DPslR$^+$:一种基于动态空间槽的分布式并行空间索引树*)

左朝树　刘心松　陈小辉　顾　攀
(电子科技大学计算机科学与工程学院　成都 610054)

摘　要　空间索引是空间数据库的关键组成部分,其性能的优劣直接决定着空间数据操作的效率。为此,在分析了现有各种空间索引的基础上,将分布并行处理技术与空间索引相融合,提出了一种DPslR$^+$树。DPslR$^+$树借助繁衍和返祖,动态分割空间槽,并将它们映射到多个节点机上。每个节点机再将其对应空间槽中的空间对象组织成R树,并将R树分裂成多个残枝,将残枝并行存入本地MultiDisk中;在内存中则按R-link组织空间对象,按R$^+$处理节点溢出。实验结果表明DPslR$^+$树具有良好的查询特性。

关键词　空间索引,空间槽,繁衍,返祖,R树,DPslR$^+$树

DPslR$^+$: A Distributed and Parallel Spatial Index Tree Based on Dynamic Spatial Slot

ZUO Chao-Shu　LIU Xin-Song　CHEN Xiao-Hui　GU Pan
(School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054)

Abstract　Spatial index is important part of spatial database, whose performance is vital to efficiency of spatial operation. After various spatial indexes were analyzed and distributed and parallel processing was introduced into spatial index, this paper has come up with a DPslR$^+$ tree, which splits space into multi-slots by multiplying and reverting and maps these slots to sites in distributed and parallel system. Each site constructs the spatial objects in their spatial slot into an R tree, splits the R into stumps and stores the stumps into multidisk. In main memory, site builds R-link tree and deals with overflow as R$^+$ tree does. Experiment results show that DPslR$^+$ tree behaves well during spatial query.

Keywords　Spatial index, Spatial slot, Multiply, Reversion, R tree, DPslR$^+$ tree

1　引言

当前,空间数据库已在政府、国防、军事、国民经济重要部门以及相关学科领域发挥着越来越重要的作用。基于GIS(Geographic Information Systems)的电子政务,基于GIS的GIG(Global Information Grid),基于多维空间数据的自动化军事指挥决策、人体解剖、电磁学以及多维决策支持等均以空间数据库为其系统基础,MMIS(Multimedia Information System)、DWH(Data Warehousing)等技术的健康发展也离不开空间数据库强有力的支持。但空间数据库在获得广泛应用的同时,各种应用需求也对其空间查询的效率提出了严峻挑战,作为空间查询的关键基础技术——空间索引的性能对空间查询的效率起着极为重要的作用。为此,国内外学者对空间索引进行了广泛深入的研究。其中,对具有良好搜索性能的R树的研究尤为活跃。对R树的研究主要以两种研究思路展开:

(1)对R树的优化改进　自从1984年Guttman[1]提出R树以来,众多研究者针对不同的应用需求对R树进行了各种改进,形成了诸多R树的变种。文[2]中介绍了hilbert R-tree和SR-tree,文[3]中介绍了基于移动空间对象的Q+Rtree,文[4]中介绍了基于多维数据仓库的R*树的改进版DCA-Tree,文[5]中介绍了基于窗口查询的HR$^+$-tree,文[6]中介绍了基于查询执行成本模型的CUR-Tree,文[7,8]中介绍了用于实现空间连接的seeded tree,文[9]中介绍了基于OLAP范围查询的R_n*-tree,文[10]中介绍了SR树,文[11]中介绍了R*树,文[12]中介绍了适合移动对象的LUR,文[13]中介绍了R-link树,文[14]中介绍了基于oversize shelves优化R$^+$树。

(2)实现R树的并行化　文[15]中介绍了并行化的R树——GPR,文[16]中介绍了并行化的R树——Master-client R-trees,文[17]中介绍了基于磁盘阵列的并行R树。

以上两种研究思路具有相容互补性,对R树优化改进后所形成的R树的变种,可将其进行并行化。但对于R树的分布并行化,尚缺乏研究。为此,我们在分析了各种空间索引之后,提出了基于动态空间槽的分布式并行空间索引树——DPslR$^+$。DPslR$^+$树是R树的一种变种,它借助于繁衍和返祖,将被索引空间动态分割成多个空间槽,将每个空间槽映射到一个节点上,该节点再将该空间槽中的空间对象组织成一棵R树,将之并行存入MultiDisk中;在内存中将R树按R-link树组织,以提高索引树的并发性,并按R$^+$树进行节点溢出处理。实验结果表明,DPslR$^+$对于大数据量的空间操作具有良好的查询特性。

2　分布式并行空间数据库系统概述

分布式并行空间数据库系统的结构如图1所示。它是由N台高性能PC通过高速网络连接在一起,构成数据库服务器群,同时并行地为大量用户提供高性能、高可靠、大数据量数据库服务的分布式并行系统。

N台高性能PC机构成N个空间数据库节点,N个空间数据库节点通过高速网络(WAN/LAN)互连起来,形成一个

*)本课题得到四川省科技攻关项目(02GG006-018)基金资助。左朝树　博士研究生,研究方向为分布式并行处理、数据库系统、网络通信。

单一映象的数据库服务器群。数据库服务器群中的任一服务器均可代表整个服务器群接受并处理任一客户请求。用户空间数据被透明地分布在服务器群的各个节点上，每个节点上的空间数据被并行存入该节点的多磁盘上。分布式并行空间数据库系统的相关子系统负责空间数据的动态副本管理、空间数据的数据平衡分布和空间数据的系统故障自动重构等。分布式并行空间数据库系统的空间索引子系统根据空间数据分布情况和查询效率要求，按预定算法对系统中的空间数据建立分布式并行空间索引树 DPslR$^+$。分布式并行空间索引子系统负责空间数据的快速查找，为分布式并行空间执行子系统提供良好支持。分布式并行空间索引是分布式并行索引子系统的重要数据结构。

