文章编号：1007-4619 (2001) 04-0267-10

Internet GIS 上矢量型空间数据传送的最优化策略

魏祖宽，裴海英
(韩国 仁川市 仁荷大学校 电子计算工学科)

摘 要： 作为在 Internet 上处理大容量空间地理信息的基于客户端的 Internet 地理信息系统(client-side Internet GIS)，解决矢量型空间数据传送的低效率性是很重要的。该文为此提出一个新的有效方法。首先，将大型矢量地图根据适当的“分裂度”分割成若干部分，每一部分称作一个“块”(Tile)；然后，当用户在客户端请求地图的某一区域时，GIS 服务器只传送和请求区域相交叠的“块”内的空间数据，并且将该数据存在用户的本地机上，以便再用。为此，该文提出一个“地图块分割”方法，“地图块查询处理”算法以及“客户服务器间空间数据的效率化传送”策略。和既存传送方法比较，利用该方法构造的 Internet GIS，系统的整体效率得到了提高。

关键词： 空间数据传送；地理信息系统；Internet GIS；空间数据库
中图分类号： TN919 **文献标识码**： A

1 引 言

空间数据可以以栅格(raster)和矢量(vector)两种方式进行表现，前者由于在查询定位和空间分析方面天生的缺陷以及相对于矢量方式的数据量小，输入方便等优点，主要用在空间地理信息的输入和相对固定的空间地理信息的传送和显示上(如用户查询结果)[1—4]。现在大部分 GIS 系统采用矢量方式进行空间地理信息的存储和组织管理，以满足越来越复杂的查询处理需要。这在需要完整空间分析功能的高级 GIS 和边缘 GIS(如空间决策支持系统 SSDS 等)中尤其如此[5,6]。

全球性的计算机网络，Internet 的出现很大程度上改变了传统的数据访问、数据分发方式。作为在 Internet 上处理空间数据和地理信息的 Internet 地理信息系统，随着 Internet 和 Web 技术的发展也在迅速地发展着[7—12]。基于服务器端的 Internet 地理信息系统和基于客户端的 Internet 地理信息系统是目前 Internet 地理信息系统的主要两类。前者，服务器端处理客户端的查询请求，并将结果数据传送客户端。为保证数据传输的快速性和减小客户端软件的规模，大部分系统的服务器端将矢量型结果数据变换成栅格数据形式，主要是栅格格式的媒介数据，如 GIF、JPEG 图形等进行传送。这样，服务器端处理矢量数据和栅格数据，客户端只处理服务器传来的查询结果—栅格数据。这类系统主要用在使用者要求简单的查询，指南等应用中。

在基于服务器端的 Internet 地理信息系统中，由于处理集中在服务器端，当客户端使用者增加时，服务器端系统负荷也迅速增加，系统性能也随之迅速下降。另外，随着 GIS 使用的深化和 Internet 连结速度的相对提高，要求客户端提供如空间分析信息等高交互性信息的要求也随着一步步的增加。而在基于服务器端的 Internet 地理信息系统中，单纯显示栅格地图数据的客户端已不能完全满足此类要求。“在客户端矢量空间数据的处理能力”要求被提出，这直接导致了基于客户端的 Internet 地理信息系统的出世，关于基于客户端 Internet 地理信息系统的研究正在迅速地进行[7,10,13,14]。

但是，由于大容量数据的传送，以及 Internet 上数据传送速度和计算机本地总线速度的巨大差异等原因，Internet GIS 有一大弱点，就是数据传送的低效率性。这个问题在传统的桌面 GIS 中通常不被考虑。但在 Internet GIS 中，却需要大量的时间来下载和初始化 GIS 控件(Plug In，ActiveX control，Java Ap-

收稿日期：2000-11-06；**修订日期**：2001-01-15
基金课题：韩国情报通信部“大学软件研究中心支援基金。”

作者简介：魏祖宽(1968—)，男，1990 年浙江大学计算机系工学学士，1997 年重庆邮电学院计算机系工学硕士，1997 年至今韩国仁荷大学校电子计算机学科博士研究生。研究领域：地理信息系统(GIS)，分布式数据库系统，空间数据库，Java 应用技术等。已发表学术论文 6 篇。

plet 等)以及地图数据。特别是在基于客户端的 Internet 地理信息系统中,由于"客户端矢量空间数据的处理能力"要求导致的大容量矢量空间数据的传送,该问题需要特别考虑。

一般地,Internet GIS 的低效率性可用下述两方法解决:(1) 提高 Internet 的连接速度;(2) 设计更高效的程序。关于前者,Internet 的下一代通信标准,Internet 2 正在完善成型中,它的使用将会大大地提高 Internet 的连接速度;关于后者,高效程序的设计正沿着构件结构设计、即用即插机制(just-in-time)的研究方向前进[15]。关于急切需要考虑的基于客户端 Internet 地理信息系统中的大容量矢量空间数据的传送低效率性问题,本文基于即用即插机制的原理,提出一种方法,在基于客户端 Internet 地理信息系统中,使服务器只在必要的时候传送必要的矢量空间数据,从而提高系统的效率。

2 既存的基于客户端的 Internet GIS

为了提供高交互性的地理信息和空间分析功能,最近提出了"基于客户端的 Internet 地理信息系统",它的一大特征是不仅在客户端处理诸如地图"放大"、"缩小"及"移动"等用户查询任务。而且,诸如"计算区域面积"、"计算空间客体间的最短路径"等空间分析功能也可以在客户端进行处理。这意味着要求客户端不仅要像基于服务器端的 Internet 地理信息系统那样提供处理服务器传来的一般查询结果-栅格数据的能力,而且需要提供接收服务器传来的原始矢量空间数据,并在此基础上进行查询和空间分析处理的能力。

通常"基于客户端的 Internet 地理信息系统"采用 Plug-In,Java Applets, ActiveX 控件技术等进行开发。GIS Plug-In 是预先安装在客户机上用 MIME 格式处理地理数据的一种控件。浏览器利用它的捕捉机制来处理服务器端传来的地理数据。目前有 MapGuide[16],GeoMedia Web Map[17]等产品。

