基于MapReduce的MIC算法并行化

吕 瑞 蔡国永 裴广战

(桂林电子科技大学广西可信软件重点实验室 桂林 541004)

摘 要 MIC是一种分析变量之间可能存在的关系的方法。该方法不仅能够有效识别出变量间各种复杂类型的关系,还能够准确描述噪音数据对存在关系的影响,对探索大数据集中变量之间的关系具有重要意义。针对该方法在处理包含大量变量的数据集时性能方面的不足,首次对它进行了基于MapReduce模型的并行化。提出的并行化方法首先对原算法进行更细颗粒度的划分,然后采用一种基于Map-Reduce-Map任务链的并行模型,该模型不仅有效地增加了并行的计算单元,还大大地降低了不必要的系统开销。最后,通过理论分析和实验验证得出,改进后的算法与原算法相比,在准确率方面具有等效性,运行速度大幅度提升且具有良好的可扩展性;实验同时指出了算法性能的提升与系统资源的关系。

关键词 大数据,MIC,关系挖掘,MapReduce,并行化

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.016

Parallelization of MIC Algorithm Based on MapReduce

LV Rui CAI Guo-yong PEI Guang-zhan

(Guangxi Key Lab of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract MIC is a kind of method to analyze the possible relationships existing between variables, which can not only effectively identify the various complex types of relationships, but also accurately describe the impact of noise on the relationships. Exploring variable relationships in the large data sets is considered significant to big data mining. Aiming at the shortage of performance in dealing with the data set containing a large number of variables, this paper proposed a parallelization method based on MapReduce. Firstly, a finer and smaller partition to the raw algorithm was conducted, and then a parallel model based on the task chain Map-Reduce-Map was adopted. The model not only effectively increases the parallel computing units, but also greatly reduces unnecessary consumption of system resource. Theoretical analysis and experimental verification demonstrate that the improved algorithm has the same accuracy as well as the original algorithm and a great improvment in terms of running speed. The relationship between speed-up ratio and the amount of process Map2 shows that our method has a good scalability in the aspects of system resources.

Keywords Big data, MIC, Relationship mining, MapReduce, Parallelization

随着信息技术的发展,基因学、物理学、地理学、政治经济学等很多应用领域在短时间内积累了大量的数据[1],其数据规模呈现出了爆炸式增长的趋势。在这些大规模的数据集中可能存在着数以千计的变量,变量之间又可能存在非常重要且未被发现的各种复杂类型关系,科学有效地发掘出这些关系对于进一步探测大数据中隐藏的规律具有非常重要的意义。这也是目前学术界热烈讨论的关于"大数据挖掘"的问题之一。

传统的分析变量之间关系的方法主要包括皮尔逊相关系数法(Pearson Correlation)、斯皮尔曼相关系数(Spearman Correlation)、信息熵(Information Entropy)[2]、互信息量(Mutual Information)[3,4]等方法。然而,这些方法在探测和识别两个变量之间的关系时存在两个明显的缺陷:(1)不具有一般性,不能准确探测多种复杂类型的二元关系,如非已知类型的函数关系和非函数关系;(2)与关系类型相关,统计量的值除了受到噪声的影响,也与关系的类型或者特征有关,从而无法根据得分准确判断变量间关系的强弱。例如,无噪声线性关系的互信息值为3.65,而正弦关系则为0.59,高噪声的线性关系和低噪声的正弦关系具有相似的互信息值。因此,这些统计量的结果不能用来准确判断和说明变量间未知的复杂关系。

针对这个问题,2011年底Reshef等人在《科学》杂志上发表的一篇文章中提出了一种很好的解决方法,该方法使用一个称为MIC(Maximal Information Coefficient)的统计量对变量间存在的关系强度进行度量[5]。该统计量可以探测多种未知的复杂关系类型。对于具有相同噪声强度的关系,MIC将给出一个与类型无关的近似相等的值;二元关系越清晰,MIC值越高。因此,借助MIC方法判断数据集中变量之间的关系具有重要的意义。为了确定变量之间关系的类型和特征,文中还基于MIC统计量进行了一些扩展并提出了一些其他的

