九月进展:成功测试了一张2200点的图的最短路径的分布式算法

有关SSSP例子的测试:

给定2200个点的数据(蛋白质)，数据的格式如下

ID interactor A ID interactor B

DIP-150N DIP-681N

DIP-785N DIP-372N

DIP-696N DIP-512N

DIP-142N DIP-13N

DIP-760N DIP-962N

DIP-665N DIP-696N

DIP-376N DIP-376N

DIP-800N DIP-376N

DIP-1102N DIP-41N

DIP-365N DIP-1102N

DIP-671N DIP-1104N

DIP-744N DIP-823N

DIP-891N DIP-744N

编写java程序将数据转换成我们想要的格式:

想要的格式如下：

[150,0,[[681,1]]]

[681,0,[[597,1],[5290,1],[3846,1],[2312,1],[1274,1],[150,1]]]

[785,0,[[372,1],[2019,1],[2492,1],[1735,1],[2022,1],[4954,1],[3801,1],[6597,1],[2121,1],[5577,1]]]

[372,0,[[785,1]]]

[696,0,[[2958,1],[700,1],[512,1],[665,1],[507,1],[2434,1],[5748,1]]]

[512,0,[[411,1],[700,1],[2254,1],[512,1],[696,1],[6541,1],[507,1],[417,1],[418,1],[1360,1]]]

[142,0,[[13,1]]]

[13,0,[[911,1],[1461,1],[716,1],[1039,1],[142,1],[1078,1],[1077,1]]]

[760,0,[[747,1],[2210,1],[962,1],[1484,1]]]

[962,0,[[760,1]]]

[665,0,[[1459,1],[18,1],[696,1]]]

[376,0,[[2252,1],[2373,1],[800,1],[695,1],[376,1]]]

[376,0,[[376,1]]]

[800,0,[[2373,1],[376,1]]]

[1102,0,[[1102,1],[5447,1],[365,1],[41,1]]]

[41,0,[[1102,1]]]

[365,0,[[1102,1],[365,1]]]

[671,0,[[1104,1]]]

[1104,0,[[671,1]]]

通过上述的转换后通过分布式运算测试Giraph源码的健壮性

转换上述格式的java程序的源代码如下：

package fdata;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class FormatData {

private static List<MyData> data = new ArrayList<MyData>();

public static void readFileToFormat(String fileName) {

File file = new File(fileName);

BufferedReader bf = null;

try {

bf = new BufferedReader( new FileReader(file) );

String tempString = null;

//略去第一行

bf.readLine();

while( (tempString = bf.readLine() ) != null ) {

String[] temp = tempString.split(" |\t");

//截取关键信息

int key = Integer.parseInt( temp[0].substring(4).substring(0, temp[0].length()-5) );

int inter = Integer.parseInt( temp[1].substring(4).substring(0, temp[1].length()-5) );

//System.out.println(temp[0].substring(4).substring(0, temp[0].length()-5) );

//System.out.println(temp[1].substring(4).substring(0, temp[1].length()-5));

//flag 0表示两个数都未找到，1表示找到KEY，2表示找到INTER，3表示两个都找到了

int flag = 0;

for(MyData tempData : data) {

if(tempData.key == key) {

if (flag == 0) {

flag = 1;

tempData.interactor.add(inter);

}

else if(flag == 2){

flag = 3;

tempData.interactor.add(inter);

break;

}

}

if(tempData.key == inter) {

if (flag == 0) {

flag = 2;

tempData.interactor.add(key);

}

else if(flag == 1){

flag = 3;

tempData.interactor.add(key);

break;

}

}

}

if ( flag == 0) {

data.add(new MyData(key,inter));

data.add(new MyData(inter,key));

}

else if ( flag == 1 ) {

data.add(new MyData(inter,key));

}

else if ( flag == 2 ) {

data.add(new MyData(key,inter));

}

}

}

catch (IOException e) {

e.printStackTrace();

}

finally {

if (bf != null) {

try {

bf.close();

}

catch (IOException e1){}

}

}

}

public static void writeFileToFormat(String fileName) {

File file = new File(fileName);

BufferedWriter br = null;

try {

br = new BufferedWriter(new FileWriter(file) );

for(MyData tempData : data) {

br.write(tempData.toString());

br.newLine();

}

}

catch (IOException e) {

e.printStackTrace();

}

finally {

if (br != null) {

try {

br.close();

}

catch (IOException e1){}

}

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

readFileToFormat("TestData.txt");

writeFileToFormat("FormatData.txt");

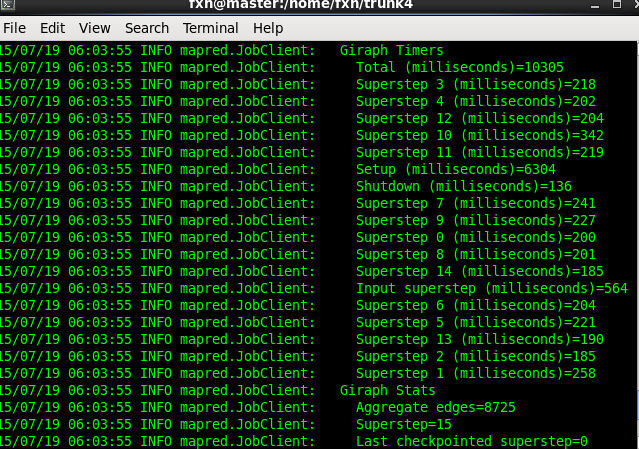
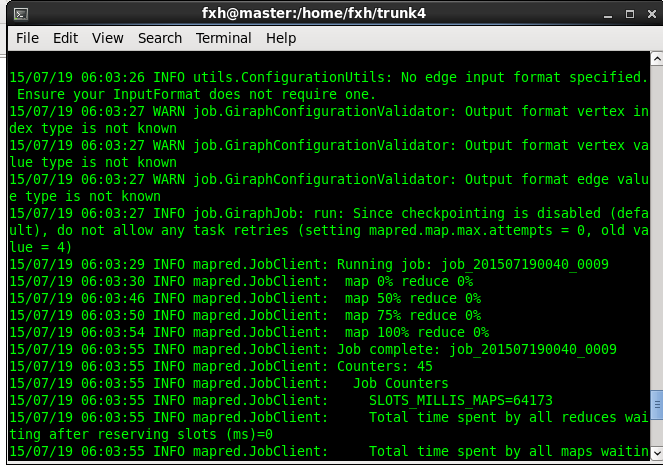
}

}

下面开始测试:

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/FormatData.txt -of org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /outShortest -w 4

以下为Giraph程序测试截图:



跑得的结果部分展示：150 0.0

3846 2.0

2038 1.7976931348623157E308

1654 7.0

2278 7.0

1478 6.0

4246 9.0

1414 8.0

1718 1.7976931348623157E308

838 7.0

2022 7.0

5910 7.0

4790 8.0

8870 10.0

5750 8.0

7670 7.0

774 6.0

758 7.0

1430 1.7976931348623157E308

4294 7.0

1670 7.0

1190 8.0

2310 8.0