An IaaS Cloud For Attack and Defense Experiments

蔡孟儒[1] 許晏峻[2] 吳育松[3]

國立交通大學資訊工程系

vtxyer@g2.nctu.edu.tw[1]

head2568.cs96@g2.nctu.edu.tw[2]

hankwu@g2.nctu.edu.tw[3]

摘要

本論文利用雲端運算中 IaaS (Infrastructure as a Service)的概念來架構一個可供資安相關課程做攻擊與防禦實驗的平台。透過硬體虛擬化技術，此平台可以非常有彈性地讓同學們於其上安裝各式的作業系統與軟體。不同組別的實驗間之網路可以做到完全隔離。更重要的是，透過一個反向的 NIPS 設計我們可以在兼顧平台對外網路連線便利性的前提下同時避免其上的攻擊實驗因操作不慎而波及到 Internet 上的無辜受害者。

關鍵字：Security、IaaS、Cloud Computing、Testbed

1. 前言

本論文著眼資訊安全相關課程需求，提供一個可以於其上從事攻擊與防禦實驗的平台；在平台上要能夠同時啟動約一百台虛擬機，並以組別為單位自成一個區域網路，讓網路內的虛擬機可以實際演練各種網路攻擊與防禦的技巧，且不讓具有攻擊性的封包流到區域網路外面，而對外部網路造成影響，甚至也不能影響到其他組別的區域網路中的虛擬機；除了網路環境外，也必須提供網頁介面讓課程同學可以從遠端對虛擬機作開機、關機、申請機器…等操作。由於目前各大 IaaS 雲端服務均不允許用戶於其上從事具有攻擊性的行為[6]，也因此針對資安課程的攻防實驗，並無法直接利用諸如 Amazon、Rackspace 等業者所提供的 IaaS 雲服務來實現之。

在攻防實驗中，往往需要針對特定的軟體漏洞而進行之，也因此平台必須要能夠讓使用者可以輕易地於其上安裝各種系統與軟體。然而，以 Emulab 為基礎的相關測試平台，在這方面就顯得稍受限制。比如目前 TWISC 提供的 Emulab 實驗平台就僅支援 FreeBSD 4.1, 5.4, 6.2 和 Fedora Core 6[5]。顯然這對於諸如要了解一些最新的 OS (如 Windows 7、Fedora 14、Ubuntu11、Redhat6、Debian6 等)相關的安全議題的實驗就無法於其上非常容易的進行之。

隨著現今硬體技術和虛擬化技術 (Virtualization technology)的進步，事實上要在一個實驗平台上支援各式最新的系統與軟體並不再是一件工程浩大、遙不可及的工作(比如需要開發相應的 kernel patch 或 driver 等)。在我們的平台中，我們使用 Xen Hypervisor 來提供一個虛擬的執行環境。在此環境上不管是最新的作業系統如 Windows 7 或甚至是古董級的系統如 Ubuntu 4.1 皆能順利運行於其上供同學從事相關的資安實驗。

由於在平台上會進行真實的攻擊行為，在虛擬網路的設計上我們透過了 iptables、Snort 等設置來達到隔離以及預防攻擊封包外流到 Internet 上的目的。

另一方面，由於我們目前的系統只有 5 個節點，為了要達到能於其上同時運行一百台虛擬機的目標。我們底層採用了分散式 file system 的設計。另外對於虛擬機的擺放位置，我們亦透過所開發的管控程式內部的 load balance 判斷機制來有效率地平均使用 4 個節點的硬體資源。本系統同時具有 fault tolerance 的設計，可以容忍單節點的失效(其上的虛擬機映像檔不會因此而遺失)。

使用者介面的部分，我們以網頁為介面來提供使用者新增需要的虛擬機，並且還提供了對所有的虛擬機做集中化管理的頁面，從該頁面可以清楚看到目前所有虛擬機的狀態，並且讓使用者可以從網頁介面將虛擬機開機、關機、休眠、刪除…等。

然而所有虛擬機都是在 NAT 之下的區域網路運作，介面上也提供讓使用者可以申請 NAT 對 port 的轉換，因此更可以模擬出真實網路的情境；並且預設每台虛擬機都會配置一個永久的 port 作為 VNC 連線的入口，如此使用者可以用雲端計算的概念透過遠端操作虛擬機，而登入密碼也會應組別不同而不同，因此沒有權限的組別將無法取得登入途徑。

總地來說，我們成功地利用各式相關技術搭建出了一個可供資安課程攻防實驗的平台。由於各大商業 IaaS Cloud 平台並不允許於其上從事諸如此類的攻擊行為，我們的平台也因此有其存在之意義。我們額外的防護措施如避免攻擊外流的設計亦是商業 IaaS 平台上所未見的。

* 本研究接受國科會編號: 99-2218-E-009-010 研究計畫經費補助

在第二章節將敘述相關研究，第三章節則說明整個平台架構的設計和功能特色，而第四章節會提出對於平台的相關實驗數據和效能測試，最後則對於本論文作一些結論。

2. 相關研究

2.1. Amazon Web Service(AWS)

AWS 包括了 Simple Storage Service (S3)、Elastic Compute Cloud (EC2)、Virtual Private Cloud (VPC)…等服務；Amazon EC2 在 2006 年 8 月 25 日發布，是一個使用 Xen 虛擬化技術來提供 IaaS 服務的系統，供用戶可彈性租用適合的機器來執行各種所需應用，用戶可經由 Amazon 網路服務介面創建、執行、終止屬於自己的虛擬機[9]。但是，像 Amazon 這樣一個商業性的大型服務平台並不允許使用在平台上進行具有危險性質的網路攻擊行為[6]，再者使用 Amazon EC2 是需要按時、按機器數目計費的，這對於經費有限的學校又是一個潛在障礙。

2.2. Eucalyptus

Eucalyptus 起初是由加州大學聖塔芭芭拉分校(University of California, Santa Barbara, UCSB)資訊科學系的一項計畫提出而被發展出來，是一套開放原始碼(Open source)的套件，讓系統管理者可以建立起 IaaS 私有雲的運算架構，而且擁有和 Amazon EC2 API 相容的特性，讓以 Eucalyptus 建立起的私有雲可和 EC2 整合[1][2][10]。然而若使用 Eucalyptus 仍必需要修改背後的系統架構，才有可能達到個組別網路隔離、攻擊性封包過濾、NAT port 轉換…等機制。此外，在使用者操作介面方面，我們也必須做到虛擬機是以多人同組為最小單位，並且還需要提供申請 NAT 對虛擬機 port 轉換的服務，這些皆需對 Eucalyptus 進行修改才能達成。

