基于多核并行的海量数据序列模式挖掘 *

俞东进, 郑苏杭, 李万清
(杭州电子科技大学 计算机学院, 杭州 310018)

摘　要: 为了在多核处理器上充分利用多核资源以提升挖掘性能,提出了一种动态与静态任务分配机制相结合的基于多核的并行序列模式挖掘算法。该算法采用数据并行与任务并行相结合的策略,在各处理器核生成局部序列模式后,再与其他处理器核协同,以最终获得所有的全局序列模式。算法通过并行局部归约技术消除了局部序列的重复生成与计算,并可结合静态与动态任务分配机制解决处理器的负载不均衡问题。理论分析和实验都证实了该算法可有效利用多核计算平台及多核体系结构优势,具有较高的运行效率和加速比。

关键词: 并行; 多核; 序列模式; 海量数据挖掘
中图分类号: TP391　　**文献标志码**: A　　**文章编号**: 1001-3695(2012)02-0478-04
doi:10.3969/j.issn.1001-3695.2012.02.020

Parallel massive mining of sequential patterns based on multi-core processors

YU Dong-jin, ZHENG Su-hang, LI Wan-qing
(*School of Computer*, *Hangzhou Dianzi University*, *Hangzhou* 310018, *China*)

Abstract: To fully utilize the multi-core resources on multi-core processors to improve mining performance, this paper presented a novel algorithm of mining parallel sequential patterns based on the multi-core processors. It combined the data parallelism and task parallelism, with global sequential patterns obtained by combining local patterns discovered in different processor cores. Through local parallel reduction, it eliminated the repetitive patterns and reduced computational effort. Besides, it achieved the workload balancing by static and dynamic task distribution mechanisms. Both theoretical analysis and practical experiments show that the algorithm takes good advantage of multi-core computing platform, having higher operating efficiency and speedup.

Key words: parallel; multi-core; sequential patterns; massive data mining

0　引言

序列模式挖掘(sequential pattern mining)或称序列挖掘是数据挖掘中的一个重要研究方向,最早是由 Agrawal 等人[1]在针对超市购物篮数据的分析中提出。其主要用于找出带有时间特征的序列数据集(如文本中的单词序列、Web 日志文件中的用户访问事件序列,以及生物数据库中的 DNA 序列和蛋白质序列)中所有超过最小支持度阈值的序列模式[2]。目前,序列模式挖掘已被广泛应用于各个领域,如客户购买行为模式的分析、Web 访问模式的预测、疾病诊断、自然灾害预测、DNA 序列分析等[3,4]。

序列挖掘一般涉及 TB 级的海量数据,巨大的运算量以及繁重的 I/O 操作使得传统的单机系统在功能和性能上都难以达到预期效果。近年来不少学者开始研究如何借助分布式计算技术提升海量数据序列模式的挖掘性能。最初的研究主要集中于如何在并行环境下扩展传统的串行算法。例如,基于 Apriori-like 串行算法,Shintani 等人[5]提出了三种并行策略:MPSPM、SPSPM 和 HPSPM。它们通过遍历数据集发现候选序列并继而利用 Apriori 特性进行剪枝以得到频繁序列;在 SPADE 串行算法基础上,Zaki[6]提出了基于共享存储并行结构的并行算法:pSPADE。该算法利用格技术和简单的连接方法来挖掘频繁序列模式,但是需要事先将序列数据库转换为垂直数据库格式。在序列挖掘的多任务分配方面,韩家炜等人提出基于 BIDE[7]的分布式闭合序列模式挖掘算法 Par-CSP,通过将伪投影分配给不同的处理器以实现并行化[8,9];Buehrer 等人[10]提出的基于多核的并行频繁序列模式挖掘算法则采用深度优先方法将搜索任务分配给各处理器核并行执行;Lucchese 等人[11]提出的闭合并行频繁项集挖掘算法可解决并行算法中负载不均衡的问题。此外,还包括 Valerie 等人在树投影基础上提出了基于分布存储的 STPF 并行算法[12],通过数据库投影方法使下一次遍历的数据库更小;邹翔等人[13]提出的分布式序列模式挖掘算法 FDMSP,通过采用前缀投影技术划分模式搜索空间使算法具有较低的 I/O 开销、内存开销和通信开销;Tatikonda 等人[14]提出的并行频繁树挖掘算法部分解决海量序列模式搜索的存储受限问题等。

