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/)。

说明：

(1) 实验环境.

   三台服务器：test165、test62、test63。test165同时是JobTracker和TaskTracker.

   测试例子：官网自带的SSSP程序，数据是自己模拟生成。

   运行命令：hadoop jar giraph-examples-1.0.0-for-hadoop-0.20.203.0-jar-with-dependencies.jar org.apache.giraph.GiraphRunner org.apache.giraph.examples.SimpleShortestPathsVertex -vif org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat -vip /user/giraph/SSSP -of org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /user/giraph/output-sssp-debug-7 -w 5

(2). 为节约空间，下文中所有代码均为核心代码片段。

(3). core-site.xml中hadoop.tmp.dir的路径设为：/home/hadoop/hadooptmp

(4).写本文是多次调试完成的，故文中的JobID不一样，读者可理解为同一JobID.

(5). 后续文章也遵循上述规则。

1. org.apache.giraph.graph.GraphMapper类

 Giraph中自定义org.apache.giraph.graph.GraphMapper类来继承Hadoop中的 org.apache.hadoop.mapreduce.Mapper<Object,Object,Object,Object>类，覆写了setup()、map()、cleanup()和run()方法。GraphMapper类的说明如下：

This mapper that will execute the BSP graph tasks alloted to this worker. All tasks will be performed by calling the GraphTaskManager object managed by this GraphMapper wrapper classs. Since this mapper will not be passing data by key-value pairs through the MR framework, the Mapper parameter types are irrelevant, and set to Object type.

BSP的运算逻辑被封装在GraphMapper类中，其拥有一GraphTaskManager对象，用来管理Job的tasks。每个GraphMapper对象都相当于BSP中的一个计算节点（compute node）。

在GraphMapper类中的setup()方法中，创建GraphTaskManager对象并调用其setup()方法进行一些初始化工作。如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/22306495)[在CODE上查看代码片](https://code.csdn.net/snippets/265353)

1. @Override
2. **public** **void** setup(Context context)
3. **throws** IOException, InterruptedException {
4. // Execute all Giraph-related role(s) assigned to this compute node.
5. // Roles can include "master," "worker," "zookeeper," or . . . ?
6. graphTaskManager = **new** GraphTaskManager<I, V, E, M>(context);
7. graphTaskManager.setup(
8. DistributedCache.getLocalCacheArchives(context.getConfiguration()));
9. }

map()方法为空，因为所有操作都被封装在了GraphTaskManager类中。在run()方法中调用GraphTaskManager对象的execute()方法进行BSP迭代计算。

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/22306495)[在CODE上查看代码片](https://code.csdn.net/snippets/265353)

1. @Override
2. **public** **void** run(Context context) **throws** IOException, InterruptedException {
3. // Notify the master quicker if there is worker failure rather than
4. // waiting for ZooKeeper to timeout and delete the ephemeral znodes
5. **try** {
6. setup(context);
7. **while** (context.nextKeyValue()) {
8. graphTaskManager.execute();
9. }
10. cleanup(context);
11. // Checkstyle exception due to needing to dump ZooKeeper failure
12. } **catch** (RuntimeException e) {
13. graphTaskManager.zooKeeperCleanup();
14. graphTaskManager.workerFailureCleanup();
15. }
16. }

2. org.apache.giraph.graph.GraphTaskManager 类

功能：The Giraph-specific business logic for a single BSP compute node in whatever underlying type of cluster our Giraph job will run on. Owning object will provide the glue into the underlying cluster framework and will call this object to perform Giraph work.

下面讲述setup()方法，代码如下。

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/22306495)[在CODE上查看代码片](https://code.csdn.net/snippets/265353)

1. /\*\*
2. \* Called by owner of this GraphTaskManager on each compute node
3. \* @param zkPathList the path to the ZK jars we need to run the job
4. \*/
5. **public** **void** setup(Path[] zkPathList) **throws** IOException, InterruptedException {
6. context.setStatus("setup: Initializing Zookeeper services.");
7. locateZookeeperClasspath(zkPathList);
8. serverPortList = conf.getZookeeperList();
9. **if** (serverPortList == **null** && startZooKeeperManager()) {
10. **return**; // ZK connect/startup failed
11. }
12. **if** (zkManager != **null** && zkManager.runsZooKeeper()) {
13. LOG.info("setup: Chosen to run ZooKeeper...");
14. }
15. context.setStatus("setup: Connected to Zookeeper service " +serverPortList);
16. **this**.graphFunctions = determineGraphFunctions(conf, zkManager);
17. instantiateBspService(serverPortList, sessionMsecTimeout);
18. }

