Item 230
git_comments:

1. / A B abs used pending reserved req granularity
2. Passing last param as TRUE to use DominantResourceCalculator
3. Resources can be set like memory:vcores
4. / A B maxcap apps subqueues
5. 5 containers will be preempted here
6. root a b
7. guaranteed,max,used,pending
8. queueName\t(priority,resource,host,expression,#repeat,reserved) app1 in a 80 * x in n1 app2 in a 20 default in n2 app3 in b 20 * x in n1 app4 in b 80 default in n2
9. default partition partition=x n1 has partition=x n2 is default partition
10. 30 preempted from app1, 30 preempted from app4, and nothing preempted from app2/app3
11. * * Queue structure is: * * <pre> * root * / \ * a b * </pre> * * Both a/b can access x, and guaranteed capacity of them is 50:50. Two * nodes, n1 has 100 x, n2 has 100 NO_LABEL 4 applications in the cluster, * app1/app2 in a, and app3/app4 in b. app1 uses 80 x, app2 uses 20 * NO_LABEL, app3 uses 20 x, app4 uses 80 NO_LABEL. Both a/b have 50 pending * resource for x and NO_LABEL * * After preemption, it should preempt 30 from app1, and 30 from app4.

git_commits:

1. **summary:** YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda)
   message: YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (cherry picked from commit 1df39c1efc9ed26d3f1a5887c31c38c873e0b784)

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Too much of preemption activity causing continuos killing of containers across queues
   description: Two queues are used. Each queue has given a capacity of 0.5. Dominant Resource policy is used. 1. An app is submitted in QueueA which is consuming full cluster capacity 2. After submitting an app in QueueB, there are some demand and invoking preemption in QueueA 3. Instead of killing the excess of 0.5 guaranteed capacity, we observed that all containers other than AM is getting killed in QueueA 4. Now the app in QueueB is trying to take over cluster with the current free space. But there are some updated demand from the app in QueueA which lost its containers earlier, and preemption is kicked in QueueB now. Scenario in step 3 and 4 continuously happening in loop. Thus none of the apps are completing.

jira_issues_comments:

1. Looping [~rohithsharma] and [~leftnoteasy] Since we use Dominant resource calculator, below piece of code in ProportionalPreemptionPolicy looks doubtful {code} // When we have no more resource need to obtain, remove from map. if (Resources.lessThanOrEqual(rc, clusterResource, toObtainByPartition, Resources.none())) { resourceToObtainByPartitions.remove(nodePartition); } {code} Assume toObtainByPartition is <12, 1> ()memory, core). After another round of preemption, this will become <10, 0>. If the above check hits with this value, its supposed to return TRUE. But the method returns FALSE. Reason is that due to dominance, if any resource item is non-zero then that is returned as true. {code} // Just use 'dominant' resource return (dominant) ? Math.max((float)resource.getMemory() / clusterResource.getMemory(), (float)resource.getVirtualCores() / clusterResource.getVirtualCores()) : Math.min((float)resource.getMemory() / clusterResource.getMemory(), (float)resource.getVirtualCores() / clusterResource.getVirtualCores()); {code} If resource.getVirtualCores() is ZERO and resource.getMemory() is Non-Zero, then this check will return +ve. We feel that this has to be checked prior and if one item is ZERO, we have to say lhs is lesser to rhs.
2. [~sunilg], Trying to understand this issue, when the toObtainResource becomes <10,0>, and assume container size are c1=<2,1>, c2=<5,3>, c3=<4,2>, c4=<2,1>. Preemption policy will kill c1..c3, my understanding of this problem is preemption policy can preempt one of the resource type (CPU/Memory) more than needed, but I'm not sure why it preempts all containers except AM.
