README

In this README, we discuss how to deploy Linux-based (Ubuntu) and Kite network and storage domains on a physical machine. We also discuss how to reproduce our experimental results presented in Section 5 of the Eurosys'22 paper titled **Kite: Lightweight Critical Service Domains**.

Description & Requirements

The paper describes the experimental setup, including the hardware, Xen hypervisor version, and operating systems used for the different domains in Section 5. Here we describe other hardware and software dependencies, used benchmarks, and how to set up the artifact evaluation environment.

How to access. Kite's source code is available at `https://github.com/ssrg-vt/kite`. Artifact evaluation files are located at *kite/Artifact* directory, which contains this README, another README that explains how to set up the benchmarking tools and run benchmarks, benchmarking scripts at *kite/Artifact/benchmarking_scripts* directory, and configuration files at *kite/Artifact/config* directory. The *config* directory contains configuration scripts for building the Ubuntu driver domain and guest domain. It also contains configuration scripts for booting up Ubuntu and Kite domains for network and storage domain evaluation in *network* and *storage* subdirectories, respectively.

Hardware dependencies. Driver domains require physical 10Gbps NIC and NVMe devices via PCI passthrough (similar to Section 5). Moreover, virtualization support to run Xen is required. Consequently, VM images can be problematic since they involve nested virtualization. Thus, **physical machine deployment** is required.

Software dependencies. Kite should work with any Linux-based OS. However, we recommend Ubuntu 18.04 LTS for the Dom0 and DomU OSs.

Benchmarks. Our evaluation requires installation of Nuttcp, Netperf, Redis, Apache, Memcached, MySQL on the server machine inside DomU. The client (load generator) machine should have corresponding client benchmark applications for Nuttcp (v8.2.2), Netperf (v2.6.0-2.1), Redis (v4.0.9), Apache (v2.4.29), sysbench (v1.1.0) (for MySQL (v5.7.29)) for network domain evaluation. For storage domain evaluation, the client machine needs MySQL server (v5.7.29), sysbench benchmark (v1.1.0), and Filebench (1.5-alpha3) benchmarks. The benchmark scripts and instructions can be found in the artifact package.

Setup (physical machine)

Xen: First, install Ubuntu 18.04.1 LTS on a 64bit x86 machine. Please select "Use LVM with the new Ubuntu installation".

Then, install the Xen 4.9.2 hypervisor and reboot the machine; GRUB should automatically boot Xen and launch Dom0:

```
# apt install xen-hypervisor-amd64
```

PCI passthrough: Find BDF numbers of the available PCI devices (NIC, NVMe) using the `lspci` command. Then, add the corresponding device to the PCI assignable list, where `xx:xx.x` represents the BDF number:

```
# modprobe xen-pciback
# xl pci-assignable-add xx:xx.x
```

Kite: Please set up Kite's build environment:

```
# apt install build-essential git
# apt install libz-dev libxen-dev
```

Next, get Kite's source and build it:

```
$ git clone https://github.com/ssrg-vt/kite
$ cd kite
$ git submodule update --init --recursive --remote
$ CC=‘echo $PWD‘/gcc8fix.sh ./build-rr.sh -j16 hw
$ cd bridge
$ ./ifconf.sh && ./run.sh
$ cd ../vbdconf
$ ./run.sh
```

Guest Domain (DomU) for server applications. First, create a logical disk drive to install a guest OS:

```
# lvcreate -L 40G -n ubuntu_guest /dev/<VG>
```

Please download Ubuntu 18.04 LTS from `https://releases.ubuntu.com/18.04/ubuntu-18.04.6-desktop-amd64.iso`. Then launch a guest VM using the provided configuration file from the artifact package:

```
# xl create -c config/ubuntu_guest_setup.cfg
```

Install a VNC client (such as vncviewer) for GUI access (using `localhost`) and finish Ubuntu guest installation.

Linux Driver Domain. : First, create a logical disk drive to install the Ubuntu driver domain:

```
# lvcreate -L 40G -n ubuntu_dd /dev/<VG>
```

Then copy the contents from `/dev/<VG>/ubuntu_guest` to save the OS installation time for the Ubuntu driver domain:

```
# dd if=/dev/<VG>/ubuntu_guest
 of=/dev/<VG>/ubuntu_dd bs=1G count=40
```

Launch the Ubuntu driver domain using the provided configuration file `ubuntu_dd_setup.cfg`:

```
# xl create -c config/ubuntu_dd_setup.cfg>
```

Next, install Xen tools. (It can be easier to simply install the Xen hypervisor again.) Then, replace `/etc/default/grub.d/xen.cfg` with the provided `config/xen.cfg`; it will prevent the driver domain itself from booting the Xen hypervisor. Finally, update GRUB by running 'update-grub'.

