海量資料基於 Multipath TCP 之 40Gb 網路傳輸

楊哲男　　古立其　　周大源
國家高速網路與計算中心
yancgn@narlabs.org.tw

摘要

近年來大型主機處理速度與存儲裝置的 I/O 性能快速提升，使得即時處理大量資料知速度大為提昇，不同領域的資訊技術可以相互融合。許多的研究人員開始發展海量資料的處理分析技術，但是海量資料通常具有時效性，一旦串流到運算伺服器就須立即處理，才能使即時得到的結果發揮其最大價值，本文希望能夠透過網路效能量測工具，並使用高速傳輸之工具，同時使用多個 10Gb 網路路徑，加速海量資料傳輸的速度，使網路骨幹能夠被有效的利用。

關鍵詞：海量資料、高速網路、FDT、iperf、MPTCP

Abstract

The recent processing power and the I/O throughput advancement of computer systems creates the opportunity for multi-disciplinary big data analysis. Most big data need to be processed in a timely manner, therefore the quick transfer of big data becomes very important. This study proposed a mechanism to combine the network performance measurement tools and the high speed data transfer tools. Running on top of multiple concurrent 10Gbps links, the mechanism effectively utilized the full potential of the network backbone and maximized the big data transfer rate.

Keywords: Big Data、High-performance Network、FDT、iperf、MPTCP

1. 前言

當一台主機有多條對外網路時，使用者一定會希望可以同時使用多條網路來增加頻寬，此類相關的技術可以使用 Layer 3 的 Equal Cost Multipath (ECMP) 路由技術或是 Layer 2 的 TRILL (IETFRFC 5556)及 IEEE 802.1aq 的技術[1]。此類技術是利用網卡位址(mac address)、IP 位址以及 TCP/UDP 的使用埠(port)來計算網路流(flow)的雜湊值(hash value)，以作為負載平衡(load balance)的根據。例如：Hash(IPsrc, IPdst, Prot, Portsrc, Portdst) mod #oif。當來源及目的端之 IP 一樣時，同一個 TCP 連線只會跑再同一條路徑上，除非故意使用不同的來源及目的埠。當傳輸資料有很多網路流時，可以分散在不同的路徑上傳輸，但是當網路流很少時，例如大資料為導向的科學應用，就有可能只使用其中一條或是兩條線路上，因此而造成部分線路被閒置；此外，由於雜湊的計算演算法，會產生一樣比例的雜湊值，所以主機必須使用頻寬大小一樣的網路頻寬才可以使用這些方法。本文將簡單介紹 Multi-Path TCP (MPTCP)運作原理，使用 MPTCP 並於搭配多張 10Gb 網卡之主機上驗證不同網路速度之頻寬亦可合併使用，並使用 iperf 網路效能測試工具及 FDT 檔案傳輸工具來測試 MPTCP 之效益。

2. 安裝工具介紹

在本章節中，我們先介紹 MPTCP 之內容，並簡單介紹用來控制每個網路卓加入或退出 MPTCP 機制之使用工具。目前適合用於海量資料傳輸之軟體相當多，我們利用 FDT 以及普遍使用之 iperf 網路效能量測工具來做為本次實驗之傳輸工具。

2.1 MPTCP

多路徑傳輸通訊協定（MPTCP）是傳輸通訊協定（TCP）的修訂版[2]，允許在單一傳輸連線中能同時有多個路徑。MPTCP 必須達成兩大相容性：應用程式相容性以及網路相容性。前者預期在 MPTCP 協定下，現今的應用程式不需要任何修改，即可運作，如圖 **1** 所示。而後者主要是明訂 MPTCP 應該能夠運行於現今 TCP 協定運作的網路上。傳送端與接收端的主機因不同的介面而有多條傳輸路徑。當兩個端點開始以 MPTCP 進行協議，他們會開啟額外的子網路流(subflow)。資料可以分散至兩條路徑傳送。

若兩個已安裝 MPTCP 的端點間要互相傳送資料，它們首先會交換連線識別碼，這些資訊稍後會用在增加新的子網路流到現存的連線中。在同一個連線中的子網路流會重組為單一的網路流，但在兩邊端點會共用同樣的傳送或接收緩衝區。MPTCP 針對每一個子網路流設定序號，用以偵測遺失並進行重傳，而連線等級的序號允許在接收端重組。連線等級的 acknowledgement 用以實作適合的流量控制[3][4]。

