面向大规模数据的快速并行聚类划分算法研究

牛新征 佘 堃
(电子科技大学计算机科学与工程学院 成都 610054)

摘 要 随着聚类分析中处理数据量的急剧增加,面对大规模数据,传统K-Means聚类算法面临着巨大挑战。为了提高传统K-Means聚类算法的效率,针对已有基于MPI的并行K-Means聚类算法和基于Hadoop的分布式K-Means云聚类算法,从聚心初始化和通信模式等入手,提出了改进思路和具体实现。实验结果表明,所提算法能大大减少通信量和计算量,具有较高的执行效率。研究结果可以为以后设计更好的大规模数据快速并行聚类划分算法提供研究依据。

关键词 云计算,K-Means,大规模数据,MPI,Hadoop
中图法分类号 TP393 **文献标识码** A

Study of Fast Parallel Clustering Partition Algorithm for Large Data Sets

NIU Xin-zheng SHE Kun
(School of Computing Science and Engineering, UESTC, Chengdu 610054, China)

Abstract With the rapid increase of data amounts in clustering algorithms' processing, traditional K-Means clustering algorithm is facing huge challenge for large data sets. In order to improve efficiency of traditional K-Means clustering algorithm, this paper proposed some improvement ideas and implementation using the cluster center initialization and communication mode, according to parallel clustering algorithm based on MPI and distributed clustering algorithm based on Hadoop in cloud. The results show that research of the algorithm can reduce the communication and computation largely, and can have higher implementation efficiency. The research fruits will help us to design better and fast parallel clustering partition algorithm for large data sets in future.

Keywords Cloud computing, K-means, Large data sets, Message passing interface, Hadoop

1 引言

聚类分析是数据挖掘中一个重要且较活跃的研究领域,其研究的是数据对象的分类问题。为了能够方便地阐述和理解数据对象,提取隐藏在数据中的内在消息或者知识,需要使用聚类分析技术。它的主要思想是将数据分成为若干个类或簇,使得在同一个簇中对象最相似,不同簇中的对象差别很大。其算法大体上可以分为划分方法、层次方法、密度方法、模型方法等[1]。通常的传统聚类算法存在着单位时间内处理量小、面对大量的数据时处理时间较长、难以达到预期效果等缺陷。针对这样的问题,将聚类分析与并行计算、云计算等相结合,设计出高效的并行聚类算法,已经成为一个比较流行的研究思路[2,3]。其中,聚类划分算法是最基本的聚类算法。本文用经典的K-Means算法作为面向大规模数据的聚类划分算法的研究基础,分析如何能以快速、高效、低成本的方式从大规模数据中挖掘有价值的、可理解的数据信息。并行计算是指同时使用多种计算资源解决计算问题的过程,具有加快程序执行速度、节省投入等优势。由于可以使用大量的廉价计算机通过集群来代替价格高昂的服务器,并行计算环境下的数据挖掘服务大大降低了数据处理成本。而云计算能够提供在虚拟计算环境下运行大规模应用的可伸缩性和可靠性、稳定性,基于云计算的大型应用具有分布性、异构性、海量数据性等特征,较适合数据密集型应用和处理[3,4]。

聚类分析普遍存在大规模数据、分布数据难以处理,参数难以确定,效率低下,聚类质量较差等问题。目前,已有许多学者研究了聚类算法的并行化和分布式执行策略[5-9]。其中,文献[5]基于MPI开展了并行K-Means聚类算法的研究并在简历数据的处理过程中应用实现,文献[6]设计了基于Hadoop决策树SPRINT算法的云数据挖掘实现。这两篇文献都是本文要比较和分析的研究成果。而文献[7]提出了基于K-Means的分布式算法,每次迭代过程中节点间都要传送大量的数据对象,导致巨大的通信代价,且其远远大于计算代价,总体效率较低。文献[8]采用了广播聚类中心的方式进行通信,每次迭代需要节点间通信,通信量相当大。文献[9]试图减少通信次数,采取了部分数据并行的思路,局部的计算复杂度相对减少,但是数据仍然需要多次读取,通信的代价也相当大。

