SPARK 上机实验

环境：一台master和7台slave服务器，每台服务器：

- Processor: 8cores
- RAM: 32GB
- Address:
  - hd-master: 183.174.61.31
  - hd-slave1-7: 183.174.61.32-38

Graph data：来自google programming contest 2002。有向图，每一个节点代表一个网页，每一条有向边代表一个超链接。

- Nodes: 875713
  - `/opt/linuxsir/spark/data/graphx/user.txt`
- Edges: 5105039
  - `/opt/linuxsir/spark/data/graphx/test.txt` ，数据集每一行为两个数字，中间以空格分割，代表从第一个节点到第二个节点的一条有向边

实验目标：使用spark-shell，借助spark完成简单的pagerank计算，同时熟悉使用scala编写代码。

---

01. 启用环境&登陆服务器
    - 在实验前助教会先使用root权限启动相关环境。
    - 学生首先登陆到master服务器上，学生用户的账号和密码均为std+学号(例:**std2018202113**)。

```
ssh std2018202113@183.174.61.31
pwd:std2018202113
```

02. 打开spark-shell完成pagerank实验

- 打开spark-shell以及确定相关参数

```
cd /opt/linuxsir/spark/bin
MASTER=spark://hd-master:17077 ./spark-shell --total-executor-cores
8 --executor-cores 2 --executor-memory 2g
```

其中，你可以尝试着改小total-executor-cores以及executor-cores的数量，观察web ui并体会运行时间的差别。**!!因为服务器资源有限,所以请不要过度调大参数导致别人的资源被占用。!!**

- 使用scala完成pagerank

在打开spark-shell之后，你会得到两个web ui，可在浏览器中打开:

  - 观察当前的application运行状况：http://183.174.61.31:4050 (这个会发生变化，以spark给你的为准)
  - 观察所有的worker以及application：http://183.174.61.31:17077

接下来可以开始编写代码。

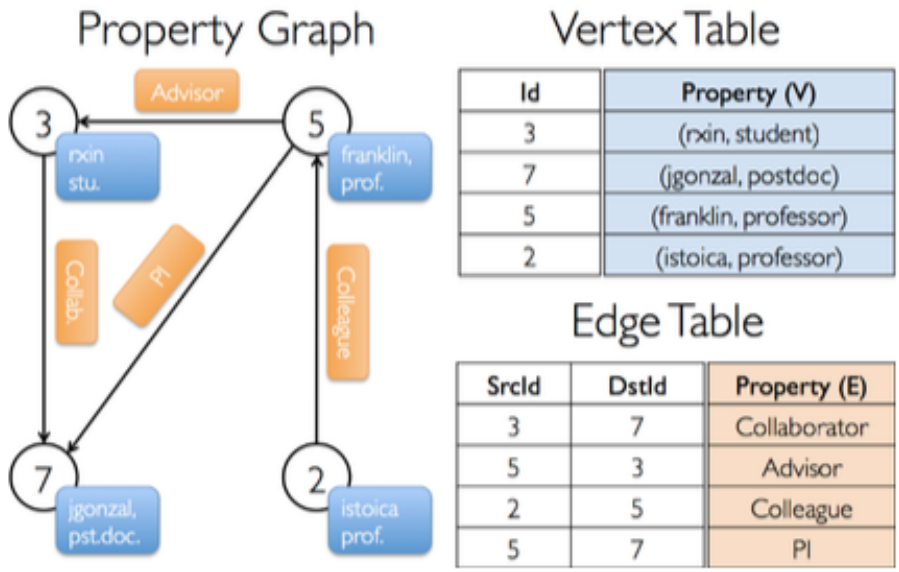
在 `/opt/linuxsir/spark/data/graphx` 目录下除了原有的 `followers.txt` ，`users.txt` ，新加入图文件为 `test.txt` ，以及对应的点名称 `user.txt` 。示例代码如下：

```
import org.apache.spark.graphx.GraphLoader
```

```
// load graph
val graph = GraphLoader.edgeListFile(sc,
"file:///opt/linuxsir/spark/data/graphx/test.txt")

// run pagerank
val ranks = graph.pageRank(0.0001).vertices

// Join the ranks with the usernames
val users =
sc.textFile("file:///opt/linuxsir/spark/data/graphx/user.txt").map
{ line =>
val fields = line.split(",")
(fields(0).toLong, fields(1)) }
val ranksByUsername = users.join(ranks).map { case (id, (username,
rank)) => (username, rank)
}

// Print first 10 ranksByUsername result
println(ranksByUsername.collect().slice(0, 10).mkString("\n"))
```

在这个基础上，你还可以使用scala对pagerank的结果进行排序，并把结果输出到文件中。（没有示例，自己探索）

- 其他的可尝试代码：



Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

建立上面的属性图并完成简单的查询。

```
import org.apache.spark.graphx.{Edge, Graph, VertexId} import
org.apache.spark.rdd.RDD
import org.apache.spark.{SparkConf, SparkContext}
val conf = new SparkConf().setAppName("MyGraphX")
// val sc = new SparkContext(conf)
// Assume the SparkContext has already been constructed

// Create an RDD for the vertices
val users: RDD[(VertexId, (String, String))] =
sc.parallelize(Array((3L, ("rxin", "student")), (7L, ("jgonzal",
"postdoc")), (5L, ("franklin", "prof")), (2L, ("istoica",
"prof"))))

// Create an RDD for edges
val relationships: RDD[Edge[String]] =
sc.parallelize(Array(Edge(3L, 7L, "collab"), Edge(5L, 3L,
"advisor"), Edge(2L, 5L, "colleague"), Edge(5L, 7L, "pi")))

// Define a default user in case there are relationship with
missing user
val defaultUser = ("John Doe", "Missing")

// Build the initial Graph
val graph = Graph(users, relationships, defaultUser)

// Count all users which are postdocs
val verticesCount = graph.vertices.filter { case (id, (name, pos))
=> pos == "postdoc" }.count println(verticesCount)
// Count all the edges where src > dst
val edgeCount = graph.edges.filter(e => e.srcId > e.dstId).count
println(edgeCount)
// Count dst is 7
val edgeCount = graph.edges.filter(e => e.dstId == 7L ).count
println(edgeCount )
// show edges who dst is 7
val someEdges = graph.edges.filter(e => e.dstId == 7L )
someEdges.glom().collect()
```

- 退出spark-shell

  :quit

---

03. 存在的一些问题

spark最多只会使用其中3个executor，这个不知道是因为配置文件还是spark本身的调度问题。