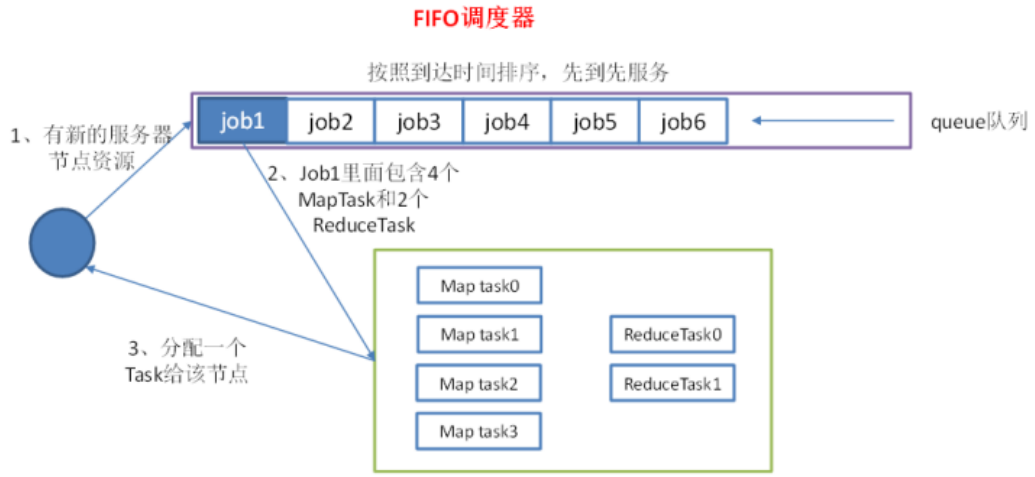
Yarn6 调度器Scheduler

> 背景：　理想情况下，我们应用对Yarn资源的请求应该立刻得到满足，但现实情况资源往往是有限的，特别是在一个很繁忙的集群，一个应用资源的请求经常需要等待一段时间才能的到相应的资源。在Yarn中，负责给应用分配资源的就是Scheduler。其实调度本身就是一个难题，很难找到一个完美的策略可以解决所有的应用场景。为此，Yarn提供了多种调度器和可配置的策略供我们选择。
>
> 目前，Hadoop作业调度器主要有三种，FIFO Schduler、Capacity Scheduler和Fair Scheduler。Hadoop2.7.2 默认的资源调度器是Capacity Scheduler

```
# yarn-default.xml
<property>
    <description>The class to use as the resource scheduler.</description>
    <name>yarn.resourcemanager.scheduler.class</name>

 <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.Capacit
y Scheduler</value>
</property>
```

先进先出调度器（FIFO）



FIFO Scheduler把应用按提交的顺序排成一个队列，这是一个先进先出队列，在进行资源分配的时候，先给队列中最头上的应用进行分配资源，待最头上的应用需求满足后再给下一个分配，以此类推。

缺点

IFO Scheduler是最简单也是最容易理解的调度器，也不需要任何配置，但它并不适用于共享集群。大的应用可能会占用所有集群资源，这就导致其它应用被阻塞。

容量调度器（Capacity）

- 支持多个队列，每个队列可配置一定的资源量，每个队列采用FIFO调度策略
- 为了防止同一个用户的作业独占队列中的资源，该调度器会对同一用户提交的作业所占资源量进行限定
- 首先，计算每个队列中正在运行的任务数与其应该分得的计算资源之间的比值，选择一个比值最小的队列
- 其次，按照作业优先级和提交时间顺序，同时考虑用户资源量限制和内存限制对队列内任务排序
- 三个队列同时按照任务的先后顺序依次执行，比如job1、job2、和job3分别排在队列最前面，是最先运行，也是同时运行