3. Is this specific to capacity-scheduler, or does it apply to fair-scheduler as well?
4. Thank you [~leftnoteasy] and [~kasha@cloudera.com] [~kasha] , we have tested this only in CS. And the issue is looking like in DominentResourceCalculator. I will analyze whether this will happen in Fair. [~leftnoteasy], I have understood your point. I can explain you the scenario based on few key code snippets. Please feel free to point out if any issues in my analysis. CSQueueUtils#updateUsedCapacity has below code to calculate absoluteUsedCapacity. {code} absoluteUsedCapacity = Resources.divide(rc, totalPartitionResource, usedResource, totalPartitionResource); {code} This will result a call to DominentResourceCalculator, {code} public float divide(Resource clusterResource, Resource numerator, Resource denominator) { return getResourceAsValue(clusterResource, numerator, true) / getResourceAsValue(clusterResource, denominator, true); {code} In our cluster, the resource allocation is as follows usedResource <10Gb, 95Cores> totalPartitionResource <100Gb, 100Cores>. Since we use dominence, absoluteUsedCapacity will come close to 1 eventhough Memory is used only 10%. IN ProportionalPreemptionPolicy, we use like below {code} float absUsed = qc.getAbsoluteUsedCapacity(partitionToLookAt); Resource current = Resources.multiply(partitionResource, absUsed); {code} So *current - guaranteed* will give us tobePreempted which will be close to <50GB, 45Cores>. Actually here memory should have been 5Gb. Now in our cluster, each container is of <1Gb, 10Cores>. Hence the *cores* will be 0 after 5 container kills, but tobePreempted will still have memory. And as mentioned in above comment, preemption will continue to kill other containers too.
5. The below is the log trace for the issue. In our cluster, there are 3 NodeManager and each with resource {{<memory:327680, vCores:35>}}. Total cluster resource is {{clusterResource: <memory:983040, vCores:105>}} with CapacityScheduler configured queue's with name *default* and *QueueA*. # Application app-1 is submitted to queue default and containers are started running the applications with 10 containers,each with {{resource: <memory:1024, vCores:10>}}. so total used is {{usedResources=<memory:10240, vCores:91>}} {noformat} default user=spark used=<memory:10240, vCores:91> numContainers=10 headroom = <memory:1024, vCores:10> user-resources=<memory:10240, vCores:91> Re-sorting assigned queue: root.default stats: default: capacity=0.5, absoluteCapacity=0.5, usedResources=<memory:10240, vCores:91>, usedCapacity=1.7333333, absoluteUsedCapacity=0.8666667, numApps=1, numContainers=10 {noformat} *NOTE : Resource allocation is by CPU DOMINANT* After 10 container running, available NodeManagers memory is {noformat} linux-174, available: <memory:323584, vCores:4> linux-175, available: <memory:324608, vCores:5> linux-223, available: <memory:324608, vCores:5> {noformat} # Application app-2 is submitted to QueueA. ApplicationMaster container started running and NodeManager memory is {{available: <memory:322560, vCores:3>}} {noformat} Assigned container container_1435072598099_0002_01_000001 of capacity <memory:1024, vCores:1> on host linux-174:26009, which has 5 containers, <memory:5120, vCores:32> used and <memory:322560, vCores:3> available after allocation | SchedulerNode.java:154 linux-174, available: <memory:322560, vCores:3> {noformat} # the preemption policy does the below calculation {noformat} 2015-06-23 23:20:51,127 NAME: QueueA CUR: <memory:0, vCores:0> PEN: <memory:0, vCores:0> GAR: <memory:491520, vCores:52> NORM: NaN IDEAL_ASSIGNED: <memory:0, vCores:0> IDEAL_PREEMPT: <memory:0, vCores:0> ACTUAL_PREEMPT: <memory:0, vCores:0> UNTOUCHABLE: <memory:0, vCores:0> PREEMPTABLE: <memory:0, vCores:0> 2015-06-23 23:20:51,128 NAME: default CUR: <memory:851968, vCores:91> PEN: <memory:0, vCores:0> GAR: <memory:491520, vCores:52> NORM: 1.0 IDEAL_ASSIGNED: <memory:851968, vCores:91> IDEAL_PREEMPT: <memory:0, vCores:0> ACTUAL_PREEMPT: <memory:0, vCores:0> UNTOUCHABLE: <memory:0, vCores:0> PREEMPTABLE: <memory:360448, vCores:39> {noformat} In the above log , observe for the queue default *CUR is <memory:851968, vCores:91>*, but actually *usedResources=<memory:10240, vCores:91>*. Here, only CPU is matching but not MEMORY. The CUR calculation is done below formula #* CUR= {{clusterResource: <memory:983040, vCores:105>}} * {{absoluteUsedCapacity(0.8)}} = {{<memory:851968, vCores:91>}} #* GAR= {{clusterResource: <memory:983040, vCores:105>}} *

