Virtual Machine HW4

Index

Benchmark Configuration

Hackbench Configuration

因為在 KVM Guest 中的 max user processes 數目限制為 1747，可以用下面指令知道：

```
ulimit -a
```

因此，在調整 hackbench 實驗的參數如下：

```
./hackbench 10 process 20
```

sender and receiver process group 數目為 10，每個 group 裡共有 20 個 sender processes 和 receiver processes，因此共有 10 * 40 = 400 個 tasks，

並每個 group 裡的 sender 會送 20 個 messages 給同個 group 裡的 receiver。

跑 hackbench 5次，取結果的中位數作為實驗結果。

Kernbench Configuration

以下面的參數設定 Kernbench 來跑 fast mode 以加速 Kernbench 的實驗時間：

```
../kernbench -M -H -f -n 1 | tee >(grep 'Elapsed' | awk '{print $3 }' >> kernbench.txt)
```

Netperf Configuration

在我的實驗中的 local send size 設定為 default local send socket buffer size，remote receive size 為 default remote receive socket buffer size

可以使用：

```
cat /proc/sys/net/ipv4/tcp_rmem
cat /proc/sys/net/ipv4/tcp_wmem
```

知道 default local send socket buffer size 和 default remote receive socket buffer size

而實驗環境的 socket buffer size 如下：

- Host default local send socket buffer size：16384 bytes
- Host default remote receive socket buffer size：131072 bytes
- Guest default local send socket buffer size：16384 bytes
- Guest default remote receive socket buffer size：131072 bytes

並做 TCP Stream 和 TCP RR 兩種 benchmark 實驗，各個實驗共做五次，並取五次結果中位數作為實驗結果：

```
bash ./kvmperf/cmdline_tests/netperf.sh [ip address] ALL 5
```

Apache Configuration

在我的實驗中，request 次數為：100000，並且同時有 100 個 request

並且每次共跑 3 個 runs

```
bash apache.sh [ip address] 3
```

取 3 個 runs 的中位數作為實驗結果

實驗前，要先使用下面指令開啟 apache2 server

```
sudo service apache2 start
```

Experiment Environment

CPU Information

實驗環境的 physical CPU 資訊為：

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          24
On-line CPU(s) list:             0-23
Thread(s) per core:              1
Core(s) per socket:              16
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           151
Model name:                      12th Gen Intel(R) Core(TM) i9-12900K
Stepping:                        2
CPU MHz:                         1219.462
CPU max MHz:                     6700.0000
CPU min MHz:                     800.0000
BogoMIPS:                        6374.40
Virtualization:                  VT-x
L1d cache:                       384 KiB
L1i cache:                       256 KiB
L2 cache:                        10 MiB
NUMA node0 CPU(s):               0-23
```

Benchmarking Explanation

Hackbench

Hackbench 用作測試 Linux scheduler 的效能，會先創建數個 groups，每個 group 中有 20 個 sender processes 和 receiver processes，

同個 group 中的 sender process 會透過 unix domain socket 方式傳封包給 group 裡的所有 receiver processes

最後得到的時間為所有的 sender processes 傳封包所花的時間總和，時間越短表示 linux scheduler 效能越好。

Kernbench

Kernbench 使用 time package 來測量編譯 linux kernel 時間以去衡量在使用不同的 -j flag 下的編譯時間：

