一种基于多核机群架构的混合索引结构

龙 柏[1,2],孙广中[1,2],熊 焰[1],陈国良[1,2]
(1.中国科学技术大学计算机学院,安徽合肥 230026; 2.安徽省高性能计算重点实验室,安徽合肥 230026)

摘 要: 本文提出了一种HKD-tree(Hybrid K-Dimensional tree)混合索引结构.该结构将KD-tree(K-Dimensional tree)和LSH(Locality Sensitive Hashing)两种索引结构进行组合,利用KD-tree作为上层结构的主干而LSH充当叶子节点,从而可以利用多核机群系统的层次并行结构特性.与传统的索引结构相比,该混合索引结构具有高效并行处理、可扩展性好等特点,适于多核机群系统平台及高维数据索引.实验结果表明,该混合索引结构在多核机群系统上的性能优于传统的索引结构.

关键词: 索引; HKD-tree; 高维数据; 多核机群; LSH
中图分类号: TP391.3 **文献标识码:** A **文章编号:** 0372-2112 (2011) 02-0275-05

A Hybrid Index Structure Based on Multi-Core Cluster

LONG Bai[1,2], SUN Guang-zhong[1,2], XIONG Yan[1], CHEN Guo-liang[1,2]
(1. *School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui* 230026, *China*;
2. *Key Laboratory on High Performance Computing of Anhui Province, Hefei, Auhui* 230026, *China*)

Abstract: We present a hybrid-index structure for high-dimensional data which named HKD-tree (Hybrid K-Dimensional Tree). To make use of two-level parallelization of multi-core clusters, we combined with KD-tree and LSH, which uses LSH in the leaf nodes of KD-tree. Compared with the traditional index structure, the hybrid index structure has effective parallel processing ability and good scalability, which is suitable for the multi-core cluster platform and high-dimensional data indexing. The experiment results show that the performance of the hybrid index structure is superior to the traditional index structure on the multi-core cluster systems.

Key words: index; HKD-tree; high-dimensional data; multi-core; LSH

1 引言

高维索引技术是研究如何通过建立有效的索引结构来实现高维数据集合检索的技术.针对这一经典问题,众多学者提出了一系列的索引结构,大致分为三类[1]:(1)基于数据空间划分的树形索引,如KD-tree、R-tree和Trie-tree及其变体[2]等;(2)采用近似方法来表示原始向量,如VA-file和IQ-tree等;(3)基于距离尺度的方法,如NB-tree和LSH(Locality Sensitive Hashing)等.其中,第一类方法仅适合低维度的情况,随着维数增加索引性能会快速下降;第二类方法则通过对高维数据进行压缩和近似存储从而提高检索速度,但检索精度和CPU检索时间方面表现不佳;对于第三类方法,它们通过降维或者计算距离值来进行高维检索,性能相对较好.

随着多核处理器[3]逐渐成为市场的主流,基于多核的机群计算方式也日益成为并行计算的平台.多核机群系统作为高性能并行计算平台具有良好的性能和潜力优势.目前SMP(Symmetrical Multi-Processing)[4](对称多处理器)机群系统正逐步成为广泛使用的高性能并行计算平台.如图1所示,该架构综合了SMP和机群的结构特点,具备节点间分布式存储和节点内共享内存的层次结构,支持节点间消息传递和节点内共享内存.只有在设计并行程序时有效结合两种结构的优势,才能充分利用该系统所提供的两级并行性,在节点间和节点内均可并行获得更好的加速比和运行效率.

传统的高维索引结构,例如KD-tree,LSH(Locality Sensitive Hashing),R-tree,S-tree等索引结构都是基于串行计算平台的,没有考虑并行计算的因素,往往不能充分发掘并行平台的特性.需要寻找有效途径利用现有的索引结构立足于高性能计算平台来解决高维索引问题.