本文在研究了已有策略的基础上,考虑了算法在处理大

到稿日期:2011-02-26 返修日期:2011-07-02 本文受中央高校基本科研业务费电子科技大学项目(ZYGX2010J075)资助。
牛新征(1978-),男,博士,讲师,主要研究方向为云计算、数据挖掘,E-mail:xinzhengniu@uestc.edu.cn;佘 堃(1968-),男,博士,教授,博士生导师,主要研究方向为网络计算、人工智能。

数据集时要求的可伸缩性和高效率性，分析了已有算法和实现中存在的问题，提出了在 K-Means 聚类算法上的两种并行化和分布式的设计和实现模型：一种是基于 MPI 的并行化 K-Means 算法，另一种是基于 Hadoop 的分布式 K-Means 算法。在前一种模型中，融入了有效的初始化聚心方法和 k 值选取方法，解决了参数难以确定的问题，也进一步提高了聚类结果的质量，使得 k 的选取对聚类过程的影响降到最低程度。同时，对通信的方法也进行了重新设计，从而提高了算法处理速度。而后一种模型中，对 Hadoop 框架上的编程模型编写进行了新的尝试，最大程度地压缩了通信量，调控了节点分配比例，进一步保证了算法的可伸缩性。最后通过实验将这两种优化之后的模型同传统的算法和已有的研究成果进行比较，结果表明，本算法的优化是有效的，具有参考价值和应用意义。

2 相关概念与描述

2.1 传统 K-Means 算法

传统 K-Means 算法的大体思想是：将集合 n 中没有先验知识的数据对象根据用户的需求划分成 k 个簇。处理流程如下：首先，随机地选择 k 个数据对象，每个对象初始地代表了一个簇的平均值或中心，也即选择了 k 个初始质心。对剩余的每个对象，根据其与各个簇中心的距离，将它赋给最近的簇，与同一质心的数据对象形成一个簇。然后重新计算每个簇的平均值。迭代执行 r 次，直到准则函数收敛，质心不再改变，即完成聚类。当找出了使准则函数值最小的 k 个划分，且结果簇是密集的，簇与簇之间区别明显时，该聚类算法效果较好。

传统的 K-Means 算法描述如下。

(1)从数据中随机选取 k 个不同的数据对象作为 k 个簇的初始化聚心：$C_1, C_2, C_3, \cdots, C_k$。

(2)For 数据集中的每个对象 D_j

寻找离 i 最近的聚心，将 D_j 分配给该聚心类。

(3)For 每个簇

重新计算新聚心，新聚心为分配给该簇的所有数据的平均值：

$$C_l=\frac{\sum_{1}^{n_l} D_j}{n_l}$$

式中，j 为属于 l 类的数据对象，n_l 为属于 l 类的对象个数 $l\in[1,k]$。

(4)计算平方误差：

$$\sum_{i=1}^{k}\sum_{j=1}^{n_l}(D_j-C_l)^2$$

式中，k 为聚心数目。

Until 平方误差不再发生变化，否则转至(2)。

2.2 MPI 与 HADOOP 介绍

并行编程环境是实现并行计算的一个非常重要的条件。MPI 的全称是 Message Passing Interface，本质上它是基于消息传递编写的并行程序函数库，是目前并行计算中广泛使用的一种程序设计模式[5]。从语法上讲，它遵守所有对库函数/过程的调用规则，可以在所有的并行系统上运行。一个 MPI 系统通常由一组库、头文件和相应的运行、调试环境构成。MPI 并行程序从程序结构上分为主从模式(Master/Slave)、单程序多数据模式和多程序多数据模式。在主从模式中，只有一个主进程和多个从进程。主进程可以分配任务和数据给从进程，从进程完成任务后再将结果返回主进程。这些操作可通过调用 MPI 库函数进行消息传递。MPI 为基于消息传递的并行程序设计提供一个高效、可扩展、统一的编程环境，是目前最为通用的并行编程环境之一。

