基于MapReduce的高铁噪声数据预处理算法研究

王仲刚[1,2]，李天瑞[1,2]，张钧波[1,2]，赵成兵[1,2]，高子喆[1,2]

（1. 西南交通大学 信息科学与技术学院，四川 成都 610031；2. 四川省云计算与智能技术高校重点实验室，四川 成都 610031）

摘 要：随着高速铁路的快速发展，安全问题受到越来越多的关注，传感器采集的噪声数据反映了列车的运行状况，并与列车的安全息息相关。随着数据集的增大，数据处理的效率显得尤为重要，目前还无法高效地处理海量的高铁噪声数据。利用并行计算的思想，提出一种基于MapReduce的海量高铁噪声数据预处理算法。在Hadoop平台上进行实验分析，证明该算法可以有效地提高海量噪声数据预处理的效率。

关键词：噪声；高速铁路；MapReduce；Hadoop；并行

中图分类号：TP393 **文献标识码**：A **文章编号**：1000-436X(2011)9A-0263-07

Parallel method for preprocessing high-speed railway noise data based on MapReduce

WANG Zhong-gang[1,2], LI Tian-rui[1,2], ZHANG Jun-bo[1,2], ZHAO Cheng-bing[1,2], GAO Zi-zhe[1,2]

(1. School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China;
2. Key Lab of Cloud Computing & Intelligent Technology, Sichuan Province, Chengdu 610031, China)

Abstract: With the development of high-speed railway, its security problem gained more and more attention. Noise data collected by sensors reflected the operation condition and was close related to the security of train. The efficiency of processing data became significant due to the volume of data growing at an unprecedented rate. It was a challenge to process massive noise data effectively. A method for preprocessing massive noise data based on MapReduce was proposed by use of the idea of parallel computing. The experiments on Hadoop platform prove that the proposed method can improve the efficiency of preprocessing massive noise data.

Key words: noise; high-speed railway; MapReduce; Hadoop; parallel

1 引言

世界各国已建成和正在建设的高速铁路均将综合安全保障体系的研究放在首位。我国近几年来也致力于高速铁路的建设，并通过各种科技计划对高速铁路关键技术进行引导创新。高速列车运营和轨道线路的噪声数据在传感器、通信和存储等技术的发展日趋成熟的情况下可以被人们采集到。分析这些数据在轨道线路的分析、控制和故障诊断等方面有重要的意义。但采集的噪声数据由于各种外部和内部的原因而含有干扰数据，分析这些数据之前须进行预处理。由于真实情况的复杂性和处理操作对数据格式的严格要求，预处理步骤是数据处理与分析中耗时最多的，因此对高铁中噪声数据的预处理研究尤为重要。传统的噪声数据预处理方法有数据平滑[1]、数据平移[2]、异常点剔除[2,3]、趋势项消除[4]和小波去噪[5]等。很多学者在单机上实现了预处理的算法。例如，陈燕等人讨论了消除趋势项的

收稿日期：2011-07-05

基金项目：国家科技支撑计划基金资助项目（2009BAG12A01-E08-1）

Foundation Item: The National Key Technology R&D Program of China (2009BAG12A01-E08-1)

平均斜率法及其不足，并提出了利用最小二乘法来消除趋势项，同时在 LabVIEW 上编程实现[6]。肖立波等人讨论了最小二乘法消除趋势项以及五点三次平滑法并在 MATLAB 上实现[7]。近年来，随着采集的噪声数据量的飞速增长，单机处理噪声数据效率低，甚至不能处理快速增长的数据，难以满足实际需要。

MapReduce 是 Google 提出的一个产生和处理大数据集的编程模型和具体实现[8,9]。它尤其擅长处理单一功能性的大型数据集。目前有很多有关 MapReduce 模型及其应用的研究。例如，谢桂兰等人研究了 Hadoop 中 MapReduce 模型的工作方式和应用方法，实验结果表明在数据规模和处理器数目选择适当的情况下可以取得良好的并行计算效果[10]。郑启龙等人研究了基于 MapReduce 模型的并行科学计算，并结合实例编写了并行程序，进行了测试和结果分析，验证了使用 MapReduce 模型描述大规模科学计算问题的实用性[11]。赵青等人研究了 MapReduce 在天文方面的应用[12]。由于高铁噪声数据是由数字信号组成，其预处理方法与现有基于 MapReduce 的应用的计算方法不同，因此现有的这些方法还无法直接应用到高铁噪声数据预处理。本文拟研究 MapReduce 在海量高铁噪声数据预处理工作中的应用，以提高海量噪声数据预处理的效率。