综合现有的高维索引技术以及高性能并行计算平台的特点,本文提出一种基于SMP机群架构的混合索

收稿日期:2009-11-06;修回日期:2010-05-10
基金项目:国家自然科学基金(No.60533020, No.60873210);安徽省自然科学基金(No.090412064);国家高技术研究发展专项经费(No.2009AA01A134);中央高校基本科研业务费专项资金

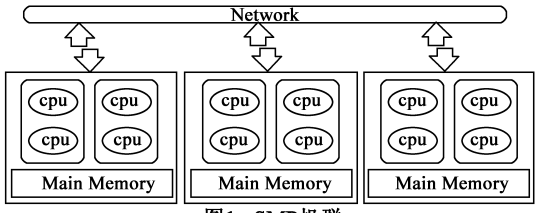

图1 SMP机群

引结构 HKD-tree(Hybrid K-Dimensional tree).它与 SMP 机群系统的层次并行结构相匹配,利用 KD-tree 作为其上层结构的主干而 LSH 充当叶子节点,充分利用 SMP 机群系统所提供的两级并行性,使得节点间和节点内的并行性均可较好发挥.与传统的索引结构相比,该混合索引结构具有支持维数高、复杂度较低并且支持并行结构良好等特点.使用随机生成数据和真实系统中的数据进行测试,实验证明该混合索引结构性能明显优于传统的索引结构.

2 相关工作

为解决高维索引问题,研究者提出了相当数量的高维索引结构.在这些高维索引结构中,它们受到应用背景、数据分布以及维数大小等因素的影响,具备不同的性能.一般来说,树型索引结构是使用最为广泛的索引结构,它的构建是基于数据空间的层次化聚类原则.按照节点构建方式可分为数据划分和空间划分两大类.基于数据划分的树型索引结构包括 R-tree 及其变种、X-tree、SS-tree 等;基于空间划分的树型索引结构包括 KD-tree,KDB-tree 和四叉树等;其他的高维数据结构还有 TV-tree、空间填充曲线、金字塔树等.下面介绍几种常见的高维索引结构.

2.1 KD-tree

KD-tree(K-Dimensional tree)[5] 是由 Jon Bentley 在 1979 年发明一个数据结构.作为一个经典的数据结构,KD-tree 及其变种是高维数据集的最流行索引结构.

KD-tree 是一种由二叉搜索树推广而来的用于高维数据检索的树结构形式(k 即为空间的维数).它与二叉搜索树不同的是它的每个结点表示 k 维空间的一个点,并且每一层都根据该层的分辨器对相应对象做出分枝决策.顶层结点按一个维划分,第二层则按照另一维划分.以此类推在余下各维之间不断地划分,直至一个结点中的点数少于给定的最大点数时,结束划分完成建树.

KD-tree 的第 n 层分辨器的定义为:$n \bmod k$(树的根节点所在层为第 0 层,根结点孩子所在层为第 1 层,以此类推)即:若它的左子树非空,则其左子树上所有结点的第 k 维值均小于其根结点的第 k 维值;若它的右子树非空,则其右子树上所有结点的第 k 维值均大于其根结点的第 k 维值;并且它的左右子树也分别为 KD-tree.

2.2 LSH

LSH(Locality Sensitive Hashing)是由 Indyk 和 Motwani 近年来新提出的一种高维索引结构[6,7].它的基本思路是针对数据点集,利用一组特定的哈希函数来建立哈希表.从而使得在一定的相似度量条件下,相似点映射到相同的区域的概率大于非相似点.

定义 1 一组哈希函数 $H=\{h_1,\cdots,h_k\}$,对于数据点 p 和 q,有

(1)若 $D(p,q)<r_1$,则 $P[h_i(q)=h_i(p)]>p_1$;

(2)若 $D(p,q)>r_2$,则 $P[h_i(q)=h_i(p)]<p_2$.

其中 P 为概率函数,i 为随机数且 $i\in\{1,\cdots,n\}$.从而得到的哈希函数被称为以 (r_1,r_2,p_1,p_2) 为参数的函数组.

对于 d 维的海明空间[8]中的数据点 $p=\{p_1,\cdots,p_d\}$,$q=\{q_1,\cdots,q_k\}$.它们的海明距离被定义为不同位的个数,此时令 $r_1=r$,$r_2=(1+c)r$,则哈希函数组 $H=\{h_1,\cdots,h_d\}$,$h_i(p)=p_i$,成为一个以 $\left(r,(1+c)r,\frac{1-r}{d},\frac{1-(1+c)r}{d}\right)$ 为参数的 LSH 函数组.通过这些函数组将数据置入哈希表中.