然而，針對傳統的 TCP 而言，假設兩條路徑具有不同的 IP 位址，則會因為資料封包的序號不連續，導致傳送錯誤。因此，在 MPTCP 中，必須要進行多個等級的序號編號，使得連線等級的序號能夠連續，且可在接收端進行重組。

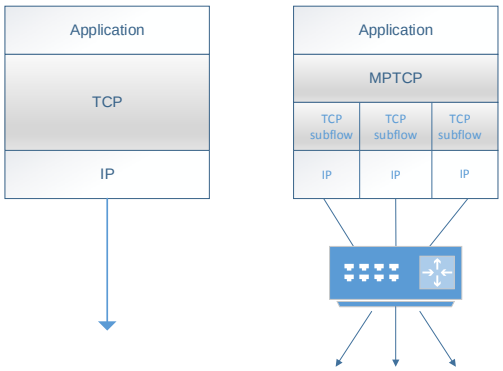

圖 1 TCP 與 MPTCP 之差異

2.2 FDT

FDT，全名是 Fast Data Transfer[5]，是由瑞士的 CERN 所開發的一種有效率的傳輸工具，它是由 java 語言所撰寫，因此可以跨平台使用。FDT 由於免安裝，只要直接下載 fdt.jar 這個檔案，即可執行。此外，它支援平行傳輸，並且可以指定 buffer size 之大小，因此可以大幅增加傳輸效率。FDT 是一種主從式架構的傳輸，檔案方向由使用者端(client)流向服務端(server)，其指令結構如下：

Server: java -jar fdt.jar [OPTIONS]
Client: java -jar fdt.jar [OPTIONS] -c <host> [file] -d <destinationDirectory>

2.3 iperf

iperf 是一個 TCP 和 UDP 的性能測量工具，能夠提供網路吞吐效能(throughput)，以及震動(jitter)、封包遺失率(packet loss)等統計資訊[6]；從而能夠幫助我們測試網路性能，定位網路瓶頸。其常用之參數說明如下：

-s ：以 server 模式啟動，例如：iperf -s。

-c ：主機以 client 模式啟動，-c 參數後面所輸入 IP 為測試 server 端地址，例如：iperf -c 10.0.0.100。

-w：指定 TCP 窗口大小，此參數影響 TCP 傳輸之效能，必須根據實際網路環境，例如網路延遲、網路卡之速度以及網路效能之需求來設定參數，例如: iperf -c 10.0.0.100 -w 512k。

-p：指定服務器端使用的端口或客戶端所連接的埠，預設埠為 5001。

-P：同時使用之串流數(streams)，例如: iperf -c 。10.0.0.1 -P 4，表示同時有四個串流傳輸，此參數可增加 TCP 之網路傳輸效能，預設值為 1。

-t：測試時間，例如： iperf -c 10.0.0.100 -t 60，及表示測試時間為六十秒，預設時間為十秒。

3. 實作 MPTCP

本章節我們將介紹本文所需之測試環境軟硬體及相關設定，並簡單描述 TCP 效能之調整方法及參數。

3.1 測試環境

為了測試不同的負載平衡結果，我們利用兩台規格一樣之 HP DL380p Gen8 之伺服器，進行記憶體對記憶體之傳輸測試，避免磁碟速度影響測試結果，其詳細規格如表 **1** 。由於本次測試需達到 40Gb 之速度，因此在每台伺服器上我們選擇了使用兩張雙阜之網路卡，來達到 40Gb 之總和速度。每個網路阜與另外一台伺服器之網路阜對接，並設定在同一個網路網址區段，如圖 **2** 所示，其詳細位址如表 **2** 。

表 1 測試機器規格

伺服器型號	HP ProLiant DL380p Gen8
CPU 型號	Intel® Xeon® E5-2620 兩顆
RAM	72GB
網路卡	Broadcom NetXtreme II BCM57810 10 Gb dual port 兩張

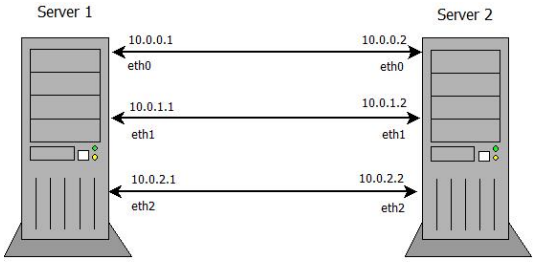

圖 2 測試主機架構

表 2 網卡 IP 位址分配

網卡配置	主機 1	主機 2
eth0	10.0.0.1/24	10.0.0.2/24
eth1	10.0.1.1/24	10.0.1.2/24
eth2	10.0.2.1/24	10.0.2.2/24
eth3	10.0.3.1/24	10.0.3.2/24
eth4	管理使用	管理使用