到稿日期:2014-11-07 返修日期:2015-01-18 本文受广西自然科学基金(2011GXNSFA018156),研究生创新项目(GDYCSZ201464)资助。

吕 瑞(1988—),男,硕士生,主要研究领域为社会计算、云计算、推荐系统,E-mail:lxhg2001@gmail.com;蔡国永(1971—),男,博士,教授,主要研究领域为社会媒体挖掘、可信网络计算;裴广战(1987—),男,硕士生,主要研究领域为网络信息传播、数据挖掘。

统计量(如 MINE,MIC-ρ2 等),这些统计量有助于分析变量间关系的具体特征(比如单调性,对称性等)。然而,当数据集中包含的变量个数为 n 时,为了求得任意两个变量之间的关系强度,需要对原算法进行 $n(n-1)/2$ 次迭代,这通常被认为是具有很高的算法时间复杂度,不适合用来处理变量数很多(如大于 1000)的大规模数据集,而这种规模的数据集在基因学、社会学、网络空间等领域变得越来越普遍[6-8]。因此,基于 MapReduce 的 MIC 并行算法研究非常必要。本文提出了一种基于 MapReduce 并行框架的改进算法,该方法首先对任务进行合理的更细粒度的划分,以增加可以并行计算的单元,然后采用一种基于链式的 MapReduce 模型[9]对算法进行优化和改进,以及采用一种均匀的子任务分发策略,从而充分利用系统的资源,降低系统开销,提升算法性能。

1 MapReduce 编程模型

MapReduce 是一种具有高效任务调度能力的分布式编程模型[10],采用"分而治之"的策略对大规模数据集进行任务的划分和算法的并行化。MapReduce 主要包含 Map(映射)和 Reduce(归约)这两个基本的并行单元。首先将分割开来的相互独立的数据块文件通过 Map 过程进行高度的并行处理和计算,再通过 Reduce 过程将结果进行汇集整理并返回输出。MapReduce 的基本模型如图 1 所示。

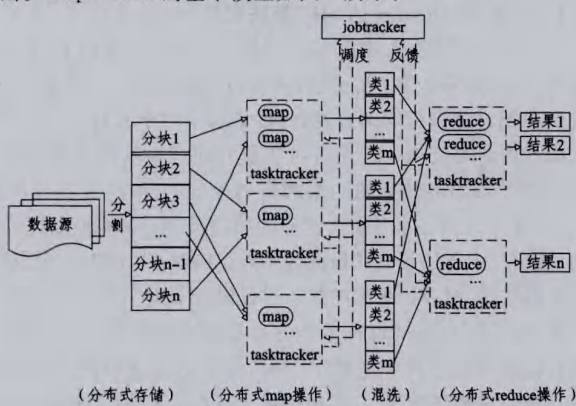


图 1 MapReduce 基本模型

MapReduce 的控制流如图 1 中虚线所示,主要由作业调度总控进程 jobtracker 和执行任务的守护进程 tasktracker 组成。jobtracker 针对数据分块在分布式文件系统中的位置与集群中相应的主机进行通信并调用主机上的 tasktracker 守护进程创建 JVM 来执行 Map 过程;同时 tasktracker 将 Map 任务执行的反馈信息及时报告给 jobtracker,jobtracker 根据 Map 任务的执行情况与空闲主机中其它 tasktracker 进程进行通信,从而创建新的 JVM 执行 Reduce 任务。

MapReduce 的数据流如图 1 中实线所示。分块中的数据以〈*key*,*value*〉的形式读入到相互独立、高度并行的 Map 过程中进行处理,同时产生新的〈*key*,*value*〉,在混洗过程(shuffle)中 jobtracker 将具有相同 *key* 值的键值对进行合并,从而生成〈*key*,{*value*1,*value*2…}〉形式的新键值对并传递到空闲主机的 Reduce 端进行进一步的处理,最后产生最终的〈*key*,*value*〉键值对并输出。