随着 Google 的 MapReduce 分布式平台的出现，一些计算复杂度很高的计算可以在可接受的时间内完成。Apache 基金会基于 MapReduce 思想开发了 Hadoop 开源项目。Hadoop 分布式计算框架作为一个开源项目，能够用于构建云计算环境(分布式计算)，通过计算能力均匀地分布到集群中的多个计算节点上，从而实现了对超大数据集的巨大计算能力。Hadoop 具有高数据吞吐量，实现了高容错性和很高的可靠性、可扩展性，它由两个主要部分组成：HDFS(分布式文件系统)和 MapReduce 编程模式。同时，通过结合 K-Means 串行传统算法及 MapReduce 的编程模式，采用相应的并行策略将其移植到 Hadoop 平台上进行分布式数据挖掘计算。而将 Hadoop 平台技术运用到数据挖掘算法中，一个关键的问题就是如何实现将传统的数据挖掘算法实行并行化[6]。

其中，MapReduce(映射和规约)编程模型可以使用户在不关心底层细节的情况下方便开发出分布式计算程序。整个运算过程中，MapReduce 模型都是使用〈*key*，*value*〉的键值对形式作为输入和输出。它简化了并行计算的编程模型，只是向上层用户提供可用的接口。

3 改进算法

3.1 基于 MPI 的并行 K-Means 改进算法

目前，经典的基于 MPI 的编程采用的是 Master/Slave 编程模式[5]，属于两层架构类型。Master 节点负责调度和汇总。在这种架构模式下，Master 节点容易产生严重的单点瓶颈问题，并且 Slave 节点的计算等汇总会对 Master 节点产生大量的负载压力。为了解决这个问题，改进算法中采用的总体架构是 3 层树状主从模式，如图 1 所示。

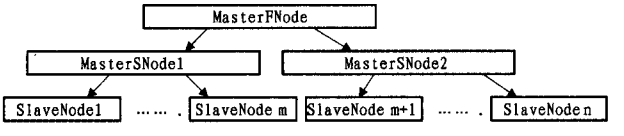

图 1 Master/Slave 3 层树状主从式

叶子 Slave 节点先将运算结果提交给中间层节点(MasterSNode)，经过该部分节点的处理后再传递给根(首层)节点(MasterFNode)，即可降低首层节点的负载和产生单点故障的可能性。MasterSNode 节点作为中间层，分担了 MasterFNode 的处理需求，也可并行地从 Slave 节点中读取数据，并行化地加速数据处理流程。

同时，在这个算法优化中，对参数 k 的选取提供了两种策略：一种是由用户指定，一种是由系统生成最优 k 值。算法尝试若干组不同的 k 值，利用准则函数值最小的 k 值作为最优 k 值进行数据聚类。准则函数的计算方法是各个数据对象与其所属聚类聚心的距离平方和。准则函数参见 2.1 节传统算法描述部分。

(1) 聚心初始化问题

首先，对于 K-Means 算法本身 k 值的选取，如果能更好地选取初始聚类中心，就能够减少聚类时间的消耗，获得较高

的加速比。

对于 K-Means 的聚心初始化问题，采用了一种新的思路和方法。初始化聚心是算法性能的瓶颈，保证了初始化数据的准确性就能够加快算法的处理速度，也降低了出错的概率。然而现在应用的初始化方法都过于繁杂[8]。本算法中采用的是多次初始化的思想，即先从数据中采样部分数据，对所有数据对象求出均对象，再对均对象随机化处理，生成若干组聚心。利用每组聚心对样本数据进行划分，根据划分选择最优情况下的一组聚心作为初始化聚心进行后续处理。判优方式为样本划分得越均匀，此次初始化就越成功。