一般认为,针对某些简单的模式查找问题,比如直接统计海量数据集中某些字段同时以相同值出现的次数,在关注序列模式挖掘算法本身的同时,更应关注于如何解决数据存储及如何提高运算速度问题。多核平台下的并行程序设计是一种解决方式,然而由于从单核到多核并不像处理器时钟频率的提升

收稿日期: 2011-07-20; **修回日期**: 2011-08-25　**基金项目**: 国家自然科学基金资助项目(60903053);浙江省重大科技计划资助项目(2008C11099-1)

作者简介: 俞东进,男,教授,硕导,博士,主要研究方向为数据挖掘、数据库技术、软件工程;郑苏杭,女,硕士研究生,主要研究方向为数据挖掘、数据库技术(zhengsuhang666@sina.com.cn);李万清,男,讲师,主要研究方向为并行计算.

那样对程序员而言是透明的, 如果没有针对多核特性设计程序, 往往不能完全获得多核带来的性能提升。目前已有多种多核平台下的并行编程方法和技术,大都是基于多线程编程思想,通过创建多个线程将一个规模较大的任务划分为几个小任务分配到多个核上同时并行执行,提升处理器的利用率。当前比较流行的是结合共享存储编程工具 OpenMP[15] 和基于 TBB (threading building blocks) 的多线程并行编程模型[16]。OpenMP 支持 C、C++或 Fortran 语言,而 TBB 主要是在现有语言基础上对并行开发概念的封装和实现,支持 Intel C++、VC 7/8 和 gcc 编译器。由于编译型语言往往通过与平台相关的库来提供多线程支持, 虽然这些库提供了一定程度的封装,减轻了程序员多线程编程的负担,但使用这些库会增加一些关键代码及相应的调试和测试成本,且由于没有统一标准,编写多线程程序需要考虑更多的细节, 提高项目成本。因此,对于目标定位于提升计算密集型程序性能的多核设计来说, 使用这些编程方式仍然较复杂。

与上述研究不同,本文针对数据共享存储及多核体系结构等特点,基于 Java 多线程技术,提出了一种新的、动态与静态任务分配机制相结合的基于多核的并行序列模式挖掘算法 SMMC(sequential mining based on multi-cores)。为在某些应用环境下直接根据候选序列及其出现的次数来挖掘知识,算法未采用闭合序列模式挖掘[7,17]思想,需保存候选序列且将关注点集中在核与核之间的协作、共享资源的分配等问题。SMMC 算法不同于以往的大多数序列模式挖掘算法通过首先挖掘所有的频繁 1 序列,然后再重扫描数据集来挖掘频繁 $N(N\geqslant 2)$ 序列,而是先确定在时间间隔内的最大长度序列集,由此集直接产生子序列集来挖掘所有的序列模式。算法通过采用数据并行与任务并行相结合的策略及动态和静态任务分配机制,可有效地解决传统多核环境下因程序结构设计不合理所造成的诸如存储受限和时效性、计算负载不均衡、I/O 开销繁重等问题。

1　相关定义

序列模式挖掘属于典型的计算密集型问题[18],其侧重点在于分析数据间的前后序列关系。为了讨论问题方便,本章对文中涉及的相关概念定义如下:

定义 1　项(item)。发生事件的属性值。设事件 A 有属性 $a_1,a_2,\cdots,a_k$,则 $a_i(1\leqslant i\leqslant k)$ 为项。

定义 2　项集(itemSet)。若干个项(item)组成的非空集合,表示为 $I=\{a_1,a_2,\cdots,a_k\}$,其中 $a_i(1\leqslant i\leqslant k)$ 是项集中的项,也称为项集中的元素(element)。

定义 3　序列(sequence)。不同项集(itemSet)的有序排列,表示为 $S=\langle I_1,I_2,\cdots,I_m\rangle$。其中 $I_i(i=1,2,\cdots,m)$ 为非空项集,也称为序列中的一个元素。序列的长度指一个序列中包含的所有元素的个数,表示为 $l=\sum_{1\leqslant i\leqslant m} I_i$ 。长度为 k 的序列称为 k-序列。项集可以看做是长度为 1 的序列。

定义 4　序列数据集(sequence dataSet)。序列数据集 D 是元组 $\langle S_{id},S\rangle$ 的集合,其中 S_{id} 是对应序列的序列号。序列数据集中元组的个数称为该序列数据集的大小,记为 L 。

