网络出版时间：2013-01-11 09:53
网络出版地址：http://www.cnki.net/kcms/detail/11.2127.TP.20130111.0953.025.html

Hadoop 平台的海量数据并行随机抽样

宛　婉，周国祥
WAN Wan，　ZHOU Guoxiang
合肥工业大学 计算机与信息学院，安徽 合肥　230009；
School of Computer and Information, Hefei University of Technology, Hefei 230009, China

Based on Hadoop massive data parallel random sampling

Abstract："Information explosion" in today's society, Data mining ,because of mass data ,facing a new challenges .When Data mining turns to cloud computing platform to realize parallel, the study of parallel data random sampling to further reduce the size of the data size. In this paper , This paper presents a mapreduce parallel sampling algorithm which not only can clean up dirty data but also achieve the goal of equal probability sampling .The algorithm just needs to scan processed data only one time. Run this algorithm in the hadoop platform and compare its performance with common random sampling .As a result , this new algorithm obtains a very high time efficiency. It is a kind of effective method which lays a good foundation for doing research on sampling in future. It can also promotes data mining in the condition of facing mass data.

Key words：cloud computing; hadoop; mapreduce; parallel computing; data mining; random sampling

摘　要：在“信息爆炸”的当今社会，海量数据对数据挖掘提出新的挑战。在数据挖掘转向云计算平台实现并行化的同时，研究并行化数据随机抽样进一步降低处理的数据规模。本文提出一种单次扫描即可实现清理脏数据并实现等概率抽样的 mapreduce 并行抽样算法。在 hadoop 平台上实现并与普通随机抽样方法进行比较，得出其时间效率非常高，是一种行之有效的方法。为以后数据挖掘中的抽样研究和推动数据挖掘在海量数据下的发展奠定良好基础。

关键词：云计算；hadoop；mapreduce；并行计算；数据挖掘；随机抽样
中图分类号：TP391.12　　**文献标识码**：A　　doi：10.3778/j.issn.1002-8331.1210-0329

0　引　言

近几十年来，数据收集和数据存储技术的快速进步使得各组织机构积累海量数据。当今社会商业竞争日趋白热化，人们迫切需要从数据的矿山中挖掘出知识的金子，由此产生数据挖掘技术。数据挖掘是在大型数据存储中，自动地发现有用信息的过程。其从兴起以来，一直是研究的热点问题。但是随着数据的海量化，需要处理的数据规模越来越大，且由于数据分析内部的复杂性，现有的数据挖掘算法在性能上已经没办法满足需求。国内外很多学者在数据挖掘中引入云计算思维，实现数据挖掘算法的并行化，很多数据挖掘算法被转移到 hadoop 上用 mpareduce 并行框架[1]实现。除此之外，我们很自然想到并行化抽样算法，减小数据量集处理规模。本文提出了一种基于 mapreduce 框架的并行随机抽样，可以实现在清理脏数据的基础上，等概率的进行抽样。

1　mapreduce 编程模型

mapreduce 采用“分布治之”的思想，把对大规模数据集的操作，分发给一个主节点管理下的各分节点共同完成，然后能整合各分节点的中间结果，得到最终的结果[2]。它有两个重要函数，可以由用户编写：map 和 reduce。map 负责把任务分解成多个任务，reduce 负责把各个任务处理的结果汇总起来。至于在并行编程中的其他种种复杂问题，如分布式存储、工作调度、负载均衡、容错处理、网络通信等，均由 mapreduce 框架负责处理[3]，可以不用程序员操心。

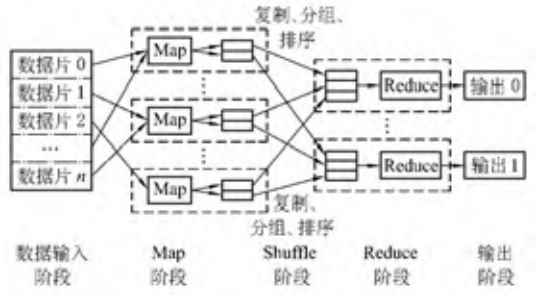

图 1　mapreduce 处理大数据集的过程

作者简介：宛婉（1988－），女，硕士生，主要研究方向：云计算； 周国祥（1956－），男 ,博士,教授,硕士生导师，主要研究方向：信息处理与智能决策，云计算。

如图 1 所示，在 map 阶段，mapreduce 框架将任务的输入数据分割成固定大小的片段（splits）,随后将每个 split 进一步分解成一批键值对$\langle K1,V1\rangle$。其作为 map 函数的输入，执行用户自定义的 Map 函数后，得到计算的中间结果$\langle K2,V2\rangle$。接着将中间结果按照 K2 进行排序，并将 key 值相同的 value 放在一起形成一个新列表，形成$\langle K2,list(V2)\rangle$元组。最后根据 key 值将这些元组分配到不同的 reduce 任务中。在 reduce 阶段，调用用户自定义的 reduce 函数，对输入的$\langle K2,list(V2)\rangle$进行相应处理，得到键值对$\langle K3,V3\rangle$并输出到 HDFS 上[4]。