3. 平台設計

3.1. 概述

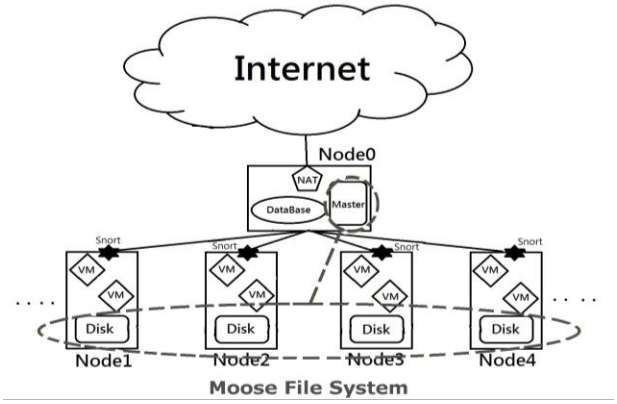

圖 1. IaaS 攻防實驗平台架構示意圖

圖 1 中，可看到目前主要有 6 個元件，上方 Node0 主要為 NAT、MySQL 資料庫和 Moose File System (MFS)的 Master Server，是和 Internet 溝通的出入口。(實作細節請參考 3.3)

而 Node1~Node4 為 Xen hypervisor、Snort 和 MFS 的 Chunk Server 與 Metadata Backup Server，並且分享出各自的磁碟。

最後 MFS 是一個邏輯上的元件，所以圖中以虛線表示，是由 Node1~Node4 分享出磁碟所構成的分散式 file system。(實作細節請參考 3.3)

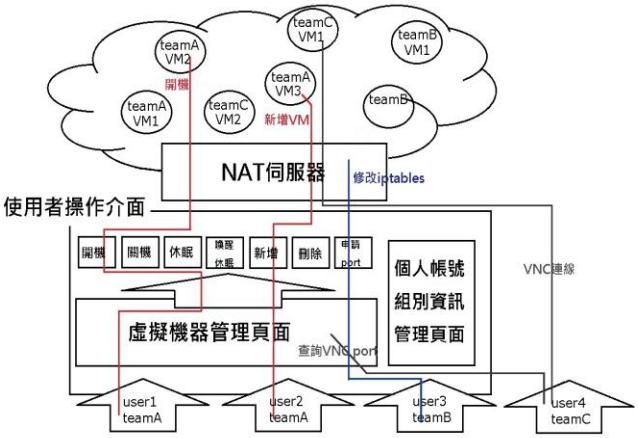

圖 2. 使用者操作邏輯介面

由圖 2 中，使用者登入後除了可編輯個人和組別資訊外，還可看到虛擬機管理頁面，在裡面可做開機、關機、休眠、喚醒休眠、新增虛擬機、刪除虛擬機、申請 NAT port 轉換等操作；並且可清楚看到所有虛擬機的狀態和相關資訊。

最後，可由 VNC 連線對虛擬機進行遠端實際操作。(詳細說明請參考 3.4.8)

3.2. 平台技術背景知識介紹

3.2.1. Virtual Network

Xen 的虛擬網路基本上有三種形式，分別是 routing、virtual NAT 和 bridging 三種形式。我們所採用的方式是利用 bridging[8]的方式。如此的好處是在實驗平台上面的虛擬機可以有自己的 IP。

3.2.2. Snort

Snort 是一套開放原始碼的網路 Intrusion Drevention System (IDS)軟體。Snort 可偵測惡意的網路攻擊封包，產生警告訊息並將其攔截。Snort 有三種封包監聽模式可以運行，分別是 sniff mode、packet log mode 和 inline mode。在 inline mode 下，Snort 會攔截住網路上的封包，在經過確認不是攻擊封包後才會將其放行至內部網路。在我們的系統中，我們採用的是 inline mode[4]。

3.2.3. Moose File System (MFS)

Moose File System 是一個分散式的 file system，他的操作就如同其他的 Unix 文件系統，擁有層狀架構(目錄樹)、文件的屬性(權限、修改時間等)，也提供 symbolic links、hard links、動態空間增減、延遲刪除和 snapshot 的功能，除此之外也可以對存取的權限加以控制[3][3] 。在 MFS 上主要擁有四種角色：

Master Server 是整個 file system 的核心，會將每個要儲存於 MFS 上的資料切成很多份，接著再將每一份依照需求，分別儲存於不同的磁碟上，最後這些資訊也就是所謂的 metadata 會被儲存於 Master 上，當下次要再寫入或讀取時只要依據 metadata 就可以知道資料的位置。

Metadata Backup Server 備份 server metadata 的 changelog 且定時的去下載 metadata file，所以當 Master 當機時，可利用這些備份檔案取代原本的 Master。

Chunk Server 實際儲存檔案資料的地方，並會與其他的 Chunk Server 作同步化，達到對每個 Chunk Server 的 load balance。且可以依據設定，對資料作冗餘儲存，達到 redundancy 的功能。

Client 利用 mfsmount process 將 Moose File System 掛載起來，掛載起來後能夠以操作本地端文件的方式操作，而不必再利用特別的 API 對其操作。而 mfsmount 是利用 FUSE 的方式實作，所以只要系統有支援 FUSE 就可以掛載 Moose File System，像是 Linux、FreeBSD、MacOS X…等皆有支援。

3.3. 平台實作原型 (Prototype)

基於上述的設計架構，我們搭建了一個由五個實體節點(機器)所組成的原型(Prototype)系統(圖 3)。五個實體節點分別為 Node1, Node2, …, Node5。在設計上為了要達到 portability (3.4.6)的需求，我們有一個虛擬節點 Node0。Node0 是執行在 Node5 上的唯一一台 VM，邏輯上可以把它看作就是 Node5。系統詳細的軟硬體配置如後。

3.3.1. 系統硬體規格

Node1~Node5 所使用的硬體型號皆為 TYAN GT20-B7002。每台機器所搭載的處理器為兩顆 INTEL XEON E5520 2.26GHZ (quad-core with hyperthreading)。記憶體為 DDR3-1333 4GB x4。硬碟為 SATA 1.5T 7200 轉 x2。網路交換器型號為 D-Link DGS-1008D 8-PORT 10/100/1000 SWITCH x2。

