Virtualization - A highlight on performance

V0 – Guillaume Urvoy-Keller

December 4, 2015

During this lab, you are going to:

- Get familiar with Vagrant and Docker and the notion of images and hub/repository
- Benchmark CPU and network capabilities of virtual machines and containers.

You must have done section 1 before the actual lab session

1 Vagrant VMs

1. Create a Vagrant folder and move to this folder
2. Browse the Atlas repository looking for a recent ubuntu box at `https://atlas.hashicorp.com/boxes/search`. Note that it must be an official box, i.e. a box pushed by the ubuntu community. Boxes are indexed with a repository name. The boxes pushed by the Ubuntu community are in the Ubuntu repository.
3. Download the box by typing on the command line (adjust with the new release if necessary):
   vagrant box add ubuntu/trusty64
4. Check that the box is there by typing:
   vagrant box list
5. Create your first VM:
   (a) Create a Ubuntu1 folder under the Vagrant folder and move to this folder
   (b) Type: **vagrant init ubuntu/trusty64**. The command creates a file called Vagrantfile that describes the VM parameters.
   (c) Start the machine by typing: **vagrant up**
   (d) If everything is ok, you should be able to ssh to the machine via a simple **vagrant ssh**
   (e) Which type of interface does the machine have?
6. We are now going to modify the VM by
   - Installing a set packages with benchmarks: sysbench (for CPU) and iperf (for network).
   - Adding a second interface in bridge mode
   - Forwarding 5001 port (the iperf server port) on the initial interface.
   - Installing docker and openvswitch

   All these actions can be done via the Vagrantfile.
   (a) Stop the VM by first exiting if you are connected to it and then from the command line:
   vagrant halt
   (b) Edit the Vagrantfile and do the following:
   - Uncomment the line with

     ```
     config.vm.network "public_network"
     ```

so as to create a bridge interface

- Uncomment and modify the forwarded port line:

```
config.vm.network "forwarded_port", guest: 5001, host: 5001 , auto_correct:  true
```

- Uncomment and modify the following lines that concerns the provisioning of the machine:

```
config.vm.provision "shell", inline: <<-SHELL
    sudo apt-get update
    sudo apt-get install -y sysbench iperf openvswitch-switch
    wget -qO- https://get.docker.com/ | sh
  SHELL
```

(c) Restart the machine with **vagrant up**

(d) Provision the machine by typing: **vagrant provision**. Check the interfaces and the presence of the new packages with **dpkg -s package_name**. Check that Docker is available by running the hello world container: **sudo docker run hello-world**. You should see first the download of the many layers of the container from the docker hub (equivalent to Atlas), then the output of this simple container:

```
$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b901d36b6f2f: Pull complete
0a6ba66e537a: Pull complete
Digest: sha256:8be990ef2aeb16dbcb9271ddfe2610fa6658d13f6dfb8bc72074cc1ca36966a7
Status: Downloaded newer image for hello-world:latest

Hello from Docker.
This message shows that your installation appears to be working correctly.
[...]
```

(e) Containers are sets of processes packed together with their own interfaces, file system, etc. Just like Vagrant, a repository exists where one can push/pull container images.

i. Browse the docker hub `https://hub.docker.com/`

ii. Find the official ubuntu repository and download the latest version that you define with a tag: **docker pull ubuntu**.

iii. Check the image is correctly downloaded with **docker images**

7. We will need a second VM with the same characteristics. An easy way of doing this in Vagrant is to create an image out of the current VM.

(a) Stop the VM **vagrant halt**

(b) Dump the current VM: **vagrant package --output Myubuntu**

(c) Add the box: **vagrant box add ubuntu_docker --name Myubuntu**

(d) Check that the box is now available with **vagrant box list**

2 CPU benchmarking

The benchmark consists in an algorithm to compute prime number. You run it with:

```
sysbench --test=cpu  --cpu-max-prime=10000 --num-threads=1 run | grep "total time:"
```

It performs 1000 independent computations. If N CPUs are available, they should last for $\frac{1}{N}$ of the duration over a single CPU if you adjust the number of threads (via the parameter –num-threads=) to N.

