八月进展:1.完成了对giraph源码的构建

2.能单机运行SSSP的例子

3.以及向已有jar包里添加文件的方法

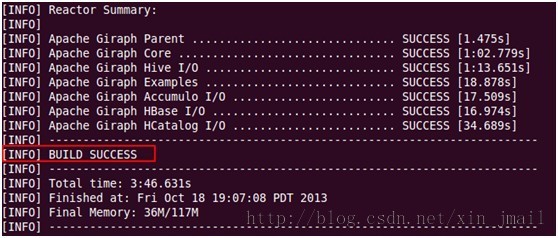
关于Giraph 源码的配置与调试

1. 官网上下载的giraph-1.0.0，放在 /home/fxh/hadoop-1.2.1目录下。
2. 进入上述目录用Maven对源码进行构建(具体Maven的作用本次日志不作记录)

命令:  mvn package -DskipTests

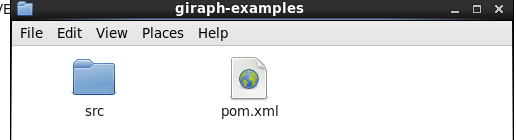
作用目录就在/home/fxh/hadoo-1.2.1/giraph-1.0.0目录下执行上述命令

成功信息如下(由于当时构建的时候忘记截图了，所以为了不影响现有工程所以在网上找了张成功后的图片,源码构建成功的样子与该图大同小异)

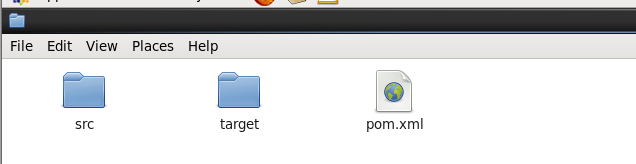


1. 发现构建好的源码在文件中被打了包，并且在每个源码目录下多了个target目录(此目录即maven对源码进行构建后新建目录，包括打包的源码都在这之下)

打包前：



打包后：



其中pom.xml文件是maven构建源码时的配置文件

1. 现在可以测试文件了, 测试giraph包是否构建成功，运行giraph 例子(单机运行SSSP)

进入 /home/fxh/Hadoop-1.2.1/giraph-1.0.0/giraph-examples/target目录，运行命令：

hadoop jar /home/fxh/hadoop-1.2.1/giraph-1.0.0/giraph-examples/target/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 /in/tiny\_graph.txt -of org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /outShortest -w 2注:

其中hadoop jar …是引用后面的jar包

紧接着org.apache.giraph.GiraphRunner是运行这个文件

然后运行org.apache.giraph.examples.SimpleShortestPathsVertex

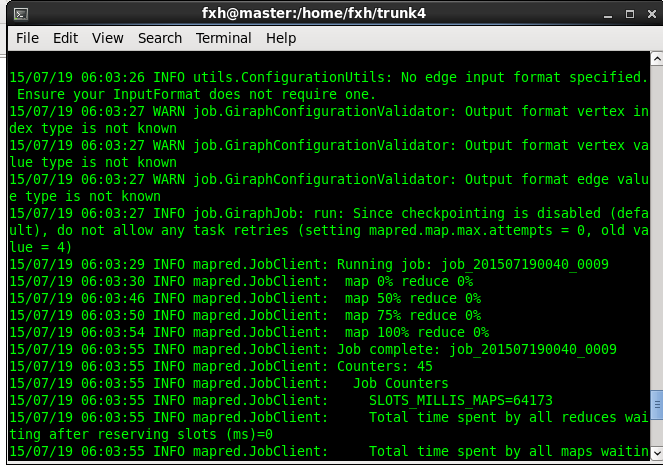
-vif …. 是指输入的格式(此时这里的格式是:org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat)

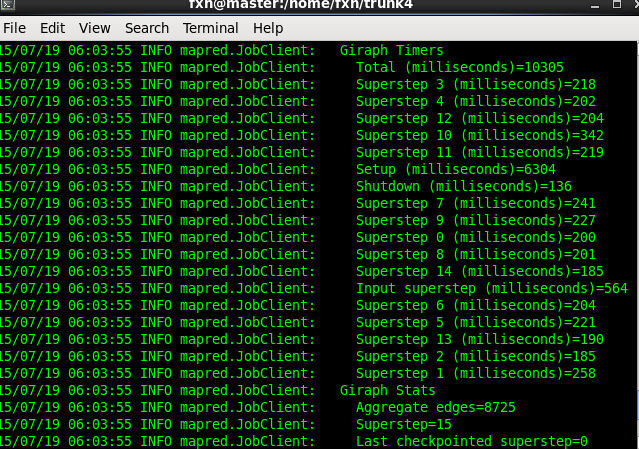
-vip … 是指输出的路径(当然此时的路径是指在HDFS上面的路径)

同理-of 是指输出的格式

-op 是指输出的路径当启用hadoop的时候是上述的路径是指在HDFS上的路径

运行成功结果如下图所示



将giraph-1.0.0-src\giraph-core\src\main\java

样例输入文件(最短路径)：

[0,0,[[1,19],[4,14],[6,18]]]

[1,0,[[0,19],[2,5],[3,7],[4,12]]]

[2,0,[[1,5],[3,3]]]

[3,0,[[1,7],[2,3],[4,8],[5,21]]]

[4,0,[[0,14],[1,12],[3,8],[6,16]]]

[5,0,[[3,21],[6,27]]]

[6,0,[[0,18],[4,16],[5,27]]]

