Preemption caused Invalid State Event: ACQUIRED at KILLED and caused a task timeout for 30mins

Details

Type:	Sub-task	Status:	CLOSED
Priority:	Major	Resolution:	Fixed
Affects Version/s:	2.2.0	Fix Version/s:	2.5.0
Component/s:	resourcemanager		
Labels:	None		
Target Version/s:	2.5.0		

Description

Capacity preemption is enabled as follows.

- yarn.resourcemanager.scheduler.monitor.enable= true ,
- yarn.resourcemanager.scheduler.monitor.policies=org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCap

Queue = a,b
Capacity of Queue A = 80%
Capacity of Queue B = 20%

Step 1: Assign a big jobA on queue a which uses full cluster capacity
Step 2: Submitted a jobB to queue b which would use less than 20% of cluster capacity

JobA task which uses queue b capcity is been preempted and killed.

This caused below problem:
1. New Container has got allocated for jobA in Queue A as per node update from an NM.
2. This container has been preempted immediately as per preemption.

Here ACQUIRED at KILLED Invalid State exception came when the next AM heartbeat reached RM.
ERROR org.apache.hadoop.yarn.server.resourcemanager.rmcontainer.RMContainerImpl: Can't handle this event at current state
org.apache.hadoop.yarn.state.InvalidStateTransitonException: Invalid event: ACQUIRED at KILLED

This also caused the Task to go for a timeout for 30minutes as this Container was already killed by preemption.
attempt_1380289782418_0003_m_000000_0 Timed out after 1800 secs

Attachments

Yarn-1408.1.patch	2 kB	13/Nov/13 10:26
Yarn-1408.10.patch	36 kB	12/Jul/14 09:39
Yarn-1408.11.patch	36 kB	15/Jul/14 08:09
Yarn-1408.2.patch	1 kB	31/Dec/13 05:09
Yarn-1408.3.patch	6 kB	14/Feb/14 10:12
Yarn-1408.4.patch	7 kB	14/Feb/14 11:40
Yarn-1408.5.patch	27 kB	27/Jun/14 16:57
Yarn-1408.6.patch	30 kB	30/Jun/14 13:07
Yarn-1408.7.patch	36 kB	04/Jul/14 02:13
Yarn-1408.8.patch	35 kB	10/Jul/14 17:14
Yarn-1408.9.patch	36 kB	11/Jul/14 15:46
Yarn-1408.patch	2 kB	13/Nov/13 06:14
YARN-1408-branch-2.5-1.patch	38 kB	15/Jul/14 23:47

Issue Links

is blocked by

YARN-2143 Merge common killContainer logic of Fair/Capacity scheduler into AbstractYarnScheduler — OPEN

Activity

Sunil G added a comment - 13/Nov/13 06:16

A patch is given based on 2.2.0

Fix reason:
A container got ALLOCATED as per Node Update from NM and the state for RMContainer is in ALLOCATE state for job in Queue A.
This time it is present in 2 lists:
newlyAllocatedContainers and liveContainers
At the same time, due to preemption this container has got KILLED.
Now it is removed from liveContainers.

Immediately when next ask came from AM for jobA, this container is given to that as it is present in newlyAllocatedContainers. And this caused a Invalid Event called ACQUIRED at KILLED.
In the heartbeat response to AM, this container is present in newlyAllocated container. So a Task will launch and will result in issue2.

So to fix the same, when KILL_CONTAINER is processed from proportional policy, remove from newlyAllocatedContainers also if present.

Hadoop QA added a comment - 13/Nov/13 09:28

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12613538/Yarn-1408.patch
against trunk revision .

-1 patch. The patch command could not apply the patch.

Console output: https://builds.apache.org/job/PreCommit-YARN-Build/2440//console

This message is automatically generated.

Sunil G added a comment - 13/Nov/13 10:26

Updated patch against the trunk

Hadoop QA added a comment - 13/Nov/13 11:07

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12613569/Yarn-1408.1.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

-1 tests included. 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.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. The javadoc tool did not generate any warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

+1 core tests. The patch passed unit tests in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager.

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/2444//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/2444//console

This message is automatically generated.

Devaraj Kavali added a comment - 22/Nov/13 10:26

Good finding Sunil. Thanks for reporting this.

I think we need to do like these,

1. Need to handle the invalid transition and during the transition container to be removed from ContainerAllocationExpirer to avoid the timeout.
2. In the patch, trying to remove from newlyAllocatedContainers. This can be removed directly from newlyAllocatedContainers using java.util.List.remove(Object o), instead of iterating, checking and then removing.
3. Can you also add test to demonstrate this case.

Please upload the patch with these comments fix.

Sunil G added a comment - 24/Nov/13 17:41

Thank you Devaraj for reviewing the patch.

I will change and update the patch as per the comments.

Sunil G added a comment - 31/Dec/13 05:09

Updated as per comments

Sunil G added a comment - 31/Dec/13 05:09

Hi Devaraj

As per your comments, I have made the changes.

1. Need to handle the invalid transition and during the transition container to be removed from ContainerAllocationExpirer to avoid the timeout.
[Sunil]: When we remove this extra preempted container from the newlyAllocatedContainers, the invalid transition got handled. Because, when heartbeat comes, this extra container will not be there in newlyAllocatedContainers and hence ACQUIRED event will not be fired at this container.

2. In the patch, trying to remove from newlyAllocatedContainers. This can be removed directly from newlyAllocatedContainers using java.util.List.remove(Object o), instead of iterating, checking and then removing.
[Sunil]: Yes, i changed it by removing directly from the list

3. Can you also add test to demonstrate this case.
[Sunil]: Change has done to remove an element from the newlyAllocatedContainers.
There are no functions added. Now the verification is done by manual testing to ensure the removal is performed.

Sunil G added a comment - 14/Feb/14 10:11

An update on this patch:

For preemption policy, getLiveConatiners() is used to get the running containers.
But container is added to this list when it is in ALLOCATED state itself.

