HW1 Report (Live Migration & Virtual Network Adapter)

A. Please simply explain how do you setup your VM

在我的 case 中，直接按造助教的講義做會導致對外不通。

因此，我們必須要將 br0 作為向外界溝通的 interface，因此在將原始網路介面(eth0) 設定為 br0 的 interface 之後， /etc/network/interfaces 檔案中必須要修改為以下。

使得 eth0 -> br0 -> internet 的狀況下仍然可以連上網路

```
# experiment
auto eth0
iface eth0 inet manual
dns-nameservers 8.8.8.8 8.8.4.4

auto br0
iface br0 inet static
address 140.114.28.143
netmask 255.255.255.0
gateway 140.114.28.254
#dns-nameservers 8.8.8.8 8.8.4.4
bridge_ports eth0
bridge_fd 9
bridge_hello 2
bridge_maxage 12
bridge_stp off

auto br0:0
iface br0:0 inet static
address 10.1.1.254
netmask 255.255.255.0
```

我們可以藉由 route 這個指令來確認我們目前對外的介面被更改成了 br0

```
yunchen@yunchen-ubuntu:~/Courses/Cloud_Computing$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Ifac
e
default         router-28.ee.nt 0.0.0.0         UG    0      0        0 br0
10.1.1.0        *               255.255.255.0   U     0      0        0 br0
140.114.28.0    *               255.255.255.0   U     0      0        0 br0
link-local      *               255.255.0.0     U     1000   0        0 br0
```

B. Show the performance testing results by *sysbench* with and without "--enable-kvm" on VM, and Host; furthermore compare among them and explain the results

以下分別是有開 kvm 和沒有開 kvm 用sysbench 測試 cpu 的實驗截圖，在跑 20000 個 prime number 的 test case下，有 kvm 的執行時間為 20.2661 s，而沒有 kvm 的則是 47.3306 s，差了兩倍以上，這是因為 kvm 就是 linux 環境下的 hypervisor，是一種 type 1 virtualization，並且和 QEMU 搭配下負責存取 CPU 和 Memory ，因此若有 kvm 來負責管理的話執行效率會更高。

```
yclo@ubuntu-vm:~$  sysbench --test=cpu --cpu-max-prime=20000 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 20000


Test execution summary:
    total time:                          20.2671s
    total number of events:              10000
    total time taken by event execution: 20.2661
    per-request statistics:
         min:                                  2.00ms
         avg:                                  2.03ms
         max:                                  5.85ms
         approx.  95 percentile:               2.10ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   20.2661/0.00
```

```
yclo@ubuntu-vm:~$  sysbench --test=cpu --cpu-max-prime=20000
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 20000


Test execution summary:
    total time:                          47.3442s
    total number of events:              10000
    total time taken by event execution: 47.3306
    per-request statistics:
         min:                                  4.68ms
         avg:                                  4.73ms
         max:                                 23.53ms
         approx.  95 percentile:               4.82ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   47.3306/0.00
```

C. Show the performance testing results by *iperf* with and without "virtio" on VM, and Host; furthermore compare among them and explain the results

以下分別是有開 virtio 和沒有，用 iperf 來測試的實驗結果，我們可以觀察到有 virtio 的 bandwidth 是 3.79 Gbits/sec，而沒有 virtio 的則是 3.59 Gbits/sec，差了 0.2 Gbits/sec 左右，這是因為 Virtio 是透過 KVM 來運行虛擬機，故可以比其它的模擬硬碟介面 (ex.IDE, SATA)得到更高的硬碟存取效率。

```
yunchen@yunchen-ubuntu:~$ iperf -c 10.1.1.1 -w 100M -t 100 -i 2
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size:  416 KByte (WARNING: requested  100 MByte)
------------------------------------------------------------
[  3] local 10.1.1.254 port 44174 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   904 MBytes  3.79 Gbits/sec
[  3]  2.0- 4.0 sec   920 MBytes  3.86 Gbits/sec
[  3]  4.0- 6.0 sec   896 MBytes  3.76 Gbits/sec
[  3]  6.0- 8.0 sec   927 MBytes  3.89 Gbits/sec
[  3]  8.0-10.0 sec   921 MBytes  3.86 Gbits/sec
[  3] 10.0-12.0 sec   904 MBytes  3.79 Gbits/sec
^C[  3]  0.0-12.8 sec  5.70 GBytes  3.82 Gbits/sec
yunchen@yunchen-ubuntu:~$ iperf -c 10.1.1.1 -w 100M -t 100 -i 2
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size:  416 KByte (WARNING: requested  100 MByte)
------------------------------------------------------------
[  3] local 10.1.1.254 port 44178 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   844 MBytes  3.54 Gbits/sec
[  3]  2.0- 4.0 sec   861 MBytes  3.61 Gbits/sec
[  3]  4.0- 6.0 sec   860 MBytes  3.61 Gbits/sec
[  3]  6.0- 8.0 sec   859 MBytes  3.60 Gbits/sec
[  3]  8.0-10.0 sec   856 MBytes  3.59 Gbits/sec
[  3] 10.0-12.0 sec   844 MBytes  3.54 Gbits/sec
^C[  3]  0.0-12.3 sec  5.13 GBytes  3.58 Gbits/sec
```

