CS 481: Operating Systems
Lab 2 Part 2

Jacob Hurst

April 11, 2019

1 Hypothesis

The Linux kernel lazily manages the page tables of processes. When a process requests some area of memory via malloc or brk system calls, the kernel will agree or disagree to allocate that memory but it will not honor that request immediately. Instead, the kernel lazily records what has been agreed upon and waits to allocate that memory until the process attempts to utilize it.

2 Methods

The laziness of the kernel is reflected by the size of page table entries (PTE). When a process requests more memory, the kernel simply updates the heap virtual memory area (VMA). No new page table entries are present in memory at this point. Once the process attempts to access the pages, the processor page faults. In response, the kernel either allocates the area accessed or generates a segmentation fault depending on whether it has agreed or disagreed to the earlier request.

To verify this claim, we analyze the systems memory (via /proc/meminfo), focusing specifically on the number of page table entries before a large (6 GB) malloc request, after that malloc request but before attempting to access data, and after attempting to access data.

To accomplish this, C code was written which mallocs 6 GB of memory and then randomly writes to the data. While the program runs, python code was written to measure the results.

2.1 Factors

1. Request size: How much memory the process should request, 6 GB was requested.
2. Sample count: The number of sample runs evaluated, a total of 30 runs were evaluated.

2.2 Response

1. PageTables: Amount of memory dedicated to the lowest level of page tables (in kB).
2. MemFree: The amount of physical memory not used by the system (in kB).

2.3 Treatments & Repetitions

30 sample runs are evaluated to ensure reproducible results. In each sample, a baseline C program was also evaluated. The baseline C program requests 60 bytes of memory.

3 Results

The figure below contains a sample of the output generated during the experiment. The top half contains diff results on multiple samples while the bottom half contains the average values from the samples.

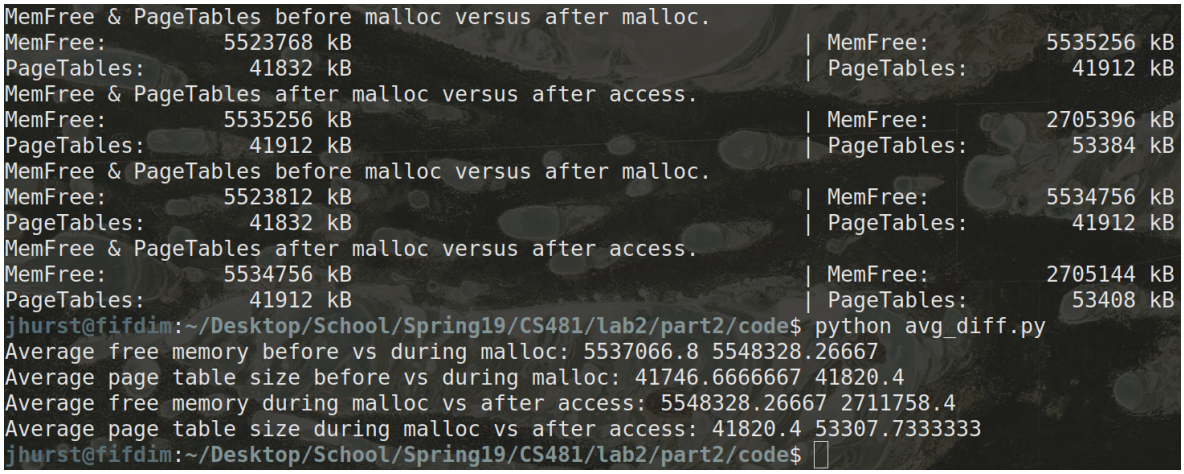

Figure 1: Output from the experiment, top half contains diff results on multiple samples and bottom half contains average results from the samples.

In the figure below, you may notice that the free memory in the system is much greater before malloc and between malloc and accessing the requested data than it is after accessing the requested data.

