HamaWhite 原创，转载请注明出处！欢迎大家加入**Giraph 技术交流群**： 228591158

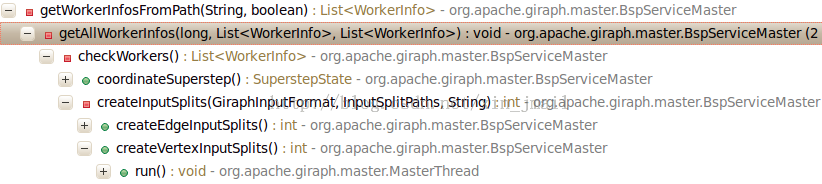
欢迎访问： 西北工业大学 - 大数据与知识管理研究室 （Northwestern Polytechnical University - BigData and Knowledge Management Lab），链接：<http://wowbigdata.cn/>，<http://wowbigdata.net.cn/>，[http://wowbigdata.com.cn](http://wowbigdata.com.cn/)。

本文的目的：说明Giraph如何借助ZooKeeper来实现Master与Workers间的同步（?不太确定）。

环境：在单机上（机器名：giraphx）启动了2个workers。

Giraph遵从单Master多Workers结构，BSPServiceMaster使用MasterThread线程来进行全局的同步。每个Worker启动成功后，会向Master汇报自身的健康状况，那么Master是如何检测Workers是否都成功启动了？

1. Master在ZooKeeper上创建两个目录，\_workerHealthyDir和 \_workerUnhealthyDir，分别用来记录Healthy Workers和UnHealthy Workers。主要在BspServiceMaster类中的getAllWorkerInfos()方法来完成，其调用关系如下，注意下getAllWorkerInfos()到MasterThread.run()方法调用关系，比较难找。



    创建的两个目录如下：

     /\_hadoopBsp/job\_201404102333\_0002/\_applicationAttemptsDir/0/\_superstepDir/-1/\_workerHealthyDir

     /\_hadoopBsp/job\_201404102333\_0002/\_applicationAttemptsDir/0/\_superstepDir/-1/\_workerUnhealthyDir

2.  每个Worker在setup()中，调用registerHealth()方法来注册自身的状态。若自身是Healthy的，则在\_workerHealthyDir目录下添加子节点 /wokerInfo.getHostNameId()，否则在\_workerUnhealthyDir目录下添加。wokerInfo.getHostNameId()为：Hostname+“\_”+TaskId。 Task1和Task2 （Task 0是master） 创建的子节点如下：

    /\_hadoopBsp/job\_201404102333\_0002/\_applicationAttemptsDir/0/\_superstepDir/-1/\_workerHealthyDir/giraphx\_1

    /\_hadoopBsp/job\_201404102333\_0002/\_applicationAttemptsDir/0/\_superstepDir/-1/\_workerHealthyDir/giraphx\_2

3. Master 在checkWorkers()方法中，在While死循环中（实际有超时限制），通过调用getAllWorkerInfos()方法来获取\_workerHealthyDir目录下的子节点，然后比较子节点数目是否达到maxWorkers(启动job时定义的，-w参数)。

    若小于maxWorkers，则继续调用getAllWorkerInfos()方法进行下一轮检测；若等于maxWorker，退出While循环，然后返回healthyWorkersInfoList：[Worker(hostname=giraphx, MRtaskID=1, port=30001), Worker(hostname=giraphx, MRtaskID=2, port=30002)] 。

    问题：由于在分布式环境中，每个Worker和Maste都是并行运行，彼此不知道对方的运行情况。**上述第3步骤中，若还有子节点还没有创建，就一直在while死循环中调用来检测getAllWorkerInfos()方法检测，效率比较低下，当然也比较**笨！。

    Giraph借用ZooKeeper来高效的进行检测。设计理念如下：

    1).  master在获取子节点时，注册Watcher（为注册器，用于触发相应事件）。若某个task创建了子节点后，就会触发Watcher事件。

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/23450101)[在CODE上查看代码片](https://code.csdn.net/snippets/287023)

1. **private** List<WorkerInfo> getWorkerInfosFromPath(String workerInfosPath,
2. **boolean** watch) {
3. List<WorkerInfo> workerInfoList = **new** ArrayList<WorkerInfo>();
4. List<String> workerInfoPathList;
5. //此处注册Watcher监听事件
6. workerInfoPathList =
7. getZkExt().getChildrenExt(workerInfosPath, watch, **false**, **true**);
9. **for** (String workerInfoPath : workerInfoPathList) {
10. WorkerInfo workerInfo = **new** WorkerInfo();
11. WritableUtils.readFieldsFromZnode(
12. getZkExt(), workerInfoPath, **true**, **null**, workerInfo);
13. workerInfoList.add(workerInfo);
14. }
15. LOG.info("D-workerInfoList: "+workerInfoList);
16. **return** workerInfoList;
17. }

    2).  若子节点数目小于maxWorkers，就调用 workerHealthRegistrationChanged的await()方法释放当前线程的锁，陷入等待状态。不会进行无用的检测。

           说明：workerHealthRegistrationChanged为PredicateLock类型（implements BspEvent接口），PredicateLock里面使用可重入锁 ReentrantLock和Condition进行线程的控制。

    3). 当某个task创建了子节点后，触发Watcher事件。调用BspService中的public final void Process(WatchedEvent event)事件，该方法根据事件的路径来激活相应的BspEvent事件。此处对应的是：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/23450101)[在CODE上查看代码片](https://code.csdn.net/snippets/287023)

1. workerHealthRegistrationChanged.signal();
2. eventProcessed = **true**;

         实验运行如下：

s(926)) - process: Got a new event, path = /\_hadoopBsp/job\_201404102333\_0002/\_applicationAttemptsDir/0/\_superstepDir/-1/\_workerHealthyDir, type = NodeChildrenChanged, state = SyncConnected

INFO bsp.BspService (BspService.java:process(960)) - process: workerHealthRegistrationChanged (worker health reported - healthy/unhealthy )

          这样就会激活master线程，开始下一轮检测。

     4). 子节点数目等于maxWorkers时，就停止。

**总结：每创建一个子节点时，才会进行一次检测，效率较高！**

**本人原创，转载请注明出处！ 本人QQ：530422429，欢迎大家指正、讨论。**