3.2 MPTCP 編譯

首先我們必須先修改 Linux kernel 裡之 Networking TCP 功能[7]，我們準備了兩台裝有 Centos 6.4 版本之主機透過六條單膜之光纖互接。依據 MPTCP 之建置方法，從官方網站下載原始檔，並修改核心編譯表。根據建置手冊，為了讓 MPTCP 能夠順利運作，必須取消 TCP-SYN

cookies、Net DMA 及 MD5 signature 之功能，並將 MPTCP 協定、IPv6 以及 Policy-Routing 之功能啟動。重新編譯作業系統之核心。為了能夠有彈性的控制每個介面可以任意加入或取消 MPTCP 之功能，亦可安裝 MPTCP 之管理工具-iproute-MPTCP。如果一切順利，可輸入指令驗證是否已完成 MPTCP 之編譯，如圖 **3** 所示。

```
$ sysctl -w net.mptcp.mptcp_enabled=1
net.mptcp.mptcp_enabled = 1
$ sysctl -w net.mptcp.mptcp_checksum=0
net.mptcp.mptcp_checksum = 0
```

圖 3 驗證 MPTCP 是否編譯成功

當我們編譯好 MPTCP 之後，必設定主機之路由表，由於 MPTCP 是以多個主網路流來傳輸封包，因此為了讓所有封包能夠正確傳輸，主機上的所有網路介面必須設定正確的路由表(routing table)，讓主機能夠了解從哪個來源來的網段，該從哪個出口出去。我們所使用的兩台主機，每台主機各有五個網路介面，其相關之 IP 資訊如表 **2** 所示。圖 **4** 為本次我們所設定之路由關係，以主機 2 來看，從 10.0.0.0/24 來的網段，主機 2 會讓他從 eth0 進入，從 10.0.1.0/24 來的網段，我們會讓他從 eth1 進入。除此之外，我們亦設定了一個管理介面，其 Gateway 的 IP 為 140.110.209.254，除了 10.0.0.0/24 至 10.0.3.0/24 以外的路由，我們皆讓他走 eth4 這個管理介面。

```
$ ip rule add from 140.110.209.100 table 1
$ ip rule add from 10.0.0.1 table 2
$ ip rule add from 10.0.1.1 table 3
$ ip rule add from 10.0.2.1 table 4
$ ip rule add from 10.0.3.1 table 5

$ ip route add 140.110.209.0/24 dev eth6 scope link table 1
$ ip route add default via 140.110.209.254 dev eth6 table 1
$ ip route add 10.0.0.0/24 dev eth0 scope link table 2
$ ip route add default via 10.0.0.1 dev eth0 table 2
$ ip route add 10.0.1.0/24 dev eth1 scope link table 3
$ ip route add default via 10.0.1.1 dev eth1 table 3
$ ip route add 10.0.2.0/24 dev eth2 scope link table 4
$ ip route add default via 10.0.2.1 dev eth2 table 4
$ ip route add 10.0.3.0/24 dev eth3 scope link table 5
$ ip route add default via 10.0.3.1 dev eth3 table 5
$ ip route add default scope global nexthop via \
  140.110.209.254 dev eth4
```

圖 4 主機 1 之路由表設定

3.3 iproute-MPTCP 編譯

iproute-MPTCP 是 iproute2 此工具之改良，讓使用者可以管理網路介面來加入或退出 MPTCP 之功能。從官方網站下載原始檔後，並且安裝適當的函式庫(library)，編譯完成之後，可輸入指令驗證是否已完成 iproute-MPTCP 之編譯，如圖 **5** 所示，若無錯誤訊息，就表示 iproute-MPTCP 已成功編譯完成。

```
$ ip link set dev eth0 multipath on
$ ip link set dev eth1 multipath on
$ ip link set dev eth2 multipath on
$ ip link set dev eth3 multipath on
$ ip link set dev eth4 multipath off
```

圖 5 MPTCP 功能驗證

3.4 效能參數設定

一般所常用的 TCP 網路傳輸控制協定，已經成為大檔案資料於網路傳輸之瓶頸[8]，若有需要使用 TCP 作為傳輸協定，必須做 TCP 優化調整之步驟，以增加傳輸效能，此外亦必須明確讓傳送及被傳送者知道 TCP/IP 之傳輸瓶頸問題，以利雙方共同改善。為了改善 TCP 的效能，在本文之測試環境中，我們做了許多調整，以增進 TCP 之效能，其調整之方法如下：