定义 5　支持度(sup)。一个序列 S 在序列数据集 D 中的绝对支持度指 D 中包含 S 的元组数目,记为 $\sup(S)$;相对支持度指 D 中包含 S 的元组在整个数据集元组中所占的百分比,即 $\sup(S)/L$ 。本文若不特别申明,均指绝对支持度。在并行计算环境下,某节点 $p_i(i=1,2,\cdots,n)$ 上包含 S 的数据序列总数称为 S 在节点 p_i 上的局部支持度,记做 $\sup_i(S)$;所有节点上包含 S 的数据序列总数之和称为 S 的全局支持度,记做 $\sup(S)=\sum_{i=1}^{n}\sup_i(S)$。

定义 6　频繁 k-序列。给定序列数据集 D,若 D 中出现的 k-序列的支持度大于等于最小支持度阈值 min_sup,则称此 k-序列为 D 上的频繁 k-序列。

定义 7　加速比。假设序列模式挖掘的串行执行时间为 T_s,使用 q 个处理器核并发执行的时间为 $T_p(q)$,则并行加速比为 $S_p(q)=T_s/T_p(q)$。

2　算法描述

为充分结合多核特性,SMMC 算法整体上通过三个阶段来搜索频繁序列模式:a)根据线程个数,用水平等间距静态投影方法均分全局数据集为各局部数据集,作为各线程的同等数据源;b)将局部数据集转换为局部序列数据集,并行搜索局部序列模式并结合局部归约方法以序列文件形式保存候选序列模式;c)结合动态任务分配机制方法并行汇总候选序列模式为频繁序列模式。SMMC 算法数据流如图 1 所示。

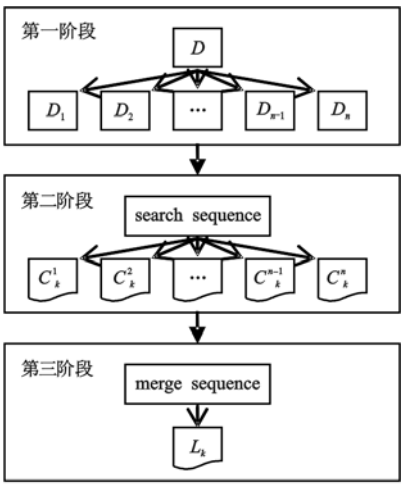

图1　SMMC算法数据流

算法描述相关符号定义如下:

P_i 为第 $i(i=1,2,\cdots,n)$ 个计算节点(处理器核)。

C_k 为所有候选 k-序列的集合。

$V_{C_k}^j$ 为所有候选序列值为 j 的 k-序列的集合。

L_k 为所有频繁 k-序列的集合,且 $L_k=\{C_k\mid\sup(C_k)\geqslant\text{min_sup}\}$。

D_i 为分配到第 i 个计算节点(处理器核)的序列数据集。

C_k^i 为分配给 P_i 的候选 k-序列的集合,且 $\sum_{i=1}^{n}C_k^i=C_k$。

$V_{C_k}^j$ 为分配给 P_i 的候选序列值为 j 的 k-序列的集合。

L_k^i 为从 C_k^i 中产生的频繁 k-序列的集合,且 $\sum_{i=1}^{n}L_k^i=\sum_{i=1}^{n}\{C_k^i\mid\sup(C_k^i)\geqslant\text{min_sup}\}=L_k$。

2.1　静态等分全局数据集

为了减少数据偏移的发生及实现多线程搜索局部序列模式,SMMC 算法首先采用水平等间距静态投影数据分解模式均分全局数据集以实现各线程数据负载均衡。假设全局数据集中有 R 条记录,采用水平等间距静态投影分配方法将完整的数据集划分为 n 份(n = 处理器的数量),使得各处理器核分配的数据集为 $R_1,R_2,\cdots,R_n(\sum_{i=1}^{n}R_i=R)$。其中第 i 个节点上对应的数据集 R_i 表示为

$$R_i = \left\{ M_i^j \mid M_i^j = T_p, p = \left\lfloor \frac{R}{n} \right\rfloor \times (i-1) + j, \right.$$
$$\left. p \in \left[\left\lfloor \frac{R}{n} \right\rfloor \times (i-1), \left\lfloor \frac{R}{n} \right\rfloor \times i \right] \right\} \tag{1}$$