3.3.2. 主機系統軟體配置實作

Node1 到 Node4 都是用以承載使用者的 VM。機器所採用的作業系統是 CentOS5.5-64bits，搭配系統自帶的 Xen 3.0 當作虛擬機的 hypervisor。在 crontab 中加入指令” mysqldump”每分鐘固定備份 Node0 之資料庫，接著安裝 snmp、snort - 2.9.0.3 作為 IPS 和 mfs-1.6.20 作為分散式 file system。

(增加 loopback device) 在 Xen 的預設中是採用 file base 的映像檔儲存方式，此方式會將映像檔掛載到 loopback device 上，然而在 CentOS5.5 中預設的 loopback device 只有 8 個，這會導致在一個節點上無法開啟大於八台的虛擬機。因為此緣故我們必須增加 loopback device 的數量。利用 mknod -m660 /dev/loop$i 指令增加 loopback device 的數量，在我們的系統中是將其增加到 32 個。(也因此本系統的理論極限是可同時運行 4x32=128 台使用者 VM)

為了確保平台上使用者的攻擊封包不會外流至 Internet 上。我們在 Node1~Node4 透過 Snort 來過濾該節點上之 VM 對外(Internet)的流量。對於 snort-2.9.0.3 的安裝必須先安裝以下 library 套件 libcap、libdnet 、 libnfnetlink 、 libnetfilter_queue 、 libnetfilter_queue-devel、daq，以 "snort -D -Q --daq-dir=/usr/local/lib --daq nfq --daq-var queue=0 -l /var/log/snort -c /etc/snort/snort.conf"指令開啟 snort 的 inline mode 且選擇監控的介面為 kernel 中的 nfqueue，接著以後 snort 會對所有進入 kernel nfqueue 中的封包進行檢測以及操作，可在 snort rules 中選擇當偵測到符合規則的封包時是要 drop 或 alert，而在我們的系統中是選擇直接 drop。

Node5 機器採用 CentOS5.5-64bits 為作業系統，搭配系統自帶的 Xen 3.0 當作虛擬機的 hypervisor。但 Node5 並不會提供用戶虛擬機，此台機器的唯一功能是在其上開啟一台映像檔位於 MFS 上的虛擬機(其後皆稱為 **Node0**)。Node0 的作業系統是 CentOS5.5-32bits，功能如下：

- 當作為整個內部網路的 router，是整個內部網路對外的唯一出口。
- 當作 DHCP 伺服器，負責對所有虛擬機做網路配置，會將特定 MAC 位址對應到 IP 位址。
- MySQL 資料庫，所有與虛擬機的相關訊息以及狀態都會儲存在此。以集中式方式儲存，避免資料的不一致性，再利用 mysqldump 定期將資料庫作備份，確保資料的冗餘性。
- 當作 Moose File System 裡的 Master server，作為 metadata 的儲存處。
- 安裝 php snmp 以及 php ssh2 之 php 套件。

3.3.3. Moose File System 設定實作

首先介紹每台機器在 MFS 上所扮演的角色，Node0 是 Master server。Node1~Node4 會扮演 master backup server、chunk server 和 client。而 Node5 則是扮演 client。以下分別介紹每台機器的實作方式:

(Master server / Node0) 藉由修改 mfsmaster.cfg 文件，指定 metadata 預設儲存位置、開啟端口、預

設使用者等設置。接著修改 mfsexports.cfg 文件設定允許掛載 MFS 的 Client 和 Chunk Server，在我們的設定中僅允許 Node1 到 Node5 掛載，最後執行 mfsmaster 指令便可啟動 Master Server。

(Metadata Backup Server / Node1~Node4) 修改 mfsmetalogger.cfg 文件指定 metadata backup 存放的位置以及開啟 port 等相關設定，而後執行 mfsmetalogger 指令即可啟動 Metadata Backup Server。若在日後 Master Server 上的資料有錯誤即可利用將 Metadata Backup Server 上的資料複製到 Master Server 再利用 mfsmetarestore -a 的指令將 Master Server 的資料復原。

(Chunk Server / Node1~Node4) 修改 mfschunkserver.cfg 用以指定開啟 port 和所對應的 master server IP 等相關設定。接著再修改 mfshdd.cfg 指定分享的資料夾，在我們的設定中是將硬碟(附註 1)格式化成 ext4 的檔案系統，接著 mount 到/mfs 上分享。最後運行 mfschunkserver 指令便可啟動 Chunk Server。

(Client / Node0~Node4) 首先在/etc/hosts 上設定 mfsmaster 對應的 ip，接著藉由 "mfsmount file -H mfsmaster"指令將 MFS 掛載到本機上，之後使用平常操作 UNIX 文件的指令就可以操作位於 MFS 上的檔案，而不須透過特別的 API。最後可使用 mfssetgoal 指令設定每份文件的冗餘數，若設定為 2 則允許一顆硬碟損毀而不會破壞資料完整性，若設為 3 則可允許兩顆硬碟損毀，以此類推，但也會使用更多空間；而在我們的系統上是設為 2，也就是會保留 2 份相同文件，因此可允許一顆硬碟損毀。

附註 (1)：在硬碟的分享上因為我們 Node1~Node4 上都配有兩顆硬碟，而通常在 crash 時都是整台機器當機，所以若是一台機器損毀，同時會有兩顆硬碟無法提供服務，並無法達到實際的可靠性，針對此情形我們在分享硬碟前，先利用 LVM 的方式將硬碟結合起來；並且，我們使用 striping 的分式儲存資料，也就是以分散式儲存在兩個硬碟，而非線性儲存(當一顆硬碟容量用盡時，才儲存於另一顆硬碟)，藉此來提高硬碟傳輸效率。

