CMPE 283: Virtualization Technologies

Assignment 2: Modifying Instruction Behaviour in KVM

Bhavya Tetali (014535144), Supriya Meduri (015262767)

Contribution of Team Members

Bhavya Tetali:

1. Built the kernel.
2. Researched about atomic variables and cpuid instruction.
3. Understood where to place the measurement code in vmx.c.
4. Wrote the code in **vmx_handle_exit** function in vmx.c and defined global variables.
5. Updated documentation.

Supriya Meduri:

1. Firstly, I rewatched the lecture 5 video.
2. Built the kernel.
3. Tried to understand the assignment requirements of leaf function.
4. Created a CPUID leaf function in **kvm_emulate_cpuid** when %eax=0x4FFFFFFF function in cpuid.c.
5. Created documentation.

Environment Setup:

1. Forked and Cloned the Linux Repository

```
$git clone https://github.com/torvalds/linux.git
```

2. Enter sudo mode

```
$ sudo bash
```

3. Install all the build essentials required to compile

```
$ apt-get install build-essential kernel-package fakeroot
libncurses5-devlibssl-dev ccache bison flex libelf-dev
```

4. Set up the config file

```
$ make menuconfig
```

5. Select kernel-based virtual machine(kvm) support option on the screen prompt

6. Increased the number of processors in the outer VM to eight.
7. Compile and build the kernel

```
$ make -j8 && make modules -j8 && make install -j8 && make
modules_install -j8
```

8. Reboot the system , to get the new kernel

```
$ reboot
```

9. Check the current version of newly built kernel

```
$ uname -a
```

Modification of kernel code:

1. Add the assignment functionality of building a leaf function.
2. In vmx.c, created two global variables : no_of_exits and cpu_cycles.
3. Additionally, in **vmx_handle_exit** function calculated the no of exits using inc function and total time spent processing all exits using rdtsc function.
4. In cpuid.c, created a new cpuid leaf in **kvm_emulate_cpuid** function which reads the no_of_exits into % eax and moves high 32 bits of cpu_cycles into %ebx and low 32 bits of cpu_cycles into %ecx when % eax =0x4FFFFFFF.
5. Else, **kvm_emulate_cpuid** function executes the default code.
6. Compile the code

```
$ sudo make -j modules M=arch/kvm/x86
```

Nested VM Setup:

1. Install virt-manager

```
$ sudo apt-get install virt-manager
$ sudo apt-get install libvirt-bin libvirt-doc
$ sudo apt-get install qemu-system
$ sudo virt-manager
```

2. Download Ubuntu iso image
3. Finish the installation process following all the setup prompts(enable nested VM to prevent further warnings) and configure the inner VM.
4. Build the test code inside the inner VM to test the changes made in the Outer VM kernel and compile it.

Note: After the build, we were not able to login to newly installed kernel module.Tried logging via grub menu and repeated the process multiple times(deleting and reinstalling vm's)