# CSIE 5310 Assignment 3 (Due on May 9th 14:10)

*Don't shout at your JBODs, they don't like it!*
-- Brendan Gregg, *Shouting in the Datacenter*

In the previous two assignments, we have tried running KVM on a virtual Arm hardware emulated by QEMU. In this assignment, you are asked to profile VM workloads, and play with KVM configuration to improve VM performance. You are asked to do a performance measurement study of VM for KVM in Linux v5.15.
Please use the environment you have set up in assignment 1 for this assignment.

## Late submission policy

- 1 pt deduction for late submissions within a day (before May 10th 14:10)
- 2 pts deduction for late submissions within 2 days (before May 11th 14:10)
- zero points for submissions delayed by more than 2 days.

## KVM Host Kernel Configuration

`VHOST_NET` kernel support is required for this assignment, configure the KVM host by doing:

```
# in your KVM source code directory
$ make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- -j30 menuconfig
```

Go to `Device Drivers` -> `Network device support` and turn on `MAC-VLAN support` and `MAC-VLAN based tap driver` (turn on means make it look like `<*>` by hitting space). Then turn on `Device Drivers` -> `VHOST drivers` -> `Host kernel accelerator for virtio net`. After that, recompile your kernel and boot the KVM host.

## Enabling KVM - VM network

The network you get from running `dhclient` does not provide communication between the host and the guest. We can achieve this by utilizing network taps.
Create a network tap in the KVM host with the following commands:

```
# you are free to change the ip address and the tap name (tap0 in this case)
ip tuntap add dev tap0 mode tap
ip link set dev tap0 up
ip addr add 192.168.0.101 brd + dev tap0
ip route add 192.168.0.0/24 dev tap0
```

Modify `run-guest.sh` to give the guest VM another network interface that uses the network tap you just created.

```
# add these two lines at the end of `run-guest.sh`
# the tap name (tap0 in ifname=tap0) must be the name you created
        -netdev tap,id=mytap0,ifname=tap0,script=no,downscript=no,vhost=off \
        -device virtio-net-pci,netdev=mytap0 \
```

Start the guest VM, and run the following to establish the KVM host - VM connection.

```
# you are free to change the ip address
# your interface name "enp0s2" may be different, you can run `dhclient` first
then `ip addr` to check which one is not used
ip addr add 192.168.0.105 brd + dev enp0s2
ip link set dev enp0s2 up
ip route add 192.168.0.0/24 dev enp0s2
```

You should then be able to ping the KVM host and the guest VM from one another.

# Benchmarks

This section describes what benchmarks to run and how to run them.

For networking workloads, you should run the servers on your VM or KVM host, and the client should be your KVM host.

In both your client and server machine (KVM host and VM), download package from the following repo:
https://github.com/chazy/kvmperf

## Test hackbench

Download and compile `hackbench`, and run hackbench:

```
wget https://raw.githubusercontent.com/linux-test-
project/ltp/master/testcases/kernel/sched/cfs-scheduler/hackbench.c
gcc hackbench.c -o hackbench -lpthread
# you can adjust how many times you want to run the benchmark (how many times you
run ./hackbench)
./hackbench 50 process 50
```

## Test kernbench

Download kernel source, install dependencies, and run kernbench:

```
apt install make flex bison
wget http://ck.kolivas.org/apps/kernbench/kernbench-0.50/kernbench
chmod +x kernbench
wget https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.tar.gz
tar xfvz linux-5.15.tar.gz
cd linux-5.15
# the -n option specifies how many runs you would like to perform (default = 5)
../kernbench -M -H -n 1 | tee >(grep 'Elapsed' | awk '{print $3 }' >>
kernbench.txt)
```

## Test netperf

First, download the deb package on your server machines (VM and KVM host).

You should next run `dpkg -i [filename]` on your servers to install the package.

Modify `kvmperf/cmdline_test/netperf.sh` and remove `TCP_MAERTS`:

```
-for _TEST in TCP_MAERTS TCP_STREAM TCP_RR; do
+for _TEST in TCP_STREAM TCP_RR; do
```

Once the netperf server is installed, run `kvmperf/cmdline_test/netperf.sh` on your **client** to test netperf. You should specify the IP of your server (either VM or KVM host) when you run the script.

### Test apache

First, run `kvmperf/cmdline_test/apache_install.sh` on both your server and client.

After the installation completes, run `kvmperf/cmdline_test/apache.sh` on your **client** to test apache. You should specify the IP of your server (either VM or KVM host) and how many times you want to test (the default value of 50 times will take too long for most of the setups) when you run the scripts.
For example

```
kvmperf/cmdline_test/apache.sh 192.168.0.105 3 # server ip = 192.168.0.105, test
three times
```

## Measurement Study Report (10 pts total)

You should submit a report for measurement study.

### VM performance comparison different configurations (6 pts)

You are responsible for tweaking the VM and KVM configurations, and answer the following questions about VM performance.

- Compare VM performance of SMP VM (2 virtual cores) versus UP VM (1 virtual core)
- Compare VM performance using transparent huge page versus regular page (enable transparent huge page on KVM)
- Compare VM performance with vhost versus without vhost (toggle vhost using `vhost=on/off` in `run-kvm.sh`)

Please make sure the number of vCPUs is strictly less than the number of pCPUs for each virtualizing layer. For example, a setup can be a 16-core x86 machine running QEMU that emulates a 4 vCPU ARM system for KVM host to run, with a 2 vcpu VM running within it (16 > 4 > 2).

You should first provide the measurement results using different configurations, and explain what causes the performance difference you observe.

You should use the same v5.15 kernel and config for compiling your KVM host and the guest kernel, except for the `VHOST_NET` configuration.

### VM performance comparison with KVM host (4 pts)

Finally, your report should compare VM performance with KVM host. Here, you should run your SMP VM (2 virtual cores) with a virtio vhost network device. Your KVM host should enable transparent huge page. The goal is to compare VM performance with KVM host running the 4 benchmarks.

The key here is, you should make sure your VM's hardware configuration is comparable with KVM host. For instance, if you test the workloads your VM uses 2 virtual cores with 2GB RAM, your workloads should be tested on the KVM host system with 2 cores and 2GB RAM. To do so, you could modify or supply options for `run-kvm.sh` and `run-guest.sh`.

Your report should compare VM performance with KVM host. First provide the measurement results of your VM and KVM host. Then according to the results, discuss if the VM runs slower or on par with KVM host, then substantiate what may cause the different performance characteristics that you observe.

# Submission

Submit a report named `report-[studentID].pdf` to NTU CooL, e.g. `report-r01234567.pdf`.

# Tips

- Transparent huge page is a kernel config option that you can specify when compiling your Linux kernel. It is by default enabled in the defconfig. You should figure out how to disable the kernel config when testing your VM.
- You may or may not find performance results that are not what you expected, try to analyze them regardless, do not try to manipulate the numbers to fit your expectations.
- (Optional tool) You can find some KVM performance counters in `/sys/kernel/debug/kvm/` on your KVM host.
- (Optional tool) perf-kvm can be used to generate traces for KVM guest OS.