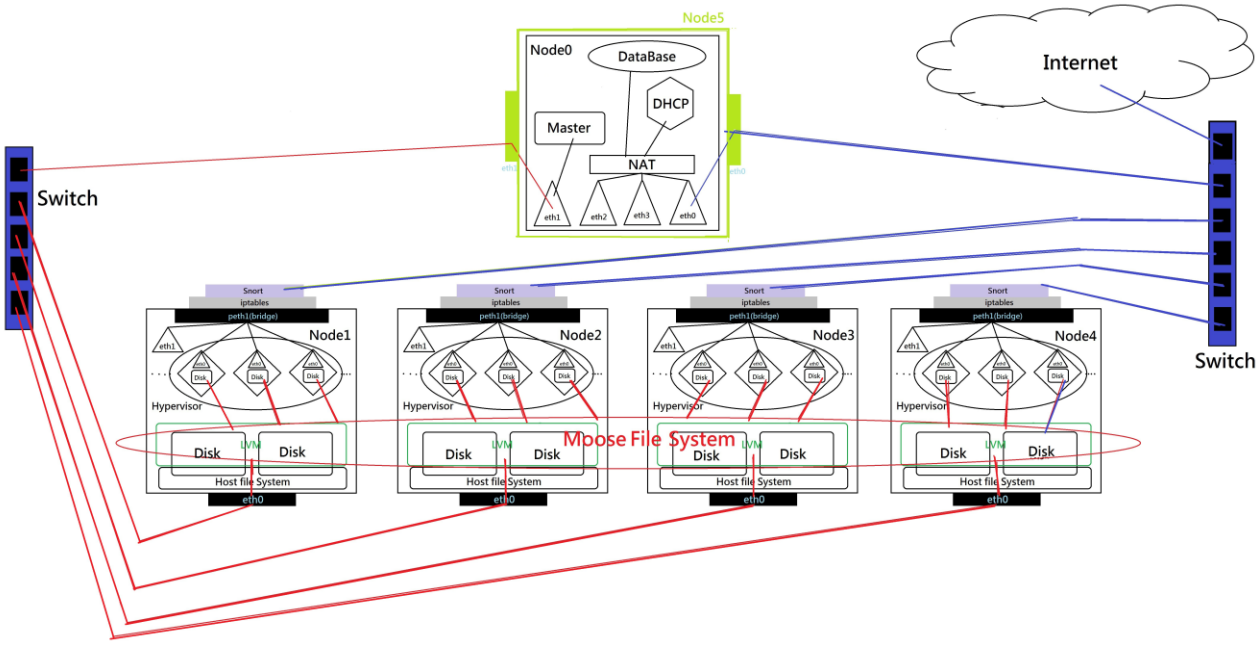

圖 3. Prototype 實作系統架構

3.3.4. 網路配置實作

Node0 總共有四張網路卡，分別為：

- eth0(public IP)用作對外網路的連結。
- eth1(192.168.247.1)當作 MFS 網路的 router，只針對 Node1~Node4 的 IP 做 route。
- eth2(192.168.128.1)用作與 Node1~Node4 內部網路溝通的 router，只針對 Node1~Node4 的 IP 做 route。
- eth3(192.168.20.1)用作與虛擬機網路溝通的 router。

Node1~Node4 四台機器都有兩張網卡，分別為：

- eth0(192.168.247.x)用於傳輸 MFS 間的資訊。當在 MFS 上有任何操作時，所有的流量都是在 eth0 上傳輸。
- eth1(192.169.247.x)以 bridge 的方式與虛擬機的網卡位於同一層下，所以不管是 Host OS 或者是虛擬機(Guest OS)，除了對 MFS 操作外所有的網路流量都是經由這張網卡傳輸。

四台機器上的 iptables 預設規則如下：

- -A FORWARD -s 192.168.20.0/255.255.255.0 -m physdev ! --physdev-out eth0 -j DROP (*將所有虛擬機的封包都丟棄*)
- -A POSTROUTING -s 192.168.20.0/255.255.255.0 -d 192.168.128.0/255.255.255.0 -m physdev ! --physdev-out eth0 -j DROP (*將虛擬機到 Node1~Node4 host 網路卡 eth1 的封包丟棄*)
- -A POSTROUTING -s 192.168.20.0/255.255.255.0 -d 192.168.247.0/255.255.255.0 -m physdev ! --physdev-out eth0 -j DROP (*將虛擬機到 Node1~Node4 host 網路卡 eth0 的封包丟棄*)
- -A POSTROUTING -s 192.168.20.0/255.255.255.0 -m physdev ! --physdev-out eth0 -j NFQUEUE --queue-num 3 (*將所有由虛擬網路出來的封包都先導向到 nfqueue*)
- 除了以上 4 條規則外，還會在新增虛擬機時再動態新增新的規則用於綁定虛擬機網卡 MAC address 和 IP address 間的對應關係以及組別間相對應之規則設定。

3.3.5. 新增虛擬機流程實作

系統中虛擬機的預設 CPU 為一顆、記憶體為 512M、磁碟大小為 10G。綜合以上架構的實作，當使用者創建一台虛擬機時，會經過以下步驟處理：

- 從四台 hypervisor 中去**挑選負載最低的機器來開啟虛擬機**，因為我們的機器上架有 snmp，所以可以依據 CPU 使用率、記憶體使用率、磁碟空間使用率或是已開啟虛擬機的個數來挑選負載最低之機器。
- 對我們預先安裝好的作業系統映像檔庫做 mfs snapshot 產生一個新檔案，再利用此檔案作為此虛擬機的映像檔，這些檔案都皆是放在 MFS 上。而因為 snapshot 可以在一秒內複製大小為 10G 的檔案，所以大大的增加虛擬機的創建時間。
- 指定虛擬機的 MAC 位址，告訴 DHCP 伺服器，而後 DHCP 伺服器就會依照機器的 MAC 位址，以相對應的 IP 分配到虛擬機中。
- 在每台 hypervisor 在 iptables 的 FORWARD chain 上，加入類似 "-A FORWARD -m mac --mac-source * -m physdev ! --physdev-out * -j ACCEPT"的增加規則，使 MAC 位址對應其特定的 IP，若都沒有對應到其指定 MAC 位址與 IP，則會因 iptables 的預設規則將封包丟棄。
- 在每台 hypervisor 上修改 iptables 的規則 POSTROUTING chain，加入 "-I POSTROUTING -s ip1 -d ip2 -m physdev ! --physdev-out eth0 -j ACCEPT "，使同組的 IP 互傳直接通過，而不會到 snort。但若是不同組別的封包，則須經過 snort 檢查才可以與其他組別的電腦溝通。修改 POSTROUTING chain 的原因是經過 iptables chain 的路徑是先過 FORWARD chain 才會到 POSTROUTING chain，所以一定要先經過 FORWARD chain 確定 MAC 位址和 IP 是正確的對應，才會對其封包做檢查。
- 更新 Xen 提供給虛擬機的 VNC 登入密碼，讓用戶可以透過遠端登入使用。如此一來，即便虛擬機本身的網路斷線，只要 hypervisor 上的網路正常，還是可以透過 VNC 連線來使用機器，不用管理者介入修復。
- 更新 Node0 上的資料庫，將所有關於虛擬機的相關訊息都寫入資料庫中，如此一來可方便對虛擬機作集中化的管理。