2 随机抽样

抽样是一种选择数据对象子集进行分析的常用方法。在数据挖掘中，使用抽样的算法可以压缩数据量，以便可以使用更好但开销较大的数据挖掘算法。下面列举几种常见的抽样策略[5]：

2.1 简单随机抽样（simple random sampling）

将所有调查总体编号，再用抽签法或随机数字表随机抽取部分观察数据组成样本。优点：操作简单，均数、率及相应的标准误计算简单。缺点：总体较大时，难以一一编号。

2.2 系统抽样（systematic sampling）

又称机械抽样、等距抽样，即先将总体的观察单位按某一顺序号分成 n 个部分，再从第一部分随机抽取第 k 号观察单位，依次用相等间距从每一部分各抽取一个观察单位组成样本[6]。优点：易于理解、简便易行。缺点：总体有周期或增减趋势时，易产生偏性。

2.3 整群抽样(cluster sampling)

先将总体依照一种或几种特征分为几个子总体（类．群），每一个子总体称为一层，然后从每一层中随机抽取一个子样本，将它们合在一起，即为总体的样本，称为分层样本。优点：便于组织、节省经费。缺点：抽样误差大于单纯随机抽样[7]。

2.4 分层抽样（stratified sampling）

将总体样本按其属性特征分成若干类型或层，然后在类型或层中随机抽取样本单位，合起来组成样本[8]。有按比例分配和最优分配两种方案.特点：由于通过划类分层，增大了各类型中单位间的共同性，容易抽出具有代表性的调查样本。该方法适用于总体情况复杂，各类别之间差异较大（比如金融客户风险/非风险样本的差异），类别较多的情况。优点：样本代表性好，抽样误差减少。

3 并行的随机抽样算法

在以上四种抽样方法中，全部都需要随机抽取样本，也就是每个样本都是等概率的被抽取。但在 Hadoop 中，每个 job 会被分解成多个 task 并行计算，而数据的总量事先是不知道的（知道 job 运行结束才能获取数总数，而数据量非常大时，扫描一遍数据的代价非常高），用户知道的只是要获取的样本量，那怎样在类似于 Hadoop 的分布式平台上进行数据抽样进而利用并行的数据挖掘算法进行挖掘？本文提出了一种并行的蓄水池算法，实现未知数据总量的单次扫描下，并行的进行等概率抽样[9]。设计思想：先保存前 k 个元素（K 为样本量）， 从第 k+1 个元素开始， 以 1/i (i=k+1, k+2,…,N) 的概率选中第 i 个元素，并随机替换掉一个已保存的记录，这样遍历一次得到 k 个元素，可以保证完全随机选取。

$$\frac{k}{m}\times\left[\left(\frac{k}{m+1}\times\frac{k-1}{k}+\frac{m+1-k}{m+1}\right)\times\cdots\times\left(\frac{k}{n+1}\times\frac{k-1}{k}+\frac{n+1-k}{n+1}\right)\right]$$

$$=\frac{k}{m}\times\left[\frac{m}{m+1}\times\frac{m+1}{m+2}\times\cdots\times\frac{n}{n+1}\right]$$

$$=\frac{k}{m}\times\frac{m}{n+1}$$

$$=\frac{k}{n+1}$$

图 2 数学归纳法部分证明

下面用数学归纳法来给出证明:

问题，证明对于任意样本号 n, n>=k,每个样本作为取出样本的概率相等，即 k/n

证明：

当 n=k 时，由前 k 个数放入蓄水池可知，每个样本的取出概率均相等，即 k/k=1。 设当前样本号为 n，其每个取出样本概率均相等，即为 k/n，要证明的是这种情况对于 n+1 也成立。

由于以 k/(n+1)来决定是否把 n+1 放入蓄水池，那么对于 n+1 其出现在蓄水池中的概率就是 k/(n+1)，对于前 n 个元素中的任意元素 m(k+1<=m<=n)，其出现在蓄水池中的概率为 m 出现在蓄水池中的概率 * [(m+1 被选中的概率 *m 没被 m+1 替换的概率 + m+1 没被选中的概率)*(m+2 被选中的概率*m 没被 m+2 替换的概率

+ m+2 没被选中的概率)*…*(n+1 被选中的概率 *m 没被 n+1 替换的概率 + n+1 没被选中的概率)]，如图 2 所示。可见，对于 n+1 每个样本取出概率也相等，即为 k/(n+1)。证毕。

