Intra Queue Preemption Use Cases

prepared by Sunil G with inputs from Eric Payne and Wangda Tan

Contents

- Overview to Intra-Queue preemption
- Scenarios with priority alone
- Scenarios with user-limit alone
- Scenarios with priority and user-limit

Overview

Intra Queue preemption helps to solve the large imbalances within a queue because of user-limit and priority. This feature works along with existing inter-queue preemption.

There were earlier some ideas of using two configurations for user-limit and priority separately. Currently we are looking for single configuration to enable < **Fifo + user-limit + priority** > based intra queue preemption as single feature. This document will try to share some use cases when all these parameters are used separately and together.

Once we capture all possible scenarios, we will try to revisit the need of second configuration for user-limit.

Priority Alone (Single User TestCases)

Scenario 1 - Simple Priority Preemption

Input:

app1 , p1, u1 <pending=20 , used=50 >

app2 , p1, u1 <pending=20 , used= 20>

app3 , p3, u1 <pending=30 , used= 0>

Preempted:

app1 , p1, u1 <preempted=11, pending=31 , used=39>

app2 , p1, u1 <preempted=19, pending=39 , used=1>

app3 , p3, u1 <pending=0 , used=30>

Configuration:

yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

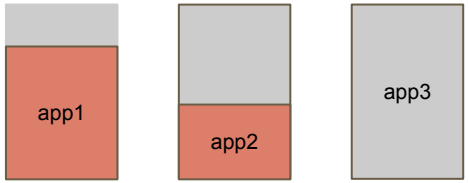
root.qA.capacity = 70% , root.qB.capacity = 30%

Cluster resource = 100 (qA.used=70, qB.used=30)

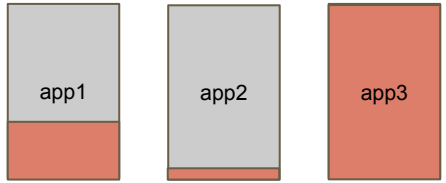
Analysis:

30 resources were preempted from app1 and app2 because of the demand from app3 with p3 demand.

App2's AM container got spared.



Before



After

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This is intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 2 - No Preemption for same priority apps

Input:

app1 , p1, u1 <pending=0 , used=50 >

app2 , p1, u1 <pending=20 , used=20>

Preempted:

app1 , p1, u1 <preempted=0, pending=0 , used=50>

app2 , p1, u1 <preempted=0, pending=20 , used=20>

Configuration:

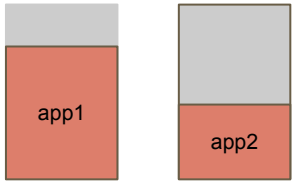
yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

root.qA.capacity = 70% , root.qB.capacity = 30%

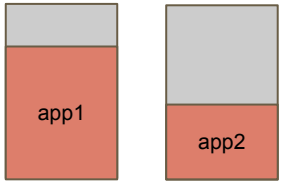
Cluster resource = 100 (qA.used=70, qB.used=30)

Analysis:

Since priority demand is from apps which are of same priority (p1), there won't be any preemption here.



Before



After

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This is intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 3 - Limit preemption based on cap-quota

Input:

app1 , p1, u1 <pending=20 , used=50 >

app2 , p1, u1 <pending=20 , used= 20>

app3 , p3, u1 <pending=80 , used= 0>

Preempted:

app1 , p1, u1 <preempted=16, pending=36 , used=34>

app2 , p1, u1 <preempted=19, pending=39 , used=1>

app3 , p3, u1 <preempted=0, pending=45 , used=35>

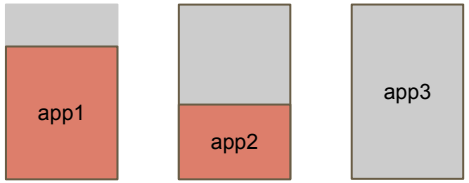
Configuration:

yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

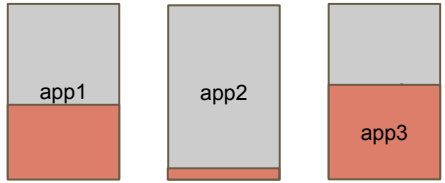
Yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.max-allowable-limit = 0.5

root.qA.capacity = 70% , root.qB.capacity = 30%

Cluster resource = 100 (qA.used=70, qB.used=30)



Before



After

Analysis:

Only 35 resources (50%) from queueA are preempted from app1 and app2. App2's AM container got spared.