2 预处理方法

本节介绍噪声数据预处理的 2 种方法：数据平移和数据平滑[1,2]。

2.1 数据平移

数字信号一般均值不为零，根据数字信号的特点，将信号的均值处理成零，令曲线在零轴上下波动可以使数据分析更为便捷，并能使数据计算的结果更为直观。故本实验使用数据平移的方法将数据的均值处理成零。

数据平移方法先计算数据集的平均值，然后用每个数据依次减去平均值，得到平移后的数值。

$$Ex = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

$$x_i^* = x_i - Ex \tag{2}$$

其中，Ex 为数据的平均值，x_i 为数据的测量值，x_i^* 为数据的修正值。平移处理只改变了数据的均值，未改变数据的其余特性，所以未对数据产生太大的影响。

2.2 数据平滑

数字信号一般含有数据误差，给数据的分析造成很大的困难。平滑处理是噪声数据的一项重要的预处理工作，它可以除去采集中偶然因素所造成的数据误差，以保障数据分析结果的准确性。

本文使用的是五点三次平滑方法，该方法是用临近的 5 个点加权来修正数据值。式(3)中 $\bar{y}_i$ 表示 y_i 的修正值。起始的 2 个数据与结尾的 2 个数据采用 $\bar{y}_{-2}, \bar{y}_{-1}, \bar{y}_1, \bar{y}_2$ 的平滑公式平滑，其余点均采用 $\bar{y}_0$ 平滑公式平滑。由于各平滑子区间采用的是附近 5 个点，所以对原数据的基本特性不会产生太大的影响。

$$\begin{cases} \bar{y}_{-2} = \frac{1}{70}(69y_{-2} + 4y_{-1} - 6y_0 + 4y_1 - y_2) \\ \bar{y}_{-1} = \frac{1}{35}(2y_{-2} + 27y_{-1} + 12y_0 - 8y_1 + 2y_2) \\ \bar{y}_0 = \frac{1}{35}(-3y_{-2} + 12y_{-1} + 17y_0 + 12y_1 - 3y_2) \\ \bar{y}_1 = \frac{1}{35}(2y_{-2} - 8y_{-1} + 12y_0 + 27y_1 + 2y_2) \\ \bar{y}_2 = \frac{1}{70}(-y_{-2} + 4y_{-1} - 6y_0 + 4y_1 + 69y_2) \end{cases} \tag{3}$$

3 MapReduce 模型介绍

MapReduce 中的作业（Job）是客户端执行的单位：它包括输入数据、MapReduce 程序和配置信息。Hadoop 把作业分成若干个任务（Task）来执行。MapReduce 的工作过程分为 Map 阶段和 Reduce 阶段。

Map 阶段：输入的数据被拆分成若干个数据片段，每个数据片段被分配给相应的 Map 任务，每个 Map 任务根据其分配的 key/value 调用自定义的方法，生成中间结果 key/value 集合 $\{<key_1, value_1>, \cdots, <key_n, value_n>\}$。该中间结果集合通过 shuffle 函数对 key 进行排序，然后会产生一个新的 key/value 集合 $\{<key_1^*, value_1^*>, \cdots, <key_n^*, value_n^*>\}$。

Reduce 阶段：将 key/value 集合 $\{<key_1^*, value_1^*>, \cdots, <key_n^*, value_n^*>\}$ 作为每个 Reduce 任务的输入，调用定义的 Reduce 函数输出处理后的 key/value 值 <key,value>。最后将所有节点上的 Reduce 任务的输出合并成最终结果。