GIS Java Applets 是用网络程序语言 Java 设计的一种控件,它通过本地 Java 虚拟机解释执行机制解释服务器传来的地理数据和执行空间运算等功能。除了它的高移植性外,在处理地理信息时,Java Applets 还具有很强的动态性和程序强壮性。但是,由于 Java 的安全机制,Java Applets 不能存储地理信息、分析结果等数据在本机;而且,Java Applets 除了其来源服务器外,不能连接别的服务器,这对分布式数据库系统的应用也不利。目前 Java Applets GIS 有 DDViewer[18]、IDGIS[17]及 ActiveMaps[19]等产品。

最近,用于开发 Internet GIS 的技术是 ActiveX 控件技术,它在 Internet 上通过 COM(OLE)技术机制执行 GIS 作业任务。在客户端,ActiveX 控件和 Web 浏览器可以实现"无缝连接"。作为 ActiveX 控件,其最大的优点是可以使用本机包括硬盘在内的所有资源。这对为提高地理数据的再用而在本机建立数据缓冲区有很重要的作用。用 ActiveX 控件技术开发的 Internet GIS 产品有 MapObject[14], Mapcafe[7], WinFrame Web[13]以及 GEOWeb[10,12]等。

3 地图的层次表现方法

GIS 中空间数据的表现有许多方法,诸如基于空间层次的表现方法、基于空间特征的表现方法等[20—22]。在基于空间层次的表现方法中,空间数据被组织成一系列的主题地图,称作"层"(layer),每一"层"地图表示一个特定的主题,如"道路"、"建筑"、"水文线"、"等高线"、"轮廓线"等。作为一个例子,图 1 即为一用"层"概念组织的地图。

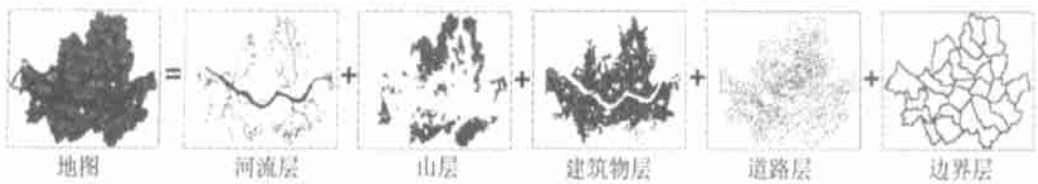

图 1 由若干主题层组成的地图例

Fig. 1 A Sample of Map Composed of Several Thematic Layers

"基于空间层次的表现方法"具有下列的优点:

(1) 查询和空间分析处理容易化

在空间数据库中,空间实体的重叠(或部分重叠)是一个需要特别考虑的问题。在处理点查询、区域查询、空间信息分析等时,空间实体的重叠(或部分重叠)使处理复杂化。而采用"层"概念组织空间

数据则可很大程度上避免这个问题。原因很简单：现实生活中同一主题的空间客体间很少出现重叠。

(2) 数据分块传送的管理效率化

当把地图表现在计算机输出设备(如计算机显示器)上时，从用户的观察立场和实际显示效果来看，是不需要显示地图的所有细节的(如同在月球上观察地球，不能看见具体的建筑物一样)。因此，“显示分辨率控制”被用在地图的显示上：只显示能显示的空间实体。这里，地图的层次表现方法提供了一个将空间实体按分辨率组织的方法。实际上，许多现有的桌面 GIS 和 Internet GIS 软件在地图的组织上均采用了这一概念。

由于这些优点，本文在“客户端 Internet 地理信息系统”中，基于“层次表现的空间数据”上，提出一种空间数据的效率性传送方法。

4　地图的块分割

4.1　术语定义

如前所述，为了 Internet 上空间数据的效率性传送，我们将含有大量空间实体的“地图层”用一定大小的矩形分割成若干部分，分块传送。这里，每一部分称作一个“块”(Tile)；块的轮廓矩形则称作“块矩形”(Tile Rectangle，TR)。当地图层内的某个空间实体位于某个块内或和其交叠时，我们称“该块包含该空间实体”或“该空间实体位于该块内”。某一“块”内所有的空间实体构成该块的“空间实体集”，这种“空间实体集”可能含有 0 至 n 个空间实体。“空间实体集”中没有空间实体成员的块称作“空白块”(Blank Tile，BT)，其余的称作“一般块”(General Tile，GT)。对应地，“空块”和“一般块”的轮廓矩形则称作“空白块矩形”(BTR)和“一般块矩形”(GTR)。另外，某一块内所有空间实体的“最小边界矩形”(Minimum Bounding Rectangle，MBR)，称作该块的“块最小边界矩形”(Tile Minimum Bounding Rectangle，TMBR)。“块矩形”(TR)和“块最小边界矩形”(TMBR)的相互关系显示如图 2。

如两空间实体相互的拓扑关系一样，“块矩形”和“块最小边界矩形”的相互关系也可定义为分离(disjoint)、相连(meet)、完全重叠(equal)、相交(overlap)、包含/被含(cover/covered-by)、完全包含/完全被含(contain/inside)等。但对本课题，只有下述四种关系是有意义的：

(1) 块矩形“位于”块最小边界矩形内；

(2) 块最小边界矩形“位于”块矩形内；

(3) 块矩形和块最小边界矩形“相交”；

(4) 无块最小边界矩形(块矩形是空块矩形 BTR)。

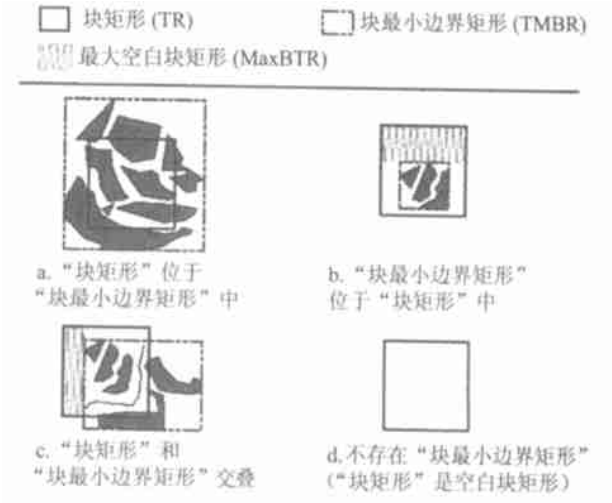

图 2　“块矩形”和“块最小边界矩形”的关系

Fig. 2　The Relation of TR and TMBR

在(2)、(3)和(4)(其中(4)是一特殊关系)的情况下，块内存在空白区域(死空间)，块内空白区域能构成的最大矩形称为“块内最大空白区域”，可以表示为“MaxBTR(TR，TMBR)”。