```
$time -f "%e %U %S %P %c %w" -o timelog make -j $tempjobs > /dev/null 2>&1
```

可以由 time 的 man page 知道各個時間的意義

- %e：Elapsed real (wall clock) time used by the process, in seconds.
- %U：Total number of CPU-seconds that the process used directly (in user mode), in seconds.
- %S：Total number of CPU-seconds used by the system on behalf of the process (in kernel mode), in seconds.
- %P：Percentage of the CPU that this job got. This is just user + system times divided by the total running time. It also prints a percentage sign.
- %c：Number of times the process was context-switched involuntarily (because the time slice expired).
- %w：Number of times that the program was context-switched voluntarily, for instance while waiting for an I/O operation to complete.

Netperf

TCP Stream

TCP Stream test 在量測 client 和 server 間透過 TCP 連接的資料傳輸率，得到的結果為每秒成功傳輸的 Megabits 數，

當量測得到的數值越大，表示每秒可以傳輸的資料越多，表示網路效能越好。

TCP RR

TCP Request-Response test 和 TCP Stream test 的差別在於在 TCP Stream test 中 server 會直接丟棄 client 傳輸過來的資料，只量測單向的傳輸速率

但 TCP RR test 中 client 會送 request 給 server，server 會立即回應該 request，並量測每秒完成的 request-response transaction 個數，得到的數值越大表示網路效能越好。

Apache HTTP Server Benchmark

Apache2 Benchmarking 的 Output format 如下：

```
Server Software:        Apache/2.4.41
Server Hostname:        192.168.0.105
Server Port:            80

Document Path:          /gcc/index.html
Document Length:        275 bytes

Concurrency Level:      100
Time taken for tests:   1130.644 seconds
Complete requests:      100000
Failed requests:        0
Non-2xx responses:      100000
Total transferred:      45500000 bytes
HTML transferred:       27500000 bytes
Requests per second:    88.45 [#/sec] (mean)
Time per request:       1130.644 [ms] (mean)
Time per request:       11.306 [ms] (mean, across all concurrent requests)
Transfer rate:          39.30 [Kbytes/sec] received
```

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    1   2.7      1      96
Processing:   167 1129 896.9    694   18689
Waiting:       57  988 663.4    680   13977
Total:        251 1130 897.4    695   18699

Percentage of the requests served within a certain time (ms)
  50%    695
  66%   1047
  75%   1257
  80%   1506
  90%   2334
  95%   2896
  98%   3569
  99%   4133
 100%  18699 (longest request)
```

我這邊只取 time per request across all concurrent requests 和 transfer rate 作為實驗結果紀錄，

當 time per request across all concurrent requests 越大，表示處理 request 的時間越久，網路效能越差，

transfer rate 表示的是網路傳輸速度，當 transfer rate 越高，表示每秒可以傳輸資料量越大，網路效能越好。

1. VM performance comparison different configurations

1.1 Compare VM performance of SMP VM versus UP VM

1.1.1 Experimental Result

實驗設定如下：

- KVM Host RAM：2GB
- KVM Guest RAM：512MB
- KVM Host enable transparent huge page
- KVM Guest turn off vhost

實驗變數調整 KVM host 和 KVM guest 的 VCPU 為 (1, 1), (2, 1), (2, 2), (4, 2), (4, 4), (8, 4), (8, 8)：

其中，hackbench on host、kernbench on host、Netperf (host → host) 和 Apache2 (host → host) 都是在尚未跑 guest VM 時測量的，因此相同的 Host vCPU 下，不再重複測量。

KVM Host vCPUs	KVM Guest vCPUs	Hackbench (Host) (s)	Hackbench (Guest) (s)	Kernbench (Host) Elapsed Time (s)	Kernbench (Host) System Time (s)	Kernbench (Host) User Time (s)	Kernbench (Host) Percent CPU (%)	Kernbench (Host) Context Switches (times)	Kernbench (Host) Sleeps (times)
1	1	9.639	38.962	2841.36	2050.66	418.40	86	348177	32916
2	1	4.661	25.589	1730.25	2195.10	452.92	153	321368	32166
2	2	X	7.710	X	X	X	X	X	X
4	2	1.357	7.055	753.35	2252.39	466.08	360	153549	31371
4	4	X	3.778	X	X	X	X	X	X
8	4	0.922	4.838	459.51	2639.05	501.81	683	102908	33492
8	8	X	2.327	X	X	X	X	X	X

