YARN ResourceManager 调度器的分析

1. 前言

1.1. 文档说明

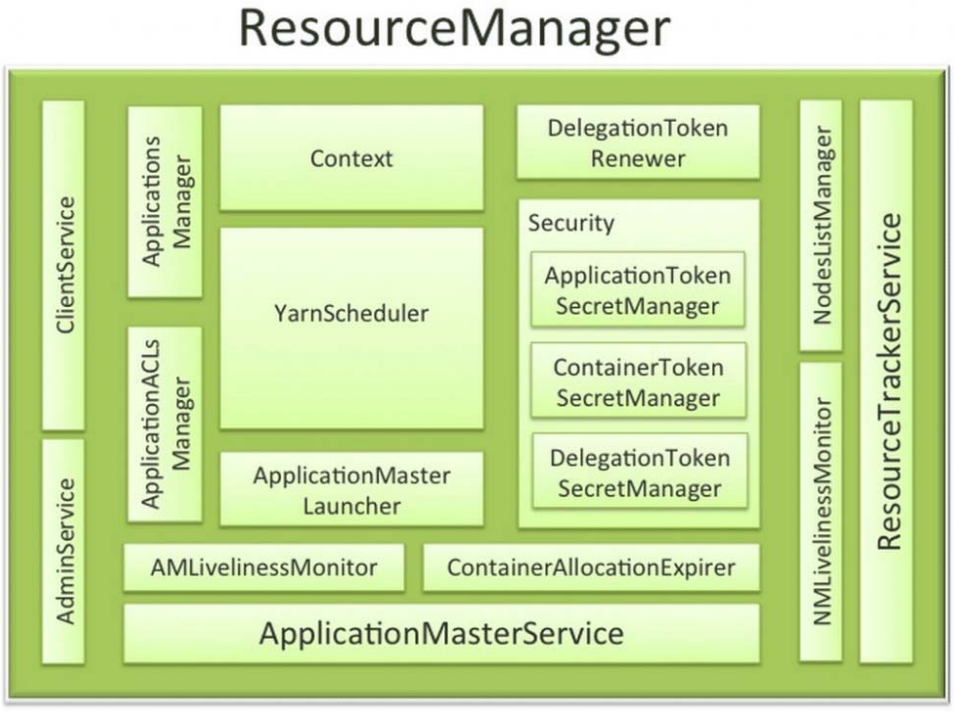
YARN 是 Hadoop 新版中的资源控制框架。本文旨在深入剖析 ResourceManager 的调度器，探讨三种调度器的设计侧重，最后给出一些配置建议和参数解释。

1.2. 名词解释

ResourceManager：以下简称 RM。YARN 的中控模块，负责统一规划资源的使用。
NodeManager：以下简称 NM。YARN 的资源结点模块，负责启动管理 container。
ApplicationMaster：以下简称 AM。YARN 中每个应用都会启动一个 AM，负责向 RM 申请资源，请求 NM 启动 container，并告诉 container 做什么事情。
Container：资源容器。YARN 中所有的应用都是在 container 之上运行的。AM 也是在 container 上运行的，不过 AM 的 container 是 RM 申请的。

2. RM 中的调度器

ResourceManager 是 YARN 资源控制框架的中心模块，负责集群中所有的资源的统一管理和分配。它接收来自 NM 的汇报，建立 AM，并将资源派送给 AM。



最初的 hadoop 版本只有 FifoScheduler（先进先出调度器）。当 Hadoop 集群在大规模使用的时候，如何整合资源和分配资源就是一个迫切的需求。对此，Yahoo 和 facebook 先后开发了 CapacityScheduler（容量调度器）和 FairScheduler（公平调度器）。在新版本中，这两个调度器在保持核心算法的基础上，也被重新开发了一次。