对一个地图层的某一块(假设为块 A)，如果它需要继续分块，则块 A 称作该地图层的“中间块”(Medial Tile，MT)；这种分割称作“块的再分割”。“块的再分割”需要考虑两个因素：分割方向(Division Direction，DD)和分割点(Division Point，DP)。

分割方向决定块 A 的再分割方向，可为垂直和水平方向，根据块 A 的块矩形长和宽的长度决定(即块矩形的水平边比垂直边长，则分割方向为垂直方向；反之则为水平方向)。当分割方向定后，块 A 的左边部分(分割方向为垂直方向时)或块 A 的上边部分(分割方向为水平方向时)称作块 A 的“第一子块”，另一部分即为“第二子块”。

分割点决定块 A 再分割子块的大小。它的值定义为块 A 的“第一子块”和块 A 的面积之比，也可看作块 A 分割边上“第一子块”所占的长度和分割边长度之比。

“块的再分割”、“分割方向”和“分割点”的概念示意如图 3。

4.2　地图层的块分割

地图层的块分割通过建立一个名叫“Tile-Tree”

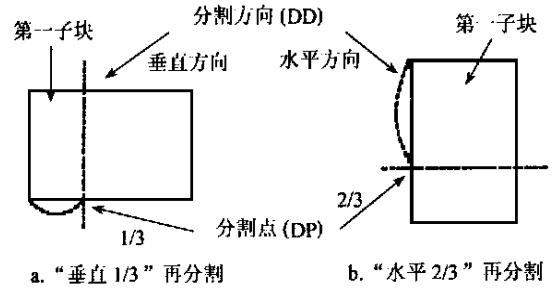

图3 块的再分割

Fig. 3 The split of tile

的过程来完成。基本处理过程是将地图块(原始的地图块即地图层)按照一定的分割方向和分割点分成两个子块,这样新的两个子块就产生了,然后再在新产生的子块上递归重复以上过程。直至满足相应的递归结束条件。

块分割时分割方向和分割点的决定策略是,首先考察目标块的"块内最大空白区域"。如它的面积超过目标块面积的10%,则据此确定分割方向和分割点,使分割成的其中一个子块即为该空白区域块。否则,根据目标块的长宽比例确定分割方向,然后对分。

块递归分割的过程受控于一个称作"分裂度"的参数 g,它定义地图层的分割块数,即目标块的面积大于"地图层"面积的 $1/2^g$ 时,可再分割;否则,它即最终块,除非它含有大于它面积50%的"块内最大空白区域"。这时,它可进行最后一次分割,以剔除掉该空白块。以上综合即构成递归结束条件。

在以上递归分割过程中,Tile-Tree 也随之建立。Tile-Tree 是一有序二叉树,根结点指向地图层;中间结点指向中间块,叶结点指向最终块。其中,对根结点和任一中间结点,它的左结点总是指向该块分割成的,"第一子块"右结点指向"第二子块"。

下面是地图层块分割的具体算法"TileDivide",它产生目标地图层的 Tile-Tree:

算法1: **TileDivide**(***pT***, ***tR***, ***sOid***)

输入: *pT*: *Tile-Tree* 结点指针

tR: 需分割的地图层或需再分割的地图块

sOid: *tR* 内包含的空间实体集

输出: 新产生的 *pT* 指向的 *Tile-Tree*

01: if(*sOid*=∅)then/ * *tR* 是空白矩形 * /

02: 建立 *BT* 叶结点 *pQ*,联结至 *pT*

03: 返回

04: end if

05: if(*tR* 的面积≤(地图层 *LMBR* 面积/2^g))then

06: / * 地图块不需要再分割 * /

07: if(*tR* 内最大空白区域的面积≥(*tR* 面积 * 50%))then

08: / * *tR* 包含大块空白区域,进行最后一次分割 * /

09: 建立中间结点 *pQ*,联结至 *pT*

10: 将 *tR* 分割成最大空白区域 $tR1$ 和其余部分 $tR2$

11: / * 这里假设 $tR1$ 是 *MaxBTR*,并且是第一子块 * /

12: 对应 $tR1$, $tR2$ 将 *sOid* 分割成 *sOidl*(=∅)和 $sOid2$(=*sOid*)

13: 递归调用 TileDivide(*pQ*·*L*, $tR1$, *sOidl*)

14: 递归调用 TileDivide(*pQ*·*R*, $tR2$, $sOid2$)

15: else

16: 建立 *GT* 叶结点 *pQ*,联结至 *pT*

17: 返回

18: end if

19: else / * 地图块需要再分割 * /

20: if(*tR* 内最大空白区域的面积≥(*tR* 面积 * 10%))then

21: / * *tR* 包含大块空白区域以之分割 *tR* * /

22: 建立中间结点 *pQ*,联结点 *pT*

23: 将 *tR* 分割成最大空白区域 $tR2$ 和其余部分 $tR1$

24: / * 这里假设 $tR2$ 是 *MaxBTR*,并且是第二子块 * /

25: 对应 $tR1$, $tR2$ 将 *sOid* 分割成 *sOidl*(=*sOid*)和 $sOid2$(=∅)

26: 递归调用 TileDivide(*pQ*·*L*, $tR1$, $sOid1$)

27: 递归调用 TileDivide(*pQ*·*R*·$tR2$, $sOid2$)

28: else / * *tR* 是一般块,进行对分 * /

29: 根据 *tR* 的长宽长度比较计算分割方向(*DD*=*H* or *V*)

