实验一：Batch job in cloud

UNIkeEN

Part A：认识 Spark 与 Hadoop

Hadoop 与 Spark 的配置运行结果分别见附录图 1、图 2。

运行 Hadoop Wordcount

输入文件内容为《Something Like This》歌词，运行结果见附录图 3。

运行 Spark Connected Component Example

运行结果见附录图 4。

华为云 ECS 是一种 IaaS

我认为华为云 ECS 是一种 IaaS，它提供了基本的计算基础设施——如虚拟机、存储空间与网络；但除操作系统外提供的环境比较有限：编程语言、数据库与应用程序等环境需要用户自己配置，故属于 IaaS。

Hadoop 与 Spark 的区别

- 基本架构
  - Hadoop 包含分布式文件系统（HDFS）的 MapReduce 模型，本身即可完成大数据存储与分析
  - Spark 不提供文件系统，专门处理分布式存储的大数据，需要借助 HDFS 或其他文件系统存储。
- 性能
  - Hadoop 的 MapReduce 模型在每步操作后都将数据写回磁盘，这导致了较慢的数据处理速度。
  - Spark：从磁盘中读取数据，中间数据全部存储在内存中，减少了磁盘 I/O 操作，通常比 Hadoop 快。
- 容错性
  - Hadoop 将每次处理的数据都写入磁盘，在 HDFS 上实现高容错性。
  - Spark 也使用了高效的容错机制，如弹性分布式数据集（RDD）的 checkpoint 机制。

Part B：使用 Spark 执行 PageRank 算法

候选人名单

运行截图见附录图 5。

```
(4037,13.68782466100178)
(15,10.932805952062104)
```

```
(6634,10.656469713599291)
(2625,9.755679770957746)
(2398,7.750205920765447)
(2470,7.498077775768758)
(2237,7.417430386169764)
(4191,6.737744460375287)
(7553,6.446227897467031)
(5254,6.387907761950121)
(2328,6.058602112229778)
(1186,6.0475330647209775)
(1297,5.781055756673953)
(4335,5.754084280496571)
(7620,5.740174509100523)
(5412,5.701062094145742)
(7632,5.667866508823103)
(4875,5.56706411352204)
(6946,5.372790269370961)
(3352,5.300100134345486)
```

PageRank 算法

描述

PageRank 算法在历史上作为计算互联网网页重要度的算法被提出。其一般定义如下：

给定一个含有 n 个结点的任意有向图，考虑一个在图上随机游走模型（一阶马尔可夫链），其转移矩阵是 M 。另有一个完全随机游走的模型，其转移矩阵的元素全部为 $1/n$。两个转移矩阵的线性组合又构成一个新的转移矩阵，在其上可以定义一个新的马尔可夫链，易证其一定具有平稳分布并满足

$$R = dMR + \frac{1-d}{n}1$$

d 为阻尼系数，R 表示的就是有向图的 PageRank。

我们的代码首先接收数据集、阻尼系数、迭代次数的参数并读取数据集：

```
val inputPath = if (args.length > 0) args(0) else "data/graphx/Wiki-Vote.txt"
val dampingFactor = if (args.length > 1) args(1).toDouble else 0.85
val epoch = if (args.length > 2) args(2).toInt else 100
val graph: Graph[Int, Int] = GraphLoader.edgeListFile(sc, inputPath)
```

然后初始化转移矩阵，每条边 (u, v) 的权重为 u 出度的倒数；再初始化每个顶点的 rank 为 1。

```
var ranks = graph.outerJoinVertices(graph.outDegrees) {
    (vertexId, _, degreeOption) => degreeOption.getOrElse(0)
}.mapTriplets(e => 1.0 / e.srcAttr).mapVertices((_, _) => 1.0)
```

然后使用上述公式进行迭代，迭代数轮后权重收敛。

```
for (_ <- 1 to epoch) {
    val contributions = ranks.aggregateMessages[Double](
        context => context.sendToDst(context.srcAttr * context.attr), _ + _)
    ranks = ranks.outerJoinVertices(contributions) {(id, rank, contribution)
        => 1.0 - dampingFactor + dampingFactor * contribution.getOrElse(0.0)
      }
```

```
    ranks.cache()
}
```

迭代过程使用了 `Spark.GraphX` 提供的以下内置函数：

- `aggregateMessages` 函数用于计算每个顶点的贡献值（即 PageRank）并将其分发给相邻顶点。
- `outerJoinVertices` 函数更新每个顶点的 rank。

最终将结果降序排序并输出前20位候选人名单。

```
println(ranks.vertices.collect().sortBy(- _._2).take(20).mkString("\n"))
```

部署

> 开发测试时使用的各软件版本为：Hadoop 2.10.2，Spark 3.3.3，sbt 1.9.7，Java 11

假设以 `root` 用户登入，首先在 master 节点启动 Hadoop 和 Spark，以下命令中的各软件、数据集、源文件位置需替换为实际存储位置。

```
cd /path/to/hadoop/ & sbin/start-all.sh
cd /path/to/spark/ & sbin/start-master.sh & sbin/start-slaves.sh
```

将数据集上传至 HDFS

```
hadoop fs -mkdir -p /user/root/data/graphx/
hadoop fs –put /path/to/Wiki-Vote.txt /user/root/data/graphx/
```

将项目源文件编译打包

```
 cd /path/to/pageRank & /path/to/sbt/sbt package
```

运行

```
cd /path/to/pageRank
/path/to/spark/bin/spark-submit --class "PageRank" \
target/scala-2.12/unikeen_pagerank_2.12-1.0.jar <inputPath> <dampingFactor> <epoch>
```

