**CMPE283: Virtualization**

**Bonus Lab Assignment: Extending KVM**

Team:

1. Gaurav Chodwadia (010732761)
2. Ekta Sorathia (010717421)

Questions:

1. **Contributions by Team members:**
   * Gaurav Chodwadia
     + With no prior experience with Linux kernel, I started with researching on how to modify, build and install Linux kernel. Also, researched and understood how to work with KVM and understood the sequences of execution in KVM module, especially starting from VM exit to VM entry.
     + I referred excellently organized web pages on the URL below

<http://lxr.free-electrons.com/source/arch/x86/kvm/>

* Question 2
  + Counted nmi\_injected, irq\_injected and exceptions for each vCPU.
* Question 4
  + To get CPU cycles elapsed at the time of VM exit, I put a line of code when control goes in vmx\_handle\_exit function in vmx.c.
  + Got the CPU cycles elapsed just before the return statement of vmx\_handle\_exit and calculated the latency by taking the difference.
  + Stored the latency in the global structure.
* Ekta Sorathia
  + I started with researching on how to store the gathered statistics and output them on request. Initially, thought to use named pipes for passing message across the system, but eventually came up with writing the statistics to kernel logs and display the same using **dmesg** on the console.
  + Accessed the global structure for stats gathering in our external module and implemented switch cases for different Question numbers.
  + Question 1
    - Created an array for all exit types and put that in the global structure.
    - Updated the exit type counter for each exit.
  + Question 3
    - Researched on how to get vCPU information in KVM.
    - Utilized the vCPU objects provided by KVM for each vCPU and add up the exit counts obtained in the Question 1.
    - Marked vCPU create time and current time. Put that in global structure.
    - Calculated the average exits per second for vCPUs.

1. **Recipes to reproduce your development steps:**
2. Install Ubuntu 16.04 LTS in a new VM with the given hardware requirements.
3. Once Host OS (Ubuntu) is installed, open a terminal and type following commands to ensure that virtualization is supported:

sudo apt-get update

sudo apt-get install cpu-checker

sudo kvm-ok

*INFO: /dev/kvm exists*

*KVM acceleration can be used*

1. Install required tools

sudo apt-get -y install cpuid git fakeroot build-essential kernel-package libncurses5-dev l ibssl-dev ccache

sudo apt-get -y install qemu-kvm libvirt-bin bridge-utils virt-manager virtinst virt-viewer

1. Download kernel source and inflate it using command:

tar xvf linux-source.tar.gz

1. Copy config from current ubuntu kernel running

cp /boot/config-`uname -r` .config

1. make menuconfig
   * This opens the GUI of Kernel Configuration
   * Just Save -> ok -> Exit
2. Build kernel using command

fakeroot make-kpkg --initrd --append-to-version=-custom kernel\_image kernel\_headers -j 4

1. Install kernel using command

sudo dpkg -i linux-image\_i386.deb

sudo dpkg -i linux-headers\_i386.deb

1. Remove previous kvm modules

sudo modprobe -vr kvm-intel

sudo modprobe -vr kvm

1. Install new kvm modules

sudo make modules\_install SUBDIRS=arch/x86/kvmINSTALL\_MOD\_DIR=kernel/arch/x86/kvm

1. Insert new modules into kernel

sudo modprobe -v kvm

sudo modprobe -v kvm-intel

sudo modprobe -v kvm-amd

1. Make required changes as stated below:

**Steps to update code**:

1. Use a data structure such that each memory address is only accessed by one vcpu at a time. Separate memory for each vcpu by using an array defined in vmx.c. This can also be achieved by updating struct kvm\_vcpu\_stat defined in .../arch/x86/include/asm/kvm\_host.h
2. Use variables form kvm\_vcpu\_stat to store statistics for each vcpu at the place in functions where exit or events are handled
3. The functions we updated are -
   * 1. vmx.c :: vmx\_create\_vcpu()
     2. vmx.c :: static int vmx\_handle\_exit()
     3. x86.c :: vcpu\_enter\_guest()
     4. asm/kvm\_host.h :: kvm\_vcpu\_stat
4. Add a structure to fetch data gathered in each of these functions to get to custom module.
5. Create EXPORT\_SYMBOL functions in vmx.c that can be called from custom module to fetch statistics. The functions we created are

extern int vcpucounts(void); extern void exportstats(struct cmpe\_stat \*vcpustats, int count);

1. Use following function to get current time in seconds
2. getnstimeofday()
   1. get\_cycles()
3. The data fetched in custom module can be processed and shown to user using printk function.

1. **Usage guide to obtain statistics:**
2. Apply the diff file
3. Install new kernel and kvm module. Use files in root of source
4. Reboot with new kernel
5. Start all VMs probably with different number of vcpus
6. Go to folder src/cmpe283
7. Execute following files to get statistics on console

makemodule.sh

que1.sh

que 2.sh

que 3.sh

que 4.sh