依次介绍每个方法的功能：

    1) locateZookeeperClasspath(zkPathList)：找到ZK jar的本地副本，其路径为：/home/hadoop/hadooptmp/mapred/local/taskTracker/root/jobcache/job\_201403270456\_0001/jars/job.jar ,用于启动ZooKeeper服务。  
    2) startZooKeeperManager()，初始化和配置ZooKeeperManager。定义如下，

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/22306495)[在CODE上查看代码片](https://code.csdn.net/snippets/265353)

1. /\*\*
2. \* Instantiate and configure ZooKeeperManager for this job. This will
3. \* result in a Giraph-owned Zookeeper instance, a connection to an
4. \* existing quorum as specified in the job configuration, or task failure
5. \* @return true if this task should terminate
6. \*/
7. **private** **boolean** startZooKeeperManager()
8. **throws** IOException, InterruptedException {
9. zkManager = **new** ZooKeeperManager(context, conf);
10. context.setStatus("setup: Setting up Zookeeper manager.");
11. zkManager.setup();
12. **if** (zkManager.computationDone()) {
13. done = **true**;
14. **return** **true**;
15. }
16. zkManager.onlineZooKeeperServers();
17. serverPortList = zkManager.getZooKeeperServerPortString();
18. **return** **false**;
19. }

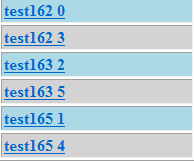
org.apache.giraph.zk.ZooKeeperManager 类，功能：Manages the election of ZooKeeper servers, starting/stopping the services, etc.

ZooKeeperManager类的setup()定义如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/22306495)[在CODE上查看代码片](https://code.csdn.net/snippets/265353)

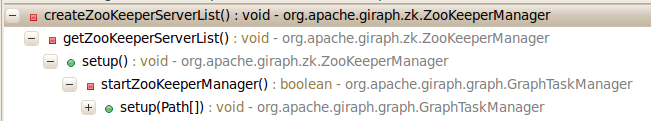
1. /\*\*
2. \* Create the candidate stamps and decide on the servers to start if
3. \* you are partition 0.
4. \*/
5. **public** **void** setup() **throws** IOException, InterruptedException {
6. createCandidateStamp();
7. getZooKeeperServerList();
8. }

createCandidateStamp()方法在 HDFS上 的\_bsp/\_defaultZkManagerDir/job\_201403301409\_0006/\_task 目录下为每个task创建一个文件，文件内容为空。文件名为本机的Hostname+taskPartition，如下截图：



运行时指定了5个workers(-w 5)，再加上一个master，所有上面有6个task。

getZooKeeperServerList()方法中，**taskPartition为0的task会调用createZooKeeperServerList()方法创建ZooKeeper server List**，也是创建一个空文件，通过文件名来描述Zookeeper servers。



createZooKeeperServerList核心代码如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/22306495)[在CODE上查看代码片](https://code.csdn.net/snippets/265353)

1. /\*\*
2. \* Task 0 will call this to create the ZooKeeper server list.  The result is
3. \* a file that describes the ZooKeeper servers through the filename.
4. \*/
5. **private** **void** createZooKeeperServerList() **throws** IOException,
6. InterruptedException {
7. Map<String, Integer> hostnameTaskMap = Maps.newTreeMap();
8. **while** (**true**) {
9. FileStatus [] fileStatusArray = fs.listStatus(taskDirectory);
10. hostnameTaskMap.clear();
11. **if** (fileStatusArray.length > 0) {
12. **for** (FileStatus fileStatus : fileStatusArray) {
13. String[] hostnameTaskArray =
14. fileStatus.getPath().getName().split(HOSTNAME\_TASK\_SEPARATOR);
16. **if** (!hostnameTaskMap.containsKey(hostnameTaskArray[0])) {
17. hostnameTaskMap.put(hostnameTaskArray[0],
18. **new** Integer(hostnameTaskArray[1]));
19. }
20. }
21. **if** (hostnameTaskMap.size() >= serverCount) {
22. **break**;
23. }
24. Thread.sleep(pollMsecs);
25. }
26. }
27. }