KVM Host vCPUs	KVM Guest vCPUs	Netperf TCP Stream (Host → Host) (Mbits/s)	Netperf TCP RR (Host → Host) (times/s)	Netperf TCP Stream (Host → Guest) (Mbits/s)	Netperf TCP RR (Host → Guest) (times/s)	Apache Transfer Rate (Host → Host) (Kbytes/sec)	Apache Time per request (Host → Host) (ms)	Apache Transfer Rate (Host → Guest) (Kbytes/sec)	Apache Time per request (Host → Guest) (ms)
1	1	843.63	2627.02	788.39	525.25	10234.40	3.966	30.94	14.361
2	1	2630.93	4490.93	1428.32	946.65	22892.54	1.773	59.69	7.444
2	2	X	X	1025.33	789.01	X	X	82.61	5.379
4	2	2689.79	4463.47	1953.32	864.37	34357.98	1.181	186.72	2.380
4	4	X	X	1975.04	824.95	X	X	265.37	1.674
8	4	2736.85	4130.32	1970.43	801.06	29197.34	1.390	422.07	1.053

8	8	X	X	1923.28	818.03	X	X	475.75	0.934

1.1.2 DiscussionA

從實驗中可以觀察到的是：

Hackbench & Kernbench：

- 不論是 KVM Host 或是 KVM Guest 隨著 vCPU 個數的增加，hackbench 的時間會減少，這是因為當 vCPU 個數增加，scheduler 就有越多 CPU 資源可以分配給 process 去使用，使得同時可以處理更多的 sender 或 receiver processes，使得 hackbench 時間縮短。
- 儘管 KVM Guest vCPU 數相同，但當 KVM Host vCPU 個數增加時，hackbench 的時間有時會減少，這是因為儘管 Guest vCPU 數相同，但當 KVM Host vCPU 個數增加，可以使得 guest 和 host 間的 vCPU 資源爭奪情況緩解，減少因為 host 要使用 vCPU 而停止 guest 使用 vCPU 的情況，以縮短 hackbench 時間。
- 當 KVM Host 的 vCPU 個數增加時，kernbench 中的 elapsed time 會減少，這是因為當 vCPU 個數增加，scheduler 就有越多 CPU 資源可以分配給 編譯 linux kernel 的 process 去使用，使得 elapsed time 減少。
- 當 KVM Host 的 vCPU 個數增加時，kernbench 中的 user time 和 system time 會增加，這是因為當 vCPU 個數增加，就要處理更多 vCPU 之間的同步和溝通上的事情，因為 cache、register、I/O、memory bandwidth 是不同 vCPU 共享的，更多 vCPU 可能造成 vCPU 間爭奪這些資源，再來是有更多的 vCPU，scheduler 要考量更複雜的 scheduling scheme 和不同 vCPU 間的 load balancing 問題，這些原因會造成 user time 和 system time 些微增加。
- 但當 KVM Host 的 vCPU 個數增加時，kernbench 中的 interrupt 個數卻減少，這是因為 vCPU 個數增加，造成更多 vCPU 可以同時處理 task，減少 context switch 所需的頻率，雖然 cache、register、I/O、memory bandwidth 競爭情況上升，但 vCPU 資源的競爭情況減少了，這也會減少 context switch 次數。

Netperf & Apach2 Benchmarking：

- 當 Host vCPU 或是 Guest vCPU 從 UP 變成 SMP 時，不論是 Netperf 或是 Apache2 的量測結果來看，網路傳輸速度都會變快，這符合我的期待：
  因為當 vCPU 個數上升時，就有更多 vCPU 能處理network workload，使得傳輸速度變快。
- 但當 Host vCPU 和 Guest vCPU 個數大於等於 2 後，Netperf TCP Stream 和 TCP RR 得到的結果不再隨 vCPU 個數變好，我的猜測為：
  因為 KVM Host vCPU 和 KVM Guset vCPU 個數比調控為 1:1和 2:1，可能導致 scheduling overhead，另外網路的傳輸可能被 bound 在 network bandwidth，即使 vCPU 資源變多了，但 network bandwidth 資源仍有競爭情況。
- 但單論 Apache2 Benchmarking 結果來看，client 為 KVM Host 而 server 為 KVM Guest 的情況下，Transfer rate 或是 time per request 都有隨著 guest 和 host 的 vCPU 個數變好，這也是符合我的期待的：
  因為當 vCPU 個數上升時，就有更多 vCPU 能處理更多的 concurrent requests，使得傳輸速度變快。