Evaluation workflow

For convenience of the reviewers, by default, we reduced the number of iterations in the benchmark scripts.

Major Claims.

- (C1): Kite achieves 10x faster boot time than an Ubuntu-based driver domain. See experiment E1 in Section 5.2, for which results are reported in Figure 4c.
- (C2): Kite's network domain performs similarly to an Ubuntu-based network domain. See experiment E2.
- (C3): Kite's storage domain performs similarly to an Ubuntu-based storage domain. See experiment E3.
- (C4): We skip Figure 5 (ROP gadgets) due to the need of extra tools; this is not a fundamental paper result, it is just given for information purposes only. The reduced attack surface also follows from reduced image sizes.

Experiment E1 [Boot time]: We measure the boot time for both Ubuntu and Kite driver domains. We use network domains but results are similar for storage domains.

First, update `config/network/ubunt_dd.cfg` with the BDF number of the network device:

```
pci=['xx:xx.x,permissive=1']
```

Then, launch an Ubuntu-based network domain and measure the boot time manually until you see the login screen:

```
# xl create -c config/network/ubuntu_dd.cfg
```

Next, terminate the Ubuntu domain. Then, to run Kite's network domain, first add the network device's BDF number into the `config/network/kite_dd.cfg` file. Next, launch the Kite network domain using the following command.

```
# xl create -c config/network/kite_dd.cfg
```

Measure the boot time manually until you see a notification that says `'Network domain is ready.'` To destroy Kite's domains, run the following:

```
# xl destroy <Kite domain id>
```

To locate domain IDs, run the following command in Dom0, where Kite's network domain is named 'netbackend':

```
# xl list
```

Kite should exhibit at least 10x faster boot time.

Experiment E2 [Network performance]: To eva-luate an Ubuntu-based network domain, first launch it:

```
# xl create -c config/network/ubuntu_dd.cfg
```

In the driver domain, create a network bridge, named `xenbr0`, with the network interface corresponding to the network device (assigned via PCI passthrough). Then, launch the Xen driver domain daemon:

```
# xl devd
```

Next, launch the Ubuntu DomU guest:

```
# xl create -c config/network/guest_on_ubuntu.cfg
```

We run server applications such as Apache, Redis, Memcached, and MySQL in this guest machine. The client machine should be connected to the same network. We use the benchmark scripts from the artifact package to measure network throughput, CPU utilization, and latency. (See the details in README_benchmark.pdf to learn about the client machine setup.) You can run the `network_benchmarks.sh` script to run multiple benchmarks using the following command.

```
# bash network_benchmarks.sh Linux
```

The benchmark results will be stored in a file named `results_Linux`. To evaluate Kite's network domain, launch Kite as explained in E1. Next, launch the Ubuntu DomU:

```
# xl create -c config/network/guest_on_kite.cfg
```

You can run the same network benchmark experiments from the client machine to evaluate Kite's network domain. You can also issue the following command to run multiple benchmarks.

```
# bash network_benchmarks.sh Kite
```

The benchmark results will be stored in a file named `results_Kite`. We expect Kite to yield similar performance to that of Ubuntu.

Experiment E3 [Storage performance]: To evaluate the Ubuntu storage domain, first launch it:

```
# xl create -c config/storage/ubuntu_dd.cfg
```

Attach the storage device to the Ubuntu storage domain:

```
# xl pci-attach <Driver Domain ID> 'xx:x.x'
```

Launch the Xen driver domain daemon:

```
# xl devd
```

Next, launch the Ubuntu DomU guest:

```
# xl create -c config/storage/guest_on_ubuntu.cfg
```

Mount the PV storage device to an empty directory:

```
# mkdir disk
# mount /dev/xvdb disk
```

We run the MySQL server with sysbench and Filebench benchmark for file server, web server, and the MongoDB server to evaluate the storage domains. You can use the benchmark scripts from the artifact package (instructions are in README_benchmark.pdf) to measure storage throughput, CPU utilization, and latency. You can issue the following command to run multiple benchmarks.

```
# bash storage_benchmarks.sh Linux
```

The benchmark results will be stored in a file named `results_Linux`. To evaluate Kite's storage domain, first change the `config/storage/kite_dd.cfg` file with storage device's BDF number. Then, run the following commands to build the storage domain application and launch Kite's storage domain:

```
# xl create -c config/storage/kite_dd.cfg
```

Next, launch the Ubuntu DomU guest:

```
# xl create -c config/storage/guest_on_kite.cfg
```

You can run the same benchmark experiments in the Ubuntu guest VM to evaluate Kite's storage domain. You can also use the `storage_benchmarks.sh` script to run multiple benchmarks issuing the following command.

```
# bash storage_benchmarks.sh Kite
```

The benchmark results will be stored in a file named `results_Kite`. We expect Kite to yield performance similar to that of Ubuntu.