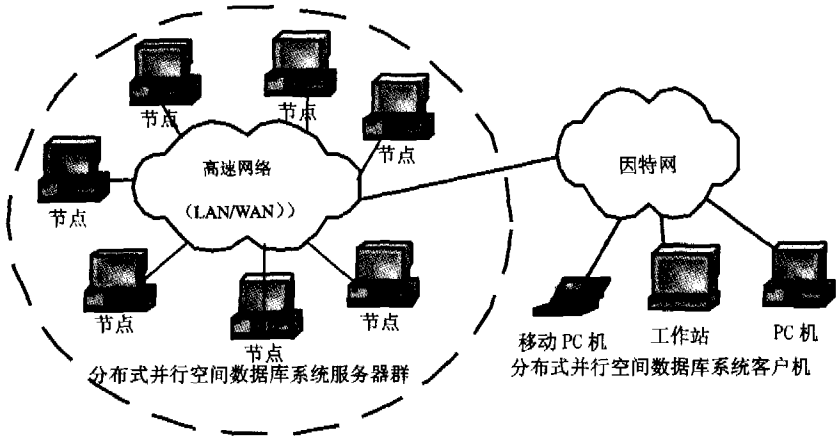


图1　分布式并行空间数据库系统体系结构

3　DPslR$^+$的定义

为严格定义 DPslR$^+$ 树，首先给出相关概念如下：

定义 1(xMBR)　设 n 维线性空间中的空间对象 S_o 被表示为 n 维线性空间中的 m 个点的集合$(p_1,p_2,p_3,\cdots,p_m)$，且第 i 个点在基底 B$(\varepsilon_1,\varepsilon_2,\varepsilon_3,\cdots,\varepsilon_n)$下的坐标为$(x_{i1},x_{i2},x_{i3},\cdots,x_{in})$，则由 $L_i\leqslant y_i\leqslant H_i$(其中 y_i 为 n 维线性空间中的第 i 维，且 $L_i=\min(x_{ij})$，$H_i=\max(x_{ij})$。$1\leqslant j\leqslant m$)围成的 n 维线性空间中的几何体，称为空间对象 S_o 相对于基底 B 的 xMBR，简称为空间对象 S_o 的 xMBR。空间对象的 xMBR 是空间对象在 n 维线性空间中的近似表示，是 DPR^{++} 对空间对象 S_o 进行空间索引的基础。

定义 2(空间槽(SS))　n 维线性空间中由 $d_i\leqslant x_i\leqslant u_i$($x_i$ 为 n 维线性空间中的第 i 维，且$(-\infty\leqslant d_i,u_i\leqslant+\infty)$围成的一个 n 维线性空间中的域 D_n，称为 n 维线性空间中的一个空间槽。用 SS 表示一个空间槽。

定义 3(空间交度(I))　将 n 维线性空间中的空间对象 S_{o1} 和 S_{o2} 看作 n 维线性空间中的点的集合，二者的重叠域 $O=\{(x_1,x_2,x_3,\cdots,x_n)\mid(x_1,x_2,x_3,\cdots,x_n)\in S_{o1}\wedge(x_1,x_2,x_3,\cdots,x_n)\in S_{o2}\}$。则空间对象 S_{o1} 和 S_{o2} 的交度

$$I=\sum_{i=1}^{r}\left|\iiint_{O_i}\mathrm{d}x_1\mathrm{d}x_2\cdots\mathrm{d}x_n\right|$$

$(O_i\cap O_j=\Phi,(i\neq j),O_1\cup O_2\cup\cdots\cup O_r=O))$。当 $n=2$ 时，交度 I 表示二维平面上的两个空间对象之间的重叠域的面积；当 $n=3$ 时，交度 I 表示三维平面上的两个空间对象之间的重叠域的体积。

定义 4(独枝(S$_t$))　在以 root 为根的树 T 中，由(V_i,A_i)确定的树 T 的子树，称为树 T 的独枝，其中 $V_i=(v_1,v_2,\cdots,v_q)$，$v_1=root$，v_q 为 T 的叶子，$v_j\neq$root 并且 v_j 不是 T 的叶子$(j\neq1\wedge j\neq q)$，而，$A_j=\{\langle v_i,v_{i+1}\rangle\mid v_i,v_{i+1}\in V_i\wedge1\leqslant i\leqslant q-1\}$。

定义 5(独枝系数(d_s))　若 $S_{t1},S_{t2},\cdots,S_{tp}$ 均为树 T 的独枝，并且 $S_{t1}\cup S_{t2}\cdots\cup S_{tp}=T$，则树 T 的独枝系数为 p。由定义 2 知，树 T 的独枝系数与树的叶子数相等。

定义 6(空间槽映射 $f(O)$)　从集合 A 到集合 B 上的一个函数，其中 $A=\{O\mid O$ 是问题空间的空间对象$\}$，$B=\{x\mid1\leqslant x\leqslant D\vee x$ 为整数$\}$。