1.2 Compare VM performance using transparent huge page versus regular page

1.2.1 Experimental Result

可以在編譯 linux kernel 清單中 disable transparent huge page：

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- -j30 menuconfig
```

進到 Memory Management options 中有 Transparent Hugepage Support 取消，並重新編譯即可：

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- -j16
```

實驗設定如下：

- KVM Host RAM：2GB
- KVM Host vCPU：8 vCPUs
- KVM Guest RAM：512MB
- KVM Guest vCPU：4 vCPUs
- KVM Guest turn off vhost

調整實驗變數為 enable 和 disable transparent huge page on KVM Host

Enable transparent huge page on KVM Host	Hackbench (Host) (s)	Hackbench (Guest) (s)	Kernbench (Host) Elapsed Time (s)	Kernbench (Host) System Time (s)	Kernbench (Host) User Time (s)	Kernbench (Host) Percent CPU (%)	Kernbench (Host) Context Switches (times)	Kernbench (Host) Sleeps (times)
Enable	0.922	4.838	459.51	2639.05	501.81	683	102908	33492
Disable	1.229	4.351	483.36	2661.80	505.76	655	157715	33318

Enable transparent huge page on KVM Host	Netperf TCP Stream (Host → Host) (Mbits/s)	Netperf TCP RR (Host → Host) (times/s)	Netperf TCP Stream (Host → Guest) (Mbits/s)	Netperf TCP RR (Host → Guest) (times/s)	Apache Transfer Rate (Host → Host) (Kbytes/sec)	Apache Time per request (Host → Host)(ms)	Apache Transfer Rate (Host → Guest) (Kbytes/sec)	Apache Time per request (Host → Guest)(ms)
Enable	2736.85	4130.32	1970.43	801.06	29197.34	1.390	422.07	1.053
Disable	2376.44	4256.71	2055.59	772.81	29207.62	1.390	373.70	1.189

1.2.2 Discussion

從實驗中可以觀察到的是：

Hackbench & Kernbench：

- 當 Disable transparent huge page 時，KVM guest 上的 hackbench 時間反而有些微地減少，這是因為當 enable transparent huge page 時，反而 KVM Host 要額外去管理 transparent huge page，因為 KVM 在 ARM 架構下，highvisor 在 EL1，可能因此放大了管理 THP 的負擔，使得 KVM Guest 的 hackbench 時間增加。

- 當 Disable transparent huge page 時，KVM Host 在 kernbench 下 context switching 次數反而增加了，原因可能為 enable transparent huge page 下，可以減少 TLB miss 或是 page fault 次數，進而減少 context switches 次數

Netperf & Apach2 Benchmarking：

- 不論是 disable 或是 enable transparent huge page 對於 Netperf & Apach2 Benchmarking 的結果並沒有太大的影響，猜測是網路效能和 memory management 沒有太大的關聯性。

1.3 Compare VM performance with vhost versus without vhost

1.3.1 Experimental Result

透過更改 run-guest.sh 中的 vhost=on/off 來開啟和關閉 vhost

實驗設定如下：

- KVM Host RAM：2GB
- KVM Host vCPU：8 vCPUs
- KVM Guest RAM：512MB
- KVM Guest vCPU：4 vCPUs
- KVM Host enable transparent huge page

