ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.11, November 2006, pp.2289−2301
DOI: 10.1360/jos172289

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于数据网格的书法字 *k* 近邻查询*

庄 毅+, 庄越挺, 吴 飞

(浙江大学 计算机科学与技术学院,浙江 杭州 310027)

Answering *k*-NN Query of Chinese Calligraphic Character Based on Data Grid

ZHUANG Yi+, ZHUANG Yue-Ting, WU Fei

(College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: Phn: +86-571-87951853, E-mail: zhuangyi@cs.zju.edu.cn, http://www.zju.edu.cn

Zhuang Y, Zhuang YT, Wu F. Answering *k*-NN query of Chinese calligraphic character based on data grid. *Journal of Software*, 2006,17(11):2289−2301. http://www.jos.org.cn/1000-9825/17/2289.htm

Abstract: In this paper, a novel *k*-Nearest Neighbor (*k*-NN) query over the Chinese calligraphic character databases based on Data Grid is proposed. First when user in the query node submits a query character and *k*, the character filtering algorithm is performed using the hybrid distance metric (HDM) index. Then the candidate characters are transferred to the executing nodes in a package mode. Furthermore, the refinement process of the candidate characters is conducted in parallelism to get the answer set. Finally, the answer set is transferred to the query node. If the number of answer set is less than *k*, then the query procedure is re-performed by increasing the query radius until the *k* nearest neighbor characters are obtained. The analysis and experimental results show that the performance of the algorithm is good in minimizing the response time by decreasing network transfer cost and increasing parallelism of I/O and CPU.

Key words: Chinese calligraphic character; *k*-nearest neighbor query; cluster hypersphere; data grid

摘 要: 提出一种在数据网格环境下的书法字 *k* 近邻查询方法.当用户在查询结点提交一个查询书法字和 *k* 时,首先以一个较小的查询半径,在数据结点进行基于混合距离尺度的书法字过滤,然后将过滤后的候选书法字以“打包”传输的方式发送到执行结点,在执行结点并行地对这些候选书法字进行距离(求精)运算,最终将结果书法字返回到查询结点.当返回的书法字个数小于 *k* 时,扩大半径值,继续循环,直到得到 *k* 个最近邻书法字为止.理论分析和实验表明,该方法在减少网络通信开销、增加 I/O 和 CPU 并行、降低响应时间方面具有较好的性能.

关键词: 中文书法字;*k* 近邻查询;类超球;数据网格
中图法分类号: TP391 文献标识码: A

中国的悠久历史和灿烂文化留下了很多优秀的书法作品,如王羲之的《兰亭集序》等.这些作品虽然可以通过书名、作者、朝代名等元数据信息进行检索,但由于书法作品图像难以通过 OCR(optical character

* Supported by the National Natural Science Foundation of China under Grant No.60533090 (国家自然科学基金); the National Science Fund of China for Distinguished Young Scholar under Grant No.60525108 (国家杰出青年基金); the China US Million Book Digital Library Project (高等学校中英文图书数字化国际合作计划)

Received 2006-06-10; Accepted 2006-08-25

recognition)识别,因此无法根据书法字的内容进行高效检索.本文提出一种基于数据网格环境的书法字 k 近邻(k-NN)查询方法.为了充分利用网格中的资源,突出数据网格资源共享的特点,该算法把网格中性能较好的 h 个结点作为高维查询执行的结点,并且采用书法字过滤、"打包"传输和流水线并行机制来减少查询的响应时间.同时,本文对该算法建立代价模型,并且说明各种参数对查询性能的影响.由于 k-NN 查询是通过嵌套调用范围查询来完成的,当用户向数据结点发送一个查询请求时,首先利用基于混合距离索引对原始书法字集进行过滤,以减少网络传输的代价,再将过滤后的候选书法字以"打包"传输的方式发送到若干个执行结点,并行地完成候选书法字的求精(距离)运算.最后,将得到的结果书法字集发送回查询结点.这样就完成了一次高维书法字的范围查询.当返回的候选书法字个数小于 k 时,再通过增大查询半径 r 的方式再次执行基于数据网格的范围查询,直到满足条件为止.实验表明,该方法能够显著提高书法字检索的效率.

本文第 1 节回顾相关工作.第 2 节介绍书法字过滤及"打包"传输的方式.第 3 节提出网格环境下的 k-NN 查询算法.第 4 节给出查询的代价模型.第 5 节通过实验从不同角度证明该算法的有效性.最后是总结及对未来工作的展望.

1 相关工作

1.1 手写体检索

本质上说,书法字是一种手写体.关于手写体的识别曾有很多研究,文献[1]回顾了在线和离线手写体识别的主流技术,其中提到目前较为成功的手写体识别研究有对华盛顿手稿进行识别[2]、对希伯来语的书写体进行分类[3].然而,很少有文献介绍中文书法字的检索和索引方面的研究工作.文献[4]提出了一种古籍内容检索方法.该方法通过多级计算古籍中汉字质心的方式,成功地对书写规范的古籍汉字进行了检索,然而,对于书写不规范且来自不同朝代的书法作品,该方法难以奏效.本文采用轮廓来表示原始书法字,避免了上述问题.

1.2 数据网格

在数据网格研究领域,美国和欧洲等国已经进行了广泛、深入的研究,并且推出了一些实验系统,其中最著名的是欧洲数据网格项目[5]、美国的国际虚拟数据网格实验室 IVDGL(International Virtual Data Grid Laboratory)项目[6]等.最著名的数据网格系统工具是 Globus 中的数据网格支撑模块和 SDSC(San Diego Supercomputer Centre)的 SRB(storage resource broker)系统.到目前为止,数据网格环境下有关数据存储、访问和传输的大多数工作,都是针对分布式文件系统的;同时,虽然目前对网格环境下的传统数据库查询进行了一定的研究[7,8],但是还很少有文献研究基于数据网格的书法字 k 近邻查询.在数据网格环境下,各结点高度自治并且异构;所处理的数据一般都是海量数据;各结点之间的连接带宽不同,其传输速度可能会有很大的差异;网络环境不稳定,经常会出现结点之间连接不上以及连接中断的情况,这些都为基于数据网格环境的书法字 k 近邻查询提出了新的要求.