其中：M_i^j 为第 i 个处理器核中的第 j 条记录，T_p 为全局数据集中的第 p 条记录。经过以上变换，全局数据集 D 被分割成 n 个规模为 $\left\lfloor \frac{R}{n} \right\rfloor$ 的局部数据集 $D_1, D_2, \cdots, D_n$，即 $D = \cup_{i=1}^{n} D_i$。

2.2　局部序列模式搜索、归约与保存

处理器核 p_i 获得局部数据集 D_i 后，先独立计算。为减少多核环境下有限存储带宽上的竞争，若 D_i 能够全部存储在内存中，则将 D_i 排序后形成的序列数据集写入内存；若 D_i 较大，不能全部存放，则对 D_i 进行二次划分，形成 D'_i，然后将 D'_i 排序后写入内存。针对当前的 MDB（memory database），算法搜索所有的 C_k^i 并存储在相应的数据结构中（图 2），随后依次查找下一个 D_i（D'_i），循环执行，直到再没有新的候选序列 C_k^i 产生为止。在此过程中，若生成的序列模式容量达到了预先定义的最大内存上限值，则先将该部分 C_k^i 以序列文件形式保存在本地硬盘，然后再计算并存储新的 C_k^i。整个过程中，处理器核 P_i 只需扫描一次 D_i 即可生成所有的局部 C_k^i。

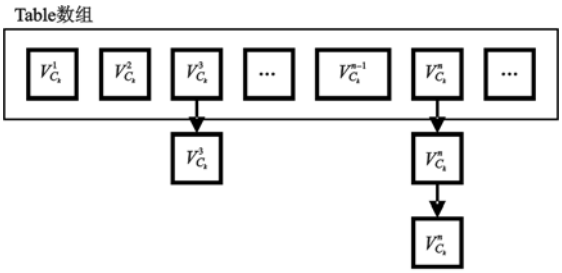

图2　序列模式存储数据结构

为了提高挖掘效率，减少 C_k^i 的数目和 L_k^i 的复杂度，算法结合局部归约技术来消除 C_k^i 的重复生成与计算。即处理器核 P_i 扫描 D_i 只保存序列值唯一的 $V_{Ci_k}{}^j$，将相同 $V_{Ci_k}{}^j$ 的候选项支持数进行累加。如针对处理器核 P_i，该归约方法具体实现过程如图 3 所示。

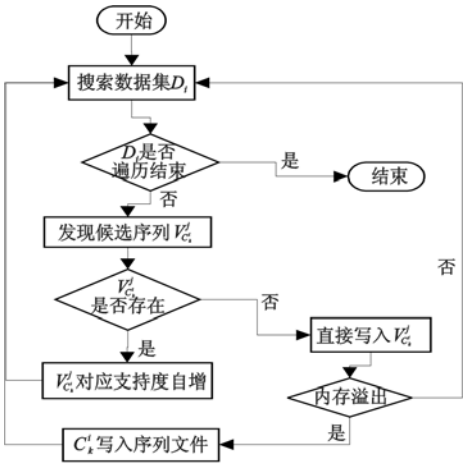

图3　候选序列归约流程

算法归约操作使得所有相同的 $V_{Ci_k}{}^j$ 位于同一链表中，从而快速实现将相同候选项的支持数进行累加，确保在 P_i 中得到的 C_k^i 不仅是 sup 为 1 的候选序列 C_k^i，而同时存在候选序列 C_k^i（sup = 1，2，⋯）。因此，算法的第一趟扫描将发现 sup 各异的候选 k-序列，然后第 n 趟扫描以第（$n-1$）趟的候选 k-序列的 sup 集合作为种子集来生成新 sup 的候选 k-序列，并将它作为下一趟扫描的种子集。如此循环执行，直到挖掘出所有的 C_k^i，并以序列文件保存。此外，为防止多核环境下对共享文件数据的读写或写写竞争，序列文件的命名采用以不同线程名区分的保护方法。

2.3　局部序列模式汇总

算法最终将 P_i 得到的局部序列模式与 P_j（$i \neq j$）交互，实现模式汇总，即把所有序列文件中相同候选项的支持度 sup 进行累加，从而获得全局序列模式。此阶段算法再次结合多线程编程的任务分解模式，通过采用动态任务分配机制，先把以不同文件编号开始的文件汇总任务依次保留在一个全局任务列表内，然后各线程同时处理分配到的任务，彼此相互独立、互不干扰。算法在一定的时间间隔内循环判断是否存在空闲处理器核，若存在，则从全局任务列表中读取一个新任务在此处理器核上进行处理。反复调用该方法，直到列表中所有的汇总任务被处理完毕，此时得到的所有 C_k^i 即为所挖掘出来的全局序列模式，最终将 C_k^i 的支持数与 min_sup 比较生成 L_k^i。为了减少数据汇总的计算量，当前任务的序列文件初始编号始终比上一汇总任务的文件初始编号增 1。