So in some race conditions, it is possible that a container can get KILLED by preemption even before it reach RUNNING state.
This scenario can be avoided if we can skip such containers which didnt reach the RUNNING state during preemption.
May be in the following cycles this container will reach RUNNING state and the can be considered for preemption.

devaraj.k please check this updated patch and let know your opinion.

Sunil G added a comment - 14/Feb/14 11:40

Updating patch to correct few test cases.

Hadoop QA added a comment - 04/Mar/14 07:26

+1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12629000/Yarn-1408.4.patch

against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 1 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

+1 core tests. The patch passed unit tests in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager.

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/3238//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/3238//console

This message is automatically generated.

Devaraj Kavali added a comment - 29/Apr/14 09:38

> So in some race conditions, it is possible that a container can get KILLED by preemption even before it reach RUNNING state.

This scenario can be avoided if we can skip such containers which didnt reach the RUNNING state during preemption.
May be in the following cycles this container will reach RUNNING state and the can be considered for preemption.

I think we don't need to wait for the container to move to RUNNING state for preemption even if it is eligible. If the container is eligible for preemption, the resources can be released with the current preemption cycle instead of waiting for the next preemption cycle to change the container state to RUNNING, so that it could save the wastage of the container launching and then killing.

Devaraj Kavali added a comment - 29/Apr/14 09:40

Correction to the above comment.

Sorry for the delay Sunil.

> So in some race conditions, it is possible that a container can get KILLED by preemption even before it reach RUNNING state.
> This scenario can be avoided if we can skip such containers which didnt reach the RUNNING state during preemption.
> May be in the following cycles this container will reach RUNNING state and the can be considered for preemption.

I think we don't need to wait for the container to move to RUNNING state for preemption even if it is eligible. If the container is eligible for preemption, the resources can be released with the current preemption cycle instead of waiting for the next preemption cycle to change the container state to RUNNING, so that it could save the wastage of the container launching and then killing.

Sunil G added a comment - 12/May/14 04:46

Please check the below scenario.
After allocating a container to an application, CS will decrement its associated Resource Request info.
Once this container is identified for preemption, preemption module in RM will do the container kill regardless whatever state the container is.

I am assuming that state of one container is AQUIRED [Waiting for Launch event to become RUNNING]. And now this is marked for preemption, so container will get preempted.

Hence Next heartbeat to AM has same container present in newlyAllocatedContainers and in completedContainers. [Allocation and Kill happened within an AM heartbeat cycle]
An Invalid state transition [AQUIRED at KILLED] will be happened while processing from newlyAllocatedContainers in AM side. This will cause task to timeout after 30mins.

If we try remove container from newlyAllocatedContainers, we can avoid invalid state transition. But this will cause task hang. [RM lost the resource request]

As per initial explanation, RM has allocated a container and AM is waiting to get that container to assign for a task.
Due to preemption, this has not been happened. Hence it will cause task to hang.

I feel we can preempt those containers which are only in RUNNING state. devaraj.k and curino, please share your thoughts.

Jian He added a comment - 22/May/14 20:58

Hi sunilg, agree that we should remove container from newlyAllocatedContainers when preemption happens. As per the race condition you mentioned, we may also preempt ACQUIRED container?
In fact, I think the best container to be preempted is the ALLOCATED container as these containers are not yet alive from the user's perspective. As per the race condition that [RM lost the resource request], today the resource request is decremented when container is allocated. we may change it to decrement the resource request only when the container is pulled by the AM ? We can do this separately if this makes sense.

Mayank Bansal added a comment - 22/May/14 22:39

I agree with jianhe and devaraj.k
We should be able to preempt the container in ALLOCATED state.

> oday the resource request is decremented when container is allocated. we may change it to decrement the resource request only when the container is pulled by the AM ?

I am not sure if thats the right thing as you dont want to run into other race conditions when container is been allocated however capacity is given to some other AM's?

Jian He added a comment - 22/May/14 23:31

Seems more problem with the approach I mentioned, if the request is not updated at the time container is allocated, and AM doesn't do the following allocate, more containers will be allocated from the same request when NM heartbeats

---

Sunil G added a comment - 23/May/14 06:31

> we may change it to decrement the resource request only when the container is pulled by the AM ?

As jianhe mentioned, this can create problem with subsequent NM heartbeats. Also I agree that the container in ALLOCATED state is the best place to do preemption, but this raise condition can come there.
CapacityScheduler raises KILL event for RMContainer(for preemption). So a solution may be like recreate resource request back, if the RMContainer state is ALLOCATED/ACQUIRED here.

Wangda Tan added a comment - 23/May/14 13:17

I think container should be preempt-able when its in allocate state, as race condition mentioned by jianhe, can we add a resource request to RMContainer? When allocate/kill within one AM heartbeat happened, we can add the resource request back.

Sunil G added a comment - 26/May/14 05:09

Yes. As I mentioned in earlier comment, it is better to add back/recreate the Resource Request back in CS.
Also it will be better to do from CapacityScheduler#killContainer, as it is invoked directly in the preemption context.

jianhe, mayank_bansal please share your thoughts.

Wangda Tan added a comment - 06/Jun/14 07:39

After read code and thought about this problem, I think it's better to store ResourceRequests deducted when allocate a container:

- If container transferred to ACQUIRED state, stored ResourceRequests will be removed directly.
- If container killed/preempted before transferred to ACQUIRED state, we should recover stored ResourceRequests.

If we don't store ResourceRequests, it will be hard for us to re-create exactly same ResourceRequest from a container: container maybe allocated from a off-rack ResourceRequest, or ResourceRequest set some special fields like relaxLocality.

A possible place to store ResourceRequest is SchedulerApplicationAttempt which covers Fair/CapacityScheduler together, and expose a new method like "recoverResourceRequest" and we can pass a reference of SchedulerApplicationAttempt to RMContainer. Once a container state become ACQUIRED, RMContainer can call this method to recover ResourceRequests.

sunilg, jianhe, mayank_bansal, sounds like a doable plan?

