This document contains instructions to set up the required benchmark environment in the Ubuntu guest VM and client machine to evaluate Xen driver domains, such as Ubuntu and Kite. This is needed for the actual physical installation.

Networking benchmark:

The guest VM runs the network servers and a client machine runs the corresponding clients that can be used to benchmark the network server. The guest VM configuration detail is discussed in the Evaluation section and Appendix of the paper.

Any physical machine with an internet access and a 10Gbps NIC connected to the machine that runs Xen driver domain and guest domains, should be sufficient to run the network benchmark scripts we used for this paper.

The client machine we used has the following configuration.

CPU	Core i5-6600K 3.50GHz
Cores	4
L1/L2	32/256 KB per core
L3	6114 KB
Memory	16 GB
Network	Intel 82599ES 10-Gigabit
Operating System	Ubuntu 18.04.3 LTS

Next, we explain how to set up different servers and their corresponding clients.

MySQL with Sysbench

On the server side:

- Install MySQL (on Ubuntu)

```
sudo apt-get install mysql-server
```

- Set the security settings including the passwords

```
sudo mysql_secure_installation utility
```

- Allow remote access

```
sudo ufw enable
```

```
sudo ufw allow mysql
```

- Specify which interface the server will use to listen on by modifying the file `/etc/mysql/mysql.conf.d/mysqld.cnf`

```
bind-address    = 127.0.0.1 ( The default. )
bind-address    = XXX.XXX.XXX.XXX
bind-address    = 0.0.0.0 ( All ip addresses. )
```

- Start the MySQL server

```
sudo systemctl start mysql
```

- Start the MySQL shell

```
sudo /usr/bin/mysql -u root -p
```

- Create an user as the following example

```
mysql>    CREATE    USER    'username'@'%'    IDENTIFIED    BY
'user_password';
```

If you want to restrict this user only from a specific IP, replace % with the IP.

- Create a database called sbtest

```
mysql> CREATE DATABASE sbtest;
```

- Grand all (for simplicity) privileges to a user account on all databases

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'username'@'%';
```

Or, to the only sbtest database

```
mysql> GRANT ALL PRIVILEGES ON sbtest.* TO 'username'@'%';
```

On the client side:

- Install Sysbench (on Ubuntu)

```
sudo apt-get install sysbench
```

- Prepare the table first (network benchmarking)

```
sysbench   oltp_read_only   --threads=4   --mysql-host=server-IP
--mysql-user=username            --mysql-password=user_password
--mysql-port=3306        --tables=10        --table-size=1000000
--db-driver=mysql prepare
```

- Then, run the benchmark (network benchmarking)

```
sysbench          oltp_read_only          --threads=thread_count
--time=time_in_sec  --mysql-host=server-IP  --mysql-user=username
--mysql-password=user_password   --mysql-port=3306   --tables=10
--table-size=1000000 --report-interval=1 --db-driver=mysql  run
```

- Finally, cleanup the database

```
sysbench          oltp_read_only          --mysql-host=server-IP
--db-driver=mysql     --mysql-db=sbtest     --mysql-user=username
--mysql-password=user_password --mysql-port=3306 cleanup
```

One can run the `sys.script.sh` from the artifact package to run the automated script we used for the presented numbers.

Memcached with Memtier

On the server side:

- Install memcached (in Ubuntu)

```
apt-get install memcached libmemcached-tools
```

- Specify which interface to listen on modifying `/etc/mamcached.conf`.

```
-l 192.168.0.101 (for specific interface)
-l 0.0.0.0 (for all interface)
```

- Start the memcached server

```
systemctl start memcached
```

- Allow the memcached port number to be accessed by the client

```
sudo ufw allow 11211
```

On the client side (on Ubuntu):

- Install the prerequisites first.

```
apt-get install build-essential autoconf automake libpcre3-dev
libevent-dev pkg-config zlib1g-dev libssl-dev
```

- Clone the memtier source repository

```
git clone https://github.com/RedisLabs/memtier_benchmark
```

- Build the memtier bench

```
cd memtier_benchmark
autoreconf -ivf
./configure
make
make install
```

- Run the benchmark

```
memtier_benchmark -s server_IP -p 11211 -P memcache_binary -c
10 -t thread_count -d data_size_in_byte
--test-time=time_in_secs --out-file=filename
```

One can run the `memcached-benchmark.sh` from the artifact package to run the automated script we used for the presented numbers.

```
./memcached-benchmark.sh Linux        #for Ubuntu
./memcached-benchmark.sh Kite         #for Kite
```

The benchmark results can be parsed using the following commands.

```
./memcached-parse.sh Linux            #for Ubuntu
./memcached-parse.sh Kite             #for Kite
```

CPU Utilization

- Setup the CPU utilization measurement tool on the server.

```
wget
http://pagesperso-orange.fr/sebastien.godard/sysstat-12.3.3.tar.xz
tar -xJf sysstat-12.3.3.tar.xz
cd sysstat-12.3.3
./configure
make
export PATH=$PATH:path_to_sysstat-12.3.3
```

- On the client machine, run the following benchmark.

```
./sys.cpu.sh
```

Redis

- Install redis on the server side:

```
sudo apt-get install redis-server
```

- Configure redis editing the file `/etc/redis/redis.conf`. Comment out the following line using # in front of the line.

```
bind 127.0.0.1 ::1
```

- Then start the redis server.

```
sudo redis-server --port 1000 --protected-mode no
```

- On the client side, run the following benchmarks.

```
./redis-benchmark-set Linux                # for Ubuntu
./redis-benchmark-get Kite                 # for Kite
```

```
./redis-benchmark-set Ubuntu                # for Ubuntu

./redis-benchmark-get Kite                  # for Kite
```

- Parse the benchmark results.

```
./redis-parse Linux             # for Ubuntu

./redis-parse Kite              # for Kite
```

Nuttcp UDP for 10Gbps NIC

On the server side, run

```
nuttcp -S
```

On the client side, run

```
nuttcp -l8972 -T30 -u -w4m -Ru -i1 server_ip
```

-l represents 9K jumbo frame size

Netperf for latency

On the server side, run

```
sudo netserver
```

On the client side run the following command for measuring latency:

```
netperf -H server_ip -l 100 -t TCP_RR -v 2 -- -o
min_latency,mean_latency,max_latency,stddev_latency,transaction
_rate
```

Storage benchmark:

The guest VM runs the servers as well as the client applications to benchmark the storage server. The guest VM configuration detail is discussed in the Evaluation section and Appendix of the paper.

First create an empty directory and mount the PV storage device there.

```
mkdir disk
sudo mount /dev/xvdb disk
```

DD:

- For writing, run the following command.
  `sudo dd if=/dev/zero of=disk/10g bs=1G count=10`

- For reading, run the following command.
  `sudo dd if=disk/10g of=/dev/null  bs=1G count=10`

We measure the time for each operation.

Filebench benchmark:

- First install the filebench benchmark running the following commands.

```
git clone https://github.com/filebench/filebench
libtoolize
aclocal
autoheader
automake --add-missing
autoconf
./configure
make
sudo make install
```

- Run the filebench benchmarks using the following commands.

```
bash -c "echo 0 > /proc/sys/kernel/randomize_va_space"

filebench -f fileserver.f
filebench -f webserver.f
filebench -f mongo.f
```

Sysbench FileIO

- To run the sysbench fileIO operations, run the following script in the guest domain.

```
./sys.disk.sh
```

- It runs the file IO operation for different file sizes. Run the following script to run sysbench varying the thread count.

```
./sys.disk.thread.sh
```