Keith Scmitt

Andrew Lawton

CIS 452 – Lab 10

1. Please specify which EOS workstation you are working on

EOS 11.

1. Use the free memory utility program to determine:
   1. the total amount of physical memory (KB) on your system: 16316880 KB
   2. the current amount of free memory (KB): 10693656 KB
2. What is your estimate of the approximate memory demand of the Sample Program?

16384 KB

1. Approximately how much does the amount of free (idle) memory change?

The memory changes approximately by 16000KB

1. Considering your computed memory demands of the Sample Program, explain why the observed change is a predictable result.

Because the malloc() calculated ~16000 KB to be allocated for the program, during run time that memory would be allocated and used as we saw from vmstat.

1. Describe and justify your choice of parameter. You can set dim directly, or change the constant '2'.

We chose to change the constant modifier to 50. We made this choice by using the malloc() calculation we used before while changing the constant until we reached a value smaller than our total free memory.

1. Given your computations and the results from experiment 2 above, is this what you expected to see? Why / why not?

Yes, this is what was expected. As the memory was used our free memory dropped until there was almost not enough to run the program.

1. Reference the man page for vmstat to understand what is exactly being displayed. What other memory field(s) change (i.e. swap device, buffers, cache)? Speculate: why might these fields have changed (i.e. explain how the system adapting to the increased memory demand)

The memory fields that changed are swdp, free, buff, so, bi, bo. These might have changed because the system was preparing to solve memory issues and increased the amount of virtual memory used as the free memory got close to full. For buffer as we decrease the total memory available buffer also dropped, this was also viewable in the cache.

1. What is the difference between a major and a minor page fault in Linux (you might need to look this up)?
   1. A minor page fault is when the code or data needed is already in memory but isn't allocated to the required process. A major page fault is when the page fault handler is trying to reclaim physical pages that were swapped out earlier due to memory shortage or when the same memory is attempted to be read into the physical memory.
2. What is the size of a page in Linux?
   1. 4096 Bytes
3. Precisely, how does this change alter the program's memory access pattern (i.e. what objects get "touched", in what order)?

The order of the objects touched is j → dim → i → count

1. Speculate: how will this change affect the program's execution time (verify your answer by executing and timing the program)?

Because we are accessing j first, the cache is biased towards predicting values for that. So when the calculation for the entire expression changes dramatically because of I, that will result in cache misses. This makes the cpu have to go into more expensive memory to get the correct value of the array.

1. Precisely, why does the change have this effect (your answer should incorporate an important concept discussed in class)?
   1. We observe this effect due to the attempt to store the page spatially local data when we access a certain memory location. Since we are mixing up the order of this, we will get more page faults because of the way we are accessing each element of the array.
2. Describe and justify your computation.
   1. (1024\*55)^2\*4/(2^10) = 12390400. This exceeds the total amount of free memory in our system.
3. Observe vmstat system statistics as the program executes. Describe what fields have changed (including non-memory fields), and how they have changed?

Swpd changed as the system required the use of virtual memory to hold the information. Buff changed as the memory was allocated the buffer was used. Cache overall decreased in size as the loops demanded more memory. In increased as the program executed, and returned to normal upon completion.

1. Describe what has happened and why (i.e. explain how the operating system is adapting to the increased memory demands of the Sample Program). Include a brief discussion of the execution time and number of page faults incurred. Your explanation should demonstrate that you understand what is happening from a virtual memory viewpoint.

As the memory fills, the CPU is working harder to find where it can fit the memory into the system. The execution time increased from 10 seconds to 43 seconds and there were no major page faults, but 112492 minor page faults. Because the Malloc() was successful before any looping began, we know that the system has the required memory needed to complete the program, however, the memory isn't being brought into RAM until it is needed. The malloc'd memory exists within the virtual memory and as the program executes the memory is brought into physical RAM. This allows us to overcap our physical memory as long as we don't exceed the total memory.