激光扫描点云数据的快速读取新方法

A New Rapid – reading Method for Point Cloud of Laser Scanner

朱根松　（南昌大学　江西南昌　330029）
（江西理工大学　江西赣州　341000）
周天瑞　潘海鹏　胡世飞　（南昌大学　江西南昌 330029）

摘要：在反求工程中，随着测量技术的发展，点云数据文件越来越大。为了提高点云数据文件的读取效率，本文分析了虚拟内存及内存映射文件的工作原理，并在此基础上提出用内存映射文件来实现点云文件的快速读取。通过实验对比，内存映射文件相对传统 I/O 在文件读取效率上有着明显的提高。

关键词：反求工程　内存映射文件　虚拟内存　点云数据

1　引言

在反求工程领域，随着测量技术的不断完善，各种测量方法的速度和效率不断提高。目前一般的光学测量系统可以在一分钟内得到几十万个数据点的测量。因而，测量结果所得的数据量非常大，形象化地称“点云”。点云数据文件越来越大，要想对大数据量文件进行快速存取，用传统的通过 I/O 文件读取技术将很难满足要求，文件输入和输出（I/O）是计算机编程中的一个巨大瓶颈。如果用通常有多线程对文件输入和输出（I/O）进行操作，涉及到线程间互拆、同步、等待等复杂问题，反而使编程变得复杂。

本文利用 WINDOWS 内存映射文件代替传统文件输入和输出（I/O）操作。对比传统文件（I/O）方式读取数据，采用内存映射文件方式读取数据的效率要明显提高。

2　Windows 虚拟内存

Windows 虚拟内存采用一种基于页（Page_based）的内存管理系统。如图 1 所示，它由页表（Page Directory）、页面（Page Table）、页（Page）组成。页由页表（Page Table）进行管理，而页表又由页目录（Page Directory）进行管理。虚拟内存大小随操作系统不同而不同，对于 WIN32 操作系统，采用的是 32 位线性地址，其虚拟内存为 232Byte，即 4GB。它由 1024 张页目录，1024 张页表，4096 张页组成，其中高 2G 地址空间由系统保留作用，低 2G 地址空间由应用程序使用。操作系统根据应用程序的请求，把需要用的逻辑空间映射到物理空间，而把不用的内存写入磁盘。

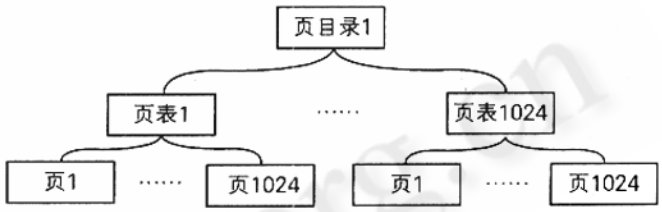

图 1　WINDOWS 虚拟内存管理系统

3　内存映射文件

内存映射文件与虚拟内存有些类似，通过内存映射文件可以保留一个地址空间的区域，同时将物理存储器提交给此区域，只是内存文件映射的物理存储器来自一个已经存在于磁盘上的文件，而非系统的页文件，而且在对该文件进行操作之前必须首先对文件进行映射，就如同将整个文件从磁盘加载到内存。由此可以看出，使用内存映射文件处理存储于磁盘上的文件时，将不必再对文件执行 I/O 操作，这意味着在对文件进行处理时将不必再为文件申请并分配缓存，所有的文件缓存操作均由系统直接管理，由于取消了将文件数据加载到内存、数据从内存到文件的回写以及释放内存块等步骤，使得内存映射文件在处理大数据量的文件时能起到相当重要的作用。另外，实际工程中的系统往往需要在多个线程之间共享数据，如果数据量小，处理方法是灵活多变

的，如果共享数据容量巨大，那么就需要借助于内存映射文件来进行。实际上，内存映射文件正是解决本地多个线程间数据共享的最有效方法。

在使用内存映射文件时，所使用的 WINDOWS API 函数主要为下面 5 个函数。

（1） HANDLE CreateFile（LPCTSTR lpFileName，DWORD dwDesiredAccess，DWORD dwShareMode，LPSECURITY_ATTRIBUTES lpSecurityAttributes，DWORD dwCreationDisposition，DWORD dwFlagsAndAttributes，HANDLE hTemplateFile）；

（2） HANDLE CreateFileMapping（HANDLE hFile，LPSECURITY_ATTRIBUTES lpFileMappingAttributes，DWORD flProtect，DWORD dwMaximumSizeHigh，DWORD dwMaximumSizeLow，LPCTSTR lpName）；

（3） LPVOID MapViewOfFile（HANDLE hFileMappingObject，DWORD dwDesiredAccess，DWORD dwFileOffsetHigh，DWORD dwFileOffsetLow，DWORD dwNumberOfBytesToMap）；

（4） BOOL UnmapViewOfFile（LPCVOID lpBaseAddress）；

（5） BOOL CloseHandle（HANDLE hObject）；

首先要通过 CreateFile（）函数来创建或打开一个文件内核对象，这个对象标识了磁盘上将要用作内存映射文件的文件。在用 CreateFileMapping（）函数来创建一个文件映射内核对象以告诉系统文件的尺寸以及访问文件的方式。由 MapViewOfFile（）函数负责通过系统的管理而将文件映射对象的全部或部分映射到线程地址空间。此时，对内存映射文件的使用和处理同通常加载到内存中的文件数据的处理方式基本一样，在完成了对内存映射文件的使用时，还要通过一系列的操作完成对其的清除和使用过资源的释放，首先通过 UnmapViewOfFile（）完成从进程的地址空间撤消文件数据的映像、通过 CloseHandle（）关闭前面创建的文件映射对象和文件对象。值得一提的是，在多线程操作中，CreateFile 函数中的 dwDesiredAccess 要设为 GENERIC_READ | GENERIC_WRITE，CreateFileMapping 函数中的 flProtect 设为 PAGE_READWRITE，MapViewOfFile 函数中的 FILE_MAP_WRITE | FILE_MAP_READ。否则会出现共享冲突。

