天河Hadoop+Spark集群

1. 增加一个登录节点login1节点，该节点为2核4G内存的Ubuntu1604 desktop系统，IP为10.186.28.100，公网IP为114.67.40.28，可通过登录节点login1进行登录其他节点，均已经开启免密ssh登录。

可以通过login1节点查看hadoop集群和spark集群的web页面。

hadoop使用默认端口8088和50070。sparks使用默认端口8080和4040.

[https://namenode1:8088](https://namenode1:8088/)

[https://namenode1:50070](https://namenode1:50070/)

[https://namenode1:8080](https://namenode1:8080/)

[https://namenode1:4040](https://namenode1:4040/)

SparkURL spark://namenode1:7077

REST URL spark://namenode1:6066

1. 软件信息如下

hadoop-2.7.3.tar.gz

jdk-8u161-linux-x64.tar.gz

zookeeper-3.4.10.tar.gz

scala-2.11.12.tgz

spark-2.2.1-bin-hadoop2.7.tgz

flink-1.5.5\_scala\_2.11

1. 集群规划：

|  |  |  |  |
| --- | --- | --- | --- |
| 主机名 | IP/公网IP | 安装的软件 | 运行的进程 |
| Login1  2核/4G内存/100G外存 | 10.186.28.100  114.67.40.28 | Ubuntu1604 desktop | 挂接350G云盘 |
| namenode1  24核/64G内存/100G外存 | 10.186.28.101  114.67.37.2 | jdk,hadoop,spark | NameNode、Master、DFSZKFailoverController(zkfc)、ResourceManager |
| namenode2  24核/32G内存/100G外存 | 10.186.28.120  114.67.37.2 | jdk, hadoop,mysql | NameNode、DFSZKFailoverController(zkfc)、ResourceManager |
| workernode1  8核/16G内存/100G外存 | 10.186.28.136 | jdk,hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode2  8核/16G内存/100G外存 | 10.186.28.121 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode3  8核/16G内存/100G外存 | 10.186.28.124 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode4  8核/16G内存/100G外存 | 10.186.28.126 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode5  8核/16G内存/100G外存 | 10.186.28.123 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode6  8核/16G内存/100G外存 | 10.186.28.129 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode7  8核/16G内存/100G外存 | 10.186.28.125 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode8  8核/16G内存/100G外存 | 10.186.28.135 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode9 8核/16G内存/100G外存 | 10.186.28.134 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode10  8核/16G内存/100G外存 | 10.186.28.131 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode11  8核/16G内存/100G外存 | 10.186.28.132 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode12  8核/16G内存/100G外存 | 10.186.28.128 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode13  8核/16G内存/100G外存 | 10.186.28.127 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode14  8核/16G内存/100G外存 | 10.186.28.133 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |
| workernode15  8核/16G内存/100G外存 | 10.186.28.130 | jdk, hadoop,spark,  zookeeper | DataNode、Worker、NodeManager、JournalNode、QuorumPeerMain |

说明：

本hadoop集群由两个NameNode组成，一个处于active状态（10.186.28.101），另一个处于standby状态（10.186.28.120）。

Active NameNode对外提供服务，而Standby NameNode则不对外提供服务，仅同步active namenode的状态。

这里配置一个zookeeper集群，用于ZKFC（DFSZKFailoverController）故障转移，当Active NameNode挂掉了，会自动切换Standby NameNode为standby状态。

有两个ResourceManager，一个是Active（10.186.28.101），一个是Standby（10.186.28.120），状态由zookeeper进行协调。

Ubuntu1604虚拟机root初始密码为Liugang#2018。

1. 外网映射

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 外网ip | 外网端口 | 内网ip | 内网端口 | 说明 |
| 114.67.37.2 | tcp-10680 | 10.186.28.101 | 22 | SSH |
| 114.67.37.2 | tcp-10681 | 10.186.28.101 | 7077 | spark:// |
| 114.67.37.2 | tcp-10682 | 10.186.28.101 | 5005 | spark debug |
| 114.67.37.2 | tcp-10683 | 10.186.28.120 | 22 | SSH |
| 114.67.37.2 | tcp-10684 | 10.186.28.120 | 3306 | MySQL |

