|  |
| --- |
| 基于Hadoop&Spark的关联规则算法实践总结报告 |

曾楠嵘

2019/1/16

**目 录**

[1 实践目的及内容 1](#_Toc535419824)

[1.1 实践目的 1](#_Toc535419825)

[1.2 实践内容 1](#_Toc535419826)

[2 实践环境及工具 1](#_Toc535419827)

[2.1 实践环境 1](#_Toc535419828)

[2.2 实践工具 2](#_Toc535419829)

[3 实践步骤 2](#_Toc535419830)

[3.1 集群搭建 2](#_Toc535419831)

[3.2 Hadoop&Spark环境搭建 3](#_Toc535419832)

[3.2.1 Hadoop部署 3](#_Toc535419833)

[3.2.2 Spark部署 4](#_Toc535419834)

[3.3 程序编写 4](#_Toc535419835)

[3.3.1 语言选择 4](#_Toc535419836)

[3.3.2 关联规则算法选择 5](#_Toc535419837)

[3.3.3 关键代码 5](#_Toc535419838)

[3.3.4 调试运行 5](#_Toc535419839)

[4 测试结果及故障模拟 5](#_Toc535419840)

[4.1 测试结果 5](#_Toc535419841)

[4.2 故障模拟 6](#_Toc535419842)

[4.2.1 故障一 6](#_Toc535419843)

[4.2.2 故障二 7](#_Toc535419844)

[4.2.3 故障三 7](#_Toc535419845)

[5 问题与解决 8](#_Toc535419846)

[5.1 问题一 8](#_Toc535419847)

[5.2 问题二 8](#_Toc535419848)

[5.3 问题三 9](#_Toc535419849)

[6 总结 9](#_Toc535419850)

[7 参考文章 10](#_Toc535419851)

[7.1 集群搭建相关 10](#_Toc535419852)

[7.2 关联规则相关 10](#_Toc535419853)

[7.3 Scala相关 10](#_Toc535419854)

1. **实践目的及内容**

**1.1 实践目的**

此次实践主要目的在于，希望通过亲身实践，加深自己对Hadoop、Spark两类大数据工具的理解，熟悉其从集群部署到运作的基本流程，了解FPGrowth算法的基本原理，掌握算法在关联规则中的基本应用，为日后的大数据学习积累经验。

* 1. **实践内容**
* 搭建Linux系统，部署集群；
* 编写scala代码，使用spark对Groceries购物数据进行关联规则的分析；
* 模拟故障，分析spark运行情况；

1. **实践环境及工具**
   1. **实践环境**

本次实践主要在个人笔记本电脑与INTEL NUC中进行。为方便操作，我在个人笔记本的Windows系统中开放Wifi热点，NUC中的Linux系统通过wifi连接，可与我的windows电脑处于同一模拟的局域网内，这样我就可以在个人笔记本中远程登陆NUC中的Linux集群。

两台机器的性能配置如图2.1所示。

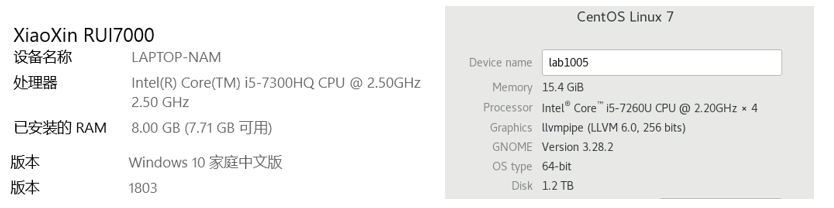


图2.1 实践机器性能配置图

关于实践中具体用到Hadoop与Spark相关版本如下：

* Hadoop 3.1.1
* JDK 1.8.0\_191
* Scala 集群 2.12.8；个人笔记本 2.11.12
* Spark 2.4.0
  1. **实践工具**
* **Xshell&Xftp**：用于个人笔记本与NUC主机的控制操作及文件传输；
* **VNC**：用于个人笔记本远程登陆NUC（在实践环境中，对于远程登陆时 的命令输入及图形界面操作的延时，VNC的操作较比Xshell要相对流程些，所以选择VNC做为主要的远程操作工具）；
* **Virtual Machine Manager**：管理Linux中的各KVM虚拟机；
* **FinalShell**: 用于NUC主机与各个虚拟机的控制操作及文件传输；

