
Subject: Re: Value question (memory consumption)
Posted by [mirek](#) on Tue, 30 Nov 2010 12:36:13 GMT
[View Forum Message](#) <> [Reply to Message](#)

andreincx wrote on Mon, 29 November 2010 16:57Memory consumption is higher with U++ allocator. I've tested with USEMALLOC flag and the memory consumption was half, but has same behavior, only part of the memory is released.

I've modified example to do new allocations and seems that memory get back to OS, at least here on Linux, if i don't use U++ allocator.

```
#include <Core/Core.h>
```

```
using namespace UPP;
```

```
#define ITEM_COUNT 1000000
```

```
CONSOLE_APP_MAIN {  
    Vector<Value> v;  
    getchar();  
    for(int i=0;i<ITEM_COUNT;i++) v.Add((int)i);  
    getchar();  
    v.Clear(); v.Shrink();  
    getchar();  
    for(int i=0;i<ITEM_COUNT;i++) v.Add((int)i);  
    getchar();  
    v.Clear(); v.Shrink();  
    getchar();  
}
```

without U++ allocator:

```
292 KiB  
38.4 MiB  
30.8 MiB  
38.8 MiB  
420 KiB
```

with U++ allocator:

```
412 KiB  
34.5 MiB (for a sec.) 70.2 MiB maybe Vector double amount of memory it need  
62.5 MiB  
70.2 MiB  
62.5 MiB
```

Linux 2.6.35-23-generic #40-Ubuntu SMP Wed Nov 17 22:14:33 UTC 2010 x86_64 GNU/Linux

Andrei

In Win32, memory consumption estimated using task manager, it goes like:

USEMALLOC:

724KB
28.2MB
1.1MB
28.2MB
1.2MB

With U++ allocator:

624KB
20.4MB
16.5MB
20.4MB
16.5MB

So the maximum usage is significantly less than for M\$ allocator..

I would like to know why Linux is so much higher.

If it is not because of Debug release, we should check the correctness of core allocation routines in POSIX:

```
void *SysAllocRaw(size_t size)
{
    sKB += int(((size + 4095) & ~4095) >> 10);
#ifdef PLATFORM_WIN32
    void *ptr = VirtualAlloc(NULL, size, MEM_RESERVE|MEM_COMMIT, PAGE_READWRITE);
#else
#ifdef PLATFORM_LINUX
    void *ptr = mmap(0, size, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS,
-1, 0);
#else
    void *ptr = mmap(0, size, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANON, -1, 0);
#endif
    if(ptr == MAP_FAILED)
        ptr = NULL;
#endif
    if(!ptr)
        Panic("Out of memory!");
    DoPeakProfile();
    return ptr;
}
```

```
void SysFreeRaw(void *ptr, size_t size)
{
    sKB -= int(((size + 4095) & ~4095) >> 10);
#ifdef PLATFORM_WIN32
```

```
VirtualFree(ptr, 0, MEM_RELEASE);
#else
munmap(ptr, size);
#endif
}
```

```
int s4kb__;
int s64kb__;
```

```
void *AllocRaw4KB()
{
static int left;
static byte *ptr;
static int n = 32;
if(left == 0) {
left = n >> 5;
ptr = (byte *)SysAllocRaw(left * 4096);
}
n = n + 1;
if(n > 4096) n = 4096;
void *p = ptr;
ptr += 4096;
left--;
s4kb__++;
DoPeakProfile();
return p;
}
```

```
void *AllocRaw64KB()
{
static int left;
static byte *ptr;
static int n = 32;
if(left == 0) {
left = n >> 5;
ptr = (byte *)SysAllocRaw(left * 65536);
}
n = n + 1;
if(n > 256) n = 256;
void *p = ptr;
ptr += 65536;
left--;
s64kb__++;
DoPeakProfile();
return p;
}
```

(the rest is the same for Win32, POSIX):

Mirek