30: / * 分割点是中点, *DP*=1/2/ *

31: 建立中间结点 *pQ*,联结至 *pT*

32: 根据所得 *DD* 和 *DP* 将 *tR* 分割成 $tR1$ 和 $tR2$

33: / * 这里假设 $tR1$ 是第一子块 * /

34: 对应 $tR1$, $tR2$ 将 *sOid* 分割成 $sOid1$ 和 $sOid2$

35: 递归调用 TileDivide(*pQ*·*L*, $tR1$, $sOid1$)

36: 递归调用 TileDivide(*pQ*·*R*, $tR2$, $sOid2$)

37: end if

38: end if

算法1结束

作为例子说明,图4即为一个地图层的块分割

和相应产生的 Tile-Tree：

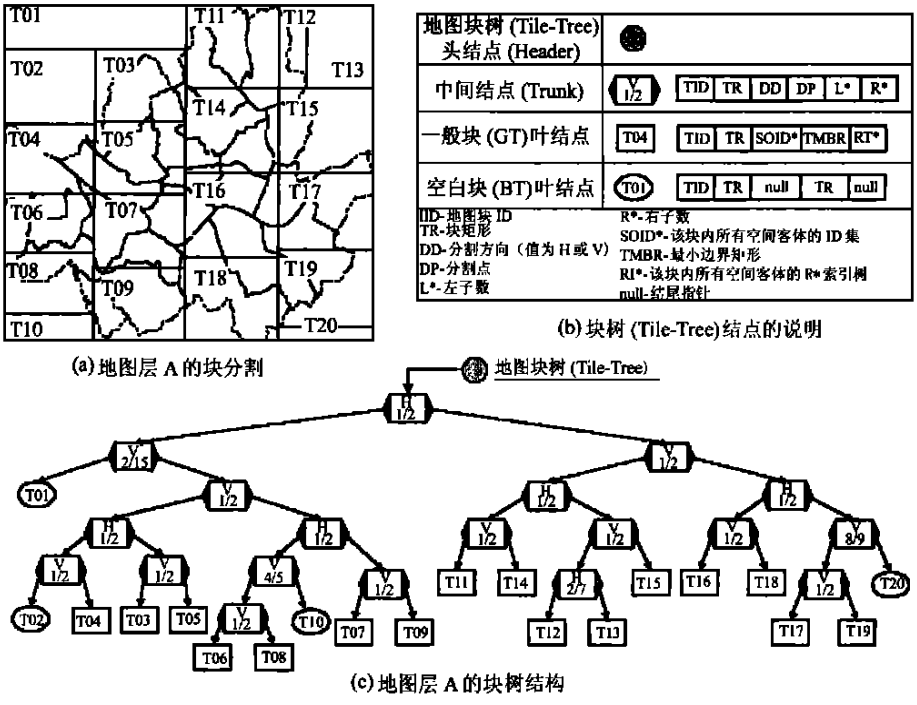

图 4　地图层的块分割及其 Tile-Tree 结构

Fig. 4　The Tile-division and tile-tree structure of layer

4.3　地图层块分割的扩展索引结构

在空间数据库中，对于空间数据的查询处理，空间索引结构是必需的[23—25]。众所周知，一系列基于空间实体 MBR 的空间索引结构已被提出，如 R-Tree、Cell-Tree、Grid-File 等[26—30]。其中最有前途的有 R-Tree 及其变形 R＋—Tree、R＊-Tree 等[27,31]。但是，这些索引结构是基于全体地图(层)建立的，没有考虑地图层的一部分(块)。另一方面，在基于客户端 Internet 地理信息系统中，由于客户端要执行空间查询处理，服务器端不仅要传送空间数据，也要传送这些空间数据的索引结构。如直接采用以上的空间索引结构，服务器端就需一次传送整个索引结构数据至客户端，这在整体上看是低效率的。因此，针对前面提出的块分割方法，本节提出相应的扩展索引结构，称作“两阶段索引结构”。说明如下：

通过节 4.2 算法“TileDivide”将地图层分成若干个块后，即可通过建立的 Tile-Tree 来管理整个地图层。客户端和服务器端的数据传送就可通过 Tile-Tree，以块为单位来进行。对于块内空间实体的查询，可就“块空间实体集”建立相应的 R＊-Tree 索引结构。服务器端传送块时，也传送相应块的 R＊-Tree 索引数据。这样，客户端就可以利用此索引数据在本机上进行块内查询处理。从整体来看，地图层的扩展索引结构由两阶段组成：(1) 高阶段是全体层的 Tile-Tree 结构，完成用户查询区域所关联地图块的查询任务；(2) 低阶段是各块的 R＊-Tree 索引结构，它完成用户查询区域在本块内所包含空间客体的查询任务。具体的扩展索引结构示意如图 5。

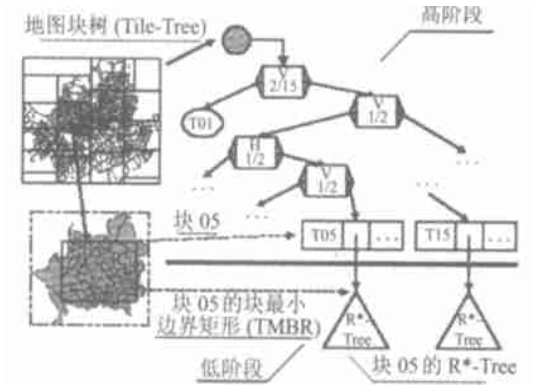

图 5　两阶段索引结构

Fig. 5　Two-Level indexing structure

5 数据传送策略

本节提出在基于客户端的 Internet 地理信息系统上空间数据的效率化传送策略。首先，给出用户查询区域的块定位算法；然后讨论空间数据效率化传送的具体步骤。

5.1 利用 Tile-Tree 查询地图块

对一用户查询(不管是点查询还是区域查询)，第一步就是找出包含目标点(对点查询)或目标区域(对区域查询)的地图块。在基于客户端的 Internet 地理信息系统上，当用户在客户端对某一地图层进行查询(假设是区域查询)时，首先就需要确定查询区域所包含的地图块，然后检查目标地图块是否位于客户机上。根据结果的不同决定在本机继续执行查询任务或向服务器请求相应的地图块数据。因此，“找出用户查询区域所包含的地图块”是很重要的一个步骤。这里利用 Tile-Tree 完成这一任务，下面即是相应的算法“QueryTile”：

