Hadoop与Spark入门(run GraphX)

覃雄派

Hadoop与Spark入门
(run GraphX)

提纲

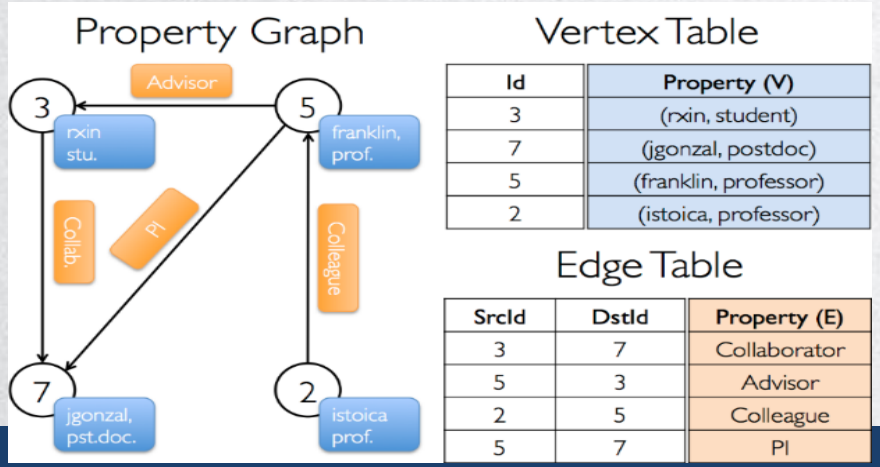
- GraphX简介
- 属性图实例
- Spark部署图
- 运行Page Rank程序

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - GraphX是Spark的一个组件，用于对图数据进行分析；GraphX对Spark的RDD数据集进行了扩展，用来描述图数据
  - Spark的图，是一种有向的属性图，即可以给顶点和边设置属性
  - 为了支持图数据的处理和分析，GraphX实现了一系列基本操作符(Operator)，包括子图(SubGraph)、顶点连接(JoinVertices)、以及消息聚集等(AggregateMessages)
  - 此外，GraphX还提供了Pregel API的变种，方便用户编程
  - 在此基础上，GraphX已经实现了一批图数据处理算法，以加快用户开发图数据处理软件

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - GraphX的属性图是一种有向图，它的顶点和边被赋予了各种属性(标签)；下图展示了一个简单的属性图
  - 比如id为3的顶点有两个标签，即rxin和student，而顶点5有两个标签，即franklin和professor，从顶点5到顶点7有一条边，这条边有一个标签，即PI
  - 这个图，描述了若干教授、博士后、学生之间的导师、同事、合作、项目负责人(PI即Principal Investigator)等关系



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

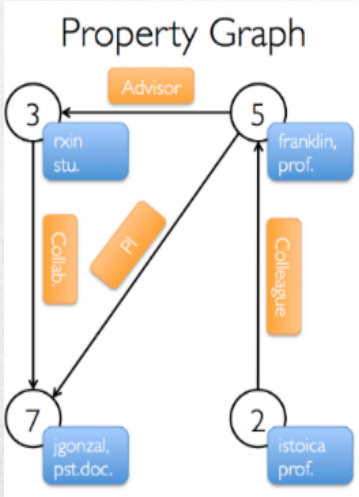
Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - 属性图实例代码
  - 可以使用如下scala代码建立这个属性图

```
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)
```



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

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - 属性图实例代码
  - 属性图建立好以后，可以对顶点进行查询，对边进行查询
  - 下面的代码把第二个顶点属性为postdoc的顶点查出来，把source id大于dest id的边查出来，最后把dest id为7的边查出来

https://www.cnblogs.com/chenmingjun/p/10797753.html

```
// 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()
```

Hadoop与Spark入门(run GraphX)

- Spark 部署图
  - (1)windows安装VMware
  - (2)VMware开辟3台虚拟机，别名和IP地址分别是
    - hd-master 192.168.31.129
    - hd-slave1 192.168.31.130
    - hd-slave2 192.168.31.131