定义 7(残枝分解 $g(T)$)　将树 T 分解成$(T_1,T_2,\cdots,T_u)$的过程，其中 $T_i=S_{i1}\cup\cdots\cup S_{ik}$，$S_{ij}$ 为树 T 的独枝，$T_i\cap T_j=\Phi,i\neq j$，$T_i\cup\cdots\cup T_u=T$，T_i 称为树 T 的残枝，u 称为分解残数，简称残数。$(T_1,T_2,\cdots,T_u))$称为树 T 的一个残解划分。

定义 8(slR$^+$ 树)　R 树的一种变种，当将其存入非挥发性存储设备时，将其看作 R 树，并根据可用非挥发性存储设备的数量决定分解残数，进行残枝分解，将分解后的残解划分的各个残枝分别存入非挥发性存储设备池中；当访问 slR$^+$ 树时，其在内存中的节点通过 link 将兄弟节点联结起来，采用 half split[21]算法处理节点分裂，按 R$^+$ 树处理节点加入上溢。进行残枝分解时采用的分解残数称为 slR$^+$ 的阶，记为 l，若 $l=4$ 时，称其为 4 阶 slR$^+$ 树。

定义 9(DPslR$^+$ 树)　由二元关系　$(V_{root},(slR_1^+,slR_2^+\cdots slR_m^+))$确定的树，称为 DPslR$^+$ 树。其中 V_{root} 称为 DPslR$^+$ 树的虚根，slR$_1^+$，slR$_2^+$，…，slR$_m^+$ 称为 $DPslR^+$ 树的子树。V_{root} 对应的 $xMBR$ 为子树 slR$_1^+$，slR$_2^+$，…，slR$_m^+$ 对应的 $xMBR$ 之和。子树 slR$_1^+$，slR$_2^+$，…，slR$_m^+$ 与虚根 V_{root} 之间无任何边相连。$r=\max(l_1,l_2,\cdots,l_m)$，$l_i$ 为子树 slR$_i^+$ 的阶，r 称为 DPslR$^+$ 树的阶，m 称为 DPslR$^+$ 树的离度，记为 b。将离度为 m 的 r 阶 DPslR$^+$ 简记为(m,r)-DPslR$^+$。

(8,6)-DPslR$^+$ 的结构如图 2。离度为 $m=8$，对应于分布式并行空间数据库系统中当前可用的节点机数，阶 $r=6$ 对应于每个节点机中可用的并行 I/O 磁盘的最大数。slR$_i^+$ 的阶 l_i 对应于节点机 i 当前可用的并行 I/O 磁盘数。空间槽映射函数将空间对象映射到空间槽 SSi 中，空间槽 SSi 对应于分布式并行空间数据库系统中的一个节点机 i，节点机 i 将其对应的空间槽中的空间对象组织成 slR$^+$ 树，并通过残枝分解函数将 slR$^+$ 分解成多个残枝，且将每个残枝分别存入一个本地磁盘中。

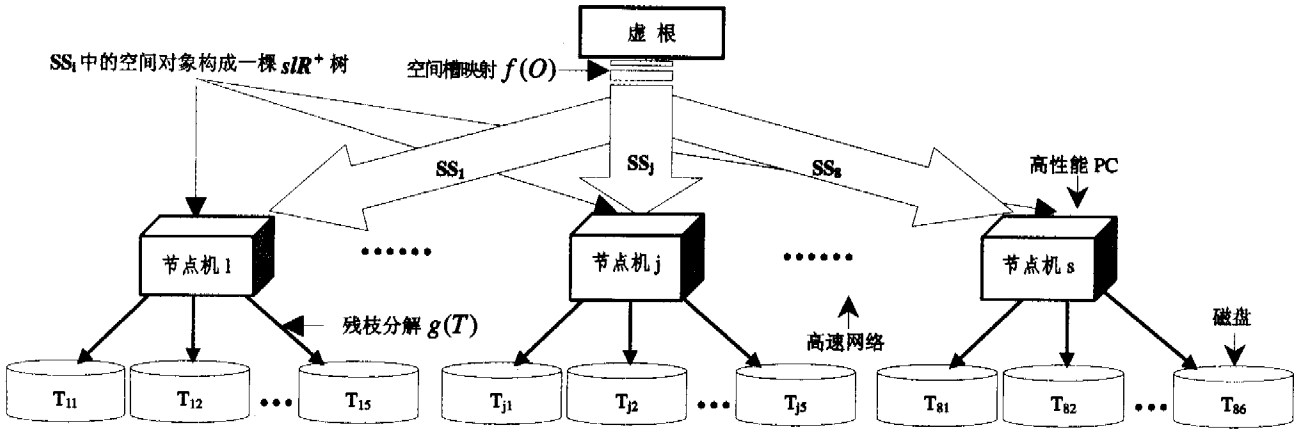


图 2 (8,6)-*DPslR*$^+$树结构

4 DPslR$^+$树枯荣过程

DPslR$^+$树是一种动态空间索引结构，随着其中空间对象的增减，分布式并行空间索引子系统将根据预定算法，触发繁衍过程或返本过程，动态维护 DPslR$^+$树，使其整体性能达到最佳或准最佳。DPslR$^+$树的状态转换如图 3，其中($1\leqslant p\leqslant e, k\leqslant t\leqslant u$)。