(2) 通信量的角度

其次，本文改进的并行算法还从减少通信量的角度来提高并行算法的加速比。在通信模式上尽力压缩通信量，可增加算法的效率。

算法将在迭代中设置标志位。若有某簇已经迭代稳定，则标志位置为 1。主节点将根据标志位的取值来决定发送的通信量。所以随着算法的进行，有些簇已经稳定，则主节点发送的数据将会递减，这样就逐步减少了通信量。因此，Slave 节点随迭代的进行逐步减少处理量和通信量，直至为空算法结束。

定义 1 给定待处理数据对象(多维向量) data。对于簇聚心结构 (id 为簇号)，数据结构为 struct centroid{data,id}，节点通信数据包结构为 struct ACK{centroid,n,flag}。其中，centroid 为聚心结构；n 为从属节点中属于这个簇的数据对象个数；flag 为局部簇标志，用于标识改动。

定义 2 根节点(首层节点)表示为 MasterFNode，中间层节点为 MasterSNode，叶子节点(第三层节点)为 Slave。

整个改进的算法具体流程如下。

算法 1 对于根节点 MasterFNode

步骤 1 算法初始化

A)将待分类的数据分成 x 个子集(x 为 Slave 节点个数)。

B)初始化判断结束标志的数组 flag[k]为全 0。

步骤 2 将数据子集发送至 MasterSNode 中间层节点，并接收算法 2 步骤中的 MasterSNode 节点发送候选聚心数组。

步骤 3 采用样本数据进行划分，记录均匀度，再接收 MasterSNode 节点发送的候选聚心数组若干组，选取最优的作为 k 个初始化聚心。

步骤 4 将 k 个聚心结构体 centroid 发送至 MasterSNode 节点。

Repeat 步骤 5

步骤 5 接收从 MasterSNode 节点发来的 ACK 结构体数组。

A)遍历所有 ACK 数组；

B)根据 flag 标志和所属簇 id，重新修订 flag 数组的值。

If 所有 ACK 结构中属于同一簇的 flag 全为 1，then 设置 flag[j]=1，j 为簇号。And if 标志数组 flag 全为 1，则循环结束，退出。