注 1：1 个 key 对应的 value 值可有多个，用 value_list 表示，例如：$<key_1,value\_list>$，$value\_list=\{1,2,\cdots,n\}$。

MapReduce 模型如图 1 所示，其中 HDFS 指 Hadoop 分布式文件系统[9]。

4 基于 MapReduce 的并行噪声数据预处理算法

本节是在第 2 节介绍的噪声数据预处理方法的基础上提出基于 MapReduce 的并行噪声数据预处理算法。

4.1 并行噪声数据平移算法

并行噪声数据平移算法如图 2 所示。

其中，x_i 是噪声数据值，Ex 是数据的平均值，$\overline{x_i}$ 是处理后的数据值。第一份数据全部存入 key_0 对应的 value_list 集中，用于计算平均值。第二份数据按顺序存入 $key_1,key_2,\cdots,key_n$ 中。key_0 中的数据用于计算均值 Ex，$key_1,key_2,\cdots,key_n$ 中的数据用于平移操作。

1）数据平移 Map 步骤

第一份读入的数据文件存入 key_0 对应的 value_list 集中，即 $<key_0,value\_list>$。第二份读入的数据文件存入 $<key_1,x_1>,<key_2,x_2>,\cdots,<key_n,x_n>$ 中。算法伪代码见 Algorithm 1 如下。

Algorithm 1:数据平移-Map＜k,v＞

Input:

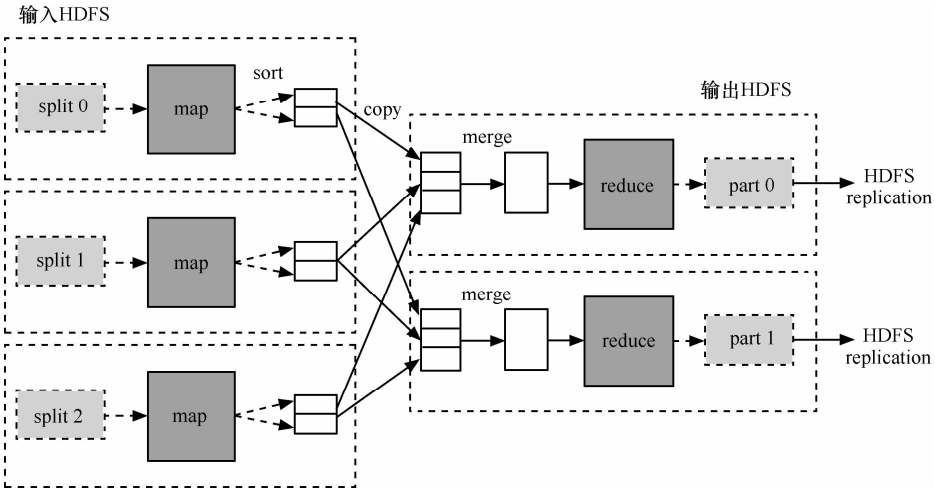


图 1　MapReduce 模型

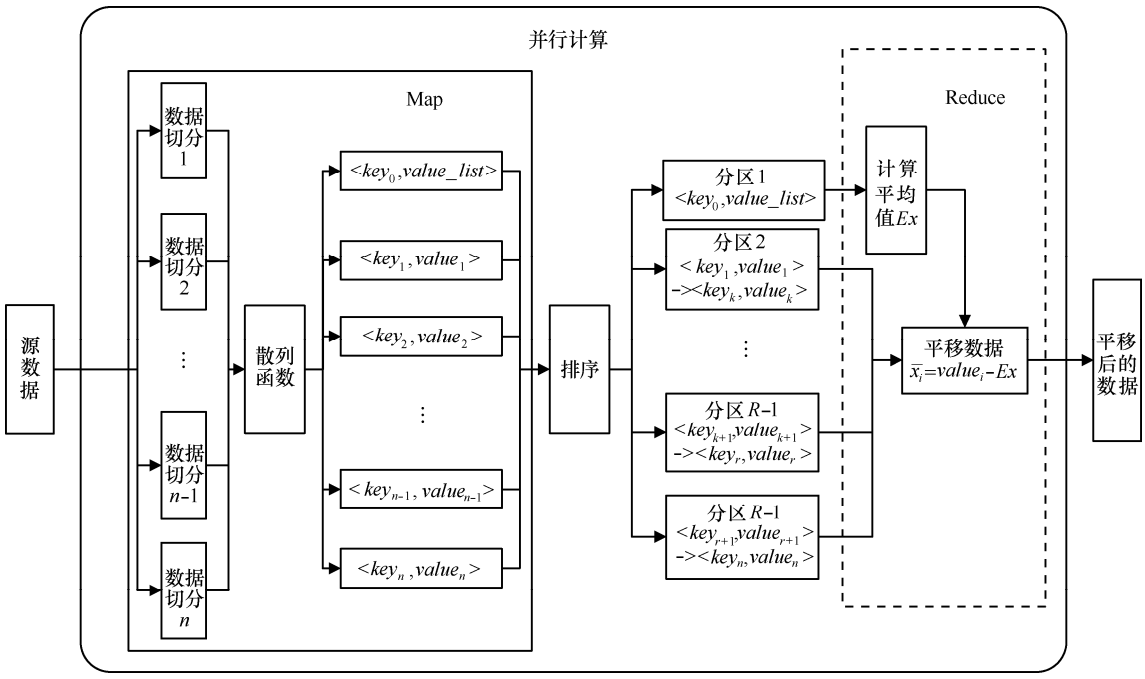


图 2　并行噪声数据平移算法

```
//key:噪声数据文件名
//V: 噪声数据集
Output:
<k,v>:k 是数据标识，v 是对应标识 k 的噪声数据值
Begin
    let k = 0
    foreach v∈V do
      k++;
      output.collect<0,v>;
      output.collect<k,v>;
    end
end
```

2) 数据平移 Reduce 步骤

通过 key_0 对应的 value_list 计算均值 Ex，$Ex = \frac{1}{n}\sum_{i=1}^{n} x_i$。通过 $< key_1, x_1 >, < key_2, x_2 >, \cdots < key_n, x_n >$ 计算出平移后的数据，即 $\overline{x_i} = x_i - Ex$，并存入文件。算法伪代码见 Algorithm 2 如下。

```
Algorithm 2: 数据平移-Reduce<k,v>
Input:
//<k,v>:k=0 的时候，V 是噪声数据集
//<k,v>: k>0 的时候，v 是 k 对应的噪声数据值
Output:<k,v>:k 是数据标识，v 是对应标识 k 的噪声数据值
begin
    let n=0
    if k=0 then
       let sum = 0
       foreach v∈V do
          n =n+1;
          sum = sum + v
       end
       average= sum /n
    end
    else
       let k ←null;
       foreach v∈V do
          v= v-average;
       output.collect<k,v>
     end
   end
end
```

4.2 并行噪声数据平滑算法

并行噪声数据平滑算法如图 3 所示。

并行噪声数据平滑算法中，由于在 MapReduce 中，每个 key 对应的数据集 $value\_list\{x_1, x_2, \cdots, x_n\}$ 经历完 Map 阶段以后，在 Reduce 阶段读取这个 key 对应的数据集 $value\_list\{x_1, x_2, \cdots, x_n\}$ 时，数据集中的数据值的顺序是随机的，而且不同的 key 对应的数据集中的数据值顺序的变化规律不同，所以如果在 Reduce 阶段使用平滑加权公式就会因为数据值顺序发生变化而不满足原有公式的使用条件，而无法在 Reduce 阶段使用平滑加权公式。

例 1 key_3 下的数据集为 $\{x_1, x_2, x_3, x_4, x_5\}$，在 Reduce 阶段 key_3 对应的数据集中的数据值的顺序会改变，故不能在 Reduce 阶段使用固定加权公式 $\{-3x_1, 12x_2, 17x_3, 12x_4, 3x_5\}$ 处理。解决方案是在 Map 阶段进行加权处理，在 Reduce 阶段进行平均处理。