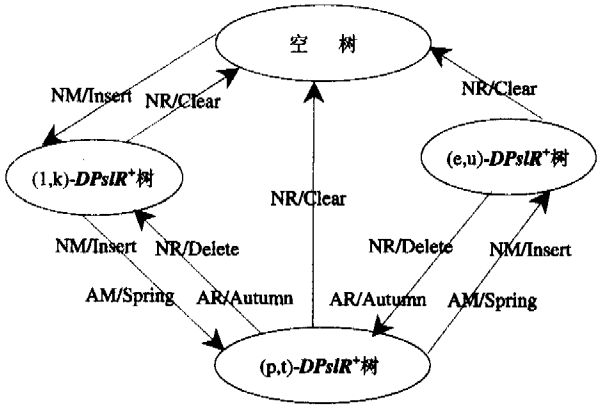


图 3 DPslR$^+$树的状态转换图

图中，NM(Natural Multiply)表示自然繁衍过程，由插入空间对象(Insert)触发；AM(Artificial Multiply)表示人工繁衍过程，由定时计算出的繁衍因子(Spring)触发；NR(Natural Reversion)表示自然返祖过程，由清除所有空间对象(Clear)或删除空间对象(Delete)触发；AR(Artificial Reversion)表示人工返祖过程，由返祖因子(Autumn)触发。

4.1 自然繁衍 NM

基本思想：当 DPslR$^+$树中尚无空间对象时，它实际上并不存在，此时可将其视为一棵空树。当插入一个或多个空间对象到 DPslR$^+$空树中时，用户所连接节点机将根据分布式并行空间数据库系统中各个节点机的数据负载状况和数据访问频度，选择适当节点机并在其上将 DPslR$^+$由空树转换为(1,k)-DPslR$^+$树(k 为节点机的本地磁盘数)。当插入空间对象到非空 DPslR$^+$树中时，则根据各个节点机的数据负载状况、数据访问频度和各个节点机的空间槽 SS，选择插入节点，将空间对象插入其中。若选择的节点机上尚无空间对象，则启动 NM。

算法描述：

```
if(Empty(RT))
{
j=SelectBestNode(Globaldataload,Globaldataaccess,SS,I); //根据全局数据负载、全局数据访问频度、DPslR+树全局空间槽、各个节点机的空间槽与空间对象的交度，选择最佳节点机。
  if(j=LocalNode)
  {
    Lds=GetLocalDiskSum(); //获取本地磁盘数
        RT=InitDPslR+(1,Lds);//初始化一棵 DPslR+树，包括通告其余节点机该树的状态。
        While ((SO=ReadSpatialObject)! =NULL) //读入一个空间对象
        LocalInsert(RT,SO);//本地插入该空间对象
        }
        Else
        {
        Rds=GetRemoteDiskSum(j); //获取节点 j 的磁盘数
        RT= InitDPslR+(1,Rds);//初始化一棵 DPslR+树，包括通告其余节点机该树的状态。
            While ((SO=ReadSpatialObject)! =NULL) //读入一个空间对象
          RemoteInsert(RT,SO,j);//远地插入该空间对象
        }
  }
  else
  {
    j=SelectBestNode(Globaldataload,Globaldataaccess,SS,I); //根据全局数据负载、全局数据访问频度、DPslR+树全局空间槽，空间对象与空间槽的交度，选择最佳节点机。
    If(HaveSo(j))
    InsertSo(j);//节点机上已有空间对象，则插入空间对象。
    else
      NMInsertSo(j);//启动 NM 插入空间对象
}
return;
```

4.2 人工繁衍 AM

基本思想：为了提高 DPslR$^+$树的查询性能，定时计算繁衍因子 S_{pring}。当 S_{pring}小于预定阈值 $\delta(\delta<0)$时，启动人工繁衍过程：从节点机 i 中读出 ΔS 个空间对象，从节点机上的 lsR$^+$树中删除 ΔS 个空间对象，将其插入节点机 j 中。繁衍因子 S_{pring}的计算如下：设 DPslR$^+$树中共有 S 个空间对象，它们被分布在 C 个节点机中，节点机 i 中的空间对象数为 S_i($S_i\geqslant 0$)；用户以等概率访问每个空间对象；用户对该空间对象集的访问按参数为 λ 的 Poisson 流到达；用户对节点机 i 中空间对象集的访问按参数为 $P_i\lambda$($P_i=S_i/S$)的 Poisson 流到达；节点机 i 的对空间对象访问的处理时间服从参数为 μ_i 的指数分布。节点机 i 对用户空间数据访问构成一个 $M/M/1/\infty$排队系统。因此访问一个空间对象的平均时间 $\overline{T_i}=1/(\mu_i-\lambda P_i)=1/(\mu_i-\lambda S_i/S)$。若从节点机 i 分裂出 ΔS 个空间对象到节点机 j 中，则访问一个空间对象的平均时间为 $\overline{T_j}=1(\mu_j-(\lambda S_j+\Delta S)/S)$，而节点机 i 中访问一个空间对象的平均时间 $\overline{T_i}=1/(\mu_j-(\lambda S_j+\Delta S)/S)$。同时，从节点机 i 分裂出 ΔS 个空间对象到节点机 j 中，所需的时间 $T_d=R_{\Delta s}+T_{\Delta S}+W_{\Delta S}+I_{\Delta S}$。其中，$R_{\Delta S}$为读出 ΔS 个空间对象所需时间，$T_{\Delta S}$为传输