{{absoluteCapacity(0.5)}} = {{ <memory:491520, vCores:52>}} #* PREEMPTABLE= GAR - CUR = {{<memory:360448, vCores:39>}} # App-2 request for the containers with {{resource: <memory:1024, vCores:10>}}. So, the preemption cycle finds that how much memory toBePreempt {noformat} 2015-06-23 23:21:03,131 | DEBUG | SchedulingMonitor (ProportionalCapacityPreemptionPolicy) | 1435072863131: NAME: default CUR: <memory:851968, vCores:91> PEN: <memory:0, vCores:0> GAR: <memory:491520, vCores:52> NORM: NaN IDEAL_ASSIGNED: <memory:491520, vCores:52> IDEAL_PREEMPT: <memory:97043, vCores:10> ACTUAL_PREEMPT: <memory:0, vCores:0> UNTOUCHABLE: <memory:0, vCores:0> PREEMPTABLE: <memory:360448, vCores:39> {noformat} Observe that *IDEAL_PREEMPT: <memory:97043, vCores:10>*, but app-2 in queue QueueA required only 10 CPU resource to be preempt, but memory to be preempt is 97043 but memory sufficiently available. Below is the calculations which does IDEAL_PREMPT, #* totalPreemptionAllowed = clusterResource: <memory:983040, vCores:105> * 0.1 = <memory:98304, vCores:10.5> #* totPreemptionNeeded = CUR - IDEAL_ASSIGNED = CUR: <memory:851968, vCores:91> #* scalingFactor = Resources.divide(drc, <memory:491520, vCores:52>, <memory:98304, vCores:10.5>, <memory:851968, vCores:91>); scalingFactor = 0.114285715 #* toBePreempted = CUR: <memory:851968, vCores:91> * scalingFactor(0.1139045128455529) = <memory:97368, vCores:10> {{resource-to-obtain = <memory:97043, vCores:10>}} *So the problem is in either of the below steps* # As [~sunilg] said, usedResources=<memory:10240, vCores:91> but preemption policy calculate wrongly that current used capacity as {{<memory:851968, vCores:91>}}. This is mainly becaue preemption policy is using absoluteCapacity for calculating for Current usage which always gives wrong result for one of the resources in DominantResourceAllocator used. I think, fraction should not be used which caused problem in DRC(Multi dimentional resources) instead we should be usedResource from CSQueue. # Even bypassing above step-1, toBePreempted calculated as resource-to-obtain: <memory:97043, vCores:10>. When a container marked for preemption, preemption policy subtract the marked container resources. I.e in the above log, resource-to-obtain will become *<memory:96043, vCores:0>* since each container memory is <1gb,10cores>. On next container marking, MEMORY has become DOMINANT and policy tries to fullfil memory i.e 96GB even CPU is fulfilled. The dominant change i.e scheduler allocates container with CPU dominant, but preemption policy going for MEMORY dominant causing the problem. This allows kills all the NON-AM containers. *And don't think that problem is only killing all the NON-AM containers but it continues loop:-(i.e when app-2 starts running containers in QueueA, app-1 ask for container request which preemption policy kill all the NON-Am containers from app-1. This repeats for ever, and both applications kills the tasks each others in loop which both applications never completes at all*

6. I understand now, this is a bad issue when DRF enabled. Thanks for explanation from [~sunilg] and [~rohithsharma]. Let me take a look at how to solve this issue.
7. I think the correct fix should be: Instead of using absUsed to compute current, we should using getQueueResourceUsage.getUsed(...) to get the current used. And add some tests, that should be enough. {code} QueueCapacities qc = curQueue.getQueueCapacities(); float absUsed = qc.getAbsoluteUsedCapacity(partitionToLookAt); float absCap = qc.getAbsoluteCapacity(partitionToLookAt); float absMaxCap = qc.getAbsoluteMaximumCapacity(partitionToLookAt); boolean preemptionDisabled = curQueue.getPreemptionDisabled(); Resource current = Resources.multiply(partitionResource, absUsed); Resource guaranteed = Resources.multiply(partitionResource, absCap); Resource maxCapacity = Resources.multiply(partitionResource, absMaxCap); {code} [~sunilg], do you want to take a shot about this?