2.3 其他

其他高维索引结构有:KDB-tree[9]、R-tree[10]、S-tree[11]和 SH-tree[12]等等.基于前面的分析,上述索引结构都是基于串行架构的,如果把它们用于多核机群系统时,就显得不大匹配.因此,基于这些现有的高维索引结构,我们提出下面的混合索引结构及其并行算法以适用于高维数据索引的多核机群系统.

3 HKD-tree 混合索引结构

在本节中,我们介绍 HKD-tree(Hybrid K-Dimensional tree)并讨论如何建立一个 HKD-tree 索引结构,以及如何将其应用于多核架构高性能并行计算平台的问题.

3.1 混合索引结构

我们首先讨论基于多核机群的高维索引结构 HKD-tree.形如 KD-tree,HKD-tree 也是针对 k 维空间数据集划分的数据结构.我们综合以往的高维索引技术,将 KD-tree 和 LSH 两种索引结构有效的结合起来得到 HKD-tree 索引结构.在这种混合索引结构中,使用 KD-tree 来构建整个树形结构的主干,而 LSH 充当其枝叶.

如图 2 所示,构造一棵 HKD-tree 分为两步.首先构造上层的 KD-tree,其次构造下层的 LSH 枝叶.首先,利用 KD-tree 对数据集进行空间划分构造 HKD-tree 的上层结构.具体过程为:

(1)判断 KD-tree 是否为空,若为空则直接作为根

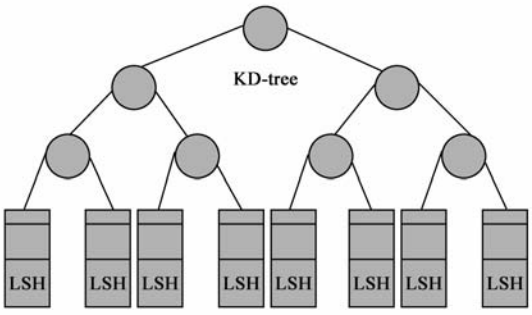

图2　HKD-tree结构

结点.否则比较该点与 KD-tree 根结点相应维的值的大小关系,进入其左、右子树进行下一步操作;

(2)若该点小于根结点相应维的值,则进入左子树进行查找操作直至某个结点的左子树或右子树为空.则将该点插入作为其叶子结点;

(3)若该点大于根结点 i 维的值,则同(2)理进入右子树进行插入操作.

然后,在该 KD-tree 的叶子节点上挂上 LSH 结构,即将剩余的数据点置入 LSH 中.利用 LSH 的构建方式完成剩余的 HKD-tree 构建工作.具体过程为:

(1)将数据集 X 转化为海明空间中的二进制串;

(2)选取恰当的 $r>0$, $c>1$,随机选取 k 个形如 $g_i(p)=\{h_{i_1}(p),h_{i_2}(p),\cdots,h_{i_n}(p)\}$ 的哈希函数;(r, c 为参数);

(3)利用这些哈希函数将数据点存入相应的哈希表中.

具体的 HKD-tree 构建算法如算法 1:

算法 1　HKD-tree 的构建(数据集为 X)

(1) if A = *Null* then;
(2) *root*: = P;
(3) else if A is the Lchild of X then;
(4) X. *Lchild*: = P;
(5) else;
(6) X. *Rchild*: = P;
(7) else if $P[i]<A[i]$ then;
(8) HKD-tree Insert(A. *Lchild*, P, A);
(9) else;
(10) HKD-tree Insert(A. *Rchild*, P, A);
(11) 对于 LSH 部分,运用前面提到的哈希函数,存入哈希表直至完成;
(12) end if.

通过前面对形如 KD-tree 和 LSH 等索引结构的讨论,可以看出此类索引结构较为单一,不适用于双层的并行结构.通过对 KD-tree 和 LSH 索引结构的有效融合后,使得这两种结构相互补充,为高效的并行化提供了可能.