3. 调度器的接口

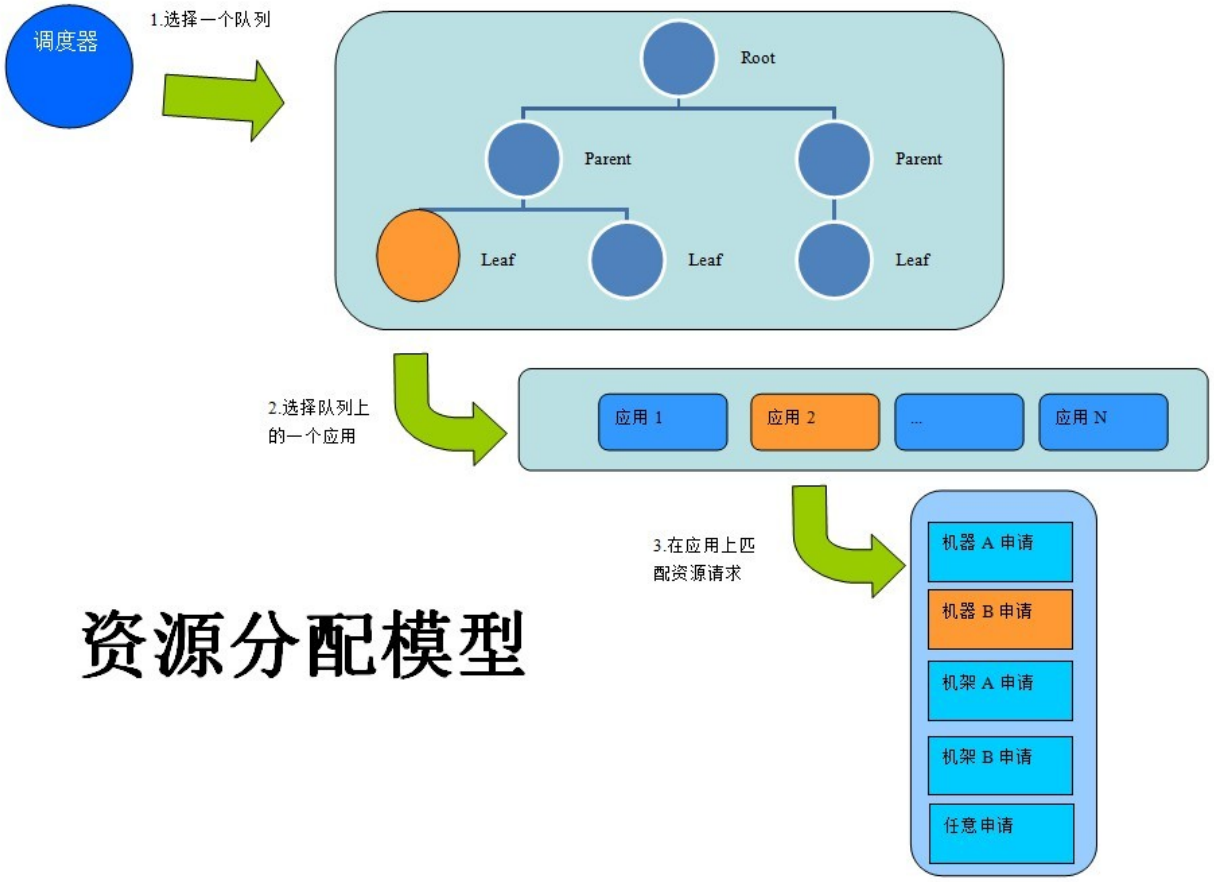
一个完整的调度器在内存中都会维护一个队列、应用、NM、Container 的关系。同时，一个调度器也是一个事件处理器，通过 RM 的异步事件调用机制知晓外部发生的事情。要跟外部交互也要发送相应的事件。调度器一共要处理 6 个调度事件。

事件	发送时机	处理逻辑
NODE_ADDED	一个 NM 被添加	增加总资源池的大小，修改内存状态。
NODE_REMOVED	一个 NM 被移除	删除一个 NM,减少总资源池的大小，回收内存状态中在这个 NM 上的 Container，对每个 container 发送 KILL 事件。
NODE_UPDATE	一个 NM 跟 RM 进行心跳	调度器会根据当前的 NM 状况，在这个 NM 上为某一个 AM 分配 Container，并记录这个 Container 的信息，留待 AM 获取。这个部分是调度器真正分配 container 的部分。后面会重点描述。
APP_ADDED	一个新的应用被提交	如果接受应用，发送 APP_ACCEPTED 事件，否则发送 APP_REJECTED 事件。
APP_REMOVED	一个应用移除	可能是正常或者被杀死。清除内存中这个应用的所有的 container，对每个 container 发送 KILL 事件。
CONTAINER_EXPIRED	一个 container 过期未被使用	修改内存中这个 contianer 相关的内存状态。