Sunil G added a comment - 06/Jun/14 09:04

> If container killed/preempted before transferred to ACQUIRED state, we should recover stored ResourceRequests.

Container count is either decremented or that entry is removed from priority based map, if the count is 0.
So once removed, a complete Resource Request has to be added back with corresponding priority.

RMContainer has priority information and it can be used to get ResourceRequest with **AppSchedulingInfo#getResourceRequests** api.
CapacityScheduler is extending PreemptableResourceScheduler interface and it has **killContainer** implementation. From this the corresponding FiCaSchedulerApp object can be found and thus can get the reference of AppSchedulingInfo.

I am thinking this way the ResourceRequest can be recreated and added back if the Container is preempted at ACQUIRED/ALLOCATED etc. **killContainer** will be invoked at the preemption event and container state also can be found here.
leftnoteasy, jianhe, mayank_bansal is this approach fine?

Sunil G added a comment - 06/Jun/14 09:29

Sorry I misunderstood, please discard my earlier comment. I feel as per your design, we are keeping a much information than needed. Rather we can just keep only requried info for a ResourceRequest creation.

> If container killed/preempted before transferred to ACQUIRED state, we should recover stored ResourceRequests.

Container count is either decremented or that entry is removed from priority based map (AppSchedulingInfo), if the count is 0.
So once removed, a complete Resource Request has to be added back with corresponding priority.

CapacityScheduler is extending PreemptableResourceScheduler interface and it has killContainer implementation. So preempt event handling can be done at **killContainer**.
I feel we can store enough information to create a new ResourceRequest at RMContainer itself. In FiCaSchedulerApp#allocate, a replica or partial information of ResourceRequest can be stored in RMContainer itself.
And api AppSchedulingInfo#updateResourceRequests can be used to update new ResourceRequest.

Wangda Tan added a comment - 06/Jun/14 10:09

Hi sunilg,

> I feel as per your design, we are keeping a much information than needed. Rather we can just keep only requried info for a ResourceRequest creation.

I meant we need keep a complete Resource Request, which includes Resource Request itself, rack/ANY level Resource Request, in case of the entire ResourceRequest removed when count reaches zero. I think we don't have difference here. 🙂, right?

> I feel we can store enough information to create a new ResourceRequest at RMContainer itself. In FiCaSchedulerApp#allocate, a replica or partial information of ResourceRequest can be stored in RMContainer itself.

+1 for this, since the resource requests is only a part of RMContainer, we don't need store it in app level

> CapacityScheduler is extending PreemptableResourceScheduler interface and it has killContainer implementation. So preempt event handling can be done at killContainer.

IMHO, it's better to put recover ResourceRequest logic in RMContainerImpl.FinishedTransition(), we can check if original state is Allocated and event is KILL. The benefit of this choice is we don't need separately modify FairScheduler and CapacityScheduler. Make sense?

> api AppSchedulingInfo#updateResourceRequests can be used to update new ResourceRequest.

Yes, but we cannot reuse this API without modifying its definition. The AppSchedulingInfo#updateResourceRequests will replace original ResourceRequests to new ResourceRequests, and our requirement is only increase the original ResourceRequest. We can either add a flag to this API indicate it's a recovering behavior or create a new API for it. And please note that, we need update QueueMetrics as well.

Thanks,
Wangda

Sunil G added a comment - 06/Jun/14 11:55

Hi leftnoteasy

> I meant we need keep a complete Resource Request, which includes Resource Request itself, rack/ANY level Resource Request, in case of the entire ResourceRequest removed when count reaches zero. I think we don't have difference here. , right?

Yes.

> pass a reference of SchedulerApplicationAttempt to RMContainer

My doubt is in this context. I felt like we may not need a SchedulerApplicationAttempt in RMContainer itself. Rather I can do like keeping required information in RMContainer. So it will be more sepcific to the handling.

> IMHO, it's better to put recover ResourceRequest logic in RMContainerImpl.FinishedTransition(), we can check if original state is Allocated and event is KILL. The benefit of this choice is we don't need separately modify FairScheduler and CapacityScheduler. Make sense?

KILL event can happen not only from Preemption. I feel we need to deal issue with repect to preemption only, correct?. Other KILL event in RMContainer may be for a reserved container KILL etc.
So we may need to do only in PreemptableResourceScheduler#killContainer. What do you feel?

> Yes, but we cannot reuse this API without modifying its definition. The AppSchedulingInfo#updateResourceRequests will replace original ResourceRequests to new ResourceRequests, and our requirement is only increase the original ResourceRequest.

I agree with the concern of modifying definition, but to an extent I think it will be better if we can reuse with a recovery mode.

> And please note that, we need update QueueMetrics as well.

+1 Yes. I agree. We have to do this also.

Thank you.
Sunil

Wangda Tan added a comment - 06/Jun/14 12:24

sunilg, thanks for reply,

> My doubt is in this context. I felt like we may not need a SchedulerApplicationAttempt in RMContainer itself. Rather I can do like keeping required information in RMContainer. So it will be more sepcific to the handling.

> KILL event can happen not only from Preemption. I feel we need to deal issue with repect to preemption only, correct?. Other KILL event in RMContainer may be for a reserved container KILL etc. So we may need to do only in PreemptableResourceScheduler#killContainer. What do you feel?

FairScheduler supports preemption as well, but it doesn't inherent PreemptableResourceScheduler interface. And do you agree that we need ALWAYS add Resource Request back when a container killed before AM acquired it (no matter this container is killed or not)? It's more like you've ordered something in Amazon, but you don't know Amazon already cancelled your order, it's not a good experience for user in this case. If you agree with me, it might be better to add logics in RMContainer transitions.

> I agree with the concern of modifying definition, but to an extent I think it will be better if we can reuse with a recovery mode.

