Sebastian Torrejon Alonzo & Fabian Kirberg Nov 9, 2023
Lab 10 CIS 452 02

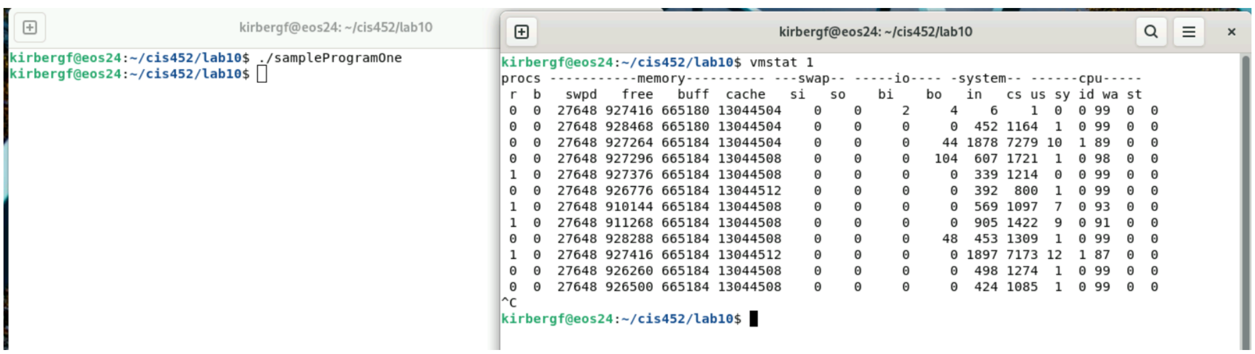
1. **Determine your system configuration:**

   - **Specify what eos system you are working on**
     - **use the free memory utility program to determine and report:**
       - **the total amount of physical memory (KB) on your system**
       - **the current amount of free memory (KB)**
       - **the total amount of swap space**

```
kirbergf@eos24: ~
kirbergf@eos24:~$ free -k
               total        used        free      shared  buff/cache   available
Mem:        16050312     1395344      961536      138020    13693432    14174400
Swap:        2097148       27648     2069500
kirbergf@eos24:~$
```

Eos system: eos24
Total amount of physical memory: 16,050,312 KB
Current amount of physical memory: 961,536 KB
Total amount of swap space: 2,097,148 KB

2. **Examine and observe the memory demand of an executing process:**
   - **Study the Sample Program**
     - **What is your estimate of the approximate memory demand of the Sample Program?**
   - **Start vmstat, make your window appropriately wide, and configure it to display statistics once per second; let the system stabilize**
     - **Note: these experiments are best performed on a "quiet" system (i.e., not many active users)**
   - **In another window, execute the Sample Program**
     - **approximately how much does the amount of free (idle) memory change?**
     - **considering your estimated memory demand of the Sample Program (question 2a), explain why the observed change is an expected result.**



Estimate of the approximate memory demand:
intPtr is malloced (dim * dim * sizeof(int)) and dim = COEFFICIENT * KB.

COEFFICIENT = 2, KB = 1024, and sizeof(int) = 4

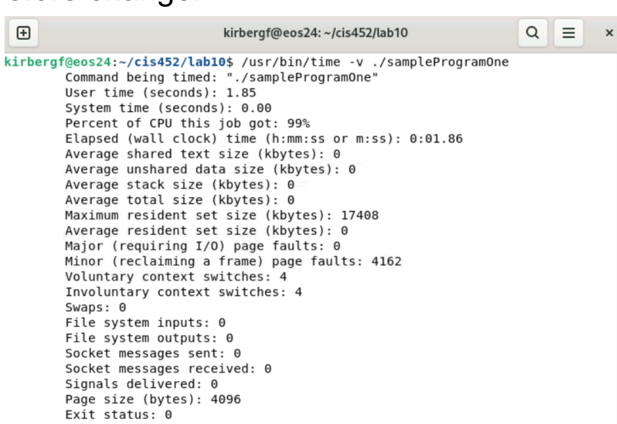
Therefore, the approximate memory demand is 2048 * 2048 * 4 - 16,777,216 bytes.

