**安装**

如想要将fairscheduler用于自己的 hadoop集群中，需要将fairscheduler的jar文件放置到CLASSPATH中。最简单的方式为：将 hadoop-\*-fairscheduler.jar的jar文件（通常包含在 *HDOOP\_HOME/build/contrib/fairscheduler中* ）拷贝到 *HADOOP\_HOME/lib* 中，或者在 *HADOOP\_CONF\_DIR/hadoop-env.sh* 中所指定的CLASSPATH中包含该jar文件。然后，还需要在 *HADOOP\_CONF\_DIR/mapred-site.xml* 中，做如下配置：

<property>   
<name>mapreduce.jobtracker.taskscheduler</name>   
<value>org.apache.hadoop.mapred.FairScheduler</value>

</property>

    完成，并启动集群后，可以通过在浏览器中访问 *http://<jobtracker URL>/scheduler* 的方式来确认是否已经成功启动。如果没问题，一个"job scheduler administration"的页面将会显示出来。该page中包含的内容在下文介绍。

**配置**

    FairScheduler包含有两个配置文件，算法相关的配置选项都需要在HADOOP\_CONF\_DIR/mapred-site.xml中进行配置，还有另外一个独立的配置文件，默认情况下位置为：HADOOP\_CONF\_DIR/fair-scheduler.xml，该文件是用来配置 pool，最小槽位数，同时运行的job数限制，以及优先抢占时间间隔等配置。这个文件会被fairscheduler每隔一段时间就动态的加载一次，这样管理员就可以在修改了配置后，不用重启jobtracker就能令配置生效。   
    最简单的情况下，管理员不需要对该文件做任何的配置就可以使用fairscheduler。

**在mapred-site.xml中配置的调度选项**

以下配置选项可以在mapred- site.xml中进行配置，用来对fairscheduler的调度算法进行配置：   
**基本选项**

|  |  |
| --- | --- |
| **配置选项** | **描述** |
| mapred.fairscheduler.preemption | boolean 选项，用来指定是否开启优先抢占功能，默认为: false. |
| mapred.fairscheduler.preemption.only.log | 抢占功能开启后，值为true时只记录日志；默认:false |
| mapred.fairscheduler.pool | 定义一个pool，如果对该选项进行了配置，那么mapred.fairscheduler.poolnameproperty的配置就会被忽略。 |
| mapred.fairscheduler.poolnameproperty | 定义哪一个jobconf的配置属性用来确定一个job所属的pool名称。是一个字符串值。默认情况下，用来确认一个job是属于哪个pool的配置选项为：user.name(one pool for each user)。另外一个比较有用的配置为：group.name，用来位每一个Unix Group创建一个pool。 |
| mapred.fairscheduler.allocation.file | 该选项用来指定fairscheduler是要加载哪个文件来作为调度器特有的配置文件，默认为：HADOOOP\_CONF\_DIR/fair- scheduler.xml。  该配置项的值必须是一个文件的绝对路径 |