+1 for extending and reusing existed interface.

Wangda

Sunil G added a comment - 06/Jun/14 13:02

Thank you Wangda for the comment.

> FairScheduler supports preemption as well, but it doesn't inherent PreemptableResourceScheduler interface.

Yes. This will be cause similar problem in Fair also.

I agree that we have to add back the ResourceRequest. Because as you mentioned in the example the user, which is AM here, will not get any update at all.
In summary, the containers which are yet to be pulled off by AM, but killed by RM (preemption cases) are the problem.

> If container transferred to ACQUIRED state, stored ResourceRequests will be removed directly.

> If container killed/preempted before transferred to ACQUIRED state, we should recover stored ResourceRequests.

Inline with this, if we want to recover ResourceRequest during RMContainer transitions(like KILL before ACQUIRED), we need a link back to the respective scheduler.
May be here we can try add the required object to event, which can add back a ResurceRequest to scheduler with modified AppSchedulingInfo#updateResourceRequests. Is this what you also thought?

Wangda Tan added a comment - 07/Jun/14 02:02

Thanks sunilg for your comment,

> May be here we can try add the required object to event, which can add back a ResurceRequest to scheduler with modified AppSchedulingInfo#updateResourceRequests. Is this what you also thought?

I'm not sure what's the "required object" you mentioned. I think we don't need to link with respective scheduler. Maybe a SchedulerApplicationAttempt with updated AppSchedulingInfo#updateResourceRequests should be enough, do you think so? It seems to me that we don't need touch other specific scheduler implementations to me for this task.

Wangda

Sunil G added a comment - 09/Jun/14 11:13

Thank you Wangda for the comment.
I was thinking of using **AppSchedulingInfo** object itself. A reference of this object can be placed in RMContainer.
Yes, there is a choice between **SchedulerApplicationAttempt** and **AppSchedulingInfo**. I feel based on heaviness, AppSchedulingInfo is better. AppSchedulingInfo#updateResourceRequests is synchronized also.
If any other problems are there with AppSchedulingInfo, we can use SchedulerApplicationAttempt object.

I will try summarizing the approach, please correct me if I am wrong.
Storage Part:
1. RMContainer needs to store the informations such as priority, hostname, capability and relaxLocality to create a new ResourceRequest.
2. RMContainer can also store the AppSchedulingInfo object. From this object, AppSchedulingInfo#updateResourceRequests can be invoked later to add back ResourceRequest.

Transition Handling:
1. RMContainer state transition: ALLOCATED to KILLED, a new transition named **PreemptTransition** can be invoked (extending FinishedTransition).
2. PreemptTransition can first create a new ResourceRequest instance with container count as 1 and with other details stored in RMContainer.

- Invoke AppSchedulingInfo#updateResourceRequests with this new ResourceRequest instance
- Invoke super.transition to handle the remaining FinishedTransition handling.

Wangda Tan added a comment - 09/Jun/14 15:19

sunilg,
This approach looks good to me.

Vinod Kumar Vavilapalli added a comment - 10/Jun/14 06:27

Pitching in late, apologies.

> After read code and thought about this problem, I think it's better to store ResourceRequests deducted when allocate a container:

+1 for this proposal. It will add a transitory memory foot-print. Let's make sure we benchmark this.

> 2. RMContainer can also store the AppSchedulingInfo object.

This is leaking abstractions and is a slippery slope. It is kind of weird to have container talking to ASI. How about us adding a killContainer API in scheduler (AbstractYarnScheduler) which checks the container state and adds back the RR to ASI when needed? Killing a container is a generic functionality anyways.

Sunil G added a comment - 10/Jun/14 07:05

Yes vinodkv. +1 for the killContainer api in AbstractYarnScheduler.
I have one doubt here, PreemptableResourceScheduler has killContainer api used in preemption context.
FairScheduler is using warnOrKillContainer method to invoke KILL event on container in preemption context. In this scenario, I hope we are trying to make this new killContainer api in AbstractYarnScheduler generic for these two schedulers for preemption. Could you please share your thoughts.

Vinod Kumar Vavilapalli added a comment - 10/Jun/14 07:13

> In this scenario, I hope we are trying to make this new killContainer api in AbstractYarnScheduler generic for these two schedulers for preemption

Yup, let's do that.

Wangda Tan added a comment - 10/Jun/14 07:37

vinodkv, thanks for jumping in,
+1 for put common killContainer API into AbstractYarnScheduler.
I've filed YARN-2143 to track this effort.

Sunil G added a comment - 10/Jun/14 08:37

Thank you Vinod. As discussed, I will work on a patch for storing ResourceRequest as transitory in RMContainer and restoring the same from killContainer API of AbstractYarnScheduler based on RMContainer state.

Vinod Kumar Vavilapalli added a comment - 10/Jun/14 18:35

Canceling patch for addressing review comments..

Wangda Tan added a comment - 26/Jun/14 10:30

Hi sunilg, do you have any updates on this issue, thanks 🙂

Sunil G added a comment - 26/Jun/14 10:35

Hi leftnoteasy
As discussed in YARN-2143 last day, I am now using the AbstractScheduler api only to repopulate ResourceRequest to Scheduler. I am working on a patch based on this input and will upload a patch in a day. Thank you. 🙂

Wangda Tan added a comment - 26/Jun/14 23:05

Thanks sunilg, looking forward your patch 🙂

Sunil G added a comment - 27/Jun/14 14:04

Hi vinodkv leftnoteasy
Please find initial patch.

Some information about the patch.

- While recovering ResourceRequest, if such an entry is found in Scheduling Info then the number of container is incremented. Else added as a new entry.
- Adding a new OffRackRequest also while recovering, if the stored request is not OffRack.
- AM would have asked for NodeLocal in another Hosts, which may not be able to recover.

Kindly review.

Hadoop QA added a comment - 27/Jun/14 14:11

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12652828/Yarn-1408.5.patch

against trunk revision .