1.3 高维索引

书法字索引属于高维索引范畴.高维索引技术经历了 20 多年的研究[9],采用的技术主要分为 3 类.一类是以 R-tree[10]为代表的基于数据和空间分片的树形索引.可是,该树形索引方法只适合维数较低的情况,随着维数的增加,其索引的性能往往比顺序检索要差,称为"维数灾难".另一类是以 VA-file[11]为代表的用近似方法表示多维向量.尽管它通过对高维数据进行压缩和近似存储来加速顺序查找速度,然而,数据压缩和量化带来的信息丢失使得其首次过滤后的查询精度并不令人满意.同时,尽管减少了磁盘的 I/O 次数,由于同时需要对位串解码进行计算,导致 CPU 运算代价很高.由于书法字索引属于高维数据索引范畴,但与传统高维索引技术相比还是存在较大不同:首先,每个字的轮廓采样点个数(维数)非常高(150 维以上),使得传统多维索引技术(如 R-tree[10],VA-file[11]等)不能很好地支持高维书法字的快速检索;其次,由于书法字形的复杂程度不同,使得每个书法字的采样点个数(维数)也不同,而传统高维索引技术只针对维数固定的数据而设计,很难直接应用于书法字索引.而基于距离尺度的索引方法[12,13]是一种有效的方法,因为它无须事先知道每个字的维数,只要采用文献

[14]中介绍的方法求得任意两个字的距离即可.这就是第三类高维索引方法.这类方法是通过将高维数据转化为一维数据来进行高维查询,包括 NB-Tree[12]和 iDistance[13]等.但是,这些方法都是采用单一尺度来过滤查询空间,当维数很高时,其范围查询效率并不理想.

以上高维索引都是针对单机环境,Jagadish[15]等人提出了在 P2P 环境下的多维索引方法——VBI-Tree,但该方法只是针对 P2P 环境而设计的,并不适合网格环境.到目前为止,很少有文献讨论网格环境下的 k-NN 查询,尤其是书法字查询.

2 相关知识

为了支持高效的基于数据网格的高维书法字的 k-NN 查询,本节分别介绍书法字过滤算法和"打包"传输技术,以进一步缩小网络传输的代价,提高查询的并行性.首先,我们简单回顾文献[14]中介绍的基于形状的书法字检索方法.

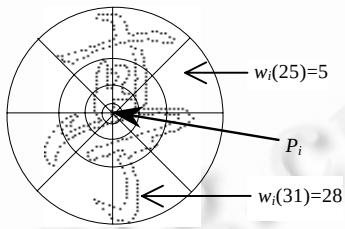
2.1 基于形状的书法字检索

对于给定的书法字数据库$\Omega=\{V_1,V_2,\ldots,V_n\}$,其中,$V_i$表示第 i 个书法字,且 $i\in[1,n]$.由于每个书法字形状复杂程度不同,所提取的轮廓采样点个数也就不一样.设 $V_i=\{p_{i1},p_{i2},\ldots,p_{im}\}$,表示第 i 个字有 m 个采样点,每个采样点 P_{ij} 由$\langle x,y\rangle$两元组坐标构成,且 $j\in[1,m]$.

由于书法字是表形的(如图 1(a)所示),因此,利用采样轮廓点来表达书法字的形状,通过比较字形的相似性来检索.由于原始的书法页面在切分过程中已经过去噪、二值化,因此轮廓提取就较为简单:直接找出黑白邻接点即为轮廓点.一般而言,采用如图 1(b)所示的极坐标比笛卡尔坐标能够更好地描述笔画的方向性.首先,将整个空间从方向上划分出 8 个区域;然后,在弦上按 $\log_2 r$ 划分为 4 份.这样,整个空间就被分为 32 份(32 bin).

(a) An original isolated calligraphic character example

(a) 一个已切分好的书法字



(b) The extraction of feature points

(b) 特征点的提取

Fig.1 The Contour Point context of calligraphic character

图 1 书法字轮廓点的表示

对轮廓上的一个给定点 p_i,其属性用以 p_i 点为中心的坐标系的 32 bin 中落入每个 bin 的像素点个数 $w_i(k)$来描述:$w_i(k)=\#\{q_j\neq p_i:q_j\in bin(k)\}$,$k=0,1,\ldots,31$,其中,$q_j$ 为落入第 k 个 bin 中的不同于 p_i 点的轮廓上的点.图 1(b)中落入第 25 个 bin 的像素个数为 5,即 $w_i(25)=5$.在对样本字轮廓上的某一点 m_i,寻找候选书法字的中对应点 n_j 时,此两点满足如下不等式:

$$dist=ED(m_i,n_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}\le\frac{length}{3} \tag{1}$$

其中 $ED(m_i,n_j)$表示点 m_i 与点 n_j 的欧式距离,$length=32$ 为归一化的像素方阵长度.在该约束下进行两个书法字的形状相似度匹配:对样本字中的每一个点 m_i,在候选字中寻找匹配点 n_j.其中,$C_{ij}=C(m_i,n_j)$表示两点的近似匹配程度大小,值越小则表明越相似.

$$C_{ij}=C(m_i,n_j)=\frac{1}{2}\times\sum_{k=0}^{31}\frac{[w_i(k)-w_j(k)]^2}{w_i(k)+w_j(k)} \tag{2}$$

在两个字完全相同的极端情况下,可以找到精确的对应点 q_j,使得 $C_{ij}=C(m_i,n_j)=0$,即两个点完全匹配;否则,该点的匹配值 PMC_i 按以下公式计算:

$$PMC_i=\min\{C(m_i,n_j):j=0,1,2,\ldots,n\} \tag{3}$$

最后,两个书法字匹配值可以用它们的所有轮廓点匹配值的总和 TMC 来表示:

$$TMC=\sum_{i=1}^{n}(PMC_i+0.1\times ED(p_i,corresp(p_i))) \tag{4}$$

2.2 书法字集过滤

定义 **1**. 数据网格(data grid)由结点(node)和边(edge)构成,表示为 $G=N\times N\times E$,其中 N 表示结点,E 表示结点之间的数据传输带宽.

定义 **2**. 数据网格中的结点分为查询结点 N_q、数据结点 N_d 和执行结点 N_e,表示为 $N=N_q+N_d+N_e$,其中:执行结点 N_e 由 h 个子执行结点 N_e^i 构成,N_e^i 表示第 i 个执行结点,且 $i\in[1,h]$.

如定义 2 所述,在网格环境中,结点在逻辑上分为查询结点 N_q、数据结点 N_d 和执行结点 N_e.查询结点负责提交用户的查询请求;数据结点负责书法字库及其索引的存储;h 个执行结点负责接收来自数据结点的经过过滤后的候选书法字,并行地执行求精(距离)运算,并将结果返回查询结点.由于书法字集存储在数据结点,对于任意一个查询,不需要也没有必要将所有书法字都传输到执行结点进行距离运算.本节提出在数据结点通过混合距离尺度索引快速地对书法字集进行过滤,从而有效地减少候选书法字集的网络传输代价.

