Enamadi Sanath Yashwin Date of Issue : 18.08.2020

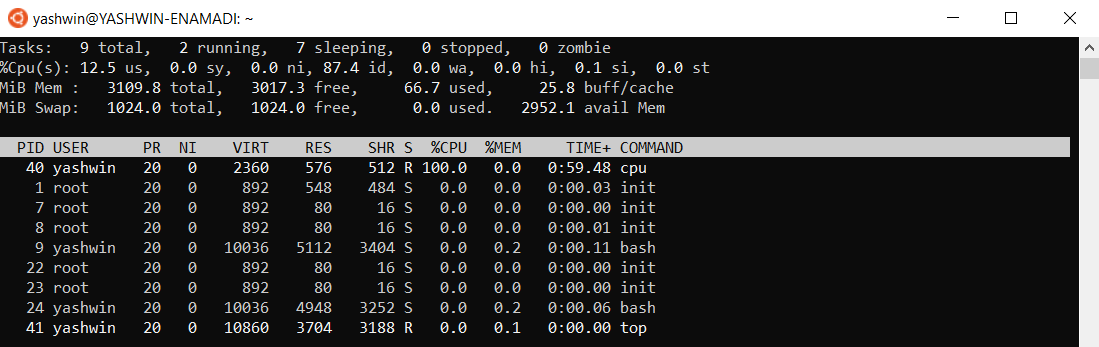
18CS01057 Date of Submission : 23.08.2020

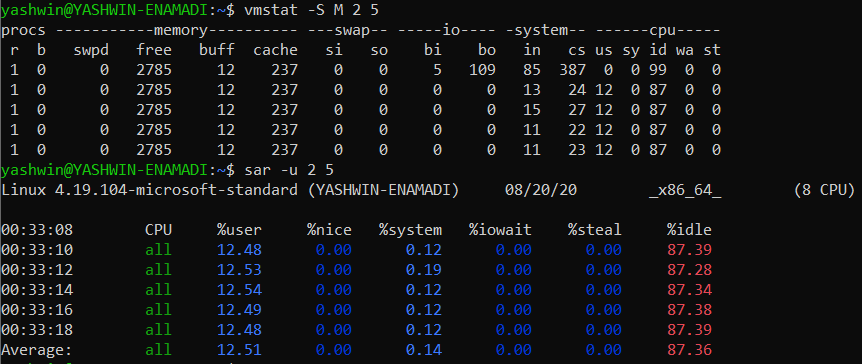
Assignment – 1 : Operating Systems Laboratory

1. The following are known using procfs:

1. My machine has 8 cores. (cat /proc/cpuinfo)
2. Total memory is 3184424 kB of which 2927296 kB is free. (cat /proc/meminfo)
3. 228164 context switches happened since system booted. (cat /proc/stat) ctxt line in the above command gives the number
4. 830 processes have been forked. (cat /proc/stat) processes line in the above command gives the number

2. The following are known using various commands shown:

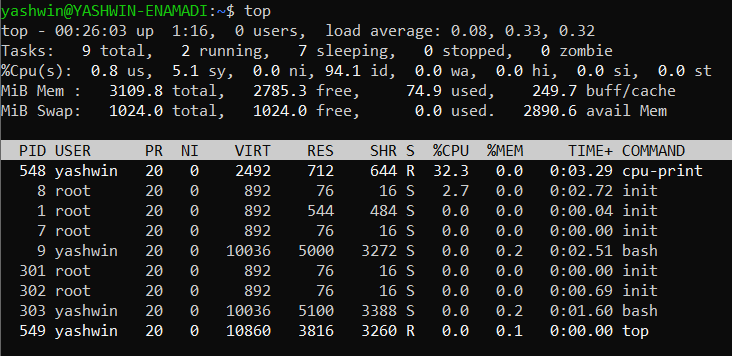
1. cpu : CPU is the bottleneck for this process. The output for the top command is as follows: 