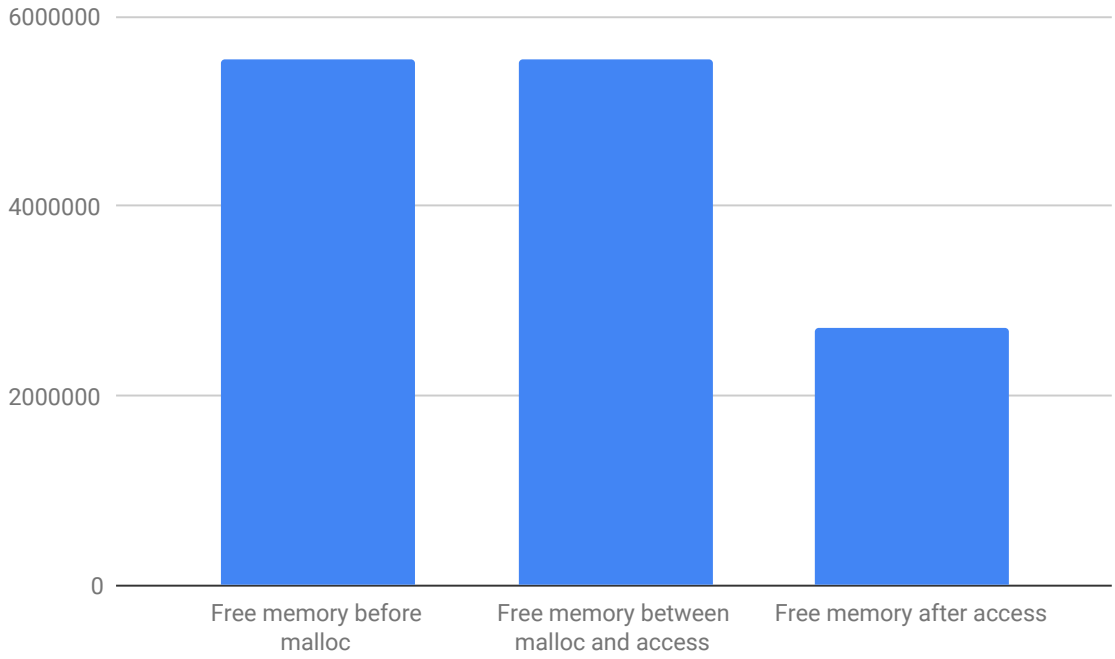

Figure 2: A chart depicting the average free system memory before malloc, between malloc and access, and after access.

In the figure below, you may notice that the total page table size in the system is lesser before malloc and between malloc and accessing the requested data than it is after accessing the requested data.

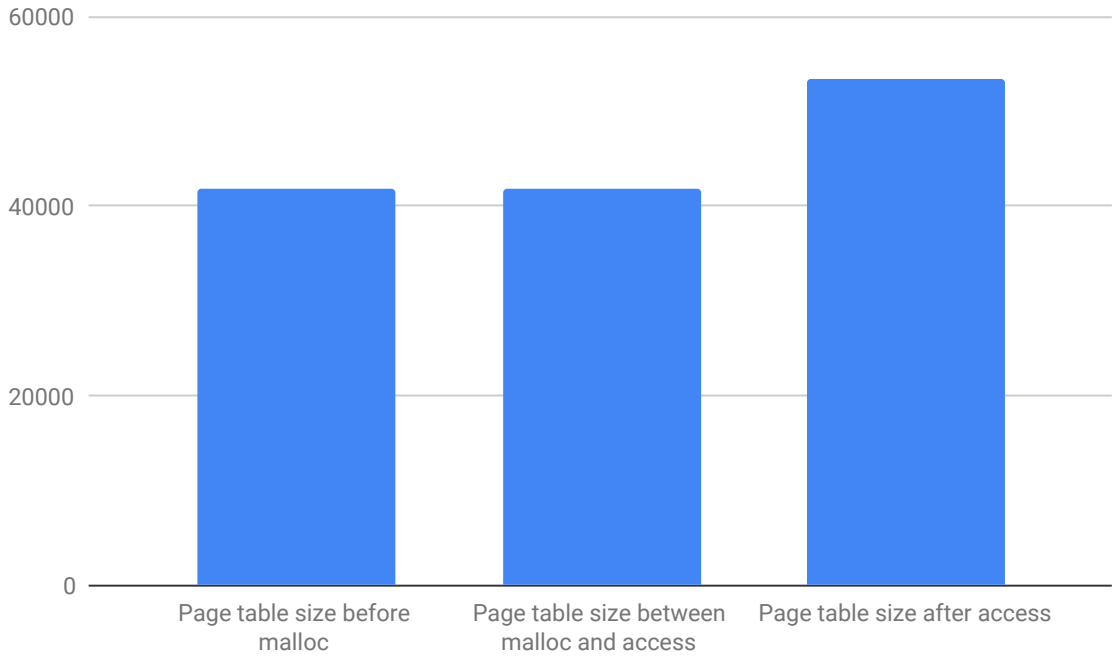

Figure 3: A chart depicting the average total page table size for the system before malloc, between malloc and access, and after access.

From this data, we gain insights on the lazy behavior of the Linux kernel. The Linux kernel uses lazy allocation of physical pages, deferring the allocation until necessary and avoiding allocating physical pages which will never actually be used. This helps prevent memory from being used up by processes that would request more than they use.