ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.1, January 2010, pp.55−67
doi: 10.3724/SP.J.1001.2010.03665

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

网格环境下基于流水线的多重相似查询优化*

胡 华[1,2], 庄 毅[2+], 胡海洋[1,2], 赵格华[3]

[1](杭州电子科技大学 计算机学院,浙江 杭州 310018)

[2](浙江工商大学 计算机与信息工程学院,浙江 杭州 310018)

[3](香港中文大学 计算机科学与工程系,香港)

Pipeline-Based Multi-Query Optimization for Similarity Queries in Grid Environment

HU Hua[1,2], ZHUANG Yi[2+], HU Hai-Yang[1,2], Dickson CHIU[3]

[1](College of Computer Science, Hangzhou Dianzi University, Hangzhou 310018, China)

[2](College of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou 310018, China)

[3](Department of Computer Science and Engineering, Chinese University of Hong Kong, China)

+ Corresponding author: E-mail: zhuang@mail.zjgsu.edu.cn

Hu H, Zhuang Y, Hu HY, Chiu D. Pipeline-Based multi-query optimization for similarity queries in grid environment *Journal of Software*, 2010,21(1):55−67. http://www.jos.org.cn/1000-9825/3665.htm

Abstract: This paper proposes a multi-query optimization algorithm for pipeline-based distributed similarity query processing (pGMSQ) in grid environment. First, when a number of query requests are simultaneously submitted by users, a cost-based dynamic query clustering (DQC) is invoked to quickly and effectively identify the correlation among the query spheres (requests). Then, index-support vector set reduction is performed at data node level in parallel. Finally, refinement of the candidate vectors is conducted to get the answer set at the execution node level. By adopting pipeline-based technique, this algorithm is experimentally proved to be efficient and effective in minimizing the response time by decreasing network transfer cost and increasing the throughput.

Key words: grid; multi-query optimization; high-dimensional indexing; data partition

摘 要: 提出一种网格环境下基于流水线技术的分布式多重相似查询的优化算法(pipeline-based distributed similarity query processing,简称 pGMSQ).首先,当用户提交若干个查询请求时,采用基于代价的动态层次聚类策略(dynamic query clustering,简称DQC)对其进行合并.然后在数据结点层,采用索引支持的向量集缩减方法快速过滤无关向量.最后,在执行结点层对候选向量执行求精操作返回结果向量.由于本查询采用了流水线技术,实验结果表明,该方法在提高查询性能的同时也提高了系统的吞吐量.

* Supported by the National Natural Science Foundation of China under Grant Nos.60873022, 60903053 (国家自然科学基金); the Zhejiang Provincal Natural Science Foundation of China under Grant Nos.Y1080148, Y1090165 (浙江省自然科学基金); the Key Program of Science and Technology of Zhejiang Province of China under Grant No.2008C13082 (浙江省科技厅重大科技项目); the Key Project of Special Foundation for Young Scholars in Zhejiang Gongshang University of China under Grant No.Q09-7 (浙江工商大学青年人才基金重点资助项目); the Open Fund Provided by State Key Laboratory for Novel Software Technology of Nanjing University of China (南京大学计算机软件新技术国家重点实验室开放基金)

Received 2008-06-05; Revised 2008-11-27; Accepted 2009-02-24

关键词: 网格;多重查询优化;高维索引;数据分片
中图法分类号: TP311 文献标识码: A

Internet和多媒体技术的飞速发展使得查询密集(并发用户查询)条件下基于内容的海量多媒体信息检索[1]的研究成为一个热点问题.为了有效地提高其查询性能,需要应用高维索引技术.然而传统的高维索引方法都是从数据本身来设计索引结构[2−6].较少有研究从对用户查询请求的优化来提高查询性能.如图 1 所示,假设用户提交 3 个查询请求(query request,简称QR),即Q_1,Q_2和Q_3,可以看出,Q_1和Q_2存在一些相关性(相交),这意味着它们具有公共的查询区域(如图 1 中阴影部分所示).通过合并Q_1和Q_2使得原来的 3 次查询能够以批量的方式快速完成(即 2 次),提高其总体查询性能.这种对独立的查询序列发现它们之间的相关性并且用于查询的优化称为多重查询优化(multi-query optimization)[1],即从用户查询的角度来提高查询性能.

网格环境为用户提供了一个统一的并行分布式计算平台[7].为了充分利用网格强大的并行计算能力,突出数据网格资源共享的特点,提出一种面向网格环境的基于流水线技术的分布式相似查询的多重优化方法——pGMSQ,以显著提高分布式相似查询效率及吞吐量.与传统分布式系统不同,网格中的结点呈现异构(处理能力不同等)特点且需要通过网格资源管理系统(grid resource management system,简称GRMS)动态来发现.pGMSQ面临的技术挑战表现在以下两个方面:1) 快速、有效地发现不同查询请求的相关性:对于若干查询请求,如何快速、有效地对其进行聚类不是一个简单的问题.2) 有效的负载平衡:对于绝大多数的并行数据库系统,设计一种高效的负载均衡方法对于提高查询的并行性非常重要,特别是对于异构的网格环境.

为了应对上述挑战,pGMSQ算法包括 3 种支撑技术,包括动态查询聚类策略,基于中心环(centroid ring)的负载均衡机制和索引支持的向量集缩减.图 2 为网格环境下的多重相似查询的框架.假设用户提交一批查询(即,m个查询向量及其半径)到查询结点N_q,pGMSQ算法的基本思想如下:首先,利用基于代价的动态查询聚类算法对提交的查询在查询结点层进行快速合并.然后将新得到的查询类发送至数据结点层,进行基于iDistance[8]索引的不相关高维向量的快速过滤处理.之后,再将得到的候选向量发送至执行结点层并行地进行求精(距离计算)操作得到结果向量,其中该层的结点是通过GRMS动态发现的.最后将结果向量返回查询结点N_q,这些方法的目的是减少网络传输的代价以及提高查询的并行性.不失一般性,我们将欧式距离作为本文的相似距离度量.

