Debugging: Diagnosing why malloc is failing

devblogs.microsoft.com/oldnewthing/20140214-00

February 14, 2014

Raymond Chen

A customer had some code which was experiencing memory allocation failures when calling `malloc` (which maps to `HeapAlloc`). The function returns `nullptr`, and `GetLastError()` reports `ERROR_NOT_ENOUGH_MEMORY`. However, there was still plenty of memory free:

- Task Manager reported working set at around 400MB, with a peak of 550MB.
- Using the `_heapwalk` function to compute the total memory used resulted in about 380MB being reported.
- The `_heapchk` function reported no errors.
- The virtual memory size for the process was a little bit more than the working set size.

The customer was continuing their investigation but was looking for some pointers since the bug took a day to emerge. Could it be heap fragmentation? (The program is uses the regular C runtime heap and does not enable the low-fragmentation heap.) One of the suggestions was to run the VMMap utility to see if the problem was exhaustion of virtual address space. And lo and behold, that was indeed the cause. The code had a bug where it was leaking threads. Since the default stack reservation for a thread is 1MB (although typically only a tiny fraction of that ends up being committed and even less being charged against working set), a slow accumulation of threads corresponds to a slow erosion of the virtual address space until you eventually run out.

Once again, it's the address space, stupid.

Raymond Chen

Follow