基于内存映射文件技术的海量影像数据快速读取方法*

胡伟忠, 刘　南, 刘仁义
(浙江大学 GIS 重点实验室, 浙江 杭州 310028)

摘　要: 随着信息技术和传感器技术的飞速发展,使得遥感图像的数据量呈几何级数的递增,而传统的利用文件指针来读取文件的方法,只能正确读取 2GB 以下的数据。针对此种情况,提出了新方法,并分析了其关键技术,用 VC + +给出了实现的代码。

关键词: 内存映射文件技术; 金字塔结构; 缓存技术; 图像分块
中图法分类号: TP751　　**文献标识码:** A　　**文章编号:** 1001-3695(2005)02-0111-02

Quick-read Means for Large Scale Images Based on Storage Mapping File Technique

HU Wei-zhong, LIU Nan, LIU Ren-yi
(GIS Key Laboratory, Zhejiang University, Hangzhou Zhejiang 310028, China)

Abstract: The rapid development of information technology and sensor technology has given rise to the geometrically progressive increase of remote sensing data. Aware of the conventional means incapability to read file data above 2GB exactly, a new reading technique has been proposed, through analysis of its critical techniques, and lists the code segments on the strength of VC + +.

Key words: Storage Mapping Files Technique; Pyramid Structure; Cache Technique; Data Block

随着信息技术和传感器技术的飞速发展,现代遥感数据源不断丰富,使得遥感影像图的数据量从以前的几兆、几十兆增长到现在的几百兆,甚至是几 GB、几十 GB。在这种情况下,如果对图像文件的处理仍采用传统的方法,显然是不行的。一方面,计算机硬件环境的发展已经远远不能满足图像处理的需要;另一方面,传统的利用文件指针方式来读取图像文件的方法,只能正确读取文件前面 2^{31} - 1(约 2GB)字节,所以不能满足大于 2GB 以上的影像文件的读取。本文是在笔者开发的中国海岸带及近海卫星遥感综合应用系统(Max_Deskpro)过程中,结合对海量遥感影像进行处理的要求,根据实际提出的一种快速读取海量图像的方法。实现该方法的主要技术包括内存映射文件技术、动态构建影像金字塔、缓存技术、图像分块技术。

1　内存映射文件

在对文件进行读写操作的时候,Win32 API 和 MFC 都提供了支持这些操作的函数和类,如 Win32 API 的 WriteFile( ), ReadFile( )和 MFC 提供的 CFile 类。这些函数在文件的数据量小于(2^{31} - 1)字节的时候,在大多数场合是可以满足用户的需求。但是当文件的数据量大于(2^{31} - 1)字节(约 2GB)时候,就不能正确地使用 Seek( )函数定位到要读取数据的位置,这是因为 Seek(LONG Loff, UINT nFrom)函数提供的 Loff 参数是 LONG 型的,它最多支持(2^{31} - 1)字节的偏移量。假如图像文件的数据量大于 2GB 的话,就不能定位到 2GB 以后的部分来读取数据。由于内存映射文件技术的主要函数都提供了两个 DWORD 型参数来分别表示文件大小或偏移量的低 32Bits 和高 32Bits,这两个参数加起来共 64Bits,因此它所支持的最大文件长度为 16EB(2^{64} - 1 字节),几乎可以满足对任何海量数据文件的处理。下面对两个关键函数进行具体说明:

```
HANDLE CreateFileMapping( HANDLE hFile, //文件内核对象
LPSECURITY_ATTRIBUTES lpFileMappingAttributes, //返回的句柄可否被子程序使用
DWORD flProtect, //指定页面的保护属性(只读、读写、拷贝等)
DWORD dwMaximumSizeHigh, //文件大小的高 32Bits
DWORD dwMaximumSizeLow, //文件大小的低 32Bits, 和 dwMaximumSizeHigh 组成 64Bits 值,来表示文件的长度
LPCTSTR lpName //文件映射对象的名字
);
```