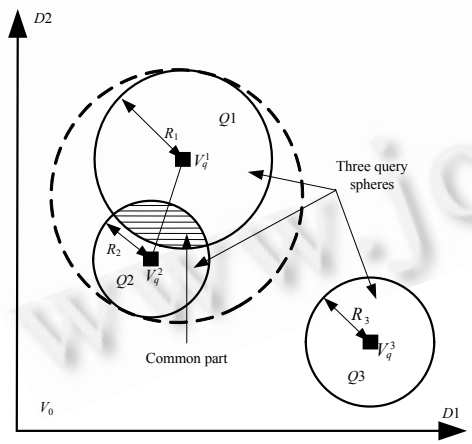


Fig.1 Intersection part of query spheres
图 1 查询超球相交部分

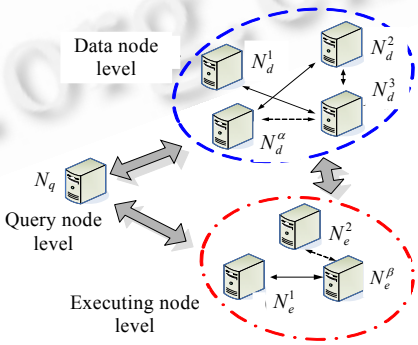


Fig.2 Topology of grid
图 2 网格的拓扑结构

本文第 1 节介绍相关工作.第 2 节为预备工作.第 3 节给出 3 种支持 pGMSQ 的支撑技术,包括基于代价的动态查询层次聚类算法、自适应负载均衡及索引支持的向量集缩减.第 4 节提出网格环境下的基于流水线技术的多重相似查询优化算法.第 5 节通过实验验证算法的有效性.第 6 节总结本文.

1 相关工作

本文工作与多维(高维)索引相关.目前高维索引分成两类:集中式高维索引[2]和分布式高维索引[5,9,10].集中式多(或高)维索引在近 20 年来一直受到学术界广泛的研究[2].在这些索引方法中,数据空间被划分成多个小的区域,其中数据子空间或重叠[3]或独立[4].近年来,相继提出了一些新的索引方法,如基于向量压缩的索引(如VA-file[11])及基于距离的高维索引(如iDistance[8]).其中,在iDistance中,首先对高维数据进行聚类,再计算每个对象与其质心的距离并用B+树索引.从而将高维查询转换为一维范围查找.以上索引都是针对集中式查询设计的.Berchtold等人[9]提出了一种基于近似的最优数据项分布的、快速的并行相似检索.之后,文献[10]提出一种基于磁盘阵列的并行相似检索.最近,基于Peer-to-Peer(P2P)的相似检索越来越受到关注.CAN[5]是第一个支持多维数据检索的P2P系统.文献[12]采用空间填充曲线将多维数据映射到一维空间.pSearch[13]是一个基于CAN的文档检索的P2P系统.另一个由Sahin等人[14]开发的系统也是基于哈希函数的CAN系统,然而,精确的查询在该系统中是低效的.SkipIndex[15]和文献[16]提出的查询算法支持在P2P网络的相似查询.上述分布式索引都集中在数据本身,很少考虑到从查询请求的角度来提高查询效率.

在数据网格研究领域,美国和欧洲等国已经进行了广泛而深入的研究,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[17]、美国的国际虚拟数据网格实验室IVDGL(International Virtual Data Grid Laboratory)项目[18]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.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[7,19],但是还很少有文献研究基于网格的多重相似查询优化.与传统分布式计算环境不同[20],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,这些都为基于网格环境的多重相似查询提出了新的要求.

多重查询优化在传统数据库查询中已进行了广泛的研究[1].一般来说,数据库管理系统(DBMS)经常会执行一组相关的查询请求,这些相关查询请求包含一些共同的查询部分(子查询表达式).多种查询优化通过发现查询计划中的公共部分来进行,包括消除共同的子查询表达式、重新排序查询计划以及使用物化视图等.

2 预备工作

本文采用符号见表 1.

Table 1 Meaning of symbols used

表 1 符号表

Symbol	Meaning	Symbol	Meaning
Ω	Vector set	$d(V_i,V_j)$	Similarity distanc
V_i	The i-th vector and $V_i\in\Omega$	Ω'	Candidate vector set
D	Dimensionality	Ω''	Answer vector set
N	The number of vectors in Ω	α	The number of data nodes
V_q^j	The j-th query vector	β	The number of executing nodes
$\Theta(V_q^j,r_j)$	The j-th query sphere, $j\in[1,m]$	$Vol(\cdot)$	The volume of $(\cdot)$
$\Theta(V_Q^j,R_j)$	The j-th query cluster, $j\in[1,m']$ and $m'<m$		

定义 1(网格形式化描述). 网格(G)可看作一张图,由结点(node)和边(edge)构成,形式化表示为 $G=(N,E)$,其中 N 为结点集,E 为边,表示结点间数据传输的网络带宽.

定义 2(网格结点). 网格(G)中的结点(N)分为 3 种类型:查询结点(N_q)、数据结点(N_d)和执行结点(N_e),形式化表示为$N=N_q+N_d+N_e$,其中N_q由 1 个结点构成,N_d包含α个数据结点(N_d^i),N_e包含β个执行结点(N_e^j),其中 N_d^i 为第i个数据结点且$i\in[1,\alpha]$, N_e^j 为第j个执行结点且$j\in[1,\beta]$(如图 2 所示).

根据定义 2,在该网格中,查询结点N_q负责提交用户的查询请求和接受返回结果(Ω'),同时对查询进行动态批量聚类与排序.α个数据结点N_d用于存储高维向量(Ω)及其索引,向量集缩减在该结点层面并行完成,返回候选向量集(Ω');求精过程(距离计算)在β个执行结点N_e上并行完成,返回结果向量集(Ω'').不失一般性,假设向量在高维空间均匀分布.

定义 3(最小包围超球). 给定m个查询超球:$\Theta(V_q^j,r_j)$,它们对应的最小包围超球(minimal bounded sphere,简称MBS)表示为$MBS\left(\bigcup_{j=1}^{m}\Theta\left(V_q^j,r_j\right)\right)\supseteq\bigcup_{j=1}^{m}\Theta\left(V_q^j,r_j\right)$,使得$Vol\left(MBS\left(\bigcup_{j=1}^{m}\Theta\left(V_q^j,r_j\right)\right)\right)$最小,其中$j\in[1,m]$.