算法 2 QueryTile(*pT*, *Q*, *sTid*)

输入：*pT*：指向 Tile-Tree 的指针

Q：完全包含于 *pT* 指向的块内的用户查询区域

sTid：用于存储查询结果——和 *Q* 交叠的地图块 *ID* 的集合(不包括空白块 *ID*)

输出：*Q* 在 *pT* 指向的块中交叠的子块 *ID* 集 *sTid*。

01：if “*pT* 指向的结点(* *pT*)是 *GT* 叶结点”then
02：* *pT* 是最终块，将其 *ID* 插入 *sTid*
03：返回
04：else if “* *pT* 是 *BT* 叶结点”then
05：返回/* 该块是空白块 */
06：else /* * *pT* 是中间结点 */
07：if “* *pT* 的子块分割线横跨区域 *Q*”then
08：根据分割线将 *Q* 对应分成 Q^1, Q^2
09：/* 这里假设 Q^1 位于 * *pT* 的第一子块内 */
10：递归调用 *QueryTile*($pT\rightarrow L$, Q^1, *sTid*)
11：递归调用 *QueryTile*($pT\rightarrow R$, Q^2, *sTid*)
12：else if “*Q* 位于 * *pT* 的第一子块内”then
13：递归调用 *QueryTile*($pT\rightarrow L$, *Q*, *sTid*)
14：else if“*Q* 位于 * *pT* 的第二子块内”then
15：递归调用 *QueryTile*($pT\rightarrow R$, *Q*, *sTid*)
16：end if
17：end if

算法 2 结束

5.2 数据传送策略

在基于客户端的 Internet 地理信息系统上，当客户端用户请求一定区域内的空间数据时，数据传送的基本思想是：服务器端只传送必需的空间数据，客户端收到后将之存储在本机，以备再用。这里利用“地图块分割传送”的方法来实现该思想。如图 6 所示，具体的处理步骤可分为四阶段：层初始化阶段、块定位阶段、数据传送和存储阶段以及本机查询处理阶段。当客户端用户对一地图请求一定区域内的空间数据时，对每一通过“显示比例控制”(该控制决定客户端有无必要请求目标地图层)的地图层，(1) 当客户端第一次请求该地图层时，执行初始化操作：建立本机缓冲文件；请求该地图层的 Tile-Tree 数据并存储于本机；(2) 利用 QueryTile 算法找出和请求区域交叠的目标地图块，并考察这些地图块是否已存储在本机；(3) 如有不在本机的地图块，向服务器请求相应的地图块数据(包括空间数据和索引数据)，接收、存储在本机；(4) 最后，利用目标地图块的“两阶段索引结构”在本机上找出最终查询结

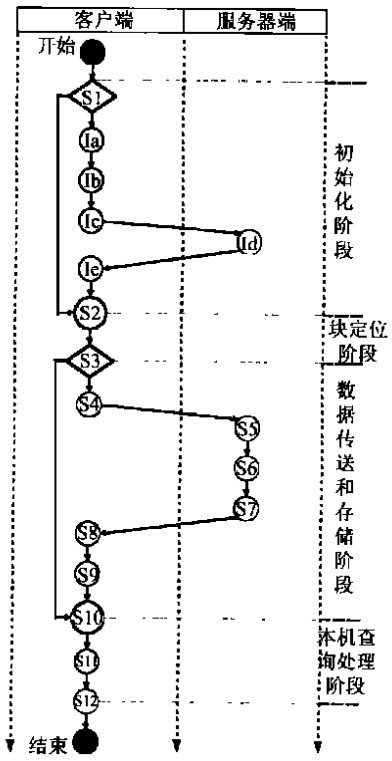

图 6 空间数据传送策略

Fig. 6 The strategy for spatial data transmission

果，并提交相应空间客体数据。

下面是空间数据传送策略的详细步骤

策略 1 RequireSpatialObjects（*LayerInfo*，*Q*）

输入：*LayerInfo*：一地图的所有构成层的信息

Q：用户的查询区域

输出：地图上查询区域 *Q* 内包含的空间客体数据

01：for（*LayerInfo* 中每一个通过显示控制检查的地图层）

02：S1(客户端)．如果第一次向服务器端请求该层数据，顺序执行下面的初始化过程；否则跳至步骤 S2

03：Ia(客户端)．建立存储该层空间客体数据的缓冲文件 *Fcache*

04：Ib(客户端)．建立存储“数据已在本机的地图块”*ID* 的集合 $STID_{local}$，置空

05：Ic(客户端)．向服务器端请求该层的 Tile-Tree *TT*

06：Id(服务器端)．应客户端要求传送 *TT*

07：Ie(客户端)．接收并存储 *TT*

08：S2(客户端)．调用 Query Tile（*TT*，*Q*），得到和 *Q* 交叠的地图块 *ID* 集合 $STID_{require}$

09：S3(客户端)．如果 $STID_{require} \subseteq STID_{local}$，跳至步骤 *S*10，否则继续

10：S4(客户端)．发送 $STID_{local}$ 和 $STID_{require}$ 至服务器

11：S5(服务器端)．计算实际需要传送的地图块，它的 *ID* 集 $STID_{transfer}$ 是：

$$STID_{transfer} = STID_{require} - (STID_{require} \cap STID_{local})$$

12：S6(服务器端)．计算实际需要传送的空间客体，它的 *ID* 集 $SOID_{transfer}$ 是：

$$SOID_{transfer} = SOID_{require} - (SOID_{require} \cap SOID_{local})$$

其中，

$$SOID_{require} = \bigcup_{TID_i \in STID_{require}} SOID_{TID_i}$$

$$SOID_{local} = \bigcup_{TID_i \in STID_{local}} SOID_{TID_i}$$

（$SOID_{TID_i}$ 块 TID_i 内的空间的空 *ID* 集）

13：S7(服务器端)．发送 $STID_{transfer}$ 内所有块对应的 *R* * -*Tree* 索引数据，发送 $SOID_{transfer}$ 内所有的空间客体数据