3.4. 系統特色介紹

3.4.1. MFS 特色

(Load Balance) 因為數據是儲存在不同的硬碟上，所以對於硬碟讀寫負載也會被平均分配到每個硬碟上，不會只對單一的硬碟造成負擔，也可提高硬體資源使用率。

(Snapshot 機制) 由於 MFS 提供對檔案進行 snapshot 的功能，且支援 live snapshot，換句話說正在進行檔案讀寫的檔案下也可以對其作 snapshot。此機制可以幫助我們更迅速的建構出一個映像檔，在一秒內就可以創建大小為 10G 的映像檔，大大的增加我們新增一台虛擬機的速度，且還可以減少對硬碟的空間的使用。

(Chunk Server 數據同步化) 每當新增或刪除一顆硬碟時，Master Server 就會自動告知所有的 Chunk Server，此時所有的 Chunk Server 會做資料的同步，將資料移到新的硬碟上，對每顆硬碟做 load balance。

3.4.2. 區域網路隔離 (Network Isolation)

我們的虛擬機是使用 bridge 的方式連接到 hypervisor 上，所以和 hypervisor 是位於一樣的網路層，而非呈現一種主從關係。虛擬機可以直接與 Node0 做溝通而不必經由 hypervisor 的導向，所以可以將 hypervisor iptables 中的規則設為，將虛擬機傳向 hypervisor host 的封包全部 drop 掉，如此一來可以杜絕虛擬機對 hypervisor host 有不正當的行為，降低 hypervisor 受攻擊的風險。

3.4.3. 阻擋攻擊性封包外流

我們透過一個反向 Network Intrusion Prevention System 來阻擋平台上的攻擊實驗，可能在無意間影響到外部網路的安全；並且，同樣地，我們也將此設計用在各個組別區域網路之間，因此各組別可不必擔心遭受到或成為外來攻擊源，致力於網路安全實驗。

3.4.4. 高擴充性 (Scalability)

在我們的架構中，隨著 hypervisor 機器數量的增加，則會因為虛擬機和 Snort 是分別架設於每一台 hypervisor 上，所以，總體所能使用的 CPU 和記憶體資源也是呈現線性的成長，而 Snort 過濾封包的負擔也可以平均分布於每個 hypervisor 上。至於在增加硬碟容量部分，因為使用了 MFS，每增加一顆新的硬碟，整體容量也就是所增加硬碟之容量，也是呈現線性成長。基於這兩點原因，說明我們的架構具有高擴充性。

3.4.5. 高可靠性 (Reliability)

在我們的架構中可能有三種當機情形，以下分別提出三種解決方案提升平台可靠性：

(Chunk Server 當機) 在此狀況下因為我們在 Moose File System 上的設定是每份資料都是以 2 份資料去儲存，所以可以容忍任一個 chunk server 損壞而不會發生錯誤。

(Node0 當機) 我們在一開始就有將 Node0 的映像檔利用快照做備份，並以此映像檔再開啟一台 Node0 的備份機器。且利用 Heartbeat 做監測，若當 Node0 當機，就可以迅速將 Node0 備份機器之網路設定成主要 Node0 之網路，再利用之前定期備份的資料庫將 MySQL 資料庫還原，最後，藉由 Metadata Backup Server 的將 metadata 資訊還原，使得 Master Server 能順利地讓整個平台順利運作。

(Hypervisor 當機) 因各個 hypervisor 間是獨立運行的，所以一個 hypervisor 當機並不會影響其他 hypervisor，只是位於當機的 hypervisor 上的虛擬機會因此而被強制關機，對此情況，可在其他 hypervisor 上重新啟動這些被強制關機的虛擬機。

3.4.6. 高可遷移性 (Portability)

對於使用者的虛擬機，由於虛擬機的映像檔是儲存於共用的 file system 上，所以當我們要對虛擬機做遷移時，可以直接利用 Xen 的指令 xm migrate，將虛擬機在我們的群集中做遷移，且 Xen 也支援 live migration[11]也就是虛擬機不須關機即可做遷移，即在使用者沒察覺的情況下對虛擬機作遷移也是辦得到的。

對於 Node0，由於其本身亦是一台虛擬機，理所當然地也可藉由 live migration 機制迅速遷移到其他機器上運作，因此當要將 Node0 更換到性能較好的 hypervisor 上也就是允許的。

3.4.7. 平台負載管控—自動休眠

基於要以現有四個節點做到支援 100 台使用者虛擬機同時啟動的效能，因此我們希望若虛擬機在沒有使用時，可以關機用以釋放資源給其他使用者，但實際狀況並無法強迫使用者養成關機的習慣，就理論上也比較不合理，所以我們在每台 hypervisor 上做 crontab 排程，定期檢查其上每台使用者虛擬機待機時間，若虛擬機已經開機超過 4 小時(理論上依課程需求較不會閒置 4 小時，此為可調整的參數)，且未至使用者操作介面點選續用功能，系統則會自動將虛擬機強制休眠。

休眠後的機器，記憶體以及相關狀態都會被存入檔案中，且將他所持有的記憶體以及 CPU 資源釋放出來，類似於關機狀態；而當使用者至使用者操作介面將虛擬機喚醒時，系統將重新分配負載較低的節點來重新開啟虛擬機，喚醒後的機器狀態會和休眠前一模一樣，因此並不會造成休眠前後資料不一致的狀況。

3.4.8. 使用者操作介面

使用者操作介面是以網頁形式呈現(參照圖 4)，大部分都是由 PHP 語言撰寫。登入機制上主要是依 PHP session 搭配後端資料庫為架構實作；在介面上，每個使用者都有個人帳號可登入，而權限則是以組別為基本單位，這是因應虛擬機的擁有者是以組別分配而設計。

虛擬機管理頁面提供對虛擬機的基本操作，包括新增(參照圖 5)、開機、關機、休眠、喚醒休眠、刪除(參照圖 6)；利用 PHP 提供的 API 對後端機器以 ssh 連線執行一系列我們所撰寫的 shell script，來自動完成這些動作，同時，也將網頁所需資訊存進後端 MySQL 資料庫中；在這個部分，我們考慮到當資料庫在與多個使用者互動情境之下有發生 race condition 的可能性，對此我們使用 PHP API 在各個 critical section 設置 file lock 的機制。

