Report

My Environment

Bare-metal computer

- OS: Microsoft Windows 11 Pro, 64-bit (Build 22621.1555) 10.0.22621
- CPU: 12th Gen Intel(R) Core(TM) i7-12700 2.10 GHz
- RAM: 32.0 GB

First layer VM

- VMware® Workstation 17 Player 17.0.0 build-20800274
- CPU: 6 Processors
- MEM: 16 GB
- Hard Disk: 60 GB
- OS: Ubuntu 22.04.2 LTS 5.19.0-41-generic

Measurement Study Report (10 pts total)

VM setup checking

檢查 `CPU` 數量是否正確

1. `lscpu`

檢查 `transparent huge page` 有無如實關閉

1. `cat /sys/kernel/mm/transparent_hugepage/enabled`

檢查 `vhost` 有無如實開啟

> 備註: 檢查輸出之 `<pid>` 有對應關係

1. `lsmod | grep vhost`
2. `ls -lh /proc/$(pgrep qemu)/fd | grep '/dev'`
3. `ps -ef | grep '\[vhost'`

4. `ip -d tuntap`

實驗方法

hackbench

目標主機上測試 process 時間。

```
./hackbench 50 process 50

## hackbench output
# Running with 50*40 (== 2000) tasks
```

kernbench

目標主機上測試 compile linux kernel 時間。

```
cd linux-5.15
../kernbench -M -H -f -n 1 | tee >(grep 'Elapsed' | awk '{print $3 }' >> kernbench.t
```

netperf

以目標主機為 server、並以 KVM host 為 client，進而測試相關數據。

```
kvmperf/cmdline_tests/netperf.sh $ip
```

apache

以目標主機為 server、並以 KVM host 為 client，進而測試相關數據。

```
systemctl start apache2


kvmperf/cmdline_tests/apache.sh $ip 3
```

VM performance comparison different configurations (6 pts)

KVM host

- CPU: 4

- MEM: 8192

SMP VM versus UP VM

> Compare VM performance of SMP VM (2 virtual cores) versus UP VM (1 virtual core)

Setup named

A: KVM guest, CPU 2 MEM 4096, vhost=off, transparent huge page enabled
B: KVM guest, CPU 1 MEM 4096, vhost=off, transparent huge page enabled

hackbench

由於 hackbench 需要較高的 CPU 資源，因此由於 CPU 數量的不同，淺而易見的 B 的所需時間比起 A 還要長。

	Time
A	109.785
B	224.737

kernbench

由於 kernbench 需要較高的 CPU、MEM 資源，因此由於 CPU 數量的不同，淺而易見的 B 的所需時間比起 A 還要長。

	Elapsed Time	User Time	System Time	Percent CPU	Context Switches	Sleeps
A	3398.43	4788.48	1539.32	186.00	320453.00	30759.00
B	7789.97	5377.63	1726.72	91.00	512848.00	33675.00

netperf

TCP STREAM

> 十次取平均

雖然有點違反常理為何 CPU 數量較多，但TCP STREAM 表現卻較差；在這方面我的推測是由於在 single processor 的情況下排程問題相較起來沒有那麼嚴重，可能是因為排程演算法邏輯問題，導致在 netperf 下 B 的表現比起 A 還要好。

	Recv Socket Size (bytes)	Send Socket Size	Send Message Size	Elapsed Time	Throughput (10^6bits/sec)

		(bytes)	(bytes)	(secs.)	
A	131072	16384	16384	10.02	1169.18
B	131072	16384	16384	10.02	1742.68

TCP REQUEST/RESPONSE

> 十次取平均

雖然有點違反常理為何 CPU 數量較多，但TCP STREAM 表現卻較差；在這方面我的推測是由於在 single processor 的情況下排程問題相較起來沒有那麼嚴重，可能是因為排程演算法邏輯問題，導致在 netperf 下 B 的表現比起 A 還要好；除此之外，還有個值得觀察的點，A 在 TCP REQUEST / RESPONSE 中效能損失沒有在 TCP STREAM 之中嚴重，推測是因為對於 interrupt 的處理，仍然是 multi-processor 能具備較多的優勢。