如图 3 所示,给定两个查询超球:$\Theta(V_q^i,r_i)$和$\Theta(V_q^j,r_j)$,它们对应的MBS同样是一个超球,表示为$MBS(\Theta(V_q^i,r_i),\Theta(V_q^j,r_j))=\Theta(V_Q^x,R_x)$.$\Theta(V_Q^x,R_x)$对应的中心$(V_Q^x)$和半径表示为$V_Q^x=V_q^i+\frac{d(V_q^i,V_q^j)-r_i+r_j}{2\times d(V_q^i,V_q^j)}\times(V_q^j-V_q^i)$或$V_Q^x=V_q^j+\frac{d(V_q^i,V_q^j)-r_j+r_i}{2\times d(V_q^i,V_q^j)}\times(V_q^i-V_q^j)$,半径$R_x=\frac{d(V_q^i,V_q^j)+r_i+r_j}{2}$.

定义 4(最大内切超球). 给定两个超球:$\Theta(V_Q,R)$和$\Theta(V_q,r)$,其对应的最大内切超球(maximal inner tangent sphere,简称MITS)表示为一个超球$\Theta(V_x,R_x)$,它包含在这两个超球的相交部分,其中$R_x=\frac{r-d(V_Q,V_q)+R}{2}$,$V_x=\frac{d(V_Q,V_q)-R+r}{2\times d(V_Q,V_q)}\times(V_Q-V_q)+V_q$或$V_x=\frac{d(V_Q,V_q)+R-r}{2\times d(V_Q,V_q)}\times\left(V_q-V_Q\right)+V_Q$.

定理 1. 给定两个查询超球:$\Theta(V_q^i,r_i)$和$\Theta(V_q^j,r_j)$,其对应的最小包围超球(MBS)表示为$\Theta(V_Q^x,R_x)$,当满足:$\frac{Vol(\Theta(V_q^i,r_i)\cup\Theta(V_q^j,r_j))}{Vol(\Theta(V_Q^x,R_x))}>\frac{1}{2}$时,则得到:$\begin{cases}2\times(r_i^D+r_j^D)>d(V_q^i,V_q^j)^D & \text{if } r_i+r_j>d(V_q^i,V_q^j)>r_i-r_j\\ 2\times(r_i^D+r_j^D)>(r_i+r_j)^D & \text{if } r_i+r_j\le d(V_q^i,V_q^j)\end{cases}$.

证明:如图 5 所示,根据半径及与中心间的距离可分成 3 种情况.为了描述方便,令A为$\Theta(V_q^i,r_i)$,B为$\Theta(V_q^j,r_j)$,C为$\Theta(V_Q^x,R_x)$,且$r_i>r_j$,则按照两个查询超球中向量总数是否大于阴影部分中的向量个数,存在以下 3 种情况:

(i) A包含B.如图 5(a)所示,即$r_i>d(V_q^i,V_q^j)+r_j$,则$\frac{Vol(A\cup B)}{Vol(C)}=\frac{Vol(A)}{Vol(A)}=1>\frac{1}{2}$.

(ii) A与B相交.如图 5(b)所示,即$r_i+r_j>d(V_q^i,V_q^j)\ge r_i-r_j$,令$\frac{Vol(A\cup B)}{Vol(C)}=\frac{Vol(A)+Vol(B)-Vol(A\cap B)}{Vol(C)}>\frac{1}{2}$,则$\frac{\frac{r_i^D\times\pi^{D/2}}{\Gamma(D/2+1)}+\frac{r_j^D\times\pi^{D/2}}{\Gamma(D/2+1)}-Vol(A\cap B)}{\left(\frac{r_i+r_j+d(V_q^i,V_q^j)}{2}\right)^D\times\pi^{D/2}}>\frac{1}{2}$.由于任意两个高维超球$A$和$B$的交集部分体积$Vol(A\cap B)$的计算代价非常高[21],因此采用$A$和$B$的MITS的体积来近似表示.如图 4 所示,给定两个相交的超球:$\Theta(V_Q,R)$和$\Theta(V_q,r)$,其对应的MITS可用一个内切阴影圆来表示,其半径$x=\frac{r-d(V_Q,V_q)+R}{2}$,则$Vol(\Theta(V_Q,R)\cap\Theta(V_q,r))\approx Vol(MITS(\Theta(V_Q,R),\Theta(V_q,r)))=\frac{x^D\times\pi^{D/2}}{\Gamma(D/2+1)}=\frac{\left(\frac{r-d(V_Q,V_q)+R}{2}\right)^D\times\pi^{D/2}}{\Gamma(D/2+1)}$.

另外,因为$Vol(A\cap B)\approx Vol(MITS(A,B))$,则$Vol(A\cap B)\approx\frac{\left(\frac{r_i+r_j-d(V_q^i,V_q^j)}{2}\right)^D\times\pi^{D/2}}{\Gamma(D/2+1)}$.合并上式,则$r_i^D+r_j^D-\frac{1}{2}\times\left(\frac{r_i+r_j+d(V_q^i,V_q^j)}{2}\right)^D>\left(\frac{r_i+r_j-d(V_q^i,V_q^j)}{2}\right)^D$.由于$r_i+r_j>d(V_q^i,V_q^j)$,则$\frac{1}{2}\times\left(\frac{r_i+r_j-d(V_q^i,V_q^j)}{2}\right)^D>\frac{1}{2}\times$

$d(V_q^i,V_q^j)^D$.另外,由于$\left(\frac{r_i+r_j-d(V_q^i,V_q^j)}{2}\right)^D>0$,所以得到$2\times(r_i^D+r_j^D)>d(V_q^i,V_q^j)^D$.