ΔS 个空间对象所需时间，$W_{\Delta S}$ 为写入 ΔS 个空间对象所需时间，$I_{\Delta S}$ 为插入 ΔS 个空间对象到树中所需时间。

$$S_{pring}=\overline{T_i}-\overline{T_j}-T_d=\frac{1}{\mu_i-(\lambda\Sigma_i-\Delta S)/S}-\frac{1}{\mu_j-(\lambda S_j+\Delta S)/S}-(R_{\Delta S}+T_{\Delta S}+W_{\Delta S}+I_{\Delta S})$$

算法描述：

```
While(1)
{
    WaitTimeOut();//等待计时器超时
    Spring=CalculateSpring(Target,P);//计算繁衍因子，Target
    为目标节点机，P 本次 AM 操作的空间对象。
    If(Spring<δ)
    {
        NoticeOthers();//通告其他节点机本节点机将进行
                        AM
        If(HaveNodeDoing())
            Continue;//其他节点机正进行 AM，则本节点机
                     停止 AM。
        else
        {
            DeleteSOs(P);//删除本节点机中的 P 个空间对象
            AdjustNodeSS(LocalNode);//调整本节点机的空
                                    间槽 SSLocal
            RemoteInsertSOs(P,Target);//将 P 个空间对象
                                远程插入目标节点机 Target 中
            AdjustNodeSS(Target);//调整节点机 Target 的
                                空间槽 SSTarget
            }
        }
    }
```

4.3 自然返祖 NR

当从 DPslR$^+$ 树中删除空间对象(Delete)，使得某个节点机中的空间对象为空时，或清除(Clear)DPslR$^+$ 树中的所有空间对象时，启动自然返祖 NR：将 DPslR$^+$ 树相应的空间槽置为空，并调整 DPslR$^+$ 树虚根的 xMBR。若 DPslR$^+$ 树虚根的 xMBR 为空，则将 DPslR$^+$ 树转换为空树。

4.4 人工返祖 AR

基本思想：当节点机 i 对应空间槽 SS_i 的访问频度 A_{utumn} 低于预定阈值 $\xi(\xi>0)$ 时，根据全局数据负载、全局数据访问频度、其他节点机的空间槽 SS 和节点机 i 的空间槽 SS_i 之交度，选择最佳节点机 j，将节点机中的空间对象插入节点机 j 中。

算法描述：

```
While(1)
{
    WaitTimeOut();
    If(Autumn<ξ)
    {
        j=SelectBestNode(Globaldataload,Globaldataaccess,
          SS,I); //根据全局数据负载、全局数据访问频度、
          DPslR+树全局空间槽、各个节点机的空间槽与空
          间槽的交度，选择最佳节点机。
        NoticeOtherAR();//通告其余节点机
        If(j! =Local and HaveOtherDoing()=False) //无其
       他节点进行 AR 并且最佳节点机不是本节点机，进行 AR
        {
            Q=DeleteAllSO();//删除本节点机中的所有空间
                            对象
            RemoteInsertSOs(Q,j);//将 Q 个空间对象远程
                                 插入目标节点机 j 中
            AdjustNodeSS(j);//调整节点机 j 的空间槽 SSj
            AdjustVRootSS();//调整 DPslR+ 树的空间槽
            }
        }
    }
```

5 DPslR$^+$ 性能分析

DPslR$^+$ 树与文献中的 GPR 树均采用同一研究思路来提高空间查询的性能，因此将 DPslR$^+$ 树与 GPR 树的查询性能进行分析对比。首先做如下记号：

h 表示从树根到空间对象所在叶子的路径上的内部节点树；q 表示在 DPslR$^+$ 树中空间对象被映射到非用户连接节点上的概率；$h_1(h_1\leqslant h)$ 表示 DPslR$^+$ 树中不在用户连接的节点机上的内部节点数；$h_2(h_2\leqslant h)$ 表示在 DPslR$^+$ 树和 GPR 树中从根到所搜索叶子经历的路径中，不在内存中的节点数；$\overline{T_{cm}}$ 表示一次通信开销；$\overline{T_{rd1}}$ 表示 GPR 树中读一次磁盘所需时间；$\overline{T_{rd2}}$ 表示 DPslR$^+$ 树中一次读磁盘的等效时间；$\overline{T_f}$ 表示 DPslR$^+$ 树中一次空间映射所需时间；$\overline{T_g}$p表示 GPR 树中一次空间对象访问所需时间；$\overline{\mu_{gp}}$表示 GPR 树中访问空间对象的等效速度；$\overline{T_{dp}}$表示 DPslR$^+$ 树中一次空间对象访问所需时间；$\overline{\mu_{dp}}$表示 GPR 树中访问空间对象的等效速度；S_p 表示加速比。

$$\overline{T_{gp}}=h_1\cdot\overline{T_{cm}}+h_2\cdot\overline{T_{rd1}}+\overline{T_{rd1}},\overline{\mu_{gp}}=1/\overline{T_{gp}}$$

$$\overline{T_{dp}}=T_f+q\cdot\overline{T_{cm}}+h_2\cdot\overline{T_{rd2}}+\overline{T_{rd}},\overline{\mu_{dp}}=1/\overline{T_{dp}}$$