1. Modify the configuration of your VM by editing the Vagrantfile, find the part on Virtualbox provisioning that start with

```
config.vm.provider "virtualbox" do |vb|
```

Vary the number of vCPUs with the parameter

```
vb.cpus = 2
```

Consider values 1 and 4 (to adjust depending on your hardware). After each change in the Vagrantfile, do a **vagrant reload**.Check that the configuration was effective by checking the GUI of Virtualbox or, from inside the VM, check the number of cpu with **more /proc/cpuinfo**

2. For each configuration, perform 10 tests and report the average in a table.
3. If your host is a linux box, compare the performance of sysbench on your virtual machine and your physical host.

3 Docker

1. Provision the VM with 2 vCPU.
2. Run our first container: **docker run -ti --name=ubuntu ubuntu /bin/bash**, install the sysbench package and benchmark the CPU with sysbench.
3. How many cpus sees the container?
4. Compare the performance with the one of the (host) VM.
5. Stop the container (type **exit**). Check the active containers with **docker ps** and also the list of stopped containers with **docker ps -a** (you should see your recently killed container) and delete it: **docker rm ubuntu**
6. Restart a similar container with the command:

```
docker run -ti --cpuset-cpus="1" --name=ubuntu ubuntu /bin/bash
```

and redo the test. What do you obtain?

4 Network benchmarking

4.1 Vagrant / Virtualbox

1. Go to a new directory, e.g. Ubuntu2. Start a second VM using the image you created at the end of Section 1.
2. Check how iperf works at `http://openmaniak.com/iperf.php`. Run the server on one node **iperf -s** and the client at the other node **iperf -c IP_address** where IP_address will be either the one of the bridge interface or the NATed one (the address of the NATed interface is in the network 10.0.2.0/24 normally. It should be eth0 as it is your first created interface in the Vagrantfile).
   (a) Do the test 10 times for each case to check if the results are stable.
   (b) Redo the tests with a small mss of 64 bytes (-M option)... if the machine accepts it. Are the results qualitatively similar?
   (c) Redo the test with 10 flows in parallel (-p option) to assess the fairness of the solution. Comment.

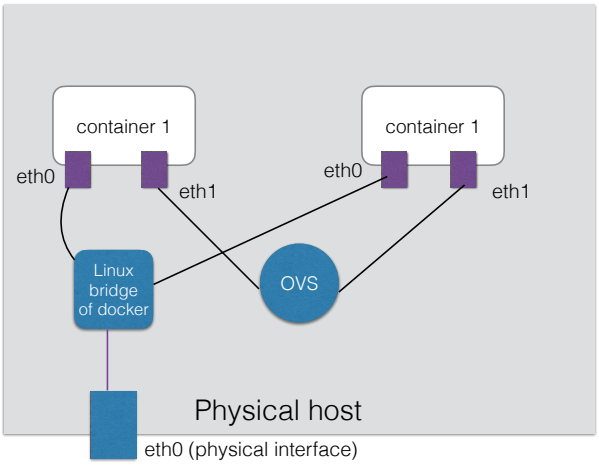

Figure 1: 2 containers inside the Virtualbox host

4.2 Docker

We are going to connect two containers and interconnect them using either the legacy method provided by Docker that is based on linux bridges or using an openvswitch – see Figure **??**

1. We next follow the instructions from
   `http://containertutorials.com/network/ovs_docker.html`
   You have to be root.

   (a) Install ovs-docker utility:

   ```
   $ sudo su
   $ cd /usr/bin
   $ wget https://raw.githubusercontent.com/openvswitch/ovs/master/utilities/ovs-docker
   $ chmod a+rwx ovs-docker
   ```

   (b) Create an OVS bridge:

   ```
   $ ovs-vsctl add-br ovs-br1
   $ ifconfig ovs-br1 173.16.1.1 netmask 255.255.255.0 up
   ```

   (c) Create two containers **in two different terminal windows**

   ```
   $ docker run -t -i --name container1 -p 5001:5001 ubuntu /bin/bash
   $ docker run -t -i --name container2 ubuntu /bin/bash
   ```

   (d) Connect the container to OVS bridge

   ```
   $ ovs-docker add-port ovs-br1 eth1 container1 --ipaddress=173.16.1.2/24
   $ ovs-docker add-port ovs-br1 eth1 container2 --ipaddress=173.16.1.3/24
   ```

   (e) Install the iperf package in each container

   (f) Start the iperf server in container1

   (g) Run the iperf client and binds to (i) the IP of eth0 (internal docker network), (ii) the forwarded port (using the IP address of the host VM) and (iii) the IP address assigned for the OVS port.

   (h) Perform 10 tests and compare them qualitatively.