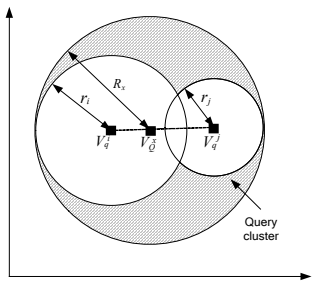


Fig.3 Corresponding MBS of the two spheres
图 3 两个超球对应的 MBS

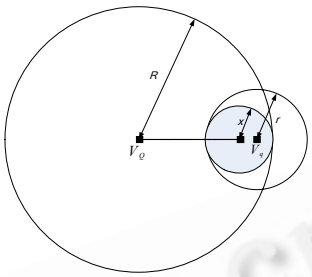
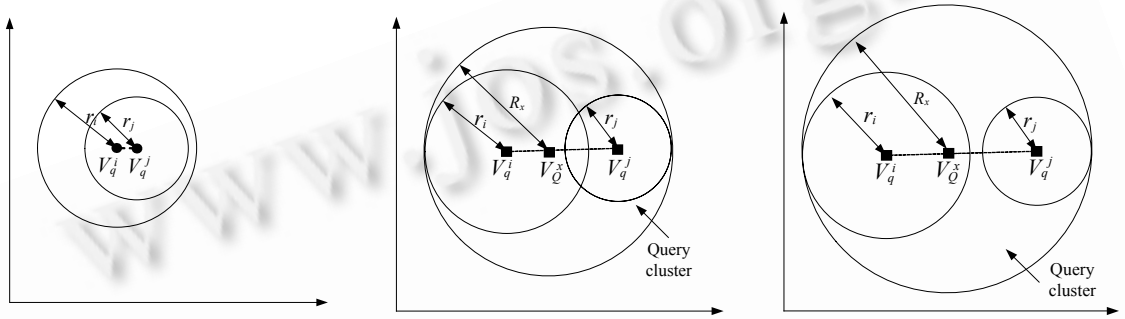


Fig.4 Corresponding MITS of the two spheres
图 4 两个超球对应的 MITS



(a) $r_i>d(V_q^i,V_q^j)+r_j$ (b) $r_i+r_j>d(V_q^i,V_q^j)\geq r_i-r_j$ (c) $r_i+r_j\leq d(V_q^i,V_q^j)$

Fig.5 Three cases for the query clustering
图 5 查询聚类的 3 种情况

(iii) B 与 A 相外切或 B 不与 A 相交,如图 5(c)所示,即 $r_i+r_j\leq d(V_q^i,V_q^j)$,令 $\frac{Vol(A\cup B)}{Vol(C)}=\frac{Vol(A)+Vol(B)}{Vol(C)}>\frac{1}{2}$,则有

$$\frac{\frac{r_i^D\times\pi^{D/2}}{\Gamma(D/2+1)}+\frac{r_j^D\times\pi^{D/2}}{\Gamma(D/2+1)}}{\frac{\left(\frac{r_i+r_j+d(V_q^i,V_q^j)}{2}\right)^D\times\pi^{D/2}}{\Gamma(D/2+1)}}>\frac{1}{2},$$

得到$2\times(r_i^D+r_j^D)>\left(\frac{r_i+r_j+d(V_q^i,V_q^j)}{2}\right)^D$.由于$r_i+r_j\leq d(V_q^i,V_q^j)$,则$2\times(r_i^D+r_j^D)>(r_i+r_j)^D$.

基于以上分析,得到$\begin{cases}2\times(r_i^D+r_j^D)>d(V_q^i,V_q^j)^D, & \text{if } r_i+r_j>d(V_q^i,V_q^j)>r_i-r_j\\ 2\times(r_i^D+r_j^D)>(r_i+r_j)^D, & \text{if } r_i+r_j\leq d(V_q^i,V_q^j)\end{cases}$. □

3 支撑技术

为了更好地支持网格环境下的分布式相似查询的多重优化(pGMSQ)处理,提出 3 种支撑技术,包括动态查询聚类算法、自适应负载均衡及索引支持的向量集缩减.这些方法的目的是降低网络传输的代价并提高查询的并行性.

3.1 动态查询层次聚类算法

在某个时段,对于用户提交的一批查询请求,较难对这些查询进行快速而有效的聚类.为此,提出一种动态查询层次聚类算法(dynamic query clustering scheme,简称 DQC),得到新的查询类.该查询请求集可形式化表示

为

$$QSet ::=\langle Q_1,Q_2,...,Q_m\rangle \tag{1}$$

其中$Q_j=\Theta(V_q^j,r_j)$是指第j个用户提交的查询请求,$j\in[1,m]$.

需要注意的是,DQC 算法的第 3 行是基于定理 1 的.也就是说,对于任意两个超球,当满足定理 1 的结论时,则可将两者合并成一个新的查询超球,称为查询类.

算法 1. Dynamic query clustering(DQC) algorithm.

输入:m query spheres;

输出:m' query clusters.

1. **while**(TRUE)
2. **for** any two hyperspheres A and B **do**
3. **if** Theorem.(1) is satisfied **then**
4. A and B are merged;
5. update the query list;
6. $m\leftarrow m-1$;
7. **end if**
8. **end for**
9. **end while** /* the value of m has been reduced to m' and $m'<m$*/
10. **return** m'(updated m) query clusters;

3.2 自适应数据负载均衡

3.2.1 基于“查询投票”的结点处理能力估计

对于多数并行数据库系统来说,数据分片对查询性能的影响非常关键.如第 1 节所述,网格是一个异构网络环境,其中每个结点的处理能力(如磁盘存储能力、磁盘转速及数据传输率)都不相同.因此,较难对每个结点(如数据结点或执行结点)的综合处理能力进行准确的模型建立.因此,作为数据分片的预处理阶段,对于每个结点的处理能力的估计对优化的数据分片非常重要.本节提出查询投票(query voting,简称 QV)的方法.

在该方法中,每个结点的处理能力可以通过来自不同用户查询请求得到的平均查询时间来度量,如图 6 所示.给定m个用户和x个结点,令T_{ij}为来自第j个用户查询请求的在第i个结点上的响应时间,其中$i\in[1,x]$且$j\in[1,m]$.

这样,对于 m 个用户和 x 个结点,可以得到一张用户时间表(user-time,简称 UT),见式(2):

$$UT=\begin{bmatrix} T_{11} & T_{12} & ... & T_{1m} \\ T_{21} & T_{22} & ... & T_{2m} \\ & & ... & \\ T_{x1} & T_{x2} & ... & T_{xm} \end{bmatrix} \tag{2}$$

对于第i个结点,其处理能力(记作ρ_i)与其平均查询时间 $\frac{1}{m}\sum_{j=1}^{m}T_{ij}$ 成反比,见式(3):

$$\rho_i \propto \frac{1}{\frac{1}{m}\sum_{j=1}^{m}T_{ij}}=\frac{m}{\sum_{j=1}^{m}T_{ij}} \tag{3}$$

也就是说,平均响应时间可以通过ρ_i的函数来表示,见式(4):

$$f(\rho_i)=\frac{1}{\frac{1}{m}\sum_{j=1}^{m}T_{ij}}=\frac{m}{\sum_{j=1}^{m}T_{ij}} \tag{4}$$

其中,$i\in[1,x]$且 $j\in[1,m]$.

