COL 732 Lab 3

Name: Bommakanti Venkata Naga Sai Aditya
Entry No: 2019CS50471

Part 1 - Page table Benchmark:

Experimental Setup:
I disabled EPT using the commands:
‘sudo modprobe -r kvm_intel’
‘sudo modprobe kvm_intel ept=0’
Then copied .out (which was compiled in host) into the VM using netcat (nc). VM listens on port 1234 and host sends the .out file to the VM on port 1234. VM stores the received input in a file. Then I gave executable permissions to the .out file and ran the file in the VM. Similarly, I enabled EPT and again followed the same procedure. I did the same for different sizes of the array.

EPT = 0

Size of Resident Memory	Time (in s)
4K	9.87
16K	9.42
64K	9.45
256K	10.44
1M	9.54
8M	9.93
16M	9.99
32M	9.94
64M	9.97
128M	10
256M	9.87

EPT = 1

Size of Resident Memory	Time (in s)
4K	1.62
16K	1.64
64K	1.66
256K	1.85
1M	1.87
8M	1.74
16M	1.76
32M	1.73
64M	1.82
128M	1.84
256M	1.72

Observations:
Time taken by fork is more when EPT = 0 when compared to the case when EPT=1. There is not much change in time when we increase the size of the array.

Justifications:
When EPT = 0 (i.e., when we are using Shadow page tables) there will be many VMExits during forks due to the following reasons:

- Whenever Guest OS marks each page of parent Read Only, it causes a trap and VMM changes shadow page tables of parent
- When Guest OS try to point CR3 to child page table, it causes trap, VMM creates shadow page table for child
- When the parent tries to modify a page, it causes a trap for copying the page and another trap to update the shadow page table.

But when EPT = 1 (i.e., when we are using Extended Page tables) it won't cause traps because GVA to GPA mapping is managed by Guest OS, not hypervisor.

So, there won’t be any VMExits. VMExits take a lot of time, so when fork is taking more time when EPT = 0 when compared to the case when EPT = 1

There is not much change in time when we increase size because the pages are copied lazily. Initially only page tables are copied, so time taken is almost the same for different sizes of residential memory.

Part 2 - TLB Benchmark:

Experimental Setup:
I disabled EPT using the commands:
'sudo modprobe -r kvm_intel'
'sudo modprobe kvm_intel ept=0'
Then copied .out (which was compiled in host) into the VM using netcat (nc). VM listens on port 1234 and host sends the .out file to the VM on port 1234. VM stores the received input in a file. Then I gave executable permissions to the .out file and ran the file in the VM. Similarly, I enabled EPT and again followed the same procedure. I did the same for different sizes of the map.

EPT = 0

Size	Time (in ns)	Cycles
4K	2.66	10.4
16K	2.67	10.4
64K	4.48	17.5
256K	7.17	28
1M	13.26	51.7
8M	58.28	227.3
16M	95.61	372.9
32M	102.54	399.9
64M	103.9	405.2
128M	122.52	477.8

EPT = 1

Size	Time (in ns)	Cycles
4K	2.66	10.4
16K	2.67	10.4
64K	4.48	17.5
256K	7.82	30.5
1M	13.37	52.2
8M	61.23	238.8
16M	96.46	376.2
32M	114.39	446.1
64M	126.98	495.2
128M	143.98	561.5

Observations:

The time taken when EPT = 0 is less when compared to the case when EPT = 1. The time taken is increasing as the size of map increases

Justifications:

When EPT = 1 (i.e., when we are using extended page tables), when there is a TLB miss we have to walk through 2 page tables (One pointed by CR3 register, other pointed by EPTP register). Number of memory references made is n + m + n*m (where GVA to GPA is n level page table and GPA to HPA is m level page table). But when EPT = 0, we just have to walk through a single page table pointed by CR3 and there will be less memory accesses. So, time taken is less when EPT = 0. Also, when EPT = 1, we will be maintaining two TLBs one for GPA to HPA, other for GVA to HPA.

As the size of the map is increasing, we have more memory accesses. So, time taken is increasing.