**开源C++函数库Boost内存池使用与测试**

[**0**条评论](http://tech.it168.com/a2011/0726/1223/000001223399_all.shtml#articlecomment)

2011-07-27 00:00    [it168网站](http://www.it168.com)原创 　作者: 王学斌　编辑: [董建伟](http://archive.it168.com/user/%b6%ad%bd%a8%ce%b0/)

　　【IT168 专稿】Boost库是一个可移植的开源C++函数库，鉴于STL(标准模板库)已经成为C++语言的一个组成部分，可以毫不夸张的说，Boost是目前影响最大的通用C++库。Boost库由C++标准委员会库工作组成员发起，其中有些内容有望成为下一代C++标准库内容，是一个“准”标准库。

　　Boost内存池，即boost.pool库，是由Boost提供的一个用于内存池管理的开源C++库。作为Boost中影响较大的一个库，Pool已经被广泛使用。

**1. 什么是内存池**

　　“池”是在计算机技术中经常使用的一种设计模式，其内涵在于：将程序中需要经常使用的核心资源先申请出来，放到一个池内，由程序自己管理，这样可以提高资源的使用效率，也可以保证本程序占有的资源数量。经常使用的池技术包括内存池、线程池和连接池等，其中尤以内存池和线程池使用最多。

　　内存池(Memory Pool)是一种动态内存分配与管理技术。通常情况下，程序员习惯直接使用new、delete、malloc、free等API申请分配和释放内存，导致的后果时：当程序长时间运行时，由于所申请内存块的大小不定，频繁使用时会造成大量的内存碎片从而降低程序和操作系统的性能。内存池则是在真正使用内存之前，先申请分配一大块内存(内存池)留作备用，当程序员申请内存时，从池中取出一块动态分配，当程序员释放内存时，将释放的内存再放入池内，并尽量与周边的空闲内存块合并。若内存池不够时，则自动扩大内存池，从操作系统中申请更大的内存池。

**内存池的应用场景**

　　早期的内存池技术是为了专门解决那种频繁申请和释放相同大小内存块的程序，因此早期的一些内存池都是用相同大小的内存块链表组织起来的。

　　Boost的内存池则对内存块的大小是否相同没有限制，因此只要是频繁动态申请释放内存的长时间运行程序，都适用Boost内存池。这样可以有效减少内存碎片并提高程序运行效率。

**安装**

　　Boost的pool库是以C++头文件的形式提供的，不需要安装，也没有lib或者dll文件，仅仅需要将头文件包含到你的C++工程中就可以了。Boost的最新版本可以到http://www.boost.org/下载。

**2. 内存池的特征**

**2.1 无内存泄露**

　　正确的使用内存池的申请和释放函数不会造成内存泄露，更重要的是，即使不正确的使用了申请和释放函数，内存池中的内存也会在进程结束时被全部自动释放，不会造成系统的内存泄露。

**2.2 申请的内存数组没有被填充**

　　例如一个元素的内存大小为A，那么元素数组若包含n个元素，则该数组的内存大小必然是A\*n，不会有多余的内存来填充该数组。尽管每个元素也许包含一些填充的东西。

**2.3 任何数组内存块的位置都和使用operator new[]分配的内存块位置一致**

　　这表明你仍可以使用那些通过数组指针计算内存块位置的算法。

**2.4 内存池要比直接使用系统的动态内存分配快**

　　这个快是概率意义上的，不是每个时刻，每种内存池都比直接使用new或者malloc快。例如，当程序使用内存池时内存池恰好处于已经满了的状态，那么这次内存申请会导致内存池自我扩充，肯定比直接new一块内存要慢。但在大部分时候，内存池要比new或者malloc快很多。

**3. 内存池效率测试**

**3.1 测试1：连续申请和连续释放**

　　分别用内存池和new连续申请和连续释放大量的内存块，比较其运行速度，代码如下：

http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include "stdafx.h"  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <iostream>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <ctime>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <vector>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/object\_pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace std;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace boost;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifconst int MAXLENGTH = 100000;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifint main ( )  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    boost::pool<> p(sizeof(int));  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    int\* vec1[MAXLENGTH];  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    int\* vec2[MAXLENGTH];  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        vec1[i] = static\_cast<int\*>(p.malloc());  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        p.free(vec1[i]);  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        vec1[i] = NULL;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"程序运行了 "<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        vec2[i] = new int;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        delete vec2[i];  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        vec2[i] = NULL;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"程序运行了 "<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    return 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockEnd.gif}

