DIR_PERMISSION = FsPermission.createImmutable((short) 0770); /** * Umask for the done dir and derivatives. */ public static final FsPermission HISTORY_DONE_DIR_UMASK = FsPermission .createImmutable((short) (0770 ^ 0777)); {code} Presumably the goal is to prevent users from listing other users job files. Looks like we can't support job-specified limits when dealing with history if the user does not share the group with JHS user.
23. **body:** v05 to ignore AccessControlException until a better solution is found.
    label: code-design
24. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12650254/MAPREDUCE-5875.v05.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 1.3.9) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:red}-1 core tests{color}. The following test timeouts occurred in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.pipes.TestPipeApplication {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4658//testReport/ Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4658//console This message is automatically generated.
25. v06 works around the permission problem by using REST API to fetch the conf from JHS. On the way it also fixes the bug in JHS that drops the scheme from the tracking URL in CompletedJob.getReport
26. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12650427/MAPREDUCE-5875.v06.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 1.3.9) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:red}-1 core tests{color}. The patch failed these unit tests in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.TestJobCounters org.apache.hadoop.mapreduce.TestMRJobClient The following test timeouts occurred in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.pipes.TestPipeApplication {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4660//testReport/ Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4660//console This message is automatically generated.
27. Jobs run MiniMRCluster get an incorrect history tracking URL and there can be a delay when history tracking URL is set via unregister. The latter can be solved by waiting but I think it's sufficient to simply log the error. It may be better to do it only in the test case and throw an exception in production code. Suggestions?
28. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12650463/MAPREDUCE-5875.v07.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 1.3.9) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:red}-1 core tests{color}. The following test timeouts occurred in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.pipes.TestPipeApplication {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4661//testReport/ Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4661//console This message is automatically generated.
29. **body:** Looks close. Couple of comments. # Instead of doing string manipulation inline, can we add a helper method preferably where we are constructing these History URLs. {code} final Path jobConfPath = new Path(jobFile.getParent(), jobDetails[0] + "_" + jobDetails[1] + "_" + jobDetails[2] + "_conf.xml"); Limits.reset(new JobConf(conf).loadFile(fs, jobConfPath)); {code} # Nit: I am okay with not addressing this. We should probably skip isEnabled checks for INFO and WARN levels as the code doesn't have it anywhere else.
    label: code-design
30. v08 to address [~kasha]'s latest review.
31. {color:green}+1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12655029/MAPREDUCE-5875.v08.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 2.0.3) warnings. {color:green}+1 release audit{color}. The applied patch does not increase the total number of release audit warnings. {color:green}+1 core tests{color}. The patch passed unit tests in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-common hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient. {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4725//testReport/ Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4725//console This message is automatically generated.
32. [~kasha], did you have more comments for this patch?
33. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12655029/MAPREDUCE-5875.v08.patch against trunk revision 596702a. {color:red}-1 patch{color}. The patch command could not apply the patch. Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4949//console This message is automatically generated.
34. v09. Rebased the patch to the latest trunk
35. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12674373/MAPREDUCE-5875.v09.patch against trunk revision 554250c. {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 1 new or modified test files. {color:green}+1 javac{color}. The applied patch does not increase the total number of javac compiler warnings. {color:green}+1 javadoc{color}. There were no new javadoc warning messages. {color:green}+1 eclipse:eclipse{color}. The patch built with eclipse:eclipse. {color:green}+1 findbugs{color}. The patch does not introduce any new Findbugs (version 2.0.3) warnings. {color:red}-1 release audit{color}. The applied patch generated 1 release audit warnings. {color:red}-1 core tests{color}. The patch failed these unit tests in hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient: org.apache.hadoop.mapred.pipes.TestPipeApplication {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4954//testReport/ Release audit warnings: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4954//artifact/patchprocess/patchReleaseAuditProblems.txt Console output: https://builds.apache.org/job/PreCommit-MAPREDUCE-Build/4954//console This message is automatically generated.