AM 会告诉调度器一些资源申请的请求和已经使用完的 container 的列表，然后获取到已经在 NODE_UPDATE 分配给这个应用的 container 的分配。可以看到调度器接受资源的申请和分配资源这个动作是异步的。

4. 资源分配模型

无论 FifoScheduler，CapacityScheduler 和 FairScheduler 的核心资源分配模型都是一样的。调度器维护一群队列的信息，用户可以向一个或者多个队列提交应用。每次 NM 心跳的时候，调度器根据一定的规则选择一个队列，再在队列上选择一个应用，尝试在这个应用上分配资源。不过，因为一些参数限制了分配失败，就会继续选择下一个应用。在选择了一个应用之后，这个应用对应也会有很多资源申请的请求。调度器会优先匹配本地资源的申请请求，其次是同机架的，最后的任意机器的。



资源分配模型

总的来说，3 种调度器就是在回答如何选择一个队列，在一个队列上如何选择一个应用的问题。

5. 调度器比较

调度器	FifoScheduler	CapacityScheduler	FairScheduler
设计目的	最简单的调度器，易于理解和上手	多用户的情况下，最大化集群的吞吐和利用率	多用户的情况下，强调用户公平地贡献资源

调度器	FifoScheduler	CapacityScheduler	FairScheduler
队列组织方式	单队列	树状组织队列。无论父队列还是子队列都会有资源参数限制，子队列的资源限制计算是基于父队列的。应用提交到叶子队列。	树状组织队列。但是父队列和子队列没有参数继承关系。父队列的资源限制对子队列没有影响。应用提交到叶子队列。
资源限制	无	父子队列之间有容量关系。每个队列限制了资源使用量，全局最大资源使用量，最大活跃应用数量等。	每个叶子队列有最小共享量，最大资源量和最大活跃应用数量。用户有最大活跃应用数量的全局配置。
队列 ACL 限制	可以限制应用提交权限	可以限制应用提交权限和队列开关权限，父子队列间的 ACL 会继承。	可以限制应用提交权限，父子队列间的 ACL 会继承。但是由于支持客户端动态创建队列，需要限制默认队列的应用数量。目前，还看不到关闭动态创建队列的选项。
队列排序算法	无	按照队列的资源使用量最小的优先	根据公平排序算法排序
应用选择算法	先进先出	先进先出	先进先出或者公平排序算法
本地优先分配	支持	支持	支持
延迟调度	不支持	不支持	支持
资源抢占	不支持	不支持	支持，看到代码中也有实现。但是，由于本特性还在开发阶段，本文没有真实试验。

FifoScheduler：最简单的调度器，按照先进先出的方式处理应用。只有一个队列可提交应用，所有用户提交到这个队列。可以针对这个队列设置 ACL。没有应用优先级可以配置。
CapacityScheduler：可以看作是 FifoScheduler 的多队列版本。每个队列可以限制资源使用量。但是，队列间的资源分配以使用量作排列依据，使得容量小的队列有竞争优势。集群整体吞吐较大。延迟调度机制使得应用可以放弃，夸机器或者夸机架的调度机会，争取本地调度。
FairScheduler：多队列，多用户共享资源。特有的客户端创建队列的特性，使得权限控制不太完美。根据队列设定的最小共享量或者权重等参数，按比例共享资源。延迟调度机制跟 CapacityScheduler 的目的类似，但是实现方式稍有不同。资源抢占特性，是指调度器