圖 4. 使用者操作介面

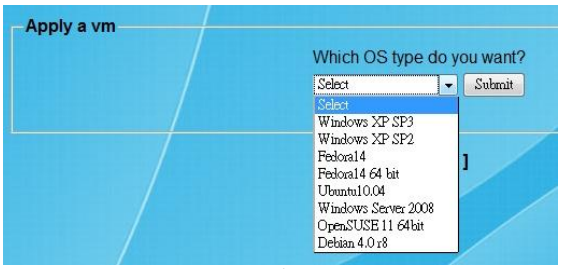

圖 5. 新增虛擬機頁面

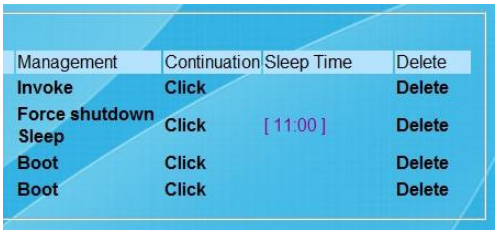

圖 6. 虛擬機管理頁面—基本操作

虛擬機管理頁面也提供查看虛擬機的所有資訊(參照圖 7)，只要在單一頁面就能夠看到虛擬機的 IP 位址、VNC 遠端連線位址、下次休眠時間、各項操作和目前狀態，包括開機 (running)、關機(shutdown)、休眠 (sleep)三種狀態，對於不同狀態，關機時提供開機和刪除兩種操作，休眠時則提供喚醒操作，而開機時可以做出關機、休眠和續用等三種操作；而續用的用意是在於為了考慮平台的負載程度，我們將在虛擬機開機 4 小時後將閒置的虛擬機休眠，因此若使用者不希望虛擬機自動休眠，可在休眠之前在此頁面續用，續用後虛擬機將重新計時 4 小時。

圖 7. 虛擬機管理頁面—虛擬機狀態資訊

當新增一台虛擬機實就會預設一個永久的 port 作為 VNC 連線的入口，以便使用者能順利遠端使用虛擬機；值得一提的是，這邊的 VNC 連線和虛擬機的網路狀態無關，而是架構在 Xen hypervisor 上的，因此就算虛擬機有任何網路上的問題，只要平台還持續運作，使用者就能以 VNC 登入虛擬機做修改操作；而至於 VNC 登入密碼部分，後端會依照組別而設置不同的組別密碼，這份組別密碼預設是以組別資訊加上 salt 並且經過 hash 運算後的結果，因此有密碼強度有一定的可靠度，當然，組別密碼也可由所屬的使用者自行更改。VNC 連線實際畫面請參照圖 8。

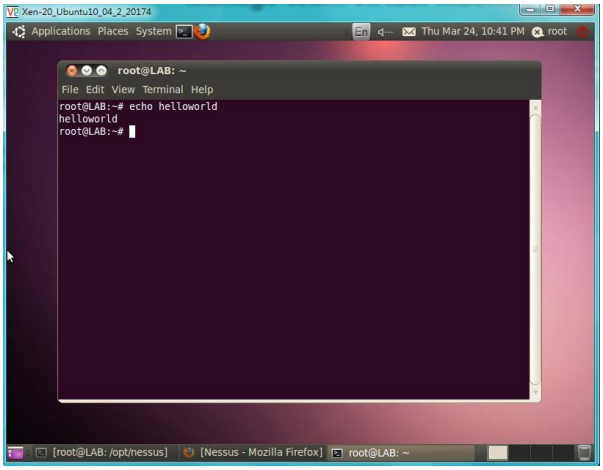

圖 8. VNC 連線畫面

圖 9. 虛擬機 port 管理

因應系統架構是讓所有虛擬機都在 NAT 之下，使用者會有開啟各種 port 來完成特定工作的需求，所以我們在介面上有提供一個管理虛擬機 port 頁面(參照圖 9)，使用者可藉此管道來了解目前 port 對應狀態和申請 NAT 的 port 轉換，這個部分也是利用 PHP API 來對後端 NAT 伺服器上的 iptables 做設定來達到。

在網頁安全性方面，對於各個輸入端都有作特定的字串過濾，因此較不必擔心有 SQL injection 等危險，而在執行有牽涉到後端資料動作前，也都會作二次檢驗，以避免偽造封包資訊所造成的潛在危險。

4. 實驗數據

4.1. 區域網路隔離測試實驗

首先，我們從使用者操作介面上新增一台虛擬機(作業系統為 Windows XP SP3，IP 為 192.168.20.24，組別為 20，參照圖 10)，對組內另一台虛擬機(作業系統為 Ubuntu 10.04.02，IP 為 192.168.20.13，組別亦為 20，參照圖 11) 使用 ping；接著，再對另外一組的虛擬機(作業系統為 Ubuntu 10.04.02，IP 為 192.168.20.15，組別為 33，參照圖 12)，使用 ping，最後查看網路相通狀況。

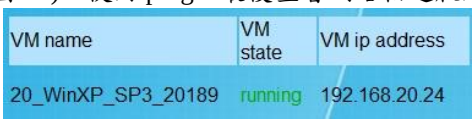

圖 10. 執行 ping 的虛擬機資訊(虛擬機管理頁面)

VM name	VM state	VM ip address
20_Ubuntu10_04_2_20174	running	192.168.20.13

圖 11. 同組虛擬機資訊(虛擬機管理頁面)

VM name	VM state	VM ip address
33_Ubuntu10_04_2_20176	running	192.168.20.15

圖 12. 非同組虛擬機資訊(虛擬機管理頁面)

由實驗結果(參照圖 13、圖 14)可看到流向不同組別的封包的確有被擋住，因此成功地將不同組別的區域網路隔離。

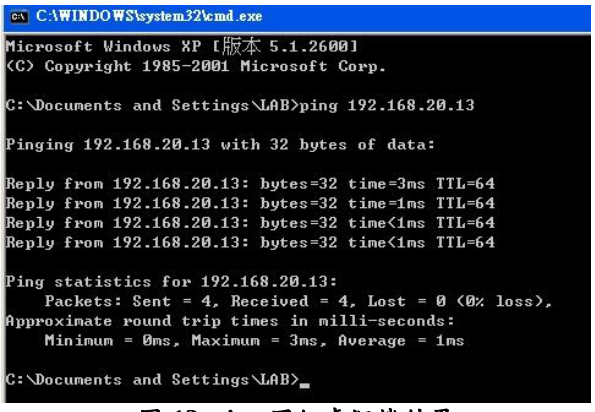

圖 13. ping 同組虛擬機結果

4.2. 阻擋攻擊性封包外流測試實驗

首先，我們從使用者操作介面上新增一台虛擬機(作業系統為 Windows XP SP3，IP 為 192.168.20.24)，並在上面安裝 Nessus 4.4.1 作為工具，實驗中，我們僅以 Nessus 所定義的基本 policy 來對 Internet 上已經架設好的實驗機器(作業系統為 Fedora 14)，進行滲透測試來模擬攻擊行為，並且查看/var/log/snort/alert 紀錄檔。

