Hadoop作业JVM堆大小设置优化

前段时间，公司Hadoop集群整体的负载很高，查了一下原因，发现原来是客户端那边在每一个作业上擅自配置了很大的堆空间，从而导致集群负载很高。下面我就来讲讲怎么来现在客户端那边的JVM堆大小的设置。

我们知道，在mapred-site.xml配置文件里面有个mapred.child.java.opts配置，专门来配置一些诸如堆、垃圾回收之类的。看下下面的配置：

```
 <property>
   <name>mapred.child.java.opts</name>
   <value>-Xmx200m</value>
   <description>Java opts for the task tracker child processes.
   The following symbol, if present, will be interpolated: @taskid@ is
   replaced by current TaskID. Any other occurrences of '@' will go unchanged.
   For example, to enable verbose gc logging to a file named for the taskid in
   /tmp and to set the heap maximum to be a gigabyte, pass a 'value' of:
         -Xmx1024m -verbose:gc -Xloggc:/tmp/@taskid@.gc

   The configuration variable mapred.child.ulimit can be used to control the
   maximum virtual memory of the child processes.
   </description>
</property>
```

默认情况下，-Xmx都是配置200m的，但是在实际情况下，这个显然是不够用的，所以导致客户端那边会设置更大的值。那怎么来限制用户随便设置Xmx的值呢？有下面两种方法：

一、可以自己定义一个变量，比如如下：

```
<property>
   <name>mapred.task.java.opts</name>
   <value>-Xmx2000m</value>
</property>

<property>
   <name>mapred.child.java.opts</name>
   <value>${mapred.task.java.opts} -Xmx1000m</value>
   <final>true</final>
</property>
```

上面的mapred.task.java.opts属性是我们自己定义的，可以公布给用户配置；然后在mapred.child.java.opts中获取到mapred.task.java.opts的值，同时mapred.child.java.opts属性的final被设置为true，也就是不让客户修改。所以用户对mapred.child.java.opts直接配置是无效的；而且这里我们在获取${mapred.task.java.opts}之后再添加了-Xmx1000m，而在Java中，如果相同的jvm arg写在一起，比如"-Xmx2000m -Xmx1000m"，后面的会覆盖前面的，也就是说最终“-Xmx1000m”才会生效，通过这种方式，我们就可以有限度的控制客户端那边的heap size了。同样的道理，其他想覆盖的参数我们也可以写到后面。

我们可以通过

```
<property>
  <name>mapred.map.child.java.opts</name>
  <value>
    -Xmx512M
  </value>
</property>

<property>
  <name>mapred.reduce.child.java.opts</name>
  <value>
    -Xmx1024M
  </value>
</property>
```

来分别配置作业的Map和Reduce阶段的heap的大小。

二、通过mapreduce.admin.map.child.java.opts和和mapreduce.admin.reduce.child.java.opts设定

上述限制客户端那边随便设置堆大小是通过重新定义一个变量给用户使用，这样用户得使用新的变量来定义一些JVM相关的设定，如果用户那边的脚本非常多，他们就需要一个一个脚本的修改mapred.child.java.opts为mapred.task.java.opts。这样会恨不方便。这里介绍另外一种方法。可以通过mapreduce.admin.map.child.java.opts和mapreduce.admin.reduce.child.java.opts来限定作业map和reduce的堆的大小。他们都是管理员设定的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。我们来看看Hadoop源码是怎么获取客户端和管理员JVM参数的获取的：

```
private static String getChildJavaOpts(JobConf jobConf, boolean isMapTask) {
```

```
    String userClasspath = "";
    String adminClasspath = "";
    if (isMapTask) {
      userClasspath =
          jobConf.get(
              JobConf.MAPRED_MAP_TASK_JAVA_OPTS,
              jobConf.get(
                  JobConf.MAPRED_TASK_JAVA_OPTS,
                  JobConf.DEFAULT_MAPRED_TASK_JAVA_OPTS)
          );
      adminClasspath =
          jobConf.get(
              MRJobConfig.MAPRED_MAP_ADMIN_JAVA_OPTS,
              MRJobConfig.DEFAULT_MAPRED_ADMIN_JAVA_OPTS);
    } else {
      userClasspath =
          jobConf.get(
              JobConf.MAPRED_REDUCE_TASK_JAVA_OPTS,
              jobConf.get(
                  JobConf.MAPRED_TASK_JAVA_OPTS,
                  JobConf.DEFAULT_MAPRED_TASK_JAVA_OPTS)
              );
      adminClasspath =
          jobConf.get(
              MRJobConfig.MAPRED_REDUCE_ADMIN_JAVA_OPTS,
              MRJobConfig.DEFAULT_MAPRED_ADMIN_JAVA_OPTS);
    }

    // Add admin classpath first so it can be overridden by user.
    return adminClasspath + " " + userClasspath;
}
```

通过上面的代码，我们可以发现Hadoop是先获取管理员的JVM参数配置，然后连接客户端那边JVM参数的配置。这样如果管理员那边的配置在客户端那边也配置了，那么客户端这边的配置将会覆盖掉管理员那边的参数配置。所以我们可以修改源码，将 return adminClasspath + " " + userClasspath;修改为 return userClasspath + " " + adminClasspath;然后在mapred-site.xml文件做如下配置：

```
<property>
    <name>mapreduce.admin.map.child.java.opts</name>
    <value>-Xmx1000m</value>
</property>
<property>
```

```
    <name>mapreduce.admin.reduce.child.java.opts</name>
    <value>-Xmx1000m</value>
</property>
```

这样，我们就可以覆盖客户端那边的配置。

总结

上面两种方法虽然能在一定程度上限制客户端使用堆的大小，但是这样的解决办法不是很好的！因为我们设定所有作业的堆大小都是1000M，但是实际情况下，很多作业不一定都用得到1000M；而且在一些情况下，有些作业用到的heap可能大于1000M，这样会使这样的作业出现OOM的问题。