8. Thank you [~leftnoteasy] for the pointer. Yes. It looks to me like the root cause of the issue is the usage of absoluteCapacity fraction in proportionalpreemption policy. And we could try directly use the real usage there as you mentioned. I will add some tests and give one. :)
9. Make sense, please try to run the test with/without the change. And if you have time, could you add the test for node partition preemption as well? Thanks, Wangda
10. For the test,how it would be using parameterized test class which uses defaultRC and dominatRC ?
11. Good suggestion [~rohithsharma], but the more urgent issue we need to solve now is currently we cannot specify CPU to tests. I think we can file a separated ticket for the parameterized test class.
12. I mean for TestProportionalPreemptinPolicy.
13. Yes [~leftnoteasy] and [~rohithsharma]. Thank you for the updates. It seems we cannot give CPU to the tests as of now. We can update that by changing buildPolicy. Meantime once this is handled, I will add case for nodepartition too.
14. Make sense [~sunilg].
15. Hi [~leftnoteasy], [~rohithsharma] Uploading an initial patch. I have changed TestProportionalCapacityPreemptionPolicy test framework to accommodate Vcores along with memory. Corrected few test cases also. Kindly share your opinion.
16. Thanks for working on this, [~sunilg]! The fix of Proportion..Policy looks good to me, some comments about test changes: - The string syntax to define resources looks great! :) - Instead of changing all test cases in TestProportional..Policy, could you make another overload method can take String[][]? This can avoid lots of changes for test cases - Initialization of ResourceCalculator should be a part of buildPolicy, for example, add a "boolean useDominateResourceCalculator" to buildPolicy - Could you change TestPro..PolicyForNodePartitions to accept CPU when doing queue/application mocking as well?
17. \\ \\ | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | pre-patch | 15m 57s | Pre-patch trunk compilation is healthy. | | {color:green}+1{color} | @author | 0m 0s | The patch does not contain any @author tags. | | {color:green}+1{color} | tests included | 0m 0s | The patch appears to include 2 new or modified test files. | | {color:green}+1{color} | javac | 7m 34s | There were no new javac warning messages. | | {color:green}+1{color} | javadoc | 9m 40s | There were no new javadoc warning messages. | | {color:green}+1{color} | release audit | 0m 23s | The applied patch does not increase the total number of release audit warnings. | | {color:green}+1{color} | checkstyle | 0m 47s | There were no new checkstyle issues. | | {color:red}-1{color} | whitespace | 0m 3s | The patch has 19 line(s) that end in whitespace. Use git apply --whitespace=fix. | | {color:green}+1{color} | install | 1m 36s | mvn install still works. | | {color:green}+1{color} | eclipse:eclipse | 0m 34s | The patch built with eclipse:eclipse. | | {color:green}+1{color} | findbugs | 1m 26s | The patch does not introduce any new Findbugs (version 3.0.0) warnings. | | {color:green}+1{color} | yarn tests | 51m 3s | Tests passed in hadoop-yarn-server-resourcemanager. | | | | 89m 8s | | \\ \\ || Subsystem || Report/Notes || | Patch URL | http://issues.apache.org/jira/secure/attachment/12742561/0001-YARN-3849.patch | | Optional Tests | javadoc javac unit findbugs checkstyle | | git revision | trunk / 147e020 | | whitespace | https://builds.apache.org/job/PreCommit-YARN-Build/8395/artifact/patchprocess/whitespace.txt | | hadoop-yarn-server-resourcemanager test log | https://builds.apache.org/job/PreCommit-YARN-Build/8395/artifact/patchprocess/testrun_hadoop-yarn-server-resourcemanager.txt | | Test Results | https://builds.apache.org/job/PreCommit-YARN-Build/8395/testReport/ | | Java | 1.7.0_55 | | uname | Linux asf902.gq1.ygridcore.net 3.13.0-36-lowlatency #63-Ubuntu SMP PREEMPT Wed Sep 3 21:56:12 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux | | Console output | https://builds.apache.org/job/PreCommit-YARN-Build/8395/console | This message was automatically generated.