基于上面的推导,对于第 i 个结点来说,其处理能力所占比率可以表示为

$$Per(i)=\frac{1/\sum_{j=1}^{m}T_{ij}}{1/\sum_{j=1}^{m}T_{1j}+1/\sum_{j=1}^{m}T_{2j}+\ldots+1/\sum_{j=1}^{m}T_{xj}} \tag{5}$$

算法 2. Query voting algorithm.

输入: Ω: Vector set, x nodes, m users;

输出: ρ_i: The processing capabilities of x nodes.

1. **for** i:=1 to x **do** /*对每个结点来说*/
2. **for** j:=1 to m **do** /*对每个用户来说*/
3. submit the j-th user query from i-th node;
4. the query executing time(T_{ij}) is recorded;
5. **end for**
6. the processing capability (ρ_i) of the i-th node is obtained by Eq.(4);
7. **end for**

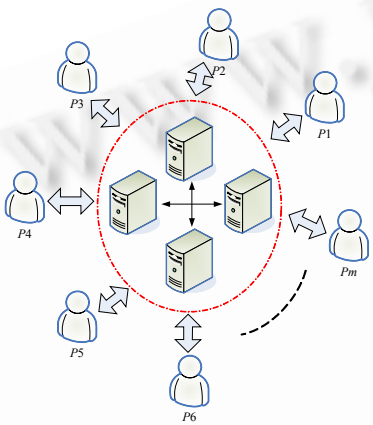


Fig.6 A query polling example
图 6 查询投票示例

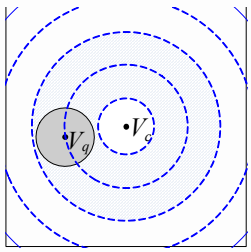


Fig.7 Centroid rings intersected with query sphere
图 7 与查询超球相交的中心

3.2.2 基于中心环的自适应数据分布策略

为了使数据结点层的向量集缩减的并行度最大化,提出一种基于中心环(centroid-ring)的自适应负载均衡策略,这样可以保证向量随机地分布在不同的数据结点上.具体来说,该方法可以保证每个结点上几乎都存在一个类超球与查询超球相交.这样,对于每个查询,向量集缩减可以在每个结点并行执行.

定义 5(中心距离,centroid distance). 给定一个向量V_i,其中心距离为其到中心点V_c的距离,记作$CD(V_i)=d(V_i,V_c)$,其中V_c=(0.5,0.5,…,0.5).

假设将高维空间按照中心距离均匀地"切"成α个"分片",如图 7 所示.该分片定义为中心环.随着中心距离的增加,中心环编号也随之增加.

定义 6(中心环,centroid ring). 给定一个向量V_i,其对应中心环(记作$R(V_i)$)的编号为$\lceil 2\alpha\cdot CD(V_i)/\sqrt{d}\rceil+1$.

对于不同数据结点的向量数据分布(如算法 3),将每个中心环中的向量按照数据结点处理能力的大小随机取出.这样可以保证几乎在每个数据结点上,查询超球都能与类超球**相交.

算法 3. Vector allocation algorithm.

** 由于每个结点上的向量集采用iDistance[6]来建立索引.而在建立iDistance索引之前,需要对高维向量聚类,得到若干类超球.

输入: Ω: The vector set, α data nodes;

输出: Ω(1 to α): The vector set stored in the α data nodes.

The high-dimensional spaces is equally sliced into α centroid rings;

for each data node N_d^j **do**

3. randomly select $|per(j)\times n|$ vectors ($\Omega(j)$) from α centroid rings;
4. the vectors $\Omega(j)$ are stored in the j-th data node;
5. **end for**

3.3 索引支持的快速向量集缩减

由于向量集(Ω)存储在数据结点层,对于一个查询,直接在数据结点层执行向量的求精(距离计算)操作显然是低效的.因此,提出iDistance[8]索引支持的向量集缩减(VSR)方法.该方法的目的是减少求精过程的CPU计算量和网络传输代价.算法 4 给出第j个数据结点的向量集缩减.其中,函数$RSearch(V_p,r)$表示返回中心为V_q半径为r的范围查询.

算法 4. Vector set reduction algorithm (VReduce).

输入: V_q: Query vector, r: Query radius, $\Omega(j)$: The vector set in the j-th data node and $j\in[1,\alpha]$;

输出: $\Omega'(j)$: The candidate vector set in the j-th data node.

1. $S1\leftarrow\varnothing$, $S2\leftarrow\varnothing$; /*初始化*/
2. **for each** cluster sphere $\Theta(O_j,CR_j)$ **do**
3. **if** $\Theta(O_j,CR_j)$ dose not intersect with $\Theta(V_q,r)$ **then**
4. break;
5. **else**
6. $S2\leftarrow RSearch(V_q,r)$;
7. $S1\leftarrow S1\cup S2$;
8. **if** $\Theta(O_j,CR_j)$ contains $\Theta(V_q,r)$ **then** end loop;
9. **end if**
10. **end for**
11. **return** $S1$; /*返回候选向量*/

4 pGMSQ 算法

基于上述 3 种支撑技术,我们提出一种基于网格并行流水线的分布式多重相似查询优化策略——pGMSQ.在介绍该算法之前,首先假设Ω中的向量数据已经按照中心环部署在数据结点层并且在每个数据结点分别采用iDistance[8]进行索引,用以支持快速的向量集缩减.pGMSQ算法分为如下 3 步:

(1) 动态查询聚类.作为pGMSQ的第 1 步,首先用户提交m个查询请求到查询结点N_q,然后执行动态查询聚类返回m'个新的查询类作为新的查询请求,其中$m'<m$.具体步骤如算法 1 所示.

(2) 全局向量集缩减.完成查询结点层的动态查询调度后,将 m'个新的查询请求($\Theta(V_Q^i,R_i)$)打包并行地发送到α个数据结点.在数据结点层并行执行索引支持的向量集缩减(参见第 3.3 节).具体来说,对于每个数据结点N_d^j,首先建立输入缓存IB_j用于保存$\Omega(j)$中的向量,然后建立一个输出缓存OB_j用于存储候选向量集$\Omega'(j)$,其中,$j\in[1,\alpha]$.一旦得到候选向量集$\Omega'(j)$,就对其进行散列操作并发送到执行结点层进行求精操作.

