Spark 入门实操

enzo liu

2017-03-20 Thu

背景

内网服务器的 spark 又跑不动了... 所以在阿里云上买两台机器, 严肃点的部署一下. 顺便把之前的 pyspark 脚本也迁移到 scala 上，可以利用起我们 java 下的资源.

部署环境

- 机器阿里云 2 台 4 核 16g
- 依赖软件
  - ansible-2.0
  - spark-2.1
  - jdk-1.8

部署过程

为了后续添加 slave 方便，在 ansible 的脚本上花了很大的功夫。根据 master 以及 slave 的 inventory 配置

- 自动配置 authorized_keys
- 自动配置 master 的 ssh_config
- 自动配置 nfs, 以及 mount master 的工作区目录

具体 ansible 脚本的执行步骤大致如下:

- 安装 jdk
- 安装 spark
- 生成以及拷贝 spark 的配置文件

  conf/slaves 配置 ssh 的别名

  conf/spark-defaults.conf 配置 master 的 url

  conf/spark-env.sh 配置 JAVA_HOME, SPARK_HOME, 各类 MEMORY
- 配置 master 的 ssh_key, 以及添加到 slave 的 authorized_keys 中
  - spark 的 start-all.sh 中通过 ssh 来启动所有的 slave 的 worker
- 配置 nfs 共享工作区

– 运行模式下需要所有的 worker 都能根据地址访问到所要执行的 jar

发布方式

- 打包 `sbt assembly`
- 上传 `scp $WORKING_DIR/target/*.jar spark:/home/spark/workspace/`

执行方式

crontab 定期调度

```
$SPARK_HOME/bin/spark-submit --class *** /home/spark/workspace/***
```

App 示例

```
val idsRDD = odpsOps.readTable(project, table, pr, read, numPartitions)
  .filter(_.schema.schema == "activity_detail").filter(_.time.isAfter(start)) // 保留最近一个月的访问记录
  .flatMap(r => toInt(r.schema.key).map(((r.dvid, r.time.toString(DateTimeFormat.shortDate)), _))) //记录转换成设备号，访问日期，演出 ID 的各式
  .groupBy(_._1).map(_._2.map(_._2)).map(_.toSet) //根据设备号和访问日期聚合，且仅保留演出 ID 的信息
  .filter(ids => ids.size > 1 && ids.size < 5) //只保留访问的演出在 2-4 之间的数据
val weightRdd = generate(idsRDD)

def listToPair[T](ls:List[T]):List[(T,T)] = ls match {
  case Nil => List()
  case a::ls => ls.map((a,_)) ++ listToPair(ls)
}

def sort(p:(Int,Int)) : (Int,Int) = if (p._1 < p._2) p else p.swap

def toScore(elementNums:Map[Int,Long], pairNums:((Int,Int),Int)) : ((Int,Int), Double) = {
  val ((a,b),n) = pairNums
  ((a,b), n / Math.sqrt(1.0*elementNums(a)*elementNums(b)))
}

def generate(ls:RDD[Set[Int]]) : RDD[((Int,Int),Double)] = {
  val flattened = ls.flatMap(s=>listToPair(s.toList))
  val elementNumbers = ls.flatMap(_.toList).countByValue()
  flattened.groupBy(sort).mapValues(_.size)
    .map(r=>toScore(elementNumbers,r))
    .flatMap(withReverse)
}

def withReverse(res: ((Int,Int), Double)) = {
  val ((activityId, relatedId), weight) = res
  Seq(res, ((relatedId, activityId),weight))
}
```

执行结果

同样的功能，由于

- 数据源切换到了阿里云的 odps(内网带宽千兆可以跑满)
- 减少了 nginx 日志的解析工作 (odps 里可以相对结构化的存储信息)

原本 2 小时的执行任务，现在 4 分钟就能搞定...