**高级配置**

|  |  |
| --- | --- |
| **配置选项** | **描述** |
| mapred.fairscheduler.sizebasedweight | 该选项是一个boolean值，如果将该选项设置成true，那么当计算job的fair share时，会将该job的size（job的map和reduce数）作为计算fair share的依据。默认情况下该选项为false，也就是说默认job的fair share计算仅仅是由job的优先级来决定。当开启该选项后，fair share的变化和job size并非线性关系，但是会将job size计入用来计算fair share的因素。这样的机制能够使得大作业，需要运行时间长的作业，能得到更多的资源，而同时小作业也能够在一定合理的时间内完成。 |
| mapred.fairscheduler.preemption.only.log | 该选项是一个boolean值，默认为false。当开启时，fairscheduler会在内部调度时采用优先抢占的计算方法，但是并不会真正的kill task，而是当需要kill task的时候，仅仅向日志中写入一条记录，告诉管理员此时如果在真正的优先抢占调度模式下，会kill掉哪个task。这样提供了用户一种试运行的环境，使得用户可以根据日志的记录情况来调整preemption的timeout是否设置的合理。日志的记录会在jobtracker的log中，如下列格式：  Should preempt 2 tasks for job\_20090101337\_0001: tasksDueToMinShare = 2, tasksDueToFairShare = 0 |
| mapred.fairscheduler.update.interval | 该选项用来设置fairscheduler更新各job的fair share的时间间隔。默认为500毫秒。通常情况下，这个时间间隔比较合适，如果集群规模超过500台甚至更多，可以考虑适当调大一点。视情况而定吧 |
| mapred.fairscheduler.preemption.interval | I该选项用来设置在资源抢占模式下，fairscheduler内部线程用来check是否需要抢占task槽位的时间间隔。默认为15秒(15000)。对于这个配置，尽量不要设置的过小，不然很有可能会误杀很多的task，造成资源浪费。 |
| mapred.fairscheduler.weightadjuster | 该选项是fairscheduler的一个扩展点，用户可以自定义自己的class类，在类中自己定义算法来调整running job的比重，从而影响对job的fair share的计算，进而影响到对job的槽位分配。用户自定义的类需要集成WeightAdjuster接口。在fairscheduler的发行版中已经附带了一个该接口的实现类，NewJobWeightBooster，该类的逻辑为：在job的运行前5分钟将job的比重加大，使得新提交上来的作业在前5分钟内获得更多的槽位资源，以期望短小的作业能够尽快运行。若要使用该实现，可以将本配置选项设置成该类的全路径名，如 org.apache.hadoop.mapred.NewJobWeightBooster。这个类本身也提供了两个配置选项来配置对作业的比重计算的调整因子：   * *mapred.newjobweightbooster.factor*  该选项用来设置新提交的作业所占比重的起始因子，默认为3 * *mapred.newjobweightbooster.duration*  设置因子的持续时间，默认为5分钟(300000ms) |
| mapred.fairscheduler.loadmanager | 该配置选项也是fairscheduler的一个扩展点，loadmanager可以让用户自己定义需要给每个tasktracker分配多少个map和多少个reduce task。该扩展点的实现类需要继承LoadManager接口。主要是可以让调度器根据每台tasktracker所剩余的内存和cpu使用率来决定分配多少task到该tasktracker |
| mapred.fairscheduler.taskselector | An extension point that lets you specify a class that determines which task from within a job to launch on a given tracker. This can be used to change either the locality policy (e.g. keep some jobs within a particular rack) or the speculative execution algorithm (select when to launch speculative tasks). The default implementation uses Hadoop's default algorithms from JobInProgress.   该选项也是一个fairscheduler的扩展点，可以让用户置顶自定义的实现类来决定选择哪些task，分配到哪些tracker上。如此可以利用它来改变task分配原则（比如将某些task本配到同一个rack上等）以及预测执行的行为算法。默认使用的时hadoop 的JobInProgress中的实现算法（随着版本不同会有不一样）。 |

**Allocation File (fair-scheduler.xml)**

Allocation File用来配置minimum share，并行job limit，每个pool的weight以及preemption timeout等。只有当某个用户或者某个pool需要设置的值跟默认的不一样时才需要进行额外配置。该配置文件默认的路径为 *HADOOP\_HOME/conf/fair- scheduler.xml。* 该配置中可以置顶如下配置项：

* pool相关的配置项，这些配置项用来pool 相关的设置，包含如下子选项：
  + *minMaps* and *minReduces* ,用来指定该pool的最小task槽位数。
  + *maxMaps* and *maxReduces* ,用来指定该pool最大并行执行的task 槽位数。
  + *schedulingMode* , 调度模式，可以是根据job的fair share来调度的fair模式，或者是先进先出的Fifo模式
  + *maxRunningJobs* ,用来指定该pool中最大并行运行的job数（默认没有限制）
  + *weight* , to share the cluster non-proportionally with other pools. For example, a pool with weight 2.0 will get a 2x higher share than other pools. The default weight is 1.0. 用来设置该pool使用集群资源的比重。比如将该值配置成2.0，那么该pool内的job获得集群槽位的可能性就2倍于该选项为1.0的pool中的 job。默认该选项为1.0
  + *minSharePreemptionTimeout* ,当current pool中所使用的集群槽位少于其min share时，最多多长时间内开始从其他抢占了别组槽位的pool中将槽位抢占回来。默认为infinite
* *user* ，用来设置对于一个用户来说，最多可以并行执行的job为多少个。通常情况下，每一个用户都会对应到一个pool，所以为每一个user定义该选项通常不太必要。
* *poolMaxJobsDefault* ,一个全局的限制选项，当每个pool没有单独设置该pool中最多能并行运行多少个job时，用该选项值来限制。
* *userMaxJobsDefault* ,一个全局的限制选项，当每个用户没有单独设置该pool中最多能并行运行多少个job时，用该选项值来限制。
* *defaultMinSharePreemptionTimeout* ,用来设置默认多长时间开始对使用集群槽位高于其min share的pool进行资源抢占。
* *fairSharePreemptionTimeout* ,用来设置当job所使用的槽位数低于其 fair share的一半后，多少时间内开始资源抢占。
* *defaultPoolSchedulingMode* ,默认的调度模式（fair | fifo） 默认为fair