14：S8(客户端)．接收 *R* * -*Tree* 索引数据，联结至 Tile-Tree *TT* 中相应的块结点

15：S9(客户端)．接收空间客体数据，加入缓冲文件 *Fcache*，修改 $STID_{local}$ 如下：

$$STID_{local} = STID_{local} \cup STID_{require}$$

16：S10(客户端)．对 *STID* require 中的每一个块 TID_i，利用其自身的 *R* * -*Tree*，执行低阶段查询，取得结果数据-空间客体集 $SOID_{result\text{-}TID_i}$。

17：S11(客户端)．计算最终查询结果 $SOID_{result}$：

$$SOID_{result} = \bigcup_{TID_i \in STID_{require}} SOID_{result-TID_i}$$

（$SOID_{result-TID_i}$ 是块 TID_i 中的查询）

18：S12(客户端)．从 *Fcache* 中读取 $SOID_{result}$ 中的空间客体数据，并提交

19：end for

策略 1 结束

6　性能评价

6.1　地图“分裂度”的决定

如 4.2 节所述，地图“分裂度” g 用来决定将一地图层分割成多少个地图块。如果将 g 定得很小，特别是小于或等于 2 时，由于在传送效率上和传统方法没什么大区别，分割传送是没意义的。反之，如果将 g 定得很大，意味着将地图层分割成很多地图块。这样 Tile-Tree 的结点将急剧增加，两端对 Tile-Tree 及地图块的管理费用（查询、存储的时间、空间）也将急剧增加；同时，每次传送的平均地图块数也将增加，系统性能因而随之下降。因此，为了取得最佳的传送效率，求取地图“分裂度” g 的理想值是很必要的。

很容易知道，如果客户端请求某一区域的空间数据时，该区域总是位于某一地图块时是最理想的。但由于客户端请求区域的不确定性，这显然是不可能的。实际上，不管怎样分割，都会出现用户请求区域横跨 4 个地图块的情况。因此，“分裂度” g 的理想值应满足以下条件：(1) 每次查询时，用户请求区域横跨的地图块数一般不超过 4 个；(2) 如有超过 4 个的情况，则其余的地图块应位于本机。也就是说，对每一地图层，每次实际向服务器请求传送的地图块数不超出 4 个。

根据显示控制中地图层的最小显示比例，我们可求出“分裂度” g 的理想值，具体如下：

1．对地图层 A（它的层最小边界矩形为 $LMBR_A$），根据其具体主题，假设其最小显示比例为 1：*x*；

2. 则：一般地，用户最大可能查询区域为 R_{max}，如下：

$$R_{max} = \text{Area}(LMBR_A)/X$$

(Area($LMBR_A$) 指 $LMBR_A$ 的面积)

3. 根据前述条件，地图层 A 的分裂度 g 由下式可求出：

$$Area(LMBR_A)/2^{g+1} \leqslant R_{max} \leqslant \text{Area}(LMBR_A)/2^{g-1}$$

即

$$2^{g-1} \leqslant x \leqslant 2^{g+1}$$

4. 从而得到 g 的理想值，为：

$$\begin{cases} 2^{g-1} \leqslant x \leqslant 2^{g+1}, \\ g > 2(\text{if } g \leqslant 2, \text{ then } g = 0) \end{cases}$$

必须说明的是，对一地图，由于不同的地图层有不同的最小显示比例，其相关联的 g 值是不同的。而且，对同一地图层，由于用户显示窗的大小不同，最小显示比例也有微小的差异，从而其 g 值也不一样。因此，地图层的分裂度 g 值是一经验值，而非准确值。根据实验验证，对一般大小和主题的地图层，其 g 值取 4—7 是恰当的。

6.2 Tile-Tree 的分析

和已存系统的传送方法比较，本文提出的方法新增了一个索引结构 Tile-Tree，它决定每次传送的地图块。考虑到它对整个系统的性能影响，对它进行空间分析和时间分析是很必要的。本小节考察 Tile-Tree 的平均所占内存大小和平均搜索时间。

我们知道，Tile-Tree 的平均所占内存大小决定于它的平均总结点数，平均搜索时间决定于它的搜索路径平均长度。对某一给定分裂度 g 的地图层，其分割成的地图块的最大可能数是 $2^g * 2$，即 Tile-Tree 的叶结点(一般块结点和空白块结点)数是 $N_{leaf} = 2^g * 2 = 2^{g+1}$，则 Tile-Tree 的中间结点数为 $N_{trunk} = 2^0 + 2^1 + \cdots 2^g$。从而 Tile-Tree 的最大总结点数为：

$$N_{max} = N_{trunk} + N_{leaf} = 2^0 + 2^1 + \cdots + 2^g + 2^{g+1} = 2^{g+2} - 1$$

另一方面，Tile-Tree 的最小总结点数为：

$$N_{min} = 2g + 1$$

因此，Tile-Tree 的平均总结点数为：

$$N_{avg} = (N_{max} + N_{min})/2 = 2^{g+1} + g$$

关于 Tile-Tree 中的搜索路径长度问题，可以看出从根结点至叶结点的搜索路径最短长度为 $L_{short} = 1$，最长长度为 $L_{long} = g + 1$，因而搜索路径平均长度为 $L_{avg} = (g+2)/2$。

考虑实验理想值 $g = 5$，则可得出：

$$N_{avg} = 69, \quad L_{avg} = 3.5 \approx 4$$

这就是说，Tile-Tree 的平均存储空间是 69 M_{node} (其中，M_{node} 是 Tile-Tree 的一个结点存储空间)；平均搜索时间复杂度是 $O(4)$。

6.3 实验验证

在本课题中，我们利用上面提出的空间数据传送策略和 ActiveX 技术构造了一个具备基本功能的基于客户端 Internet 地理信息系统。其系统结构如下(图 7)所示：