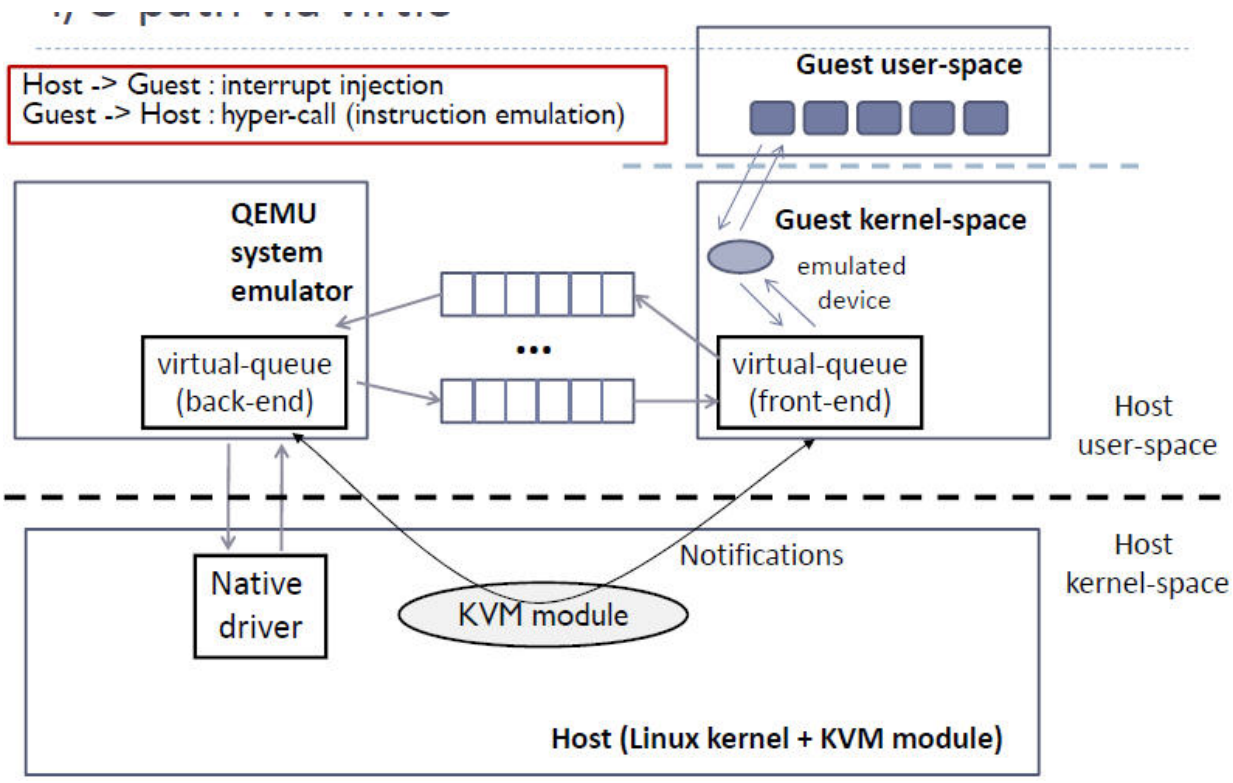
調整實驗變數為 turn on 和 turn off KVM Guest

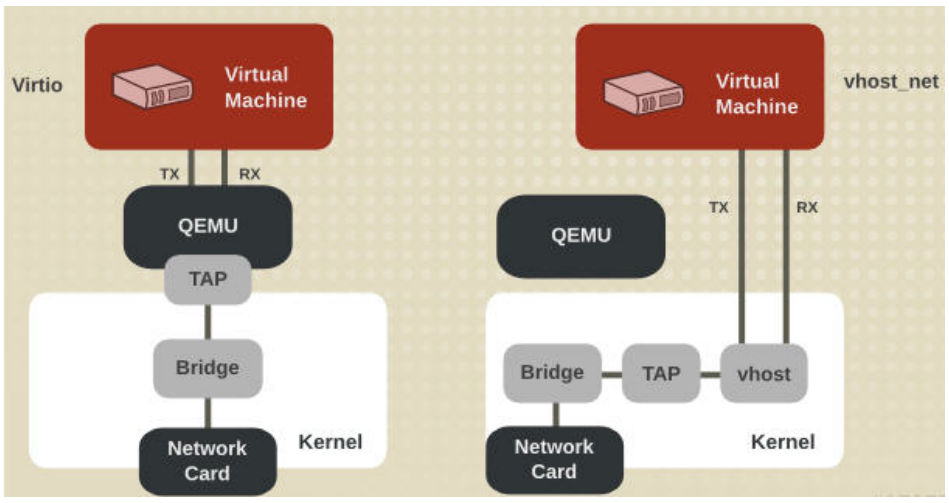
vhost	Netperf TCP Stream (Host → Guest) (Mbits/s)	Netperf TCP RR (Host → Guest) (times/s)	Apache Transfer Rate (Host → Guest) (Kbytes/sec)	Apache Time per request (Host → Guest)(ms)
on	2931.85	1593.22	307.45	1.445
off	1970.43	801.06	422.07	1.053