1. **实践步骤**

实践步骤主要可分为集群搭建、Hadooph&Spark环境搭建、程序编写、调试运行、故障模拟五大部分。下面为各部分的具体说明。

* 1. **集群搭建**

在实践的集群搭建中，是部署的7台服务器，一台用作nameNode,充当master角色，其余6台用作dataNode，充当worker角色。由于对secondary NameNode的相关知识还未进一步接触，暂时未搭建。

根据NUC的性能配置情况，对于nameNode与各worker的性能分配如下：

* master: memory 2G; storage 30G;
* worker:memory 1.5G; storage 30G;

当集群全部运作时，NUC主机的资源分配情况如图3.1所示。

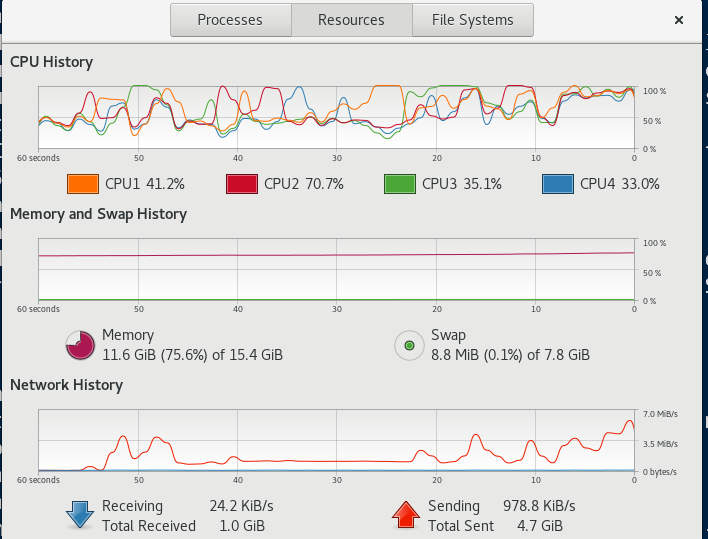


图3.1 NUC主机资源分配情况

* 1. **Hadoop&Spark环境搭建**

在搭建过程，先是部署的Hadoop集群环境，然后在Hadoop集群环境的基础上，再部署的Spark环境。下面为具体的部署过程及相关配置文档内容。

* + 1. Hadoop部署

由于Hadoop主要包括hdfs、yarn、mapReduce三大模块的部署相对复杂些，下面为大致的部署流程。

* Master相关
  + 配置IP，修改**/etc/hosts**,添加master及6台worker DNS解析；
  + 安装JDK,配置java环境变量
  + 安装hadoop,配置hadoop环境变量
  + 修改$HADOOP\_HOME/etc/hadoop/**hadoop-env.sh**,导入JAVA\_HOME
  + 修改$HADOOP\_HOME/etc/hadoop/**core-site.xml**，设置fs.defaultFS、hadoop.tmp.dir
  + 修改$HADOOP\_HOME/etc/hadoop/**hdfs-site.xml**,设置dfs.namenode.secondary.http-address、dfs.replication、

dfs.namenode.heartbeat.recheck.interval、

dfs.permissions.enabled

* + 修改$HADOOP\_HOME/etc/hadoop/**mapred-site.xml**，设置mapreduce.framework.name
  + 修改$HADOOP\_HOME/etc/hadoop/**yarn-site.xml**，设置 yarn.resourcemanager.hostname、

yarn.nodemanager.aux-services

* + 修改$HADOOP\_HOME/etc/hadoop/**workers**，添加6台worker的IP
* 通过Virtual Machine Manager复制Master，生成6台workers
* Workers相关
  + 配置IP
* 设置ssh免密登陆
* 启动hadoop集群，格式化nameNode
* web界面访问
  + hadoop端口 9870
  + yarn端口 8088
    1. Spark部署

部署好hadoop后，spark的部署相对简单些。下面为大致的部署流程。

* 安装scala,配置环境变量
* 安装spark，配置环境变量
* 修改$SPARK\_HOME/conf/**slaves**,添加6台worker的IP
* 修改$SPARK\_HOME/conf/**spark-env.sh**,设置基本属性
  + JAVA\_HOME
  + SCALA\_HOME
  + HADOOP\_HOME
  + HADOOP\_CONF\_DIR
  + SPARK\_MASTER\_IP
  + SPARK\_MASTER\_PORT
* 修改$SPARK\_HOME/conf/**spark-defaults.conf**,设置基本属性
  + spark.yarn.jars
  + spark.master
* 启动spark集群$SPARK\_HOME/sbin/start-all.sh(必须到spark相关目录，否则易于hadoop的命令冲突)
* web页码访问端口8080
  1. **程序编写**
     1. 语言选择