Else 向 MasterSNode 节点发送簇号 i(满足 flag[i]=0 的簇中心 centroid 数据结构)。

End

算法 2 对于中间层节点 MasterSNode

步骤 1 算法初始化

A)接收数据子集，并把数据发送到算法 3 中处理的叶子 Slave 节点上。

B)收集该部分的子节点发来的初始化聚心 centroid，并上传至算法 1 中处理的根节点 MasterFNode。

Repeat 步骤 2—步骤 5

步骤 2 接收根节点 MasterFNode 发送的 k 个聚心结构体 centroid。

If 接收到退出请求，then 退出循环。

步骤 3 将 k 个聚心结构体向下传送至指定叶子 Slave 节点上。

步骤 4 接收 Slave 节点发送的 ACK 数组。

步骤 5 合并 ACK 数组，上传发送至 MasterFNode。

End

算法 3 对于叶子节点 Slave

步骤 1 算法初始化

A)接收子集数据且存储，计算子集数据的平均值和构造若干次 centroid 结构。

B)接收聚心数组 centroid，设置局部簇标志 flag[g]，并置为 0。

Repeat 步骤 2—步骤 5

步骤 2 执行传统 K-Means 算法部分，

For data in 数据子集

比较 data 与 centroid 中的 data，找到最小距离的簇，将 data 分配给 i 簇。

If data 上轮所属簇 id 在本轮 centroid 中不存在，

then 将 data 删除处理子集。

步骤 3 根据新的数据点划分，求平均值，计算新的聚心 centroid。

步骤 4 执行新旧聚心比较

If 相同，then 局部簇标志置 1，Else 置 0。

步骤 5 发送 ACK 数组至 MasterSNode 中间层节点，并接受 MasterSNode 中间层节点的 centroid 聚心数组。

If 接收到退出请求，then 退出。

End

3 层节点的数据通信过程如图 2 所示。

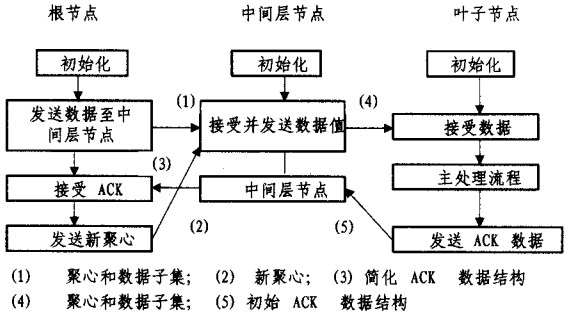

图 2 3 层节点的数据通信图

定义 3 运行时间主要由通信的时间和计算的时间组成。假设数据集的大小是 N，子节点有 W 个，传送一个数据的时间为 T_t。

本算法的总时间由下列时间部分构成：(1) 划分数据及初始化聚心需要时间 $N+k\times n$(n 初始化的次数)；(2) 传送数据 T 需要时间 $N/W\times T_t/2$；(3) 发送聚心需要时间 $k\times T_t$；(4) 发送信息至中间层节点需要时间 $(W-3)\times 5\times T_t\times k/2$；(5)中间层节点发送至根节点需要时间 $2\times 5\times T_t\times k$；(6) 计算中间层节点需要时间 $(W-3)\times k$；(7)计算全局聚心需要时间 $2\times k$。

上述过程迭代 m 次所需总时间为 $T=(1)+(2)+m\times((3)+(4)+(5)+(6))+(7)$。因此，根据文献[5]中的运行时间分析，本文提出的并行优化算法并没有增加算法的时间复杂度。

3.2 基于 Hadoop 的分布式 K-Means 改进算法

目前，Hadoop 上的 K-Means 算法已经有学者做过相应的研究与实现[6]，然而已有的实现通信量都过大，而且 Hadoop 系统也在算法层面上浪费了大量的资源。因此，我们对

算法做出优化和改进，以加快算法分布式处理速度。其主体架构如图 3 所示。

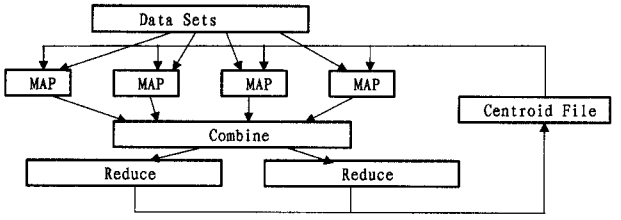

图 3　基于 Hadoop 的分布式 K-Means 算法架构

采用下面的优化思路：(1) Reduce 程序可由单节点运行改为多节点运行；(2)分布式数据挖掘平台运行的优化，本次应用的开发加入了 Combine 函数操作，Combine 能够减少中间结果中〈*key*，*value*〉对的数目，从而减少网络流量；(3)程序本身的优化，本文基于 Hadoop 的并行 K-Means 算法处理可以分为初始化工作和迭代工作两个部分。下面从 Map()，Combine()，Reduce() 3 个函数上说明处理的流程。

第一阶段，即初始化工作。对于 Map 函数，设置 *map* 个数为聚类个数，且以多维数据集文件分片为输入，通过输入数据分片下的数据对象均值的计算，可以求出数据对象均值。而对于 Reduce 函数，以 *k* 个数据对象均值为输入，直接输出到文件系统上的指定文件 centroid file 上。该阶段中 Combine 未操作。