能够依据公平资源共享算法，计算每个队列应得的资源，将超额资源的队列的部分容器释放掉的特性。

6. 参数配置

要彻底理解各个参数配置中关系，就不能忽略掉调度器的调度算法和调度细节。

6.1. 调度器的集群配置

以下是和调度器密切相关的集群配置。

配置文件	配置项	值	含义
mapred-site.xml	yarn.app.mapreduce.am.resource.mb	1024	ApplicationMaster 的 container 占用的内存大小
	mapreduce.map.memory.mb mapreduce.reduce.memory.mb	512/512	map/reduce 阶段申请的 container 的内存的大小
	mapreduce.map.java.opts mapreduce.reduce.java.opts		用户设定的 map/reduce 阶段申请的 container 的 JVM 参数。最大堆设定要比申请的内存少一些，用于 JVM 的非堆部分使用。
	mapreduce.admin.map.child.java.opts mapreduce.admin.reduce.child.java.opts	-Xmx500m	管理员设定的 map/reduce 阶段申请的 container 的默认 JVM 启动参数。启动 container 的命令行会先连接管理员设定参数，然后再连接用户设定参数。
yarn-site.xml	yarn.nodemanager.vmem-pmem-ratio	4.0	container 可使用的虚拟映射地址是物理内存的多少倍。
	yarn.scheduler.minimum-allocation-mb	512	container 最小可申请的内存。在调度器中，很多资源计算部分会转化为这个最小值的 N 倍进行计算。所以，设定可分配内存等资源的时候，最好是刚好为这个最小值的倍数。

配置文件	配置项	值	含义
	yarn.scheduler.maximum-allocation-mb	2048	container 最多可申请的内存数量
	yarn.resourcemanager.scheduler.class	FairScheduler /CapacityScheduler	ResourceManager 加载的调度器类实例
yarn-site.private.xml	yarn.nodemanager.resource.memory-mb	4096	每个 nodemanager 可分配的内存总量

6.2. 容量调度器

容量调度器(CapacityScheduler)，由 Yahoo 最初开发，被设计出来使得 hadoop 应用能够被多用户使用，且最大化整个集群资源的吞吐量。

6.2.1. 容量调度器的配置

在 hadoop 集群配置中启动容量调度器之后，调度器会从 classpath 中加载 capacity-scheduler.xml 文件，完成容量调度器的初始化。总结起来有如下特性：

1) **动态更新配置**:容量调度器的配置文件在运行时可以随时重新加载来调整分配参数。除非重启 ResourceManager，否则队列只能添加不能删除，但是允许关闭。修改配置文件后，使用以下命令可以刷新配置。

   ```
   yarn rmadmin -refreshQueues
   ```

2) **树形组织队列:**容量调度器的队列是按照树形结构组织的。根队列只有一个 root 队列。子队列分享父队列的资源。每个队列设定一个容量值代表可使用父队列的容量值，容量值会影响队列的排序。父队列的所有子队列的容量相加一定是 100，否则加载失败。还有一个最大容量值表示子队列绝对不会超过的使用上限。
3) **队列应用限制:**队列可以设定最大提交的应用数量和 AM 占用资源的百分比。AM 占用资源的百分比这个配置是用来计算队列的最大活跃应用数量。这里有个小问题。调度器中最大活跃应用数量=AM 占用资源的百分比*队列最大可使用资源量/最小的 container 分配额度。但是我们在 mapred-site.xml 中会配置 AM 的内存额度会比最小 container 分配额度大，造成最大活跃应用数量虚高（可以理解，如果 YARN 加入不同的计算框架,AM 的分配会不一致，所以这里使用最小 container 分配额度来计算。但是，如果是这样的话，应该直接计算 AM 的内存使用量来控制）。

4) **用户参数限制:**用户可以提交应用到多个队列。不同队列间用户的应用运行情况，不相互影响。用户在队列中最大可占用资源受两个参数控制，一个是单用户占据队列的百分比上限，一个是单用户内存使用上限。具体参看下面的参数表。
5) **资源分配选择**:不同队列之间，按照队列的资源使用比排序。同一队列中的应用按照应用 id 排序，也就是先进先出。
6) **延迟调度**:当调度次数小于本地延迟调度次数的时候不接受机架调度。本地延迟调度次数，由 yarn.scheduler.capacity.node-locality-delay 配置，默认是-1，不开启延迟调度。官方文档中没有提及这个参数。而任意调度的延迟调度上限是应用申请的机器的数量，不能配置。

6.2.2. 容量调度器的参数计算关系

容量调度器通过各种的资源参数限制，来控制整体的资源分布。搞清楚容量调度器上的参数关系，也就基本理解了容量调度器了。在 RM 的管理界面中，你会看到很多参数，如下：