算法 5. Global vector set reduction algorithm (GVReduce).

输入: $\Omega(j)$: The vector set in the j-th data node, $\Theta(V_Q^i,R_i)$: m' new query cluster spheres, α: Number of the data nodes;

输出: Ω'(1 to α): The candidate vector set.

1. **for** j:=1 to α **do** /*对于α个数据结点来说*/
2. **for** i:=1 to m' **do**
3. $\Omega'(j)\leftarrow\Omega'(j)\cup VReduce(V_Q^i,R_i,j)$;
4. **end for**
5. $\Omega'(j)$ is cached in the output buffer OB_j;
6. **end for**

在算法 5 中,函数$VReduce(V_q,r,j)$(参见算法 4)返回第j个数据结点的候选向量集$\Omega'(j)$.向量集缩减是在不同数据结点上并行执行的.

(3) 求精操作.作为最后一步,通过 GRMS 动态地发现β个空闲结点作为执行结点,并在其上并行地计算候选向量与查询向量V_q^i的距离.如算法 6 所述,当距离值小于或等于r_i时,其中$i\in[1,m]$,则该结果向量可以通过打包方式发送到查询结点N_q.与全局向量集缩减中采用新的m'查询类作为查询超球不同,在求精过程中,是采用m个原来的查询超球来执行距离计算操作,其中$m'<m$.

算法 6. Refine algorithm.

输入: $\Omega(j)$: The candidate vector set in the j-th data node, V_q^i: m query vectors and m query radius r_i, β: Number of the executing nodes;

输出: Ω'(1 to β): The answer vector set.

1. **for** j:=1 to β **do** /*对于β个执行结点来说*/
2. $\Omega'(j)\leftarrow\varnothing$;
3. **for** i:=1 to m **do**
4. **if** $d(V_i,V_q^i)\le r_i$ **and** $V_i\in\Omega(j)$ **then** $\Omega'(j\leftarrow\Omega'(j)\cup V_i$;
5. **end for**
6. $\Omega'(j)$ is cached in the output buffer OB_j;
7. **end for**

在第j个执行结点N_e^j,建立一个输入缓存IB_j用于存储候选向量集$\Omega'(j)$,同时为$\Omega'(j)$分配在结点N_e^j上的内存空间M_j,用于暂存$\Omega(j)$的向量.另外,建立一个输出缓存OB_j,用于暂时缓存结果向量.一旦在OB_j中的结果向量集大小等于包的大小,则以打包方式发送回查询结点N_q.

算法 7 为 pGMSQ 算法.其中函数 $GVReduce(\Omega,m'$ query clusters)为全局向量缩减算法(参见算法 5);函数$Refine(\Omega',m$ query spheres)是对候选向量集Ω'进行求精操作,如算法 6 所示;步骤 4~步骤 7 并行执行.

算法 7. The pGMSQ algorithm.

输入: A query list containing several queries;

输出: Ω'': Query result.

1. **while** the query list is not empty **do**
2. $\Omega'\leftarrow\varnothing$, $\Omega''\leftarrow\varnothing$; /*初始化*/
3. m queries are extracted from the query list and submitted to the query node N_q;
4. The m queries are clustered in the query node; /*在查询结点层*/
5. $\Omega'\leftarrow GVReduce(\Omega,m'$ query clusters,$\alpha)$; /*在数据结点层*/
6. The β executing nodes are dynamically discovered by the GRMS; /*动态发现β个空闲结点作为执行结点*/
7. $\Omega''\leftarrow Refine(\Omega',m$ query spheres,$\beta)$; /*在执行结点层*/
8. The query answer Ω'' is returned to the query node N_q;
9. **end while**

由于以上讨论的查询是相对静态的,当用户的查询请求持续不断地产生时,为了进一步提高系统吞吐量,则

需采用流水线技术,即在查询结点、数据结点和执行结点上的操作并发执行,使得这 3 个结点层在每一时刻都能够同时工作.

定义 7(加速比). 给定 m 个查询请求,加速比是指 m 次执行基于网格的相似查询(GSQ)所需的时间与基于网格并行流水线的多重查询(pGMSQ)所需时间之比,记作 $Speedup=\dfrac{\sum_{i=1}^{m}TIME_{\mathrm{GSQ}}}{TIME_{\mathrm{pGMSQ}}}$.

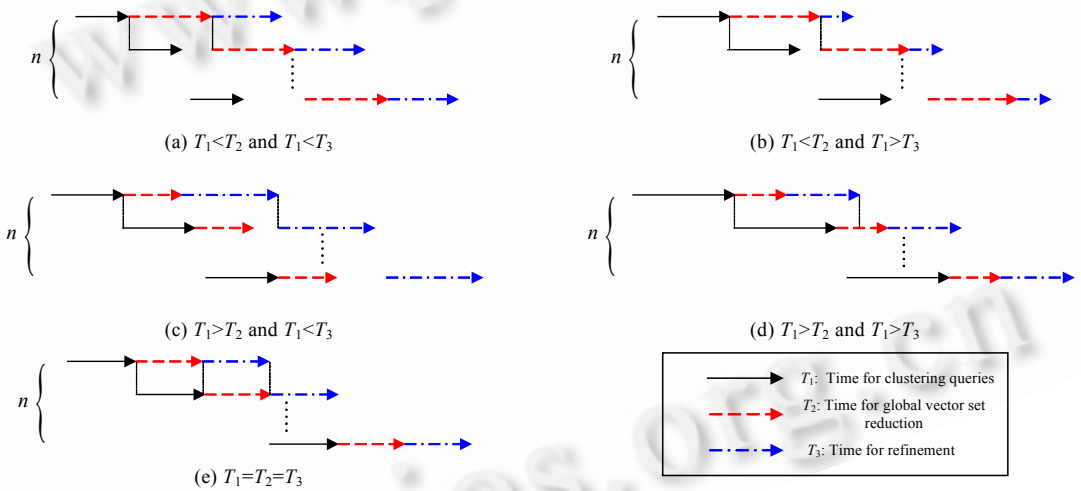
假设查询提交及对其聚类所需时间为T_1,向量集缩减所需时间为T_2,求精操作所需时间为T_3,根据T_1,T_2和T_3的大小,分成 5 种情况讨论,如图 8 所示,给定n批查询,其加速比分别表示为