	(Local/Remote) Socket Send (bytes)	(Local/Remote) Size Recv (bytes)	Request Size (bytes)	Resp. Size (bytes)	Elapsed Time (secs.)	Trans. Rate (/sec)
A	16384/16384	131072/131072	1	1	10.01	616.48
B	16384/16384	131072/131072	1	1	10.01	707.23

apache

> 由於多項數據取平均會失去其意義，故在三次實驗中取 **Time taken for tests** 最短 (最優) 之實驗結果做為最終比較之實驗結果。

雖然在 single processor 的情況下沒有排程問題，因此 B 在 Connection Times (Total, min)、Connection Times (Total, max) 表現較為優秀，但是由於 apache 不光是要處理 request，還要將 response 打包再回傳給 client，如此一來 CPU 越多越能快速地完成這些任務，因此論全體的表現、或是 median 來說，A 皆能夠表現的較為優秀。

	Time taken for tests (seconds)	Requests per second	Transfer rate (Kbytes/sec)
A	535.305	186.81	7582.20
B	982.091	101.82	4132.80

	Connection Times (Total, min)	Connection Times (Total, median)	Connection Times (Total, max)
A	67	522	1640
B	70	971	1460

transparent huge page versus regular page

> Compare VM performance using transparent huge page versus regular page (enable transparent huge page on KVM)

setup named

A: KVM guest, CPU 2 MEM 4096, vhost=off, transparent huge page enabled
B: KVM guest, CPU 2 MEM 4096, vhost=off, transparent huge page disabled

hackbench

由於 hackbench 需要較高的 CPU 資源，雖然 CPU 數量相同，但因為 KVM host 是否提供 transparent huge page 也造成了一些影響，transparent huge page 提供了對於連續性 MEM 的映射關係，不但如此還能有效地降低 TLB 的壓力；因此能夠發現雖然 hackbench 對於 CPU 要求較高，但是對於 MEM 也是有一定程度的需求，所以在 B 的所需時間比起 A 還要長。

	Time
A	109.785
B	136.986

kernbench

由於 kernbench 需要較高的 CPU、MEM 資源，原先我以為開啟 transparent huge page 能夠提供連續性 MEM 映射、降低 TLB 壓力，如此一來結果應該與 hackbench 結果一致，但經過思考我推測，因為 compile kernel 過程中需要反覆 compile、link 多個檔案，但是由於檔案多於零碎，因此推測開啟 transparent huge page 並未提供顯著的影響。

	Elapsed Time	User Time	System Time	Percent CPU	Context Switches	Sleeps
A	3398.43	4788.48	1539.32	186.00	320453.00	30759.00
B	3355.66	4598.68	1560.96	183.00	324856.00	32594.00

netperf

TCP STREAM

> 十次取平均

開啟 transparent huge page 能夠提供連續性 MEM 映射、降低 TLB 壓力，當網路塞車發生排隊時，有較好排列、較大且連續的 MEM 能對此情境有幫助，因此 B 比起 A 能稍微有較好的表現。

	Recv Socket	Send	Send	Elapsed	Throughput

	Size (bytes)	Socket Size (bytes)	Message Size (bytes)	Time (secs.)	(10^6bits/sec)
A	131072	16384	16384	10.02	1169.18
B	131072	16384	16384	10.02	1221.84

TCP REQUEST/RESPONSE

> 十次取平均

在 RR 的測試情境，需要處裡大量的 interrupt，此時開啟 transparent huge page 能夠提供的優勢卻無法套用至此情境，因此 B 與 A 表現略同。