1. 系统版本为Ubuntu 16.04 LTS 64bit，服务器版

login1节点为Ubuntu 16.04 LTS桌面版，可以通过login1节点查看hadoop集群和spark集群的web页面，hadoop使用默认端口8088和50070，spark使用默认端口8080和4040。

1. 集群安装目录如下：

export JAVA\_HOME=/usr/local/jdk1.8.0\_161

export ZOOKEEPER\_HOME=/usr/local/zookeeper-3.4.10

export HADOOP\_HOME=/usr/local/hadoop-2.7.3

export SCALA\_HOME=/usr/local/scala-2.11.12

export SPARK\_HOME=/usr/local/spark-2.2.1

export FLINK\_HOME=/usr/local/flink-1.5.5

1. 数据节点每个节点挂载150GB的IPSAN存储，15个数据节点，目前HDFS总大小为2.2TB 。

root@namenode1:~# hadoop fs -df -h

Filesystem Size Used Available Use%

hdfs://geosciences 2.2 T 620.9 M 2.2 T 0%

root@namenode1:~#

1. Namenode2的配置为24核vCPU｜64GB内存｜100GB存储。这台namenode2服务器安装MySQL。MySSQL的用户密码为root/123456。
2. spark的运行模式是spark on yarn。
3. hdfs使用IPSAN网络挂盘的分布式存储，使用15个150GB的SAN（UNIX），具体信息如下

|  |  |  |
| --- | --- | --- |
| IPSAN挂载位置位置 | 对应的节点 | 挂载点路径 |
| 10.222.1.14:/vol/geosci\_lg\_1 | workernode1 | /usr/local/data |
| 10.222.1.14:/vol/geosci\_lg\_2 | workernode2 | /usr/local/data |
| 10.222.1.14:/vol/geosci\_lg\_3 | workernode3 | /usr/local/data |
| 10.222.1.14:/vol/geosci\_lg\_4 | workernode4 | /usr/local/data |
| 10.222.1.14:/vol/geosci\_lg\_5 | workernode5 | /usr/local/data |
| 10.222.1.14:/vol/geosci\_lg\_6 | workernode6 | /usr/local/data |
| 10.222.1.14:/vol/geosci\_lg\_7 | workernode7 | /usr/local/data |
| 10.222.1.14:/vol/geosci\_lg\_8 | workernode8 | /usr/local/data |
| 10.222.1.15:/vol/geosci\_lg\_9 | workernode9 | /usr/local/data |
| 10.222.1.15:/vol/geosci\_lg\_10 | workernode10 | /usr/local/data |
| 10.222.1.15:/vol/geosci\_lg\_11 | workernode11 | /usr/local/data |
| 10.222.1.15:/vol/geosci\_lg\_12 | workernode12 | /usr/local/data |
| 10.222.1.15:/vol/geosci\_lg\_13 | workernode13 | /usr/local/data |
| 10.222.1.15:/vol/geosci\_lg\_14 | workernode14 | /usr/local/data |
| 10.222.1.15:/vol/geosci\_lg\_15 | workernode15 | /usr/local/data |

1. 控制台账号信息如下：

web登录地址：http://cloud.nscc-gz.cn/ （建议使用Chrome浏览器）

登录用户名：geoscience-lg

登录密码：Liugang#2018

1109642384

1. Login1节点上安装了vsftpd，提供了FTP服务。数据放在/data下。远程以root登陆FTP服务器，可以访问该节点上的任何目录信息，需要进行安全设置。

sftp://114.67.40.28

root/Liugang#2018

port:22

1. 为了共享login1下的/data目录，安装samba，共享/data节点。
2. 下载了geotools-19.0-bin.zip，将其中的jar文件合并到了namenode1:/usr/local/spark-2.2.1/jars中，合并的规则是保留原有文件，并且删除了同名低版本文件。这样spark在打包jar文件夹的时候就会将geotools的所有jar文件也打包到workernode上。
3. flink-1.5.5 -scala11安装教程