3.2　并行查询算法

基于上述内容我们在此提出 HKD-tree 的并行算法.如图 1 和图 2 所示,可以发现 HKD-tree 的逻辑结构和 SMP 机群的并行结构具有很高的相似度.利用这一特性使得 HKD-tree 高维索引结构在 SMP 机群系统上可进行高效的并行化.主要有两个步骤:

(1)选取一个计算节点作为主节点构建 HKD-tree 的上层结构,即 KD-tree 的构建;

(2)当硬件计算节点的数目和 HKD-tree 拥有的子树数目相等时展开并行化操作,一个计算节点进行一个子树全部相关操作(即 LSH 构建).将整个 HKD 树的任务平均分配给每个计算节点,从而将并行化的问题有效解决,然后合并返回值.

基于 HKD-tree 混合索引结构以及 SMP 机群系统,需要考虑的关键问题是数据划分问题,也就是选择一个恰当的时机和位置发挥并行计算的优势.问题的关键在于,何时让 LSH 参与进来.以查询为例,首先,令查询目标为 K 维向量 $\boldsymbol{p}$,KD-tree 的深度为 h,则所需查询时间为 $T_s=T_1+T_2$.其中 T_1 为 KD-tree 部分的查询时间;T_2 为 LSH 部分的查询时间.易知 T_1、T_2 均为与 h 有关的函数,且 T_1 随 h 的增加而递减,T_2 随 h 的增加而递增.从而 T_s 存在最优解为 $\min(T_1+T_2)$.其次,针对 HKD-tree 及 SMP 多核机群的结构特点,根据 KD-tree 的性质,每个子树是由同一个分辨器进行分类所得.在进行相关操作时获得的效率最高.同时考虑并行处理中节点间通信以及硬盘读写的代价问题,将 LSH 置于同一个计算节点内处理更为适合.

算法 2　HKD-tree 并行查询算法

(1) if A = *Null* then;
(2) return *Null*;
(3) else if A = P then;
(4) return A;
(5) else if $P[i]<A[i]$ then;
(6) HKD-tree Search(A. *Lchild*, P);
(7) else;
(8) HKD-tree Search(A. *Rchild*, P);
(9) If 硬件设备有 n 个计算节点并且构建的 HKD-tree 在 $\log_2(m+1)$层上有 m 个子树;
(10) 当 $m=n$ 时,分配每个计算节点执行一棵子树的 LSH 相关操作展开并行查询操作;
(11) 对于查询 q,运用构建中用到的哈希函数,提取 $g_i(q)$,$1\leqslant i\leqslant k$ 所命中的哈希表项;
(12) 对这些表项顺序排序,记得到查询结果;
(13) 处理各子树返回值,得出查询结果;
(14) end if.

综上所述,我们采用此数据划分方案.即假设已构建的 KD-tree 具有的子树数目为 m,目前已有的硬件设备计算节点数为 n.从而由 $m=n$,得出 LSH 位于 HKD-tree 的第 $\log_2(m+1)$层.即 $h=\log_2(m+1)$时 T_s 存在最小值.我们认为在硬件设备的处理器数目(计算节点

数)与 KD-tree 的子树数目相等时,将 LSH 挂载到 KD-tree 上充当叶子结点最为合适.

HKD-tree 并行查询算法如算法 2.

4 实验

在实验中,利用统一的数据集并基于两路四核的多核 SMP 机群系统(Quad-Core AMD Opteron(tm) 2.0GHz 8G 内存)对 KD-tree、LSH 和 HKD-tree 三种索引结构在不同维数和不同数据规模下的查询时间进行比较.从而客观的评价这三种索引结构在高维数据下的性能表现.针对 LSH 结构的特点[13],在此取 $r=4, c=2$.

实验的源代码都通过 C 语言实现,具体到并行程序我们利用 OpenMP[14] + MPI[15] 相结合的方式实现.其中 LSH 部分基于 Alexandr. Andoni 提供的部分源代码(http://web.mit.edu/andoni/).我们进行了代码优化和改进并将其进行了并行化实现.