上述格式就是输入的是一张7个点的图

应该输出的结果：

6 18.0

5 43.0

0 0.0

2 24.0

1 19.0

3 22.0

4 14.0

实际输出的结果：

6 18.0

5 43.0

0 0.0

2 24.0

1 19.0

3 22.0

4 14.0

发现样例输出的结果和实际输出的结果相同，证明此工程在hadoop上成功的运行了

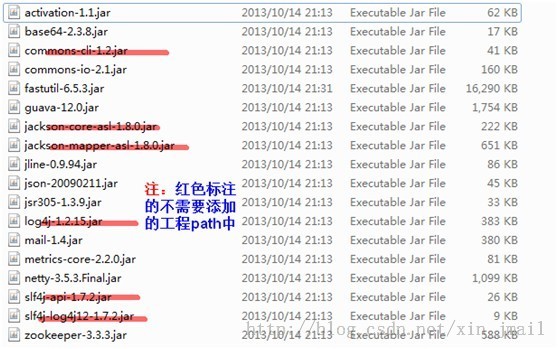
5. 在Eclipse中新建Java工程 Giraph-Debug.

6. 把giraph-1.0.0\giraph-core\src\main\java目录下的org和com文件拷贝到Giraph-Debug工程的src文件夹下。

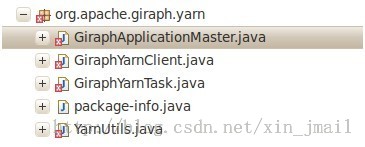
7. 把/home/fxh/hadoop-1.2.1/hadoop-0.20.203.0和/home/fxh/hadoop-1.2.1/hadoop-0.20.203.0/lib（不包含lib里面的文件夹）里面的jar包导入到工程的Path中。

8. 进入/home/fxh/hadoop-1.2.1/giraph-1.0.0/giraph-core/target目录，在giraph-1.0.0-for-hadoop-0.20.203.0-bin.tar.gz上面右击，选择ExtractHere。会解压出giraph-1.0.0-for-hadoop-0.20.203.0文件夹，把里面lib文件夹里的部分jar导入到工程的Path中。

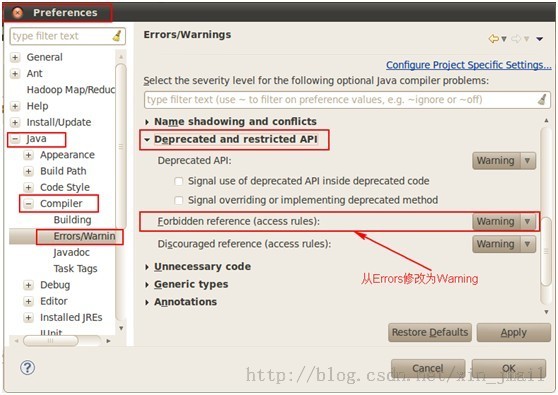
注：因/home/hadoop/giraph-1.0.0/giraph-core/target/giraph-1.0.0-for-hadoop-0.20.203.0/lib和/home/hadoop/hadoop-0.20.203.0/lib中的jar包有重复，因此以Hadoop中的为准，只导入hadoop不包含而giraph包含的jar包。



9. 把工程的src下的org.apache.giraph.yarn包删掉，目前不测试yarn。原因是该包错误较多，删除前是224个错误，删除后只有6个错误。



10. 修改下图的错误，把编译器的级别从Error设置为Warning。



在src下，新建org.apache.commons.net.util包，把从网上下载的Base64.java拷贝进去。只剩3个错误，再注释包含下面3个错误的代码。

/\*if[HADOOP\_1\_SECURITY]

else[HADOOP\_1\_SECURITY]\*/

import org.apache.hadoop.ipc.StandbyException;

/\*end[HADOOP\_1\_SECURITY]\*/

/\*if[HADOOP\_1\_SECRET\_MANAGER]

else[HADOOP\_1\_SECRET\_MANAGER]\*/

try {

secretManager.checkAvailableForRead();

} catch (StandbyException e) {

LOG.error("SaslNettyServer: Could not read secret manager: " + e);

}

/\*end[HADOOP\_1\_SECRET\_MANAGER]\*/

12. 至此，工程中已无错误。配置GiraphRunner.java的运行参数。参数如下：  
org.apache.giraph.examples.SimpleShortestPathsVertex -vif org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat -vip /home/fxh/trunk4/tiny\_graph.txt -of org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /home/fxh/trunk4/output-sssp-debug -w 1  
然后右键->RunAs-> Run on Hadoop.（将运行参数配置在eclipse中）

13.然后运行工程出现以下的问题：

发现是zookeeper没有启动,这里我用的方法是使用外部的zookeeper

具体的安装方法参考

<http://coolxing.iteye.com/blog/1871009>

14：由于将工程文件导入到eclipse中本身就只是作调试和修改用的具体运行的话还是应当在hadoop上运行所以接下来介绍怎么将修改的代码放入包中

暂时只有一种就是将在eclipse上修改后的java文件保存在Giraph-Debug的src文件夹下找到该java文件的.class文件复制然后打入响应giraph文件夹下被打包的文件下

我举个例子

比如修改的文件是Giraph-Debug下的bin/org/apache/giraph/examples/SimpleShortestPathsVertex.class文件

你只需要将这个文件复制

然后放在{GIRAPH\_HOME}/giraph-examples/target/giraph-examples-1.0.0-for-hadoop-0.20.203.0-jar-with-dependencies.jar包中响应位置覆盖在就行了