DOI:10.3969/j.issn.1673-4785.201411036
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.TP.20150630.1555.003.html

CMP 上基于数据集划分的 K-means 多核优化算法

申彦[1,2],朱玉全[2]
(1.江苏大学 信息管理与信息系统系,江苏 镇江 212013;2.江苏大学 计算机科学与通信工程学院,江苏 镇江 212013)

摘 要:虽然现在多核 CPU 非常普及,但传统 K-means 聚类算法由于没有专门进行并行化设计,不能充分利用现代 CPU 的多核计算能力,算法针对大规模数据集的聚类效率有待进一步提高。因此,对 K-means 算法进行 CMP 并行化改进,提出了一种 Multi-core K-means(MC-K-means)算法。该算法对 K-means 的聚类任务进行了分解,设计了独立且均衡的聚类子任务并分配给各线程并行执行,以此利用现代 CPU 的多核计算能力。实验结果表明,MC-K-means 相比 K-means 获得了较高的多核加速比,提高了针对大规模数据集的聚类能力。

关键词:K 均值算法;聚类算法;单片多核;大规模数据集;数据挖掘;无监督学习;大数据
中图分类号: TP181 **文献标志码**:A **文章编号**:1673-4785(2015)04-0607-08

中文引用格式:申彦,朱玉全. CMP 上基于数据集划分的 K-means 多核优化算法[J]. 智能系统学报, 2015, 10(4): 607-614.
英文引用格式:SHEN Yan, ZHU Yuquan. An optimized algorithm of K-means based on data set partition on CMP systems[J]. CAAI Transactions on Intelligent Systems, 2015, 10(4): 607-614.

An optimized algorithm of K-means based on data set partition on CMP systems

SHEN Yan[1,2], ZHU Yuquan[2]
(1. Department of Information Management and Information System, Jiangsu University, Zhenjiang 212013, China; 2. School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China)

Abstract: The traditional K-means clustering algorithm is not designed to focus on parallelization, which can not make use of the multi-core computing capability of the modern CPU. Therefore, the clustering efficiency of the traditional K-means for massive data set should be further improved. In this paper, a novel algorithm named Multi-core K-means (MC-K-means) after redesigning the original K-means that focuses on parallelization in a chip multi-processor CMP environment is proposed. In order to utilize the multi-core computing capability of the modern CPU, MC-K-means partitions the clustering tasks into some independent and balanced subtasks and distributes these subtasks to the threads to execute parallel. The experimental results showed that the MC-K-means algorithm received the relatively higher speedup rate compared to the K-means algorithm, which improves the handling capacity for massive data set.

Keywords: k-means; clustering algorithm; CMP; massive data set; data mining; unsupervised learning; big data

聚类是一项重要的研究工作,已经成为数据挖掘、统计分析以及压缩算法等领域的研究重点。聚类研究领域有大量经典的算法涌现,如 K-means, PAM, WaveCluster 等。其中 K-means 算法因其简单、易于实现,获得了广泛的应用。现代数据挖掘技术的一个突出特点是需要处理大规模数据集。经典的 K-means 算法在处理大规模数据集时,无法一次性把数据集全部装载入内存,需要多次扫描硬盘上的数据,整个聚类过程相当耗时。因其应用的广泛性,很多研究人员选择对其进行优化,使其适应大规模数据集聚类的应用需求。值得注意的是,在过去的几十年中,CPU 的主频几乎每两年提高一倍,与

收稿日期:2014-11-28. **网络出版日期**:2015-06-30.
基金项目:国家自然科学基金资助项目(71271117);国家科技支撑计划基金资助项目(2010BAI88B00);江苏省自然科学基础研究计划基金资助项目(BK2010331);江苏省博士研究生创新计划基金资助项目(CX10B_016X);江苏省博士后科研资助计划项目(1401056C).
通信作者:申彦. E-mail:104186179@qq.com.

此相对应的内存频率却没有相对应的提高。内存与 CPU 之间处理数据的能力差距越来越大,极大地影响了应用程序的性能。同时,工程师们开始认识到,仅仅提高单核芯片的频率会产生过多热量且无法带来希望的性能改善。于是,CMP(chip multi-processor)成为了先进处理器的发展趋势。CMP 可以在大幅提高处理性能的同时降低 CPU 主频,减少能源消耗。然而仅简单提供 CMP 环境并不能直接带来应用程序性能的提高,需要研发人员针对 CMP 环境对有关算法进行优化,才能使得应用程序更好的利用 CPU 的多核计算能力,提高程序的运行效率[1]。

本文针对提高大规模数据集聚类效率的问题,着重研究单机多核环境下(CMP)K-means 算法的并行化改进,提出了一种 Multi-core K-means (MC-K-means)算法。该算法对原 K-means 算法的聚类任务进行了分解,设计了相互独立且均衡的聚类子任务交由各线程并行执行,能够充分利用现代 CPU 的多核计算能力,提高大规模数据集的聚类效率。

1 研究背景

1.1 确定性聚类的基本概念

金属橡胶隔振器在飞机液压管道上的应用如图 1 所示,从图 1 看出,金属橡胶放置在外围卡箍的凹槽内,传统管道固定一般直接与外围卡箍接触或之间有薄的橡胶垫作为隔振装 。