MapReduce 的并行特性主要体现在 Map 过程和 Reduce 过程。Map 可以将大规模的原始数据进行分块处理,提取需要的数据特征,然后在 Shuffle 阶段[11]对具有相同 *key* 值的数据进行重新整合和归纳,从而构造新的并行运算单元,实现 Reduce 过程再度并行化。因此定义良好的 Map 过程和较好的中间输出结果是实现下一步并行操作和将具有复杂依赖关系的算法进行并行化的关键所在。MapReduce 编程模型的优势主要包含以下 3 个方面:(1)它不仅能轻松地应用于大规模数据处理,而且能将很多繁琐的细节隐藏起来,比如:自动并行化、负载均衡[12]和灾备管理[13]等;(2)MapReduce 采用了无共享大规模集群系统,具有较好的可扩展性,可以通过简单地增加主机数目,实现计算性能的线性增长,而过去的大多数分布式处理框架在扩展性方面都与 MapReduce 相差甚远;(3)在性能方面,MapReduce 模型采用了推测执行、JVM 重用、分发代码等多项技术最大限度地利用了系统资源,保证了应用程序的安全高效执行。

2 MIC 算法

2.1 MIC 算法思想

MIC 算法的思想源于直观的观察。假设两个变量 X,Y 之间存在一定的关系,无论是函数的或者非函数的,也不管是单一的关系或者是多种关系的组合,都能够在这两个变量构成的散点图上用一个足够密的网格将这些关系点进行包裹,如图 2 中第三列对散点图进行的 x 行 y 列划分,并且总可以通过改变网格中分割点的位置使坐标点尽可能落在少量的单元格内,使其具有最大的互信息量。网格也可以看作是由随机变量 X,Y 诱导出的一个二维联合概率分布,每个单元格代表一个概率值,其大小等于该单元格中坐标点的数目与散点图中所有点总数的比值。从而,x 行 y 列网格的互信息量 $I(X;Y)$ 的计算表达式为:

$$\begin{aligned} I(X;Y) &= H(X)+H(Y)-H(X,Y) \\ &= \sum_x \log\frac{1}{p(x)}+\sum_y \log\frac{1}{p(y)}+\sum_{x,y} p(x,y)\log p(x,y) \\ &= \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \end{aligned}$$

其中,$H(X,Y)$ 是随机变量 X,Y 的联合熵,$H(\cdot)$ 为边缘熵,$p(x,y)$ 是第 x 行 y 列的单元格的概率,$p(x)$ 和 $p(y)$ 分别表示第 x 行和第 y 列的概率。

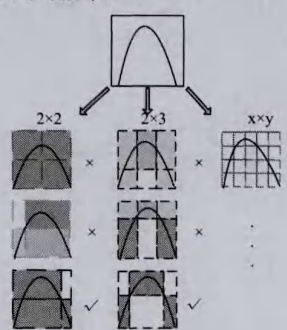


图 2 二维网格划分

如图 2 所示,对散点图基于不同数目的行和列进行划分(如 2×2,2×3 等),可以分别得到相应的具有最大互信息量的最优网格(图 2 中用"√"标记)。将所有的最优网格以矩阵的形式进行排列可以得出如图 3 所示的特征矩阵,特征矩阵的行表示对散点图进行网格化时指定的行数,列表示网格化时指定的列数。

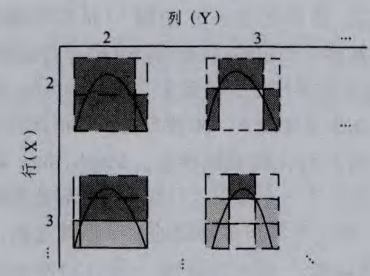


图3 散点图的特征矩阵

基于已构建的特征矩阵，MIC 的值定义为特征矩阵中的最大值，即：

$$MIC=\max\{m_{xy}\mid m_{xy}=\frac{\max\{I_G\}}{\log\min(x,y)}\text{且}\ xy<n^{0.6}\}$$

其中，I_G 是散点图中所有 x 行 y 列网格的互信息值的集合；m_{xy} 表示特征矩阵中 x 行 y 列元素的值；$xy<n^{0.6}$ 给出了 x 和 y 的范围，n 为散点图中点的个数。