On same priority level app2 will be selected first for preemption, based on submission order.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

User-Limit Alone (Same App priority)

Scenario 4 - Simple User-limit Preemption

Input:

app1 , p1, u1 <pending=20 , used=25 >

app2 , p1, u2 <pending=20 , used= 25>

app3 , p1, u3 <pending=30 , used= 50>

Preempted:

app1 , p1, u1 <preempted=0, pending=12 , used=33>

app2 , p1, u2 <preempted=0, pending=12 , used=33>

app3 , p1, u3 <preempted=16, pending=46 , used=34>

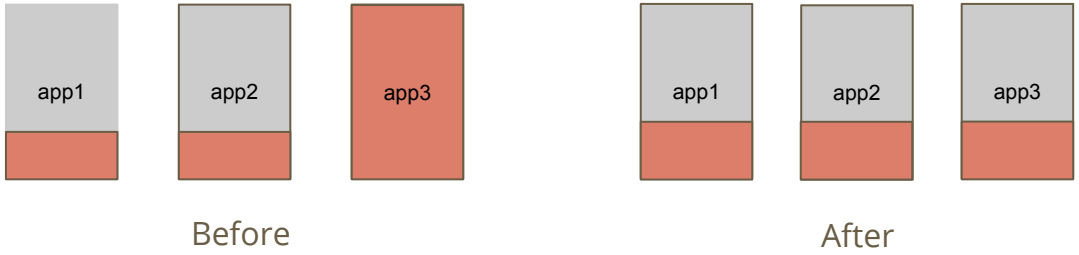
Configuration:

yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

root.qA.capacity = 100%

minimum-user-limit-percent=33%

Cluster resource = 100 (qA.used=100)



Analysis:

16 resources were preempted from app3 who was above user-limit.

These resources will be shared between app1 and app2.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 5 - No Userlimit Preemption(all under quota)

Input:

app1 , p1, u1 <pending=20 , used=50 >

app2 , p1, u2 <pending=20 , used= 50>

Preempted:

app1 , p1, u1 <preempted=0, pending=20 , used=50>

app2 , p1, u2 <preempted=0, pending=20 , used=50>

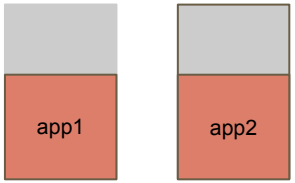
Configuration:

yarn.resourcemanager.monitor.capacity.preemption .intra-queue-preemption.enabled = true

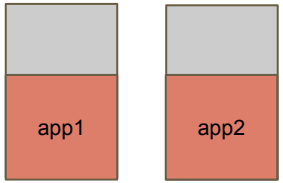
minimum-user-limit-percent=50%

root.qA.capacity = 100%

Cluster resource = 100



Before



After

Analysis:

No resources will be preempted as all users under its minimum-user-limit quota.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 6 - No Userlimit Preemption(all above quota)

Input:

app1 , p1, u1 <pending=20 , used=40 >

app2 , p1, u2 <pending=20 , used= 40>

app3 , p1, u3 <pending=30 , used= 40>

Preempted:

app1 , p1, u1 <preempted=0, pending=20 , used=40>

app2 , p1, u2 <preempted=0, pending=20 , used=40>

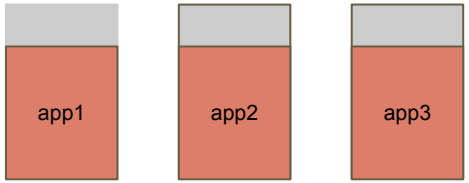
app3 , p1, u3 <preempted=0, pending=30 , used=40>

Configuration:

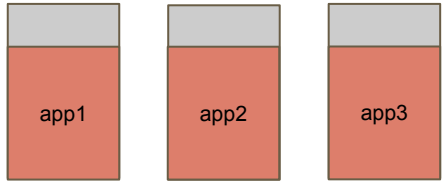
yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

minimum-user-limit-percent=33% (queueA)

Cluster resource = 150 (queueA=100, queueB=50) , queueA is over-utilizing 20 more resources



Before



After

Analysis:

No resources will be preempted as all users are above quota.

So even if there are demands, there will not be any preemption.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

User-Limit & App priority together

Scenario 7 - Preemption stops once user-limit is met (Case 1)

Input:

app1 , p1, u1 <pending=50 , used=100>

app2 , p2, u2 <pending=150 , used=0>

Preempted:

app1 , p1, u1 <preempted=50, pending=100, used=50>

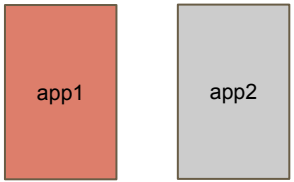
app2 , p2, u2 <preempted=0, pending=100 , used=50>

Configuration:

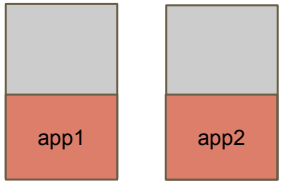
yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

minimum-user-limit-percent=25% (queueA)

Cluster resource = 100 (queueA=100)



Before



After

Analysis:

Once resources are shared as per user-limit quota (50% each), there wont be any more preemption.