4　实现数据快速读取的面对对象方法

本文以深圳智泰精密仪器有限公司生产的激光扫描仪产生的 ASC 文件为例，介绍实现数据快速读取的面对对象方法。ASC 文件是一种纯文本文件，每行分别记录一个数据点的（x，y，z）坐标。激光扫描仪的光刀每扫描一次，就产生一条数据线，激光扫描仪管理软件根据一定的间隔将数据线离散成数据点写入 ASC 文件。因此，虽然受背景光、工件表面颜色的影响而产生一定的噪声数据点，而使每次光刀扫描产生的点云数据不在同一平面内。但是在一定容许误差范围内，可以把 ASC 文件看成由一系列垂直坐标轴（本文为 Y 轴）的平面点云数据组成。首先设计如下双向数据结构用于保存读取的数据点。

```
struct _PcData{
    double Err;// 用于保存容许误差
    CArray < PCVECTEX, PCVECTEX > List;
    //typedef struct _pc{ double x,y,z;} PCVECTEX
    struct _PcData * pPre;//用于指向前一平面的点云数据
    struct _PcData * pNext;// 用于指向后一平面的点云数据
};
```

设置容差 Err，首先读取几个系列的平面点云数据，计算出光刀扫描间隔均值，再取其 0.2 ~ 0.5 为容差 Err 的值即可。用此数据结构也可处理散乱点云数据，读取的数据自动存入以两倍容差 Err 的为间隔的一系列长方体内，可大大提高后续的点云切片处理的效率。读取 ASC 文件类 CAscFile 设计如下：

```
class CAscFile{
  public:
    struct PcData m_pc;//存放点云数据
    far char * p_Mem;//指向内存映射文件数据
  public:
    ……
    BOOL OpenAscFile(char * name);//打开磁盘文件并建立内存映射文件
    BOOL ReadData()//读取数据
    VOID CloseAscFile()//关闭内存映射文件和磁盘文件
    ……
};
```

限于篇幅，本文只给出 OpenAscFile（char * name）实现代码。至于 ReadData（）主要用 c/c++ 提

供的 stdio::ReadString 和 scanf 两个函数实现。

```
BOOL CAscFile::OpenAscFile(char * name)
{
    if(name==NULL)
    {
      AfxMessageBox("File open error",MB_OK);
      return FALSE;
    }
    //打开磁盘文件
    m_hFile = CreateFile(name,GENERIC_READ|GENERIC_WRITE,FILE_SHARE_READ,
        NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
    if(!m_hFile)
    {
      AfxMessageBox("File open error",MB_OK);
      return FALSE;
    }
    //创建内存映射文件
    m_hMap = CreateFileMapping(m_hFile,NULL,PAGE_READWRITE,0,0,NULL);
    if(m_hMap!=NULL&&GetLastError()==ERROR_ALREADY_EXISTS)
    {
      CloseHandle(m_hMap);
      CloseHandle(m_hFile);
      m_hMap=m_hFile=NULL;
      return FALSE;
    }
    //取得数据地址
    p_Beginoffile = (far char *) MapViewOfFile(m_hMap,
        FILE_MAP_WRITE|FILE_MAP_READ,0,0,0);
    if(!p_Beginoffile)
    {
      CloseHandle(m_hMap);
      CloseHandle(m_hFile);
      m_hMap=m_hFile=NULL;
      p_Beginoffile = (unsigned char *)NULL;
      return FALSE;
    }
    return TRUE;
}
```

5 实验结果对比

在联想昭阳 280S(配置:CPU CM1.4GHz,内存 512MB,操作系统 WinXp)电脑上,分别采用传统 I/O 文件读取、内存映射文件和 Surface10.6,读取文件 1(数据点数:43586)和文件 2(数据点数:181343)。计时开始时间,传统 I/O 文件读取从 CFile::Open()开始,内存映射文件从 CreateFile()开始。计时结束时间,传统 I/O 文件读取和内存映射文件均以读完点云数据为结束时间。表 1 为实验数据,从表 1 可以得出,内存映射文件读取速度相对传统 I/O 文件读取速度有大幅度的提升,尤其是在读大的数据文件时更明显。

表 1　实验数据

	传统 I/O 文件读取	内存映射文件
文件 1(ms)	674	426
文件 2(ms)	1592	903

6 结论

本文分析了 Win32 虚拟内存及内存映射文件的工作原理,得出相对传统 I/O 文件读取,内存映射文件在读取效率上有明显提高。但仍须改进方法,以得到更高的读取效率。

参考文件

1 Jeffrey Richter. windows 核心编程[M],北京:机械工业出版社,2000.

2 李文望,快速成型中的三维数据准备[J],鹭江职业大学学报,2005,13(1):54-58.

3 谭小洪、达飞鹏、王丙文等,OOP 在图形处理中的应用--三维点云的处理[J],现代电子技术,2001,12:14-16.

4 常永、昌王芳、冯新喜,使用 Win32 优化内存和文件 I/O 管理[J],现代电子技术,2003,4:32-35.

5 孙文庆、刘秉权、肖镜辉,基于内存映射文件的数据共享技术研究与应用[J],微计算机应用,2005,3:192-195.