Capacity Scheduler 被设计为允许应用程序在一个可预见的和简单的方式共享集群资源，即"作业队列"。Capacity Scheduler 是根据租户的需要和要求把现有的资源分配给运行的应用程序。Capacity Scheduler 同时允许应用程序**访问还没有被使用的资源，以确保队列之间共享其它队列被允许的使用资源。**管理员可以控制每个队列的容量，Capacity Scheduler 负责把作业提交到队列中。

公平调度器（Fair Scheduler）

特点

支持多队列多用户，每个队列中的资源量可以配置，同一个队列中的作业公平共享队列的所有资源

例子

比如有三个队列：queueA、queueB和queueC，每个队列中的job按照优先级分配资源，优先级越高分配的资源越多，但是每个job都会分配到资源以确保公。在资源有限的情况下，每个job理想情况下获得的计算资源与实际获得的计算资源存在一种差距，这个差距就叫缺额，在同一个队列中，job的资源缺额越大，越先获得资源优先执行。作业是按照缺额的高低来先后执行的，而且可以看到上图有多个作业同时运行。

总结

先进先出

优先提交的，优先执行，后面提交的等待。（火车过隧道）。

容量调度

允许创建多个任务对列，多个任务对列可以同时执行。但是一个队列内部还是先进先出。CDH默认的调度器。

公平调度

第一个程序在启动时可以占用其他对列的资源（100%占用），当其他对列有任务提交时，占用资源的对列需要将资源还给该任务。还资源的时候，效率比较慢。

Capacity调度器配置使用

调度器的使用是通过`yarn-site.xml`配置文件中的`yarn.resourcemanager.scheduler.class`参数进行配置的，默认采用`Capacity Scheduler`调度器。

假设我们有如下层次的队列：

```
root
   ├── prod
   └── dev
         ├── mapreduce
         └── spark
```

下面是一个简单的Capacity调度器的配置文件，文件名为`capacity-scheduler.xml`。在这个配置中，在root队列下面定义了两个子队列prod和dev，分别占40%和60%的容量。需要注意，一个队列的配置是通过属性yarn.sheduler.capacity..指定的，代表的是队列的继承树，如root.prod队列，一般指capacity和maximum-capacity。

修改配置文件

```
vim /usr/local/hadoop/etc/hadoop/capacity-scheduler.xml
```

```
<configuration>
 <property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>prod,dev</value>
  </property>
 <property>
    <name>yarn.scheduler.capacity.root.dev.queues</name>
    <value>mapreduce,spark</value>
  </property>
    <property>
    <name>yarn.scheduler.capacity.root.prod.capacity</name>
    <value>40</value>
  </property>
    <property>
    <name>yarn.scheduler.capacity.root.dev.capacity</name>
    <value>60</value>
  </property>
    <property>
    <name>yarn.scheduler.capacity.root.dev.maximum-capacity</name>
    <value>75</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.dev.mapreduce.capacity</name>
```

```
    <value>50</value>
  </property>
   <property>
    <name>yarn.scheduler.capacity.root.dev.spark.capacity</name>
    <value>50</value>
  </property>
</configuration>
```

我们可以看到，dev队列又被分成了mapreduce和spark两个相同容量的子队列。dev的maximum-capacity属性被设置成了75%，所以即使prod队列完全空闲dev也不会占用全部集群资源，也就是说，prod队列仍有25%的可用资源用来应急。我们注意到，mapreduce和spark两个队列没有设置maximum-capacity属性，也就是说mapreduce或spark队列中的job可能会用到整个dev队列的所有资源（最多为集群的75%）。**而类似的，prod由于没有设置maximum-capacity属性，它有可能会占用集群全部资源。**

关于队列的设置，这取决于我们具体的应用。比如，在MapReduce中，我们可以通过mapreduce.job.queuename属性指定要用的队列。如果队列不存在，我们在提交任务时就会收到错误。如果我们没有定义任何队列，所有的应用将会放在一个default队列中

注意：**对于Capacity调度器，我们的队列名必须是队列树中的最后一部分**，如果我们使用队列树则不会被识别。比如，在上面配置中，我们使用prod和mapreduce作为队列名是可以的，但是如果我们用root.dev.mapreduce或者dev. mapreduce是无效的。