Queue State:	RUNNING
Used Capacity:	0.0%
Absolute Capacity:	75.0%
Absolute Max Capacity:	100.0%
Used Resources:	memory: 0
Num Active Applications:	0
Num Pending Applications:	0
Num Containers:	0
Max Applications:	200
Max Applications Per User:	2000
Max Active Applications:	8
Max Active Applications Per User:	60
Configured Capacity:	75.0%
Configured Max Capacity:	100.0%
Configured Minimum User Limit Percent:	100%
Configured User Limit Factor:	10.0

以下是容量调度器中的各种参数定义和计算关系：

队列容量 = yarn.scheduler.capacity.<queue-path>.capacity/100
队列绝对容量 = 父队列的 队列绝对容量*队列容量
队列最大容量 = yarn.scheduler.capacity.<queue-path>.maximum-capacity/100
队列绝对最大容量 = 父队列的 队列绝对最大容量*队列最大容量
绝对资源使用比 = 使用的资源/全局资源
资源使用比 = 使用的资源/(全局资源 * 队列绝对容量)
最小分配量 = yarn.scheduler.minimum-allocation-mb

用户上限 = MAX(yarn.scheduler.capacity.<queue-path>.minimum-user-limit-percent,1/队列用户数量)
用户调整因子 = yarn.scheduler.capacity.<queue-path>.user-limit-factor
最大提交应用 = yarn.scheduler.capacity.<queue-path>.maximum-applications
如果小于 0 设置为(yarn.scheduler.capacity.maximum-applications*队列绝对容量)
单用户最大提交应用=最大提交应用*(用户上限/100)*用户调整因子
AM 资源占比（AM 可占用队列资源最大的百分比) = yarn.scheduler.capacity.<queue-path>.maximum-am-resource-percent
如果为空，设置为 yarn.scheduler.capacity.maximum-am-resource-percent
最大活跃应用数量 = 全局总资源/最小分配量*AM 资源占比*队列绝对最大容量
单用户最大活跃应用数量 = (全局总资源/最小分配量*AM 资源占比*队列绝对容量)*用户上限*用户调整因子
本地延迟分配次数 = yarn.scheduler.capacity.node-locality-delay

有了这个参数计算关系，就可以根据自己想要的形态配置对应的参数了

6.3. 公平调度器

公平调度器(FairScheduler)，由 Facebook 开发，被设计出来使得 hadoop 应用能够被多用户公平地共享整个集群资源的调度器。

6.3.1. 公平调度器的参数配置

在 hadoop 集群配置中启动公平调度器之后，调度器会从 classpath 中加载 fair-scheduler.xml 和 fair-allocation.xml 文件，完成公平调度器的初始化。其中 fair-scheduler.xml 主要描述重要特性的配置，fair-allocation.xml 主要描述了具体的队列及其参数配置。总结起来有如下特性：

1) **动态更新配置**:公平调度器的 fair-allocation.xml 配置文件在运行时可以随时重新加载来调整分配参数。除非重启 ResourceManager，否则队列只能添加不能删除。修改 fair-allocation.xml 后，使用以下命令可以刷新配置。
   `yarn rmadmin -refreshQueues`
2) **树形组织队列**:公平调度器的队列是按照树形结构组织的。根队列只有一个 root 队列。父子队列除了 ACL 参数外，其余参数都不继承。
3) **队列应用参数**:应用只能提交到叶子队列。受队列最大应用数量限制。队列可以设定权重，最小共享量和最大使用量。权重和最小共享量将影响在公平排序算法中的排名，从而影响资源调度倾向。队列还可以设定最大运行的应用数量。
4) **用户参数限制**:一个用户可以提交应用到多个队列。用户受全局的最大可运行数量限制。
5) **资源分配选择**:资源分配的时候，使用公平排序算法选择要调度的队列，然后在队列中使用先进先出算法或者公平排序算法选择要调度的应用。

6) **资源抢占:**调度器会使用公平资源共享算法计算每个队列应该得到的资源总量。如果一个队列长时间得不到应得到的资源量，调度器可能会杀死占用掉该部分资源的容器。