	(Local/Remote) Socket Send (bytes)	(Local/Remote) Size Recv (bytes)	Request Size (bytes)	Resp. Size (bytes)	Elapsed Time (secs.)	Trans. Rate (/sec)
A	16384/16384	131072/131072	1	1	10.01	616.48
B	16384/16384	131072/131072	1	1	10.01	569.11

apache

> 由於多項數據取平均會失去其意義，故在三次實驗中取 **Time taken for tests** 最短 (最優) 之實驗結果做為最終比較之實驗結果。

開啟 transparent huge page 能夠提供連續性 MEM 映射、降低 TLB 壓力，但是在 apache server 之下，整體的表現沒有顯著地幫助；值得觀察的是，開啟 transparent huge page 後 Connection Times (Total, max) 表現較佳，推測是因為有了較好的 MEM 管理，當網路塞車發生排隊時，有較好排列、較大且連續的 MEM 會對此情境有額外的幫助，因此 B 在多數情況下與 A 表現一致，但是 B 能在 Connection Times (Total, max) 表現較佳。

	Time taken for tests (seconds)	Requests per second	Transfer rate (Kbytes/sec)
A	535.305	186.81	7582.20
B	545.780	183.22	7436.68

	Connection Times (Total, min)	Connection Times (Total, median)	Connection Times (Total, max)
A	67	522	1640
B	83	535	1378

with vhost versus without vhost

> Compare VM performance with vhost versus without vhost (toggle vhost using `vhost=on/off` in `run-guest.sh`)

setup named

A: KVM guest, CPU 2 MEM 4096, vhost=off, transparent huge page enabled
B: KVM guest, CPU 2 MEM 4096, vhost=on, transparent huge page enabled

hackbench

能夠觀察到 vhost 開啟與否對於 CPU 與 MEM 資源的影響較小。

	Time
A	109.785
B	104.919

kernbench

能夠觀察到 vhost 開啟與否對於 CPU 與 MEM 資源的影響較小。

	Elapsed Time	User Time	System Time	Percent CPU	Context Switches	Sleeps
A	3398.43	4788.48	1539.32	186.00	320453.00	30759.00
B	3122.90	4416.30	1445.29	187.00	300888.00	32483.00

netperf

TCP STREAM

> 十次取平均

開啟 vhost 與否都有 ring buffer 存在於 KVM guest 之中 (因為都是 virtio device)，而 TCP STREAM 不像是 TCP REQUEST/RESPONSE 有大量的 event 發生，但是由於 vhost 的 ring buffer 一端位於 KVM host kernel 那端，對於資料內容需要 copy_to_user，可能導致一定程度的效能損失；因此開啟 vhost 後在 event 部分能夠提供的輔助有限，但是資料搬移需要更長的時間，如此一來 B 比起 A 的表現略遜一些。

	Recv Socket Size (bytes)	Send Socket Size (bytes)	Send Message Size (bytes)	Elapsed Time (secs.)	Throughput (10^6bits/sec)

	Recv Socket Size (bytes)	Send Socket Size (bytes)	Send Message Size (bytes)	Elapsed Time (secs.)	Throughput (10^6bits/sec)
A	131072	16384	16384	10.02	1169.18
B	131072	16384	16384	10.01	1134.63

TCP REQUEST/RESPONSE

> 十次取平均

開啟 vhost 的其中一個優點便是可以使得 data-plane 移動至 KVM host，如此一來 B 大幅減少了 Context Switches 所需的次數，所以 B 在此項目表現的比起 A 還要好。

	(Local/Remote) Socket Send (bytes)	(Local/Remote) Size Recv (bytes)	Request Size (bytes)	Resp. Size (bytes)	Elapsed Time (secs.)	Trans. Rate (/sec)
A	16384/16384	131072/131072	1	1	10.01	616.48
B	16384/16384	131072/131072	1	1	10.01	879.29

apache

> 由於多項數據取平均會失去其意義，故在三次實驗中取 **Time taken for tests** 最短 (最優) 之實驗結果做為最終比較之實驗結果。

開啟 vhost 能降低處理 event 造成的效能損失，因此觀察 B 在 Connection Times (Total, min) 能夠表現比較優秀；但是開啟 vhost 時 ring buffer 一端位於 KVM host kernel 那端，如此一來會導致資料搬移時，需要使用 copy_to_user, copy_from_user，如此一來對於大量的 request 與 response 會造成一定程度的效能損失，因此整體來說 A 的表現相較於 B 比較好。