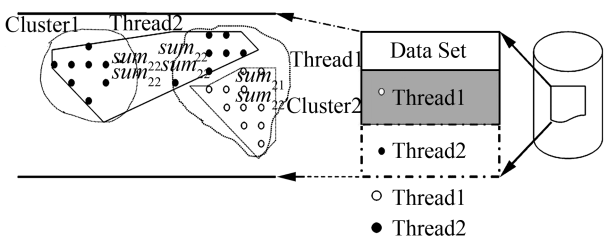


图 1 MC-K-means 算法示意

Fig. 1 Illustration of MC-K-means

定义 1 确定性聚类的输入可以用一组有序对 (X, s) 或 (X, d) 来表示,这里 X 表示一组样本,s 和 d 分别是度量样本间相似度或相异度(距离)的标准。确定性聚类系统的输出是一个个分区,例如 $C=\{C_1, C_2, \cdots, C_k\}$,其中 $C_i(i=1,2\cdots,K)$ 是 X 的子集,且满足:$C_1 \cup C_2 \cup, \cdots, \cup C_k = X$;$C_i \cap C_j = \varnothing$, $i \neq j$。

C 中的成员 $C_1, C_2, \cdots, C_k$ 叫做类或簇(Cluster),每一个类或簇都是通过一些特征描述的,通常有如下几种表示方式:

1) 通过它们的中心或类的边界点表示空间的一类点。

2) 使用聚类树中的结点,图形化地表示一个类。

3) 使用样本属性的逻辑表达式表示类。

1.2 相关的研究工作

为了解决 K-means 算法对大规模数据集聚类效率较低的问题,有研究者提出了只需要扫描一遍原始数据集即产生聚类结果的算法。这些算法只需读入大规模数据集中的一部分进入主存或者分批读入数据集进行聚类,扫描数据集一遍即完成聚类。相应的算法有 random-kmeans, Dynamic incremental K-means[2], Single pass kernel K-means[3], scalable-kmeans[4],等。其中,由 Microsoft Research 的 Redmond 等提出的 scalable-kmeans 算法性能优越,受到了广泛的重视,并被集成到 SQL SERVER 2008 中。类似研究的主要目的是优化 K-means 算法,减少数据集的读取次数。有研究者从优化 K-means 聚类初始条件设置的角度,利用自适应技术、启发式算法以及半监督技术等实现 K-means 初始聚类中心或者聚类个数的优化选择,加速 K-means 聚类的收敛过程,提高聚类的效率以及结果的质量[5-7]。有研究人员从减少大规模数据集数据维度的角度,降低聚类迭代过程的计算量,提高 K-means 聚类算法的效率[8-9]。以上相关的研究工作切实提高了 K-means 聚类的效率,然而这些新算法并没有利用分布式环境提高聚类的效率。最近有研究人员进行了 SMP、DMP 环境下的集群多处理器 K-means 聚类的研究工作,提高了大规模数据集的聚类效率[10-11]。直接针对共享内存多处理器系统以及分布式内存多处理器环境进行 K-means 的并行化,需要考虑复杂的数据划分、节点容错等并行化的基本问题且需要消耗大量的节点间同步以及数据网络传输的时间。随着类似 Google MapReduce 以及 Apache 的 Hadoop 的出现和广泛使用,在这些编程模型的基础之上进行分布式开发变得相对容易,分布式的基本问题可以依靠基础编程模型来解决。很多研究人员利用 MapReduce 的算法模型,针对 K-means 聚类过程的并行化进行了大量深入的研究工作,取得了很多重要的研究成果,使得 K-means 算法可用于大规模数据集聚类的应用场合。然而这些算法更多考虑的是多处理器分布式场景下的 K-means 并行化,较少考虑到单机 CPU 的多核利用。除此之外,并不是所有的聚类算法都适合以 MapReduce 的形式进行并行化的,且为了适应 MapReduce 的编程架构,有时反而会增加额外的计算量与通信量[12-14]。

现代 CPU 技术的发展,使得单机的运行环境也发生了极大的变化。多核处理器的出现提高了 CPU 的计算性能,降低了 CPU 的功耗。尽管如此,传统的算法并不能直接从多核 CPU 中获益,需要针对多核 CPU 的特点进行并行化改进与优化,才能充分利用多核 CPU 的计算能力。因此,研究单机 CMP 环境下 K-means 算法的并行化方法对提高单机 K-means 算法的聚类效率具有重要的现实意义,并且

与 DMP、SMP 环境下的 K-means 聚类过程可以有效的结合,作为 DMP、SMP 环境下 K-means 聚类算法的有效补充。也有研究人员开始着手研究 CMP 环境下 K-menas 算法的并行化,但是相关研究尚处于起步阶段,算法实现仍存在进一步改进的空间[15-16]。

1.3　多核处理器的出现

2005 年,当主频接近 4 GHz 时,CPU 的主要制造厂商英特尔和 AMD 公司发现单纯的主频提升已经无法明显提升系统整体性能。由于 CPU 片内流水线过长,使得单位频率效能低下,加上由于缓存的增加和对漏电流控制的不利,造成 CPU 功耗大幅度增加。随着功耗的增大,散热问题也越来越成为一个无法逾越的障碍。于是,出现了多核心 CPU 的解决方案。