App2 still has more demand and it has high priority. Still user-limit will be checked first.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 8 - Preemption stops once user-limit is met (Case 2)

Input:

app1 , p1, u1 <pending=20 , used=0 >

app2 , p1, u1 <pending=20 , used= 50>

app3 , p3, u2 <pending=30 , used= 40>

Preempted:

app1 , p1, u1 <preempted=0, pending=20, used=0>

app2 , p1, u1 <preempted=0, pending=20 , used=50>

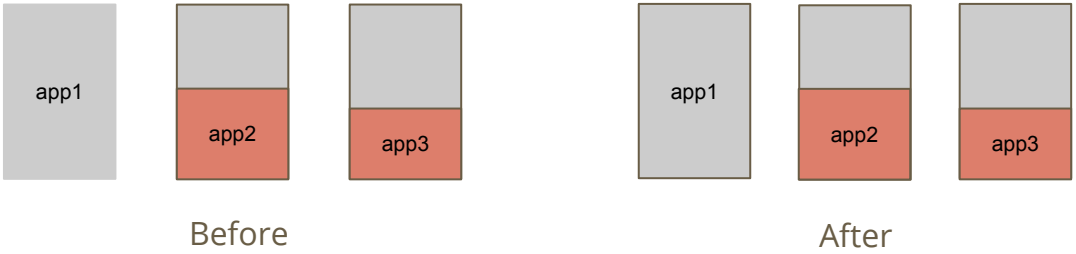
app3 , p3, u2 <preempted=0, pending=30 , used=40>

Configuration:

yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

minimum-user-limit-percent=50% (queueA)

Cluster resource = 100 (queueA=100)



Analysis:

No resources will be preempted as user2 is under its quota and user1 is not above-its quota.

So even if there are demands, there will not be any preemption.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 9 - Preemption happens due to user-limit NOT priority

Input:

app1 , p1, u1 <pending=20 , used=100 >

app2 , p2, u2 <pending=100 , used= 0>

Preempted:

app1 , p1, u1 <preempted=50, pending=70, used=50>

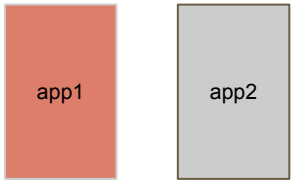
app2 , p2, u2 <preempted=0, pending=50 , used=50>

Configuration:

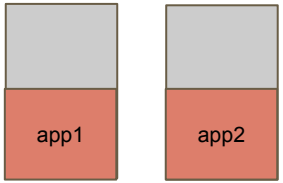
yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

minimum-user-limit-percent=50% (queueA)

Cluster resource = 100 (queueA=100)



Before



After

Analysis:

Resources are preempted here due to user-limit quota. Once quota is reached (50%), preemption stops eventhough high priority app2 has more demand.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 10 - Preemption due to user-limit and priority together

Input:

app1 , p1, u1 <pending=20 , used=20 >

app2 , p2, u1 <pending=70 , used= 20>

app3 , p3, u2 <pending=40 , used= 60>

Preempted:

app1 , p1, u1 <preempted=19, pending=39, used=1>

app2 , p2, u1 <preempted=0, pending=41 , used=49>

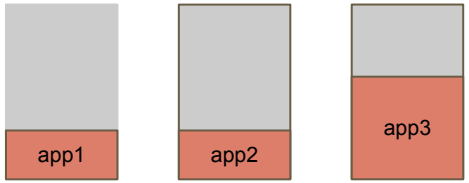
app3 , p3, u2 <preempted=10, pending=50 , used=50>

Configuration:

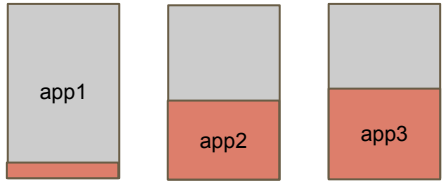
yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

minimum-user-limit-percent=50% (queueA)

Cluster resource = 100 (queueA=100)



Before



After

Analysis:

User-limit: user2 is over-utilizing 10 resources. As per user-limit, this will be preempted for user1's demand