第二阶段，即迭代工作。对于 Map 函数，通过多维数据集分片和 centroid file 上数据的输入，对数据分片中的每个数据实施聚心的距离计算，求出距离最小的聚心。如果数据分片中的对象所属簇 *id* 在本阶段 centroid file 中没有找到，则将数据进行删除处理和集中，以逐步减少处理量。因此，可以得到类编号 *key* 和数据对象 *value* 值。而对于 Reduce 函数，进行 Combine 相似的操作，对相同 key 值的键值对进行规约操作，生成新一轮聚心，并与 centroid file 中的值进行比较。需要将 Combine 的输出〈*key*，*value*〉对作为输入。最后，如果比较相同，则无输出，否则输出新聚心到 centroid file 中。同时，Combine 函数配合 Map 和 Reduce 函数操作，需要对相同 *key* 值的键值对进行规约。而以 Map 函数的输出〈*key*，*value*〉对则为输出 Reduce 函数需要的输入 *key*、相同 *key* 值情况下所有 *value* 的均值及相同 *key* 的个数的组合结构。

第三阶段，即整体执行和调整工作。保证初始化工作进行一次，而迭代工作循环进行，直至最后的比较中无输出，整个并行聚类过程结束。

4　实验及其结果分析

4.1　实验环境

本实验由两部分优化算法组成，所以测试的环境不同。基于 MPI 的改进算法，在两台 CPU 主频为 2.0GHz、内存为 2GB、操作系统为 Windows 7 的计算机上进行，fork 出 8 个进程供程序运行；开发环境：MicroSoft VC 2008。而基于 Hadoop 的改进算法，在一台主频为 2.8GHz 的 4 核 i5 处理器，内存为 4GB、操作系统为 Windows server 2003 上创建的 3 台 ubuntu10.04 虚拟机上进行；开发环境：eclipse＋Hadoop plugin。

4.2　实验结果

为了验证算法的优越性，实验选择将改进算法与传统的 K-Means 算法、已有的基于 MPI 的并行 K-Means 算法[5]、已有的基于 Hadoop 的算法[6]做比较，分别用 10000，100000，1000000，2000000 条数据来进行测试。数据样本如文献[5]的中数据方式构造多维数据。经过运行及实验计算得到表 1 所列结果。

表 1　算法运行时间比较表

模式\数据量(个)	10000	100000	1000000	2000000
常规 K-Means	168ms	8782ms	3526544ms	7225487ms
已有 MPI 策略	83ms	817ms	7999ms	16281ms
改进 MPI 策略	61ms	650ms	5701ms	13103ms
已有 Hadoop 策略	872ms	2472ms	59149ms	103241ms
改进 Hadoop 策略	3715ms	5565ms	16578ms	27618ms

从表中的数据可以看出，基于 MPI 的并行方式比基于 MapReduce 的 Hadoop 要快一些，这是因为 Hadoop 包含了过多的应用架构逻辑。当处理数据进一步增大成海量数据时，应用逻辑的耗时将成为总耗时的一部分，这时 MPI 的快速处理优势会进一步展现。然而，Hadoop 快速的开发周期和稳定的运行效果，是 MPI 所不及的，所以两种方式都有各自的优势。

常规的 K-Means 算法由于其串行执行的特性，耗费时间是最多的。运用并行方式后，数据处理的速度显著提高了，对 MPI 的 Master/Slave 模式进行改进后得出的算法对数据处理的时间有了明显的提升。本文算法的改进优势是明显的，虽然数量级上还保持一致，但是减少的时间会随着数据量的增加而体现得更加明显，如图 4 所示。

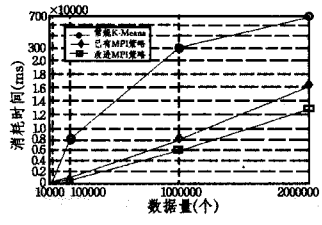

图 4　基于 MPI 的并行 K-Means 改进算法比较

而对于基于 Hadoop 的分布式 K-Means 改进算法，当处理小量数据时，改进的 Hadoop 比常规的耗时更多。这是因为此时数据还不是主导时间的因素，处理过程的增加导致改进算法的耗时比常规多。然而，当数据量增至 1000000 时，改进算法的优势就体现出来了。而且随着数据量的增加，优势也会越来越明显，如图 5 所示。