其实较早以前已经有研究人员提出了利用单芯片多核心处理器(CMP)技术来代替复杂度越来越高的单核心 CPU。IBM、IP、SUN 等企业也在服务器领域投入了一定的多核 CPU 进行商用。然而由于当时的服务器多核 CPU 价格过于昂贵、应用面窄、并没有真正发展起来。

2006 年,多核 CPU 进入了迅猛的发展时期,Intel 的 Core, Xeon 以及 AMD 的 Athlon, Barcelona 等受到了广泛的欢迎。这些 CPU 在性能得到极大提升的同时,功耗反而得到了降低。

值得注意的是 OS 并不能自动的让某个应用程序直接利用 CPU 的多核,而是需要进行有关算法的 CMP 并行化改进。对于数据挖掘中的聚类、关联规则挖掘等计算密集型、I/O 密集型应用而言,对原有算法进行并行化改进,提高算法的执行效率,尽快给出挖掘结果成为了当务之急。研究具有较强的现实意义[1]。

2　K-means 算法详细描述

K-means 算法,也被称为 K-平均或 K-均值算法,是目前得到广泛应用的一种聚类算法[17]。其相似度的计算根据一个簇中对象的平均值来进行。K-means 算法以 k 为参数,把 n 个数据点分为 k 个簇,使得簇内具有较高的相似度,而簇间的相似度较低。

算法首先随机地选择 k 个对象,每个对象初始地代表了一个簇的平均值或中心。对剩余的每个对象根据其与各个簇中心的距离,将它赋予最近的簇,然后重新计算每个簇的平均值。这个过程不断重复,直到准则函数收敛。准则函数定义为: $E = \sum_{i=1}^{k}\sum_{x\in C_i}|x-\overline{x_i}|^2$。这里的准则函数 E 是数据集中所有数据点的平方误差总和,x 是数据集空间中的点,$\overline{x_i}$ 是簇 C_i 的平均值。准则函数 E 使得生成的结果簇尽可能的紧凑和独立。

算法 1　K-means (Dataset D, ClusterNumber K)

输入:事务数据库 D,聚类簇的数量 K

输出:K 个聚类,使得平方误差准则 E 最小

1) assign initial value for means;

//任意选择 k 个对象作为初始的簇中心

2) REPEAT;

3) FOR $j=1$ to n DO assign each x_j to the closest clusters mean;

//根据簇中对象的平均值将每个对象分配给最近的簇

4) FOR $i=1$ to k DO $\overline{x_i}=\frac{1}{|C_i|}\sum_{x\in C_i}x$;

//更新簇的平均值,即计算每个簇中对象的平均值

5) Compute $E=\sum_{i=1}^{k}\sum_{x\in C_i}|x-\overline{x_i}|^2$;

//计算准则函数 E

6) Until $E_{new}-E_{last}<\varepsilon$, ε 为预先设定的一个较小的值;

//表示 E 不再产生明显的变化

K-means 是解决聚类问题的一种经典算法,该算法实现起来较为简单且有非常好的可扩展性。因此,很多科研人员在研究针对大规模数据集的高效聚类算法时,往往会以 K-means 算法作为首选进行改进和优化。

分析 K-means 算法的时间复杂度,其运行时间主要消耗在:1)数据集读取所产生的 I/O;2)判断每一个数据点(数据记录)的所属类别;3)计算每一个类别(簇)的中心;4)计算准则函数 E。而这 4 个阶段均可以很好地并行化,以此利用现代 CPU 的多核特性,最大化的发挥 CPU 的性能,提高聚类效率。

为此,本文提出了一种 Multi-core K-means 算法(MC-K-means),该算法对上述 4 个过程分别进行并行化,充分利用 CPU 的多核特性,进一步提高 K-means 算法的聚类效率。新算法可作为 SMP、DMP 分布式环境下聚类算法以及增量 OneScan 聚类算法的有效补充,提高单节点的聚类效率,从而提高整体的聚类效率。

3　CMP 上基于数据集划分的大规模数据集 K-means 多核优化算法

3.1　MC-K-means 算法详细描述

在 CMP 环境下对 K-means 聚类算法进行改进以适应大规模数据集,关键是要改进原有算法的串行执行部分为并行执行。分析 K-means 算法可以发现,在较为消耗资源的数据集读取阶段、数据点所属类别判断阶段、每个新簇的簇中心计算阶段以及准则函数的计算阶段,这些阶段均可进行并行化改进,

且由于其所需计算的性质,都可以较好的做到多核之间的负载均衡。

改进后的 MC-K-means 算法详细描述如下。其中关键步骤如图 1 所示,为了方便描述,图中以 2 线程为例进行说明,可推广到 n 线程的情形。

算法 2 MC-K-means (Dataset D, ClusterNumber K)

输入:事务数据库 D,聚类的簇的数量 K

输出:K 个聚类,使得平方误差准则 E 最小

1) random assign initial value for means;