36. +1, pending Jenkins
37. TestPipeApplication failure is MAPREDUCE-6115. File .keep causing the RAT plugin check failure is from YARN-2652 .
38. Committing this.
39. FAILURE: Integrated in Hadoop-trunk-Commit #6245 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/6245/]) MAPREDUCE-5875. Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild. (Gera Shegalov via kasha) (kasha: rev e8a31f2e1c34514fba2f480e8db652f6e2ed65d8) * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/Cluster.java * hadoop-

mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app/src/main/java/org/apache/hadoop/mapreduce/v2/app/MRAppMaster.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/src/main/java/org/apache/hadoop/mapreduce/v2/hs/CompletedJob.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/counters/Limits.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/JobSubmitter.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient/src/test/java/org/apache/hadoop/mapreduce/v2/TestMRJobs.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/jobhistory/HistoryViewer.java * hadoop-mapreduce-project/CHANGES.txt

40. Thanks for the contribution, Gera. Just committed this to trunk and branch-2.
41. SUCCESS: Integrated in Hadoop-Yarn-trunk #709 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk/709/]) MAPREDUCE-5875. Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild. (Gera Shegalov via kasha) (kasha: rev e8a31f2e1c34514fba2f480e8db652f6e2ed65d8) * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient/src/test/java/org/apache/hadoop/mapreduce/v2/TestMRJobs.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/counters/Limits.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/JobSubmitter.java * hadoop-mapreduce-project/CHANGES.txt * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/Cluster.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/src/main/java/org/apache/hadoop/mapreduce/v2/hs/CompletedJob.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/jobhistory/HistoryViewer.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app/src/main/java/org/apache/hadoop/mapreduce/v2/app/MRAppMaster.java
42. FAILURE: Integrated in Hadoop-Hdfs-trunk #1899 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk/1899/]) MAPREDUCE-5875. Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild. (Gera Shegalov via kasha) (kasha: rev e8a31f2e1c34514fba2f480e8db652f6e2ed65d8) * hadoop-mapreduce-project/CHANGES.txt * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/jobhistory/HistoryViewer.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app/src/main/java/org/apache/hadoop/mapreduce/v2/app/MRAppMaster.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/counters/Limits.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/JobSubmitter.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/Cluster.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient/src/test/java/org/apache/hadoop/mapreduce/v2/TestMRJobs.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/src/main/java/org/apache/hadoop/mapreduce/v2/hs/CompletedJob.java
43. FAILURE: Integrated in Hadoop-Mapreduce-trunk #1924 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk/1924/]) MAPREDUCE-5875. Make Counter limits consistent across JobClient, MRAppMaster, and YarnChild. (Gera Shegalov via kasha) (kasha: rev e8a31f2e1c34514fba2f480e8db652f6e2ed65d8) * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/Cluster.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient/src/test/java/org/apache/hadoop/mapreduce/v2/TestMRJobs.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/jobhistory/HistoryViewer.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/src/main/java/org/apache/hadoop/mapreduce/v2/hs/CompletedJob.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/counters/Limits.java * hadoop-mapreduce-project/CHANGES.txt * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app/src/main/java/org/apache/hadoop/mapreduce/v2/app/MRAppMaster.java * hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/JobSubmitter.java
44. It looks like this patch breaks {{mapred job -status <job-id>}}. See MAPREDUCE-6288.
45. [~rkanter], Oh my, we messed something up with this JIRA. Both MAPREDUCE-6288 and MAPREDUCE-6286 are broken because we dropped the logic in v08 during rebase to v09.
46. Removed this from 2.7.0. See the discussion on [MAPREDUCE-6288|https://issues.apache.org/jira/browse/MAPREDUCE-6288?focusedCommentId=14387563&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-14387563].