DPslR$^+$ 树采用 MultiDisk 进行并行 I/O。

$\overline{T_{rd1}}=\alpha\cdot\overline{T_{rd2}}(\alpha>1)$。

$\overline{T_f}$为空间映射所需计算时间，远小于通信时间和读磁盘时间，所以

$$\overline{T_{cm}}=\beta\cdot\overline{T_f},\overline{T_{rd2}}=\gamma\cdot T_f(\beta\gg1,\gamma\gg1)$$

$$S_p=\overline{\mu_{dp}}/\overline{\mu_{gp}}=(h_1\cdot\beta+h_2\cdot\alpha\cdot\gamma+\alpha\cdot\gamma)/(1+q\cdot\beta+h_2\cdot\gamma+\gamma)\approx(h_1\cdot\beta+h_2\cdot\alpha\cdot\gamma+\alpha\cdot\gamma)/(q\cdot\beta+h_2\cdot\gamma+\gamma)$$

①当 $h_1=0,q=0$ 时，

$S_p=\frac{h_2\alpha+\alpha}{h_2+1}>1(\alpha>1)$，DPslR$^+$ 树的空间对象查询速度比 GPR 树快，随着 α 增大，其效率提高更明显。

②当 $h_1=0,q=0,h_2=0$ 时，

$S_p=\frac{h_2\alpha+\alpha}{h_2+1}=\alpha$，DPslR$^+$ 树的空间对象查询速度是 GPR 树空间对象查询速度的 α 倍，随着 α 增大，其效率提高更明显。

③当 $q=h_1/\gamma$ 时，

$$S_p=\frac{q\beta+h_2\alpha+\alpha}{q\beta/\gamma+h_2+1}>\frac{q\beta+h_2\alpha+\alpha}{q\beta+h_2+1}>1\quad(\alpha>1),$$

DPslR$^+$ 树的空间对象查询速度比 GPR 树快，随着 α 增大，其效率提高更明显。

④当 $\alpha=1,h_1>q$ 时，

$S_p=\frac{h_1\beta/\gamma+h_2+1}{q\beta/\gamma+h_2+1}>1(\alpha>1)$，DPslR$^+$ 树的空间对象查询速度比 GPR 树快，随着 α 增大，其效率提高更明显。

⑤当 $\alpha=1,h_1<q$ 时，

$S_p=\frac{h_1\beta/\gamma+h_2+1}{q\beta/\gamma+h_2+1}<1(\alpha>1)$，DPslR$^+$ 树的空间对象查询速度比 GPR 树慢。

⑥当 $\alpha=1,h_1=q$ 时，

$S_p=\frac{h_1\beta/\gamma+h_2+1}{q\beta/\gamma+h_2+1}=1(\alpha>1)$，DPslR$^+$ 树的空间对象查询速度与 GPR 树相等。

6 试验结果

为测试 DPslR$^+$ 树与 GPR 树对空间查询性能的优化情况，在此采用 C^{++} 语言实现分布式并行空间索引子系统。在保持对外接口不变的情况下，通过函数重载方式实现 DPslR$^+$ 树与 GPR 树。测试前，系统中已存在 10000 个二维空间对象(多边形)和 10000 个三维空间对象(多面体)。空间查询请求模拟 Possoin 流到达，空间查询含插入空间对象、删除空间对象、更新空间对象属性和搜索空间对象属性。测试

时，首先记录 DPslR$^+$树中每个空间查询的时间，计算空间查询的平均时间；之后，改变宏定义，启动 GPR 树并记录 GPR 树中每个空间查询的时间，计算空间查询的平均时间。空间查询代码亦采用 C^{++}语言编写。具体测试环境如下：

①硬件环境：10 台 $P_4$2G，512M 内存的普通 PC 作为服务器，每个服务器配置 3 个 7200RPM 硬盘；10 台 PIII 550，128M 内存的普通 PC 作为查询客户机；100M 交换机和 100M 网卡。

②软件环境：服务器的编译器为 gcc；服务器的 OS 为 Redhat Linux 7.3；查询客户机的 OS，5 台为 Redhat Linux7.3，5 台为 Windows2000；空间查询程序在 Linux 下的编译器为 gcc，Windows 下的编译器为 Visual C^{++}。

测试结果如图 4 和图 5 所示。

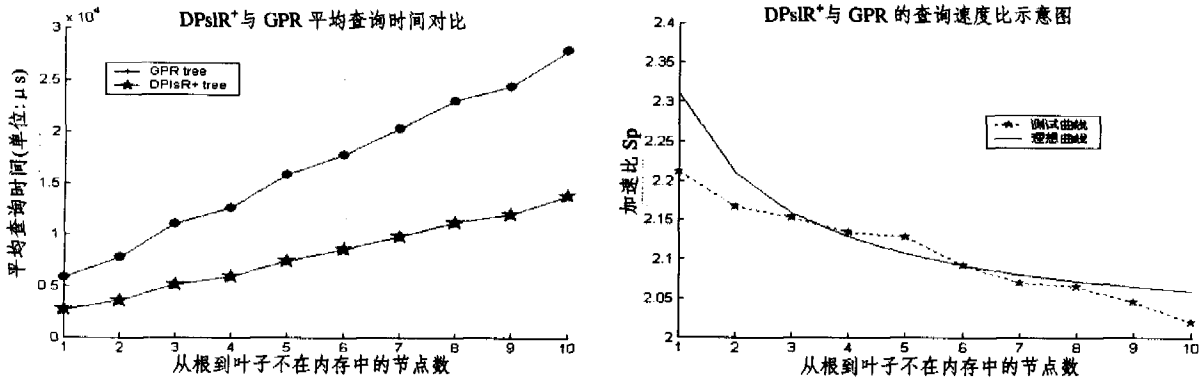
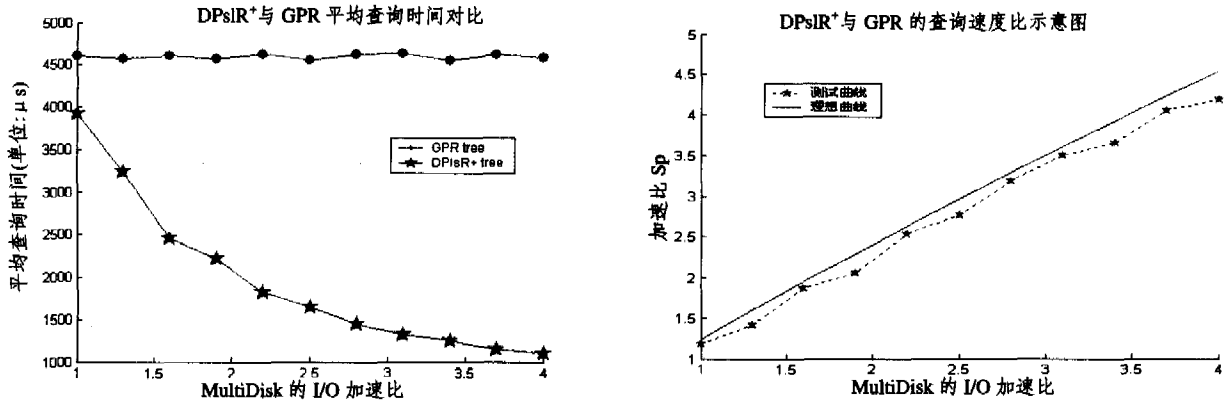