//任意选择 k 个对象作为初始的簇中心

2) thread_count = Runtime.getRunTime().availabkleProcessors/(1−blockCoefficient);0<=blockCoefficient<1;

//计算线程数

3) executeService service = Excute.newFixedThreadPool(thread_count);

// 创建线程池

4) divide the data set into n parts equally and create n tasks to read every data set, where n = thread_count;

//装载数据

5) thread_count = Runtime.getRunTime().availabkleProcessors;

6) threadPool = Excute.newFixedThreadPool(thread_count);

//创建线程池,计算准则函数 E

7) divide the data set into n parts equally and create n tasks to compute the category, where n = thread_count;

8) repeat

9) for every data_set$_j$

10) let one task$_j$ to assign each data point of the data_set$_j$ to the closest clusters center and record the category;

11) until every data_set$_j$ is finished;

12) for every equally data_set$_j$//计算每个簇的簇中心;

13) let one task$_j$ to compute the sum$_{ij}$ and num$_{ij}$ of data_set$_j$ partly, $j=1,2,\cdots,n$, $i=1,2,\cdots k$;

14) until every equally data_set$_j$ is finished;

15) join the results of every task$_j$ to get total_sum$_i$ = $(\sum_{j=1}^{j=n} \text{sum}_{ij})$, total_num$_i$ = $(\sum_{j=1}^{j=n} \text{num}_{ij})$, $i=1,2,\cdots,k$;

16) cluster_center$_i$ = total_sum$_i$/total_num$_i$, $i=1,2,\cdots,k$;

//每个线程均可访问中心值,以便再次划分数据

17) for every equally data_set$_j$//计算每个簇部分准则函数

18) let one task$_j$ to compute the E_{ij} of data_set$_j$ partly, $j=1,2,\cdots,n$, $i=1,2,\cdots,k$;

19) until every equally data_set$_j$ is finished;

20) join the results of every task$_j$ to get $total\_E_i$ = $(\sum_{j=1}^{j=n} E_{ij})$, $i=1,2,\cdots,k$;

21) $E = \sum_{i=1}^{i=k} \text{total\_E}_i$;

22) until $E_{\text{new}} - E_{\text{last}} < \varepsilon$, ε is a preset very small threshold

在读入外存数据时,考虑到数据源可能存在于网络数据库中,在读取时会有一定的延时,多开线程可有效利用 CPU 的多核,因此考虑设置 Runtime.getRunTime().availabkleProcessors/(1-blockCoefficient)大小的线程池,其中 blockCoefficient = 数据记录 I/O 阻塞时间/数据记录处理时间,在运行时可根据数据源的延时动态调整。

在装载数据之后,判断每一数据点的所属类别时采用的是欧几里得距离的平方 $d(x,y)^2 = \left[\sum_{i=1}^{n} |x_i - y_i|^2\right]$。该计算对于每个数据点的计算量均是相同的,等分数据即可做到负载平衡。除此之外,该过程是计算密集型的,多开线程对提高效率无益,反而会因为 CPU 频繁的线程切换而降低运行效率。因此开设线程个数与 CPU 核心数 availabkleProcessors 相同的线程池;又因为距离计算任务的计算量对每个数据点是一样的,所以 MC-K-means 算法等分数据,创建 availabkleProcessors 个任务进行数据点类别判断的计算,并交由线程池调度执行。

计算每一个聚类簇的簇中心,仍然是一个计算密集型的任务,因此在此阶段开设线程数与 CPU 核心数相同的线程池。MC-K-means 算法针对之前等分的数据集,每个线程 j 计算被分配的数据集归属于每个分类 i 的 sum$_{ij}$ 以及 num$_{ij}$,并汇总 j 个线程的结果得到 total_sum$_{ij}$ 以及 total_num$_{ij}$,最终得到 cluster_center$_i$。采用针对等分数据集的方法使得簇中心计算的各任务相对均衡。在准则函数 E 的计算过程中也采用了同样的负载均衡的方法。

CMP 系统是共享内存的,上述 MC-K-means 算法仅在访问共享变量及每部分数据处理完毕时需要进行同步,避免了数据集通过网络在节点之间传输造成的时间消耗,算法具有较高的执行效率。

4 实验结果以及分析

为了验证算法的有效性,依据前述 MC-K-means 算法的主要思想,使用 Java 语言实现了 MC-K-means 以及 K-means 算法[18-19]。实验平台为 HP PRO 3380 MT,Window XP_SP3,4 GB 内存,jdk7u51 以及 HP ProLiant DL388p Gen8, RedHat 9.0,32 GB 内存,jdk7u51。因为是做 CPU 多核加速的有关实

验,所以需要针对不同实验平台中不同类型的 CPU 进行测试。HP PRO 3380 MT 平台采用的 CPU 是 Intel i3-3240@ 3.40 GHz,核心类型为 Ivy Bridge,64 位 CPU,双核,四线程,支持超线程技术,3 MB 三级缓存,双通道。HP ProLiant DL388p Gen8 平台采用的 CPU 是 Intel Xeon E5-2609@ 2.40 GHz,核心类型为 Sandy Bridge,64 位 CPU,四核,四线程,10 MB 三级缓存,四通道。

