Big picture

In this project, you are going to implement a vCPU scheduler and a memory coordinator to dynamically manage the resources assigned to each guest machine. Each of these programs will be running in the host machine's user space, collecting statistics for each guest machine through hypervisor calls and taking proper actions.

During one interval, the vCPU scheduler should track each guest machine's vCpu utilization, and decide how to pin them to pCpus, so that all pCpus are "balanced". Balance means that each pCpu handles similar workload. The "pin changes" will have overhead, so the vCPU scheduler should try its best to make as few as possible.

During one interval, the memory coordinator should track each guest machine's memory utilization, and decide how much extra free memory should be given to each guest machine. The memory coordinator will set the memory size of each guest machine and trigger the balloon driver to inflate and deflate. The memory coordinator should react properly when the memory resource is insufficient.

Tools that you need or might help you(this works only for ubuntu, will release shortly for RedHat)

1. qemu-kvm, libvirt-bin, libvirt-dev are packages you need to install, so that you can launch virtual machines with KVM and develop programs to manage virtual machines.
2. libvirt is a toolkit prividing lots of APIs to interact with the virtualization capabilities of Linux.
3. Virtualization is a page you should check.
4. virsh, uvtool, virt-top, virt-clone, virt-manager, virt-install are tools that may help you playing with virtual machines.

Deliverables

You need to implement two **separate** C/C++ programs, one for vCPU scheduler(vcpu_scheduler.cc) and another for memory coordinator(memory_coordinator.cc)). Both programs should accept one input parameter, the time interval (in seconds) your scheduler or coordinator will trigger. For example, if we want the vCPU scheduler to take action every 12 seconds, we will start your program using

```
./program_name 12
```

You need to submit one compressed folder containing two **separate** subfolders(cpu and memory), each containing a Makefile and the corresponding codes. We will compile your program using

```
make
```

Grading

This is not performance oriented project, we will only test the functionality.

The Rubric will be:

1. vCPU scheduler functionality - 5 points
2. memory coordinator functionality - 5 points

Step by Step --- Which environment I should use to develop?

1. If you are going to use your own computer, make sure your cpu support hardware virtualization and the Linux is directly installed (native) on the hardware. **Note**: you cannot do this project in a virtualbox hosted Linux.
2. Please make sure balloon driver is enabled.
3. Boot Ubuntu from a USB flash drive. **Note**: it will be a bit slow.
4. Use the deepthought server, more information will be released soon.

Step by Step --- Where can I find the APIs I might need to use?

1. libvirt-domain provides APIs to monitor and manage the guest virtual machines.
2. libvirt-host provides APIs to query the information regarding host machine.
3. Development Guide libvirt's official guide, you might read to get a general idea of libvirt, but it does not really help a lot. Too many parts are TBD.

Step by Step --- VCPU Scheduler

1. The first thing you need to do is to connect to the Hypervisor, virConnect* functions in libvirt-host are what you need to check. In our project, please connect to the local one which is "qemu:///system".
2. Next, you need to get all active running virtual machines within "qemu:///system", virConnectList* functions will help you.
3. You are ready to collect VCPU statistics, to do this you need virDomainGet* functions in libvirt-domain. If you also need host pcpu information, there are also APIs in libvirt-host.
4. You are likely to get VCPU time in nanoseconds instead of VCPU usage in % form. Think how to transform or use them.
5. You can also determine the current map (affinity) between VCPU to PCPU through virDomainGet* functions.
6. Write your algorithm, and according to the statistics, find "the best" PCPU to pin each VCPU.
7. Use virDomainPinVcpu to dynamically change the PCPU assigned to each VCPU.
8. Now you have a "one time scheduler", revise it to run periodically.
9. Launch several virtual machines and launch test workloads in every virtual machine to consume CPU resources, then test your VCPU scheduler.

Step by Step --- Memory Coordinator

1. The first thing you need to do is to connect to the Hypervisor, virConnect* functions in libvirt-host are what you need to check. In our project, please connect to the local one which is "qemu:///system".
2. Next, you need to get all active running virtual machines within "qemu:///system", virConnectList* functions will help you.
3. Tricky: to get as much memory statistics as possible, you need virDomainSetMemoryStatsPeriod fucntion.
4. Now you are ready to get memory statistics, first think what kind of memory statistics you are expecting to get. Then viewvirDomainGet* and virDomainMemory* functions to find those you need.
5. You can also get the host memory information through virNodeGet* in libvirt-host.
6. Write your algorithm, choose your policy, decide how much extra free memory you should give to each virtual machine according to the numbers you get.
7. Use virDomainSetMemory to dynamically change the memory of each virtual machine, which will indirectly trigger balloon driver.
8. Now you have a "one time scheduler", revise it to run periodically.
9. Launch several virtual machines and launch test workloads in every virtual machine to consume memory resources gradually, then test your memory coordinator.