3　算法性能分析

假设全局序列数据集 D 采用水平等间距静态投影分配方法划分成局部数据集 $D_1, D_2, \cdots, D_n$ 的时间为 T_1。若处理器核 P_i 在数据集 D_i 中搜索第 j 个序列模式所耗费的时间为 Q_i^j（$j = 1, 2, \cdots, m$；$i = 1, 2, \cdots, n$），其中 m 表示 D_i 发现的序列模式总数，n 为处理器核总数，则处理器核 P_i 挖掘局部序列模式耗费的总时间为

$$T_i = \sum_{j=1}^{m} Q_i^j \quad i = 1, 2, \cdots, n \tag{2}$$

因此多核并行生成局部序列模式的总时间为

$$T_2 = \max(T_i) = \max\left(\sum_{j=1}^{m} Q_i^j\right) \quad i = 1, 2, \cdots, n \tag{3}$$

传统串行方法计算所有局部序列模式所需要的时间相当于每个处理器核单独处理的时间总和，即

$$T'_2 = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \sum_{j=1}^{m} Q_i^j \quad i = 1, 2, \cdots, n \tag{4}$$

在局部序列汇总阶段，处理器核 P_i 的汇总时间为

$$T_i = \sum_{i=q}^{k} S_i + t_i \quad q = 1, 2, \cdots, k; i = 1, 2, \cdots, n \tag{5}$$

其中：S_i 为存储候选序列的第 i 个序列文件；k 为 SMMC 算法产生的序列文件总数；t_i 为处理器核访问全局任务列表及重新划分任务等操作的系统开销，且 $t_i \approx \sum_{i=q}^{k} S_i$。所以多核耗费的总时间为

$$T_3 \approx \max(T_i) \approx \max\left(\sum_{i=q}^{k} S_i + t_i\right)$$
$$q = 1, 2, \cdots, k; i = 1, 2, \cdots, n \tag{6}$$

使用静态任务分配方式串行递归汇总数据文件所耗费的时间近似为

$$T'_3 \approx \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \left(\sum_{i=q}^{k} S_i + t_i\right)$$
$$q = 1, 2, \cdots, k; i = 1, 2, \cdots, n \tag{7}$$

综上，使用 SMMC 算法所耗的总时间为

$$T = T_1 + T_2 + T_3 =$$
$$T_1 + \max\left(\sum_{j=1}^{m} Q_i^j\right) + \max\left(\sum_{i=q}^{k} S_i + t_i\right)$$
$$q = 1, 2, \cdots, k; i = 1, 2, \cdots, n \tag{8}$$

传统的串行算法所耗费的总时间约为

$$T = T_1 + T_2 + T_3 =$$
$$T_1 + \sum_{i=1}^{n}\sum_{j=1}^{m} Q_i^j + \sum_{i=1}^{n}\left(\sum_{i=q}^{k} S_i + t_i\right)$$
$$q = 1,2,\cdots,k;i = 1,2,\cdots,n \tag{9}$$

由于 $\max\left(\sum_{j=1}^{m} Q_i^j\right) \leqslant \sum_{i=1}^{n}\sum_{j=1}^{m} Q_i^j$, $\max\left(\sum_{i=p}^{k} S_i + t_i\right) \leqslant \sum_{i=1}^{n}\left(\sum_{i=p}^{k} S_i + t_i\right)$,且 t_i 值较小,所以 SMMC 算法相对于静态任务分配方式及传统的串行方法性能提高显著,是针对多核环境所设计的一种合理的程序结构。

4　实验结果和分析

4.1　实验环境

SMMC 算法已成功应用于某市公安交警部门基于海量交通数据流的伴随车识别案例中。伴随车是指频繁地在一个给定的时间间隔内相伴出现的两辆及以上车辆。由于伴随车具有一定的违法犯罪嫌疑,如果能在海量的卡口过车数据流中快速识别出伴随车,将对公安、交警维护社会治安提供不小的助益。实验对伴随车定义为:假设 N 辆车在某一时间段 T 内相继或同时经过一个或多个监控点,若出现的次数大于等于预先设定的最小支持度阈值 min_sup,则称这 N 辆车为伴随车。实验设定 N 和 T 的值分别为 2 辆和 60 s,min_sup 为 3。