Amount of free memory change:
16,632 KB as on the 7th line the sample program is executed and the and the amount of free memory is reduced by 16,632 KB.

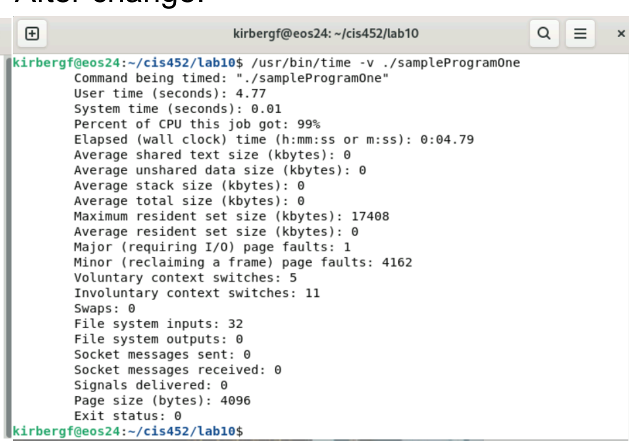
Observer change as expected result:
As 16,777,216 Bytes is equal to 16,384 KB, the observed change is expected as it is approximately the amount of memory that sampleProgramOne was expected to use for its execution.

3. **Examine the effect of memory access patterns:**
    - **Read the man pages for the time utility program. Then use /usr/bin/time together with command-line arguments as described for time to obtain complete statistics (i.e., run in verbose mode). Execute and time the Sample Program.**
        - **obtain basic statistics**
            - **what is the size of a page in Linux?**
            - **how long does the program take to run?**
    - **Change the memory access statement in the Sample Program to read:**
      `intPtr[j * dim + i] = (i + j) % count;`
      then re-compile and re-run the program, collecting statistics on execution time.
        - **Precisely, how does this change alter the program's memory access pattern (i.e. what memory objects get "touched", and in what order)? A diagram will help here.**
        - **How does this change affect the program's (user) execution time?**
        - **Precisely, why does the change have the observed effect (your answer must incorporate an important concept related to paging and virtual memory)?**

Before change:



After change:



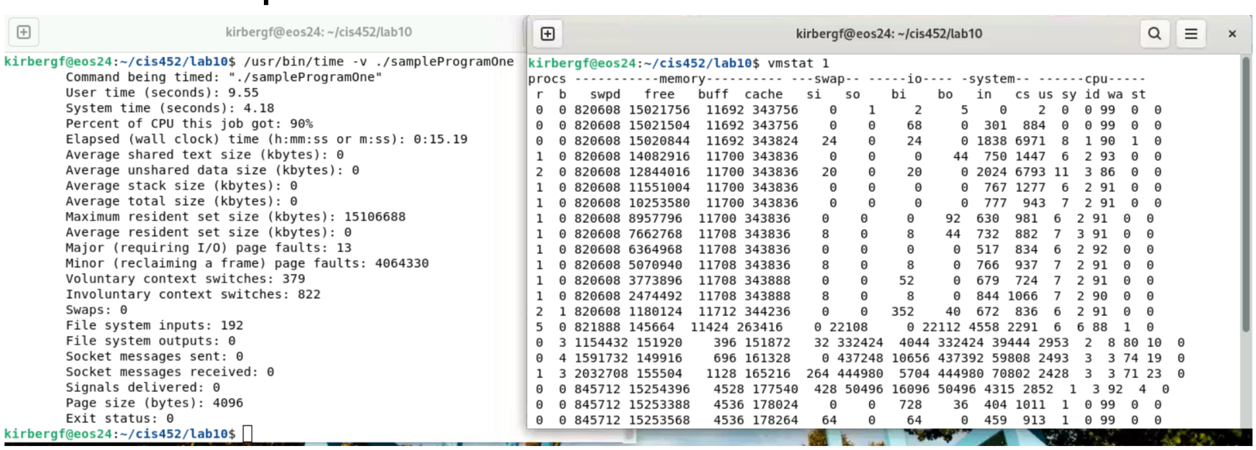
Size of page: 4096 bytes
Run time before change: 1.85 seconds
Change in memory access pattern and diagram:
After the change, the memory objects now get "touched" column by column ([0,0], [1,0], [2,0], etc.), whereas previously it was row by row ([0,0], [0,1], [0,2], etc.).