2.2 确定初始分割点

假设对散点图进行 x 行 y 列的划分，如何确定分割点的位置从而使划分后的网格的互信息量达到最大是一个核心问题。针对这个问题，原文中作者给出了一种称为 ApproxMaxMI 的基于动态过程的启发式逼近算法。假设由两个变量构成的坐标点集存在于一个二维平面直角坐标系中(X 轴，Y 轴分别代表两个变量)，如图 4 所示。该算法首先将 X 轴固定分割为 x 行，然后基于分割后的 X 轴诱导出对 Y 轴的原始分割，最后通过优化算法选择 Y 轴原始分割中的 y 个分割点构成子分割，从而使优化后 x 行 y 列网格具有最大的互信息值。同理，先将 Y 轴固定分割为 x 行，再基于已分割的 Y 轴诱导出对 X 轴的原始分割，并选择最优的 y 个分割点构成最优的网格划分，得出第二个 x 行 y 列网格的最大互信息值。最后，选取互信息值较大的那个分割方法作为最终基于 x 行 y 列划分的最优分割方法并获得最大互信息值 m_{xy}。

上述方法中所述的固定分割方法和诱导分割方法的示意图如图 4 所示。假设以 X 轴进行固定分割(设分为 x 行)，以 Y 轴为诱导分割(设分为 y 列)。固定分割的方法首先将所有坐标点按 X 值的大小进行排序，并计算每个 X 值对应的坐标点的个数(即 X 值相同，Y 值不同)，然后在保证所有 X 值相同的坐标点处在同一行的情况下，尽可能地将所有的点均匀分布在所有行中(图 4 中每行包含 4 个点)，从而使基于行分割的边缘熵达到最大。基于已经分好的行，诱导分割的原始分割点位于所有两个相邻的并且落在同一行的点之间，如图 4 中 A,B,C,D 4 个彼此相邻的点位于同一行，所以有原始分割点 y_3，y_4，y_5 与之对应，而点 E,F 位于不同的行，则不需要建立分割点。假设 Y 轴有 $t(y\leqslant t)$ 个原始分割点，则可以建立的基于 Y 轴子分割点的 x 行 y 列网格共 C_t^y 种。计算每一种网格的互信息量，从而构成基于 x 行 y 列分割的互信息值的集合 I_G'。交换 X 轴和 Y 轴的计算顺序，得出另一个互信息值的集合 I_G''。则

$$I_G=\max\{\max\{I_G'\},\max\{I_G''\}\}=\max\{I_G',I_G''\}$$

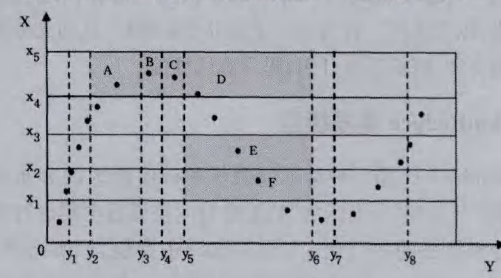


图4 散点图的固定分割和诱导分割

3 基于 MapReduce 的 MIC 算法并行化

3.1 基于 MapReduce 的 MIC 算法思想

结合 MIC 算法的思想和 MapReduce 的并行框架模型[14,15]，首先将 MIC 算法进行更细粒度的划分：(1)将多个变量自动配对并进行均匀分发，保证集群中每个并行的计算单元都有均衡的负载；(2)为每对变量分别产生基于 X 轴固定分割和 Y 轴固定分割的初始网格划分，进一步扩大并行度；(3)对每一个固定分割和其诱导的原始分割计算最优的子分割，输出最大互信息值和最优分割点。同时，为了进一步提高算法的性能，对并行算法的执行过程进行了进一步优化，采用基于 Map-Reduce-Map 任务链的并行模型对算法进行改进，使其保证所有子任务能够在一次 job 任务提交中完成，减少不必要的系统开销；同时采用分布式缓存技术，将少量临时的中间结果进行缓存，减少对分布式文件系统的 I/O 操作。综合以上考虑，本文中提出的基于 MapReduce 的并行化方法的框架模型如图 5 所示。