三个可选命令行参数分别代表数据集路径、阻尼系数、迭代次数。当参数省略时，数据集路径为 `data/graphx/Wiki-Vote.txt`（与之前过程保持一致），阻尼系数为 0.85，迭代次数为 100。

附录

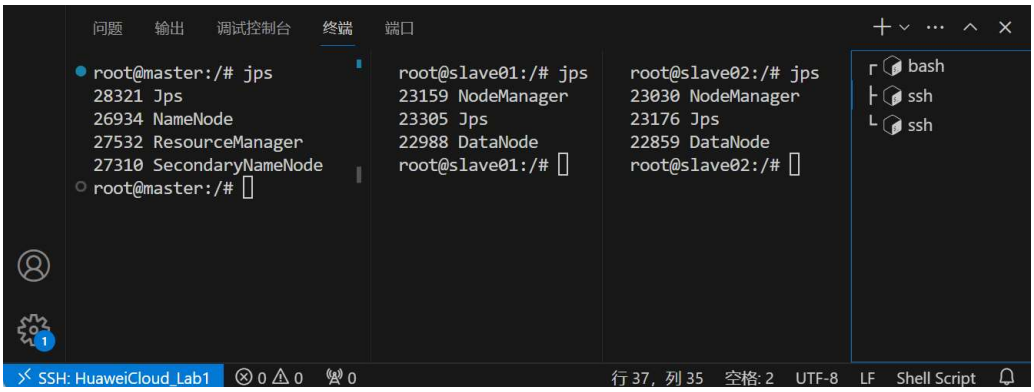

图 1：Hadoop 配置运行结果

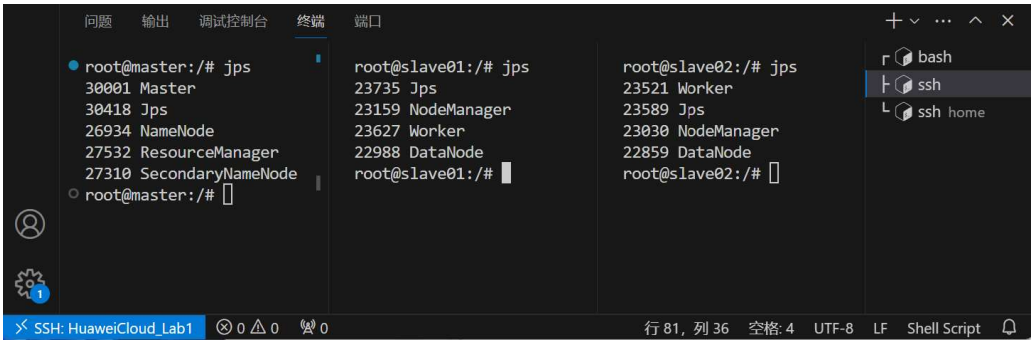

图 2：Spark 配置运行结果

```
root@master:/home# hadoop fs -cat /output/part-r-00000
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.10.2.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
:       1
Batman  1
But     1
I       13
I'm     1
I've    2
Oh,     4
She     1
Superman        1
Where'd 3
a       1
and     5
been    2
before  1
```

图 3：Hadoop Word Count 运行结果

```
23/11/29 13:31:16 INFO scheduler.DAGScheduler: Job 4 finished: collect at ConnectedComponentsExample.scala:63, took 0.124804 s
(justinbieber,1)
(matei_zaharia,3)
(ladygaga,1)
(BarackObama,1)
(jeresig,3)
(odersky,3)
```

图 4：Spark Connected Component Example 运行结果

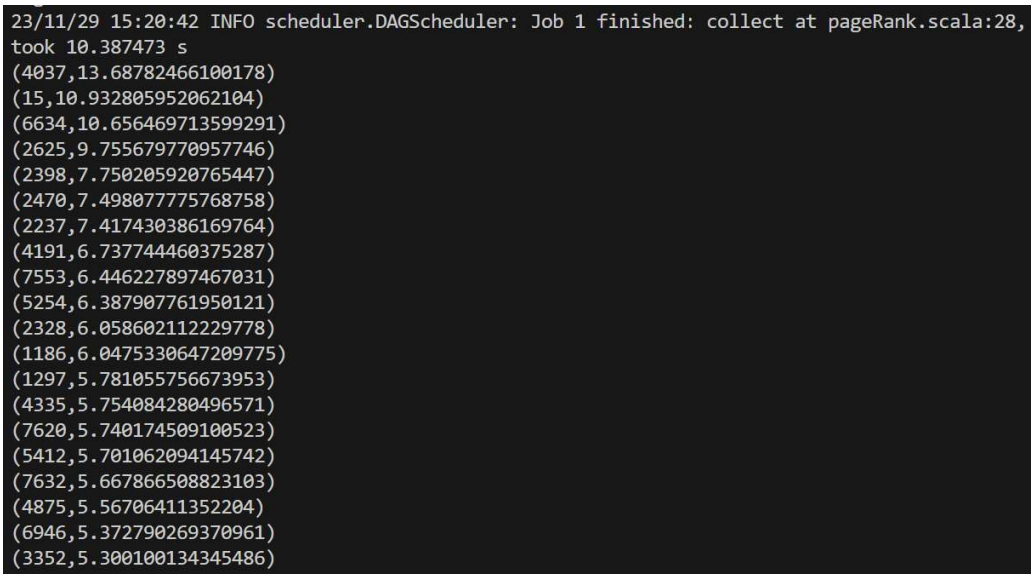

图 5：PageRank 运行结果