6.3.2. 公平排序算法

公平排序算法是公平调度器的核心算法。调度器在选取哪个队列和队列中的哪个应用需要优先得到资源调度的时候使用。每个队列或者应用都有以下几个供排序的参数：

1) **资源需求量**：当前队列或者应用希望获得的资源的总量。
2) **最小共享量**：队列的最小共享量在配置中指定。应用的最小共享量为 0。
3) **资源使用量**：当前队列或者应用分配到的总资源。
4) **权值**：队列的权重值在配置中指定。在开启 sizebasedweight 特性的情况下，应用的权重=（log2(资源需求量)）*优先级*调整因子。优先级当前都是 1,。当应用运行超过 5 分钟，调整因子为 3。

排序算法的核心是两个比较体的比较算法，具体如下：

1) 计算比较体是否需要资源。即资源使用量是否小于资源需求量且小于最小共享量。
2) 如果两者都需要资源，计算资源分配比=资源使用量/Min（资源需求量，最小共享量）。资源分配比较小的优先。
3) 如果一个需要，一个不需要，需要的优先。
4) 如果两者都不需要资源，计算使用权值比=资源使用量/权值。使用权值比较小的优先。
5) 如果 2 或者 4 中的比较相同，则先提交的优先。

6.3.3. 公平资源共享算法

公平调度器为公平地分配资源，在开启资源抢占的特性的情况下，可能会杀死部分运行中的容器，释放超额的资源。公平调度器启动的时候会建立一个 UpdateThread 的线程，负责计算公平资源量和进行资源抢占。其中，调度器使用了公平资源共享算法重新计算队列的公平资源量。

公平资源共享算法的目的是为了每个队列公平地使用资源，这个公平体现在每个队列得到的资源比等于他们的权值比。如果只是单纯地求一个**资源权重比**，可以直接相除。但是由于需要满足队列的资源分配满足最小共享量、最大资源量这些队列上下界的限制，**权值资源比**不能直接计算。反过来，如果知道**资源权重比**，就可以计算出集群一共需要多少资源，而且，集群资源需求量=Fun(**资源权重比**)是单调的。所以可以使用二分**资源权重比**的方法，计算集群资源需求量，使得集群资源需求量逼近当前的集群资源量。

具体算法流程如下：

1) 设置根队列的公平资源量为全局资源总和
2) 根队列调用 recomputeFairShares，计算公平资源量
    a) 计算当前队列的分配量=MIN（队列总需求，公平资源量）

b) 计算资源权重比最大值。最大值=2^n,使得 Fun(最大值)>集群资源量>Fun(最大值/2)。
c) 计算资源权重比。采用二分法计算，二分法次数最多 25 次。每个队列的公平资源量=（权重*权重资源比，用最小共享量修正下界，用资源需求量修正上界）
d) 设置每个子队列的公平资源量=资源权重比*权值。
e) 各个子队列调用 recomputeFairShares，递归计算。

6.3.4. 公平调度器的使用设计建议

队列的最小共享量越大，在集群繁忙的时候分配到的资源就越多。但是，如果每个用户都将最小共享量设置到最大，不利于集群间的资源共享。建议，将队列愿意共享出来的内存大小和队列的权值挂钩。含义就是，你愿意共享得越多，在整体资源分配中就越能分得更多的资源。

6.3.5. 公平调度器的全局配置

一些重要的全局配置，如下：

配置项	值	含义
queueMaxAppsDefault	0	自动创建的队列的可提交应用为 0，防止被而已提交应用。
userMaxAppsDefault	100	用户全局可运行应用
defaultQueueSchedulingMode	fair	队列中的应用排序算法，可使用 fair 算法或者 fifo 算法
defaultMinSharePreemptionTimeout	600	当获得资源量小于最小共享量时，抢占资源之前可等待的时间
fairSharePreemptionTimeout	600	当获得资源量小于共享资源量时，抢占资源之前可等待的时间

7. 总结

如果，你刚接触 Hadoop 的话，FifoScheduler 会是个不错的选择。

如果，你只是想控制下部分应用的优先级，同时又要最大化利用集群的话，选择 CapacityScheduler 吧。

如果，你想很多人公平地共享资源的话，只有 FairScheduler 能满足你。