47. FAILURE: Integrated in Hadoop-trunk-Commit #7478 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/7478/]) Reverted MAPREDUCE-6286, MAPREDUCE-6199, and MAPREDUCE-5875 from branch-2.7. Editing CHANGES.txt to reflect this. (vinodkv: rev e428fea73029ea0c3494c71a50c5f6c994888fd2) * hadoop-mapreduce-project/CHANGES.txt
48. FAILURE: Integrated in Hadoop-Yarn-trunk-Java8 #150 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk-Java8/150/]) Reverted MAPREDUCE-6286, MAPREDUCE-6199, and MAPREDUCE-5875 from branch-2.7. Editing CHANGES.txt to reflect this. (vinodkv: rev e428fea73029ea0c3494c71a50c5f6c994888fd2) * hadoop-mapreduce-project/CHANGES.txt
49. FAILURE: Integrated in Hadoop-Yarn-trunk #884 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk/884/]) Reverted MAPREDUCE-6286, MAPREDUCE-6199, and MAPREDUCE-5875 from branch-2.7. Editing CHANGES.txt to reflect this. (vinodkv: rev e428fea73029ea0c3494c71a50c5f6c994888fd2) * hadoop-mapreduce-project/CHANGES.txt
50. FAILURE: Integrated in Hadoop-Hdfs-trunk #2082 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk/2082/]) Reverted MAPREDUCE-6286, MAPREDUCE-6199, and MAPREDUCE-5875 from branch-2.7. Editing CHANGES.txt to reflect this. (vinodkv: rev e428fea73029ea0c3494c71a50c5f6c994888fd2) * hadoop-mapreduce-project/CHANGES.txt
51. FAILURE: Integrated in Hadoop-Mapreduce-trunk-Java8 #150 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk-Java8/150/]) Reverted MAPREDUCE-6286, MAPREDUCE-6199, and MAPREDUCE-5875 from branch-2.7. Editing CHANGES.txt to reflect this. (vinodkv: rev e428fea73029ea0c3494c71a50c5f6c994888fd2) * hadoop-mapreduce-project/CHANGES.txt
52. FAILURE: Integrated in Hadoop-Hdfs-trunk-Java8 #141 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk-Java8/141/]) Reverted MAPREDUCE-6286, MAPREDUCE-6199, and MAPREDUCE-5875 from branch-2.7. Editing CHANGES.txt to reflect this. (vinodkv: rev e428fea73029ea0c3494c71a50c5f6c994888fd2) * hadoop-mapreduce-project/CHANGES.txt
53. FAILURE: Integrated in Hadoop-Mapreduce-trunk #2100 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk/2100/]) Reverted MAPREDUCE-6286, MAPREDUCE-6199, and MAPREDUCE-5875 from branch-2.7. Editing CHANGES.txt to reflect this. (vinodkv: rev e428fea73029ea0c3494c71a50c5f6c994888fd2) * hadoop-mapreduce-project/CHANGES.txt
54. I'm unable to reset the state to reopened here due to the revert so am just adding a release note instead to highlight it.
55. **body:** Uh sorry, the commit message timeline is confusing on this. This is indeed present in the trunk and branch-2.8/branch-2, was only reverted in branch-2.7. My bad - sorry for the noise - the addendum vs. 2.7 notes confused me :(Resetting to earlier state.
    label: code-design
56. Revert from branch-2.8 given discussion in MAPREDUCE-6288.
57. As discussed in MAPREDUCE-6288, I have revert this patch from trunk and branch-2. We need to reopen this jira for an improved solution. But it looks like I cannot reopen it - can someone able to do this?
58. SUCCESS: Integrated in Jenkins build Hadoop-trunk-Commit #12080 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/12080/]) Revert "MAPREDUCE-5875. Make Counter limits consistent across JobClient, (junping_du: rev fbb7d6bcbb887ce52ab1e9d5a1fed67a7f8a4be8) * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-app/src/main/java/org/apache/hadoop/mapreduce/v2/app/MRAppMaster.java * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/JobSubmitter.java * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/src/main/java/org/apache/hadoop/mapreduce/v2/hs/CompletedJob.java * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/Cluster.java * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/counters/Limits.java * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-jobclient/src/test/java/org/apache/hadoop/mapreduce/v2/TestMRJobs.java * (edit) hadoop-mapreduce-project/hadoop-mapreduce-client/hadoop-mapreduce-client-core/src/main/java/org/apache/hadoop/mapreduce/jobhistory/HistoryViewer.java
59. [~junping_du], it's "Closed" so I don't think we can reopen it. We should probably just open up a new JIRA to address this issue.
60. It's closed because it was already in a release. You absolutely need to open a new JIRA for it.
61. Drop 2.9.0 as the patch get reverted from branch-2. Just clone MAPREDUCE-6925 for improved fix. To be clear, this fix never land in any non-alpha release, so we shouldn't claim the fix is released or any regression due to revert of this fix.
62. **body:** bq. To be clear, this fix never land in any non-alpha release, so we shouldn't claim the fix is released or any regression due to revert of this fix. Doesn't matter. The release notes for alpha1 said that this patch is there. At a minimum, -beta1 will need to include a note that says it was removed or, preferably, removed but replaced with something better.
    label: code-design

63. bq. At a minimum, -beta1 will need to include a note that says it was removed or, preferably, removed but replaced with something better. I believe MAPREDUCE-6924 already address this.