-1 patch. The patch command could not apply the patch.

Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4119//console

This message is automatically generated.

Hadoop QA added a comment - 27/Jun/14 17:54

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12652860/Yarn-1408.5.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

-1 core tests. 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.ahs.TestRMApplicationHistoryWriter

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4122//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4122//console

This message is automatically generated.

Wangda Tan added a comment - 28/Jun/14 01:20

Hi sunilg,
Thanks for working out this patch so fast!

A major problems I've seen.
ResourceRequest stored in RMContainerImpl should include rack/any RR,
Currently, there's only one ResourceRequest stored in RMContainerImpl, this may should not enough for recovering in following cases:
Case 1: RR may contain other fields like relaxLocality, etc. Assume a RR is node-local, the relaxLocaity=true (default), and it's rack-local/any RR's relaxLocality=false. In your current implementation, you cannot fully recover original RRs.
Case 2: Rack-local RR will be missing. Assume a RR is node-local, when do resource allocation, the outstanding rack-local/any numContainer will be decreased, you can check AppSchedulingInfo#allocateNodeLocal for the logic of how outstanding rack/any #containers decreased.

My thoughts about how to implement this is:
In FiCaScheduler#allocate, appSchedulingInfo.allocate will be invoked. You can edit appSchedulingInfo.allocate to return a list a RRs, include node/rack/any if possible.
Pass such RRs to RMContainerImpl

And could you please elaborate on this?

> AM would have asked for NodeLocal in another Hosts, which may not be able to recover.

Does it make sense to you? I'll review minor issues and test cases in next cycle.

Thanks,
Wangda

Sunil G added a comment - 30/Jun/14 13:07

Thank you very much leftnoteasy for the comments.

> ResourceRequest stored in RMContainerImpl should include rack/any RR,

+1. Yes, RackLocal was missed and while recreating there will be problems as you mentioned (relaxLocaity).

> You can edit appSchedulingInfo.allocate to return a list a RRs

I think we can have a new api in appSchedulingInfo to return list of ResourceRequests (node local, rack local and any).

I updated a patch as per the comments, kindly check and share your opinion.

Hadoop QA added a comment - 30/Jun/14 14:26

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12653155/Yarn-1408.6.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

-1 core tests. 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.ahs.TestRMApplicationHistoryWriter

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4145//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4145//console

This message is automatically generated.

Wangda Tan added a comment - 30/Jun/14 15:34

Hi sunilg,
Thanks for updating the patch, overall approach LGTM, some comments,

1)

> I think we can have a new api in appSchedulingInfo to return list of ResourceRequests (node local, rack local and any).

I would suggest to modify existing appSchedulingInfo.allocate to return list of RRs. There existed outstanding resource decrement logic in allocate(), we can simply add decremented RR to a list and return them. It looks more like by-product of ASI.allocate to me.

2)

```
    if (type.equals(NodeType.NODE_LOCAL)) {
      list.add(nodeRequests.get(hostName));
    }
```

It's better to clone RR instead of add ref to list. It works, but it's better to set a #container correctly and prevent RR changed in ASI in the future.

3) TestCapacityScheduler:
It's good to have a test for FairScheduler here too. I think we can put the test to org.apache.hadoop.yarn.server.resourcemanager.scheduler, and make it parameterized for Fair/Capacity/FIFO.

Two minor comment for TestCapacityScheduler.
3.1

```
    for (ResourceRequest request : requests) {
      // Skip the OffRack and RackLocal resource requests.
      if (request.getResourceName().equals(node.getRackName())
```

```
          || request.getResourceName().equals(ResourceRequest.ANY)) {
        Assert.assertEquals(request.getNumContainers(), 1);
        continue;
      }

      // Resource request must have added back in RM after preempt event handling.
      Assert.assertNotNull(app.getResourceRequest(request.getPriority(),
        request.getResourceName()));
    }
```

We can make it simpler to,

```
    for (ResourceRequest request : requests) {
      // Resource request must have added back in RM after preempt event handling.
      Assert.assertEquals(1, app.getResourceRequest(request.getPriority(),
        request.getResourceName()).getNumContainers());
    }
```

Because we added them back, there's no difference between node/rack/any.

3.2

```
    // allocate container
    List<Container> containers = am1.allocate(new ArrayList<ResourceRequest>(),
        new ArrayList<ContainerId>()).getAllocatedContainers();
```

Should we wait for containers allocated in a while loop? This works now because previous we called "rm1.waitForState(nm1, ...)". But it's better to wait container allocated explictly

Thanks,
Wangda

Sunil G added a comment - 04/Jul/14 02:13

Thank you leftnoteasy for the comments.

1.

> I would suggest to modify existing appSchedulingInfo.allocate to return list of RRs.

Yes, this is better. I have modified appSchedulingInfo.allocate to get the list RRs.

2.

> make it parametrized for Fair/Capacity/FIFO.

I could see the test cases of Fair uses extending few TestBase class and working on that, not clearly uses MockRM.
So I kept a separate test case for Fair for now. Pls share your thoughts on same.

I have updated a patch and please help to review the same.

Wangda Tan added a comment - 04/Jul/14 02:52

sunilg,
Thanks update, just two minor comments,

1) Can we put RMContainerImpl.setResourceRequests to RMContainer to prevent type cast such as

```
((RMContainerImpl)rmContainer).setResourceRequests(resourceRequestList);
```

2) For TestCapacityScheduler and TestFairScheduler,
Can we verify after recovered, does RR in ASI contain {node, rack, any}?

Hadoop QA added a comment - 04/Jul/14 03:12

+1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12654025/Yarn-1408.7.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 3 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 1.3.9) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

+1 core tests. The patch passed unit tests in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager.

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4197//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4197//console

This message is automatically generated.

Sunil G added a comment - 04/Jul/14 07:43

Thank you leftnoteasy for the comments.