主機 buffer size 調整：

由於本文之模擬測試是利用 Centos 6.4 作為測試之作業系統，雖然此版本之核心支援自動調整之機制，不過系統的最大接收值(maximum receive window)太小，因此我們必須修改預設之最大接收值成為 32MB。

主機 MTU size 調整：

最大傳輸單元（Maximum Transmission Unit，MTU）是指一種通信協議的某一層上面所能通過的最大數據封包大小。MTU 數值越高，所得到之效能也會越好，但 MTU 設定越高，還必須示網路交換器或路由器是否有匹配的設定，如果 MTU 超過網路所負荷的量，也就是伺服器主機設定的值比網路交換器或路由器的設定值還大時，封包在傳送時會一再的分裝再重組，這會造成骨幹交換器的負擔，使網路傳輸效能低落，因此我們最好將網路 MTU 做最佳化調整。在本文之測試主機中，我們將所有阜之 MTU 調整至 9000 Bytes。

介面傳送佇列(tx queue)、CPU IRQ：

我們將每個介面之傳送佇列數設定為 10000，並將每個介面之 IRQ CPU Affinity 綁定在第四個 CPU 核心上。

測試時之 buffer size、傳輸串流數目：

由於主機 1 與主機 2 之 RTT 約 0.2ms 至 0.3ms，網路總頻寬為 40Gbps，因此根據 BDP 之計算方式[9]，我們將測試時所需之 socket buffer size 設定為 2MB，傳輸串流數目統一設定為 8 個。

4. MPTCP 模擬測試

為了驗證 MPTCP 之功能，一開始我們將所有

網路皆皆脫離 MPTCP 之設定中，將 eth0 至 eth3 共四個網路介面陸續加入 MPTCP 之路由表測試，並且利用 iperf 與 FDT 作為測試工具。

4.1 iperf 測試

在 iperf 之測試中，其步驟如下：

Step1：啟動兩台主機上的 iproute-MPTCP 工具，於主機 1 及主機 2 之 eth0 至 eth4 路由表正確設定，並將 eth0 加入 MPTCP。

Step2：利用 iproute-MPTCP 工具，於主機 1 及主機 2 之 eth1 加入 MPTCP，共兩個介面加入。

Step3：利用 iproute-MPTCP 工具，於主機 1 及主機 2 之 eth2 加入 MPTCP，共三個介面加入。

Step4：利用 iproute-MPTCP 工具，於主機 1 及主機 2 之 eth3 加入 MPTCP，共四個介面加入。

如圖 **6** 所示，我們只針對 eth0 之 IP 作為測試位址，透過測試結果可以發現經由 MPTCP 之功能，利用 MPTCP 將四個 10Gb 之網路介面合併使用可達到幾乎接近滿載之網路效能。

```
$ iperf -c 10.0.0.2 -w 1m -P 8
------------------------------------------------------------
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 2.00 MByte (WARNING: requested 1.00 MByte)
------------------------------------------------------------
[ 6] local 10.0.0.1 port 57577 connected with 10.0.0.2 port 5001
[ 3] local 10.0.0.1 port 57574 connected with 10.0.0.2 port 5001
[ 5] local 10.0.0.1 port 57575 connected with 10.0.0.2 port 5001
[ 7] local 10.0.0.1 port 57576 connected with 10.0.0.2 port 5001
[ 4] local 10.0.0.1 port 57583 connected with 10.0.0.2 port 5001
[ 9] local 10.0.0.1 port 57581 connected with 10.0.0.2 port 5001
[ 8] local 10.0.0.1 port 57584 connected with 10.0.0.2 port 5001
[10] local 10.0.0.1 port 57582 connected with 10.0.0.2 port 5001
[ID] Interval       Transfer     Bandwidth
[ 6]  0.0-10.0 sec  5.29 GBytes  4.54 Gbits/sec
[ 3]  0.0-10.0 sec  6.49 GBytes  5.58 Gbits/sec
[ 5]  0.0-10.0 sec  5.27 GBytes  4.53 Gbits/sec
[ 7]  0.0-10.0 sec  5.48 GBytes  4.71 Gbits/sec
[ 4]  0.0-10.0 sec  5.80 GBytes  4.98 Gbits/sec
[ 9]  0.0-10.0 sec  5.89 GBytes  5.06 Gbits/sec
[ 8]  0.0-10.0 sec  5.47 GBytes  4.69 Gbits/sec
[10]  0.0-10.0 sec  6.14 GBytes  5.27 Gbits/sec
[SUM] 0.0-10.0 sec  45.8 GBytes  39.4 Gbits/sec
```