虽然spark支持Java、Phthon，但既然spark是基于Scala编写，同时未来借此实践了解一门新的语言，最后在程序语言上，我选择了Scala。由于有Java和Python语言的基础，在Scala语言的学习上就显得十分轻松。个人感觉在语法上结合了Java和Python的优点，语法简洁灵活，是一门不错的语言。

* + 1. 关联规则算法选择

在关联规则算法的选择上，由于Spark内部已实现了FPGrowth算法，且在效率上，FPGrowth要由于Apriori，所以，在程序编写时，我选用FPGrowth。

* + 1. 关键代码

关键部分代码主要包括数据文件处理、求频繁集、求关联规则三部分，具体的代码如图3.2所示。



图3.2 关键代码

* + 1. 调试运行

在windows下的IDEA中编写好程序代码后，生成任务jar包，通过Xshell传给NUC中的集群试运行。

运行时，主要采取的Spark运行模式是Standalone-client和Yarn-client模式，因为client模式可从控制台直接观察任务运行情况。另外，由于该实践仅是针对Groceries一个任务，未涉及客户端同时提交多个任务的情况，所以为便于观察任务执行情况，使用client模式即可。

1. **测试结果及故障模拟**
   1. **测试结果**

本次实验采用的数据集是Groceries数据集。该数据集是某个杂货店一个月真实的交易记录，共有9835条消费记录，169个商品。在测试中，我设定的minSupport为5%，minConfidence为30%，在次条件下，通过FPGrowth算法，一共得出三条关联规则，具体情况如图4.1所示。

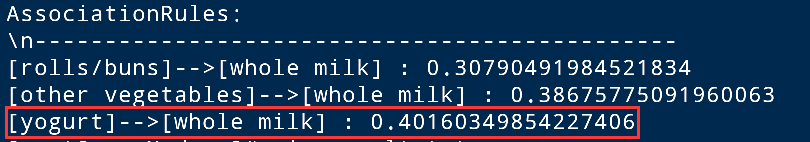


图4.1 运行结果

由结果可以看出，该商店在当月内，同时购买yogurt和whole mile 概率最高，相关的还有 rolls/buns和other vegetables。所以该超市可以将，yogurt、rolls/buns、other vegetables的柜架摆在whole mile规矩的附件。

* 1. **故障模拟**

为初步检验Spark的稳定性，我简单设置了3类故障。令人满意的是，虽然在故障发生时，Spark运行groceries任务的时间有相应增加，但最终的结果和正常情况下的结果一致。

所以，在我实践的任务强度内，Spark是具有很好的稳定性的。

下面为3次故障模拟的详细说明。

* + 1. 故障一

故障一是简单模拟集群出现小范围故障的情况。

我在6台worker全部开启，并执行groceries任务的情况下，强制关闭worker6。通过Spark的web界面，可以看到，任务的运行几乎没有影响，任务的执行时间和6台workers正常运行时所用的时间几乎一样。