参考：

<https://ci.apache.org/projects/flink/flink-docs-release-1.6/ops/deployment/cluster_setup.html#configuring-flink>

1. 下载解压安装包

登录到namenode1，运行命令：

root@namenode1:~# wget [http://mirrors.hust.edu.cn/apache/flink/flink-1.5.5/flink-1.5.5-bin-scala\_2.11.tgz](http://mirrors.hust.edu.cn/apache/flink/flink-1.6.0/flink-1.6.0-bin-hadoop27-scala_2.11.tgz)

这个命令将flink的安装包下载到root用户目录下，然后解压到指定安装目录：

root@namenode1:~# tar -zxvf flink-1.5.5-bin-scala\_2.11.tgz -C /usr/local/

1. 配置环境变量

编辑profile文件

root@namenode1:~# vim /etc/profile

添加

export FLINK\_HOME=/usr/local/flink-1.5.5

并将bin目录配置

:$FLINK\_HOME/bin

1. 配置集群节点

进入安装目录的conf目录下

root@workernode1:~/flink-1.5.5# cd conf/

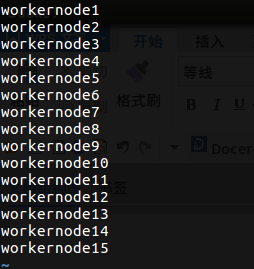
编辑flink-conf.yaml文件

root@workernode1:~/flink-1.5.5/conf# vim flink-conf.yaml

将jobmanager.rpc.address = localhost改为namenode1

编辑slaves文件将所有节点的ip/hostname添加，每个占一行

root@workernode1:~/flink-1.5.5/conf# vim slaves



1. 分发

通过scp命令将namenode1的整个flink文件拷贝到所有节点**对应的目录下**：

root@namenode1:/usr/local# scp -r root@namenode1:/usr/local/flink-1.5.5/ root@workernode1:/usr/local/

root@namenode1:/usr/local# scp -r root@namenode1:/usr/local/flink-1.5.5/ root@workernode2:/usr/local/

root@namenode1:/usr/local# scp -r root@namenode1:/usr/local/flink-1.5.5/ root@workernode3:/usr/local/

.......

运行完成可以到对应的节点查看，可以看到有flink-1.5.5这个文件夹。

1. 启动与停止

root@namenode1:/usr/local/flink-1.5.5# bin/start-cluster.sh

root@namenode1:/usr/local/flink-1.5.5# bin/stop-cluster.sh

1. flink-1.6.0-with hadoop27-scala11安装教程

参考：

<https://ci.apache.org/projects/flink/flink-docs-release-1.6/ops/deployment/cluster_setup.html#configuring-flink>

1. 下载解压安装包

登录到namenode1，运行命令：

root@namenode1:~# wget <http://mirrors.hust.edu.cn/apache/flink/flink-1.6.0/flink-1.6.0-bin-hadoop27-scala_2.11.tgz>

这个命令将flink的安装包下载到root用户目录下，然后解压到指定安装目录：

root@namenode1:~# tar -zxvf flink-1.6.0-bin-hadoop27-scala\_2.11.tgz -C /usr/local/

1. 配置环境变量

编辑profile文件

root@namenode1:~# vim /etc/profile

添加

export FLINK\_HOME=/usr/local/flink-1.6.0

并将bin目录配置

:$FLINK\_HOME/bin

1. 配置集群节点

进入安装目录的conf目录下

root@workernode1:~/flink-1.4.2# cd conf/

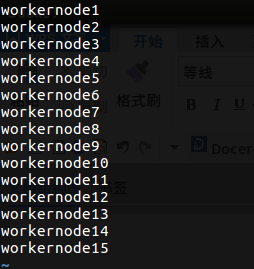
编辑flink-conf.yaml文件

root@workernode1:~/flink-1.4.2/conf# vim flink-conf.yaml

将jobmanager.rpc.address = localhost改为namenode1

编辑slves文件将所有节点的ip/hostname添加，每个占一行

root@workernode1:~/flink-1.4.2/conf# vim slaves



1. 分发

通过scp命令将namenode1的整个flink文件拷贝到所有节点**对应的目录下**：

root@namenode1:/usr/local# scp -r root@namenode1:/usr/local/flink-1.6.0/ root@workernode1:/usr/local/