该函数创建一个文件内存映射对象,以告知系统文件映射对象需要多大的物理存储器。由于内存映射文件的物理存储器实际上是本地硬盘上的一个文件,而不是从系统的页文件中分配的内存,所以 CreateFileMapping( )函数不分配进程的地址空间,不占用内存空间,因此在实际应用中通常是一次性把整个文件创建为内存映射对象。如果要把文件中的数据映射到进程的地址空间中,则需要调用 MapViewofFile( )函数。该函数的具体定义如下:

```
LPVOID MapViewofFile(
HANDLE hFileMappingObject, //CreateFileMapping( )函数返回的文
```

收稿日期: 2004-03-10; 修返日期: 2004-04-22
基金项目: 国家"863"计划资助项目(2001AA630301);国家自然科学基金资助项目(40271087);浙江省自然科学基金资助项目(401006)

件内存映射对象句柄
DWORD dwDesiredAccess, //对页数据的访问方式
DWORD dwFileoffsetHigh, //映射开始位置的高 32Bits
DWORD dwFileoffsetLow, //映射开始位置的低 32Bits, 和 dwFileoffsetHigh 组成 64Bits 值, 来表示文件的偏移地址
DWORD dwNumberofBytesToMap //映射的字节数, 即在进程的地址空间中分配的字节数);

调用该函数后, 访问文件中的数据, 就如同它位于内存中一样, 这样就省去了对文件执行 I/O 操作的时间, 所以它比用 CFile 类和 Win32API 的 WriteFile( ), ReadFile( ) 速度上要快得多。在使用该函数的时候, 由于它的返回值 LPVOID 是一个 32Bits 的指针类型, 所以在处理海量数据文件的时候, 还要进行分块映射, 并且一次映射到进程地址空间的数据量要小于 2^{31} - 1 字节数。文件内存映射的开始位置(即文件的偏移地址) 还必须满足是操作系统分配粒度的整数倍(通常操作系统的分配粒度是 64K) 。图 1 是使用内存映射文件的一般步骤。

2 金字塔技术

随着遥感影像数据量变得越来越大, 为了提高图像实时缩放和显示的速度, 快速地获取不同分辨率下的图像信息, 往往会对原始图像建立金字塔结构, 根据不同的显示要求, 调用不同金字塔等级中的数据, 来达到快速显示、漫游的目的。图像金字塔结构是一个多分辨率的层次模型, 从金字塔的底层到顶层, 其分辨率越来越低, 对应层的数据量也越来越小(一般情况下是呈 4^n 递减), 所以在不考虑数据压缩的情况下建立金字塔后的图像比原图像增加大约 1/3 的数据量。图 2 是一个三层金子塔结构。

图 1 内存映射文件的步骤

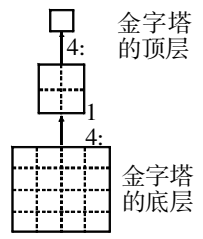

图 2 金字塔结构

从图 2 中可以看出, 从金字塔的底层开始每四个相邻的像素经过重采样生成一个新的像素, 依此重复进行, 直到金字塔的顶层。重采样的方法一般有以下三种: 双线性插值、最临近像元法、三次卷积法。其中最临近像元法速度最快, 如果对图像的边缘要求不是很高的话, 最适合使用该方法。三次卷积由于考虑的参考点数太多、运算较复杂等原因, 速度最慢, 但是重采样后图像的灰度效果较好。

对一般的 PC 机来说, 把整幅图像一次性读入内存来生成金字塔是不大可能的。所以在对文件建立金字塔的时候, 只能是根据实际情况, 首先对原始影像图进行分块。分块的规则是随意的, 本系统为了避免对文件数据的来回映射, 所以采用每 128 行作为一数据块, 这样可以顺序地对文件进行映射, 最大限度地避免了对文件某些行、列的重复映射, 使映射的次数更少, 读取图像数据的速度更快。

