CS 550 : Operating Systems

Assignment 3

Report

By

Prathamesh Nitin Lonkar

Bnumber: B0081727

Part A

- The objective of the first part was to print the virtual address and time at which the page fault has occurred for a particular process, to do this we had to accept a pid from the command line as argument and then compare it with the current pid at which the page fault has occurred.

- There are many page faults occurring within a second for many processes hence we had to filter out the page faults for the required pid, for this we used the current->pid variable as it will contain the pid at which the page fault has occurred.

- Once we confirm that the page fault has occurred at a given pid we needed to access the virtual address at which that page fault occurred and the time at which it occurred. To do so i have used the ‘si’ variable from the pt_regs structure which stores the virtual address of the page fault, and for time I have used the ktime_get() function to store the time.

- This will print all the address and the time at which the page fault has occured.

- I am also string the page fault in a structure for transferring the data to the user space for plotting purpose.

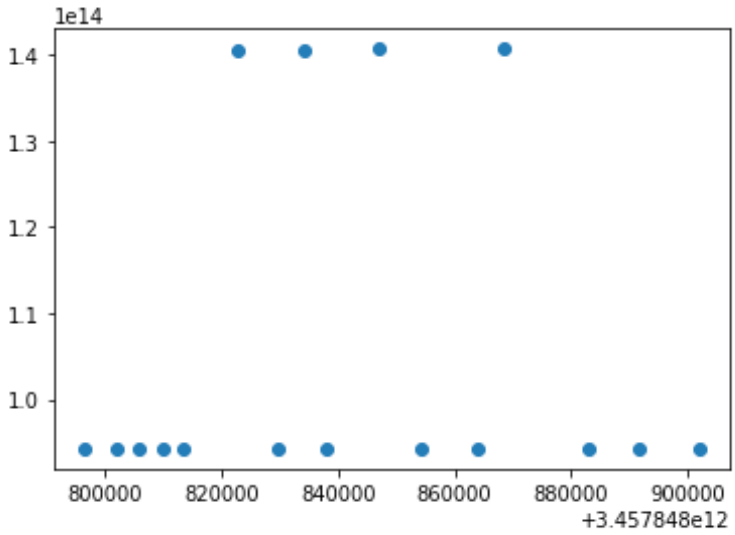
```
struct kernelData{
        int pid;
        unsigned long vaddress;
        long long time;

}kerndata;
```

Part B

- In the part B we are required to scatter plot the data which we recieved in the part A.
- We have to try this for various functions such as kernel compilation (compute and I/O intensive), sysbench (compute intensive), iperf (network I/O intensive) code.
- I have plotted scatter plot's for three functions which are:
  1. bash process running in the kernel:
     - The bash process gave around 40 to 50 page faults at a time, which were mostly located at the higher end or the lower end of the virtual address space as seen below, This behaviour of the page fault might be because of the type of process the bash is, as it it a main kernel function it will give most of the space to this process hence there are no page faults occurring in the middle virtual address.
     - The above functioning might also be the reason why the number of page faults are lower than other process.
     - The plot is as shown below:

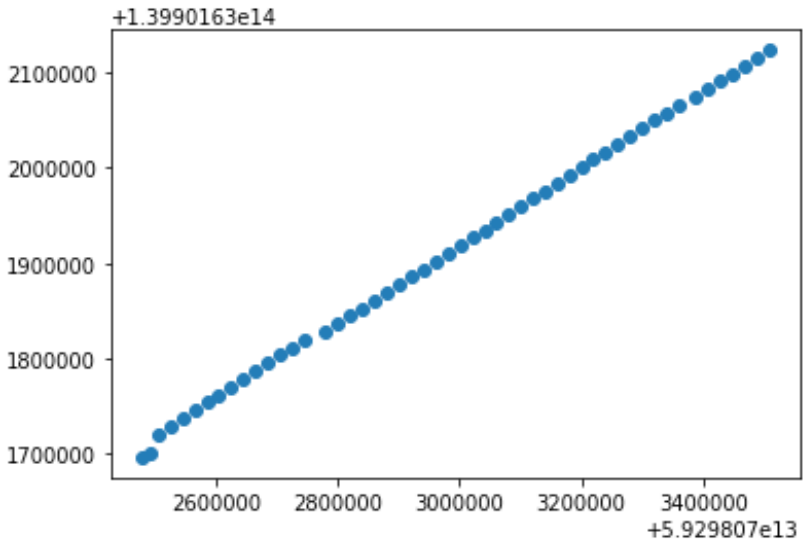
```
Out[26]: <matplotlib.collections.PathCollection at 0x9f3d4847f0>
```



2. The second function was the sysbench(compute intensive):

- For this application I got around 150 to 160 page faults at a time which were much greater than that of the bash, there might be many reasons for this to occurr.
- Firstly as it was a compute intensive process hence required a lot of memory and hence might be constantly removed from the memory and added back hence there is higher chance of tlb miss and page fault.
- Secondly as it is an external application the OS might not be giving it much preference while allocating the memory for it hence this might cause more page faults than other applications.
- The Scatter plot for this operation is as follows:

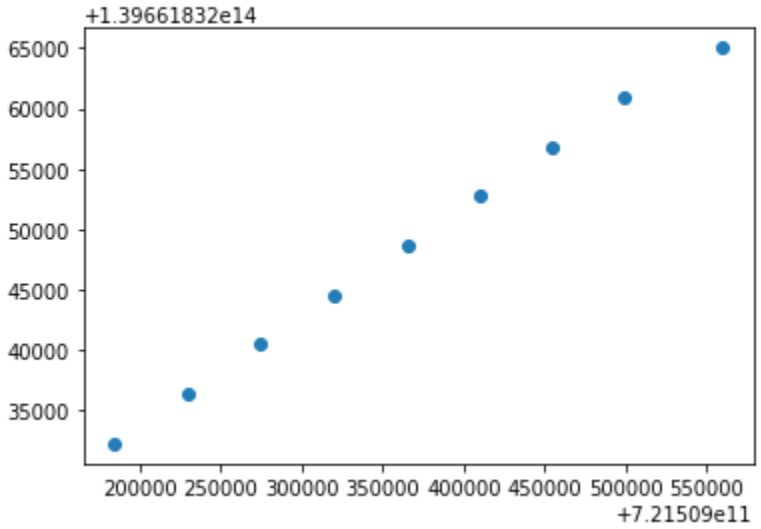
```
Out[27]: <matplotlib.collections.PathCollection at 0x9f3d4de4a8>
```



3. For the third function I have user the iperf3 (network I/O intensive)) process to create to a server and connect from a client to transfer the data packets from client to server and record the page faults during this transfer by passing the pid of the server.

- For this process I had to create a server and a client and transfer data to record the page faults occurring.
- After this process has ended we can see that the number of page fault is depended on the amount of data packets been transferred as I transferred less amount of data packets the number of page faults was also less. But if i was to transfer more number of data packets then there is a chance that the number of page faults also increase.
- As you can see in the plot below that the behaviour of the page faults is similar to that of sysbench but less number of page faults are occurring.
- This might be because I have transferred less number and size of data packets or the Os is giving more preference to the network process.
- The plot is as follows:

```
Out[28]: <matplotlib.collections.PathCollection at 0x9f3d52bbe0>
```



- I have transfered the data to the user space by using the miscellaneous character device read function. I tried performing the transfer using ioctl, but was unable to do so as my system kept getting crashed.
- Hence I am transferring the data to the user space then converting it into a csv file and the plotting the graph using Python’s matplotlib inbuilt library.

References

- https://elixir.bootlin.com/linux/latest/source/samples/kprobes/kprobe_example.c