 　　测试环境：VS2008，WindowXP SP2，Pentium 4 CPU双核，1.5GB内存。



　　结论：在连续申请和连续释放10万块内存的情况下，使用内存池耗时是使用new耗时的47.46%。

**3.2 测试2：反复申请和释放小块内存**

　　代码如下：

http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include "stdafx.h"  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <iostream>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <ctime>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <vector>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/object\_pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace std;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace boost;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifconst int MAXLENGTH = 500000;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifint main ( )  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    boost::pool<> p(sizeof(int));  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        int \* t = static\_cast<int\*>(p.malloc());  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        p.free(t);  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"程序运行了 "<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        int\* t = new int;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        delete t;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"程序运行了 "<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    return 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockEnd.gif}

 　　测试结果如下：



　　结论：在反复申请和释放50万次内存的情况下，使用内存池耗时是使用new耗时的64.34%。

**3.3 测试3：反复申请和释放C++对象**

　　C++对象在动态申请和释放时，不仅要进行内存操作，同时还要调用构造和析购函数。因此有必要对C++对象也进行内存池的测试。

　　代码如下：

http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include "stdafx.h"  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <iostream>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <ctime>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <vector>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/object\_pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace std;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace boost;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifconst int MAXLENGTH = 500000;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifclass A  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gifpublic:   
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    A()  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        m\_i++;   
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    ~A( )  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        m\_i--;   
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gifprivate:  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    int m\_i;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockEnd.gif};  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifint main ( )  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    object\_pool<A> q;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        A\* a = q.construct();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        q.destroy(a);  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"程序运行了 "<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; i < MAXLENGTH; ++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        A\* a = new A;   
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        delete a;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"程序运行了 "<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    return 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockEnd.gif}

 　　测试结果如下：



　　结论：在反复申请和释放50万个C++对象的情况下，使用内存池耗时是使用new耗时的112.03%。这是因为内存池的construct和destroy函数增加了函数调用次数的原因。这种情况下使用内存池并不能获得性能上的优化。

**4. Boost内存池的分类**

　　Boost内存池按照不同的理念分为四类。主要是两种理念的不同造成了这样的分类。

　　一是Object Usage和Singleton Usage的不同。Object Usage意味着每个内存池都是一个可以创建和销毁的对象，一旦内存池被销毁则其所分配的所有内存都会被释放。Singleton Usage意味着每个内存池都是一个被静态分配的对象，直至程序结束才会被销毁，这也意味着这样的内存池是多线程安全的。只有使用release\_memory或者 purge\_memory方法才能释放内存。

　　二是内存溢出的处理方式。第一种方式是返回NULL代表内存池溢出了;第二种方式是抛出异常代表内存池溢出。

　　根据以上的理念，boost的内存池分为四种。

**4.1 Pool**

　　Pool是一个Object Usage的内存池，溢出时返回NULL。

**4.2 object\_pool**

　　object\_pool与pool类似，唯一的区别是当其分配的内存释放时，它会尝试调用该对象的析购函数。

**4.3 singleton\_pool**

　　singleton\_pool是一个Singleton Usage的内存池，溢出时返回NULL。

**4.4 pool\_alloc**

　　pool\_alloc是一个Singleton Usage的内存池，溢出时抛出异常。

**5. 内存池溢出的原理与解决方法**

**5.1 必然溢出的内存**

　　内存池简化了很多内存方面的操作，也避免了一些错误使用内存对程序造成的损害。但是，使用内存池时最需要注意的一点是要处理内存池溢出的情况。

　　没有不溢出的内存，看看下面的代码：

http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include "stdafx.h"  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <iostream>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <ctime>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <vector>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/object\_pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace std;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace boost;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifint \_tmain(int argc, \_TCHAR\* argv[])  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    int iLength = 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; ;++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        void\* p = malloc(1024\*1024);  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        if (p == NULL)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif            break;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        ++iLength;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"共申请了"<<iLength<<"M内存,程序运行了"<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    return 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockEnd.gif}

 　　运行的结果是“共申请了1916M内存,程序运行了 69421 个系统时钟”，意思是在分配了1916M内存后，malloc已经不能够申请到1M大小的内存块了。

　　内存池在底层也是调用了malloc函数，因此内存池也是必然会溢出的。而且内存池可能会比直接调用malloc更早的溢出，看看下面的代码：

http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include "stdafx.h"  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <iostream>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <ctime>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <vector>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/object\_pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace std;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace boost;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifint \_tmain(int argc, \_TCHAR\* argv[])  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    boost::pool<> pl(1024\*1024);  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    int iLength = 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; ;++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        void\* p = pl.malloc();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        if (p == NULL)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif            break;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        ++iLength;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"共申请了"<<iLength<<"M内存,程序运行了"<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    return 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockEnd.gif}

