Item 153
git_comments:

1. * * Number of maximum applications for each of the reservations in this Plan. * * @return maxAppsForreservation
2. Set new configs
3. Set the reservation queue attributes for the Plan
4. * * User limit value for each of the reservations in this Plan. * * @return userLimit
5. * * Determine whether to hide/show the ReservationQueues
6. Sanity check
7. * * This represents a dynamic queue managed by the {@link ReservationSystem}. * From the user perspective this is equivalent to a LeafQueue that respect * reservations, but functionality wise is a sub-class of ParentQueue *
8. * * Number of maximum applications per user for each of the reservations in * this Plan. * * @return maxAppsPerUserForreservation
9. run reinitialize on each existing queue, to trigger absolute cap recomputations
10. * * User limit factor value for each of the reservations in this Plan. * * @return userLimitFactor
11. * * This methods to change capacity for a queue and adjusts its * absoluteCapacity * * @param entitlement the new entitlement for the queue (capacity, * maxCapacity, etc..) * @throws SchedulerDynamicEditException
12. * * This represents a dynamic {@link LeafQueue} managed by the * {@link ReservationSystem} *
13. note: we currently set maxCapacity to capacity this might be revised later
14. used by the super constructor, we initialize to zero
15. the following parameters are common to all reservation in the plan
16. Sanity check
17. create the default reservation queue
18. Test add another reservation queue and use setEntitlement to modify capacity
19. Reinitialize and verify all dynamic queued survived
20. Define 2nd-level queues
21. Verify all allocations match
22. clear queue by killling all apps
23. submit an app
24. wait for events of move to propagate
25. now move the app to plan queue
26. Test add one reservation dynamically and manually modify capacity
27. expected
28. Test invalid entitlement (sum of queues exceed 100%)
29. set capacity to zero
30. Test invalid addition (adding non-zero size queue)
31. expected a1 contains applications
32. check postconditions
33. expected a1 is not zero capacity
34. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
35. check preconditions
36. create a queue
37. setup a context / conf
38. expected
39. * * Licensed to the Apache Software Foundation (ASF) under one * or more contributor license agreements. See the NOTICE file * distributed with this work for additional information * regarding copyright ownership. The ASF licenses this file * to you under the Apache License, Version 2.0 (the * "License"); you may not use this file except in compliance * with the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
40. verify that setting, adding, subtracting capacity works
41. * * This method increase the entitlement for current queue (must respect * invariants, e.g., no overcommit of parents, non negative, etc.). * Entitlement is a general term for weights in FairScheduler, capacity for * the CapacityScheduler, etc. * * @param queue the queue for which we change entitlement * @param entitlement the new entitlement for the queue (capacity, * maxCapacity, etc..) * @throws YarnException
42. * * Add to the scheduler a new Queue. Implementations might limit what type of * queues can be dynamically added (e.g., Queue must be a leaf, must be * attached to existing parent, must have zero entitlement). * * @param newQueue the queue being added. * @throws YarnException
43. * * Gets the list of names for queues managed by the Reservation System * @return the list of queues which support reservations
44. * * Remove an existing queue. Implementations might limit when a queue could be * removed (e.g., must have zero entitlement, and no applications running, or * must be a leaf, etc..). * * @param queueName name of the queue to remove * @throws YarnException
45. at this point we should have no more apps
46. check that all static queues are included in the newQueues list
47. Check if the queue is a plan queue
48. note: epsilon checks here are not ok, as the epsilons might accumulate and become a problem in aggregate
49. use the default child reservation queue of the plan
50. Check if the queue will be dynamically managed by the Reservation system
51. use the default child queue of the plan for unreserved apps
52. use the reservation queue to run the app
53. externalizing in method, to allow overriding

git_commits:

1. **summary:** YARN-1707. Introduce APIs to add/remove/resize queues in the CapacityScheduler. Contributed by Carlo Curino and Subru Krishnan

message: YARN-1707. Introduce APIs to add/remove/resize queues in the CapacityScheduler. Contributed by Carlo Curino and Subru Krishnan

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
   label: code-design
2. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
3. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
   label: test
4. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
5. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
6. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
7. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
   label: code-design
8. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
9. **summary:** Making the CapacityScheduler more dynamic
   description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
10. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
    label: code-design

11. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
12. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
13. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
    label: code-design
14. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
15. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
    label: code-design
16. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
17. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
18. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
    label: code-design
19. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
20. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
21. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
22. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
    label: code-design
23. **summary:** Making the CapacityScheduler more dynamic

description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.

24. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
25. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
26. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
27. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
28. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
    label: code-design
29. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
30. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
31. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
32. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
    label: code-design
33. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
34. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
35. **summary:** Making the CapacityScheduler more dynamic
    description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the

following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
label: test
36. **summary:** Making the CapacityScheduler more dynamic
description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
37. **summary:** Making the CapacityScheduler more dynamic
description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
38. **summary:** Making the CapacityScheduler more dynamic
description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
39. **summary:** Making the CapacityScheduler more dynamic
description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
40. **summary:** Making the CapacityScheduler more dynamic
description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
41. **summary:** Making the CapacityScheduler more dynamic
description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
42. **summary:** Making the CapacityScheduler more dynamic
description: The CapacityScheduler is a rather static at the moment, and refreshqueue provides a rather heavy-handed way to reconfigure it. Moving towards long-running services (tracked in YARN-896) and to enable more advanced admission control and resource parcelling we need to make the CapacityScheduler more dynamic. This is instrumental to the umbrella jira YARN-1051. Concretely this require the following changes: * create queues dynamically * destroy queues dynamically * dynamically change queue parameters (e.g., capacity) * modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% We limit this to LeafQueues.
label: code-design

jira_issues_comments:

1. **body:** I'm very supportive of features like removing queues (adding is already supported). However, as I just commented on YARN-1051, I think we are better of relying on enhancing/reducing priorities rather than adding/removing queues.
label: code-design
2. The attached patch is part of the YARN-1051 effort, as for the other patches in this series does not work on itself but it has been cut for ease of reviewing. Given previous discussions, we introduced subclasses for ParentQueue and LeafQueue that are dynamically addable/removable/resizeable, as well as changes in the CapacityScheduler to support the "move" of applications across queues. These are core features, we tested on a cluster running lots of gridmix and manual jobs, and seems to work fine, but I am sure there are corner cases and possibly metrics that are not updated correctly under all cases. We should also create a new set of tests for the dynamic behavior of the CapacityScheduler.
3. **body:** {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12657926/YARN-1707.patch against trunk revision . {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:red}-1 tests included{color}. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4439//console This message is automatically generated.
label: test
4. Thanks for uploading the patch [~curino], [~subru]. They're great additions to current CapacityScheduler. I took a look at your patch, *First I have a couple of questions about its background, especially {{PlanQueue}}/{{ReservationQueue}} in this patch. I think understanding background is important for me to get a whole picture of this patch. What I can understand is,* # {{PlanQueue}} can have a normal {{ParentQueue}} as its parent, but all children of {{PlanQueue}} can only be {{ReservationQueue}}. Is it possible that multiple {{PlanQueue}} exist in the cluster? # {{PlanQueue}} is initially setup in configuration, as same as {{ParentQueue}}, it has absolute capacity, etc. But different from {{ParentQueue}}, it has user-limit/user-limit-factor, etc. # {{ReservationQueue}} is dynamically initialized by PlanFollower, when a new reservationId acquired, it will create a new {{ReservationQueue}} accordingly # {{PlanFollower}} can dynamically adjust queue size of {{ReservationQueue}}s to make resource reservation can be satisfied. # Is it possible that sum of reserved resource exceeds limit of {{PlanQueue}}/{{ReservationQeueu}} and preemption triggered? # How to deal with RM restart? It is possible that RM restart during resource reservation, we may need to consider how to persistent such queues Hope you could share your ideas about them. *For requirement of this ticket (copied from JIRA),* # create queues dynamically # destroy queues dynamically # dynamically change queue parameters (e.g., capacity) # modify refreshqueue validation to enforce sum(child.getCapacity())<= 100% instead of ==100% # move app across queues I found #1-#3 are dedicated used by {{PlanQueue}}, {{Reservation}}. IMHO, it should be better to added them to