图 4　外存节点变化时 DPslR$^+$树与 DPR 树的查询性能



(a)　DPslR$^+$树与 DPR 树的平均查询时间对比图　　(b)　DPslR$^+$树与 DPR 树的查询加速比示意图

图 5　DPslR$^+$树的 MulitDisk 性能变化时 DPslR$^+$树与 DPR 树的查询性能

从图 4 可见，随着树中外存节点数的增加，DPslR$^+$树的查询性能均优于 GPR 树的查询性能；但查询加速比随着外存节点数的增加逐渐降低，并最终趋于定值。

从图 5 可见，随着 DPslR$^+$树 MultiDisk 的加速比提高，DPslR$^+$树的查询性能几乎线性改善，而 GPR 树的查询性能几乎保持恒定；并且随着 MultiDisk 加速比的提高，树的查询加速比几乎线性提高。

结论　空间查询速度是空间数据库发展应用的关键。空间索引对空间查询的影响极大。将空间索引与分布并行处理技术融合在一起，并借助于 MultiDisk 提高 I/O 性能，从而实现分布式并行空间索引树 DPslR$^+$树。理论分析和实验结果均表明，DPslR$^+$树具有良好的查询性能。通过实验测试表明，就二维空间对象和三维空间对象而言，DPslR$^+$树具有良好的查询性能。但对于四维以上的空间对象，如何划分 DPslR$^+$树中的空间槽以及繁衍过程需进行进一步研究。

参 考 文 献

1　Guttman A. R-Trees: A Dynamic Index Structure for Spatial Searching. In: 1984 Proc. ACM SIGMOD Conf., 1984. 47～57

2　Ciferri R R, Salgado A C, et al. A performance comparison among the traditional R-trees, the hilbert R-tree and the SR-tree. In: 2003 Proc. SCCC Conf., 2003. 3～12

3　Xia Yuni, Prabhakar S. Q+Rtree: efficient indexing for moving object databases. In: 2003 Proc. DASFAA Conf., 2003. 175～182

4　Hu Kong-Fa, Dong Yi-Sheng, et al. DCA-Tree: a high performance structure for incremental update cube on MDDW. In: 2002 Proc. Machine Learning and Cybernetics Conf., 2002. 2069～2072

5　Tao Yufe, Papadias D. Efficient historical R-trees. In: 2001 Proc. SSDBM Conf., 2001. 223～232

6　Ross K A, Sitzmann I, et al. Cost-based unbalanced R-trees. In: 2001 Proc. SSDBM Conf., 2001. 03～212

7　Lo Ming-Ling, Ravishankar C V. The design and implementation of seeded trees: an efficient method for spatial joins. Knowledge and Data Engineering, IEEE Transactions on, 1998, 10(1): 136～152

8　Mamoulis N, Papadias D. Slot index spatial join. Knowledge and Data Engineering, IEEE Transactions on, 2003, 15(1): 211～231

9 Jurgens M , Lenz H-J. The R* -tree: an improved R* -tree with materialized data for supporting range queries on OLAP-data. In: 1998 Proc. Database and Expert Systems Applications Conf. , 1998. 186～191

10 Kanth K V R, Agrawal D, et al. Index non-uniform spatial data. In: 1997 Proc. Database Engineering and Applications Symposium Conf. , 1997. 289～298

11 Sozer A, Yazici A. Access structures for fuzzy spatial queries. In: 2002 Proc. Fuzzy Information Processing Society, 2002. 383～388

12 Kwon Dongseop, Lee Sangjun, et al. Index the current positions of moving objects using the lazy-update R-tree. In: 2002 Proc. Mobile Data Management Conf. , 2002. 113～120

13 Shekhar S, Chawla S, et al. Spatial databases—accomplishments and research needs. Knowledge and Data Engineering, IEEE Transactions on , 1999, 11:45～55

14 Gunther O, Noltemeier H. Spatial database indices for large extended objects. In:1991 Proc. Data Engineering Conf. , 1991. 520～526