实验采用某城市道路监控点所捕获的过往车辆信息为测试源,主要包括车牌号、过车时间、过车方向和卡口地点,数据集规模为 300 万条记录左右。实验环境为:4 个 Intel Core 2 Quad Duo 处理器,主频 2.40 GHz,内存 4 GB,Windows XP 操作系统,所有算法的代码用 Java 语言编写,并在 Eclipse 3.5 环境下编译。

4.2　算法运行时间与结果比较

实验最终产生的频繁序列个数如图 4 所示。从图 4 可以看出,随着数据集规模的不断增大,算法所产生的频繁序列也不断增多。尽管频繁序列数目与数据集规模大小并非同比增长,但是依据已产生的频繁序列数目可以肯定,每个数据集对应生成的候选序列模式增长幅度较大,从而增加了序列文件归约的复杂性与计算量,最终延长了结果集的生成时间。

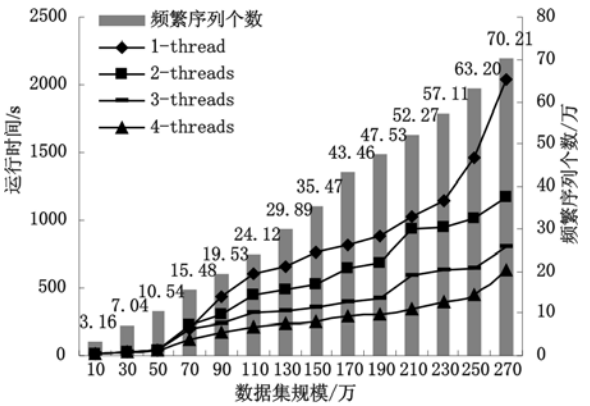

图4　不同数据集记录数和线程数的SMMC算法执行时间对比

由图 4 可知,当数据集规模较小时,SMMC 算法在不同线程个数下的运行时间差距不大;但当数据集规模大于等于 50 万条记录时,随着线程个数的增加,算法在各数据集上的执行时间都有了明显的减少,而且降幅相对稳定。由此可见,SMMC 算法通过多线程并发执行可有效利用多核计算环境提高性能。为了更直观地验证算法的有效性,实验给出了不同记录数及线程数下 SMMC 算法的运行时间和加速比,如图 5 所示。其中:T10 表示 10 万条过车记录,T90 表示 90 万条过车记录,以此类推。

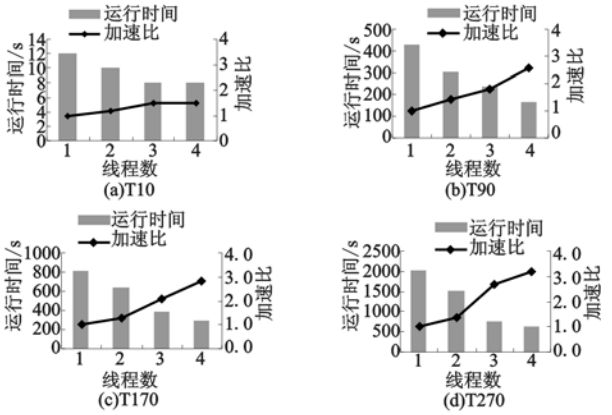

图5　不同数据集记录数SMMC算法运行加速比

由图 5 可知,当数据集记录数只有 10 万条时,算法的加速比随着线程数量的增加提升不明显,这主要是由于在数据量较少时,单线程已可在较短时间内完成任务;而当记录数达 90 万、170 万、270 万时,随着线程数的增加,算法可取得较为理想的运行加速比。因此,SMMC 算法采用的动态负载均衡策略能有效利用所有处理器核,减少了各核的闲置时间,并最终实现多核带来的性能提升。其在海量交通数据流中快速识别出的伴随车辆,为公安、交警部门作出科学决策提供了有力的理论基础,从而有效打击了机动车的违法犯罪行为。

