Intra-queue Fair-share Preemption Policy

Requirements:

1. In intra-queue preemption, we want to support preemption based on FairOrderPolicy along with FifoOrderPolicy. This is to ensure that no job gets starved and all jobs are able to get equal resources.
2. All users in a queue should be getting equal resources.
3. All apps for a user should be getting equal resources.
4. Users should maintain their user-limits after preemption.

Terminology:

Term	**Formula**	**Description**
Fair share per app	$\frac{Queue\ Resources}{number\ of\ apps}$	FairShare across all the apps in the queue
Fair share per app with UL	$\frac{UserLimit}{number\ of\ apps\ by\ that\ user}$	FairShare across all the apps of a user is the Userlimit for that user distributed evenly across all apps of that user.
User-Limit	$\max\{\frac{total\ queue\ resources}{number\ of\ users}, user\text{-}limit\text{-}percent\}$	The max percent of resources a user can use.

Design:

For fair share preemption, we check for following 2 conditions:

1. After preemption, users consume upto their userLimit, if there are pending requests from other users in the queue.
2. After preemption, an app (whose container is being preempted) doesn't drop below its fair-share.

Fair-share is calculated for each app of a user.
For an app, fair share is calculated as follows:

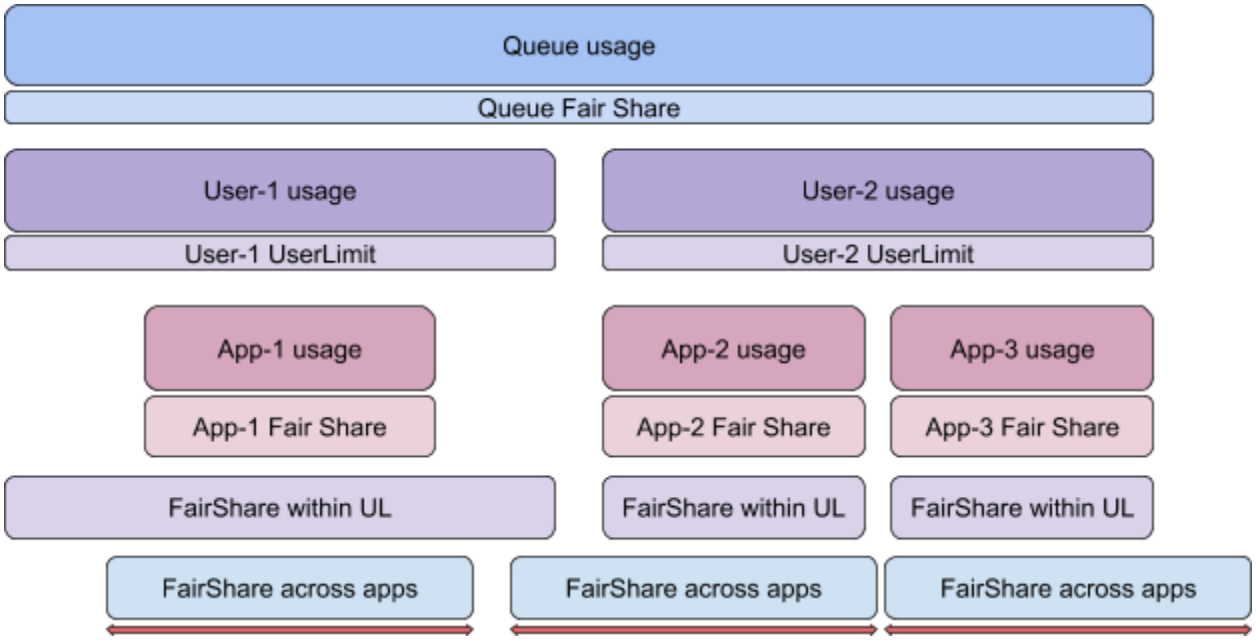
fairSharePerApp = total Queue Cap / no: of apps

idealFairSharePerAppwithUL = UL / no: of apps of that user

if(fairSharePerApp * apps_of_user > UL)

fairSharePerApp = idealFairSharePerAppwithUL;

Following image illustrates fair-share assignment for users and apps.



As part of COMPX-4883, we are making following changes inside FifoIntraQueuePreemptionPlugin when FairOrderingPolicy is enabled:

1. In computeAppsIdealAllocation():
    a. we calculate the FairShare of each app of a user.
2. In skipContainersBasedOnIntraQueuePolicy(), we check that:
    a. We don't drop an app below its fair-share after preempting its container.

Taking complexity and optimality into account, the current algorithm seems to be preempting resources along the same logic as fairOrder scheduling. Hence, it has been picked.

Other Algorithms considered:

Algorithm-1

FairShare per app = queueResources / num of apps in queue

Fails to put any check on userLimit constraints.

Eg:

User1(UL = 50%)

-App1(FS = 33%, used = 50%)

-App2(FS = 33%, used = 50%)

User2(UL = 50%)

-App3(FS = 33%, used = 0%)

User1 stays beyond its UL of 50%. (33% + 33%)

Algorithm-2

FairShare per app = UserLimit / num of apps by user

Fails in cases where sum of all userLimits across users is over 100%.

Eg:

User1(UL = 100%)

-App1(FS = 50%, used = 50%)

-App2(FS = 50%, used = 50%)

User2(UL = 100%)

-App3(FS = 50%, used = 00%)

User2 will never be able to preempt resources from user1. (used == FS)

Algorithm-3

fairSharePerApp = total Queue Cap / no: of apps

idealFairSharePerAppwithUL = UL / no: of apps of that user

fairSharePerUser = total Queue Cap / # of users

if(fairSharePerUser >= UL)

fairSharePerApp = idealFairSharePerAppwithUL;

Fails in cases where userLimits is not of the form (100 / n), n being a natural number.

Eg:

User1(UL = 55%, FS per user = 50%)

-App1(FS = 33%, used = 28%)

-App2(FS = 33%, used = 28%)

User2(UL = 55%, FS per user = 50%)

-App3(FS = 33%, used = 46%)

In this case, we will preempt resources from app3 to give to app1 and app2. However, the scheduler will see User1 is above its UL. So scheduler will reassign the preempted resources to app3. This will create an infinite preemption cycle.