
Subject: Re: Core 2019

Posted by [Novo](#) on Fri, 21 Jun 2019 15:59:14 GMT

[View Forum Message](#) <> [Reply to Message](#)

mirek wrote on Fri, 21 June 2019 03:16

Now what is different is that standard GCC allocator has threshold at 4MB. U++ allocator at 224MB. In practice, this means that if you alloc / free 5MB block in std, it gets released back to system immediately. With U++, blocks up to 224 MB are not returned to the system immediately. If they are really unused, this just means that system will retrieve them when there is a need for more physical memory.

Mirek

This info doesn't match what I'm reading in the docs I posted above.

The lower limit for this parameter is 0. The upper limit is
DEFAULT_MMAP_THRESHOLD_MAX: 512*1024 on 32-bit systems or
4*1024*1024*sizeof(long) on 64-bit systems.

Note: Nowadays, glibc uses a dynamic mmap threshold by default. The initial value of the threshold is 128*1024, but when blocks larger than the current threshold and less than or equal to DEFAULT_MMAP_THRESHOLD_MAX are freed, the threshold is adjusted upward to the size of the freed block. When dynamic mmap thresholding is in effect, the threshold for trimming the heap is also dynamically adjusted to be twice the dynamic mmap threshold. Dynamic adjustment of the mmap threshold is disabled if any of the M_TRIM_THRESHOLD, M_TOP_PAD, M_MMAP_THRESHOLD, or M_MMAP_MAX parameters is set.

So, the old allocator on 64-bit systems had threshold 32Mb (4*1024*1024*sizeof(long)). In the new one it is determined dynamically. Initial value is 128Kb.

I traced my app with a tool which intercepts all malloc/free calls. I know exactly what is stressing the allocator :roll:

Memory block sizes:

File Attachments

1) [Screenshot_2019-06-21_11-51-00.png](#), downloaded 729 times