1.3.2 Discussion

從實驗中可以觀察到的是

- Turn on vhost 會造成 TCP Stream 和 TCP RR 得到的結果變好，這是因為原先使用 virtio-net 當 KVM Host 要傳輸資料給 KVM Guest 時，要 interrupt 到 QEMU 來取資料放到 virtqueue 中，再 interrupt injection KVM Guest 通知 Guest 去 virtqueue 取資料。
  但使用 vhost 可以使得 virtio-net dataplane offload 到 KVM Host 中處理，減少 kernel space 和 user space 的切換次數，提高 network 傳輸的 throughput 因此，TCP Stream 和 TCP RR 結果變好並不意外。





- 但 Turn on vhost 會造成 Apache2 benchmarking 得到的結果變差，原因猜測為 vhost-net 需要 KVM Guest 和 Host 共享同一塊 network device buffers，造成 memory 爭奪的情形使得對於某些 workload 來說，反而效能會變差。

2. VM performance comparison with KVM host

2.1 Experimental Result

實驗設定如下：

- KVM Host enable transparent huge page
- KVM Guest turn on vhost
- KVM Host RAM：2GB
- KVM Host vCPU：8 vCPUs
- KVM Guest TAM：2GB
- KVM Guest vCPU：8 vCPUs

	Hackbench	Kernbench Elapsed Time (s)	Kernbench System Time (s)	Kernbench User Time (s)	Kernbench Percent CPU (%)	Kernbench Context Switches (times)	Kernbench Sleeps (times)
Host	0.922	459.51	2639.05	501.81	683	102908	33492
Guest	5.204	933.45	4765.09	1468.69	667	199826	33104

Netperf TCP Stream (Host → Host) (Mbits/s)	Netperf TCP Stream (Host → Guest) (Mbits/s)	Netperf TCP RR (Host → Host) (times/s)	Netperf TCP RR (Host → Guest) (times/s)	Apache Transfer Rate (Host → Host) (Kbytes/sec)	Apache Time per request (Host → Host)(ms)	Apache Transfer Rate (Host → Guest) (Kbytes/sec)	Apache Time per request (Host → Guest)(ms)
2736.85	4130.32	1970.43	801.06	29197.34	1.390	400.91	1.108

2.2 Discussion

從實驗中可以觀察到的是：

- Hackbench 得到的結果中，Host scheduling latency 會比 guest 來得低，這是可以想見的，因為 Guest 得和 Host 爭奪資源，另外，要 guest 的 linux scheduler 要和 host 的 linux scheduler 協作會造成額外 overhead，增加 latency。
- Kernbench 得到的結果中，Guest 都比 Host 來得差，原因同樣，KVM Guest 跑在 KVM Host 上，Guest 得和 Host 爭奪資源，並且會更為頻繁地 context switch，造成結果更差。
- 反而 TCP Stream 的結果中，server 為 guest 的 throughput 較佳，可能原因為 enable vhost-net on guest 狀況下，造成 host → guest 的傳輸 throughput 提升。

Reference

- virtio 与vhost_net介绍_virtio和vhost_魏言华的博客-CSDN博客
- 网络虚拟化之virtio-net和vhost - 知乎 (zhihu.com)
- Zero-copy receive for vhost.pdf (linuxfoundation.org)
- Introduction to virtio-networking and vhost-net (redhat.com)
- Deep dive into Virtio-networking and vhost-net (redhat.com)