root@namenode1:/usr/local# scp -r root@namenode1:/usr/local/flink-1.6.0/ root@workernode2:/usr/local/

root@namenode1:/usr/local# scp -r root@namenode1:/usr/local/flink-1.6.0/ root@workernode3:/usr/local/

.......

运行完成可以到对应的节点查看，可以看到有flink-1.6.0这个文件夹

1. 启动

root@namenode1:/usr/local/flink-1.6.0# bin/start-cluster.sh

1. 通过FTP将数据上载到login1的/data/spatialhadoop
2. Login1 TeamViewer 1109642384/euw678
3. namenode2安装远程远程桌面(没有安装，参考下面连接)

<https://blog.csdn.net/szw4090416/article/details/52859318>

1. namenode1挂接login1下的/data共享目录。
2. 在namenode1上将，

login1节点下的/data/spatialhadoop下的数据

上载到hdfs://geosciences/data/spatialhadoop目录下。在每个数据集中添加一个schema结尾的文件，例如：counties/county\_small.tsv，在同一个目录下添加一个文件 county\_small.tsv.schema,该文件的规则是添加一个要素类名称，后续以\t分割，把WKT变成WKT%POLYGON，表示几何类型为POLYGON，后面的为属性字段名称。

处理过的数据在namenode1下/data/gtl目录。里面主要是tsv和ssv格式数据。csv格式的数据格式不正确，其实是ssv格式的。

1. 运行Hadoop版本的WordCount
2. 运行spark版本的WordCount
3. 运行beam版本的WordCount

1）建立工程 root@namenode1:/gtl/wordcount/beam# mvn archetype:generate \

-DarchetypeGroupId=org.apache.beam \

-DarchetypeArtifactId=beam-sdks-java-maven-archetypes-examples \

-DarchetypeVersion=2.8.0 \

-DgroupId=org.example \

-DartifactId=word-count-beam \

-Dversion="0.1" \

-Dpackage=org.apache.beam.examples \

-DinteractiveMode=false

2）打包，并在Flink-Cluster上执行 root@namenode1:/gtl/wordcount/beam/word-count-beam# mvn package exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \

-Dexec.args="--runner=FlinkRunner --flinkMaster=namenode1 --filesToStage=target/word-count-beam-bundled-0.1.jar \

--inputFile:/gtl/wordcount/beam/word-count-beam /pom.xml --output=:/gtl/wordcount/beam/word-count-beam /tmp/counts" -Pflink-runner

$FLINK\_HOME/bin/flink run -c org.apache.beam.examples.WordCount target/word-count-beam-bundled-0.1.jar --runner=FlinkRunner --flinkMaster=namenode1 --filesToStage=target/word-count-beam-bundled-0.1.jar --inputFile=/gtl/wordcount/beam/word-count-beam/pom.xml --output=/gtl/wordcount/beam/word-count-beam/tmp/counts

找不到文件。

3）打包，并在Spark上执行 root@namenode1:/gtl/wordcount/beam/word-count-beam# mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount -Dexec.args="--runner=SparkRunner --inputFile=pom.xml --output=counts" -Pspark-runner

$ mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \

-Dexec.args="--inputFile=pom.xml --output=counts" -Pdirect-runner

$ mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \

-Dexec.args="--inputFile=pom.xml --output=counts --runner=ApexRunner" -Papex-runner

$ mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \

-Dexec.args="--runner=FlinkRunner --inputFile=pom.xml --output=counts" -Pflink-runner

You can monitor the running job by visiting the Flink dashboard at http://<flink master>:8081

$ mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \

-Dexec.args="--runner=SparkRunner --inputFile=pom.xml --output=counts" -Pspark-runner

Make sure you complete the setup steps at /documentation/runners/dataflow/#setup

$ mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \

-Dexec.args="--runner=DataflowRunner --project=<your-gcp-project> \

--gcpTempLocation=gs://<your-gcs-bucket>/tmp \

--inputFile=gs://apache-beam-samples/shakespeare/\* --output=gs://<your-gcs-bucket>/counts" \

-Pdataflow-runner

$ mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.WordCount \

-Dexec.args="--inputFile=pom.xml --output=/tmp/counts --runner=SamzaRunner" -Psamza-runner

1. 运行 gtl.beam.app.Sampler ，测试spark和flink两种集群，并测试HDFS数据加载。

*1）*flink-local 运行通过。

*root@namenode1:/gtl/geobeam# mvn package exec:java -Dexec.mainClass=gtl.beam.app.Sampler -Dexec.args="--inputFile=/gtl****/gtl/data/dat/counties/county\_small.tsv*** *--outputFile=/gtl****/gtl/data/dat/counties/county\_small.sampler*** *--sampleNumber=2 --runner=FlinkRunner" -Pflink-runner*

*2）*flink-cluster standalone运行通过。

*root@namenode1:/gtl/geobeam#* mvn package exec:java -Dexec.mainClass=gtl.beam.app.Sampler -Dexec.args="--runner=FlinkRunner --flinkMaster=namenode1 --filesToStage=target/geobeam-bundled-1.0-SNAPSHOT.jar --inputFile=/gtl/gtl/data/dat/counties/county\_small.tsv --outputFile=/gtl/gtl/data/dat/counties/county\_small.sampler --sampleNumber=2 " -Pflink-runner

*3)* flink-cluster运行通过。

*root@namenode1:/gtl/geobeam# mvn package exec:java -Dexec.mainClass=gtl.beam.app.Sampler -Dexec.args="--runner=FlinkRunner --flinkMaster=namenode1 --filesToStage=target/geobeam-bundled-1.0-SNAPSHOT.jar --inputFile=/data/gtl/areawater/areawater.ssv --outputFile=/data/gtl/areawater/areawater.ssv.sampler --sampleNumber=2 " -Pflink-runner*

通过采样数据分析原始数据格式，格式为SSV。需要根据数据编写schema文件。生成的sampler文件先传递到login1上

*root@namenode1:/gtl/geobeam# scp /data/gtl/areawater/areawater.ssv.sampler* [*root@login1:/data/gtl/areawater/areawater.ssv.sampler*](mailto:root@login1:/data/gtl/areawater/areawater.ssv.sampler)

然后下载到本地机上传到svn上。

4) flink-cluster运行通过。

*root@namenode1:/gtl/geobeam# mvn package exec:java -Dexec.mainClass=gtl.beam.app.Sampler -Dexec.args="--runner=FlinkRunner --flinkMaster=namenode1 --filesToStage=target/geobeam-bundled-1.0-SNAPSHOT.jar --inputFile=**/data/gtl/edges/edges.ssv --outputFile=/data/gtl/edges/edges.ssv.sampler --sampleNumber=3 " -Pflink-runner*

通过采样数据分析原始数据格式，格式为SSV。需要根据数据编写schema文件。生成的sampler文件先传递到login1上

*root@namenode1:/gtl/geobeam# scp /data/gtl/edges/edges.ssv.sampler* [*root@login1:/data/gtl/edges/edges.ssv.sampler*](mailto:root@login1:/data/gtl/edges/edges.ssv.sampler)

然后下载到本地机上传到svn上。

*root@namenode1:/gtl/geobeam# mvn package exec:java -Dexec.mainClass=gtl.beam.app. Sampler -Dexec.args=" --inputFile=****hdfs://geosciences/data/spatialhadoop/counties/county\_small.tsv*** *--outputFile=****hdfs://geosciences/data/spatialhadoop/counties/county\_small.sampler*** *--smapleNumber=2 --runner=FlinkRunner" -Pflink-runner*