> Can we put RMContainerImpl.setResourceRequests to RMContainer

As vinodkv mentioned in ~~YARN-2022~~, RMContainer usually is a read only interface. So I feel we can place setResourceRequests in RMContainerImpl itself and downcast the object as needed to invoke setResourceRequests.

> Can we verify after recovered, does RR in ASI contain node, rack, any

```
    for (ResourceRequest request : requests) {
      Assert.assertEquals(1,
          app.getResourceRequest(priority, request.getResourceName())
              .getNumContainers());
    }
```

Here the loop is now running on “requests” and it has all node, rack, any requests. This can check whether container count is 1 for that specific request. So indirectly all 3 requests were verified as whether its present in “app”. Please check this and let me know whether its inline with your thought.

Thank you.

Wangda Tan added a comment - 04/Jul/14 07:47

Hi sunilg,
Thanks for your reply, this make sense to me.
This patch LGTM, +1

Could you take a look at this patch? vinodkv, mayank_bansal.
Thanks!

Jian He added a comment - 09/Jul/14 01:55

seems one race condition that if container is preempted before container moves to ACQUIRED state, but the container is already acquired by AM, the resource requests will be added back, and we'll allocate more containers than expected?

Wangda Tan added a comment - 09/Jul/14 02:33

Thanks jianhe for this commit, maybe a possible solution is move RMContainerImpl#setResourceRequest from AcquiredTransition to pullNewlyAllocatedContainer?

Sunil G added a comment - 09/Jul/14 05:04

Thanks jianhe for the comment.

```
// Remove from the list of newly allocated containers if found
    newlyAllocatedContainers.remove(rmContainer);
```

In **FicaSchedulerApp#containerCompleted**, the preempted container is removed from newlyAllocatedContainers (as in patch). So pullNewlyAllocatedContainersAndNMTokens will not get this container from newlyAllocatedContainers. So duplicate containers will not reach AM. I missed to add this to FSSchedulerApp. Please check this part and let me know if I am inline with your scenario.

Wangda Tan added a comment - 09/Jul/14 06:51

Hi sunilg
Thanks for find a new race condition 🙂
Actually, what jianhe mentioned is not this one.
It is possible that, pullNewlyAllocatedContainer happened before container's state transition to ACQUIRED because following code snippet (SchedulerAppAttempt#pullNewlyAllocatedContainersAndNMTokens):

```
1       returnContainerList.add(container);
2       i.remove();
3       rmContainer.handle(new RMContainerEvent(rmContainer.getContainerId(),
            RMContainerEventType.ACQUIRED));
```

It is possible that container killed after line-1 and before rmContainer handle ACQUIRED event. And in such case, RR will be missed even if container preempted.

Wangda Tan added a comment - 09/Jul/14 06:54

But after think about this for a while, is it really a problem? Because container will be added to returnContainerList in this case, and AM should know that, right? jianhe.

Jian He added a comment - 09/Jul/14 17:26

ah, given this event is synchronously processed, the race I mentioned should not happen, never mind.

```
rmContainer.handle(new RMContainerEvent(rmContainer.getContainerId(),
        RMContainerEventType.ACQUIRED));
```

Mayank Bansal added a comment - 10/Jul/14 06:42

Thanks sunilg for the patch.

Patch looks good , There are some minor comments
1. You current patch is not applying on the trunk, Please rebase on trunk.

2. There are lot of unwanted formatting changes, can you please revert them back. Some examples are as follows

```
-       .currentTimeMillis());
+         .currentTimeMillis());
```

```
-     RMContainer rmContainer =
-         new RMContainerImpl(container, attemptId, node.getNodeID(),
-           applications.get(attemptId.getApplicationId()).getUser(), rmContext,
-           status.getCreationTime());
+     RMContainer rmContainer = new RMContainerImpl(container, attemptId,
+         node.getNodeID(), applications.get(attemptId.getApplicationId())
+             .getUser(), rmContext, status.getCreationTime());
```

Please check this in all the patch.

Sunil G added a comment - 10/Jul/14 15:24

Thank you mayank_bansal for the review. I have updated patch against trunk and fixed formatting problems. Kindly check.

Hadoop QA added a comment - 10/Jul/14 15:45

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655007/Yarn-1408.8.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 3 new or modified test files.

-1 javac. The patch appears to cause the build to fail.

Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4259//console

This message is automatically generated.

Sunil G added a comment - 10/Jul/14 17:14

Reattaching patch again as there was a test case problem.

Hadoop QA added a comment - 10/Jul/14 18:12

+1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655031/Yarn-1408.8.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 3 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 2.0.3) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

+1 core tests. The patch passed unit tests in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager.

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4260//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4260//console

This message is automatically generated.

Wangda Tan added a comment - 11/Jul/14 08:55

Hi sunilg,
It seems your patch cannot apply against trunk, could you please rebase it?
I just reviewed your latest patch, some minor comments

1)

```
    // If container state is moved to ACQUIRED, request will be empty.
    if (requests == null || requests.isEmpty()) {
      return;
    }
```

Now request can only be null or non-empty, right?

2)
Could you update TestFairScheduler to wait for new container allocated at the end of test as same as TestCapacityScheduler?

3)
It seems ~~YARN-2181~~ changed RMContainerImpl and now TestFairScheduler#testRecoverRequestAfterPreemption will encounter a NPE. You can use "createSchedulingRequest" to avoid such error, like below

```
    ApplicationAttemptId appAttemptId =
        createSchedulingRequest(1 * 1024, "root.default", "user", 1, 1);
```

You can check TestFairScheduler#testChoiceOfPreemptedContainers as an example.

4)
Add timeout to tests you added.

Thanks,
Wangda

Sunil G added a comment - 11/Jul/14 15:46

1.

> Now request can only be null or non-empty, right?

Yes leftnoteasy. It can be only these options. I updated the check.

2. I also updated patch against trunk. Also reworked on other comments given.

Kindly check.