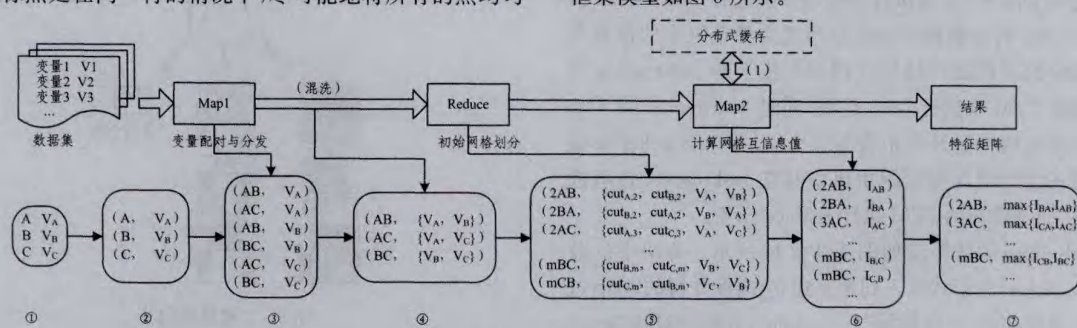


图5 基于 MapReduce 的并行过程

图 5 的第一行表示了本文使用的基于链式 MapReduce 的任务流程图；第二行则通过一个简单的例子演示框架模型中数据的变化过程。为了描述方便，假设数据集中的变量 $key=\{k_i\mid i\in[1,n]\text{且}(k_i<k_j\ \text{iff}\ i<j)\}$，$key$ 对应的 $value=\{v_1,v_2,\cdots,v_n\}$(状态①)。Map1 函数从数据集中读取 $\langle k_i,v_i\rangle$ 并产生一个键值对集合 M_i，其中 $M_i=\{\langle k_s k_t,v_i\rangle\mid(t=i,i>1\text{且}\ s\in[1,i-1])\text{或}(s=i\text{且}\ t\in[i+1,n])\}$(状态③)。对于集合 M_i 中的每个键值对，将其 key 值与最大整数进行与操作，然后和 Reduce 的进程数求余(即 $(key\&Integer.MAX\_VALUE)\%numReduce$)，从而确定该键值对将被哪一个

Reduce进程处理并使每个 Reduce 进程获得近似相等数量的键值对(过程 Shuffle),具有相同 *key* 值的键值对将会被送到同一个 Reduce 进程并形成新的键值对$\langle k_ik_j,\{v_i,v_j\}\rangle$(状态④)。假设将散点图最多划分为 *MAX_Y* 列,则其最多被划分的行数为 $MAX\_X=\lfloor n^{0.6}/MAX\_Y\rfloor$,在 Reduce 过程使用2.2节中的方法分别构建基于变量 k_i(以 k_i 变量所在的轴为X轴)的 $x(x\in[2,MAX_X])$行固定分割和基于 k_i 固定分割 k_j 的原始诱导分割,以及基于变量 k_j(以 k_j 变量所在的轴为X轴)的 $x(x\in[2,MAX_X])$行固定分割和基于 k_j 固定分割 k_i 的原始诱导分割,并分别产生 MAX_X-1 个新的键值对$\langle xk_ik_j,\{cut_{i,x},cut_{j,x},v_i,v_j\}\rangle$和$\langle xk_jk_i,\{cut_{j,x},cut_{i,x},v_j,v_i\}\rangle$$(x\in[2,MAX\_X])$(状态⑤),其中 xk_ik_j 和 xk_jk_i 分别表示X轴被分割为 x 行时的变量对 k_ik_j(k_i 为X轴,k_j 为Y轴)和 k_jk_i(k_j 为X轴,k_i 为Y轴);$cut_{i,x}$和$cut_{j,x}$分别表示当X轴被分割为 x 行时变量 v_i 所在的轴的分割点组成的向量和 v_j 所在轴的分割点组成的向量,从而下一步的并行度将再提高$2*(MAX\_X-1)$倍;最后通过 Map2 确定任意两个变量之间的最优子分割,获得键值对$\langle xk_ik_j,I_{ij}\rangle$(状态⑥),其中 I_{ij} 是一个包含 MAX_Y-1 个元素的向量,表示分割行数为 x、列数为 $y(y\in[2,MAX\_Y])$时所有可能网格的最大互信息值。如果$\langle xk_jk_i,I_{ji}\rangle$存在于分布式缓存,则输出键值对$\langle xk_ik_j,\max\{I_{ij},I_{ji}\}\rangle$,否则将键值对写入分布式缓存,其中 $\max\{I_{ij},I_{ji}\}$ 表示由向量 I_{ij},I_{ji} 中相同位置上较大的值构成的新向量。则 MAX_X 个键值对$\langle xk_ik_j,I_{ij}\rangle$$(x\in[2,MAX_Y])$即可构成行数为 MAX_X、列数为 MAX_Y 的特征矩阵(1行1列时单独由公式计算获得),从而变量 k_ik_j 的 MIC 值即为矩阵中的最大值。