15 Fu Xiaodong, Wang Dingxing, et al. GPR-Tree: a global parallel index structure for multiattribute declustering on cluster of workstations. In:1997 Proc Advances in Parallel and Distributed Computing Conf. , 1997. 300～306

16 Schnitzer B, Leutenegger S T. Master-client R-trees: a new parallel R-tree architecture. In: 1999 Proc. Scientific and Statistical Database Management Conf. , 1999. 68～77

17 Xiao Weiqi, Feng Yucai. Parallel Ladex spatial index mechanism. In:Geoscience and Remote Sensing, 1997. IGARSS '97. Remote Sensing - A Scientific Vision for Sustainable Development, 1997 IEEE International , 1997, 1: 216～218

(上接第 66 页)

的波动幅度要明显低于常规 PID 控制器控制的队列长度。在不同的参数组合控制下,控制效果如图 4(d),(e)所示。由仿真结果可见,量化因子,时延等级,可用带宽波动幅度对队列长度均有影响。总体上讲,可用带宽波动越强烈越易使缓冲区队列长度进入抽空或饱和非线性区,导致带宽利用率降低,信元丢失率升高。回路时延始终是恶化网络性能的重要因素,按最大时延设计控制器尽管可以保证系统的稳定性,但具有一定的保守性。

结论 本文针对单瓶颈节点网络模型,设计模糊拥塞控制器。在式(1)～(3)两个非线性环节作用下,以交换机队列长度为控制目标,假设最大传输时延可估计的情况下,考虑存在时延抖动,可用带宽大幅波动时,设计控制器,保证网络 QoS。分别选择不同的控制器参数组合,仿真结果验证了方法的有效性。但是按照最大回路时延设计控制器,会给系统带来一定的保守性;模糊规则的制定依赖于先验知识。如何降低系统保守性,寻求控制策略的简单性和有效性的折中是需要进一步讨论的问题。

参 考 文 献

1 Bonomi F, Fendick K W. The rate-based flow control framework for the available bit rate ATM service. IEEE J. Select. Areas commun, 1995, 13(7): 1267～1283

2 Chong S, Lee S, Kang S. A simple , scalable, and stable explicit rate allocation algorithm for MAX-MIN flow control with minimum rate guarantee. IEEE/ACM Transaction on Networking, 2001, 19(3): 322～335

3 Kalyanaranman S, Jain R, Fahmy S, Goyal R, Vandalore B. The ERICA switch algorithm for ABR traffic management in ATM networks. IEEE/ACM Transaction on networking, 2002, 8(1): 87～98

4 Pitsillides A, Lambert J. Adaptive Congestion Control in ATM Base Networks: Quality of Service and High Utilization. Computer Communications, 1997(20): 1239～1258

5 Quet P F, Ramakrishnan S, Ozbay H. On theController Design for Congestion Control with a Capacity Predictor. In: Proc. of the Conf. on Design and Control, Orlando, FL, 2001. 598～603

6 Fei X, He X. Fuzzy Neural Network Based Traffic Prediction and Congestion Control in High-Speed Networks. J. Comput. Sci & Technol, 2000(15): 144～149

表 2 仿真参数

	图 b	图 c	图 d
期望队列	500(cell)		
MCR, PCR 值	$MCR=0, PCR=3.3\times10^4$		
最大时延	$\tau_{1\max}=100\text{ms}, \tau_{2\max}=30\text{ms}, \tau_{3\max}=15\text{ms}$		
量化因子	$\delta_e=0.0012$ $\delta_{\dot{e}}=0.001$ $\delta_{K_I}=500$ $\delta_{K_d}=0.08$		
	$\delta_{K_p}=300$ $\delta_r=0.4$	$\delta_{K_p}=200$ $\delta_r=0.6$	$\delta_{K_p}=300$ $\delta_r=0.4$
可用带宽 (cell/s)	$N\begin{pmatrix}45000,\\ 2000^2\end{pmatrix}$	$N\begin{pmatrix}45000,\\ 1000^2\end{pmatrix}$	$N\begin{pmatrix}45000,\\ 6000^2\end{pmatrix}$

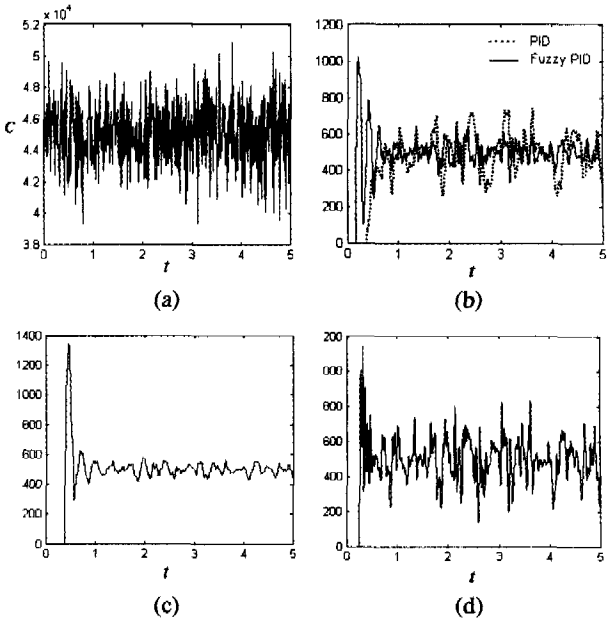


图 4 仿真结果图