对比算法为文献[15]描述的基于 MapReduce 的并行 K-means 算法(PKMeans_MR)以及文献[16]描述的 PKMeans_MT 算法。在实验中根据上述文献描述的算法思想分别实现了各算法进行对比实验。因为相同初始化条件下各算法最终聚类的结果是一样的,所以实验主要对比分析相关算法的执行时间以及加速率情况。

4.1　人工生成数据集测试

人工生成数据集由数据生成程序根据 K 个高斯分布随机产生,每个高斯分布设置一个随机权重来确定是否产生数据。每一个高斯分布的中心是随机产生的,区间为[-5,5]。数据点每个维度值的产生区间为[0.7,1.5]。产生的人工数据集包含 100 维,180 000 个数据点,数据集使用二进制 bin 文件保存。本次实验因为数据集以及聚类测试算法是在同一台计算机上的,所以读取数据集时的阻塞系数 blockCoefficient 设置为 0。聚类簇数量设置为 $k=5$。

整个实验随机产生 5 个不同的人工生成数据集,针对每个数据集,分别执行 MC-K-means、PKMeans_MR、PKMeans_MT 以及 K-means 算法各 10 次,各自共运行 50 次,最后以聚类时间的平均值作为算法聚类效率的评价。实验结果如图 2、3 所示。

该数据集较为规整,算法运行收敛较快。从实验结果可以看出:针对服务器领域的 Xeon E5-2609,虽然主频较低,但依靠较大的 L3 缓存以及四通道的内存控制器,使得各算法取得了较高的执行效率。K-means 算法在该平台环境下执行消耗了 25.28 s,PKMeans_MR 消耗了 10.01 s,PKMeans_MT 消耗了 8.15 s,MC-K-means 则需要 7.02 s。而在 i3-3240 所在平台环境下,K-means 算法消耗了 28.24 s,PKMeans_MR 消耗了 11.47 s,PKMeans_MT 消耗了 9.35 s,MC-K-means 在该平台则需要 9.02 s。从并行化改造后各算法的执行结果来看,在 Xeon E5-2609 所在平台,算法获得了更高的加速比。这主要是由于 Xeon E5-2609 是四核四线程的,拥有真实的四核心,可以更好地并行完成各并行化算法划分的多任务聚类工作。而 i3-3240 是双核心的,依靠超线程技术实现的四线程并行,但是 CPU 的双物理核心需要频繁的进行线程上下文的切换,消耗了一部分的运行时间,获得了较低的加速比。

其中,MC-Kmeans 算法在读取数据集、判断每个数据点的所属类别、计算簇中心以及计算准则函数 E 这 4 个阶段均进行了并行化改进。且在这 4 个阶段中,每个数据点的任务量是相当的,因此 MC-K-means 算法所采取的等分数据集的方法可以取得较好的负载均衡性,算法取得了对比算法中最高的加速率。

PKMeans_MT 算法取得了较低的加速率。分析算法可知,该算法仅在读取数据集以及迭代计算每个数据点的归属时利用 parfor 函数进行了并行化。对新分类中心点的计算及准则函数的计算均没有并行化,且算法需要依托 MATLAB 平台,故取得了较低的加速率。

PKMeans_MR 算法在对比算法中取得了最低的加速率。分析算法可知,PKMeans_MR 算法改进原 K-means 算法,使其以 MapReduce 方式运行,节点之间在迭代时需要多次通信、多次同步,且算法需适应 MapReduce 固有模式,降低了算法的运行效率,因此取得了最低的加速率。但对比原 K-means 算法仍提高了一倍多的运行效率。

可以看出,对 K-means 进行并行化改进,适应了多核 CPU 的发展趋势,可以极大程度提高算法的运行效率,满足处理大规模数据集的需要。

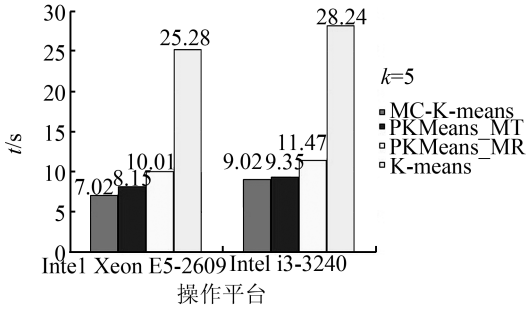


图 2　运行时间对比

Fig. 2　Comparison of run time

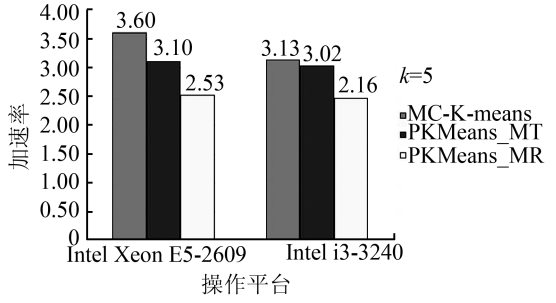


图 3　加速率对比

Fig. 3　Comparison of speedup rate

逐步增大生成数据集的规模,生成的人工数据集为 100 维,分别包含 180 000 个数据点,360 000 个数据点,540 000 个数据点,720 000 个数据点,900 000个数据点。在 Intel Xeon E5-2609 平台测试每个算法的加速率。实验每次随机产生 2 个相同大