Change in execution time: Increased to 4.77 seconds

Change as the observed effect:

The change has this effect because having the memory accessed column by column results in memory being accessed “out of order”/in an unsorted manner. This affects the concept of caching, as there is less spatial locality, resulting in a longer execution time.

4. **Examine the use of virtual memory:**
    - **Perform the following**
        - **Change the memory access pattern for the Sample Program back to its original form**
        - **Change the LOOP value to 1 (or be prepared to wait a long time)**
    - **Adjust the COEFFICIENT parameter in the Sample Program to a value that causes the memory demand of the program to exceed the total amount of physical memory on your machine (as determined in question 1 above). Note: memory demand should exceed the amount of RAM on your system, but must be less than (RAM + Swap).**
        - **What value did you use, justify your computation**
    - **Configure and run vmstat to display statistics once every second and use /usr/bin/time in verbose mode to execute and time the program**
        - **Observe vmstat system statistics as the program executes. What happens to the amount of free memory (during and after the run)? Describe all the other fields that have changed (including non-memory fields), and describe why they have changed.**
        - **Explain how the operating system is adapting to the increased memory demand of the Sample Program. Include a brief discussion of the execution time and the number of page faults incurred. Your explanation should demonstrate that you understand what is happening from a virtual memory viewpoint.**



Value:
COEFFICIENT = 63

As the memory demand must exceed the amount of RAM (16050312 KB) but must also be less than the RAM + Swap (16050312 + 2097148 = 18147460 KB).

Using a coefficient of 64 the memory demand of the program is ((1024 * 63) * (1024 * 63) * sizeof(int)) = 16647192576 bytes or 16257024 KB.

Changes:

The amount of free memory gradually decreases during the program execution. After execution the amount of free memory returns to approximately the initial values.

ID	Description	Why it changed
r	# of processes in a running state	Has a sharp increase right before virtual memory used increases because memory is being managed to accommodate the processes
b	# of processes in uninterruptible sleep state	Has a sharp increase during virtual memory usage as these processes are likely related to I/O that are waiting for swapping to finish
swpd	Amount of virtual memory used (swap space)	Has a sharp increase as free/idle memory runs low during program execution
free	Amount of idle memory	Gradually decreases during program execution
buff	Amount of memory used as buffers	Decreases during virtual memory usage as it is being utilized to temporarily hold data
cache	Amount of memory used as cache	Decreases during virtual memory usage as pages are being stored on the cache
si	Amount of memory swapped in from disk	Increases during virtual memory usage as pages are swapped in to physical memory
so	Amount of memory swapped to disk	Increases during virtual memory usage as pages are swapped out

		of physical memory
bi	Blocks received from a block device	Increases during virtual memory usage as blocks are read from disk
bo	Blocks sent to a block device	Increases during virtual memory usage as blocks are written to disk
in	# of interrupts per second	Increased I/O operations used for swapping result in an increase of interrupts
cs	# of context switches per second	Increased I/O operations used for swapping result in an increase of context switches to maximize CPU usage while waiting for I/O operations
us	Time spent running non-kernel code	Increases during swapping due to an increase in tasks
sy	Time spent running kernel code	No substantial changes
id	Time spent idle	No substantial changes
wa	Time spent waiting for I/O	Swapping involves I/O operations which results in increases I/O wait times
st	Time stolen from a virtual machine	No substantial changes