同步其他节点

将新的配置同步到集群所有节点,重启Yarn。

```
xsync capacity-scheduler.xml
```

重启完yarn后，再通过以下命令指定任务提交到的对列。

```
mapreduce.job.queuename=对列名
mapreduce.job.queuename=mapreduce
```

全部命令

```
hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.14.0.jar pi -D mapreduce.job.queuename=mapreduce 10 10
```

程序运行过程中，我们可以通过在网址栏上输入集群所在节点对应ip的端口查看详细信息: http://node01:8088/cluster

相信大家也都看见了，我们在运行上述命令的时候显式申明了队列名称为`mapreduce`。但如果我们在提交MapReduce任务的时候，没有指定队列会发生什么呢?

在上文中我们也说了，如果我们没有定义任何队列，所有的应用将会放在一个`default`队列中。但如果我们在配置文件中加上下面的内容，在未申明程序所属队列名的情况下，就会默认以当前用户名作为队列名。

```
<!-- 如果提交一个任务没有到任何的队列，是否允许创建一个新的队列，默认为true  -->
<property>
    <name>yarn.scheduler.fair.allow-undeclared-pools</name>
    <value>true</value>
</property>

<!–是否提交到默认队列 ，以用户名为默认对列 -->
<property>
    <name>yarn.scheduler.fair.user-as-default-queue</name>
    <value>true</value>
</property>
```

Yarn多租户资源隔离

在一个公司内部的Hadoop Yarn集群，肯定会被多个业务、多个用户同时使用，共享Yarn的资源，如果不做资源的管理与规划，那么整个Yarn的资源很容易被某一个用户提交的Application占满，其它任务只能等待，这种当然很不合理，我们希望每个业务都有属于自己的特定资源来运行MapReduce任务，Hadoop中提供的**公平调度器–Fair Scheduler**，就可以满足这种需求。

Fair Scheduler除了需要在yarn-site.xml文件中启用和配置之外，还需要一个XML文件fair-scheduler.xml来配置资源池以及配额，而该XML中每个资源池的配额可以动态更新，之后使用命令：`yarn rmadmin –refreshQueues` 来使得其生效即可，不用重启Yarn集群。

需要注意的是：动态更新只支持修改资源池配额，如果是新增或减少资源池，则需要重启Yarn集群。

编辑yarn-site.xml

yarn集群主节点中yarn-site.xml添加以下配置

```
cd /usr/local/hadoop/etc/hadoop
```

```
<!--  指定使用fairScheduler的调度方式  -->
<property>
    <name>yarn.resourcemanager.scheduler.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairSchedule
r</value>
</property>

<!--  指定配置文件路径  -->
<property>
    <name>yarn.scheduler.fair.allocation.file</name>
    <value>/export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop/fair-
scheduler.xml</value>
</property>

<!-- 是否启用资源抢占，如果启用，那么当该队列资源使用
yarn.scheduler.fair.preemption.cluster-utilization-threshold 这么多比例的时候，就从
其他空闲队列抢占资源
  -->
<property>
    <name>yarn.scheduler.fair.preemption</name>
    <value>true</value>
```

```
</property>

<property>
    <name>yarn.scheduler.fair.preemption.cluster-utilization-threshold</name>
    <value>0.8f</value>
</property>

<!-- 默认提交到default队列  -->
<property>
    <name>yarn.scheduler.fair.user-as-default-queue</name>
    <value>true</value>
</property>

<!-- 如果提交一个任务没有到任何的队列，是否允许创建一个新的队列，设置false不允许  -->
<property>
    <name>yarn.scheduler.fair.allow-undeclared-pools</name>
    <value>false</value>
</property>
```

