capacity schedule

队列A

队列B

队列C

1 多队列 每个队列可以分配到一定的资源，每个队列采用FIFO策略执行作业运行

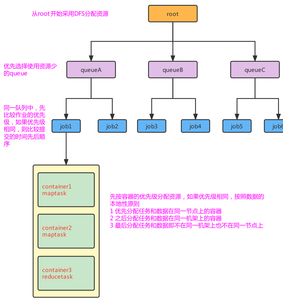
2 容量保证 每个队列都有最低的运行资源保证和资源运行上限

3 灵活性 当一个队含有空闲资源时，A需资源运行job，可以给占B的资源，但是，当B有新的job需要提交时，A必须归还给占B的资源

4 安全性 对job的大小进行限定，方式独占该队列的资源。

capacity schedule 调度算法

采用抢占式，必定带着优先级



fair schedule

队列A 0.2的资源，每个job占资源的0.5

队列B，0.5的资源，每个job占资源的20%

队列C，0.3的资源，每个job占资源的1/3

fair schedule 与capacity schedule的不同

1 容器调度器优先选择资源使用少的队列

2 公平调度器优先选择缺额大的队列

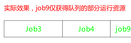
缺额的概念

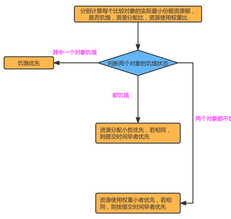
job9加入队列

理想效果，job9获得全部运行的资源

缺额

实际效果，job9仅获得队列的部分运行资源

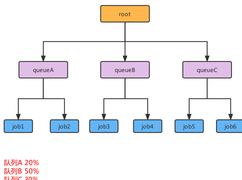




举例，将作业，容器抽象为一个对象

```
情况1二者有一个饥饿
 假设在某个时刻 对象1的资源需求量4，配置的最小资源2，资源的使用量为1，对象2的资源使用量为2，其余参数与对象1相同

按如下步骤计算

实际最小资源份额：minshare=min（资源需求量，配置的最小资源）
是否饥饿：bool isneedy=资源使用量<minshare
资源分配比:minshareRatio=资源使用量/max(minshare,1)
资源使用权重比:useToweightRaatio=资源使用量/权重

先判断对象1是否饥饿
minshare=min4,2)=2
bool isneedy=1<minshare=1<2=true
对象1饥饿

在判断对象2是否饥饿
minshare=min(4,2)=2
bool isneedy=2<minshare=2<2=false
对象2不饥饿

对象1先被分配资源

情况2 二者都不饥饿

假设在某个时刻，对象1，对象2的资源需求量4，配置的最小资源2，资源的使用量为1为2，w1=0.2,w2=0.5

按如下步骤计算
实际最小资源份额：minshare=min(4,2)
bool isneedy=2<2=false
对象1，对象2都不饥饿
开始计算资源使用权重比
r1=2/0.2=10
r2=2/0.5=4
权重比小的对象先被分配资源
r2<r1对象2先被分配资源

情况三 二者都饥饿
假设在某个时刻，对象1，对象2的资源需求量4，对象1配置的最小资源2，对象2配置的最小资源3，对象1的资源使用量1，对象2的资源使用量1

先计算对象1是否饥饿：
实际最小资源份额 minshare=min(4，2)=2
是否饥饿 isneedy=1<minshare=true
再判断对象2是否饥饿
minshare=min(4,3)=3
isneedy=1<3=true

此时对象1对象2都饥饿，开始计算资源分配比
对象1=1/max(2,1)=0.5
对象2=1/max(3,1)=1/3=0.333
对象2的资源分配比少，为了公平，分配给对象2资源
```

fair schedule 调度算法



队列A 20%
队列B 50%
队列C 30%

1 队列的资源分配

A需求20B需求50C需求30

第一次分配：绝对公平 100/3=33.33
queueA ： 分33.33 ------>多13.33
queueB ：分33.33------>分16.67
queueC ： 分33.33------>多3.33

第二次分配：绝对公平
(13.33+3.33)/1=16.66
queueA get 33.33-13.33=20
queueB get 33.33+16.66
queueC get 33.33-3.33=30

2 作业的资源分配
情况1不加权关注job的个数
需求：有一个queue的资源有12个，有4个job，对资源的需求分别是：
job1->1 job2->2 job3->6 job4->5

第一次算：12/4=3
job1 :3 多2
job2:3 多1
job3:3 差3
job4:3 差2

第二次算： （2+1）/2=1.5
job1:1
job2 :2
job3: 3+1.5=4.5
job4: 3+1.5=4.5
此时queue中已经没有可用资源，分配完毕
只能等job1,job2运行完成时再分配。

2 作业的资源分配
情况1加权关注job的权重
需求：有一个queue的资源有16个，有4个job，对资源的需求量分别是：
job1 4 job2 2 job3 10 job4 4
w1 5 w2 8 w3 1 w4 2
第一次分配 16/(5+8+1+2)=1
job1 5 多1
job2 8 多6
job3 1 少9
job4 2 少2
第二次分配：先回收分配多的资源1+6 ;7/(1+2)=2.33
job1 4
job2 2
job3 1+2.33=3.33 少6.67
job4 2+2.33*2=6.66 多2.66
第三次分配：先回收分配多的资源2.66
2.66/1=2.66

job1 4
job2 2
job3 3.33+2.66*1=6 少4
job4=4
此时queue已经没有可用资源，分配完毕，只能等其他job运行完再申请队列资源