2.2.1 预备知识

给定两个书法字 V_i 与 V_j,它们之间的相似距离用 $d(V_i,V_j)$来表示,即 $d(V_i,V_j)=TMC(V_i,V_j)$.超球心 V_q 和半径 r 所构成的超球可以表示为$\Theta(V_q,r)$.

定义 **3**(始点距离). 给定两个维数相同的字 V_i 和 V_o,V_i 的始点距离(start distance,简称 SD)是 V_i 与 V_o 的距离,记为 $SD(V_i)=d(V_i,V_o)$,其中,V_o 中的每个采样点的坐标值都为 0.

由于每个书法字字型的复杂程度不同,所提取的轮廓采样点个数也就不一样,这样得到的始点距离值显然无法比较,需要对其作维数统一化处理.以下给出统一化始点距离(uniform start distance,简称 USD)的概念.

定义 **4**(统一化始点距离). 给定字 V_i,V_i 的统一化始点距离表示为 $USD(V_i)=\dfrac{SD(V_i)}{d_i}\times D$,其中 d_i 为字 V_i 的维数,D 为统一化的维数.

假设 n 个书法字通过层次聚类算法(如 BIRCH[16])得到 T 个类,对于任意一个类 C_j(其中 $j\in[1,T]$),每个类中书法字的个数表示为$\|C_j\|$,且满足 $\sum_{j=1}^{T}\|C_j\|=n$.

定义 **5**(类半径). 对于任意一个类 C_j,该类半径为其质心 O_j 到该类中距离其最远的点的距离,记作 CR_j.

给定任意一个类 C_j 及其类半径 CR_j,对应类超球表示为$\Theta(C_j,CR_j)$.

定义 **6**(质心距离). 给定一个书法字 V_i,它的质心距离(centroid distance,简称 CD)为到其对应类 C_j 的质心 O_j 的距离,表示为 $CD(V_i)=d(V_i,O_j)$,其中,$V_i\in C_j$,且 $j\in[1,T]$.

基于混合距离尺度(hybrid-distance-metric,简称 HDM)的书法字过滤方法是基于以下 3 点提出来的:首先,高维空间中,书法字之间的相似性可以通过该书法字与某个参考书法字的距离来度量和排序;第二,由于距离是一维值,可以使用 B+树来对其建立索引;第三,单一基于质心[13]或统一化始点距离尺度[12]的书法字过滤方法很难有效过滤不相关书法字,而基于质心和统一化始点距离的混合距离可以较为有效地缩小搜索空间,从而进一步降低网络传输的代价.

2.2.2 数据结构

首先通过层次聚类[16]将 n 个书法字聚成 T 类,然后求得每个书法字的统一化始点距离和质心距离,这样,书法字 V_i 可以表示为一个四元组:

$$V_i::=\langle i,cid,USD,CD\rangle \tag{5}$$

其中,i 为书法字 V_i 的编号,cid 表示该书法字所属类的编号.

为了将书法字 V_i 对应的 *USD* 和 *CD* 的信息都包含在该书法字的索引键值中,以便建立一个统一的索引来更有效地过滤无关书法字,可以将其键值表达成如(6)式所示:

$$key(V_i)=c\times\lfloor CD(V_i),\theta\rfloor+\frac{USD(V_i)}{\mathrm{MAX}\_USD} \tag{6}$$

其中,由于 $CD(V_i)$为实数,因此,$\lfloor CD(V_i),\theta\rfloor$表示对 $CD(V_i)$四舍五入取到其小数点后第θ位;c 为一个足够大的常数,用于对$\lfloor CD(V_i),\theta\rfloor$进行线形扩展,得到大于 1 的整数.同时,由于 $USD(V_i)$可能大于 1,需要通过对其除以 MAX_*USD* 进行归一化,使得其值小于 1,其中,MAX_*USD* 也为常数.这样就使得每个书法字对应的 USD 和 CD 的值域不重叠.

2.2.3 索引生成算法

HDM 索引结构如图 2 所示,它由一张哈希表和 T 个分片索引构成,其中 T 为聚类个数.通过聚类,每个类超球中的书法字采用一棵 B+树建立索引,作为 HDM 的一个分片索引.T 个类需要建立 T 棵 B+树,同时需要生成一张哈希表来根据书法字所在类的编号快速地定位到对应的分片索引.一般采用最简单的一一对应的方式来完成哈希映射,即其分片索引的编号由某一书法字所在类的编号来确定.该索引存储于网格中的数据结点.

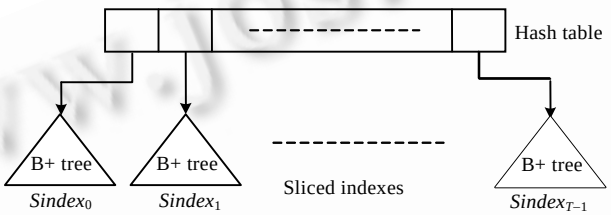


Fig.2 The index architecture of the HDM

图 2 HDM 的索引框架

算法 **1**. HDM 索引创建.

输入:高维书法字集Ω.

输出:高维索引 $bt(1\text{ to }T)$.

1. 采用 BIRCH 聚类算法将高维书法字聚成 T 类;
2. **for** j:=1 to T **do**
3. $bt(j)\leftarrow$***newDMFile***(j);
4. **for** each character in the j-th cluster **do**
5. 计算该类中书法字的 USD 及 CD;
6. $Key(V_i)=c\times\lfloor CD(V_i),\theta\rfloor+USD(V_i)/\mathrm{MAX}\_USD$;
7. 将 $Key(V_i)$插入第 j 棵 B+ tree;
8. **end for**;
9. **return** $bt(j)$
10. **end for**

2.3 “打包”传输方式

当从一个结点向另一个结点传输数据时,采取“打包(package)”传输方式可以极大地提高数据传输效率.其主要思想是:把需要传输的书法字“打”成若干“包”,每个“包”包含若干个书法字,每次把一个包当成一个消息进行传输,而不是把一个书法字当成一个消息进行传输.

(1) 基于“打包”的书法字传输方式,既可以减少每一次数据传输所要消耗的启动传输的代价,又可以减少传输每个消息的头文件所耗费的代价.

(2) 基于“打包”的书法字传输方式具有很好的鲁棒性.如果传输失败,还能够恢复被中断的传输,即能够在

最后一个被传输的包的开始位置恢复传输.