具体的任务完成情况如图4.2所示。

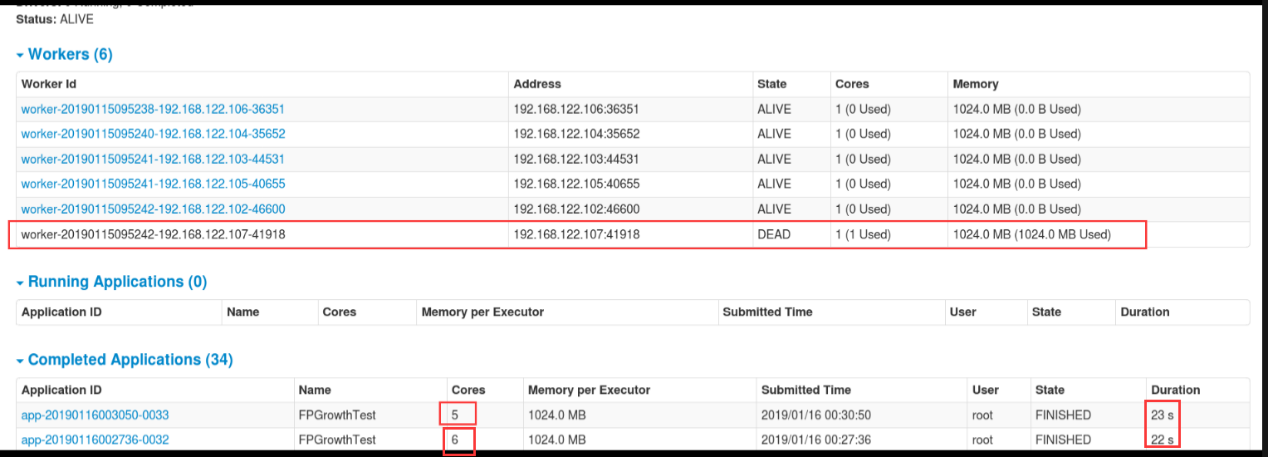


图4.2 故障一任务完成情况

* + 1. 故障二

故障二是简单模拟集群出现中度范围故障的情况。

我在故障一的基础上，在5台workers正常运行时，强制关闭其中的3台workers。通过park的web界面，可以看到，3台集群的关闭对任务的运行产生了较大影响，任务务的执行时间为5min,比之前的23s高出了10倍多的时间，尽管最后的结果是正确的。

具体的任务完成情况如图4.3所示。

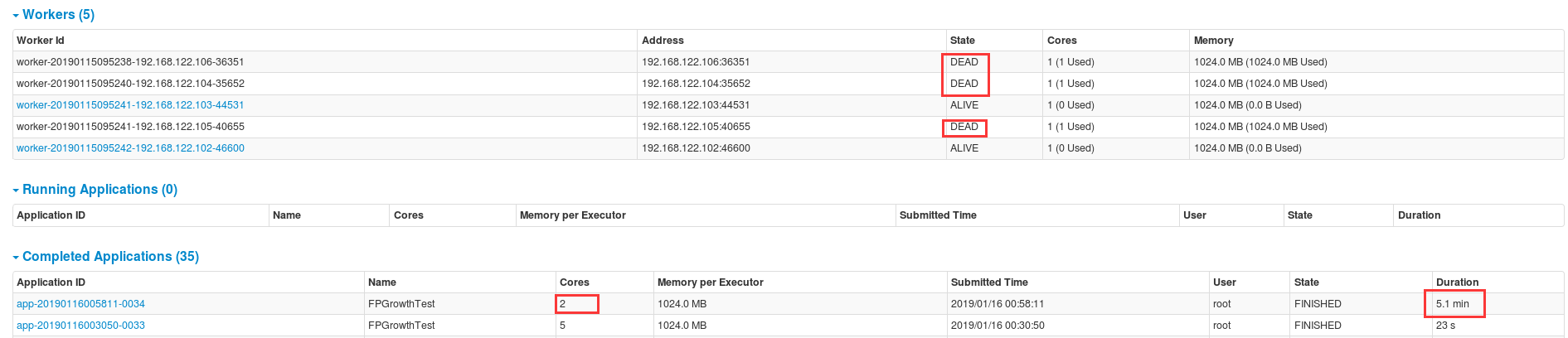


图4.3 故障二任务完成情况

* + 1. 故障三

故障三是简单模拟集群出现大范围故障的情况。

我在6台worker全部开启，并执行groceries任务的情况下，强制关闭5台workers。然而通过Spark的web界面，看到的却是，5台集群的关闭对任务的运行产生的影响远小于故障二，任务务的执行时间仅为1.5min。

于是，仅有一台worker的情况，正常运行之前的任务，发现运行的时间仅有15s,比6台workers同时运行的时间减少了近一半。

具体的任务完成情况如图4.4所示。

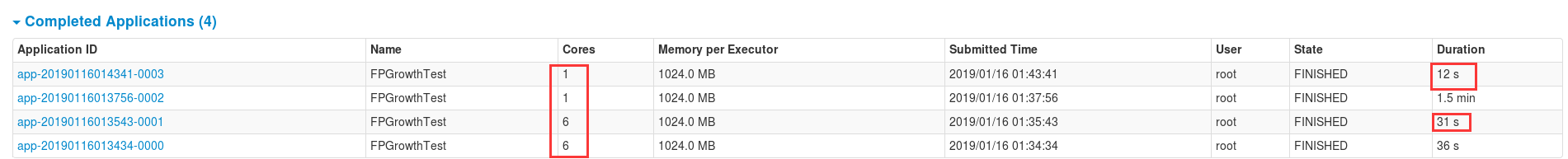


图4.4 故障三任务完成情况

在网上查阅相关资料后，我初步的猜想是，实践中运行的groceries任务量，相比Spark的运行能力，所造成的压力是比较轻微的。所示在使用6台workers运行时，workers与master之间的信息交互占据了较大比重的时间。