4 mapReduce 实现

要实现该抽样算法，只需编写 mapper 即可。在 map 函数中，用户定义一个数组保存选中的 k 个元素，待扫描完所有元素后，在析构函数中将数组中的数据写到磁盘中[10]。具体实现代码如下图所示：

```
public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
    if (index < k) {
        samples[index++] = value.toString();
    } else {
        randomValue = rnd.nextInt(index++);
        if (randomValue < k) {
            samples[randomValue] = value.toString();
}}}
protected void cleanup(Context context) throws IOException,
        InterruptedException {
    String[] fields;
    for (int i = 0; i < k; i++) {
        fields = samples[i].split("\t", 2);
        mapKey.set(fields[0]);
        mapValue.set(fields[1]);
        context.write(mapKey, mapValue);
}}}
```

图 3 mapreduce 实现代码

但是用户运行 job 时，需指定每个 map task 的采样量。比如，用户该 job 的 map task 个数为 s，则每个 map task 需要采集 k/s 个元素。但是，在 hadoop 中，并不能像设置 reduce task 那样直接设置 map task 数据的[11]。这时候，可以在 mapred-site.xml 里的 mapred.min.split.size 设置 split 的大小，这样可以间接设置 map task 数目。因为 hadoop 是将每个 split 分配一个 map task 任务的[12]。

同时，在 map 函数中，可添加语句对键值对 $\langle K1,V1\rangle$ 进行筛选，这就可以做到清理脏数据[13]，把一些不符合条件的记录先过滤掉，以提高抽样的有效性。

5 实验验证

本文中所有的实验都是在 hadoop 平台上运行的。平台由 17 台机器，204 核（17 个 AMD4234 双路 12 核）构成。其中有 7 台机器内存为 48G，10 台机器内存 16G。

hadoop 版本是 hadoop-0.20.2-cdh3u3，java 版本是 1.6.0_31。每台机器之间用千兆以太网，通过交换机连接。在实验中，采用比例作为评价指标。

实验数据是收集的应用数据，维度是 30 维。其大小为 138270431156 字节，约为：128.77G 共有 201569668 条记录，其中有些记录为不完全记录即为脏数据。经过清理后的数据为 152983354 条记录。设 min.split.size 为 1MB，设 K=1000,总共有 2072 个 map task,所以样本容量为 2072000 条。具体见图 3 所示，整个抽样过程花费时间仅为 15419ms，约为 4 分钟。

```
    Map input records=2072000
    Reduce shuffle bytes=9440
    Spilled Records=1088
    Map output bytes=31223196
    CPU time spent (ms)=39960
    Total committed heap usage (bytes)=3415212032
    Combine input records=2072000
    SPLIT_RAW_BYTES=2000
    Reduce input records=544
    Reduce input groups=34
    Combine output records=544
    Physical memory (bytes) snapshot=3549319168
    Reduce output records=34
    Virtual memory (bytes) snapshot=9352511488
    Map output records=2072000
time cost: 15419ms
```

图 4 mapreduce 运行结果

下面给出实验所抽取样本在 34 个类别与原数据在 34 个类别比例。

表 1 抽样前后比例表

类别	样本数	样本比例	原数据	原比例	样本原数据比
SH	70963	0.034249	5244764	0.034249	0.01353
YN	37624	0.018158	2803620	0.018158	0.01342
LMG	20195	0.009747	1507457	0.009747	0.013397
BJ	67480	0.032568	4990182	0.032568	0.013523
…	…	…	…	…	…
XG	2378	0.001148	174085	0.001148	0.01366
HLJ	27537	0.01329	2044629	0.01329	0.013544
总和	2072000	1	1.53E+08	1	0.013544

由表 1 可知，样本在 34 个类别上的分布和原数据在 34 个类别上的分布是一致的，这充分说明了本文提出的并行随机抽样数据的有效性，即正确反应原数据的分布规律，为接下来的数据挖掘提供可信的数据源。为了让结果更直观，把上表画成如图 4 的形式：