Priority: For user1, app2 is of high priority. Hence kill app1's 19 container sparing AM.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 11 - Preemption due to user-limit and priority together

Input:

app1 , p1, u1 <pending=20 , used=20 >

app2 , p2, u1 <pending=20 , used= 20>

app3 , p3, u2 <pending=40 , used= 60>

Preempted:

app1 , p1, u1 <preempted=10, pending=30, used=10>

app2 , p2, u1 <preempted=0, pending=0 , used=40>

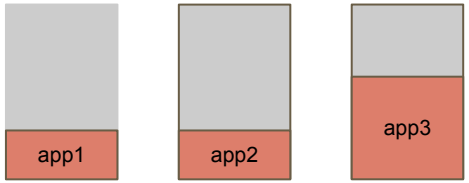
app3 , p3, u2 <preempted=10, pending=50 , used=50>

Configuration:

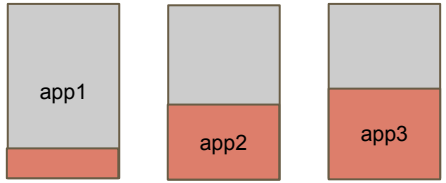
yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

minimum-user-limit-percent=50% (queueA)

Cluster resource = 100 (queueA=100)



Before



After

Analysis:

User-limit: user2 is over-utilizing 10 resources. As per user-limit, this will be preempted for user1's demand

Priority: For user1, app2 is of high priority. Hence kill app1's 10 container.

Issue: app3 of p3 priority lost containers. This is however acceptable.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 12 - Preemption within over-utilized user & priority

Input:

app1 , p1, u1 <pending=20 , used=30 >

app2 , p2, u1 <pending=20 , used= 35>

app3 , p3, u2 <pending=40 , used= 35>

Preempted:

app1 , p1, u1 <preempted=29, pending=49, used=1>

app2 , p2, u1 <preempted=0, pending=6 , used=49>

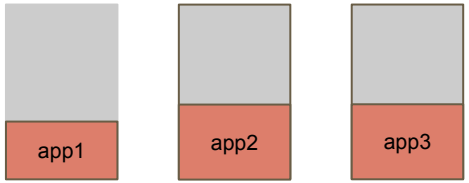
app3 , p3, u2 <preempted=0, pending=25 , used=50>

Configuration:

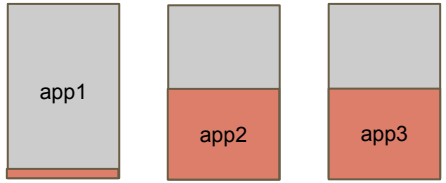
yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

minimum-user-limit-percent=50% (queueA)

Cluster resource = 100 (queueA=100)



Before



After

Analysis:

User-limit: user1 is over-utilizing 15 resources. As per user-limit, select 15 from low priority app of user1.

Priority: For user1, app2 is of high priority. Hence to meet app1's demand, preempt 14 from app1 sparing AM.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used

Scenario 13 - Preemption within under-utilized user & priority

Input:

app1 , p1, u1 <pending=15 , used=5 >

app2 , p2, u1 <pending=20 , used= 10>

app3 , p3, u2 <pending=0 , used= 15>

Preempted:

app1 , p1, u1 <preempted=4, pending=19, used=1>

app2 , p2, u1 <preempted=0, pending=16 , used=14>

app3 , p3, u2 <preempted=0, pending=0 , used=15>

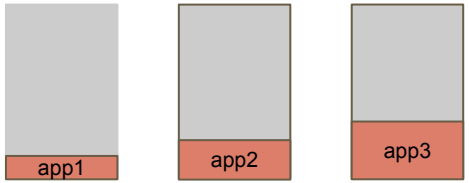
Configuration:

yarn.resourcemanager.monitor.capacity.preemption.intra-queue-preemption.enabled = true

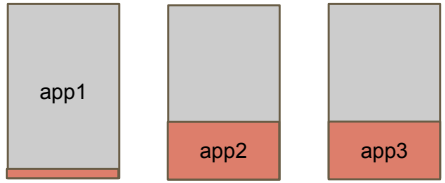
root.qA.capacity = 30% , root.qB.capacity = 70%

Cluster resource = 100 (qA.used=30, qB.used=70)

Analysis:



Before



After

Priority: For user1, app2 is of high priority. Hence here we will meet only app2's demand. User-limit won't come into play here.

Note 1: "pX" X will be considered as integer and in app priority cases, higher integer means higher priority. So p2 > p1.

Note 2: This intra queue scenario, hence we will not be explaining cross queues.

Pending

Used