如果助教部署配置有变，以其部署配置为准

Spark 层	(1) beeline 连接到 thrift server (2) thrift server 通过 metastore 查阅 MySQL 里的元信息 (3) thriftserver 把 SQL 查询交给 Spark 运行 (Master/Slave) (4) Spark 可以存取本地文件，也可以存取 HDFS Master	 Worker	 Worker	
Hive 层	metastore MySQL			
YARN 层	ResourceManager	NodeManager	NodeManager	…
HDFS 层	NameNode Secondary NameNode	DataNode	DataNode	…
Hardware 各个节点	hd-master 节点 192.168.31.129	hd-slave1 节点 192.168.31.130	hd-slave2 节点 192.168.31.131	…

Spark 和 Hadoop 的进程及其调用关系

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - 这段代码是用scala编写的，可以用spark shell(scala)来运行
  - 启动spark shell的命令为

```
cd /opt/linuxsir/spark/bin MASTER=spark://hd-master:17077 ./spark-shell

\\退出用:quit
```

  - 注意首先启动HDFS、YARN、 Hive metastore、Spark

```
在192.168.31.129节点上
\\ 启动hdfs和yarn
cd /opt/linuxsir/hadoop/sbin
./start-dfs.sh
./start-yarn.sh

\\ 启动Hive metastore
cd /opt/linuxsir/hive
./bin/hive --service metastore -p 19083
&

\\ 接着启动spark
cd /opt/linuxsir/spark
./sbin/start-all.sh
```

如果助教发布最新实验指导书，请以其为准

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - 在Spark软件包的/opt/linuxsir/spark/data/graphx目录下，有users.txt以及follower.txt等两个文件，分别表示用户和用户的关系
  - 这两个文件具有特定的格式
  - users.txt文件的内容如下；每一行表示一个节点，首先是节点编号，接着是节点的名称，最后是节点的描述

```
1,BarackObama,Barack Obama
2,ladygaga,Goddess of Love
3,jeresig,John Resig
4,justinbieber,Justin Bieber
6,matei_zaharia,Matei Zaharia
7,odersky,Martin Odersky
8,anonsys
```

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - follower.txt文件的内容如下
  - 每一行给出一条边，用开始节点编号和结束节点编号表示

```
2 1
4 1
1 2
6 3
7 3
7 6
6 7
3 7
```

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - 可以在这个数据集上运行PageRank算法，计算每个用户的PageRank值
  - PageRank的示例代码，用Scala语言缩写
  - 首先，导入必要的类
  - 装载数据集，把边表(即关系表)装载进来，创建Graph

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

// Load the edges as a graph
val graph = GraphLoader.edgeListFile(sc, "file:///opt/linuxsir/spark/data/graphx/followers.txt")
```

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - 运行PageRank算法

```
// Run PageRank
val ranks = graph.pageRank(0.0001).vertices
```

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - 把算出的PageRank值，和User数据做关联
  - 以便显示每个用户的PageRank，而不是把用户的编号显示出来

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

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - 显示结果

```
// Print the result
println(ranksByUsername.collect().mkString("\n"))
```

  - 输出结果如下

```
(justinbieber,0.15)
(matei_zaharia,0.7013599933629602)
(ladygaga,1.390049198216498)
(BarackObama,1.4588814096664682)
(jeresig,0.9993442038507723)
(odersky,1.2973176314422592)
```

Hadoop与Spark入门(run GraphX)

- Spark GraphX简介
  - Page Rank
  - 这段代码是用scala编写的，可以用spark shell(scala)来运行
  - 执行完Page rank程序后
  - 关闭软件的顺序为退出spark-shell
  - 关闭Spark、Hive metastore、YARN、HDFS

启动spark shell的命令（参考前文）

与启动顺序相反，启动顺序为HDFS、YARN、Hive metastore、以及Spark

在192.168.31.129节点上

\\停止spark
cd /opt/linuxsir/spark
./sbin/stop-all.sh

\\ 停止Hive metastore
netstat -ntlp|grep 19083
kill -9 16336 \\16336是进程号

\\停止hadoop hdfs/yarn
cd /opt/linuxsir/hadoop/sbin
./stop-yarn.sh
./stop-dfs.sh

Hadoop与Spark入门