Pool and user elements only required if you are setting non-default values for the pool/user. That is, you do not need to declare all users and all pools in your config file before running the fair scheduler. If a user or pool is not listed in the config file, the default values for limits, preemption timeouts, etc will be used.   
pool和user的配置项仅仅在需要对其设置非默认值时才需要。所以，并不需要明确的对所有的 pool或者user进行单独的配置。如果一个pool或者user没有在config文件中明确配置，那么就会用默认的值。   
config文件的配置通常如以下格式:   
<?xml version="1.0"?>   
<allocations>   
<pool name="sample\_pool">   
  <minMaps>5</minMaps>   
  <minReduces>5</minReduces>   
  <maxMaps>25</maxMaps>   
  <maxReduces>25</maxReduces>   
  <minSharePreemptionTimeout>300</minSharePreemptionTimeout>   
</pool>   
<user name="sample\_user">  
   <maxRunningJobs>6</maxRunningJobs>   
</user>  
<userMaxJobsDefault>3</userMaxJobsDefault>   
<fairSharePreemptionTimeout>600</fairSharePreemptionTimeout>   
</allocations>

该配置文件创建了一个pool名为 sample\_pool，集群确保给该pool的最小map和reduce数为之5，该pool的最小share preemption的时间维度为300秒，这就意味着，如果在5分钟内该pool仍然没有使用槽位打到5个map 5个reduce，那么调度器就要从别的超出其自身min share的pool中抢占计算槽位来满足该pool的计算任务。该pool还设置了最大运行的map和reduce数，这就是说，当该pool内的所有作业加起来使用了25个map，25个reduce后，就不会再给这个pool内的job分配槽位，即使在计算过程中该pool的fair share升高也不会。

上述配置文件还对每个用户最多可以运行的job数做了限制，为3个。但是sample\_user可以并行运行6个作业。最后，该配置文件还设置了抢占槽位的时间超时为600秒。如果一个job所占用的槽位低于其fair share的一半超过10分钟，那么他将从其他pool中抢占计算槽位为己用。这里需要注意的是，要使这个配置选项生效，首先需要在mapred- site.xml中开启该功能。

    其他没有在该配置文件中明确设置的pool，将不会被保证最小槽位数以及抢占槽位的timeout。并且，没有明确限制并行运行job上限的user或者 pool，默认都是可以运行无限多个作业的。

**管理作业**

FairScheduler提供了两种实时的对作业运行情况进行调整管理的手段：

1. 可以在jobtracker运行时期实时的修改 config文件，对pool或者user的min share，limits，weights，抢占超时或者调度模式等进行修改，fairscheduler会每隔10-15秒钟重新加载一遍配置文件。
2. 可以在调度器webui的页面上进行实时的修改，调度链接为 *http://<JobTracker URL>/scheduler* 。在该页面上，可以修改作业的优先级，将一个作业转移到其他pool中去，并实时的查看每个作业的fairshare。

如下job运行信息能够在webui上查看到：

* *Submitted* - 提交作业的日期和时间
* *JobID, User, Name* -
* *Pool* - 当前job所属的pool，可以修改该选项以将该job转移到其他pool中去
* *Priority* - 作业优先级，可以在webui上修改作业的优先级
* *Maps/Reduces Finished* : 已经完成map数和reduce数
* *Maps/Reduces Running* : 当前正在运行的map数和reduce数
* *Map/Reduce Fair Share* : fair share权重值

另外，还可以通过 *http://<JobTracker URL>/scheduler?advanced* 来查看一些作业运行期间的高级信息。可以看到

* *Maps/Reduce Weight* :每隔job的weight信息。该 weight值取决于作业的优先级以及在开启了weightsize的情况下每个job的size，以及每个作业的运行时间。