18. Thank you [~leftnoteasy] for the comments. I have uploaded a patch by addressing the comments. Kindly check.
19. \\ \\ | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | pre-patch | 16m 18s | Pre-patch trunk compilation is healthy. | | {color:green}+1{color} | @author | 0m 0s | The patch does not contain any @author tags. | | {color:green}+1{color} | tests included | 0m 0s | The patch appears to include 2 new or modified test files. | | {color:green}+1{color} | javac | 7m 42s | There were no new javac warning messages. | | {color:green}+1{color} | javadoc | 9m 42s | There were no new javadoc warning messages. | | {color:green}+1{color} | release audit | 0m 21s | The applied patch does not increase the total number of release audit warnings. | | {color:green}+1{color} | checkstyle | 0m 49s | There were no new checkstyle issues. | | {color:red}-1{color} | whitespace | 0m 2s | The patch has 1 line(s) that end in whitespace. Use git apply --whitespace=fix. | | {color:green}+1{color} | install | 1m 35s | mvn install still works. | | {color:green}+1{color} | eclipse:eclipse | 0m 32s | The patch built with eclipse:eclipse. | | {color:green}+1{color} | findbugs | 1m 24s | The patch does not introduce any new Findbugs (version 3.0.0) warnings. | | {color:red}-1{color} | yarn tests | 60m 51s | Tests failed in hadoop-yarn-server-resourcemanager. | | | | 99m 18s | | \\ \\ || Reason || Tests || | Timed out tests | org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.TestNodeLabelContainerAllocation | \\ \\ || Subsystem || Report/Notes || | Patch URL | http://issues.apache.org/jira/secure/attachment/12743055/0002-YARN-3849.patch | | Optional Tests | javadoc javac unit findbugs checkstyle | | git revision | trunk / 7405c59 | | whitespace | https://builds.apache.org/job/PreCommit-YARN-Build/8405/artifact/patchprocess/whitespace.txt | | hadoop-yarn-server-resourcemanager test log | https://builds.apache.org/job/PreCommit-YARN-Build/8405/artifact/patchprocess/testrun_hadoop-yarn-server-resourcemanager.txt | | Test Results | https://builds.apache.org/job/PreCommit-YARN-Build/8405/testReport/ | | Java | 1.7.0_55 | | uname | Linux asf907.gq1.ygridcore.net 3.13.0-36-lowlatency #63-Ubuntu SMP PREEMPT Wed Sep 3 21:56:12 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux | | Console output | https://builds.apache.org/job/PreCommit-YARN-Build/8405/console | This message was automatically generated.
20. Test case failures are not related to this patch. TestNodeLabelContainerAllocation is passing locally in trunk.
21. Thanks [~sunilg], Some comments: Some comments: 1) It seems we don't need useDominantResourceCalculator/rcDefault/rcDominant in TestP..Policy, pass a boolean parameter to buildPolicy should be enough, you can also overload a buildPolicy to avoid too much changes. 2) testPreemptionWithVCoreResource seems not correct, root.used != A.used + b.used 3) TestP..PolicyFroNodePartitions: One comments is wrong: {code} + "(1,1:2,n1,x,20,false);" + // 80 * x in n1 "b\t" // app4 in b + "(1,1:2,n2,,80,false)"; // 20 default in n2 {code} It should be 20 * x and 80 default 4) It seems TestP..PolicyFroNodePartitions setting for DRC is missing, could you check?
22. Thank you [~leftnoteasy] for the comments. Uploading a patch addressing the issues. Regarding one comment, bq.testPreemptionWithVCoreResource seems not correct, root.used != A.used + b.used {noformat} "root(=[100:200 100:200 100:200 100:200],x=[100:200 100:200 100:200 100:200]);" "-a(=[50:100 100:200 20:40 50:100],x=[50:100 100:200 80:160 50:100]);" + // a "-b(=[50:100 100:200 80:160 50:100],x=[50:100 100:200 20:40 50:100])"; {noformat} Here now root.used = a.used+b.used. Please help to check.

23. [~sunilg], Thanks for update, but testPreemptionWithVCoreResource has the similar issue. {code} {"100:100", "10:100", "0"}, // used {code} Could you fix it as well?