另外，我在spark-default.conf中对于一些spark运行参数，如spark.default.parallelism，的处理为默认，所以可能没有较大限度的利用当前的集群资源。关于Spark参数调优，我将在之后的学习中，深入了解。

1. **问题与解决**

由于本次实践是我第一次从零开始的、相对完整的进行集群搭建与测试，尽管现在回顾，发现本次实践还是相对简单与轻松的，但在实践过程中，的确遇到了蛮多问题，不过在解决问题的过程中，我学到的知识远远超过了实践本身所需要的知识。

下面我对几个典型的问题情况与解决方案进行简要说明。

* 1. **问题一**
* 问题描述
  + start-bfs.sh后，仅namenode启动，datanode未启动
* 问题分析
  + 由于namenode和datanode中保存的namespaceID不同所引起的。可能情况是启动过集群后，又重新执行了hadoop namenode -format导致的。
* 解决方案
  + stop-bfs.sh
  + 删掉datanode配置的dfs.data.dir目录
  + start-bfs.sh
  + 注意：这样一来所有文件就都没有了，要慎重！
  1. **问题二**
* 问题描述
  + eyond the 'VIRTUAL' memory limit. 2.2 GB of 2.1 GB virtual memory used.
* 问题分析
  + 为Yarn的虚拟内存计算方式导致。任务程序申请的内存为1G，Yarn根据此值乘以一个默认比例（2.1），得出申请的虚拟内存的值。当Yarn计算的用户程序所需虚拟内存值大于计算出来的值时，就会报出以上错误。
* 解决方案
  + 在yarn-site.xml文件中，将yarn.nodemanager.vmem-pmem-ratio的值调高一些，如设为2.5
  1. **问题三**
* 问题描述
  + spark-submit之后，报 NoClassDefFoundError异常
* 问题分析
  + 在程序运行时，某些类没有找到
  + 我在$SPARK\_HOME/jars目录下发现，scala-compiler-\*.jar和scala-library-\*.jar对应用的版本为2.11.12，而我之前安装的Scala版本为最新的2.12.8，从而找出程序编译与运行的版本不一致。
* 解决方案
  + 在windows中下载Scala 2.11.12,并用其编译原程序

1. **总结**

本次实践从任务计划的制定到最后的总结报告，大概历时半个月。尽管现在回顾整个实践内容，发现其实并没有我最开始想象的那么困难。但在实践过程中，确实是遇到了许多意想不到的问题，而且有些问题还困扰了蛮久，不过最后都得到了合理的解决，这也是现在最欣慰的了。下面是我在实践过程中一些收获。

* 进一步了解了Linux磁盘挂载与分区的相关细节；
* 进一步了解了Hadoop中hdfs与yarn的运行机制；
* 进一步了解了Spark的运行模式及相关运行机制；
* 了解了FPGrowth关联规则算法；
* 了解了Scala语言的基本知识及简单应用；
* 成功从Eclipse转到IDEA；
* 更加熟悉了GitHub的使用；
* 熟悉了跨系统的远程管理技巧；

1. **参考文章**
   1. **集群搭建相关**

* 虚拟化技术之KVM，搭建KVM <https://blog.csdn.net/CloudXli/article/details/78306546>
* KVM虚拟机网络配置 Bridge方式，NAT方式<https://blog.csdn.net/hzhsan/article/details/44098537/>
* Hadoop3.x集群搭建

<https://www.cnblogs.com/luhouxiang/p/4829443.html>

* Spark2.x集群部部署

https://blog.csdn.net/weixin\_36394852/article/details/76030317

* 1. **关联规则相关**
* Spark调优综述

<https://yq.aliyun.com/articles/461770?spm=a2c4e.11163080.searchblog.114.299d2ec1rNqQ2U>

* 频繁项集与关联规则 FP-growth 的原理和实现

<https://blog.csdn.net/huagong_adu/article/details/17739247>

https://www.ibm.com/developerworks/cn/analytics/library/machine-learning-hands-on2-fp-growth/index.html

* 基于Spark的FPGrowth（关联规则算法）

https://blog.csdn.net/wangqi880/article/details/52910078

* 1. **Scala相关**
* Scala 教程

<http://www.runoob.com/scala/scala-tutorial.html>

* Scala Standard Library

https://www.scala-lang.org/api/2.12.8/index.html