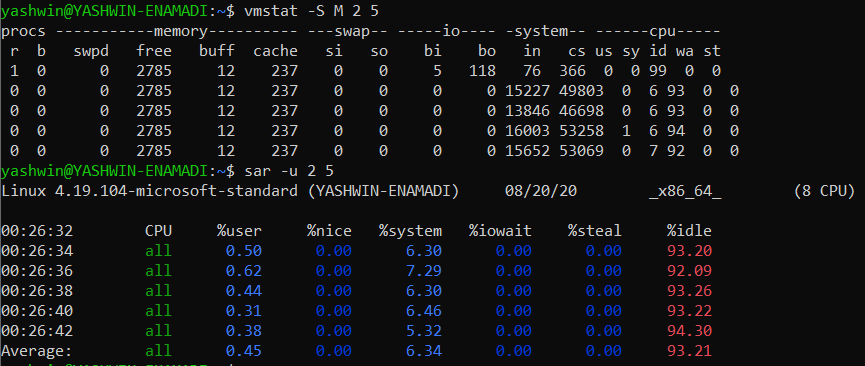


It is clearly evident that cpu(PID : 40) is using 100% of CPU.

As this program is highly computational in nature, without accessing any other resources such as I/O devices and disk, it is quite expected that the bottleneck for this process is CPU.

1. cpu-print :

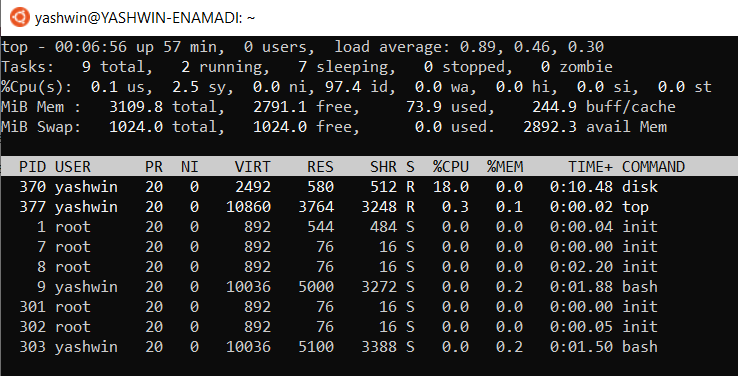


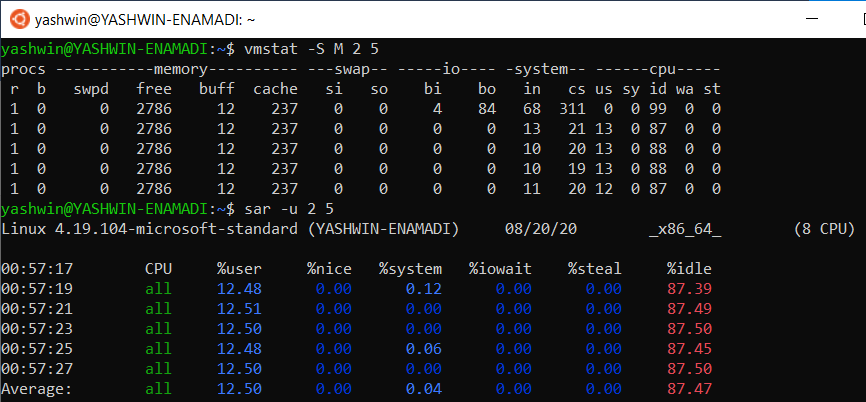


In this case, it is very evident from the output of vmstat cmd that, high number of interrupts and context switches are occurred during the execution. So, the bottleneck for this process is I/O operations. In the cpu section, we also notice CPU is idle for most of the time. Hence, CPU is definitely not the bottleneck for this process.

As the program prints output for every short duration of time, it is quite expected that either CPU or I/O operations must be the bottleneck for this process. But user time is very less for the process. So, it is clear that I/O operations become the bottleneck.