圖 15 是將 /var/log/snort/alert 紀錄來自 192.168.20.24 (實驗虛擬機)的攻擊性封包抓出來，可看到本實驗結果，基本的 policy 就觸發了 snort 53 種 alert 訊息；可見 snort 可以成功的阻擋攻擊性封包的外流。

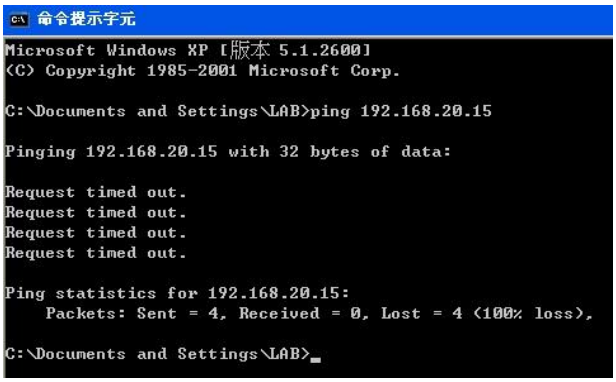

圖 14. ping 不同組虛擬機結果

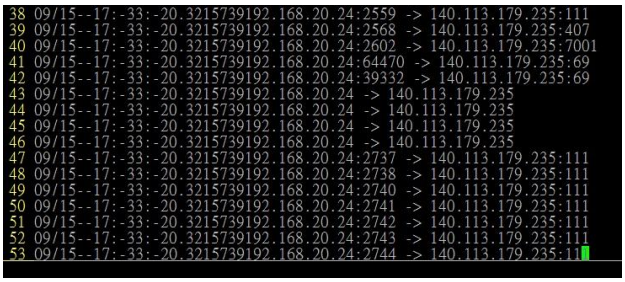

圖 15. grep 192.168.20.24 檢查 snort alert 次數

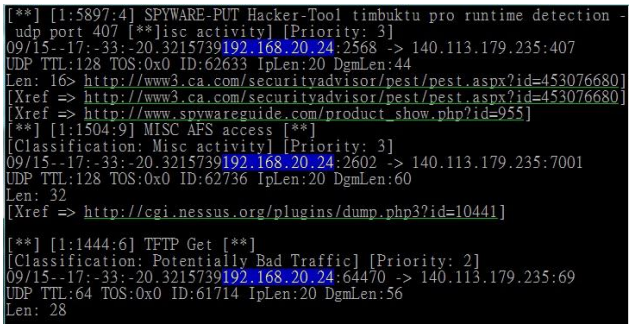

圖 16. cat /var/log/snort/alert 部分訊息截圖

4.3. 平台各項效能評測數據

在以下所有測試圖表中，**藍線**表示只在一個 Node 上開啟所有虛擬機(單節點)，而**紅線**所表示將所有虛擬機平均分散在四個節點中開啟(多節點)。至於 **X 軸**代表開啟之虛擬機的總數(在此測試中最多是同時開啟 26 台虛擬機)，**Y 軸**代表速率。

4.3.1. 磁碟讀取速度

- **此實驗是根據 'hdparm –t /dev/sda'作測試。**

由圖 17 所見，若虛擬機是分散在多台機器上(多節點)，則效能優於將節點集中在一台機器上(單節點)。而隨著機器數增多，讀取效能會些微下降。這是因為處於開機狀態的虛擬機本身也有些許的背景磁碟存取動作。然而 file system 主要之讀取頻寬還是由那台有在執行 hdparm 的 VM 所佔據。

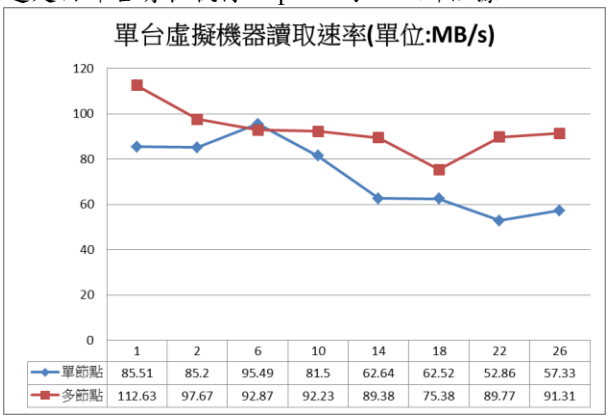

圖 17. 只在一台虛擬機作磁碟讀取測試，其他虛擬機僅處於開機狀態，並未參與測試。

由圖 18 所見，若虛擬機是分散在多節點上上，則效能優於集中在單節點上。而隨著機器數增多，讀取效能的下降幅度亦比單節點的情況來得緩和。

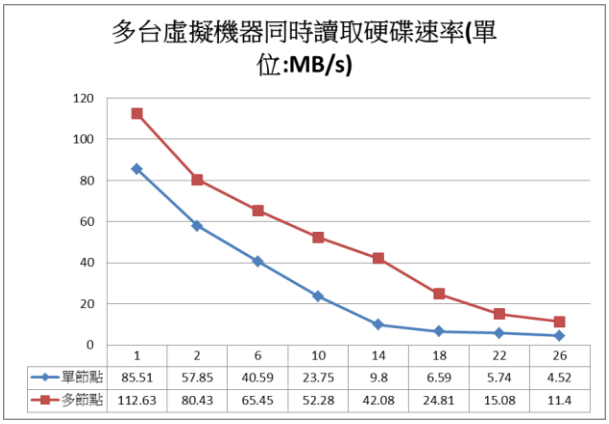

圖 18. 所有開啟之虛擬機同時作磁碟讀取

整體而言，我們可以看得出來把 VM 平均分布在多節點的情況之磁碟讀取表現都比把 VM 集中在單節點上來得好。這呼應了我們的在啟動 VM 時會依據節點負載來平均分布 VM 的設計是有其意義的(3.3.5)。另一方面，採用分散式 file system 的確也有助於均分虛擬機磁碟存取上的 overhead，使得系統能達到絕佳的延展性。