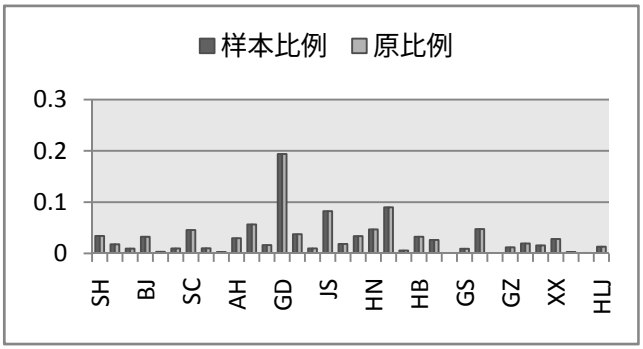

图 5　mapreduce 运行结果比例图

6　与普通随机抽样比较

现在采用普通的随机抽样，即先扫描全部娄据得到记录总数，然后利用 Random 类来得到相应数目的抽样[14]。下面是代码：

```
public static void main(String[] args) throws IOException {
    // 读取文件
    String fileName = "D:/student_info.txt";
    // 获取文件的内容的总行数
    int totalNo = getTotalLines(fileName);
    Scanner scanner = new Scanner(System.in);
    float gailv = scanner.nextFloat();
    int del_num = (int) (totalNo * gailv);
    for (int i = 0; i < del_num; i++) {
        Random rand = new Random();
        // 指定读取的行号
        int lineNumber = (int) (rand.nextDouble() * totalNo);
        // 读取指定行的内容把其与到目标文件中
        readLineVarFile("d:/student_info.txt", lineNumber);
    }}
```

其中 getTotalLines 的作用为扫描全表统计记录总数。在相同的实验环境（因为不是并行框架，所以只是针对单台机器多线程运行），针对相同的数据集，选用 4，8，16 线程。整个抽样过程最快统计时长为：78484568ms，约为 2.18 小时。其时长远远大于 mapreduce 并行随机抽样所需的 4 分钟。其原因主要是需要先把整个文件扫描一遍统计出记录总数，这将花费大量时间。

7　结　论

本文提出了一种在 hadoop 上运用 mapreduce 并行框架实现的单次扫描的随机抽样算法，给出了数学证明。并且和普通需知道样本总数的抽样方法进行比较[15]。相对于普通抽样算法的较长时间开销，这种算法不仅可以在单次扫描海量数据时完成对脏数据的清洗和过滤，还可以实现等概率的抽取样本。并且因为是并行计算，所以其时间性能是相当高。进一步提高了抽样数据的质量和有效性，为后续数据挖掘提供了很好的基础。但是实现并行随机抽样只是其它复杂抽样的基础，以后的工作可以在此基础上实现复杂抽样算法的并行化。

参考文献：

[1] Dean J，Ghemawat S． MapReduce: Simplified Data Processing on Large Clust［J］． Communications of the ACM，2008，51(1) : 107－ 113．

[2] Hadoop Streaming ［EB/OL］． ［2011 － 12 － 23］． http: / /hadoop.apache.org /common /docs /r0.15.2 //streaming.html．

[3]Hadoop T W. The Definitive Guide[M].YAHOO! Press,2009

[4] 李建江，崔健，王聃，等． MapReduce 并行编程模型研究综述[J]． 电子学报，2011，39 (11) : 2635 － 2642．

[5] Langendoen K，Romein J，Bhoedjang R，et al． Integrating Polling，Interrupts，and Thread Management ［C］． In: Proceedings of the 6thSymposium on the Frontiers of Massively Parallel Computation． Los Alamitos: IEEE Computer Society，1996: 13 － 22．

[6] Wenisch, T.F.; Wunderlich, R.E.; Falsafi, B.; Hoe, J.C. Statistical sampling of microarchitecture simulation [C]. Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International

[7] Bryan, P.D.; Conte, T.M. Combining cluster sampling with single pass methods for efficient sampling regimen design[C]. Computer Design, 2007. ICCD 2007. 25th International Conference.

[8] Tantan Liu; Fan Wang; Agrawal, G. Stratified Sampling for Data Mining on the Deep Web[C]. Data Mining (ICDM), 2010 IEEE 10th International Conference

[9]高纳德.计算机程序设计艺术第一卷[M].北京:国防工业出版社，2007：65

[10] 谢桂兰，罗省贤. 基于 Hadoop MapReduce 模型的应用研究[J]. 微型机与应用 ，2010,(08)

[11] 李珺. 基于 Hadoop 云计算模型探究[J]. 信息安全与技术 ，2011,(06)

[12] Ghemawat S，Gobioff H，Leung S． The Google File System ［C］．In: Proceedings of the 19th ACM SIGOPS Symposium on OperatingSystems Principles (SOSP ’ 03) ，Bolton Landing，NY． New York，USA: ACM，2003: 29 － 43．

[13] Al-Mudimigh, A.S.; Ullah, Z. Prevention of Dirty Data and the Role of MADAR Project[C]. Computer Modeling and Simulation (EMS), 2011 Fifth UKSim European Symposium.2011: 412 – 414.

[14] 秦如新，陈静，冯一宁. 一种新的关联规则抽样算法[J]. 中国农业大学学报 ，2007,(03)

[15]范明,孟小峰,译.数据挖掘概念与技术[M].北京机械工业出版社,2001.