1) 数据平滑 Map 步骤

读入的数据文件，对每个 x_i 进行加权处理，并存入临近的 5 个 key 对应的数据值，形成键值对 $\{< key_{i-2}, -3x_i >, < key_{i-1}, 12x_i >, < key_i, 17x_i >, < key_{i+1}, 12x_i >, < key_{i+2}, -3x_i >\}$。算法伪代码见 Algorithm 3 如下。

```
Algorithm 3: 平滑处理-Map<k,v>
Input:
//key:噪声数据文件名
//V: 噪声数据集
Output:
<k,v>,k 是数据标识，v 是对应标识 k 的噪声数据值
Begin
    let k = 0
    foreach v∈V do
       k++;
       output.collect<k-2,-3v>;
       output.collect<k-1,12v>;
       output.collect<k,17v>;
       output.collect<k+1,12v>;
       output.collect<k+2, 3v>;
    end
end
```

2) 数据平滑 Reduce 步骤

通过每个 key_i 对应的数据集计算 sum_i，$sum_i = v_{i-2} + v_{i-1} + v_i + v_{i+1} + v_{i+2}$。计算出平滑后的数据

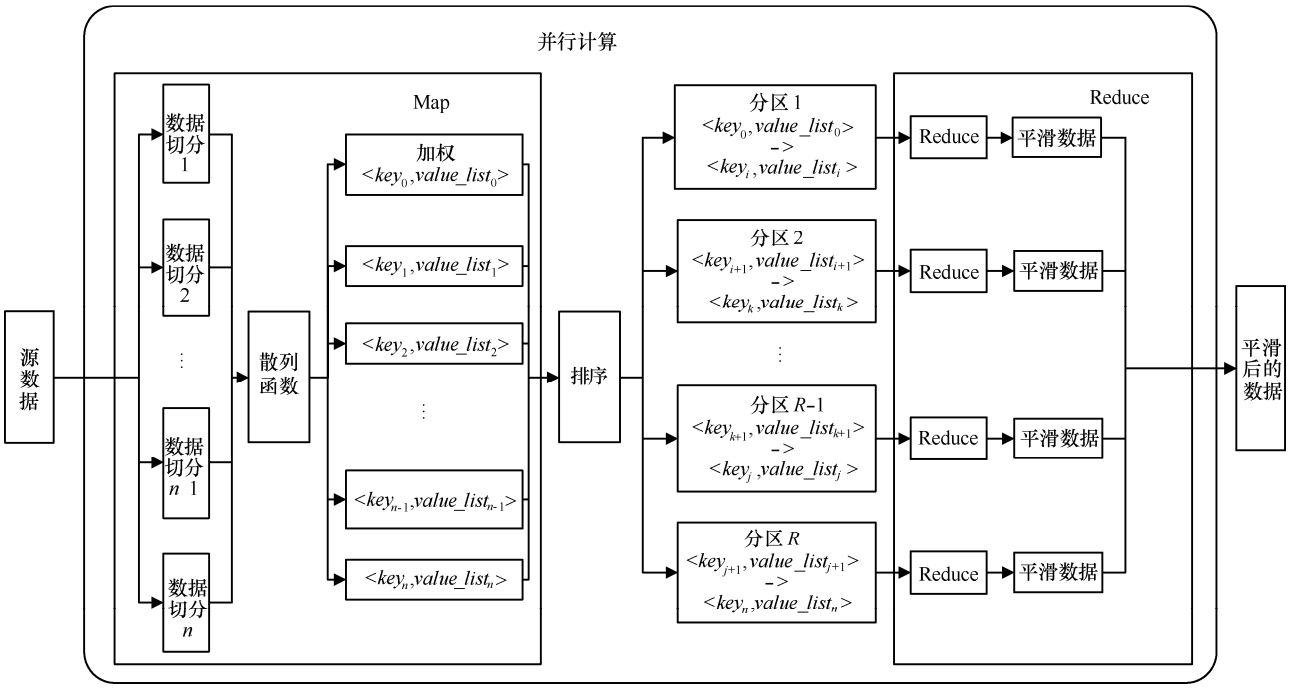


图 3　并行噪声数据平滑算法

$x_i = sum_i/35$，存入文件。算法伪代码见 Algorithm 4 如下。

Algorithm 4：平滑处理-Reduce＜k,v＞

Input:

//k:噪声数据文件名

//V:k 对应的噪声数据集合

Output:

＜k,v＞,k 为空，v 是平滑后的噪声数据值

```
begin
    let value = 0;
    for each v∈V do
      value+ = v;
    end
    value= value/35
    let k = null;
   output.collect＜k,v＞
end
```

5　实验分析

本节对噪声数据预处理并行算法利用 Speedup 和 Sizeup 指标进行评价[13,14]。本实验是在 Hadoop 平台上运行的，Hadoop 平台可以进行大规模的数据并行计算。实验的硬件环境是 6 台服务器，每台服务器配置为：4CPU，16 核，内核频率为 2.16GHz，内存是 12G。实验的软件环境是 Hadoop 版本为 0.20.2，Java 的版本是 1.6.0.01。

本文实验使用的数据是高铁噪声数据集，具体信息如表 1 所示，原始的噪声数据是由采集器采集的电压值，本实验使用的是将其处理后的声压值数据。

表 1　高铁噪声数据集

噪声数据集	记录条数	文件个数	文件大小/MB
HSRN-1	16451260	20	204
HSRN-2	30902520	40	408
HSRN-3	49353780	60	612
HSRN-4	65805040	80	816
HSRN-5	82256300	100	1020

5.1　Speedup

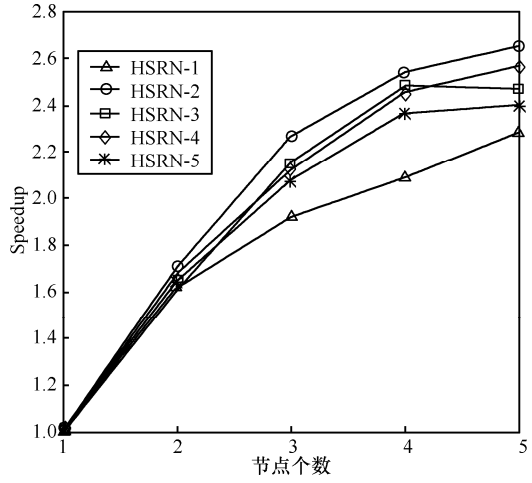
在 Speedup 的实验中，通过增加实验的节点（服务器）的数量并保持数据的大小来测试 Speedup 特性。Speedup 的计算公式如下：

$$Speedup(p) = \frac{T_1}{T_p} \tag{4}$$

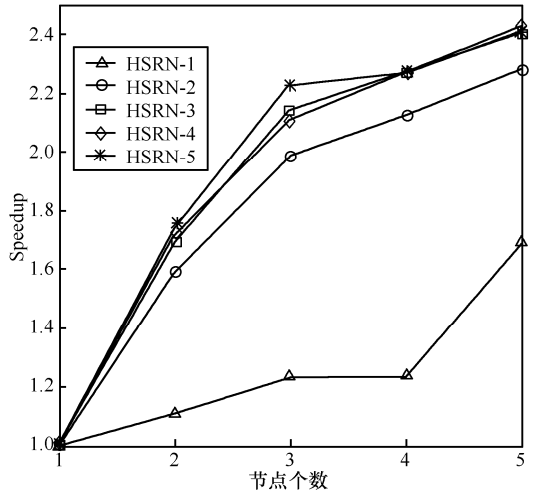
其中，p 是节点（服务器）的数量，T_1 是在一个节点参与实验的情况下系统执行的时间，T_p 是在 p 个节点参与实验的情况下系统的执行的时间。随着数据量的增大，Map 阶段读入数据的时间开销会线性增长，而系统准备时间的增长率小于读入数据的时间

的增长率，所以随着数据量的增大，并行处理的效率相应提升。

本实验使用2种预处理方法，分别是数据平移和数据平滑。通过改变噪声数据集的大小和节点数来评价Speedup。节点的个数从1个增加到5个。实验噪声数据集分别为HSRN-1至HSRN-5。图4显示了这2种方法上处理噪声数据集的Speedup指标情况。实验结果表明Speedup指标性能显著，参与实验的节点个数越多Speedup的效果越好。