CapacityScheduler and don't couple them with ReservationSystem, but I cannot think about other solid senarios can leverage them. I hope to get feedbacks from community before we couple them with ReservationSystem. And as mentioned by [~acmurthy], can we merge add queue to existing add new queue mechanism? #4 should be only valid in {{PlanQueue}}. Because if we change this behavior in {{ParentQueue}}, it is possible that some careless admin will mis-setting capacities of queues under a parent queue, if sum of their capacity don't equals to 1, some resource may not be able to be used by applications. *Some other comments (Majorly about move app because we may need consider scope of create/destory queues first):* 1) I think we need consider how moving apps across queues work with YARN-1368. We can change queue of containers from queueA to queueB, but with YARN-1368, during RM restart, container will report it is in queueA (we don't sync them to NM when do moveApp operation). I hope [~jianhe] could share some thoughts about this as well. 2) Move application in CapacityScheduler need call finishApplication in resource queue and submitApplication in target queue to make QueueMetrics correct. And submitApplication will check ACL of target queue as well. 3) Should we respect MaxApplicationsPerUser in target queue when trying to move app? IMHO, we can stop moving app if MaxApplicationsPerUser reached in target queue. Thanks, Wangda

5. [~wangda] Thanks for the very detailed comments. I agree that understanding the context is essential & glad to help with that. Overall your understanding is spot on, please find answers to your questions below: 1) Yes, it is possible to have multiple PlanQueues (e.g., if two organization want to dynamically allocate their resources, but not share among them). This is also good to "try" reservation on a small scale and slowly ramp up at each org's pace. 2) The extra confs are needed to automate the initialization of key parameters of the dynamic ReservationQueues (without requiring full specification of each of those). 3) Correct 4) Correct 5) First: the Plan guarantees that the sum of reservations never exceed available resources (replanning if needed to maintain this invariant to handle failures). On the other hand, like it happens for normal scheduler we can leverage "overcapacity" to guarantee high cluster utilization. More precisely, depending on the configuration (or dynamically on whether reservations have gang semantics or not) we can allow resources allocated to PlanQueue and ReservationQueue to exceed their guaranteed capacity (i.e., set the dynamic max-capacity above the guaranteed one). In this case preemption might kick in if other apps with more rights on resources have pending askss. Part of the changes in YARN-1957 were driven by this. 6) To limit the scope of changed, we agreed to have a follow up JIRA to address HA. The intuition we have is that it is sufficient to persist the Plan alone. During recovery, the _Plan Follower_ will resync the Plan with the scheduler by creating the dynamic queues for currently active reservations. We will be happy to have your input when we work on the HA JIRA. [~curino] will answer your questions specify to this JIRA.
6. Hi [~subru], Thanks for your elaboration, it is very helpful for me to understand the background. Regards, Wangda
7. **body:** Thanks again for the fast and insightful feedback. *Regarding how the patch matches the JIRA:* Our initial implementation was indeed making the changes (i.e., the dynamic behaviors) in ParentQueue and LeafQueue themselves. Previous feedback pushed us to have subclasses to in a sense isolate the changes to dynamic subclasses. I think we can go back to the version modifying directly ParentQueue and LeafQueue if there is consensus. #4 is required because we cannot transactionally “add Q1, resize Q2” so that the invariant “size of children is == 100%” is maintained. As a consequence we must relax the constraints (either in ParentQueue if we remove the hierarchy, or as it is today in PlanQueue). The good news is that the percentages from the configuration are not interpreted as actual percentages, but rather used as relative "weights" (ranking queues in used_resources / guaranteed_resources). This means that even a careless admin will not get resources unused. For example, if we set two queues to 10,40 (i.e., something that doesn't add up to 100), the behavior is equivalent to setting them to 20,80 (as they are used only for relative ranking of siblings). I think this is also ok for hierarchies (worth double checking this part). So all in all we can pull up to {{ParentQueue}} and {{LeafQueue}} all the dynamic behavior if there is consensus that this is the right path. *Regarding move:* 1) Good catch... We will wait for feedback from Jian on this. 2) I think we had that at some point and did not work correctly. We will try again. 3) There are few invariants we do not check. {{MaxApplicationsPerUser}} is one of them, but also how many applications can be active in the target queue, etc... As I was mentioning in my previous comment, this is likely fine for the limited usage we will make of this from {{ReservationSystem}}, but it is worth expand the checks we make (see {{FairScheduler.verifyMoveDoesNotViolateConstraints(..)}}) to expose move to users via CLI.