24. Yes [~leftnoteasy] . You are correct, thanks for pointing out. I update the patch. :)
25. Kicking jenkins again.
26. \\ \\ | (x) *{color:red}-1 overall{color}* | \\ \\ || Vote || Subsystem || Runtime || Comment || | {color:blue}0{color} | pre-patch | 15m 55s | Pre-patch trunk compilation is healthy. | | {color:green}+1{color} | @author | 0m 0s | The patch does not contain any @author tags. | | {color:green}+1{color} | tests included | 0m 0s | The patch appears to include 2 new or modified test files. | | {color:green}+1{color} | javac | 7m 38s | There were no new javac warning messages. | | {color:green}+1{color} | javadoc | 9m 37s | There were no new javadoc warning messages. | | {color:green}+1{color} | release audit | 0m 22s | The applied patch does not increase the total number of release audit warnings. | | {color:green}+1{color} | checkstyle | 0m 46s | There were no new checkstyle issues. | | {color:red}-1{color} | whitespace | 0m 3s | The patch has 1 line(s) that end in whitespace. Use git apply --whitespace=fix. | | {color:green}+1{color} | install | 1m 33s | mvn install still works. | | {color:green}+1{color} | eclipse:eclipse | 0m 32s | The patch built with eclipse:eclipse. | | {color:green}+1{color} | findbugs | 1m 29s | The patch does not introduce any new Findbugs (version 3.0.0) warnings. | | {color:green}+1{color} | yarn tests | 51m 9s | Tests passed in hadoop-yarn-server-resourcemanager. | | | | 89m 7s | | \\ \\ || Subsystem || Report/Notes || | Patch URL | http://issues.apache.org/jira/secure/attachment/12743659/0004-YARN-3849.patch | | Optional Tests | javadoc javac unit findbugs checkstyle | | git revision | trunk / 688617d | | whitespace | https://builds.apache.org/job/PreCommit-YARN-Build/8431/artifact/patchprocess/whitespace.txt | | hadoop-yarn-server-resourcemanager test log | https://builds.apache.org/job/PreCommit-YARN-Build/8431/artifact/patchprocess/testrun_hadoop-yarn-server-resourcemanager.txt | | Test Results | https://builds.apache.org/job/PreCommit-YARN-Build/8431/testReport/ | | Java | 1.7.0_55 | | uname | Linux asf905.gq1.ygridcore.net 3.13.0-36-lowlatency #63-Ubuntu SMP PREEMPT Wed Sep 3 21:56:12 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux | | Console output | https://builds.apache.org/job/PreCommit-YARN-Build/8431/console | This message was automatically generated.
27. Latest patch LGTM, committing.
28. Committed to branch-2/trunk, thanks [~sunilg] and analysis from [~rohithsharma].
29. FAILURE: Integrated in Hadoop-trunk-Commit #8151 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/8151/]) YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (wangda: rev 1df39c1efc9ed26d3f1a5887c31c38c873e0b784) * hadoop-yarn-project/CHANGES.txt * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/ProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicyForNodePartitions.java
30. Thank you very much [~leftnoteasy] for reviewing and committing this patch. Thank you [~rohithsharma] for the analysis and review.
31. FAILURE: Integrated in Hadoop-Yarn-trunk-Java8 #254 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk-Java8/254/]) YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (wangda: rev 1df39c1efc9ed26d3f1a5887c31c38c873e0b784) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicyForNodePartitions.java * hadoop-yarn-project/CHANGES.txt * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/ProportionalCapacityPreemptionPolicy.java
32. FAILURE: Integrated in Hadoop-Yarn-trunk #984 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk/984/]) YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (wangda: rev 1df39c1efc9ed26d3f1a5887c31c38c873e0b784) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicyForNodePartitions.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/ProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/CHANGES.txt * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicy.java
33. FAILURE: Integrated in Hadoop-Mapreduce-trunk-Java8 #252 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk-Java8/252/]) YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (wangda: rev 1df39c1efc9ed26d3f1a5887c31c38c873e0b784) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicyForNodePartitions.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/CHANGES.txt * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/ProportionalCapacityPreemptionPolicy.java
34. FAILURE: Integrated in Hadoop-Hdfs-trunk-Java8 #242 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk-Java8/242/]) YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (wangda: rev 1df39c1efc9ed26d3f1a5887c31c38c873e0b784) * hadoop-yarn-project/CHANGES.txt * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicyForNodePartitions.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/ProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicy.java
35. SUCCESS: Integrated in Hadoop-Hdfs-trunk #2181 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk/2181/]) YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (wangda: rev 1df39c1efc9ed26d3f1a5887c31c38c873e0b784) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/ProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicyForNodePartitions.java * hadoop-yarn-project/CHANGES.txt
36. SUCCESS: Integrated in Hadoop-Mapreduce-trunk #2200 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk/2200/]) YARN-3849. Too much of preemption activity causing continuos killing of containers across queues. (Sunil G via wangda) (wangda: rev 1df39c1efc9ed26d3f1a5887c31c38c873e0b784) * hadoop-yarn-project/CHANGES.txt * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicyForNodePartitions.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/TestProportionalCapacityPreemptionPolicy.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/monitor/capacity/ProportionalCapacityPreemptionPolicy.java