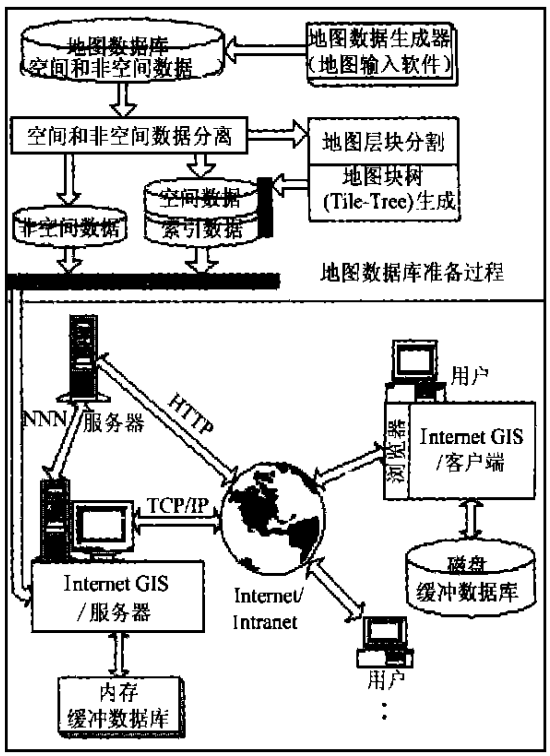

图 7 实验系统的结构

Fig. 7 The architecture of proposed system

本文利用该实验系统，用图 1 的地图作为样本数据，在客户端测试了系统响应时间和本机使用的存储空间等性能参数指标。其中，图 1 地图各层的分裂度 g 设定如表 1。

表 1 图 1 中各地图层块分裂度 g 的值

Table 1 The value of g about the layers in figure 1

	河流层	山层	建筑物层	道路层	边界层
分裂度 g	3	4	5	5	0

首先我们定义了一系列操作顺序(如：初始化—

连结服务器—放大—移动—缩小—…等)，然后在此基础上，进行用户模拟操作，并记录相应操作完成所需的响应时间和该时刻客户端缓冲数据库所用的存储空间。另外，在实验中，对一使用传统传送策略的既存系统 GEOWeb V1.0[10]，在客户端进行了同样的系统响应时间和本机使用的存储空间等性能参数指标的测试。系统响应时间比较和本机使用的存储空间比较，分别列于下面(图 8、图 9)：

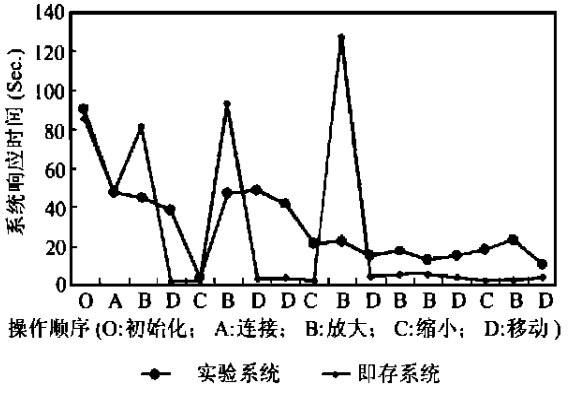

图 8　系统响应时间的比较

Fig. 8　The comparison of system response time

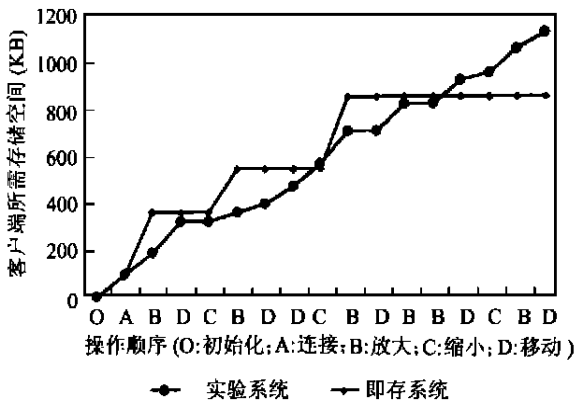

图 9　客户端所需存储空间的比较

Fig. 9　The comparison of client storage space (KB)

图 8 表示在两系统中，对相同的地图数据，在相同的操作顺序下，不同的系统响应时间。图 9 表示在两系统中响应操作下客户端使用的临时存储空间(用于存储缓冲空间数据)。从图 8 中我们可以看出系统平均响应时间有了很大的改善。特别是，有地图层新加入传送时(这时传统方法一次传送整个地图层)。从数据上看，系统平均响应时间提高了大约 60%程度。当然，和传统方法比较，由于提案方法新增了 Tile-Tree 结构，并且本机数据按地图块组织存储等因素，如图 9 所示，使用提案方法系统的客户端使用存储空间比既存系统大。但这种以“空间换取时间”的方法对提高整个系统的性能是值得的。

7　总　结

本文着重于解决基于客户端的 Internet 地理信息系统中大容量空间数据的传送效率性问题。解决方法的主要思想是，从服务器端往客户端传送空间数据时，不以地图层为单位传送，而以更小的地图块为单位传送。论文首先提出在服务器端对地图数据的分割算法，然后给出在客户端对地图块的查询定位算法。最后，提出系统的空间数据传送策略。

本文最重要的地方是如何将一地图层分成适当的地图块。文中讨论了地图层“分裂度”最佳理想值的求值方法。作为结论，得出对一普通大小和一般主题的地图层，“分裂度”取 4 至 7 是适当的。

关于性能评价，实验考察了两个性能指标：用户响应时间、客户端所需存储空间。和使用传统方法的既存系统比较，缩短了用户响应时间，提高了系统性能。

参　考　文　献 (References)

[1] Aronoff S. Geographic Information Systems [M]. WDL Publications, 1991.

[2] Guting R H. An Introduction to Spatial Database Systems [J]. *VLDB Journal*, 1994, 3(4): 357—399.

[3] Asplan Viak. An Introduction to Geographic Information Systems (Second Edition) [M]. John Wiley & Sons Inc., 1999.

[4] Maichael N. DeMers. Fundamentals of Geographic Information Systems (Second Edition) [M]. John Wiley & Sons Inc., 1999.