运行提示没有*HDFS*支持，可能需要安装*Hadoop*版本的*Flink*。

*root@namenode1:/gtl/geobeam# $FLINK\_HOME/bin/flink run -c gtl.beam.app.Sampler ./target/geobeam-bundled-1.0-SNAPSHOT.jar --inputFile=/gtl****/gtl/data/dat/counties/county\_small.tsv*** *--outputFile=/gtl****/gtl/data/dat/counties/county\_small.sampler*** *--smapleNumber=2 --runner=FlinkRunner*

运行提示没有输入的文件不存在，可能是需要在每个*workernode*上的同样目录放置同样的数据？但为什么*WorldCount*可以运行？

*root@namenode1:/gtl/geobeam# mvn package exec:java -Dexec.mainClass=gtl.beam.app.Sampler -Dexec.args="--runner=FlinkRunner --flinkMaster=namenode1 --filesToStage=target/geobeam-bundled-1.0-SNAPSHOT.jar --inputFile=/gtl****/gtl/data/dat/counties/county\_small.tsv*** *--outputFile=/gtl****/gtl/data/dat/counties/county\_small.sampler*** *--sampleNumber=2 " -Pflink-runner*

*root@namenode1:/gtl/geobeam# mvn package -Dexec.mainClass=gtl.beam.app.Sampler -Dexec.args=" --inputFile=****hdfs://geosciences/data/spatialhadoop/counties/county\_small.tsv*** *--outputFile=****hdfs://geosciences/data/spatialhadoop/counties/county\_small.sampler*** *--sampleNumber=2 --runner=SparkRunner" -Pspark-runner*

打包通过。

*root@namenode1:/gtl/geobeam#* $SPARK\_HOME/bin/spark-submit --class *gtl.beam.app.Sampler* --master yarn --deploy-mode cluster --driver-memory 4g --executor-memory 2g --executor-cores 1 --queue thequeue *./target/geobeam-bundled-1.0-SNAPSHOT.jar --inputFile=****hdfs://geosciences/data/spatialhadoop/counties/county\_small.tsv*** *--outputFile=****hdfs://geosciences/data/spatialhadoop/counties/county\_small.sampler*** *--sampleNumber=2 --runner=SparkRunner*

执行错误,Yarn。

1. 运行gtl.beam.app.Analyzer，传入参数为HDFS上的文件，hdfs://geosciences/data/spatialhadoop/counties/county\_small.tsv,测试spark和flink两种集群。

*root@namenode1:/gtl/geobeam#* *mvn package exec:java -Dexec.mainClass=gtl.beam.app.Analyzer* *-Dexec.args="--inputFile=/gtl****/gtl/data/dat/counties/county\_small.tsv*** *--outputFile=/gtl****/gtl/data/dat/counties/county\_small.analyzer --sampleNumber=5*** *--runner=FlinkRunner" -Pflink-runner*

运行通过。

mvninstall*-*DskipTests

*root@namenode1:/gtl/geobeam#* *mvn package exec:java -Dexec.mainClass=gtl.beam.app.Analyzer -Dexec.args="--inputFile=/data****/gtl/areawater/areawater.ssv*** *--outputFile=/data****/gtl/areawater/areawater.ssv.analyzer --sampleNumber=2*** *--runner=FlinkRunner" -Pflink-runner*

可能数据格式不对，需要本地调试。测试数据可以写出样本数据进行测试。

1. 运行geobeam的range查询并记录时间

*root@namenode1:/gtl/geobeam# mvn package exec:java -Dexec.mainClass=gtl.beam.app.Querier -Dexec.args="--inputFile=/gtl****/gtl/data/dat/counties/county\_small.tsv*** *--outputFile=/gtl****/gtl/data/dat/counties/county\_small.querier*** *--minX=-130.0 --minY=-100.0 --maxX=30.0 --maxY=50.0 --runner=FlinkRunner" -Pflink-runner*

运行通过。

1. 运行geospark的range查询并记录时间