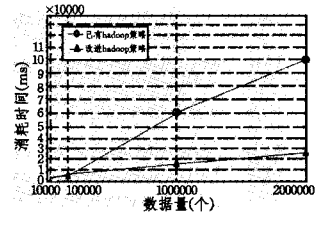

图 5　基于 Hadoop 的分布式 K-Means 改进算法比较

结束语　传统的数据挖掘算法在面对大规模数据时表现出计算能力不够等缺陷，同时也伸缩性、扩展性也较局限，使得数据挖掘算法不能拥有很好的大数据集挖掘能力。由于云计算和并行计算技术提供的巨大计算能力和低廉的实现方式，研究者都尝试着实现两种环境下的聚类快速划分算法。本文在已有的研究基础上进行算法的改进和重新构造，利用

(下转第 151 页)

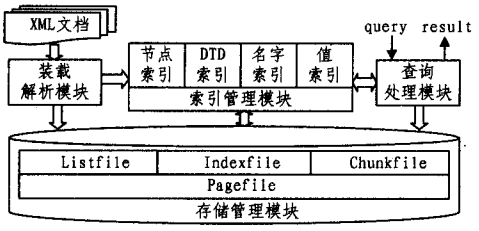

图 4 改进的 XISS 系统结构

5.3 查询测试数据

仿真实验用到的实验数据和查询语句见表 1 和表 2。

表 1 XML 数据集

XML Data	Size (Byte)	Files	Elements	Attribute	DTD File (Byte)
Shakespeare'S play	7.9M	37	327k(22)	0	2k
SigmodRecord	3.5M	989	839k(47)	4775(3)	1k

表 2 查询语句

XML Data	Path Query	Query
Shakespeare'S play	Play/title	Q1
	Play/person group/grades;	Q2
	//act//speech	Q3
	play/act[title="ACTI"]	Q4
	play/scene	Q5
SigmodRecord	sigmodrecord/number	Q6
	//issue/articles/article/title	Q7
	//article/title[articlecode="0001"]/authors	Q8
	/authors[authorposition="01"]	Q9

5.4 结果与分析

5.4.1 建立索引比较

表 3 为两种索引仿真结果。从表 3 对比结果可知，传统 XISS 和改进的 XISS 索引的建立时间和索引大小均相差不大。主要是因为改进的 XISS 索引对 XML 文档和 DTD 同时建立了索引。但由于 DTD 比 XML 文档要小得多，因此它对建立索引总时间影响相当小。

表 3 两种索引建立索引比较

XML Data	XISS		改进的 XISS	
	建立索引时间 (S)	索引大小 (MB)	建立索引时间 (S)	索引大小 (MB)
Shakespeare'S Play	45	24.5	39	21.7
SigmodRecord	38	22.8	37	19.8

5.4.2 查询响应时间

对于两种索引技术的查询响应时间，数据集 Shakespeare'S Play 和 SigmodRecord 的查询结果分别如图 5 和图 6 所示。从仿真结果可知，对于 3 个带有谓词的查询 Q4，Q8，Q9，其查询效率明显提高，简单查询效率也有不同程度提高。因此，改进的 XISS 索引提高了 XISS 索引查询效率。