添加配置文件fair-scheduler.xml

yarn主节点执行以下命令，添加`faie-scheduler.xml`的配置文件

```
vim /usr/local/hadoop/etc/hadoop/fair-scheduler.xml
```

```
<?xml version="1.0"?>
<allocations>
<!-- users max running apps  -->
<userMaxAppsDefault>30</userMaxAppsDefault>
<!-- 定义队列  -->
<queue name="root">
    <minResources>512mb,4vcores</minResources>
    <maxResources>102400mb,100vcores</maxResources>
    <maxRunningApps>100</maxRunningApps>
    <weight>1.0</weight>
    <schedulingMode>fair</schedulingMode>
    <aclSubmitApps> </aclSubmitApps>
    <aclAdministerApps> </aclAdministerApps>

    <queue name="default">
        <minResources>512mb,4vcores</minResources>
        <maxResources>30720mb,30vcores</maxResources>
        <maxRunningApps>100</maxRunningApps>
        <schedulingMode>fair</schedulingMode>
        <weight>1.0</weight>
        <!--  所有的任务如果不指定任务队列，都提交到default队列里面来 -->
        <aclSubmitApps>*</aclSubmitApps>
    </queue>

<!--

weight
资源池权重

aclSubmitApps
允许提交任务的用户名和组；
```

```
格式为： 用户名 用户组

当有多个用户时候，格式为：用户名1,用户名2 用户名1所属组,用户名2所属组

aclAdministerApps
允许管理任务的用户名和组；

格式同上。
 -->
    <queue name="hadoop">
        <minResources>512mb,4vcores</minResources>
        <maxResources>20480mb,20vcores</maxResources>
        <maxRunningApps>100</maxRunningApps>
        <schedulingMode>fair</schedulingMode>
        <weight>2.0</weight>
        <aclSubmitApps>hadoop hadoop</aclSubmitApps>
        <aclAdministerApps>hadoop hadoop</aclAdministerApps>
    </queue>

    <queue name="develop">
        <minResources>512mb,4vcores</minResources>
        <maxResources>20480mb,20vcores</maxResources>
        <maxRunningApps>100</maxRunningApps>
        <schedulingMode>fair</schedulingMode>
        <weight>1</weight>
        <aclSubmitApps>develop develop</aclSubmitApps>
        <aclAdministerApps>develop develop</aclAdministerApps>
    </queue>

    <queue name="test1">
        <minResources>512mb,4vcores</minResources>
        <maxResources>20480mb,20vcores</maxResources>
        <maxRunningApps>100</maxRunningApps>
        <schedulingMode>fair</schedulingMode>
        <weight>1.5</weight>
        <aclSubmitApps>test1,hadoop,develop test1</aclSubmitApps>
        <aclAdministerApps>test1 group_businessC,supergroup</aclAdministerApps>
    </queue>
</queue>
</allocations>
```

同步数据，重启yarn集群

利用xsync同步各个节点的数据信息。

`cd /export/servers/hadoop-2.6.0-cdh5.14.0/etc/hadoop`

`xsync yarn-site.xml`

`xsync fair-scheduler.xm`

重启yarn集群

`stop-yarn.sh`

`start-yarn.sh`

创建普通用户hadoop

在主节点node01上执行以下的命令添加普通用户

`useradd hadoop`

设置密码
`passwd hadoop`

赋予hadoop用户权限

修改hdfs上面tmp文件夹的权限，不然普通用户执行任务的时候会抛出权限不足的异常。以下命令在root用户下执行。

`groupadd supergroup`
`usermod -a -G supergroup hadoop` 修改用户所属的附加组
`su - root -s /bin/bash -c "hdfs dfsadmin -refreshUserToGroupsMappings"` 刷新用户组信息

使用hadoop用户提交程序

切换到hadoop用户
`su hadoop`

运行程序
`hadoop jar /ur/lcaol/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0-cdh5.14.0.jar pi 10 20`

浏览器查看结果

浏览器界面访问，查看Scheduler，可以清晰的看到任务提交到了hadoop队列里面去了。

`http://node01:8088/cluster/scheduler`