   label: code-design
8. Hi [~curino], Thanks for your reply, For regarding how the patch matches the JIRA: Since I don't have other solid use cases in my mind that others besides {{ReservationSystem}} can leverage these features, I don't have strong opinions to merge such dynamic behaviors into {{ParentQueue}}, {{LeafQueue}}. Let's wait for more feedbacks. I agree that we can consider queue capacity as a "weight", it will be easier for users to configure, and it's a backward-compatible change also (except it will not throw exception when sum of children of a {{ParentQueue}} doesn't equals to 100). bq. As I was mentioning in my previous comment, this is likely fine for the limited usage we will make of this from ReservationSystem I think for moving application across queue is not a ReservationSystem specific change. I would suggest to check it will not violate restrictions in target queue before moving it. Thanks, Wangda
9. Agreed on all of the above. {quote} I think for moving application across queue is not a ReservationSystem specific change. I would suggest to check it will not violate restrictions in target queue before moving it. {quote} This makes sense, we should compile a list of invariant to check for (I have a few in mind, but feedback is likely useful). Thanks, Carlo
10. **body:** This patch is a more "minimal" set of changes rebased on trunk after we committed to trunk YARN-2378, YARN-2389. We also simplified and added more tests. The dynamic behavior is for PlanQueue and ReservationQueue.
    label: code-design
11. Minor fixes
12. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12663294/YARN-1707.2.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 3 new or modified test files. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4690//console This message is automatically generated.
13. **body:** Hi [~curino], Thanks for updating, I just took a look, some minor comments, 1) CapacityScheduler#removeQueue {code} if (disposableLeafQueue.getCapacity() > 0) { throw new SchedulerConfigEditException("The queue " + queueName + " has non-zero capacity: " + disposableLeafQueue.getCapacity()); } {code} removeQueue check disposableLeafQueue's capacity > 0, but addQueue doesn't check. In addition, After previous check, ParentQueue#removeChildQueue/addChildQueue doesn't need check its capacity again. And they should throw same type of exception (both SchedulerConfigEditException or both IllegalArgumentException) 2) CS#addQueue {code} throw new SchedulerConfigEditException("Queue " + queue.getQueueName() + " is not a dynamic Queue"); {code} Should "dynamic Queue" should be "reservation queue" comparing to similar exception throw in removeQueue? 3) CS#setEntitlement {code} if (sesConf.getCapacity() > queue.getCapacity()) { newQueue.addCapacity((sesConf.getCapacity() - queue.getCapacity())); } else { newQueue .subtractCapacity((queue.getCapacity() - sesConf.getCapacity())); } {code} Maybe it's better to merge the add/substractCapacity to changeCapacity(delta) Or just create a "setCapacity" in ReservationQueue? 4) CS#getReservableQueues Is it better to rename it to getPlanQueues? 5) ReservationQueue#getQueueName {code} @Override public String getQueueName() { return this.getParent().getQueueName(); } {code} I'm not sure why doing this, could you please elaborate? This makes this.queueName and this.getQueueName has different semantic. 6) ReservationQueue#substractCapacity {code} this.setCapacity(this.getCapacity() - capacity); {code} With EPSILON, it is possible this.capacity < 0 set substract, its better to cap this.capacity in range of [0,1]. Also addCapacity 7) DynamicQueueConf I think unfold it to two float as parameter for setEntitlement maybe more straigtforward, is it possible more fields will be add to DynamicQueueConf? 8) ParentQueue#setChildQueues Since only PlanQueue need sum of capacity <= 1, I would suggest make this method protected, and PlanQueue can overwrite this method. Or add a check in ParentQueue#setChildQueues. Wangda
    label: code-design

14. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12663571/YARN-1707.3.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 3 new or modified test files. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4719//console This message is automatically generated.
15. **body:** Wangda, Thanks for the great feedback. You spot a bunch of oddities that were there due to previous versions of the reservation system, but not needed anymore, I think the updated version is definitely cleaner. We address 1,2 by: * moving addQueue and removeQueue to PlanQueue (as they were only invoked on instances of the subclass). * making uniform checks from within the PlanQueue for capacity > 0, and throw uniform SchedulerConfigEditException * fixing the log, and making the logs more uniform We address 3,6 by: * merge addCapacity and subtractCapacity into a single changeCapacity * make checks of range limits 0,1 (this reduced code both in CS and ReservationQueue... good call!) We address 4 by: * getReservableQueues() has been renamed to getPlanQueues() Regarding 5: ReservationQueue#getQueueName * This is the result of our previous conversations with Vinod, Bikas, and Arun. The idea is that the user should not be aware of the fact that we use queues to implement reservations, and thus it shouldn't see the name of the reservation queue to be listed in the UI, but rather the name of the parent PlanQueue. More precisely, we have options for the UI to show or not the subqueues, but this differentiation is needed here to allow that: getQueueName for a ReservationQueue return the parent, while getReservationQueueName() returns the actual local name. Regarding 7: DynamicQueueConf * We currently are only dynamically assigning capacity, but you can imagine in the future that this is extended to set many more parameters for a queue (user-limit factors, max applications, etc..). The conf-based mechanism is future-proofing against this. Regarding 8: ParentQueue#setChildQueues * I don't understand the comment. This check is automatically bypassed for PlanQueue (that by design have no children see CapacityScheduler near line 562). We are testing the new version of the patch now, and will post patch soon.
label: code-design
16. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12664517/YARN-1707.4.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 3 new or modified test files. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4737//console This message is automatically generated.
17. Hi [~curino], Thanks for updating, I think current approach looks good to me, except Regarding 5, I just have a chat with Subru, as you mentioned, changing this is majorly making ReservationQueues not existed in user side. But I still concern about changing the semantic, since it's still a very important semantic of CSQueue. I hope to get more feedbacks about this before moving forward, I'll think about this myself as well. Thanks, Wangda
18. **body:** I share the uncertainty towards getQueueName returning the name of the parent. It seems the least intrusive way to mask the user from our internal queue management. The alternative is to use a QueueName class instead of String, and allow the receiver to use a "displayName" or "localName"... but this has awfully broad footprint. Alternatively we can add a "getDisplayName" to CSQueue to be used by the UI, and implement it everywhere as getQueueName except in ReservationQueue where we can answer with parent name. Again a bit large of a change for a rather minor outcome. Thanks, Carlo
label: code-design
19. Thanks for sharing this, Carlo! It's very helpful to have such investigation result, any thoughts, [~jianhe]? Wangda
20. The patch seems missing some file changes and cause compilation failure. (e.g. getReservationID)