    首先获取taskDirectory（\_bsp/\_defaultZkManagerDir/job\_201403301409\_0006/\_task）目录下文件，如果当前目录下有文件，则把文件名（Hostname+taskPartition）中的Hostname和taskPartition存入到hostNameTaskMap中。扫描taskDirectory目录后，若hostNameTaskMap的size大于serverCount（等于GiraphConstants.java中的ZOOKEEPER\_SERVER\_COUNT变量，定义为1），就停止外层的循环。外层循环的目的是：因为taskDirectory下的文件每个task文件时多个task在分布式条件下创建的，有可能task 0在此创建server List时，别的task还没有生成后task文件。Giraph默认为每个Job启动一个ZooKeeper服务，也就是说只有一个task会启动ZooKeeper服务。

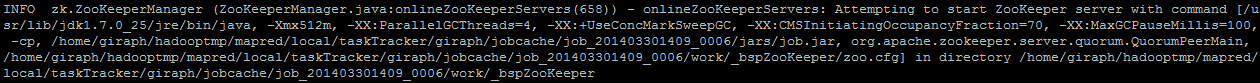
    经过多次测试，task 0总是被选为ZooKeeper Server ，因为在同一进程中，扫描taskDirectory时，只有它对应的task 文件（其他task的文件还没有生成好），然后退出for循环，发现hostNameTaskMap的size等于1，直接退出while循环。那么此处就选了test162 0。

    最后，创建了文件：\_bsp/\_defaultZkManagerDir/job\_201403301409\_0006/zkServerList\_test162 0

http://img.blog.csdn.net/20140330192226750

    onlineZooKeeperServers()，根据zkServerList\_test162 0文件，Task 0 先生成zoo.cfg配置文件，使用ProcessBuilder来创建ZooKeeper服务进程，然后Task 0 再通过socket连接到ZooKeeper服务进程上，最后创建文件 \_bsp/\_defaultZkManagerDir/job\_201403301409\_0006/\_zkServer/test162 0 来标记master任务已完成。worker一直在进行循环检测master是否生成好 \_bsp/\_defaultZkManagerDir/job\_201403301409\_0006/\_zkServer/test162 0**，**即worker等待直到master上的ZooKeeper服务已经启动完成。

启动ZooKeeper服务的命令如下：



    3) determineGraphFunctions()。

     GraphTaskManager类中有CentralizedServiceMaster对象和CentralizedServiceWorker 对象，分别对应于master和worker。每个BSP compute node扮演的角色判定逻辑如下：

     a) If not split master, everyone does the everything and/or running ZooKeeper.

     b) If split master/worker, masters also run ZooKeeper

     c) If split master/worker == true and giraph.zkList is set, the master will not instantiate a ZK instance, but will assume a quorum is already active on the cluster for Giraph to use.

该判定在GraphTaskManager 类中的静态方法determineGraphFunctions()中定义，片段代码如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/xin_jmail/article/details/22306495)[在CODE上查看代码片](https://code.csdn.net/snippets/265353)

1. **private** **static** GraphFunctions determineGraphFunctions(
2. ImmutableClassesGiraphConfiguration conf,
3. ZooKeeperManager zkManager) {
4. // What functions should this mapper do?
5. **if** (!splitMasterWorker) {
6. **if** ((zkManager != **null**) && zkManager.runsZooKeeper()) {
7. functions = GraphFunctions.ALL;
8. } **else** {
9. functions = GraphFunctions.ALL\_EXCEPT\_ZOOKEEPER;
10. }
11. } **else** {
12. **if** (zkAlreadyProvided) {
13. **int** masterCount = conf.getZooKeeperServerCount();
14. **if** (taskPartition < masterCount) {
15. functions = GraphFunctions.MASTER\_ONLY;
16. } **else** {
17. functions = GraphFunctions.WORKER\_ONLY;
18. }
19. } **else** {
20. **if** ((zkManager != **null**) && zkManager.runsZooKeeper()) {
21. functions = GraphFunctions.MASTER\_ZOOKEEPER\_ONLY;
22. } **else** {
23. functions = GraphFunctions.WORKER\_ONLY;
24. }
25. }
26. }
27. **return** functions;
28. }

    默认的，Giraph会区分master和worker。会在master上面启动zookeeper服务，不会在worker上启动ZooKeeper服务。那么Task 0 就是master+ZooKeeper，其他Tasks就是workers。