随机生成数据 利用程序随机生成了不同维数特征向量:16,32,48,64,80,96 和 128.向量集合的规模为 1000.如图 3 所示,将未并行化的 HKD-tree 混合索引结构与 LSH 和 KD-tree 相比较.在不同的维数下,三种索引结构的性能表现依次为 HKD-tree、LSH、KD-tree.同时可以看出 LSH 和 KD-tree 的表现相当.

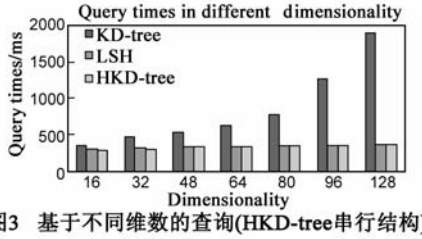

图3 基于不同维数的查询(HKD-tree串行结构)

在对上述索引结构进行并行化之后,基于 SMP 机群的实验结果如图 4,可以看出 HKD-tree 混合索引结构的性能明显的优于 LSH 和 KD-tree.

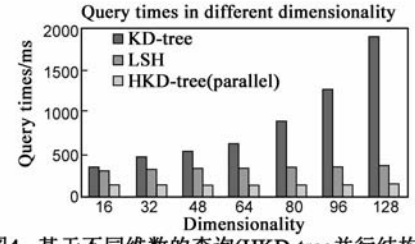

图4 基于不同维数的查询(HKD-tree并行结构)

系统生成数据 在基于图像内容的检索系统(http://search.ustc.edu.cn/cbir/)中,使用从互联网上下载的不同规模大小的图片库,并通过直方图处理和边缘检测[16]等方法进行特征提取,变为高维向量并进行索引.在实验中,选取的规模分别为 1000、2000、3000、4000 和 5000 的图片库.向量的维度为 640.如图 5 所示,根据不同的数据规模,HKD-tree 的表现最好.从而不难得出结论,HKD-tree 混合索引结构是合理的、有效并且较优的.

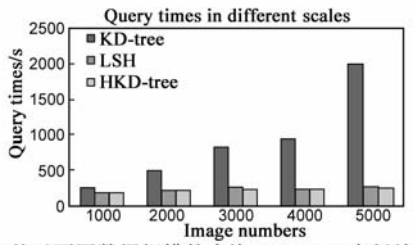

图5 基于不同数据规模的查询(HKD-tree串行结构)

同样,在对上述索引结构进行并行化之后,基于 SMP 机群的实验结果如图 6,可以看出 HKD-tree 混合索引结构的性能仍然明显的优于 LSH 和 KD-tree.

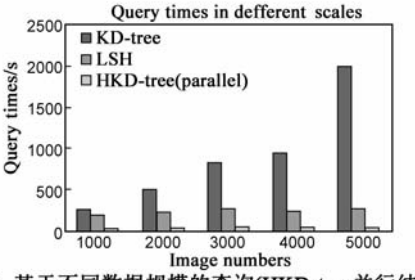

图6 基于不同数据规模的查询(HKD-tree并行结构)

综上所述,实验结果有力的证明了我们提出的 HKD-tree 混合索引结构的优良性能以及其对 SMP 机群系统的有效支持.对比上述实验结果可以得出并行化后的加速比达到 5.2.同时,实验结果还表明 HKD-tree 基于 SMP 机群系统架构下,检索性能将增加 30%左右.因此,将该混合索引结构用于 SMP 机群系统可显著提高检索性能.

5 结论

在本文中,我们结合 KD-tree 和 LSH 索引结构及算法提出一个有效的混合索引结构 HKD-tree,并且适时予以并行化使其与 SMP 机群系统结构相匹配,从而较为有效的解决高维数据的索引问题.HKD-tree 的优势在于继承了传统的高维索引结构的优势并紧密结合多核技术的发展.它的数据划分使之能够很好的支持并行搜索.实验表明,HKD-tree 混合索引结构及其并行算法可有效支持高维数据检索.总而言之,HKD-tree 混合索引结构及其并行算法在 SMP 机群系统架构下具有良好的性能.实验结果表明,HKD-tree 混合索引结构优于 LSH 和 KD-tree 索引结构.