21. [~jianhe] that is expected. As I was saying in one of the [early comments | https://issues.apache.org/jira/browse/YARN-1707?focusedCommentId=14075076&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-14075076] we are cutting YARN-1051 into several smaller patches for ease of reviewing, but we are trying to make each patch work as standalone (too many dependencies, and a bit of a waste of time, as they will not be valuable independently). So the fact that doesn't compile is expected. We mark them as patch available to signal they are ready to be reviewed. [~wangda]: we have implemented the getDisplayName alternative I mentioned above, and we are in the process of testing it. We will post an updated patch soon (again not a stand-alone one necessarily). Thanks again to both of you for quick rounds of review and insightful comments.
22. **body:** Hi Carlo, thanks for your work ! I looked at the patch, some comments and questions: - to simplify, we can use getNumApplications() method {code} disposableLeafQueue.getApplications().size() > 0 || disposableLeafQueue.pendingApplications.size() > 0 {code} - PlanQueue.java 80 column limit - why “newQueue.changeCapacity(sesConf.getCapacity());” is inside the check and “queue.setMaxCapacity(sesConf.getMaxCapacity());” is outside the check - CapacityScheduler#getReservationQueueNames seems getting the child reservation queues of the given plan queue. We can use the planQueue#childQueues directly - DynamicQueueConf, how about calling it QueueEntitlement to be consistent ? - CapacityScheduler#parseQueue method, I think we can simplify the condition for isReservableQueue flag something like this: {code} boolean isReservableQueue = conf.isReservableQueue(fullQueueName); if (isReservableQueue) { ParentQueue parentQueue = new PlanQueue(csContext, queueName, parent, oldQueues.get(queueName)); queue = hook.hook(parentQueue); } else if ((childQueueNames == null || childQueueNames.length == 0)) {code} - just to simplify, this log msg may be put after previous “qiter.remove();” to avoid the removed boolean flag. {code} if (LOG.isDebugEnabled()) { LOG.debug("updateChildQueues (action: remove queue): " + removed + " " + getChildQueuesToPrint()); } {code} - we can add a new reinitialize in ReservationQueue which does all these initializations. {code} CSQueueUtils.updateQueueStatistics(schedulerContext.getResourceCalculator(), ses, this, schedulerContext.getClusterResource(), schedulerContext.getMinimumResourceCapability()); ses.reinitialize(ses, clusterResource); ((ReservationQueue) ses).setMaxApplications(this .getMaxApplicationsForReservations()); ((ReservationQueue) ses).setMaxApplicationsPerUser(this .getMaxApplicationsPerUserForReservation()); {code} - IIUC, right now, queueName here is for the planQueue(inherits parentQueue), and the reservationID is for the reservationQueue(inherits from leafQueue). I think if we can get the proper reservationQueueName(leafQueue) upfront and pass it as the queueName parameter into this method, we can avoid some if/else condition changes inside this method and the method signature. {code} private synchronized void addApplication(ApplicationId applicationId, String queueName, String user, boolean isAppRecovering, ReservationId reservationID) {code}
label: code-design
23. [~jianhe] Thanks for the feedback... The version I just posted contains the getDisplayName implementation, but does not address your last comments yet. We will get to those next.
24. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12665477/YARN-1707.5.patch against trunk revision 9ad413b. {color:red}-1 patch{color}. Trunk compilation may be broken. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4780//console This message is automatically generated.
25. Carlo, thanks updating the patch. In addition to Jian's comment, I think the changes for displayQueueName looks good to me. I don't have further comments about this patch for now. Thanks, Wangda
26. Thanks [~jianhe] for your feedback. I am uploading a new patch that addresses your comments.
27. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12666118/YARN-1707.6.patch against trunk revision 08a9ac7. {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:red}-1 javadoc{color}. The javadoc tool appears to have generated 3 warning messages. See https://builds.apache.org/job/PreCommit-