 　　运行的结果是“共申请了992M内存,程序运行了 1265 个系统时钟”，意思是在分配了992M内存后，内存池已经不能够申请到1M大小的内存块了。

**5.2 内存池的基本原理**

　　从上面的两个测试可以看出内存池要比malloc溢出早，我的机器内存是1.5G，malloc分配了1916M才溢出(显然分配了虚拟内存)，而内存池只分配了992M就溢出了。第二点是内存池溢出快，只用了1265微秒就溢出了，而malloc用了69421微秒才溢出。

　　这些差别是内存池的处理机制造成的，内存池对于内存分配的算法如下，以pool内存池为例：

　　1. pool初始化时带有一个块大小的参数memSize，那么pool刚开始会申请一大块内存，例如其大小为32\*memSize。当然它还会申请一些空间用以管理链表，为方便述说，这里忽略这些内存。

　　2. 用户不停的申请大小为memSize的内存，终于超过了内存池的大小，于是内存池启动重分配机制;

　　3. 重分配机制会再申请一块大小为原内存池大小两倍的内存(那么第一次会申请64\*memSize)，然后将这块内存加到内存池的管理链表末尾;

　　4. 用户继续申请内存，终于又一次超过了内存池的大小，于是又一次启动重分配机制，直至重分配时无法申请到新的内存块。

　　5. 由于每次都是两倍于原内存，因此当内存池大小达到了992M时，再一次申请就需要1984M，但是malloc最多只能申请到1916M，因此malloc失败，内存池溢出。

　　通过以上原理也可以理解为什么内存池溢出比malloc溢出要快得多，因为它是以2的指数级来扩大内存池，真正调用malloc的次数约等于log2(1916)，而malloc是实实在在进行了1916次调用。所以内存池只用了1秒多就溢出了，而malloc用了69秒。

**5.3 内存池溢出的解决方法**

　　对于malloc造成的内存溢出，一般来说没有太多办法可想。基本上就是报一个异常或者错误，然后让用户关闭程序。当然有的程序会有内存自我管理功能，可以让用户选择关闭一切次要功能来维持主要功能的继续运行。

　　而对于内存池的溢出，还是可以想一些办法的，因为毕竟系统内存还有潜力可挖。

　　第一个方法是尽量延缓内存池的溢出，做法是在程序启动时就尽量申请最大的内存池，如果在程序运行很久后再申请，可能OS因为内存碎片增多而不能提供最大的内存池。其方法是在程序启动时就不停的申请内存直到内存池溢出，然后清空内存池并开始正常工作。由于内存池并不会自动减小，所以这样可以一直维持内存池保持最大状态。

　　第二个方法是在内存池溢出时使用第二个内存池，由于第二个内存池可以继续申请较小块的内存，所以程序可继续运行。代码如下：

http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include "stdafx.h"  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <iostream>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <ctime>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <vector>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif#include <boost/pool/object\_pool.hpp>  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace std;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifusing namespace boost;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/None.gifint \_tmain(int argc, \_TCHAR\* argv[])  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    boost::pool<> pl(1024\*1024);  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    int iLength = 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; ;++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        void\* p = pl.malloc();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        if (p == NULL)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif            break;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        ++iLength;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_t clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"共申请了"<<iLength<<"M内存,程序运行了"<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_begin = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    iLength = 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    boost::pool<> pl2(1024\*1024);  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    for (int i = 0; ;++i)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        void\* p = pl2.malloc();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        if (p == NULL)  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif            break;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif        ++iLength;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    clock\_end = clock();  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    cout<<"又申请了"<<iLength<<"M内存,程序运行了"<<clock\_end-clock\_begin<<" 个系统时钟"<<endl;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/InBlock.gif    return 0;  
http://tech.it168.com/CodeHighlighter/Images/OutliningIndicators/ExpandedBlockEnd.gif}

 　　运行结果如下：

http://image20.it168.com/201107_0x0/708/db215286e02e301a.jpg

　　结果表明在第一个内存池溢出后，第二个内存池又提供了480M的内存。

**5.4 内存池溢出的终极方案**

　　如果无论如何都不能再申请到新的内存了，那么还是老老实实告诉用户重启程序吧。