(1) 当$T_1<T_2$, $\lim\limits_{n\to\infty}Speedup=\lim\limits_{n\to\infty}\dfrac{\sum_{i=1}^{n}TIME_{\mathrm{GSQ}}}{TIME_{\mathrm{pGMSQ}}}=\lim\limits_{n\to\infty}\dfrac{n\times(T_1+T_2+T_3)}{T_1+n\times T_2+T_3}\approx\dfrac{T_1+T_2+T_3}{T_2}<3.$

(2) 当$T_1>T_2$且$T_1<T_3$, $\lim\limits_{n\to\infty}Speedup=\lim\limits_{n\to\infty}\dfrac{\sum_{i=1}^{n}TIME_{\mathrm{GSQ}}}{TIME_{\mathrm{pGMSQ}}}=\lim\limits_{n\to\infty}\dfrac{n\times(T_1+T_2+T_3)}{T_1+T_2+n\times T_3}\approx\dfrac{T_1+T_2+T_3}{T_3}<3.$

(3) 当$T_1>T_2$且$T_1>T_3$, $\lim\limits_{n\to\infty}Speedup=\lim\limits_{n\to\infty}\dfrac{\sum_{i=1}^{m}TIME_{\mathrm{GSQ}}}{TIME_{\mathrm{pGMSQ}}}=\lim\limits_{n\to\infty}\dfrac{n\times(T_1+T_2+T_3)}{n\times T_1+T_2+T_3}\approx\dfrac{T_1+T_2+T_3}{T_1}<3.$

(4) 当$T_1=T_2=T_3$时,令$T_1=T_2=T_3=T$,则 $\lim\limits_{n\to\infty}Speedup=\lim\limits_{n\to\infty}\dfrac{\sum_{i=1}^{n}TIME_{\mathrm{GSQ}}}{TIME_{\mathrm{pGMSQ}}}=\lim\limits_{n\to\infty}\dfrac{3n}{n+1}\approx 3.$



(a) $T_1<T_2$ and $T_1<T_3$ (b) $T_1<T_2$ and $T_1>T_3$

(c) $T_1>T_2$ and $T_1<T_3$ (d) $T_1>T_2$ and $T_1>T_3$

(e) $T_1=T_2=T_3$

Fig.8 Five cases of pipeline-based parallel query

图 8 基于流水线并行查询的 5 种情况

基于上面的分析可以看出,当$T_1=T_2=T_3$时,每个结点层能够更有效地同时工作且相互无影响.系统在理论上具有最好的并行性.本文在第 5.4 节实验部分通过调节数据结点(α)和执行结点(β)的个数来 120 获到最大的流水线并行性,即满足$T_1=T_2=T_3$.

5 实验比较

为了验证pGMSQ方法的有效性,构建一个局域网环境来模拟网格环境,采用随机函数动态产生执行结点.同时用C语言实现基于iDistance的向量集缩减算法并将其部署在每个数据结点.B+树作为一维索引结构且索引页大小为 4 096 字节.采用两组数据:(i) 真实数据,来自UCI KDD Archive[22]的颜色直方图数据,包含 68 040 个 32 维的颜色直方图特征,每维值域范围为 0~1;(ii) 合成数据,即 5 000 000 个 100 维的向量随机产生,满足均匀分布,其中每维值域范围也是 0~1.在实验中,随机产生 100 个用户的查询请求,对于每组实验分别运行 100 次,以得

到平均时间.

5.1 VSR对查询的影响

本次实验采用两种方法分别研究向量集缩减(VSR)对pGMSQ查询性能的影响.方法 1 不采用VSR算法:向量集Ω的求精处理直接在数据结点层N_d进行而不执行向量集缩减,然后将结果向量集Ω'发送回查询结点N_q.方法 2 采用VSR算法:首先,向量集Ω通过算法 4 缩减为候选向量集Ω',然后,对Ω'中的向量进行求精处理,得到结果向量,最后,结果向量集Ω'发送回查询结点N_q.如图 9 所示,随着数据量的增加,采用VSR的查询性能大大优于没有采用的.因为被iDistance[6]过滤的无关向量会极大地减少在求精过程中的计算量及网络传输代价.

5.2 DQC对查询的影响

本次实验分别采用两种方法研究动态查询聚类(DQC)对查询性能的影响.方法 1 不采用 DQC 算法.方法 2 采用该算法.从图 10 可以看出,数据量的增加导致采用方法 2 的 DQC 比不采用的其性能大为提高.且两者查询性能差异变大,因为通过使用 DQC,多重查询的性能会进一步得以提高.

5.3 维数对查询的影响

本次实验研究维数对加速比的影响,采用合成数据作为本次实验的测试数据,其中每个向量的维数从20~100.本次实验数据量为 1 000 000.如图 11 所示,随着维数的增加,其加速比在缓慢减少.因为维数的增加会导致在求精过程中消耗更高的CPU运算代价.另外,对于每个结果向量来说,维数的增加也会导致数据传输代价的提高.

5.4 α和β对基于流水线并行查询的影响

本次实验同样采用第 2 组数据集来研究数据结点个数(α)和执行结点个数(β)对流水线并行查询加速比的影响.如图12所示,随着α的增加,其加速比也在缓慢的增加.当数据结点个数超过40时,加速比在减少.这是因为,随着数据结点的增加,从查询结点到数据结点层的传输代价在增加,它会部分抵消向量集缩减带来的性能提高.另外,从图 13 可以看出,随着β的增加,其加速比首先会缓慢增加.当β超过 30 时,加速比在减少.这是因为随着执行结点的增加,从数据结点到执行结点层的传输代价在增加,它会部分抵消并行查询带来的性能提高.

5.5 m对查询的影响

本次实验采用第 2 组数据集研究用户查询请求个数(m)对加速比的影响.本次实验中,假设数据量为1 000 000、数据结点个数(α)为 40 和执行结点个数(β)为 30,当 m 从 20 增大到 100 时,从图 14 可以看出,m 对加速比无太大影响.