3.2 基于 MapReduce 的最优子分割

上文对 Map1 和 Reduce 过程进行了详细说明,本节重点介绍基于 MapReduce 算法的另一个重要过程 Map2,并通过伪代码展示算法的基本思路。该过程假设已对散点图进行了 x 行 k 列的原始分割,从而在 k 个列分割中找出 $y(y\in[2,MAX_Y])$个分割点使组成的 x 行 y 列新网格在所有可能的 x 行 y 列网格中具有最大互信息值。

假设 v_x 和 v_y 是两个变量,相应的坐标值为 val_x,val_y,建立以 v_x 为X轴,v_y 为Y轴的散点图,如图3所示。且已知对X轴进行的 x 行固定分割为 $cutx=\langle x_0,x_1,\cdots,x_x\rangle$,X轴进行的诱导分割为 $cuty=\langle y_0,y_1,\cdots,y_k\rangle$,$x_i$,$y_i$ 为分割点,且 $y\leqslant k$。其中$\langle y_0,y_s,y_t\rangle$表示Y轴的3个分割点,$N(y_t)$表示第 t 个分割点$(0,y_t)$所在的分割线之前的坐标点的个数;P_{ty} 表示从Y轴前 t 个分割点中选择的 y 个最优分割点,I_{ty} 表示从Y轴前 t 个分割点中选择的 y 个最优分割点后生成的网格的互信息值,其中:

$$F(s,t,y)=\frac{N(y_s)}{N(y_t)}(I_{s,y-1}-H(cutx))-\frac{N(y_t)-N(y_s)}{N(y_t)}H(\langle y_s,y_t\rangle,cutx)$$

表示从前 t 个分割点中找出 y 个最优分割点并借助第 s 个分割点进行迭代。

基于以上假设和已知条件,Map2 过程用伪代码表示如下:

Map2 算法:求解最优子分割
输入:$\langle v_xv_y,\{cutx,cuty,val_x,val_y\}\rangle$
MAX_Y:最大分割的列数
k:向量 cuty 的维数,即初始的分割点数
输出:$\langle v_xv_y,Iv_xv_y\rangle$
1. for t=2 to k do
2. Find s∈{1,…,t}
max {H(⟨y_0,y_s,y_t⟩)−H(⟨y_0,y_s,y_t⟩,cutx)}
3. $P_{t,2}$←⟨y_0,y_s,y_t⟩
4. $I_{t,2}$←H(cutx)+H($P_{t,2}$)−H($P_{t,2}$,cuty)
5. end for
6. for y =3 to MAX_Y do
7. for t=y to k do
8. Find s∈{y−1,…,t} maximizing
$F(s,t,y):=\frac{N(y_s)}{N(y_t)}(I_{s,y-1}-H(cutx))-\frac{N(y_t)-N(y_s)}{N(y_t)}H(\langle y_s,y_t\rangle,cutx)$
9. $P_{t,y}$← $P_{s,y-1}\cup y_t$
10. $I_{t,y}$←H(cutx)+H($P_{t,y}$)−H($P_{t,y}$,cuty)
11. end for
12. end for
13. $Iv_xv_y=(I_{k,2},\cdots,I_{k,MAX\_Y})$
/* $I_{k,j}$为 x 行 j 列分割时最大互信息值,j∈[2,MAX_Y] */
14. return Iv_xv_y