YARN-Build/4805//artifact/trunk/patchprocess/diffJavadocWarnings.txt for details. {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:red}-1 core tests{color}. The patch failed these unit tests in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager: org.apache.hadoop.yarn.server.resourcemanager.TestMoveApplication org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.TestCapacitySchedulerQueueACLs The following test timeouts occurred in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager: org.apache.hadoop.yarn.server.resourcemanager.applicationsmanager.TeTests org.apache.hadoop.yarn.server.resourcemanager.TestWorkPreservinTests org.apache.hadoop.yarn.server.resourcemanager.TesTests {color:green}+1 contrib tests{color}. The patch passed contrib unit tests. Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4805//testReport/ Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4805//console This message is automatically generated.

28. **body:** - rename dyQConf/sesConf to entitleMent {code} setEntitlement(String queue, QueueEntitlement dyQConf) public synchronized void setEntitlement(String inQueue, QueueEntitlement sesConf) throws SchedulerConfigEditException, {code} - should userLimit be reinitialized in ReservationQueue/PlanQueue as well ? {code} setUserLimit(parent.getUserLimitForReservation()); setUserLimitFactor(parent.getUserLimitFactor()); {code} - I think the newlyParsedParentQueue doesn't have the updated value at this point? {code} this.maxAppsForReservation = newlyParsedParentQueue.maxAppsForReservation; this.maxAppsPerUserForReservation = newlyParsedParentQueue.maxAppsPerUserForReservation; {code} - indentation format {code} } else { queue = new LeafQueue(csContext, queueName, parent,oldQueues.get(queueName)); // Used only for unit tests queue = hook.hook(queue); } {code} - SchedulerConfigEditException, a better name ? this exception is used in multiple scenarios. - PlanQueue#showReservationsAsQueues / showReservationsAsLeafs, make boolean flag name and method name consistent
label: code-design
29. Thanks [~jianhe] for your comments. I am updating a patch that has the following fixes: * renamed dyQConf/sesConf to entitlement * userLimit be reinitialized in ReservationQueue/PlanQueue * Indendation fixed * Renamed SchedulerConfigEditException to SchedulerDynamicEditException * Consistently used showReservationsAsQueues for both method as well as the flag The newly parsed queues will have the maxApps* as CapacityScheduler#reinitialize() invokes parseQueues() which is where they are updated.
30. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12666353/YARN-1707.7.patch against trunk revision d9a03e2. {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 3 new or modified test files. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4819//console This message is automatically generated.
31. Subra, thanks for your update, looks good to me overall, we are almost there. just few minor things: - Given we have the hideReservationQueues flag, I think if we still return reserveQueue name for the reservationQueue, it won't be exposed. {code} @Override public String getDisplayName() { return this.getParent().getQueueName(); } {code} - the visibility may not need to be changed {code} - private boolean unreserve(FiCaSchedulerApp application, Priority priority, + protected boolean unreserve(FiCaSchedulerApp application, Priority priority, {code} - why we added this null check. {code} if (application != null) { synchronized (application) { return assignReservedContainer(application, node, reservedContainer, clusterResource); } } {code} - I think we may pass the EntitileMent class into the changeCapacity method and update both capacity and maxCapacity {code} newQueue.changeCapacity(entitlement.getCapacity()); // note: we currently set maxCapacity to capacity // this might be revised later queue.setMaxCapacity(entitlement.getMaxCapacity()); {code} - the YarnException type may not be needed {code} QueueEntitlement entitlement) throws SchedulerDynamicEditException, YarnException { {code}
32. **body:** Thanks [~jianhe] for your insights. I am uploading a new patch that has the following fixes: * Removed display name * Reverted unnecessary visibility change & null check * Pass QueueEnititlement to changeCapacity() * Handling move to Plan Queue, including unit test case I have not removed YarnException from setEntitlement as it is thrown getAndCheckLeafQueue()
label: code-design
33. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12666416/YARN-1707.8.patch against trunk revision 8f1a668. {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 3 new or modified test files. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4820//console This message is automatically generated.
34. Uploading a new patch with a minor change. Renamed ReservationQueue#changeCapacity to ReservationQueue#setEntitlement for consistency.
35. **body:** {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12666598/YARN-1707.9.patch against trunk revision 51a4faf. {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:red}-1 tests included{color}. The patch doesn't appear to include any new or modified tests. Please justify why no new tests are needed for this patch. Also please list what manual steps were performed to verify this patch. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4829//console This message is automatically generated.
label: test
36. +1 for the latest patch, thanks [~subru] and [~curino] !
37. Thanks [~jianhe] and [~leftnoteasy] for taking the time to do a thorough review. I am proxying for [~curino] also as he did most of the work for the patch. As discussed we will commit this to YARN-1051 branch once we have +1s for few other sub-JIRAs.
38. Rebased patch after sync-ing branch yarn-1051 with trunk
39. {color:red}-1 overall{color}. Here are the results of testing the latest attachment http://issues.apache.org/jira/secure/attachment/12668203/YARN-1707.10.patch against trunk revision 6c08339. {color:green}+1 @author{color}. The patch does not contain any @author tags. {color:green}+1 tests included{color}. The patch appears to include 3 new or modified test files. {color:red}-1 javac{color:red}. The patch appears to cause the build to fail. Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4899//console This message is automatically generated.
40. I committed this to the YARN-1051 branch. Thanks [~jianhe] and [~leftnoteasy] for the reviews.
41. FAILURE: Integrated in Hadoop-trunk-Commit #6189 (See [https://builds.apache.org/job/Hadoop-trunk-Commit/6189/]) YARN-1707. Introduce APIs to add/remove/resize queues in the CapacityScheduler. Contributed by Carlo Curino and Subru Krishnan (cdouglas: rev eb3e40b833b9d82c1556843f960194dc42e482f3) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestReservationQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AbstractYarnScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerDynamicEditException.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-

resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/QueueEntitlement.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/webapp/dao/CapacitySchedulerQueueInfo.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ReservationQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacitySchedulerDynamicBehavior.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/YarnScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ParentQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/PlanQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/CapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/LeafQueue.java

42. SUCCESS: Integrated in Hadoop-Yarn-trunk #700 (See [https://builds.apache.org/job/Hadoop-Yarn-trunk/700/]) YARN-1707. Introduce APIs to add/remove/resize queues in the CapacityScheduler. Contributed by Carlo Curino and Subru Krishnan (cdouglas: rev eb3e40b833b9d82c1556843f960194dc42e482f3) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/PlanQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/LeafQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerDynamicEditException.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AbstractYarnScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestReservationQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/CapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacitySchedulerDynamicBehavior.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/QueueEntitlement.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ReservationQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/webapp/dao/CapacitySchedulerQueueInfo.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ParentQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/YarnScheduler.java
43. SUCCESS: Integrated in Hadoop-Hdfs-trunk #1891 (See [https://builds.apache.org/job/Hadoop-Hdfs-trunk/1891/]) YARN-1707. Introduce APIs to add/remove/resize queues in the CapacityScheduler. Contributed by Carlo Curino and Subru Krishnan (cdouglas: rev eb3e40b833b9d82c1556843f960194dc42e482f3) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/YarnScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/LeafQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AbstractYarnScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ReservationQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/QueueEntitlement.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacitySchedulerDynamicBehavior.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerDynamicEditException.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/webapp/dao/CapacitySchedulerQueueInfo.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/CapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/PlanQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ParentQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestReservationQueue.java
44. FAILURE: Integrated in Hadoop-Mapreduce-trunk #1916 (See [https://builds.apache.org/job/Hadoop-Mapreduce-trunk/1916/]) YARN-1707. Introduce APIs to add/remove/resize queues in the CapacityScheduler. Contributed by Carlo Curino and Subru Krishnan (cdouglas: rev eb3e40b833b9d82c1556843f960194dc42e482f3) * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-

resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AbstractYarnScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/LeafQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/YarnScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/QueueEntitlement.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacitySchedulerDynamicBehavior.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestCapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ParentQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/webapp/dao/CapacitySchedulerQueueInfo.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerDynamicEditException.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/PlanQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/CapacityScheduler.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/TestReservationQueue.java * hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/capacity/ReservationQueue.java