(a)并行数据平移



(b)并行数据平滑

图4 Speedup

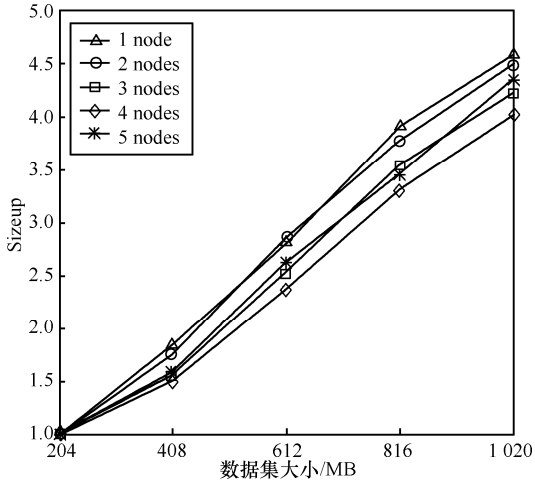
5.2 Sizeup

Sizeup评价噪声数据增长时并行算法的性能，定义如下：

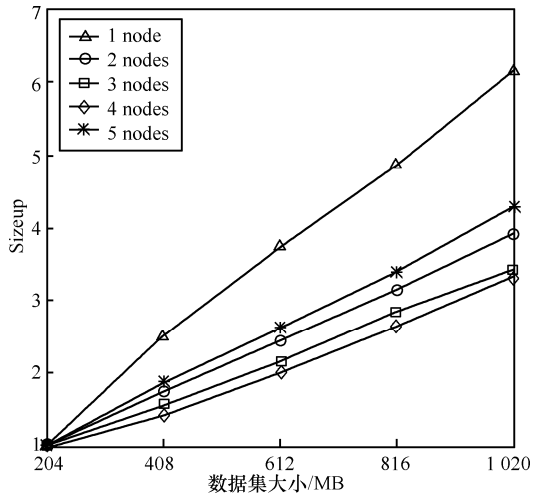
$$Sizeup(D,n)=\frac{T_{S_n}}{T_{S_1}} \tag{5}$$

其中，T_{S_1}是处理噪声数据集D时系统执行的时间，T_{Sn}是处理n倍于噪声数据集D时系统的执行时间。

本实验使用并行数据平移和并行数据平滑2种预处理方法。节点的数目从1增加到5以测试Sizeup效果。噪声数据集的大小从204MB增加到1 020MB。图5显示了这2种方法上处理噪声数据集的Sizeup效果。实验结果表明随着节点数目的增加，Sizeup指标性能越来越显著。



(a) 并行数据平移



(b) 并行数据平滑

图5 Sizeup

6 结束语

本文首先回顾了传统噪声数据预处理方法中的平移、平滑处理过程。为了处理海量高铁噪声数据，将传统噪声数据预处理方法与并行编程模型MapReduce相结合，提出了基于MapReduce模型的高铁噪声数据并行预处理算法，并开发了相应的MapReduce程序和部署到Hadoop分布式平台进行

实验验证。在不同大小的噪声数据集上进行 Speedup 和 Sizeup 性能对比实验，结果表明当数据集不断增大时，2 种噪声数据预处理并行算法性能更好，说明本文提出的并行方法能有效地处理海量高铁噪声数据。下一步工作将增加异常点检测及趋势项消除到高铁噪声数据预处理过程中，并开发相应的海量高铁噪声数据预处理实时系统，用于处理不断增长的高铁噪声数据。

参考文献：

[1] 孙苗钟. 基于 MATLAB 的振动信号平滑处理方法[J]. 电子测量技术,2007,32(6):55-57.
SUN M Z. Smooth processing methods of vibration signal based on MATLAB[J]. Electronic Measurement Technology,2007,32(6):55-57.

[2] 高品贤. 测试信号分析处理方法与程序[M]. 成都：西南交通大学出版社,1999.
GAO P X. Methods and Program for Test Signal Analysis and Processing[M]. Chendu: Southwest Jiaotong University,1999.