4 并行 MIC 算法实验分析

为了测试改进后 MIC 方法的性能,申请了台湾高速网络与计算中心提供的 Hadoop 云计算环境。在实验时,该系统的配置为 1 个主节点,10 个从节点,配置 Intel Core Quad Q6600 2.40Hz CPU,8GB 内存;操作系统内核版本为 Linux 2.6.26-2,Java 版本为 1.6.0_26,Hadoop 版本为 0.20.0。

4.1 算法等效性

为了验证改进后的算法和原算法在结果上的等效性,从 WHO 数据集中随机抽取了 8 组变量(如图 6 所示),每组变量包含 203 个点,计算改进前后算法的 MIC 值,如图 6 所示。

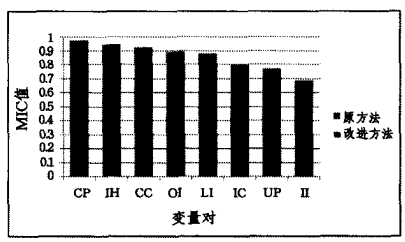


图 6 算法改进前后的对比

从图 6 中可知,改进后算法和原算法的实验结果完全相同,证明了改进算法和原算法具有等效性。因此,改进算法可以被应用在基于 MapReduce 并行框架的云平台(如 Hadoop)上进行大规模数据集中大量变量之间的关系挖掘。

4.2 算法性能分析

为了分析并行化后算法的性能,进行了以下 4 组实验,如图 7 所示。实验(a)和(b)测试算法运行时间与数据集大小(变量个数、坐标点个数)的关系;实验(c)测试算法运行时间与分布式系统资源(Map 和 Reduce 进程数)的关系;实验(d)则表明了算法的加速比与系统资源(Map/Reduce 数)的关系。由于 4 组实验主要用来测试算法性能,因此实验中采用在坐标平面上产生随机点的方式为不同变量产生不同规模的仿真数据集。其实验结果如图 7 所示。

(下转第 103 页)

j_{t+1}，有$\langle q_i, j_1, r_1\rangle \in TOP, \langle r_1, j_2, r_2\rangle \in TOP, \cdots, \langle r_t, j_{i+1}, p_0\rangle \in TOP$。这样，$q_i$ 有两个不同的后继 r_1 和 q_{i-1}（如果 $i<m$）或者 r_1 和 q_1（如果 $i=m$），这与定理 1 矛盾。

定理 4 归约算法执行后，集聚进程持有所有进程中的数据经过⊕运算后的结果。

证明：运用定理 2 以及⊕运算的交换律和结合律即可得到结果。

结束语 由于造成归约算法相互不同的根本原因在于逻辑拓扑的不同，为了揭示归约算法的根本共性，首先形式地给出了归约操作逻辑拓扑的定义并得到该定义的基本性质。在此基础上，给出了一个归约算法的统一描述。描述将所有基于树型的多到一归约算法纳入到同一个框架中，这有助于人们对归约算法的理解，因为它提供了另一种分析归约算法的视角。实际上，该描述也是一个归约算法的形式定义。

参 考 文 献

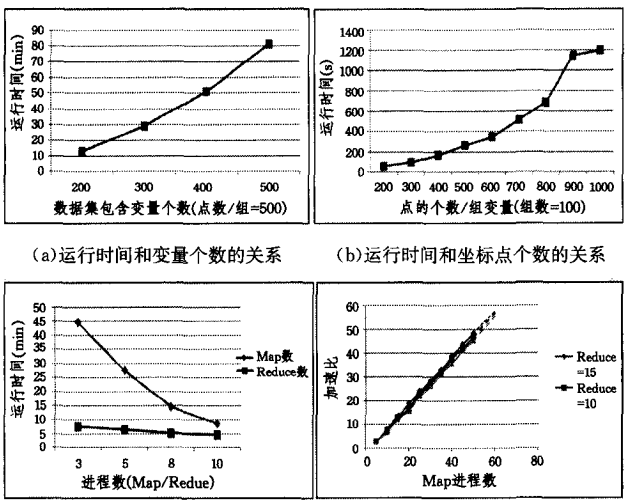
[1] Balaji P, Kimpe D. On the Reproducibility of MPI Reduction Operations: ANL/MCS-P4093-0713[R]. Argonne, IL, USA: Argonne National Laboratory, 2013