(3) 如果结点间每次传输一个书法字,那么网络上任意的延迟都会使接收数据的结点操作停止执行,而采用“打包”传输方式,执行结点可以把接收到的包中的书法字进行缓存,当下一个“包”出现网络延迟时,就可以对缓存中的书法字进行操作.

3 网格环境下的 *k* 近邻查询算法

问题表述:从查询结点 N_q 发出查询请求,要求对存储于数据结点 N_d 中的高维书法字集 Ω,执行以查询书法字 V_q 的 k 近邻查询,将查询结果传输到 N_q.

网格是以资源共享为基础的协同计算环境,其中的任何资源,包括数据库、CPU、磁盘、设备等都可以被网格中的任一用户使用.假设网格中存在 h 个具有更强的 CPU 处理能力和更快的网络传输速度的结点 N_e^1, N_e^2,…, N_e^h,我们可以将它们作为精确匹配操作的执行结点,并行地完成距离(求精)计算,再把结果传输到发出查询请求的结点 N_q.因此,本文提出了一种适用于数据网格环境的高维书法字 k-NN 查询算法.本质上,k 近邻查询是通过迭代执行范围查询完成的.因此,先从网格环境下的范围查询算法开始研究.假设通过预处理,已经建立了用于书法字集快速过滤的 HDM 索引.当与查询超球 $\Theta(V_q,r)$ 相交的类超球为 t 个($t\le T$)时,利用 HDM 索引得到过滤后的候选书法字 Ω',利用散列函数对这 t 个相交的类超球中的候选高维书法字集 Ω' 进行散列,得到 h 个桶,其中,$h\le t$.然后采用流水线并行机制,把过滤后的书法字以“打包”方式传输到执行结点 N_e^1, N_e^2,…, N_e^h,在这些结点并行执行求精(距离)运算,最后把运算得到的结果书法字(记为 Ω'')传输到查询结点 N_q.

该算法可以分为 3 个阶段,如图 3 所示.

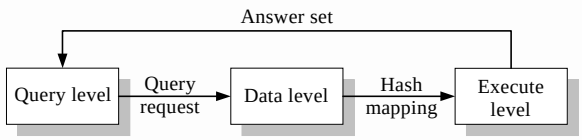


Fig.3 The work flow of grid-based character *k*-NN search

图 3 基于网格的书法字 *k*-NN 查询执行流程

(1) 书法字过滤.在该阶段,首先在查询结点 N_q 将用户的查询请求(即查询书法字 V_q 及半径 r 的查询超球)发送到数据结点 N_d,然后在该结点判断查询超球 $\Theta(V_q,r)$ 与 T 个类超球是否相交,进而利用 HDM 索引对不相关的书法字进行快速排除(过滤),从而有效减少将候选书法字集从数据结点发送至执行结点的网络传输代价.在数据结点 N_d 为书法字集 Ω 设置一个输入缓冲区 IB_1,再设置一个输出缓冲区 OB_1,用于缓存产生的候选书法字集 Ω',当 OB_1 中候选书法字集的大小等于一个传输包的大小时,就以“打包”的方式把候选书法字传输到对应的执行结点 N_e.

算法 **2**. 书法字过滤(character filter).

输入:书法字集 Ω 及查询超球 $\Theta(V_q,r)$.

输出:被过滤后的候选书法字集 Ω'(1 to t).

1. **for** i:=1 to T **do**
2. **if** $\Theta(O_i,CR_i)$ intersects $\Theta(V_q,r)$ **then** /* 表示两个超球相交 */
3. $\Omega'(i)\leftarrow$***Search***(V_q,r,i);
4. 将得到的 $\Omega'(i)$ 输出到输出缓冲区 OB_1;
5. **else if** $\Theta(O_i,CR_i)$ contains $\Theta(V_q,r)$ **then** /* 表示类超球包含查询超球 */
6. $\Omega'(i)\leftarrow$***Search***(V_q,r,i);
7. 将得到的 $\Omega'(i)$ 输出到输出缓冲区 OB_1;
8. **end loop**;

9. **end for**

Search(V_q,r,i) /* 在数据结点层面 */

10. $left\leftarrow c\times\lfloor d(V_q,O_i)-r,\theta\rfloor+[USD(V_q)-r]/\mathrm{MAX\_}USD$;

11. $right\leftarrow c\times\lfloor CR_i,\theta\rfloor+[USD(V_q)+r]/\mathrm{MAX\_}USD$;

12. $S\leftarrow$***BRSearch***$[left,right,i]$;

13. **return** S;

在算法 2 中,函数 ***Search***(V_q,r,i)具体执行并且返回范围查询在第 i 个分片索引上得到的候选书法字集$\Omega'(i)$.需要说明的是,当查询超球与第 i 个类超球相交时,其对应的 USD 的查询范围为$[USD(V_q)-r,USD(V_q)+r]$,而 CD 的查询范围为$[\lfloor d(V_q,O_i)-r,\theta\rfloor,\lfloor CR_i,\theta\rfloor]$,这样,将两者结合就得到总的查询范围,如第 10 行~第 11 行所示;***BRSearch***$(left,right,i)$用于对第 i 个分片索引进行标准的 B+树范围查询.

(2) 散列阶段.经过第 1 阶段的过滤,得到候选书法字集Ω'.同时,由于Ω'是由 t 个与$\Theta(V_q,r)$相交的类超球中经过过滤后的书法字组成,令$\Omega'(j)$表示与查询超球相交的第 j 个类超球所对应的候选书法字集,且$\sum_{j=1}^{t}\Omega'(j)=\Omega'$,其中,$j\in[1,t]$.在开始散列操作之前,首先通过网格资源发现机制,得到 h 个与数据结点连接速率最好的空闲执行结点,且 $h\le t$,然后,分别将候选书法字集$\Omega'(j)$通过哈希映射并以“打包”方式发送至对应的执行结点 N_e^i 中,$i\in[1,h]$.例如,将$\Omega'(1)$传输到执行结点 N_e^1,并在该结点进行距离计算.同时,对子书法字集$\Omega(2)$进行过滤处理,得到过滤结果$\Omega'(2)$,…,再把$\Omega(h)$的过滤结果$\Omega'(h)$传输到结点 N_e^h,并且在结点 N_e^h 进行距离计算的同时,对书法字集的第$(h+1)$个桶中的子书法字集$\Omega(h+1)$先进行过滤,得到过滤结果$\Omega'(h+1)$,这时就要查看 N_e^i 到 N_e^h 中是否存在一个结点处于“闲置”状态:假设结点 N_e^i 处于“闲置”状态,就把$\Omega'(h+1)$传输到结点 N_e^i,并且进行距离计算操作;如果不存在,就要等待 N_e^i 到 N_e^h 中的某一个结点可用为止.