Hadoop QA added a comment - 11/Jul/14 16:46

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655230/Yarn-1408.9.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 2.0.3) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

-1 core tests. 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.TestWorkPreservingRMRestart
org.apache.hadoop.yarn.server.resourcemanager.TestRMRestart

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4274//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4274//console

This message is automatically generated.

Mayank Bansal added a comment - 11/Jul/14 17:57

Thanks sunilg for the patch.
Patch Looks good, Can you check these teste failures.

Thanks,
Mayank

Wangda Tan added a comment - 11/Jul/14 23:43

Thanks for update, will +1 if test failure is not relevant.

Jian He added a comment - 12/Jul/14 01:03

Thanks Sunil for the patch, some minor comments:

- add comments to recoverResourceRequest method about why doing this.
- we can use getCurrentAttemptForContainer here

```
ApplicationId appId = rmContainer.getContainerId()
    .getApplicationAttemptId().getApplicationId();
SchedulerApplication<T> schedulerApp = applications.get(appId);
SchedulerApplicationAttempt schedulerAttempt = schedulerApp
    .getCurrentAppAttempt();
```

Sunil G added a comment - 12/Jul/14 03:36

Thank you jianhe for the comments.
I have updated the patch as per your comment.

Hi mayank_bansal and leftnoteasy,
These test cases are passing locally, and seems not related to this patch. I will kick jenkins again to confirm.
Thank you.

Hadoop QA added a comment - 12/Jul/14 04:32

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655357/Yarn-1408.10.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 2.0.3) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

-1 core tests. 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.applicationsmanager.TestAMRestart

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4282//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4282//console

This message is automatically generated.

Sunil G added a comment - 12/Jul/14 05:44

Kicking jenkins to verify test case.

Hadoop QA added a comment - 12/Jul/14 11:42

+1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655374/Yarn-1408.10.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 2.0.3) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

+1 core tests. The patch passed unit tests in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager.

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4284//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4284//console

This message is automatically generated.

Wangda Tan added a comment - 13/Jul/14 15:14

LGTM, +1
Thanks,

Jian He added a comment - 14/Jul/14 06:40

More comments after looking at the latest patch:

- is it possible that schedulerAttempt here is null? e.g. preemption happens after the attempt completed.

```
    SchedulerApplicationAttempt schedulerAttempt
        = getCurrentAttemptForContainer(rmContainer.getContainerId());
    schedulerAttempt.recoverResourceRequests(requests);
```

- AbstractYarnScheduler#recoverResourceRequest, how about renaming to recoverResourceRequestForContainer ?
- assert the size of the requests. it can be empty and the assertion will be skipped. similarly for CapacityScheduler test

```
    List<ResourceRequest> requests = rmContainer.getResourceRequests();
    // Once recovered, resource request will be present again in app
    for (ResourceRequest request : requests) {
      Assert.assertEquals(1,
          app.getResourceRequest(priority, request.getResourceName())
              .getNumContainers());
    }
```

- Alternatively, calling warnOrKillContainer twice and setting WAIT_TIME_BEFORE_KILL to a small value may do the work.

```
    // Create a preempt event by sending KILL event. In real cases,
    // FairScheduler#warnOrKillContainer will perform below steps.
    ContainerStatus status = SchedulerUtils.createPreemptedContainerStatus(
        rmContainer.getContainerId(), SchedulerUtils.PREEMPTED_CONTAINER);
    scheduler.recoverResourceRequest(rmContainer);
    app.containerCompleted(rmContainer, status, RMContainerEventType.KILL);
```

Wangda Tan added a comment - 14/Jul/14 14:47

> is it possible that schedulerAttempt here is null? e.g. preemption happens after the attempt completed.

After thinking about this, we can discuss following cases
1) New attempt started after old attempt completed
Scheduler (fair/capacity/fifo) will first set old attempt is stopped, but will not remove it from existing SchedulerApplication. And recoverResourceRequests will only take effect when attempt is not stopped

```
+    if (!isStopped) {
+      appSchedulingInfo.updateResourceRequests(requests, true);
```

Then it will add a new attempt, but will use setCurrentAttempt method, will not cause getCurrentAttempt become null.
And because the old attempt completed, we shouldn't recover new resource request, so no-op is not a problem here too.

2) Application completed after old attempt completed
It is possible that SchedulerApplication become null here because old application already completed and removed from scheduler. So add a null check before

```
+    schedulerAttempt.recoverResourceRequests(requests);
```

Should be enough.

Is it make sense to you, jianhe/sunilg?

Mayank Bansal added a comment - 14/Jul/14 17:47

At jianhe 's point .
I think its good to check schedulerAttempt is not null before accessing it.
Make Sense?

Thanks,
Mayank

Sunil G added a comment - 14/Jul/14 18:41

Thank you jianhe for the comments. As leftnoteasy and mayank_bansal mentioned, I also feel that we need to have that null check to overcome any race condition with respect to application attempt.

I also updated patch with respect to other comments.

> calling warnOrKillContainer twice and setting WAIT_TIME_BEFORE_KILL

visibility of **warnOrKillContainer** is private as of now. So it needs to be changed to protected to directly invoke from test class. I updated as per this. Please let me know your thoughts

---

Hadoop QA added a comment - 14/Jul/14 19:38

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655590/Yarn-1408.11.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 2.0.3) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

-1 core tests. 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.webapp.TestRMWebServices
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesCapacitySched
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesAppsModification
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesNodes
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesApps
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesFairScheduler

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4296//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4296//console

This message is automatically generated.

Mayank Bansal added a comment - 14/Jul/14 21:58

sunilg,
Can you check these test failures?

Thanks,
Mayank

Jian He added a comment - 14/Jul/14 22:09

one minor comment while investigating test failures, visibility for AbstractYarnScheduler#recoverResourceRequestsForContainer can be protected instead of public

---

Karthik Kambatla (Inactive) added a comment - 15/Jul/14 01:55

Let me take a quick look at this as well.