[3] 童丽，曾泳涨，王正明. 异常点剔除及其并行实现[J]. 数值计算与计算机应用,2000,32(3):171-177.
TONG L, ZENG Y Z, WANG Z M. Picking out outlier data and its parallel algorithm [J]. Journal on Numerical Methods and Computer Applications,2000,32(3):171-177.

[4] 侯运鑫，张桂香，邵梅等. 基于 Matcom 与 VC++的信号趋势项处理[J]. 山东理工大学学报(自然科学版),2009,23(1):53-55.
HOU Y X, ZHANG G X, SHAO M, et al. Dealing with trend item of collected signals based on Matlab and VC++[J]. Journal of Shandong University of Technology(Science and Technology),2009,23(1):53-55.

[5] 吴红艳，高国宏，李俊颖. 小波包降噪在工业信号预处理中的应用[J]. 科技广场,2010,3:115-117.
WU H Y, GAO G H, LI J Y. Application of wavelet packet en-noising in industry signal pretreatment[J]. Science Mosaic,2010,3:115-117.

[6] 陈燕，刘哲，郑宾. 基于 LabVIEW 的测试信号预处理方法研究[J]. 国外电子测量技术,2008,27(10):4-5.
CHEN Y, LIU Z, ZHEN B. Study on test signal pre-processing method based on LabVIEW [J]. Foreign Electronic Measurement Technology,2008,27(10):4-5.

[7] 肖立波，任建亭，杨海峰. 振动信号预处理方法研究及其 MATLAB 实现[J]. 计算机仿真,2010,27(8):330-333.
XIAO L B, REN J T, YANG H F. Study on vibration signal pre-processing method based on Matlab[J]. Computer Simulation, 2010,27(8):330-333.

[8] DEAN J, GHEMAWAT S. MapReduce: a flexible data processing tool[J]. Communications of the ACM,2010,53(1):72-77.

[9] WHITE T. Hadoop: the Definitive Guide [M]. 1005 Gravenstein Highway North, Sebastopol, CA: O Reilly Media,2009.

[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,2010,29(8):4-7.
XIE G L, LUO S X. Study on application of MapReduce model based on Hadoop[J]. Microcomputer & Its Applications,2010,29(8):4-7.

[11] 郑启龙，房明，汪胜等. 基于 MapReduce 模型的并行科学计算[J]. 微电子学与计算机,2009,26(8):13-17.
ZHEN Q L, FANG M, WANG S, et al. Scientific parallel computing based on MapReduce model[J]. Microelectronics & Computer,2009, 26(8):13-17.

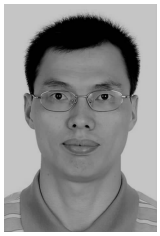
[12] 赵青，孙济洲，肖健等. 基于 MapReduce 模型的分布式天文交叉证认[J]. 计算机应用研究,2010,27(9):3322-3325.
ZHAO Q, SUN J Z, XIAO J, et al. Distributed astronomical cross-match based on MapReduce model[J]. Application Research of Computers,2010,27(9):3322-3325.

[13] XU X W, KRIEGEL H P, JAGER J. A fast parallel clustering algorithm for large spatial databases[J]. Data Mining and Knowledge Discovery,1999,3(3):263-290.

[14] ZHAO W, MA H, HE Q. Parallel k-means clustering based on MapReduce[J]. Lecture Notes in Computer Science, 2009, 5931: 674-679.

作者简介：

王仲刚（1987-），男，四川成都人，西南交通大学硕士生，主要研究方向为云计算与数据挖掘。



李天瑞（1969-），男，福建莆田人，博士，西南交通大学教授、博士生导师，主要研究方向为智能信息处理、粗糙集与粒计算、云计算等。

张钧波（1986-），男，浙江余姚人，西南交通大学博士生，主要研究方向为云计算、数据挖掘、粗糙集与粒计算。

赵成兵（1987-），男，重庆铜梁人，西南交通大学硕士生，主要研究方向为云计算与数据挖掘。

高子喆（1987-），男，陕西宝鸡人，西南交通大学硕士生，主要研究方向为云计算与数据挖掘。