(3) 求精阶段.将散列到各执行结点的候选书法字分别并行地计算与查询书法字 V_q 的距离,将距离值小于或等于 r 的书法字返回给查询发送结点 N_q,完成范围查询操作.具体步骤为:在执行结点 N_e^i,为候选子书法字集$\Omega'(j)$设置一个输入缓冲区 $IB(j)$,$M(j)$为候选子书法字集$\Omega'(j)$在执行结点 N_e^i 分配的内存空间,用于存储接收到的$\Omega'(j)$中的书法字;再设置一个输出缓冲区 $OB(j)$,用于缓存产生的结果书法字集.当 $OB(j)$中结果书法字集的大小等于一个传输包的大小时,就以“包”的方式把结果书法字传输到发出查询请求的结点 N_q.

算法 **3**. 求精过滤(refinement).

输入:发送到某个执行结点的候选书法字$\Omega'(j)$,查询书法字 V_q 及半径 r.

输出:返回结果书法字$\Omega''(j)$.

1. $\Omega''(j)\leftarrow\varnothing$;
2. **if** $d(V_i,V_q)\le r$ and $V_i\in\Omega'(j)$ **then**
3. 将得到的结果书法字 V_i 输出到输出缓冲区 $OB(j)$;
4. **end if**

与范围查询不同的是:开始是用一个较小的半径去进行范围查询(第 1 行),当得到的候选字个数小于 k 时(第 3 行),再重新增大查询半径(第 4 行).由于通过上述方法得到的候选书法字个数不一定正好为 k 个,可能会大于 k(第 10 行),当遇到该情况时,需要进行$(\|\Omega''\|-k-1)$次循环(第 11 行),依次找到在该结果书法字集Ω''中距离查询字 V_q 最远的$(\|\Omega''\|-k-1)$个字(第 12 行),并将它们删除(第 13 行).这样,恰好得到 k 个最近邻书法字,其中:函数 ***CharacterFilter***(V_q,r)为对Ω进行以 V_q 为中心,r 为半径的书法字过滤,如算法 2 所示;函数 ***Refinement***(Ω',V_q,r)为对Ω'进行以 V_q 为中心,r 为半径的书法字求精,如算法 3 所示;函数 ***Farthest***(S,V_q)用于返回候选书法字集 S 中与 V_q 距离最远的书法字.下面是整个 k-NN 查询的完整算法.

算法 **4**. $kNNSearch(V_q,k)$.

输入:查询字 V_q,k.

输出:查询结果 S.

```
1.  r←0,Ω′←∅,Ω″←∅;                          /* 初始化 */
2.  发送查询请求到数据结点 N_d;
3.  while (||Ω″||<k)                        /* 当从执行结点返回的结果书法字个数小于 k 时,继续循环 */
4.      r←r+Δr;                             /* Δr 很小,用于逐步增加半径值 */
5.      利用资源管理机制在网格中找到 h 个性能较好的结点作为求精操作的执行结点;
6.      Ω′←CharacterFilter(V_q,r);          /* 在数据结点完成书法字过滤 */
7.      把Ω′中的候选书法字按照“打包”方式传输到 h 个执行结点;
8.      Ω″←Refinement(Ω′,V_q,r);            /* 在执行结点完成求精过滤 */
9.      将Ω″中的结果书法字按照“打包”方式发送到查询结点 N_q;     /* 第 8 步和第 9 步并行执行 */
10.     if (||Ω″||>k) then                  /* 当返回结果书法字Ω″个数大于 k 时 */
11.         for i:=1 to ||Ω″||−k−1 do
12.             V_far←Farthest(Ω″,V_q);
13.             Ω″←Ω″−V_far;                /* 将 V_far 从结果书法字集Ω″中删除 */
14.         end for
15.     end if
16. end while
```

其中,第 6 步和第 7 步并行执行,因为过滤得到的候选书法字在发送至执行结点之前是先缓存于数据结点中的,当缓存中的书法字个数达到传输包大小时,再将它们“打包”发送到对应的执行结点.同理,第 8 步和第 9 步也是并行执行的.

4 代价模型

由于 k 近邻查询是通过迭代调用范围查询来完成的,为简单起见,本节给出基于数据网格的范围查询的代价模型.假设 n 为书法字的总数.f 表示 B+树中每个节点的平均出度;T_S 为磁盘寻道时间;T_L 为延迟时间;T_T 为数据传输时间;T_C 为 CPU 进行一次判断所需的时间;T_{Query} 为范围查询时间;T_{Total} 表示总查询时间.

在数据结点,当查询超球$\Theta(V_q,r)$与 t 个类超球相交($t\le T$)时,如图 4 所示,第 i 个类超球中的书法字个数可近似表示为

$$NUM(i)=\frac{Vol(\Theta(O_i,CR_i))}{\sum_{j=1}^{T}Vol(\Theta(O_i,CR_i))}\times n.$$

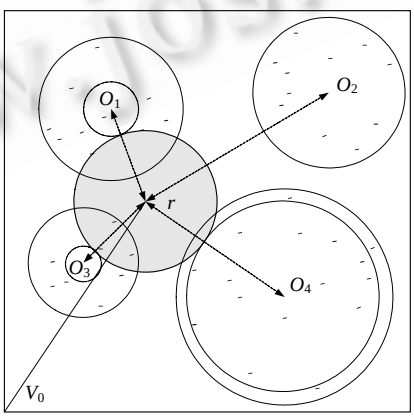


Fig.4 An example of calligraphic character filtering

图 4 书法字过滤的例子

同时,由于每个分片索引对应一棵 B+树,因此,第 i 棵 B+树(分片索引)的高度 h_i、每个节点的平均出度和元素个数 $NUM(i)$近似满足式(8).

$$f\times(f+1)^{h_i-1}=NUM(i) \tag{8}$$

求解式(8),得到该树的高度为

$$h_i=\left\lceil\frac{\lg NUM(i)-\lg f}{\lg(f+1)}\right\rceil+1 \tag{9}$$

在数据结点上,对于第 i 个分片索引上的范围查询,整个查询分为两部分:首先是从根节点到叶节点,共访问 h_i 个节点;其次为在叶节点上的范围查询,该范围查询对应到第 i 个分片索引中,需要访问的书法字总数为

$$num(i)=\frac{Vol\left(\left(\overline{\Theta(V_o,USD(V_q)-r)}\cap\Theta(V_o,USD(V_q)+r)\right)\cap\overline{\Theta\left(O_i,\left\lfloor(d(V_q,O_i)-r),\theta\right\rfloor\right)}\cap\Theta\left(O_i,\left\lfloor CR_i,\theta\right\rfloor\right)\right)}{\sum_{j=1}^{T}Vol(\Theta(O_j,CR_j))}\times n \tag{10}$$

由于总共需要进行 t 次范围查询,因此,其总查询代价为

$$T_{Query}=\sum_{i=1}^{t}\left[\left(\left\lceil\frac{\lg NUM(i)-\lg f}{\lg(f+1)}\right\rceil+1+\left\lceil\frac{num(i)}{f}\right\rceil\right)\times(T_S+T_L+T_T)\right] \tag{11}$$

通过对书法字库(Ω)的过滤得到候选书法字集(Ω')之后需要对其进行散列,以便发送到执行结点进行距离运算.散列操作的过程就是决定将这 t 组候选书法字发送到哪些执行结点,其 I/O 代价和判断所需的 CPU 代价可以忽略.