Karthik Kambatla (Inactive) added a comment - 15/Jul/14 02:06

Quickly glanced through the patch. Changes seem reasonable. Given Jian, Vinod and others have already reviewed this closely, I haven't done a detailed review.

Sunil G added a comment - 15/Jul/14 08:09

Test case failures are in webapp and its due to connection bind exception.
I corrected visibility as mentioned by jianhe. Attaching patch again to re-run test cases.

Hadoop QA added a comment - 15/Jul/14 09:21

+1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655718/Yarn-1408.11.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 2.0.3) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

+1 core tests. The patch passed unit tests in hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager.

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4303//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4303//console

This message is automatically generated.

Hadoop QA added a comment - 15/Jul/14 21:43

-1 overall. Here are the results of testing the latest attachment
http://issues.apache.org/jira/secure/attachment/12655718/Yarn-1408.11.patch
against trunk revision .

+1 @author. The patch does not contain any @author tags.

+1 tests included. The patch appears to include 4 new or modified test files.

+1 javac. The applied patch does not increase the total number of javac compiler warnings.

+1 javadoc. There were no new javadoc warning messages.

+1 eclipse:eclipse. The patch built with eclipse:eclipse.

+1 findbugs. The patch does not introduce any new Findbugs (version 2.0.3) warnings.

+1 release audit. The applied patch does not increase the total number of release audit warnings.

-1 core tests. 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.webapp.TestRMWebServices
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesCapacitySched
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesAppsModification
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesNodes
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesApps
org.apache.hadoop.yarn.server.resourcemanager.webapp.TestRMWebServicesFairScheduler

+1 contrib tests. The patch passed contrib unit tests.

Test results: https://builds.apache.org/job/PreCommit-YARN-Build/4312//testReport/
Console output: https://builds.apache.org/job/PreCommit-YARN-Build/4312//console

This message is automatically generated.

Mayank Bansal added a comment - 15/Jul/14 21:43

+1 Committing

Thanks sunilg for the patch.
Thanks jianhe, vinodkv and wangda for the reviews.

Thanks,
Mayank

Hudson added a comment - 15/Jul/14 21:53

FAILURE: Integrated in Hadoop-trunk-Commit #5887 (See https://builds.apache.org/job/Hadoop-trunk-Commit/5887/)
~~YARN-1408~~ Preemption caused Invalid State Event: ACQUIRED at KILLED and caused a task timeout for 30mins. (Sunil G via mayank) (mayank: http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1610860)

- /hadoop/common/trunk/hadoop-yarn-project/CHANGES.txt
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainer.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AppSchedulingInfo.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerApplicationAttempt.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/fica/FiCaSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FSSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairScheduler.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/TestRMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairSchedulerTestBase.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/TestFairScheduler.java

---

Mayank Bansal added a comment - 15/Jul/14 23:46

Committed to trunk, branch 2 and branch-2.5.

branch-2.5 needed some rebase , Updating the patch for branch-2.5

Thanks,
Mayank

Mayank Bansal added a comment - 15/Jul/14 23:47

rebasing against branch 2.5

Hudson added a comment - 16/Jul/14 10:37

FAILURE: Integrated in Hadoop-Yarn-trunk #614 (See https://builds.apache.org/job/Hadoop-Yarn-trunk/614/)
~~YARN-1408~~ Preemption caused Invalid State Event: ACQUIRED at KILLED and caused a task timeout for 30mins. (Sunil G via mayank) (mayank: http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1610860)

- /hadoop/common/trunk/hadoop-yarn-project/CHANGES.txt
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainer.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AppSchedulingInfo.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerApplicationAttempt.java

- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/fica/FiCaSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FSSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairScheduler.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/TestRMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairSchedulerTestBase.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/TestFairScheduler.java

---

Hudson added a comment - 16/Jul/14 13:31

FAILURE: Integrated in Hadoop-Mapreduce-trunk #1833 (See https://builds.apache.org/job/Hadoop-Mapreduce-trunk/1833/)
~~YARN-1408~~ Preemption caused Invalid State Event: ACQUIRED at KILLED and caused a task timeout for 30mins. (Sunil G via mayank) (mayank: http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1610860)

- /hadoop/common/trunk/hadoop-yarn-project/CHANGES.txt
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainer.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AppSchedulingInfo.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerApplicationAttempt.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/fica/FiCaSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FSSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairScheduler.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/TestRMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairSchedulerTestBase.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/TestFairScheduler.java

---

Hudson added a comment - 16/Jul/14 13:50

FAILURE: Integrated in Hadoop-Hdfs-trunk #1806 (See https://builds.apache.org/job/Hadoop-Hdfs-trunk/1806/)
~~YARN-1408~~ Preemption caused Invalid State Event: ACQUIRED at KILLED and caused a task timeout for 30mins. (Sunil G via mayank) (mayank: http://svn.apache.org/viewcvs.cgi/?root=Apache-SVN&view=rev&rev=1610860)

- /hadoop/common/trunk/hadoop-yarn-project/CHANGES.txt
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainer.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/RMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/AppSchedulingInfo.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/SchedulerApplicationAttempt.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/common/fica/FiCaSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FSSchedulerApp.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/main/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairScheduler.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/rmcontainer/TestRMContainerImpl.java
- /hadoop/common/trunk/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/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/FairSchedulerTestBase.java
- /hadoop/common/trunk/hadoop-yarn-project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-resourcemanager/src/test/java/org/apache/hadoop/yarn/server/resourcemanager/scheduler/fair/TestFairScheduler.java

People

Assignee:

Sunil G

Reporter:

Sunil G

Votes:

0 Vote for this issue

Watchers:

13 Start watching this issue

Dates

Created:

13/Nov/13 06:11

Updated:

15/Aug/14 05:44

Resolved:

15/Jul/14 23:47