小的人工数据集,针对每个数据集,分别执行 MC-K-means、PKMeans_MR、PKMeans_MT、K-means 算法各 10 次,各算法每阶段各自共运行 20 次,最后分别记录下针对每个大小的数据集各算法聚类的平均时间。k 的取值为 5。实验结果如图 4 所示,各算法加速率如图 5 及表 1 所示。

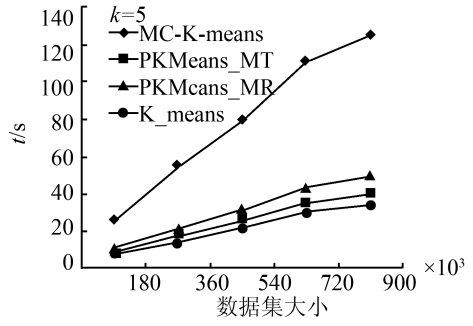


图 4 算法线性测试

Fig. 4 Scalability test of different algorithms

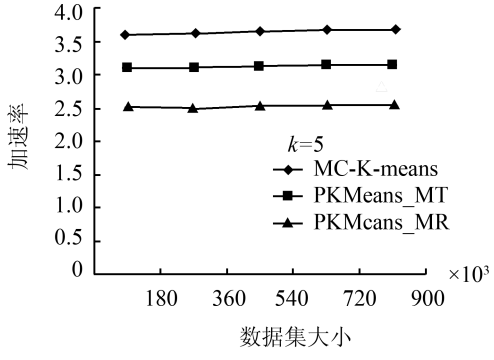


图 5 不同数据集大小下的加速率

Fig. 5 Speedup rate of different data sizes

表 1 不同数据集大小下的加速率详情

Table 1 Speedup rate details of different data sizes

Data Set Size/×10³	MC-K-means	PKMeans_MT	PKMeans_MR
180	3.6	3.1	2.52
360	3.62	3.11	2.51
540	3.64	3.14	2.53
720	3.68	3.15	2.54
900	3.68	3.15	2.53

从实验结果可以看出,随着数据集规模的不断增长,各算法的运行时间均较为线性的增加。其中 PKMeans_MR 算法因为节点数并没有增加,所以加速率基本保持不变。MC-K-means 以及 PKMeans_MT 的加速率均随着数据集规模的增加呈现提高的趋势,但都在接近各自极限后不再提高,保持一个相对稳定的加速率。这是因为随着数据集规模的增大,线程之间切换的资源消耗所占比重逐步降低,多核优势逐渐显现。

验证不同数据集大小情况下,不同的聚类数目 k 的取值对多核加速效率的影响。设置 k 为 10,再次变换数据集大小从 180K 至 900K,重复上述实验过程,记录实验结果如图 6 所示,各算法加速率如图 7 及表 2 所示。

表 2 不同数据集大小下的加速率详情

Table 2 Speedup rate details of different data size

Data Set Size/×10³	MC-K-means	PKMeans_MT	PKMeans_MR
180	3.71	3.16	2.52
360	3.73	3.23	2.53
540	3.75	3.27	2.54
720	3.76	3.29	2.53
900	3.75	3.29	2.54

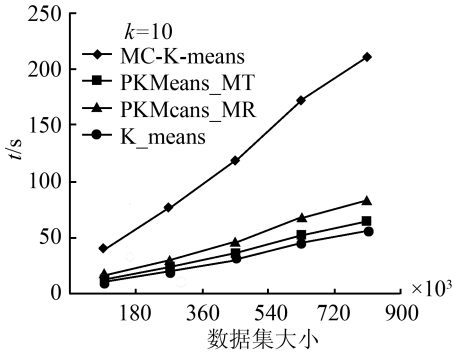


图 6 算法线性测试

Fig. 6 Scalability test of different algorithms

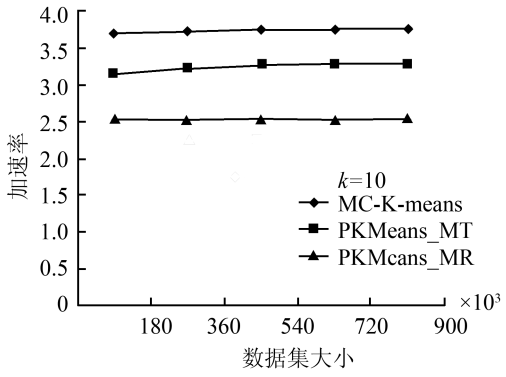


图 7 不同数据集大小下的加速率

Fig. 7 Speedup rate of different data size

从实验结果可以看出,除 PKMeans_MR 算法因节点数不变,加速率保持相对稳定外,k 取值的提高,有利于各并行化算法子任务的并行执行,减少同步,故提高了相应算法的加速率。