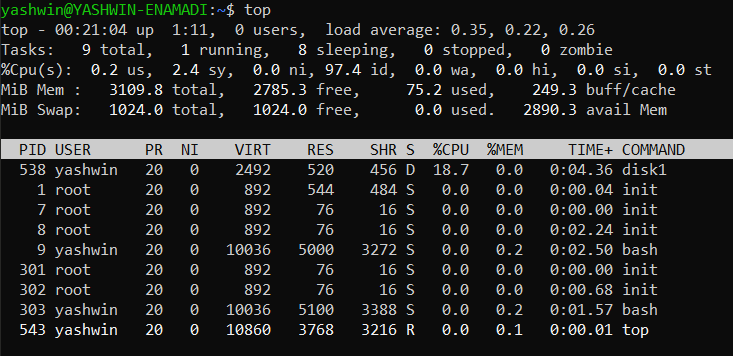
1. disk :

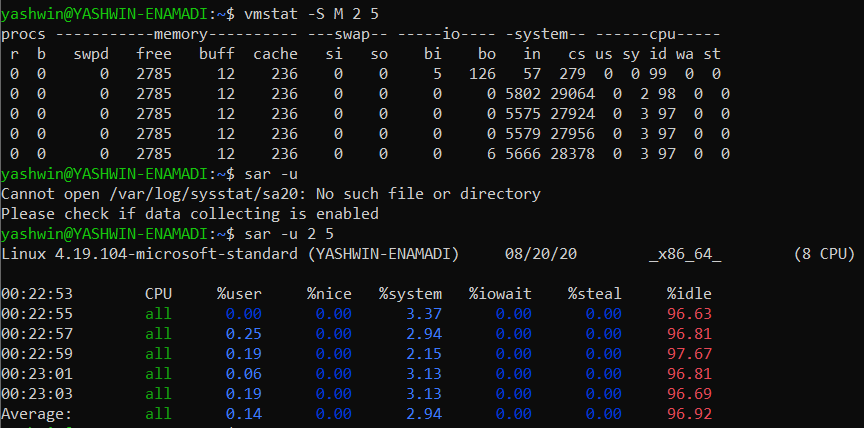




In this process, the disk is the bottleneck. Since, the process accesses the disk for reading every file among 10k files, each of size 2MB, the disk becomes the bottleneck of the process.

1. disk1 :





We see that there are large number of context switches happening, during the execution of the process. These are due to the process accessing the disk quite often. So, disk access is the bottleneck of this process.

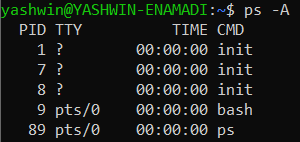
3.

|  |  |
| --- | --- |
| cpu:    User time : 12  Sys time : 0 | cpu-print:    User time : 1  Sys time : 7 |
| disk:    User time : 13  Sys time : 0 | disk1:    User time : 0  Sys time : 3 |

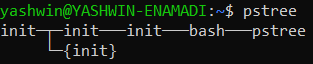
4.

|  |  |
| --- | --- |
| This process (cpu) involves only computation and hence there are significant nonvoluntary context switches as the OS interrupts the process to give space to other processes as well. | This process (disk) reads from the disk regularly. So, whenever it accesses the disk, it voluntarily makes a context switch. Hence, there is no any need for the OS to interrupt the process. |

5. The pid for the bash shell is 9

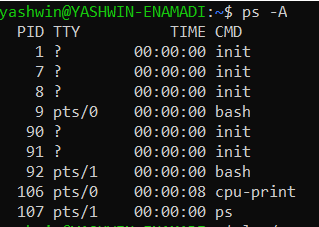


The output for the pstree command is as follows :

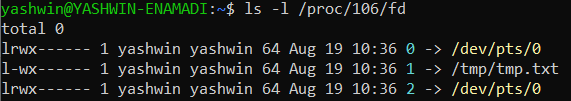


Initially, the init process with pid = 1, is created. It has two children as shown above. Process with pid = 1 has a child with pid = 7 and this inturn has a child with pid = 8 and this child with pid = 8 creates the bash shell (pid = 9)

6. The pid of cpu-print is 106



The file descriptors are as shown



We issued the output redirection command “>” to the file /tmp/tmp.txt. So, the file descriptor 1, which usually points to stdout is changed to point to the mentioned file.

7. Here, we used inter-process communication. The pipe allows the output of the first command (./cpu-print) to be passed on to the input of the second command(grep hello).

The file descriptors are as shown :