定义 7. 从结点 A 向结点 B 传输一条消息的代价定义为 $T_{AB}(X)=C_0+C_{AB}\times X$,其中,$X$ 表示结点 A 和结点 B 之间的数据传输量;C_0 表示两结点间通信初始化一次所花费的时间,近似于一个常数,通常包括为消息的传递所做的准备、通知目标结点它将会收到消息、处理目标结点的答复等等.通常情况下,网络的传输带宽是随时间变化的,可以使用网络传输带宽的统计平均值.假设结点 A 和 B 之间的网络传输率为一个常数 C_{AB},表示单位数据传输所用的时间.

根据定义 7 可以得到,将候选书法字集(Ω')从数据结点传输至执行结点所需时间为

$$T_{Trans}=\frac{|\Omega'|}{|P|}\times(C_o+C_{DE}\times|P|) \tag{12}$$

其中,$|P|$表示每个包的大小,C_{DE} 表示数据结点 N_d 到执行结点 N_e 之间的网络传输率.

又因为对候选书法字集(Ω')的求精运算是在若干个执行结点并行计算的,因此其 CPU 代价 T_{CPU} 可表示为

$$T_{CPU}=\max\{T_{CPU}(1),T_{CPU}(2),\ldots,T_{CPU}(t)\} \tag{13}$$

其中,$T_{CPU}(i)$表示在第 i 个执行结点上的距离运算代价,且满足式(14).

$$T_{CPU}(i)=\frac{Vol\left(\left(\overline{\Theta(V_o,USD(V_q)-r)}\cap\Theta(V_o,USD(V_q)-r)\right)\cap\overline{\Theta\left(O_i,\left\lfloor(d(V_q,O_i)-r),\theta\right\rfloor\right)}\cap\Theta\left(O_i,\left\lfloor CR_i,\theta\right\rfloor\right)\right)}{\sum_{j=1}^{T}Vol(\Theta(O_j,CR_j))}\times n\times T_C \tag{14}$$

结果书法字(Ω'')从执行结点发送回查询结点所需时间可表示为

$$T'_{Trans}=\frac{|\Omega''|}{|P|}\times(C_o+C_{EQ}\times|P|) \tag{15}$$

其中:$|\Omega''|$表示返回查询结点的结果书法字个数,且 $|\Omega''|=\dfrac{Vol(\Theta(V_q,r))}{\sum_{j=1}^{T}Vol(\Theta(O_j,CR_j))}\times n$;$C_{EQ}$ 表示从执行结点 N_e 到发送结点 N_q 之间的网络传输率.

整个基于数据网格的范围查询的代价可以表示为

$$C_{total}=T_{Query}+T_{Trans}+T_{CPU}+T'_{Trans} \tag{16}$$

合并式(9)~式(16)得到式(17),可以看出,该查询代价正比于书法字总数,且反比于索引平均出度.

$$T_{Total}=\sum_{i=1}^{t}\left[\left(\left\lceil\frac{\lg NUM(i)-\lg f}{\lg(f+1)}\right\rceil+1+\left\lceil\frac{num(i)}{f}\right\rceil\right)\times(T_S+T_L+T_T)\right]+t\times T_c+\frac{|\Omega'|}{|P|}\times(C_o+C_{DE}\times|P|)+$$

$$\max\{T_{CPU}(1),T_{CPU}(2),\ldots,T_{CPU}(t)\}+\frac{|\Omega''|}{|P|}\times(C_o+C_{EQ}\times|P|) \quad (17)$$

5 实验结果与分析

针对影响k-NN查询操作算法的各种参数,本文主要进行了3组模拟实验.实验结果表明,该算法在减少网络通信开销、增加I/O和CPU并行、降低响应时间方面具有较好的性能.我们用C语言实现了基于混合距离尺度的书法字过滤算法并将其部署在数据结点.该算法采用B+树作为单维索引结构且索引页大小设为4 096字节,所有实验的模拟运行环境为局域网.本文测试所用的书法字库[17]来自中美百万册数字图书馆项目,它包含了从书法库中提取了12 000个预先切分好的书法字的轮廓点形状特征,每个特征点为一个二元组,包括x和y的坐标值.

5.1 基于数据网格的书法字检索

我们实现了一个基于数据网格的书法字检索系统,如图5所示.从切分好的书法作品[17]中取12 000个单字做测试,以其中一个"天"字为样本进行检索,前21个返回图像中有11个是正确的.查全率和查准率[14]是图像检索领域中通用的衡量检索好坏的尺度.图6给出了对30个样本检索后的平均查全率和查准率曲线图.可以看出,本文提出的"形状相似性方法"的效果要优于基于"推土机原理"[18]以及"投影"[19]方式.