virtio

no virtio

D. Show the performance measurements by *iperf* and *sysbench* during the live migration is progressing; furthermore, describe your observations

首先是 iperf 的結果，我們可以在執行 live migration 的時候觀察到 bandwidth 有顯著的降低，從本來的3.75 Gbits/sec 降到 3.63 Gbits/sec 左右。

```
yunchen@yunchen-ubuntu:~$ iperf -c 10.1.1.1 -w 100M -t 100 -i 2
------------------------------------------------------------
Client connecting to 10.1.1.1, TCP port 5001
TCP window size:  416 KByte (WARNING: requested  100 MByte)
------------------------------------------------------------
[  3] local 10.1.1.254 port 44128 connected with 10.1.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 2.0 sec   874 MBytes  3.67 Gbits/sec
[  3]  2.0- 4.0 sec   886 MBytes  3.72 Gbits/sec
[  3]  4.0- 6.0 sec   851 MBytes  3.57 Gbits/sec
[  3]  6.0- 8.0 sec   895 MBytes  3.75 Gbits/sec
[  3]  8.0-10.0 sec   888 MBytes  3.72 Gbits/sec
[  3] 10.0-12.0 sec   876 MBytes  3.68 Gbits/sec
[  3] 12.0-14.0 sec   879 MBytes  3.69 Gbits/sec
[  3] 14.0-16.0 sec   864 MBytes  3.62 Gbits/sec
[  3] 16.0-18.0 sec   900 MBytes  3.78 Gbits/sec
[  3] 18.0-20.0 sec   886 MBytes  3.72 Gbits/sec
[  3] 20.0-22.0 sec   904 MBytes  3.79 Gbits/sec
[  3] 22.0-24.0 sec   908 MBytes  3.81 Gbits/sec
[  3] 24.0-26.0 sec   919 MBytes  3.85 Gbits/sec
[  3] 26.0-28.0 sec   918 MBytes  3.85 Gbits/sec
[  3] 28.0-30.0 sec   890 MBytes  3.73 Gbits/sec
[  3] 30.0-32.0 sec   875 MBytes  3.67 Gbits/sec
[  3] 32.0-34.0 sec   877 MBytes  3.68 Gbits/sec
[  3] 34.0-36.0 sec   891 MBytes  3.74 Gbits/sec
[  3] 36.0-38.0 sec   890 MBytes  3.73 Gbits/sec
[  3] 38.0-40.0 sec   871 MBytes  3.65 Gbits/sec
[  3] 40.0-42.0 sec   860 MBytes  3.61 Gbits/sec
[  3] 42.0-44.0 sec   868 MBytes  3.64 Gbits/sec
[  3] 44.0-46.0 sec   820 MBytes  3.44 Gbits/sec
[  3] 46.0-48.0 sec   883 MBytes  3.70 Gbits/sec
[  3] 48.0-50.0 sec   902 MBytes  3.79 Gbits/sec
[  3] 50.0-52.0 sec   885 MBytes  3.71 Gbits/sec
[  3] 52.0-54.0 sec   867 MBytes  3.64 Gbits/sec
[  3] 54.0-56.0 sec   873 MBytes  3.66 Gbits/sec
[  3] 56.0-58.0 sec   881 MBytes  3.70 Gbits/sec
[  3] 58.0-60.0 sec   888 MBytes  3.73 Gbits/sec
[  3] 60.0-62.0 sec   888 MBytes  3.73 Gbits/sec
[  3] 62.0-64.0 sec   892 MBytes  3.74 Gbits/sec
^C[  3]  0.0-64.6 sec  27.8 GBytes  3.70 Gbits/sec
```