为了验证 MC-K-means 算法在执行过程中各并行化挖掘任务的负载是均衡的,在 i3-3240 所在平台,修改聚类的过程为 K-means 读取数据集→MC-K-means 读取数据集→K-means 聚类→MC-K-means 聚类。利用 JDK7 平台的 jvisualvm[20] 工具对生成的 180K 数据集的聚类全过程进行了监控,并记录如图 8 所示。

从图 8 中的实验记录全过程来看,MC-K-means 在读取数据集以及聚类阶段,线程池中各线程的负载是较为均衡的,没有出现长时间的空闲状态,算法充分利用了 CPU 各核心的计算能力。

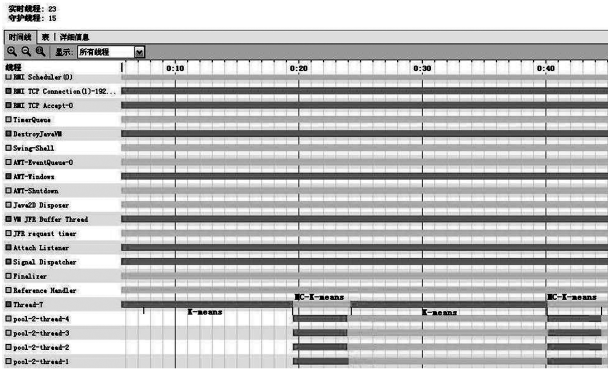


图 8　jvisual 监控截图

Fig. 8　Screenshot of jvisual

4.2　真实数据集测试

使用 UCI 上的真实数据集 Forest CoverType dataset 对相关算法进行测试。该数据集共有 581 012 条记录,每条记录有 54 个维度,聚类时簇的个数设置为 $k=7$。

该数据集较为庞大且聚类需多次迭代,挖掘耗时较多。在 Xeon E5-2609 所在平台,K-means 算法完成聚类需要 98.65 s,MC-K-means 需要 26.59 s,PKMeans_MT 需要 30.45 s,PKMeans_MR 需耗时 39.15 s;而在 i3-3240 所在平台,K-means 算法完成聚类需要 103.42 s,MC-K-means 需要 28.02 s,PKMeans_MT 需要 33.15 s,PKMeans_MR 需耗时 41.22 s。从图 9、10 中可以看出,与人工生成数据集的测试结果类似,并行优化后的各算法在 Xeon E5-2609 平台获得了更高的加速比,且 MC-K-menas 算法依靠其更多执行步骤的并行化、更为直接的底层算法实现以及均衡的任务负载取得了最高的加速率。

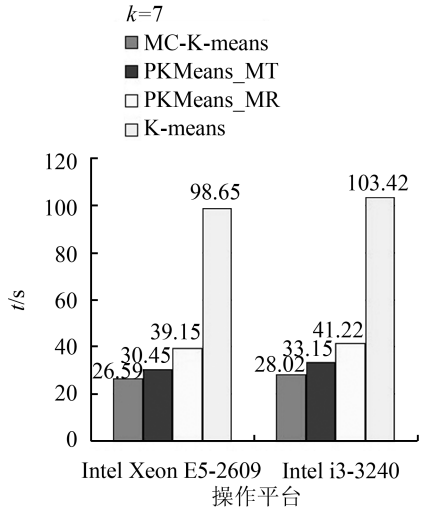


图 9　运行时间对比

Fig. 9　Comparison of run time

实验结果再次验证了 MC-K-means 算法可以在 CPU 的各核心之间均衡的负载各挖掘任务,充分利用现代 CPU 的多核计算能力,提高对大规模数据集的聚类效率。

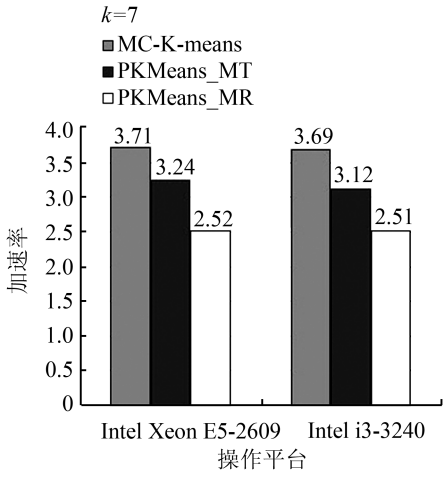


图 10　加速率对比

Fig. 10　Comparision of speedup rate

5　结束语

本文考虑到现阶段多核 CPU 的普及,针对经典的 K-means 算法进行了多核并行优化,提出了一种 MC-K-means 算法。该算法把 K-means 聚类任务按数据集等分为多个相互独立的挖掘子任务,并动态分配给多个线程并行执行,充分利用现代 CPU 的多核计算能力。实验结果证明了该算法可以充分利用现在的多核 CPU,取得了较高的加速比,提高了聚类算法处理大规模数据集的能力。

虽然 MC-K-means 算法有着上述优势,但是当前版本的算法仍然存在一些问题,有待进一步改进。例如算法可以进一步扩展到集群聚类的领域。这些研究内容将在以后的研究过程中进一步补充完善。

参考文献:

[1] SUBRAMANIAM V. Programming concurrency on the JVM mastering synchronization, STM, and actors[M]. Beijing: China Machine Press, 2013: 1-27.