5　结束语

本文提出了一种动态与静态任务分配机制相结合的基于多核的并行序列模式挖掘算法 SMMC。该算法的前半部分采用数据并行的静态分配思想,运用多线程使各计算节点独立处理所分配的数据;算法的后半部分采用动态任务分配机制,结合多核处理器特性,实现各核并行汇总局部序列模式。理论分析和实验结果证明,该算法可有效利用多核计算平台及多核体系结构优势,具有较高的运行效率和加速比。进一步的研究方向包括将算法灵活运用于分布式存储的数据集,进一步缩小搜索空间以提高算法效率,实现算法与多维序列模式挖掘、增量式序列模式挖掘的结合等。

参考文献:

[1] AGRAWAL R, SRIKAN R. Mining sequential patterns[C]//Proc of the 11th International Conference on Data Engineering. Los Alamitos: IEEE Computer Society,1995: 3-14.

[2] 荣冈, 刘进锋, 顾海杰. 数据库中动态关联规则的挖掘[J]. 控制理论与应用,2007,24(1):129-133.

[3] DONG Guo-zhu, PEI Jian. Sequence data mining[M]. Berlin: Springer-Verlag,2007.

[4] 朱扬勇,熊赟. DNA 序列数据挖掘技术[J]. 软件学报,2007,18(11):2766-2781.

[5] SHINTANI T, KITSUREGAWA M. Mining algorithms for sequential patterns in parallel: hash based approach[C]//Proc of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin: Springer-Verlag,1998: 283-294.

[6] ZAKI M J. Parallel sequence mining on shared-memory machines [C]//Proc of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Diego: AAAI Press, 1999: 57-65.

(下转第 521 页)

261.8%,显然 216% <261.8%,山东省规模以上工业经济效益比江西省要好,因此,基于最小机会损失值法则的组合评价法所得的结果更接近实际情况。又如,青海省的平均值法总排序为 19,模糊 Borda 法的排名为 17,最小机会损失组合评价法排名为 16,而平均值法排序为 16 的是安徽省,安徽省规模以上工业经济效益综合指数为 209%,广东省规模以上工业经济效益综合指数为 174.3%,而青海省规模以上工业经济效益综合指数为 331.55%,显然青海省应排序较为靠前。由以上分析可知,基于最小损失值法则的组合评价模型计算出的结果更接近实际情况,评价效果更好。

表 2　规模以上工业企业主要经济效益三种组合评价法结论

地区	最小机会损失	排序	平均值法	排序	模糊 Borda	排序
北京	0.379526	18	0.405338	17	179.8083	15
天津	0.363156	19	0.393311	18	111.8771	19
河北	0.477715	8	0.494986	8	240.5832	11
山西	0.23681	25	0.271053	24	43.2828	23
内蒙古	0.416766	14	0.424575	14	206.1247	13
辽宁	0.244816	24	0.268411	25	31.32987	25
吉林	0.435578	12	0.452474	12	247.7975	9
黑龙江	1	1	1	1	465	1
上海	0.300572	22	0.329804	22	87.59442	20
江苏	0.457414	9	0.481907	9	269.6608	8
浙江	0.198738	27	0.227395	27	14.4916	29
安徽	0.382361	17	0.406026	16	129.9571	18
福建	0.45259	10	0.467574	10	245.1297	10
江西	0.642231	6	0.659859	5	343.4493	5
山东	0.644909	5	0.645653	6	335.2141	6
河南	0.819986	2	0.831288	2	425.6929	2
湖北	0.439355	11	0.45375	11	187.7955	14
湖南	0.681149	4	0.703309	4	378	4
广东	0.405952	15	0.423025	15	163.5882	16
广西	0.176641	28	0.18721	28	5.344145	30
海南	0.421913	13	0.450485	13	217.7399	12
重庆	0.335647	21	0.36313	21	71.80041	21
四川	0.341104	20	0.365267	20	70.98652	22
贵州	0.15795	30	0.174953	29	20.63419	27
云南	0.275895	23	0.282774	23	40.41333	24
西藏	0.168435	29	0.116774	30	17.69539	28
陕西	0.529845	7	0.534373	7	290.3383	7
甘肃	0.218324	26	0.242574	26	24.82013	26
青海	0.396708	16	0.391051	19	134.9553	17
宁夏	0.027102	31	0.049618	31	0.833333	31
新疆	0.807689	3	0.803732	3	420.3387	3