	Time taken for tests (seconds)	Requests per second	Transfer rate (Kbytes/sec)
A	535.305	186.81	7582.20
B	691.043	144.71	5873.43

	Connection Times (Total, min)	Connection Times (Total, median)	Connection Times (Total, max)
A	67	522	1640
B	29	668	2790

VM performance comparison with KVM host (4 pts)

setup named

A: KVM guest, CPU 2 MEM 4096
B: KVM host, CPU 2 MEM 4096

hackbench

雖然硬體規格相同，但是由於 A 被多包了一層 VM，如果 instruction trap (or page fault 等) 會導致 KVM 的 EL1 -> EL2 -> EL1 退至 KVM highvisor，再 EL1 -> EL2 -> EL1 才回至 VM guest，如此導致了效能損失，如此 A 比起 B 需要更長的時間才能完成。

	Time
A	109.785
B	39.137

kernbench

雖然硬體規格相同，但是由於 A 被多包了一層 VM，如果 instruction trap (or page fault 等) 會導致 KVM 的 EL1 -> EL2 -> EL1 退至 KVM highvisor，再 EL1 -> EL2 -> EL1 才回至 VM guest，如此導致了效能損失；值得觀察的是 kernbench 的效能損失沒有 hackbench 那麼嚴重，可能的推測是 kernbench 不需要像是 hackbench 那麼多 process 同時跑，因此 Context Switches 的次數也較少故造成的損失較少；但以結果來說，A 比起 B 仍需要更長的時間才能完成。

	Elapsed Time	User Time	System Time	Percent CPU	Context Switches	Sleeps
A	3398.43	4788.48	1539.32	186.00	320453.00	30759.00
B	1882.91	2903.06	561.95	184.00	210735.00	32236.00

netperf

雖然硬體規格相同，但是由於 A 被多包了一層 VM，導致 interrupt 等需要被 trap 之操作需要退至 KVM highvisor 處理；因此不論在 TCP STREAM 抑或是 TCP REQUEST/RESPONSE，B 皆表現的比 A 優秀。

TCP STREAM

> 十次取平均

	Recv Socket Size (bytes)	Send Socket Size (bytes)	Send Message Size (bytes)	Elapsed Time (secs.)	Throughput (10^6bits/sec)
A	131072	16384	16384	10.02	1169.18
B	131072	16384	16384	10.01	1780.02

TCP REQUEST/RESPONSE

> 十次取平均

	(Local/Remote) Socket Send (bytes)	(Local/Remote) Size Recv (bytes)	Request Size (bytes)	Resp. Size (bytes)	Elapsed Time (secs.)	Trans. Rate (/sec)
A	16384/16384	131072/131072	1	1	10.01	616.48
B	16384/16384	131072/131072	1	1	10.01	2315.51

apache

> 由於多項數據取平均會失去其意義，故在三次實驗中取 **Time taken for tests** 最短 (最優) 之實驗結果做為最終比較之實驗結果。

此測試同 netperf 觀察結果，值得觀察與說明的是 Connection Times (Total, min)，可以發現 KVM host 能夠 2 ms 內回傳網頁內容，這是因為 interrupt 無須 trap (雖然 B 還是在 vmware 之中，但是相比於 A 無須 trap)，所以可以壓至如此短的時間之內；整體來說 B 皆表現優於 A。

	Time taken for tests (seconds)	Requests per second	Transfer rate (Kbytes/sec)
A	535.305	186.81	7582.20
B	218.646	457.36	18563.28

	Connection Times (Total, min)	Connection Times (Total, median)	Connection Times (Total, max)
A	67	522	1640
B	2	206	1094