电类工程导论(C 类)实验 8 报告

贾萧松 516030910548

一、实验概述

Lab8 中我们学习了 Hadoop 的相关知识。

1. Hadoop

Hadoop 是一个分布式系统基础架构。用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。

Hadoop 的框架最核心的设计就是：HDFS 和 MapReduce。HDFS 为海量的数据提供了存储，则 MapReduce 为海量的数据提供了计算。

2. MapReduce

MapReduce 是一个编程模型，也是一个处理和生成超大数据集的算法模型的相关实现。用户首先创建一个 Map 函数处理一个基于 key/value pair 的数据集合，输出中间的基于 key/value pair 的数据集合；然后再创建一个 Reduce 函数用来合并所有的具有相同中间 key 值的中间 value 值。

3. HDFS

分布式文件系统（Hadoop Distributed File System），简称 HDFS。HDFS 有高容错性的特点，并且设计用来部署在低廉的硬件上；而且它提供高吞吐量来访问应用程序的数据，适合那些有着超大数据集的应用程序。

在本次实验中，我们使用 Hadoop 的样例完成了对 π 值的估测。

二、实验环境

Ubuntu14.04+jdk1.7+Hadoop2.8.2

三、实验内容

运用 hadoop 估算 π 的值。
通过调用 Hadoop 样例得到的表格如下。

Number of Maps	Number of samples	Time(s)	π
2	10	38.829	3.800
5	10	55.372	3.280
10	10	61.061	3.200
2	100	27.907	3.120
10	100	55.277	3.148

估算出精确度大于 5 位的 π ,在这里，由于已经发现 map 越多，效率越低，于是我采用的是 1 个 Map，1 亿 sample，用时 31.115s， 估算出来的 π 值为 3.14159256

为了进一步探究在不同 Map 和 Sample 下 time 和精确度的值我做了如下实验。

1. Map x Sample 值不变

Number of Maps	Number of samples	Times(s)	π
1	1024	26.134	3.1484375
2	512	30.902	3.1484375
4	256	38.478	3.1484375

由上图可以看出三点：

a. Map x Sample 的值相同，估算出来的 π 值一样。

这里经过查询的得知，Hadoop 中这个样例的实现使用的是拟蒙特卡罗法(quasi-Monte Carlo method)，亦称数论法。它是蒙特卡罗法的一种变形。由蒙特卡洛的原理，可知取样的数目越多，其结果越精确。

而产生随机数用的是 Halton sequence，其特点为给定两个初始数，产生一系点，偏差很小，可以看成是随机的。而在这个程序中给定的两个初始数可能是固定的，这也就解释了为什么只要测 Map x Sample 的值即取点的总数量确定估算出来的 π 值也就确定。

b. 既然取点数量相同，由实验数据，也就可以猜测 Map 的数量对运算时间影响比较大。

于是我做了如下两个实验

2. Map 的数量固定

Number of Maps	Number of samples	Times(s)	π
1	256	24.73	3.140625
1	512	25.442	3.1484375
1	1024	24.446	3.1484375
1	2048	25.449	3.14453125
1	4096	24.388	3.1396484375
1	100000000	31.115	3.14159256

从结果可以看出，Map 数量确定以后，sample 的数量对时间影响极小，即使增加到 1 亿，时间也不过长了 6~7s,可见产生随机点并计算的时间并不是主要瓶颈。

3. Sample 的值固定

Number of Maps	Number of samples	Times(s)	π
2	1000	27.803	3.144
4	1000	46.486	3.14
8	1000	55.933	3.141
16	1000	80.379	3.1425
32	1000	146.079	3.141

由上一个实验可得，产生随机点并计算的时间对结果影响并不大，那么从这个实验就可以看出 Map 的数量增加确实会导致计算时间的快速增长，是主要的瓶颈。

4.结论

由该样例的原理可知，Hadoop 中计算 π 的样例中采用大量采样的统计学方法，属于数据密集型的工作。这应该是适合于 Hadoop 的。至于为什么增加 Map 影响比较大可能因为这是一个伪分布式，只有一台机器实际工作，那么把任务切开确实反而会影响效率。

在学习 Hadoop 的过程中，我了解到 Hadoop 的设计逻辑是使用本地计算减少通信。适用的是数据量大，数据之间关联性不太强，计算量不大的场合。Hadoop 解决通信瓶颈的办法是计算向数据 io 迁移，但是这只在数据之间关联性比较弱的时候才比较有效，像本实验中的蒙特卡洛法就比较合适。

但是对于很多算法，因为数据之间关联性非常强，大量的通信是不可避免的，再加上会有密集的计算，这两点决定了 Hadoop 的并行方式对于很多科学计算问题是非常低效的。

四、问题与解决

1. Java 环境配置问题

在配置环境的过程中，按照提供的配置步骤执行，其中遇到的一个普遍的问题就是在安装 Java 环境过程中，Java 版本不匹配的问题。

这个问题的产生的原因很简单，在以往的 Lab 中为了安装 Lucene，同学们的 Ubuntu 中已经装过了 Java 1.8，而在配置教程的方法是安装 Java 1.7，此时默认路径是 Java 1.8 的，自然就报错了。

解决这个问题的办法很多，我找到的一种在我看来比较便捷的，既然由于先装的 Java 1.8 导致默认的 Java Path 是 Java 1.8，那么把 PATH 改一下不就可以了。于是，我在 Hadoop 账户中

```
hadoop@jia-virtual-machine:~/Desktop/vmware-tools-distrib$ gedit ~/.bashrc
```

然后

```
export PATH=/usr/lib/jvm/java-7-openjdk-amd64/bin:$PATH
```

这行代码的含义就是把 Java 1.7 的路径放在环境变量最优先查找的地方。

这样就解决了这个问题，其他完全按照教程安装就可以了。在向多名同学推荐了这个方法后，都很好的解决了问题。

五、总结

在 Lab8 中，我学习到了不少 Hadoop 相关的的知识，并完成了 Hadoop 的配置及测试样例的使用。

最后，衷心感谢老师和助教们的精心准备和辛勤付出，你们整理完善的 ppt 和到位的答疑大大提高了我学习 Hadoop 和配置 Hadoop 的效率，感谢~

贾萧松 516030910548
2017.11.10