接著是 sysbench 測試 CPU、Memory 的結果，我們可以發現 CPU 在執行同樣的 workload 時，若中間有發生 live migration 則執行時間會增加(20.0835->20.1734s)，Memory 在發生 Live Migration 時，Throughput 會減少(47904 MB/sec -> 11338 MB/sec)

```
         avg:                                  2.01ms
         max:                                  3.57ms
         approx.  95 percentile:               2.02ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   20.0835/0.00

yclo@ubuntu-vm:~$ sysbench --test=cpu --cpu-max-prime=20000 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 20000


Test execution summary:
    total time:                          20.1745s
    total number of events:              10000
    total time taken by event execution: 20.1734
    per-request statistics:
         min:                                  2.00ms
         avg:                                  2.02ms
         max:                                  9.34ms
         approx.  95 percentile:               2.07ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   20.1734/0.00
```

Baseline

Live Migration

```
10240.00 MB transferred (47903.91 MB/sec)

Test execution summary:
    total time:                          0.2138s
    total number of events:              1310720
    total time taken by event execution: 0.1512
    per-request statistics:
         min:                                  0.00ms
         avg:                                  0.00ms
         max:                                  0.06ms
         approx.  95 percentile:               0.00ms

Threads fairness:
    events (avg/stddev):           1310720.0000/0.00
    execution time (avg/stddev):   0.1512/0.00

yclo@ubuntu-vm:~$ sysbench --test=memory --memory-block-size=8K --memory-total-size=1G run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1

Doing memory operations speed test
Memory block size: 8K

Memory transfer size: 1024M

Memory operations type: write
Memory scope type: global
Threads started!
Done.

Operations performed: 131072 (1451301.66 ops/sec)

1024.00 MB transferred (11338.29 MB/sec)
```

Baseline

Live Migration

E. What is “Live Migration” and Why we need it.

Live Migration 是不停止虛擬機器或其所執行服務的情況下，將正在執行的虛擬機器移動至另一台實體主機的技術，其主要發生在想要將硬件升級或是灌其他軟體，但卻不能暫停現有服務時，和server 突然當機，需要修復沒有直接相關。

(參考：http://blog.51cto.com/findman/260748)

F. Please describe how to maintain the network connection when the VM is being migrated

Live Migration 主要可以分為五個步驟：(1) 建立目標機器和來源機器的連線 (2) 傳送來源機器的虛擬機配置和機器訊息 (3) 傳送虛擬機的內存記憶體 (4) 暫停來源機的執行並且傳送執行狀態 (5) 在新機器上繼續執行

我們可以發現，透過(2)(3)(4) 的處理，基本上我們在新的機器上就有了網路的相關設定，接下來只要將 Program state 傳送過去就可以繼續執行，因此不會在一半網路就中斷。

G. What is fault-tolerance in cloud system and why we need it?

和一般定義的 fault-tolerance 類似，在 cloud system 中的 fault-tolerance 指的是當 layer 或是機器發生問題時，會自動修復錯誤的部分的功能，我們特別在雲端中需要他，是因為雲端的環境和一般的桌機電腦差異很大，由很多不同種類的機器組成，但每種機器發生問題的訊息都不一樣，因此我們需要好的寄至來解決這個問題。

H. What are the relationships between live migration and fault-tolerance?

live migration 是人有意識的去轉移服務給其他的機器，以便升級當下執行服務的機器，而fault-tolerance 則是避免當雲端中有機器故障，卻因為規模龐大，難以修復的問題，他們都是良好的管理一個雲端系統不可或缺的部分，當搭配使用，可以避免雲端故障，也能在想維修或升級時進行處理。