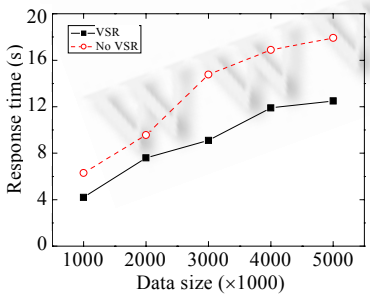


Fig.9 Effect of VSR

图 9 VSR 对查询的影响

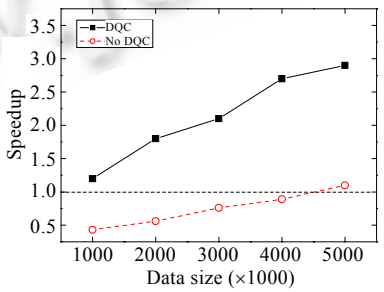


Fig.10 Effect of DQC

图 10 DQC 对查询的影响

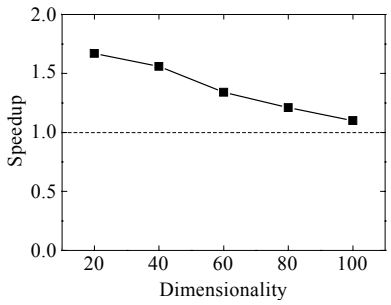


Fig.11 Effect of dimensionality

图 11 维数对查询的影响

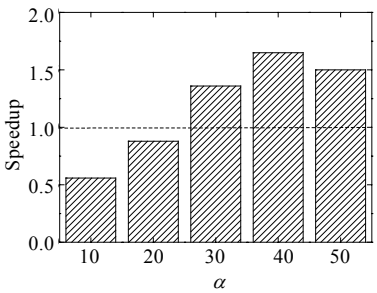


Fig.12 α vs. Speedup
图 12 α和加速比

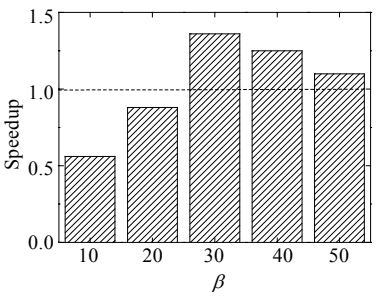


Fig.13 β vs. Speedup
图 13 β和加速比

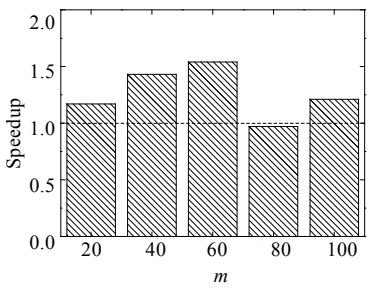


Fig.14 Effect of m
图 14 m 对查询的影响

6 总 结

本文提出了一种面向网格环境的基于流水线技术的分布式多重相似查询优化算法(pGMSQ).为了有效支持 pGMSQ,提出 3 种支撑技术,如动态查询聚类策略、基于中心环的自适应负载均衡和索引支持的向量集缩减,来减少查询的时间.实验结果表明,本文提出的方法能够有效地减少查询中的通信代价及最大化 I/O 和 CPU 的并行性,特别适合于海量的分布式相似检索.今后,我们将对 pGMSQ 算法加以扩展,以支持基于高维子空间相似查询的多重优化策略.

References:

[1] Sellis TK. Multi-Query optimization. ACM Trans. on Database Systems, 1988,13(1):23−52.

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

[3] 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). 1984. 47−54.

[4] Beckmann N, Kriegel HP, Schneider R, Seeger B. The R*-tree: An efficient and robust access method for points and rectangles. In: Garcia-Molina H, Jagadish HV, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2000). 1990. 322−331.

[5] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network. In: Proc. of the ACM SIGCOMM Int'l Conf. on Data Communication (SIGCOMM). 2001. 161−172.

[6] Aspnes J, Shah G. Skip graphs. In: Proc. of the ACM-SIAM Symp. on Discrete Algorithms. 2003. 384−393.

[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] 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, 2005,2(30):364−397.

[9] Berchtold S, Bohm C, Braunmuller B, Keim DA, Kriegel HP. Fast parallel similarity search in multimedia databases. In: Peckham J, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'97). 1997. 1−12.

[10] Papadopoulos AN, Manolopoulos Y. Similarity query processing using disk arrays. In: Haas LM, Tiwary A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'98). 1998. 225−236.

[11] Weber R, Schek H, 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 Int'l Conf. on Very Large Databases (VLDB'98). 1998. 194−205.

[12] Lee J, Lee H, Kang S, Choe S, Song J. CISS: An efficient object clustering framework for DHT-based peer-to-peer applications. In: Proc. of the DBISP2P. 2004. 215−229.

[13] Tang C, Xu Z, Dwarkadas S. Peer-to-Peer information retrieval using self-organizing semantic overlay networks. In: Proc. of the ACM SIGCOMM Int'l Conf. on Data Communication (SIGCOMM). 2003.

[14] Sahin OD, Gupta A, Agrawal D, Abbadi AE. A peer-to-peer framework for caching range queries. In: Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE 2004). 2004.

[15] Zhang C, Krishnamurthy A, Wang RY. Skipindex: Towards a scalable peer-to-peer index service for high dimensional data. Technical Report, TR-703-04, Princeton University, 2004.

[16] Kalnis P, Ng WS, Ooi BC, Tan KL. Answering similarity queries in peer-to-peer networks. Information Systems, 2006,31(1): 57−72.

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

[18] 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.

[19] 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).

[20] Zhuang Y, Li Q, Chen L. Multi-Query optimization for distributed similarity query processing. In: Proc. of the IEEE Int'l Conf. on Distributed Computing Systems (ICDCS 2008). 2008. 639−646.

[21] Shen HT, OOi BC, Zhou XF, Huang Z. Towards effective indexing for very large video sequence database. In: Özcan F, ed. Proc. of the 24th ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD). 2005. 730−741.

[22] UCI KDD Archive. 2002. http://www.kdd.ics.uci.edu

附中文参考文献:

[19] 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.

胡华(1964－),男,江西南昌人,博士,教授,主要研究领域为软件工程,协同计算.

胡海洋(1977－),男,博士,副教授,CCF 高级会员,主要研究领域为软件工程.

庄毅(1978－),男,博士,讲师,CCF 会员,主要研究领域为多媒体数据库,分布式计算,索引技术.

赵格华(1966－),男,博士,讲师,主要研究领域为工作流,信息处理.