圖 6 iperf with MPTCP 測試(4 ports)

4.2 FDT 測試

由 iperf 測試我們驗證了 MPTCP 的確可以達到同時使用多條網路之功效，本段我們使用是用於海量資料傳輸之工具-FDT 來做為驗證之工具。其步驟與 iperf 測試步驟相同，依序將 eth0 至 eth4 加入 MPTCP。圖 **7** 與圖 **8** 分別為 server 端及 client 端之測試畫面。

```
$ java -jar fdt.jar -S -bs 2M -p 54321
FDT [ 0.18.6-201307011457 ] STARTED ...

22/07 15:29:08   Net Out: 32.808 Gb/s      Avg: 31.258 Gb/s
```

圖 7 FDT server 端設定即測試畫面

```
$ java -jar fdt.jar -c 10.0.0.2 /dev/zero -p 54321 -P 5 –ss 2M -d /dev/null
FDT [ 0.18.6-201307011457 ] STARTED ...

22/07 15:29:08   Net In: 32.428 Gb/s      Avg: 31.428 Gb/s
```

圖 8 FDT client 端設定即測試畫面

4.3 測試結果分析

表 **3** 為 FDT 與 iperf 測試之結果，當傳輸介面只利用一個 10Gb 阜使用時，其所得到之結果皆為 9.8Gbps。當傳輸介面增加為兩個至四個 10Gb 阜時，我們可發現，iperf 幾乎是接近所有 10Gb 之總和速度，而 FDT 約為 80%之總和速度。兩個工具之差異如圖 **9** 所示，從結果中我們發現，透過增加網路阜到 MPTCP 裡，確實可發揮多條網路之傳輸效能，對於有大資料傳輸需求之海量資料確實可發揮效益。

表 3 MPTCP 測試結果

	FDT (Gbps)	iperf (Gbps)
1	9.8	9.8
2	16.4	19.6
3	24.4	29.8
4	32.5	39.4

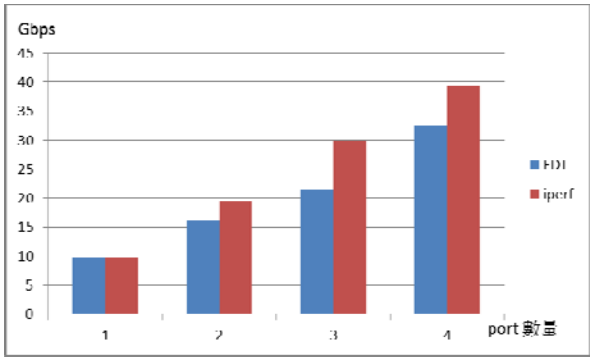

圖 9 FDT 與 iperf 之測試比較圖

5. 結論

在海量資料傳輸需求越來越大的網路世界裡，若可透過合併頻寬之技術，應用於儲存、高等物理或防災等網路應用上，將可替使用者加快網路之傳輸速度，未來單一 40Gb 阜及骨幹 100Gb 環境建立後，透過 MPTCP 將實際的達到每秒傳遞一部

DVD 影片之可能性。

參考文獻

[1] J. Touch and R. Perlman, Transparent Interconnection of Lots of Links(TRILL): Problem and Applicability Statement, RFC 5556, May 2009

[2] An overview of Multipath TCP - O. Bonaventure, M. Handley and C. Raiciu. USENIX login; , October 2012.

[3] C. Raiciu, M. Handley, and D. Wischik, “Coupled Congestion Control for Multipath Transport Protocols,” RFC 6356, October 2011.

[4] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, “Design, Implementation and Evaluation of Congestion Control for Multipath TCP,” USENIX NSDI, March

[5] FDT, http://monalisa.cern.ch/FDT/

[6] iperf, http://sourceforge.net/projects/iperf/

[7] MPTCP, http://multipath-tcp.org/pmwiki.php?n=Main.HomePage

[8] 楊哲男, “海量資料傳輸基於負載平衡協定之比較”, 2012 台灣網際網路研討會, Oct. 2012.

[9] speedguide, http://www.speedguide.net/bdp.php