[2] AARON B, TAMIR D E, RISHE N D, et al. Dynamic incremental K-means clustering[C]// Proc of the 2014 International Conference on Computational Science and Computational Intelligence, CSCI 2014. Los Alamitos, CA: IEEE Computer Society, 2014: 308-313.

[3] SARMA T H, VISWANATH P, REDDY B E. Single pass kernel k-means clustering method[J]. Sadhana - Academy Proceedings in Engineering Sciences, 2013, 38(3): 407-419.

[4] BRADLEY P, FAYYAD U, REINA C. Scaling clustering algorithms to large databases[R]. Redmond: Microsoft Research Report, 1998: 9-15.

[5] 陈光平,王文鹏,黄俊. 一种改进初始聚类中心选择的 K-

means 算法[J]. 小型微型计算机系统,2012,33(6):1320-1323.
CHEN Guangping, WANG Wenpeng, HUANG Jun. Improved initial clustering center selection method for k-means algorith[J]. Journal of Chinese Computer Systems, 2012, 33(6): 1320-1323.

[6] MAHMUD M S, RAHMAN M M, AKHTAR M N. Improvement of k-means clustering algorithm with better initial centroids based on weighted average[C]//Proc of the 7th International Conference on Electrical and Computer Engineering, ICECE 2012. Los Alamitos, CA: IEEE Computer Society, 2012: 647-650.

[7] PATIL R, JONDHALE K C. Edge based technique to estimate number of clusters in k-means color image segmentation [C]//Proc of the 3rd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2010. Piscataway, NJ: IEEE Computer Society, 2010: 117-121.

[8] JING Liping, NG M K, HUANG zhexue. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data[J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(8): 1026-1041.

[9] BISHNU P S, BHATTACHERJEE V. A dimension reduction technique for k-means clustering algorithm[C]//Proc of the 1st International Conference on Recent Advances in Information Technology, RAIT-2012. Piscataway, NJ: IEEE Computer Society,2012: 531-535.

[10] DOBBELIN R, SCHUTT T , REINEFELD A. An analysis of SMP memory allocators: mapreduce on large shared-memory systems[C] //Proc of the 41st International Conference on Parallel Processing Workshops (ICPPW), 2012. Piscataway, NJ: IEEE, 2012: 48-54.

[11] DI F G, BLASA F, CAFIERO S, et al. Fault tolerant decentralised k-means clustering for asynchronous large-scale networks[J]. Journal of Parallel and Distributed Computing, 2013, 73(3): 317-329.

[12] 赵卫中,马慧芳,傅燕翔,等.基于云计算平台 Hadoop 的并行 k-means 聚类算法设计研究[J].计算机科学,2011,38(10):166-169.
ZHAO Weizhong, MA Huifang, FU Yanxiang, et al. Reasearch on parallel k-means algorithm design based on hadoop platform [J] . Computer Science, 2011,38(10): 166-169.

[13] 王晓华. MapReduce 2.0 源码分析与编程实战[M].北京:人民邮电出版社, 2014:1-55.

[14] MARTHA, V, ZHAO Weizhong, XV Xiaowei. H-MapReduce: A framework for workload balancing in MapReduce [C]// Proc of the International Conference on Advanced Information Networking and Applications, AINA. Piscataway, NJ: IEEE, 2013: 637-644.

[15] ZHAO Weizhong, MA Huifang, HE Qing. Parallel k-means clustering based on mapreduce[C]//Proc of the 1st International Conference on Cloud Computing, CloudCom 2009. Germany: Springer Verlag, 2009: 674-679.

[16] FAHIM A M. Parallel implementation of k-means on multicore processors[J]. Computer Science and Telecommunications, 2014, 1(41): 53-61.

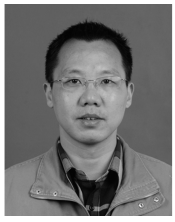
[17] ZALIK K R. An efficient k-means clustering algorithm[J]. Pattern Recognition Letters, 2008, 29(9): 1385-1391.

[18] HERBERT S, DALE S. A comprehansive introduction [M]. Beijing: China Machine Press, 2013

[19] JAVIER F G. Java 7 concurrency cookbook[M]. Beijing: Posts & Telecom Press, 2014.

[20] Monitoring and managing java se 6 platform applications [EB/OL]. [2005-12-18]. http://java.sun.com/developer/technicalArticles/J2SE/monitoring.

作者简介:



申彦,男,1982 年生,讲师,博士,主要研究方向为数据挖掘、智能信息系统。获 2014 年度中国商业联合会科学技术奖三等奖。发表学术论文 11 篇,其中被 EI 检索 5 篇。

朱玉全,男,1965 年生,教授,博士生导师,主要研究方向为数据挖掘、智能信息系统、信息系统集成。获 2014 年度中国商业联合会科学技术奖三等奖,全国多媒体课件大赛一等奖和江苏省优秀软件产品奖(金慧奖)各 1 项,省部级科技进步奖 4 次,申请发明专利 10 项,其中授权发明专利 3 项,获批计算机软件著作权 7 部。发表学术论文 70 余篇,10 多篇被 EI 检索,出版编著 2 部。

[责任编辑:陈峰]