4　结束语

本文把经济学中最小机会损失准则用到组合评价中来,提出了基于最小机会损失决策准则的新的组合评价方法。该组合评价方法用使组合机会损失价值到每个单一评价方法中最小的机会损失值距离最小的思想建立模型,从而确定各单一评价方法的权重,用权重求得组合评价的最终评价值,并进行排序。本文建立的模型能够使各方案的组合评价值尽可能地接近多个单一评价方法中最小的机会损失值,从而达到每个方案都尽可能地减少了自己失去最高排序的风险损失。该方法概念清晰、涵义明确、算法简单,并用实例证明了其在处理多属性决策经济问题时效果良好,从而具有一定的推广和实用价值。

参考文献:

[1] CHEN Guo-hong, CHEN Yan-tai. The research progress & development trend of comprehensive evaluation methods[C]//Proc of International Conference on Management Science&Engineering. Harbin: Harbin Institute of Technology Press, 2002:462-470.

[2] 郭显光,一种新的综合评价方法 ——组合评价法[J]. 统计研究, 1995,12(5):56-591.

[3] GREGORY A J. The road to integration. Reflections on the development of organizational evaluation theory and practice[J]. Omega-International Journal of Management Science, 1996, 24(3):295-370.

[4] 陈国宏,陈衍泰,李美娟..组合评价系统综合研究[J]. 复旦学报:自然科学版,2003,42(5):668-672.

[5] 毛定祥. 一种最小二乘意义下主客观评价一致的组合评价方法[J]. 中国管理科学,2000,10(5): 95-97.

[6] 徐泽水,达庆利. 多属性决策的组合赋权方法研究[J]. 中国管理科学,2002,10(2) :84-87.

[7] 陈华友. 多属性决策中基于离差最大化的组合赋权方法[J]. 系统工程与电子技术,2004,26(2):194-197.

[8] 陈衍泰,陈国宏,李美娟. 应用合作博弈确定组合评价权重系数的方法研究[J]. 中国管理科学,2005,13(3):89-94.

[9] 《运筹学》教材编写组. 运筹学[M]. 北京:清华大学出版社, 2005.

[10] 明安联. 经济学中的矩阵理论[M]. 成都:四川科学技术出版社, 1991.

[11] 郭亚军. 综合评价理论与方法[M]. 北京:科学出版社,2001.

(上接第 481 页)

[7] WANG Jian-yong, HAN Jia-wei. BIDE: efficient mining of frequent closed sequences[C]//Proc of the 20th International Conference on Data Engineering. 2004:79-90.

[8] CONG Sheng-nan, HAN Jia-wei, PADUA D. Parallel mining of closed sequential patterns[C]//Proc of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. 2005: 562-567.

[9] ZHU Tian, BAI Si-xue. A parallel mining algorithm for closed sequential patterns[C]//Proc of the 21st International Conference on Advanced Information Networking and Applications Workshops. [S. l.]: IEEE Computer Society, 2007: 392-396.

[10] BUEHRER G, PARTHASARATHY S, CHEN Y K. Adaptive parallel graph mining for CMP architectures[C]//Proc of the 6th International Conference on Data Mining. Washington DC: IEEE Computer Society, 2006: 97-106.

[11] LUCCHESE C, ORLANDO S, PEREGO R. Parallel mining of frequent closed patterns: harnessing modern computer architectures [C]//Proc of the 7th IEEE International Conference on Data Mining. 2007: 242-251.

[12] GURALNIK V, KARYPIS G. Parallel tree-projection-based sequence mining algorithms[J]. Parallel Computing, 2004, 30(4): 443-472.

[13] 邹翔,张巍,刘洋,等. 分布式序列模式发现算法的研究[J]. 软件学报,2005,16(7):1262-1269.

[14] TATIKONDA S, PARTHASARATHY S. Mining tree-structured data on multicore systems[C]//Proc of VLDB Endowment. 2009:694-705.

[15] 宋刚, 蒋孟奇. 有限元单元计算子程序的 OpenMP 并行化[J]. 计算机工程, 2008,34(6): 80-84.

[16] 胡斌, 袁道华. TBB 多核编程及其混合编程模型的研究[J]. 计算机技术与发展,2009,19(2): 89-101.

[17] YAN Xi-feng, HAN Jia-wei, AFSHAR R. CloSPan: mining closed sequential patterns in large datasets[C]//Proc of the 3rd SIAM International Conference on Data Mining. 2003:166-177.

[18] 王鹏, 孟丹, 詹剑锋,等. 数据密集型计算编程模型研究进展[J]. 计算机研究与发展,2010,47(11):1993-2002.