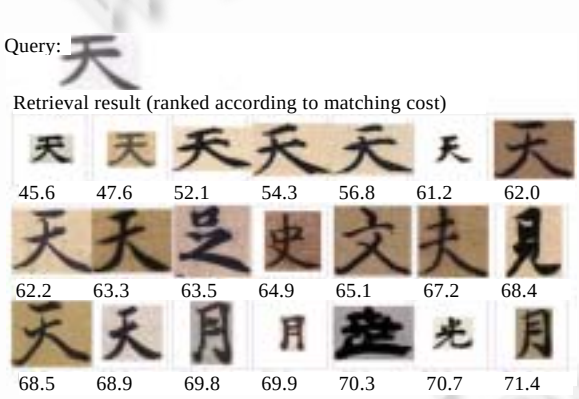


Fig.5 An retrieval example based on shape-similarity

图5 采用形状相似性方法的一个检索例子

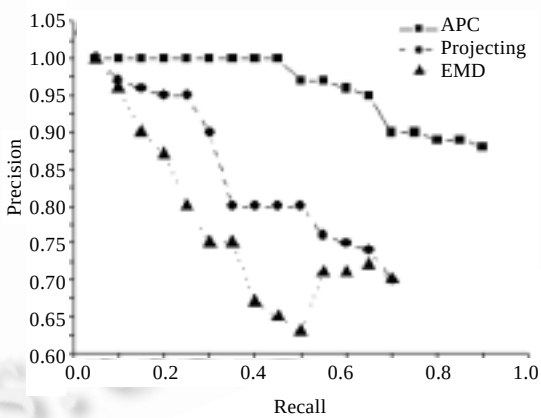


Fig.6 Recall vs. precision

图6 查全率与查准率比较

5.2 传输速率对查询性能的影响

第1组实验研究网络传输速率对k-NN查询性能的影响.假设查询结点N_q与数据结点N_d的速率为d_1,数据结点N_d与执行结点N_e之间的网络传输速度为d_2,执行结点N_e与查询结点N_q之间的网络传输速度为d_3.实验中采用有12 000个书法字的数据库,用户从查询结点发送查询请求到数据结点的时间(T_1)远远小于将候选书法字从数据结点传输到执行结点的时间(T_2)和将结果书法字从执行结点传输到查询结点的时间(T_3). 图7和图8中的δ和σ分别表示候选书法字传输时间(T_2)和书法字传输时间(T_3)占总响应时间的百分比与传输速率之间的关系.可以看出,在书法字总数一定的情况下,随着网络连接速度的增加,候选书法字及结果书法字传输时间占总响应时间的百分比在逐步减少,但候选书法字传输时间比结果书法字传输时间要多,这是因为结果书法字是在候选书法字的基础上,通过在执行结点的求精(距离)运算而得到的,其数据量较之候选书法字大为减少.

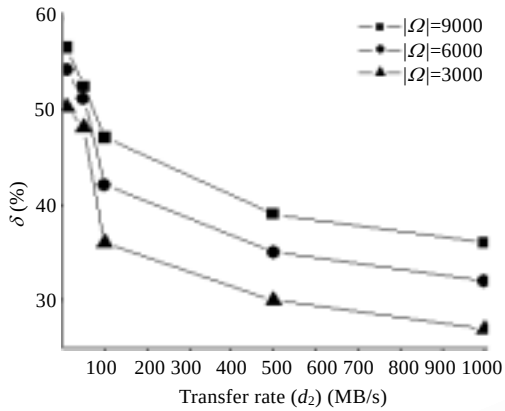


Fig.7 δ vs. transfer rate

图 7 δ与传输速率

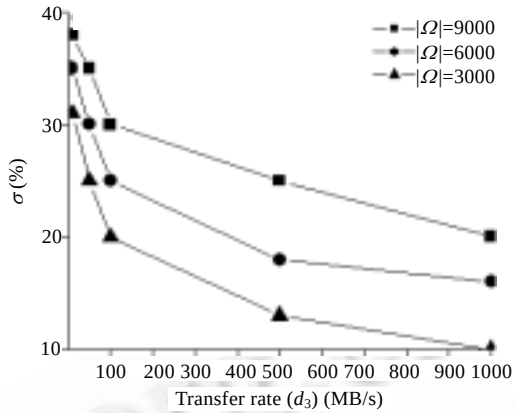


Fig.8 σ vs. transfer rate

图 8 σ与传输速率

5.3 书法字过滤对查询性能的影响

本次实验研究书法字过滤对 k-NN 查询性能的影响.方法 1 不进行书法字过滤:把书法字集Ω直接从数据结点 N_d 传输到执行结点 N_e 进行距离计算.方法 2 进行书法字过滤:利用书法字过滤算法把书法字集Ω过滤为Ω'后,再把Ω'传输到执行结点 N_e 进行距离计算.从图 9 可以看出,当 k 一定的情况下,经过过滤后的总查询响应时间明显要优于未经过过滤的查询响应时间,且随着 k 的增加,两者的性能差别越来越大.这是由于过滤后的书法字可以明显减少网络传输及距离计算的代价,同时基于索引的书法字过滤所需的时间远远小于网络传输的代价,可以忽略不计.

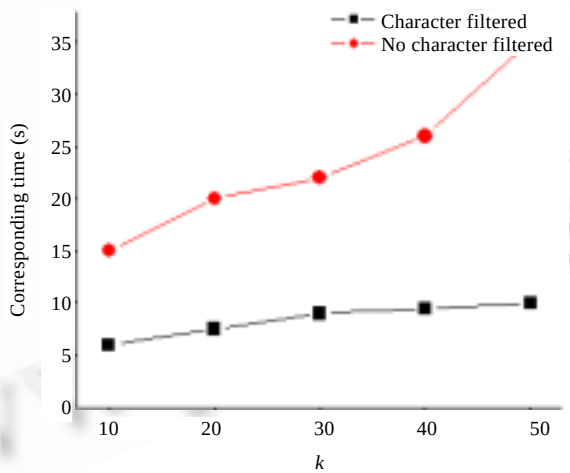


Fig.9 Effect of character filtering on the query performance

图 9 书法字过滤对查询性能的影响

5.4 数据量、k和T对性能加速比的影响

我们分别对 k-NN 查询中的 k 及执行结点个数(h)对性能加速比做一个评估.加速比表示为在单机上执行 k-NN 查询所需的时间与在网格环境执行相同 k-NN 查询的时间的比值.从图 10 可以看出,随着数据量的增加,其查询加速比也随之提高.当数据量为 4 200 时,其加速比大于 1.这是因为对于数据量不大的基于网格的 k-NN 检索,其网络传输代价往往大于执行结点上的 CPU 距离计算代价,同时,由于书法字的相似匹配需要较高的 CPU 代价,因此当数据量达到一定程度时,网格的并行计算能力极大地加速了书法字的匹配速度.从图中还可以

看出,并非数据量越大加速比就越高.当书法字个数达到 10 000 时,加速比的增长率减缓,这是因为大数据量所带来的网络传输代价会抵消并行计算所带来的系统性能的提升.图 11 显示了 k 对其加速比的影响.假定在书法字个数和执行结点个数一定的条件下,当 k 从 10 增加到 50 时,其对查询加速比的影响不是很大.从图 12 可以看出,当执行结点个数(h)增加时,k-NN 查询的性能加速比有所提高,但提高的幅度较为缓慢.这是因为在执行结点个数增加的同时也增加了从数据结点发送到执行结点的代价,从而会抵消一部分性能的提高.