参考文献:

[1] R Datta, et al. Image retrieval: Ideas, influences, and trends of the new age[J]. ACM Computing Surveys, 2008, 40(2): 1 - 60.

[2] 薛向阳,罗航哉,吴立德. LIFT:一种用于高维数据的索引结构[J]. 电子学报,2001,29(2):192 – 195.
Xue Xiangyang, Luo Hangzai, Wu Lide. LIFT: An index structure for high dimensional data[J]. Acta Electronica Sinica, 2001,29(2):192 – 195. (in Chinses)
[3] Michael Gschwind, et al. Synergistic processing in Cell's multicore architecture[J]. IEEE Computer Society, 2006, 26(2): 10 – 24.
[4] Austin Hung, et al. Symmetric multiprocessing on programmable chips made easy[A]. Proceedings -Design, Automation and Test in Europe, DATE '05[C]. Munich: Institute of Electrical and Electronics Engineers Inc, 2005. 240 – 245.
[5] J L Bentley. Multidimensional binary search trees in database applications[J]. IEEE Trans on Software Engineering, 1979, 5(4):333 – 340.
[6] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms or approximate nearest neighbor in high dimensions[J]. Communications of the ACM, 2008, 51(1):117 – 122.
[7] M Datar, et al. Locality-sensitive hashing scheme based on p-stable distributions[A]. Proc of the 20th Symposium on Computational Geometry[C]. New York: Association for Computing Machinery, 2004. 253 – 262.
[8] P Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation[J]. Journal of the ACM, 2006, 25(3):307 – 323.
[9] J T Robinson. The K-D-B-Tree: A search structure for large multidimensional dynamic indexes[A]. Proc. of ACM SIGMOD Int. Conf. on Management of Data[C]. New York: Association for Computing Machinery, 1981. 10 – 18.
[10] Antonin Guttman. R-trees: A dynamic index structure for spatial searching[J]. IEEE Trans Commun, 1984, 14(2): 299 – 609.
[11] Uwe Deppisch. S-tree: A dynamic balanced signature index for office retrieval[A]. Proc of the 9th ACM SIGIR Int Conf on Research and development[C]. New York: Association for Computing Machinery, 1986. 77 – 87.
[12] Dang Tran Khanh. The SH-Tree: A novel and flexible super hybrid index structure for similarity search on multidimensional data[J]. International Journal of Computer Science and Applicaions, 2006, 3(1): 1 – 25.
[13] Piotr Indyk, Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality[A]. Proceedings of 30th Symposium on Theory of Computing[C]. New York: Association for Computing Machinery, 1999. 604 – 613.
[14] Christian Terboven, et al. Data and thread affinity in OpenMP programs[A]. Proc. the 2008 workshop on Memory access on future processors[C]. New York: Association for Computing Machinery, 2008. 377 – 384.
[15] Gabriele Jost, et al. Comparing the OpenMP, MPI, and hybrid programming paradigms on an SMP cluster[A]. The Fifth European Workshop on OpenMP[C]. New York: Association for Computing Machinery, 2003. 1 – 10.
[16] J F Canny. A computational approach to edge detection[J]. Transactions on Pattern Analysis and Machine Intelligence, 1986, 8(6):679 – 698.

作者简介:

龙　柏　男,博士研究生. 1980 年生于安徽合肥. 主要研究方向为高性能计算、信息检索等.

孙广中　男,博士、讲师. 1978 年生于安徽蚌埠. 2000 年和 2005 年在中国科学技术大学分别获工学学士和工学博士学位. 主要从事并行计算、信息检索等方面的研究工作.

熊　焰　男,1960 年生,博士、教授、博士生导师,研究方向为分布式处理、计算机网络和移动计算以及信息安全.
Email: yxiong@ustc.edu.cn

陈国良　男,教授,博士生导师,中国科学院院士. 1938 年 6 月出生于安徽颍上. 1961 年毕业于西安交通大学无线电系计算机专业. 现任国家高性能计算中心(合肥)主任,主要从事并行计算、高性能计算等方面的研究工作.