[2] Thakur R, Rabenseifner R, Gropp W. Optimization of Collective Communication Operations in MPICH: ANL/MCS-P1140-0304 [R]. Argonne, IL, USA: Argonne National Laboratory, 2004

[3] Rabenseifner R. New optimized MPI reduce algorithm [OL]. http://www.hlrs.de/organization/par/services/models/mpi/myreduce.html

[4] Dongarra J J, Whaley R C. A User's Guide to the BLACS v1.0.1: UT CS-95-281[R]LAPACK Working Note #94, University of Tennessee, 1995

（上接第 83 页）



(a)运行时间和变量个数的关系　(b)运行时间和坐标点个数的关系

(c)运行时间与系统资源的关系　(d)进程数与加速比的关系

图 7　算法性能分析

从实验(a)、(b)可知，在相同系统资源下，算法的运行时间与数据集中的变量个数正相关，与每组变量中包含的点个数正相关。从实验(c)可知，算法在相同数据规模时，系统资源越多，算法运行的时间越短，且 Map 进程数的改变对算法运行时间的影响更明显。实验(d)显示了改进后算法的加速比与系统资源的关系，由图可知，在相同数据规模时并行化后算法的加速比与 Map 进程数(并行度)呈线性增长的关系，而 Reduce 进程数则影响较弱，从而并行算法中 Map 任务的并行度对算法性能的提升起到了主要作用。实验(d)中虚线部分表示对线性关系趋势的预测，说明了该算法具有良好的可扩展性，能够随着集群中并行计算单元数的增加而得到有效的扩展。

结束语 鉴于 MIC 算法在处理包含大量变量数据集时算法性能方面的不足，对其进行了基于 MapReduce 模型的并行化，并用实验证明改进后算法具有较好的可扩展性和优良的加速比，同时说明了改进后算法与系统资源、数据规模的关系。

参 考 文 献

[1] Zhou A Y. Data intensive computing-challenges of data management techniques[J]. Communications of CCF, 2009, 5(7): 50-53

[2] Białynicki-Birula I, Mycielski J. Uncertainty relations for information entropy in wave mechanics [J]. Communications in Mathematical Physics, 1975, 44(2): 129-132

[3] Wells III W M, Viola P, Atsumi H, et al. Multi-modal volume registration by maximization of mutual information[J]. Medical Image Analysis, 1996, 1(1): 35-51

[4] Batina L, Gierlichs B, Prouff E, et al. Mutual information analysis: a comprehensive study[J]. Journal of Cryptology, 2011, 24(2): 269-291

[5] Reshef D N, Reshef Y A, Finucane H K, et al. Detecting novel associations in large data sets[J]. Science, 2011, 334(6062): 1518-1524

[6] Labrinidis A, Jagadish H V. Challenges and opportunities with big data [J]. Proceedings of the VLDB Endowment, 2012, 5(12): 2032-2033

[7] McAfee A, Brynjolfsson E, Davenport T H, et al. Big Data[J]. The management revolution. Harvard Bus Rev, 2012, 90(10): 61-67

[8] Lohr S. The age of big data[Z]. New York Times, 2012, 16(4): 10-15

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

[10] Wang H, Huang W, Zhang Q, et al. An improved algorithm for the packing of unequal circles within a larger containing circle [J]. European Journal of Operational Research, 2002, 141(2): 440-453

[11] White T. Hadoop: The definitive guide [M]. O'Reilly Media, Inc., 2012

[12] Groot S, Kitsuregawa M. Jumbo: Beyond MapReduce for workload balancing [C] // 36th International Conference on Very Large Data Bases. Singapore, 2010

[13] Zheng Q. Improving MapReduce fault tolerance in the cloud [C]// 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). IEEE, 2010: 1-6

[14] Lee K H, Lee Y J, Choi H, et al. Parallel data processing with MapReduce: a survey[J]. ACM SIGMOD Record, 2012, 40(4): 11-20

[15] McKenna A, Hanna M, Banks E, et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data[J]. Genome research, 2010, 20(9): 1297-1303