[5] Guting R H. Gral: an Extensible Relational Database System for Geographic Applications [C]. Proc. 15 th Int. Conf. on Very Large Data Bases, Amsterdam, Netherland, 1989, 33—44.

[6] Burrough P A. Principles of Geographic Information Systems for Land Resources Assesment [M]. Oxford Univ. Press, 1986.

[7] ArcNews. Scalable Internet Mapping Solutions from ESRI: Maps on the New Frontier [M]. ArcNews, fall, 1997.

[8] Luis Manuel Callejas. Web-Based GIS: A brief History [C]: http://blaze.trentu.ca/~ermlc/web-gis.html.

[9] Yew Kuan Choo, Chen-Chen Lee. Integrated Distributed Geographical Information Systems (IDGIS) [C]. 1997 ESRI International User Conference, July 1997, 101—108.

[10] Cho Y S. Kim H. Y. Kim J. H, Bae H. Y. The Design and Implementation of Componentized Web-enabled GIS [C]. ICIMU 98, Sept. 1998, 89—95.

[11] Peng Z. R. An Assessment of the Development of Internet GIS [C]. Proceedings of the ESRI User Conference, 1997, 89—97.

[12] Wei Z K, Kim H Y, Cho Y S, Bae H Y. Design and Implementation

of Web-enabled C/S GIS Using ActiveX Technology [J]. *Computer Engineering & Application*, 1999, **35**(4): 63—66. [魏祖宽, 金洪然, 赵永胜, 裴海英. 利用 ActiveX 技术的 Web-enabled C/S 地理信息系统的设计和实现[J]. 计算机工程与应用, 1999, **35**: (4): 63—66.]

[13] CITRIX. WinFrame WorkGroups 1.7 [M]. Citrix Systems Inc., 1997.

[14] ESRI. MapObject 1.0 Refrence [M]. ESRI Inc., 1997.

[15] Guenther, Buchmann A. Research Issues in Spatial Databases [J]. *ACM SIGMOD RECORD*, 1990, **19**(4): 61—68.

[16] Autodesk. Mapguide 2.5 Refrence [M]. Autodesk Inc., 1998.

[17] Intergraph. GeoMedia Web Map 1.0 Refrence [M]. Intergraph Corporation Press, 1998.

[18] James T. Hathaway. Demographic Data Viewer 3.0 Tutorial [M]. CIESIN, Columbia University Press, 1998.

[19] IntemetGIS. ActiveMaps 2.0 Manual [M]. InternetGIS Press, 1997.

[20] F. Wang. Spatial Data Model for Feature-Based GIS [C]. Proceedings of the Workshop on Computer Science and Information Technologies CSIT99, Russia, 1999, 107—111.

[21] Wei Z K, Cho S K, Kim I H, Bae H Y. A New Spatial Representation in GIS [C]. Proc. of the 5th Intemational Conference on GeoComputation, University of Greenwich, Kent, United Kingdom, Aug. 2000, 36—42.

[22] Clodoveu A. Davis Jr., Alberto H. F. Laender. Multiple Representations in GIS: Materialization Through Map Generalization, Geomretric and Spatial Analysis Operations [C]. ACM GIS '99, Kansas City, MO USA, 1999, 60—65.

[23] Kriegel H P, Heep P, Heep S, Schiwietz M, Schneider R. An Access Method Based Query Processor for Spatial Database Systems [C]. Int. Workshop on Database Management Systems for Geographic Applications, Capri, Italy, 1991, 24—31.

[24] Samet H. The Design and Analysis of Spatial Data Structures [M]. Addison Wesley, 1990.

[25] Seeger B, Kriegel H P. Techniques for Design and Implementation of Efficient Spatial Access Methods [C]. Proc. 14th Int. Conf. on Very Large Databases, Los Angeles, CA., 1988, 102—110.

[26] Volker Grede, Multidimensional Access Methods [J]. *ACM Computing Surveys*, June, 1998, **30**(2): 170—208.

[27] Beckmann N, Kriegel H P, Schneider R, Seeger B. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles [C]. Proc. Of ACM SIGMOD, Atlantic City, NJ., 1990, 322—331.

[28] Brinkhoff T, Kriegel H P. Schneider R. Comparison of Approximations of Complex Objects used for Approximation-based Query Processing in Spatial Database Systems [C]. Proc. 9th Int. Conf. On Data Engineering, Vienna, Austria, 1993, 336—341.

[29] Brinkhoff T, Hom H, Kriegel H P, Schneider R. A Storage and Access Architecture for Effcient Query Processing in Spatial Database Systems [C]. Proc. 3rd Symp. on Large Spatial Databases, Singapore, 1993, 112—119.

[30] Gutman A. R-trees: A Dynamic Index Structure for Spatial Searching [C]. Proc. Of ACM SIGMOD, Vol.13, 1984, **132**: 47—57.

[31] Kriegel H P, Hom H, Schiwietz M. The Performance of Object Decomposition Techniques for Spatial Query Processing [C]. Proc. 2nd Symp. on Large Spatial Databases, Zurich, Switzerland, 1991, 112—123.

Optimization Strategy for Spatial Vector Data Transmission in Internet GIS

WEI Zu-kuan, BAE Hae-young

(*Dept. of Computer Science & Engineering, INHA University, INCHON* 402-751, *KOREA*)

Abstract: It is important to solve the low performance of spatial vector data transmission in the client-side Intermet GIS (Geographic Information System) which handles large-size spatial geographic information on the Internet. This paper has proposed a new efficient method for it. The basic idea is that, firstly, a large-size vector map is divided into several parts, where each part is called a 'Tile', according to an appropriate division granularity. Secondly, when an user requires a certain region in the map at client side, the GIS server only transmits spatial data in the tiles which overlap with the requested region. And the received data are stored in client local machine for reuse. For the support of this idea, a method of tile-division, an algorithm for tile-query processing and a strategy for efficient spatial data transmission between server side and client side have been provided. Comparing with the traditional methods of client-side Internet GIS, the performance improvement is achieved by the usage of the proposed method.

Key words: spatial data transmission; GIS; internet GIS; spatial database