3 算法的设计和实现

3.1 设计思想

得到显示范围数据的基本步骤是这样的: 利用内存映射文件技术把整幅影像文件创建为内存映射对象, 然后按照事先定义的缓存大小(不同的图像大小是不一样的), 把影像数据的一部分映射到进程的地址空间; 根据图像当前的坐标显示范围(大地坐标或经、纬坐标) 来计算图像的像素显示范围, 然后根据图像像素显示范围和屏幕显示范围来确定当前图像显示的金字塔级别。

从图像像素显示范围的起始行、起始列开始读取数据的步骤是: 首先判断当前要读取的数据是否已经在缓存中, 如果没有在缓存中, 则先调用 UnmapFileofView( ) 函数解除上一次的内存映射空间, 然后再调用 MapFileofView( ) 函数重新映射新的数据; 根据返回的指针, 一边读取数据, 一边构建当前等级的图像金字塔; 依此重复进行, 直到读完全部数据为止。图 3 是读取数据的流程图。

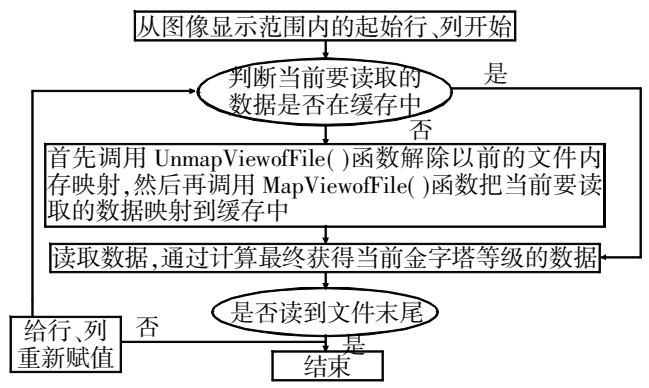

图 3 读取数据的算法流程

3.2 算法的实现

```
/* lLeft, lTop, lRight, lBottom 是图像的像素显示范围。OutHeight, lOutWidth 是图像在当前金字塔等级中的行、列总数。m_PyramidStep 是当前显示的图像金字塔等级, 数据经读取重新组织后, 存放在 pArray 数组指针里面 */
void GetBlockData( long lLeft, long lTop , long lRight, long lBottom, long lOutWidth, long lOutHeight, char * pArray[ ], int m_PyramidStep)
{
long width = lRight-lLeft + 1; //图像的宽度
long height = lBottom-lTop + 1; //图像的高度
long realWidth = ( lOutWidth + 3) /4* 4; //实际宽度
//遍历图像的像素显示范围
for ( int i = 0; i < height; i = i + height/lOutHeight)
{
long lRow = lTop + i ;
for ( int j = 0; j < width; j = j + width/lOutWidth)
{
/* GetData( ) 函数首先判断当前要读取的数据是否在缓存中, 如果没有则先把数据映射到缓存中, 然后再读取图像块的数据, 并根据金字塔级别重采样生成该级别下的数据 */
GetData( lRow, lLeft + j, pArray[ 0], pArray[ 1], pArray[ 2], m_PyramidStep);
pArray[ 0] + +; pArray[ 1] + +; pArray[ 2] + +;
}
//补空(输出图像行字节数必须为 4 的整数倍)
pArray[ 0] + = realWidth - lOutWidth;
pArray[ 1] + = realWidth - lOutWidth;
pArray[ 2] + = realWidth - lOutWidth;
}
//指针复位
pArray[ 0] - = realWidth * lOutHeight;
pArray[ 1] - = realWidth * lOutHeight;
pArray[ 2] - = realWidth * lOutHeight;
}
```

4 运行的结果

本程序在 CPU 为 P4 2.4GHz, 内存为 256MB 的 PC 机上运行, 打开一幅 8GB 的影像文件所花的时间大约为 15s。与其他当前流行的图像处理软件如 ERDAS Image 8.5, ESRI 的 ARCGIS 8.2 进行比较的过程中发现, 这两个软件只能正确读取 2GB 以下的数据, 对文件的数据量大于 2GB 的 (下转第 107 页)