37. Discussed with [~vinodkv], since 2.7.2 is almost done, set target version of this ticket to 2.7.3.
38. [~sunilg], I tried to apply patch to branch-2.7 but failed, could you update patch for branch-2.7? Thanks,
39. Yes. Is not applying there. i will share a 2.7 patch for same. Thank You [~leftnoteasy].
40. Attaching a branch2.7 patch. Locally the test cases are passing.
41. Thanks [~sunilg], patch looks good, will commit in a few days if no opposite opinions.
42. Committed to branch-2.7 as well, thanks [~sunilg]!
43. FAILURE: Integrated in Hadoop-trunk-Commit #8835 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/8835/]) move YARN-4326/YARN-3849 from 2.8.0 to 2.7.3 (wangda: rev a30eccb38c83b20af5d0705f9834165b74468314) * hadoop-yarn-project/CHANGES.txt

44. FAILURE: Integrated in Hadoop-Yarn-trunk #1428 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk/1428/]) move YARN-4326/YARN-3849 from 2.8.0 to 2.7.3 (wangda: rev a30eccb38c83b20af5d0705f9834165b74468314) * hadoop-yarn-project/CHANGES.txt
45. FAILURE: Integrated in Hadoop-Mapreduce-trunk-Java8 #691 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk-Java8/691/]) move YARN-4326/YARN-3849 from 2.8.0 to 2.7.3 (wangda: rev a30eccb38c83b20af5d0705f9834165b74468314) * hadoop-yarn-project/CHANGES.txt
46. FAILURE: Integrated in Hadoop-Yarn-trunk-Java8 #703 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk-Java8/703/]) move YARN-4326/YARN-3849 from 2.8.0 to 2.7.3 (wangda: rev a30eccb38c83b20af5d0705f9834165b74468314) * hadoop-yarn-project/CHANGES.txt
47. FAILURE: Integrated in Hadoop-Mapreduce-trunk #2632 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk/2632/]) move YARN-4326/YARN-3849 from 2.8.0 to 2.7.3 (wangda: rev a30eccb38c83b20af5d0705f9834165b74468314) * hadoop-yarn-project/CHANGES.txt
48. SUCCESS: Integrated in Hadoop-Hdfs-trunk #2562 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk/2562/]) move YARN-4326/YARN-3849 from 2.8.0 to 2.7.3 (wangda: rev a30eccb38c83b20af5d0705f9834165b74468314) * hadoop-yarn-project/CHANGES.txt
49. FAILURE: Integrated in Hadoop-Hdfs-trunk-Java8 #624 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk-Java8/624/]) move YARN-4326/YARN-3849 from 2.8.0 to 2.7.3 (wangda: rev a30eccb38c83b20af5d0705f9834165b74468314) * hadoop-yarn-project/CHANGES.txt
50. [~sunilg], [~leftnoteasy], does this apply to 2.6.x as well? Should this be backported to branch-2.6?
51. Yes, I think so. This will be a good addition in 2.6 line, I will try see to back port the same to 2.6.
52. Mark this JIRA target to 2.6.4 per discussion above.
53. Yes. [~junping_du] I will provide a patch for 2.6 now.
54. Attaching branch2.6 patch. Locally all test cases were passing. [~junping_du]/[~rohithsharma] Could you please take a look.
55. Back ported to 2.6.4. Thanks [~sunilg] for providing branch-2.6 patch Run test cases with and without patch to ensure test cases are healthy.
56. FAILURE: Integrated in Hadoop-trunk-Commit #9080 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/9080/]) Add YARN-3849 to Release 2.6.4 entry in CHANGES.txt (rohithsharmaks: rev 76e72708511100dbfaba910e120892b87b87edee) * hadoop-yarn-project/CHANGES.txt
57. Thanks [~sunilg] and [~rohithsharma]!
58. Closing the JIRA as part of 2.7.3 release.