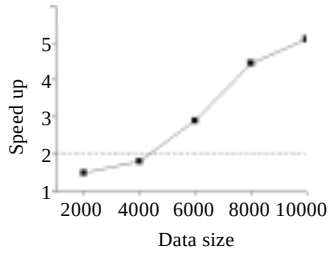


Fig.10 Data size vs. speedup

图 10 数据量对加速比的影响

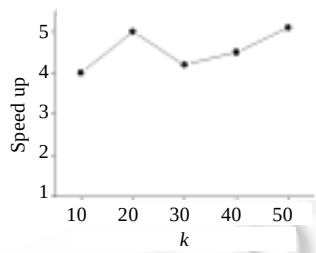


Fig.11 k vs. speedup

图 11 k 对加速比的影响

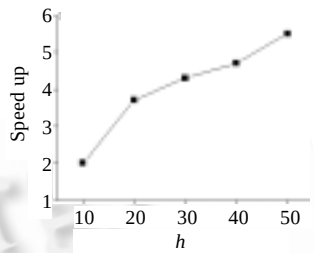


Fig.12 h vs. speedup

图 12 h 对加速比的影响

6 总结与将来工作

本文提出了一种基于数据网格的书法字 k-NN 查询算法.该算法把网格中性能较好的 h 个结点作为查询执行的结点.同时,采用基于混合距离尺度的书法字过滤、“打包”传输及流水线并行机制来减少查询的响应时间.理论和实验结果表明,该 k-NN 算法在最小化网络通信开销和最大化 I/O 和 CPU 并行方面具有很好的性能,但是在网格环境下,该查询仍有大量尚未解决的问题,还有很多研究工作要做.具体工作包括以下几个方面:1) 在网格环境下,如果参与查询操作的高维书法字存在多个副本或者存在多个数据结点,如何将数据存储于这些结点从而增加操作的并行性;2) 查询结果的缓存问题,即当一个数据库产生查询结果时,采用什么样的方式把结果进行缓存,使得以后的查询不用访问数据库就可以从缓存中得到所需要的查询结果.

References:

[1] Palmondon R, Srihari SN. On-Line and off-line handwriting recognition: A comprehensive survey. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2000,22(1):63−84.

[2] Rath TM, Kane S, Lehman A, Partridge E, Manmatha R. Indexing for a digital library of George Washington’s manuscripts: A study of word matching techniques. Technical Report, MM-36, Boston: University of Massachusetts, 2002.

[3] Yosef IB, Kedem K, Dinstein I, Beit-Arie M, Engel E. Classification of hebrew calligraphic handwriting styles: Preliminary results. In: Proc. of the 1st Int’l Workshop on Document Image Analysis for Libraries. Palo Alto, 2004. 299−305.

[4] Shi BL, Zhang L, Wang Y, Chen ZF, Content-Based Chinese antique books retrieval through visual similarity criteria. Journal of Software, 2001,12(9):1336−1342 (in Chinese with English abstract).

[5] Segal B. Grid computing: The European data grid project. In: Proc. of the 2000 IEEE Nuclear Science Symp. and Medical Imaging Conf. Lyon, 2000.

[6] Hoschek W, Jaen-Martinez J, Samar A, Stockinger H, Stockinger K. Data management in an international data grid project. In: Proc. of the 1st IEEE/ACM Int’l Workshop on Grid Computing. Berlin: Springer-Verlag, 2001. 17−20.

[7] Smith J, Gounaris A, Watson P, Paton NW, Fernandes AAA, Sakellariou R. Distributed query processing on the grid. In: Proc. of the 3rd Int’l Workshop on Grid Computing. Berlin: Springer-Verlag, 2002. 279−290.

[8] Yang DH, Li JZ, Zhang WP. Grid-Based join operation. Journal of Computer Research and Development, 2004,41(10):1848−1855 (in Chinese with English abstract).

[9] Böhm C, Berchtold S, Keim DA. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Computing Surveys, 2001,33(3):322−373.

[10] Guttman A. R-Tree: A dynamic index structure for spatial searching. In: Yormark B, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'84). Boston: ACM Press, 1984. 47−54.

[11] Weber R, Schek HJ, Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases (VLDB'98). New York: Morgan Kaufmann Publishers, 1998. 194−205.

[12] Fonseca MJ, Jorge JA. NB-Tree: An indexing structure for content-based retrieval in large databases. In: Proc. of the 8th Int'l Conf. on Database Systems for Advanced Applications. Kyoto: IEEE Computer Society, 2003. 267−274.

[13] Jagadish HV, Ooi BC, Tan KL, Yu C, Zhang R. iDistance: An adaptive B^+-tree based indexing method for nearest neighbor search. ACM Trans. on Data Base Systems, 2005,30(2):364−397.

[14] Zhuang YT, Zhang XF, Wu JQ, Lu XQ. Retrieval of Chinese calligraphic character image. In: Aizawa K, Nakamura Y, Satoh S, eds. Proc. of the Pacific Rim Conf. on Multimedia (PCM 2004). Berlin, Heidelberg: Springer-Verlag, 2004. 63−84.

[15] Jagadish HV, Ooi BC, Vu QH, Zhang R, Zhou AY. VBI-Tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes. In: Proc. of the 22nd IEEE Int'l Conf. on Data Engineering (ICDE 2004). New York: IEEE Computer Society Press, 2004.

[16] Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases. In: Jagadish HV, Mumick IS, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'96). New York: ACM Press, 1996. 103−114.

[17] The cadal project. 2006. http://www.cadal.zju.edu.cn

[18] Cohen S, Guibas L. The earth mover's distance under transformation sets. In: Proc. of the Int'l Conf. on Computer Vision (ICCV'99). New York: IEEE Computer Society Press, 1999. 173−187.

[19] Wu YS, Ding XQ. The Recognition of Chinese Character: Principle, Approach and Implementation. Beijing: Higher Education Press, 1992 (in Chinese).

附中文参考文献:

[4] 施伯乐,张亮,王勇,陈智锋.基于视觉相似性的计算机古籍内容检索方法.软件学报,2001,12(9):1336−1342.

[8] 杨东华,李建中,张文平.基于数据网格环境的连接操作算法.计算机研究与发展,2004,41(10):1848−1855.

[19] 吴佑寿,丁晓青.汉字识别——原理、方法与实现.北京:高等教育出版社,1992.

庄毅(1978－),男,浙江杭州人,博士生,主要研究领域为高维数据查询,网格计算和多媒体检索.

吴飞(1973－),男,博士,副教授,CCF 高级会员,主要研究领域为多媒体检索,机器学习.

庄越挺(1965－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为多媒体检索,数字图书馆,视频动画.