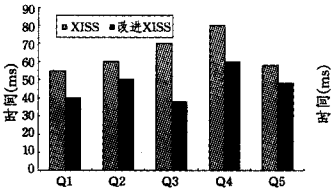

图 5 Shakespeare's Play 查询结果

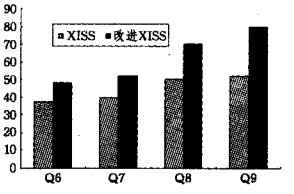

图 6 SigmodRecord 查询结果

结束语 随着 XML 应用的日益广泛，快速准确地查询 XML 文档中的数据已经越来越受到重视。XISS 索引是一种典型的支持正则路径表达式的索引结构，采用查询子路径表达式和中间结果间的连接来对输入进行查询，存在对复杂路径表达式查询效率低的缺陷。本文的 XISS 索引，将模式信息加入到 XML 文档的索引结构中，对 XISS 的索引结构和分解子路径表达式分别进行了改进，最后对算法进行了仿真实验。实验结果表明，改进的 XISS 索引缩短了建立索引的时间，加快了查询速度，从而提高了查询效率。

参 考 文 献

[1] 春平，王超，张鹏. XML 编程从入门到精通[M]. 北京：北京希望电子出版社，2002

[2] 王静，孟小峰，王宇，等. 以目标节点为导向的 XML 路径查询处理[J]. 软件学报，2005，16(5)：827-837

[3] 张鹏，冯建华，房志峰. 一种基于二叉树的 NativeXML 数据库文档编码机制[J]. 计算机应用，2008，28(9)：2331-2334

[4] 张博，耿志华，周傲英. 一种支持高效 XML 路径查询的自适应结构索引[J]. 软件学报，2009，20(7)：1812-1824

[5] 颖捷. XML 索引与查询的若干关键技术研究[D]. 上海：复旦大学，2008

[6] 秦玉杰，李革，黄柯棣. 基于 XML 技术的三维仿真模型的存储[J]. 计算机仿真，2005，22(12)：4-7

[7] 刘振中，董道国，薛向阳. 对 XML 数据索引的回顾[J]. 计算机科学，2004，31(4)：78-83

[8] 陈荣鑫. 基于函数式中间语言的 XML 查询并行化[J]. 重庆理工大学学报：自然科学版，2011，25(7)：81-86

(上接第 137 页)

MPI 和 Hadoop 两种编程模式来进行新的算法搭建，不仅充分利用了这两种技术的优势，同时将多种设计中的优化思想吸收入到该算法中，提高了已有的 K-Means 聚类算法的效率。实验结果证明，所采用的思想和具体实现是有效的，希望对后续研究开发者有启发和应用价值。

参 考 文 献

[1] Wikipedia. K-Means clustering[EB/01]. http://en.wikipedia.org/wiki/K-Means

[2] 崔建，李强，杨龙坡. 基于垂直数据分布的大型稠密数据库快速关联规则挖掘算法[J]. 计算机科学，2011，38(4)：216-220

[3] 郑湃，崔立真，王海洋，等. 云计算环境下面向数据密集型应用的数据布局策略与方法[J]. 计算机学报，2010，33(8)：1472-1480

[4] 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

[5] 冯丽娜. 并行 K-Means 聚类方法在简历数据中的应用研究[D]. 云南：云南大学，2010

[6] 杨宸铸. 基于 HADOOP 的数据挖掘研究[D]. 重庆：重庆大学，2010

[7] Kantabutra S, Couch A L. Parallel K-Means Clustering Algorithm on NOWS[J]. Technical Journal, 2000, 6(1): 243-247

[8] Forman G, Zhang B. Distributed Data Clustering can be Efficient and Exact[J]. SIGKDD Explorations, 2000, 2(2): 34-38

[9] Boutsinas B, Gnardellis T. On Distributing the Clustering Process[J]. Patter Recognition Letters, 2002, 23(4): 999-1008

[10] 梁红，李伟生. XML 文档的并行聚类算法[J]. 计算机科学，2004，31(10)：243-245

[11] Quinn M J. ParM: Pallel Programming in C with MPI and OpenMP[S]. Beijing: Tsinghua University Press, 2005