Hadoop YARN公平调度(FairScheduler)介绍

一、介绍

FairScheduler是一个资源分配方式，在整个时间线上，所有的applications平均的获取资源。Hadoop NextGen能够调度多种类型的资源。默认情况下，FairScheduler只是对内存资源做公平的调度(分配)。当集群中只有一个application运行时，那么此application占用这个集群资源。当其他的applications提交后，那些释放的资源将会被分配给新的applications，所以每个applicaiton最终都能获取几乎一样多的资源。不像Hadoop默认的Scheduler(CapacityScheduler)，CapacityScheduler将applications以queue的方式组成，它可以让short applications在何时的时间内完成，而不会starving那些长期运行的applications，它也是一个合理的方式在多个用户之间共享集群。最终，Fair共享也可以与application priorities一起工作-----“priorities”作为权重来使用，以决定每个application需要获取资源的量。

Scheduler将applications以queues的方式组织，在这些queues之间公平的共享资源。默认，所有的users共享一个queue，名称为“default”。如果application在请求资源时指定了queue，那么请求将会被提交到指定的queue中；仍然可以通过配置，根据请求中包含的user名称来分配queue。在每个queue内部，调度策略是在运行中的applicaitons之间共享资源。默认是基于内存的公平共享，不过FIFO和multi-resource with Dominant Resource Fairness也能够配置。Queues可以分级来划分资源，配置权重以特定的比例共享集群资源。

FairScheduler允许为queues分配担保性的最小的共享资源量，这对保证某些用户、groups或者applications总能获取充足的资源是有帮助的。当一个queue中包含applications时，它至少能够获取最小量的共享资源，当queue不在需要时，那些过剩的资源将会被拆分给其他的运行中的application。这就让Scheduler在有效利用资源是，保证了queue的capacity。

FairScheudler在默认情况下允许所有的application运行，但是这也可以通过配置文件来限制每个用户下和每个queue下运行applications的个数。这对限制一个用户一次提交大量applications是有用的，或者通过限制running applications个数来提升性能，因为大量的running applicaiton会导致创建大量的中间数据或者过多的上下文切换。限制applications不会导致随后的提交失败，只是在Scheduler queue中等待，直到先前的application结束。

二、Hierarchical queues

FairScheduler支持分层的queues。所有的queues继承自“root” queue。有效的资源在root子节点中，以典型的公平调度的方式分布；子节点再将分配给自己的资源以相同的方式分配给自己的子节点。applications只能在queue的叶子节点上调度。可以通过FairScheduler相关的配置文件，Queues可以被指定作为其他queues的子节点。

Queue的名字，以其父节点path作为开始，以“.”作为分割符。比如名称为“queue1”的queue作为root的子节点，那么应该表示为“root.queue1”，名称为“queue2”的queue为“parent1”的子节点，那么应该表示为“root.parent1.queue2”。当指明一个queue时，root部分是可选的，比如“queue1”就是指queue1，而queue2指“parent1.queue2”。

此外，FairScheduler允许为每个queue指定不同的policy，每个queue都可以根据用户希望的方式共享queue的资源。这些自定义的Policy可以通过实现“org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.SchedulingPolicy”来构建；FifoPolicy，FairSharePolicy(默认)，以及DominantResouceFairnessPolicy都是内置的，可以直接使用。

三、Installation

为了使用FairScheduler，首先需要在yarn-site.xml配置：
Java代码 收藏代码

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
</property>
```

四、配置

定制FairScheduler涉及到2个文件。首先，scheduler有关的选项可以在yarn-site.xml中配置。此外，多数情况，用户需要创建一个“allocation”文件来列举存在的queues和它们相应的weights和capacities。这个“allocation”文件每隔10秒钟加载一次，更新的配置可以更快的生效。

1、yarn-site.xml中的配置

yarn.scheduler.fair.allocation.file：“allocation”文件的位置，“allocation”文件是一个用来描述queue以及它们的属性的配置文件。这个文件必须为格式严格的xml文件。如果为相对路径，那么将会在classpath下查找此文件(conf目录下)。默认值为“fair-scheduler.xml”。

yarn.scheduler.fair.user-as-default-queue：是否将与allocation有关的username作为默认的queue name，当queue name没有指定的时候。如果设置成false(且没有指定queue name)或者没有设定，所有的jobs将共享“default” queue。默认值为true。

yarn.scheduler.fair.preemption：是否使用“preemption”(优先权，抢占)，默认为fasle，在此版本中此功能为测试性的。

yarn.scheduler.fair.assignmultiple：是在允许在一个心跳中，发送多个container分配信息。默认值为false。

yarn.scheduler.fair.max.assign：如果assignmultuple为true，那么在一次心跳中，最多发送分配container的个数。默认为-1，无限制。

yarn.scheduler.fair.locality.threshold.node：一个float值，在0~1之间，表示在等待获取满足node-local条件的containers时，最多放弃不满足node-local的container的机会次数，放弃的nodes个数为集群的大小的比例。默认值为-1.0表示不放弃任何调度的机会。

yarn.scheduler.fair.locality.threashod.rack：同上，满足rack-local。

yarn.scheduler.fair.sizebaseweight：是否根据application的大小(job的个数)作为权重。默

认为false，如果为true，那么复杂的application将获取更多的资源。

2、一个例子

```
<?xml version="1.0"?>
<allocations>
  <queue name="sample_queue">
    <minResources>10000 mb,0vcores</minResources>
    <maxResources>90000 mb,0vcores</maxResources>
    <maxRunningApps>50</maxRunningApps>
    <weight>2.0</weight>
    <schedulingPolicy>fair</schedulingPolicy>
    <queue name="sample_sub_queue">
      <aclSubmitApps>charlie</aclSubmitApps>
      <minResources>5000 mb,0vcores</minResources>
    </queue>
  </queue>

  <user name="sample_user">
    <maxRunningApps>30</maxRunningApps>
  </user>
  <userMaxAppsDefault>5</userMaxAppsDefault>

  <queuePlacementPolicy>
    <rule name="specified" />
    <rule name="primaryGroup" create="false" />
    <rule name="default" />
  </queuePlacementPolicy>
</allocations>
```

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。
本文链接: 【】（）