4.3.2. 磁碟寫入速度

- **此實驗是根據 "time `dd if=/dev/zero of=500M bs=10M count=50;sync`"作測試。**

由圖 19 所見，隨著機器數增多，寫入效能並不會明顯下降，這應該是因為 file system 之磁碟寫入頻寬還是由在執行 benchmark 程式的那一台機器獨享。而其他處於背景沒有太多磁碟動作的 VM 並沒有佔掉太多的磁碟寫入頻寬(相較於圖 17 所示的讀取頻寬)。

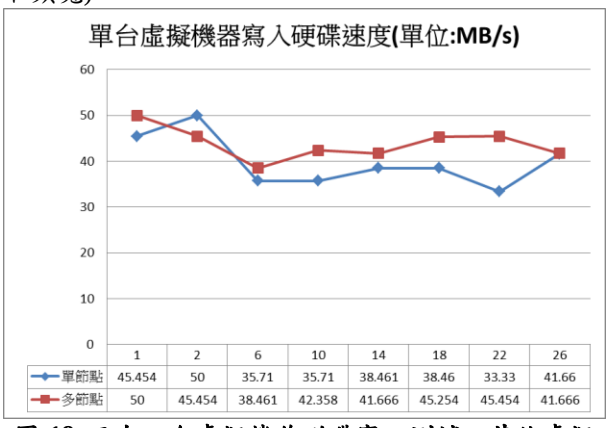

圖 19. 只在一台虛擬機作磁碟寫入測試，其他虛擬機僅處於開機狀態，並未參與測試。

由圖 20 可見，當所開啟的 VM 均進行磁碟寫入時，寫入速度就有明顯隨著機器數目增加而下降的趨勢。

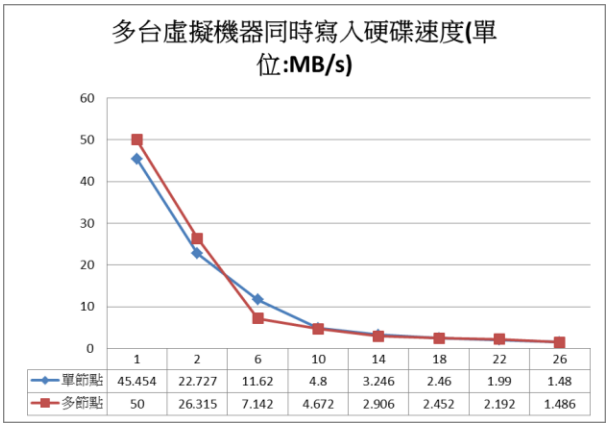

圖 20. 所有開啟之虛擬機同時作磁碟寫入測試

整體而言，在磁碟寫入效能方面，虛擬機分散在多節點並沒有顯著優於將虛擬機集中在單節點上的情況。我們推測因為在寫入磁碟時，需要與 Master Server 作的網路通訊更為繁雜，再者是因為寫入時為了容錯，需要建立兩個 replica（3.4.5），所需的網路傳輸頻寬更龐大，也因此效能瓶頸可能是出在網路傳輸上，因此並不會因為虛擬機分布在多節點上而有較佳的寫入性能表現。當每台機器都對磁碟寫入時，會有大量的頻寬在區網內傳輸，所以我們由特定網卡負責專門傳輸 MFS 上的頻寬資料是有意義的(可參考 3.3.3)。

4.3.3. 網路速度

- **此實驗是根據 'netperf –H [IP address]'做測試。**

由統計結果圖 21 可見，在只有一台虛擬機對外網連線時，各 node 上運行虛擬機的數量和對外速度並沒有特別明顯的關係。多節點的情況並沒有一個顯著會優於單節點的趨勢存在。這應該是因為對外之網路頻寬還是由一台機器獨享，而處於背景的那些虛擬機本身所佔據的連外頻寬是微乎其微。

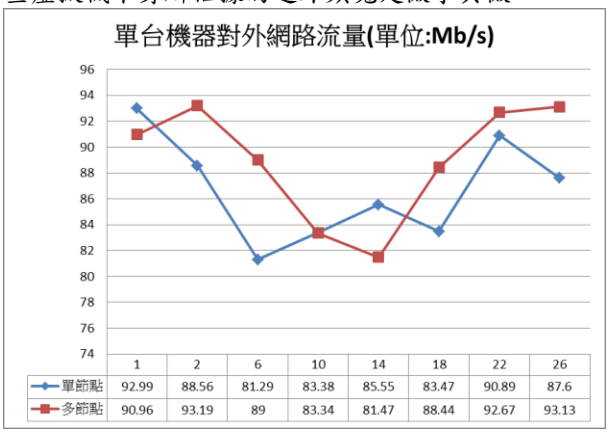

圖 21. 只在一台虛擬機作網路連外測試，其他虛擬機僅處於開機狀態，並未參與測試。

由圖 22 所見，網路頻寬會被開啟的虛擬機數量均分，因此多台機器對外速度和開啟的機器數量成

反比。所以若想增加對外網路速度，可藉由增加 Node0 對外之頻寬。

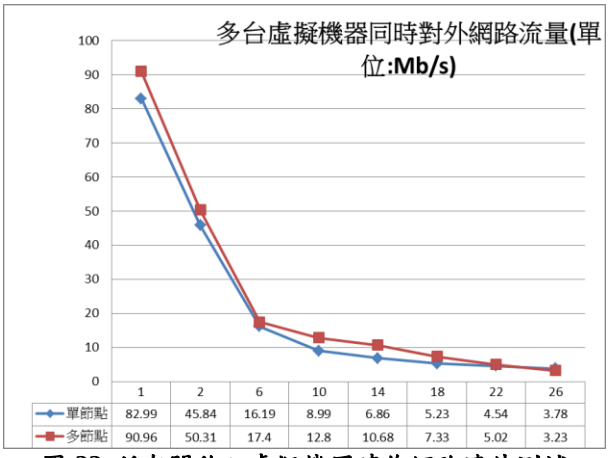

圖 22. 所有開啟之虛擬機同時作網路連外測試

由圖 23 所見，隨著虛擬機數量的增加，是偏向線性下降的趨勢，我們推斷是因為虛擬機網路只能撐到 200Mbits/s，因此在只有兩台虛擬機時，並無法將內網之 Gigabyte 網路頻寬用盡，而當虛擬機增加時利用到的頻寬也隨之增大，因此會有這種結果。而在單節點上和在多節點上，測試結果差異不大，是因為資料互傳都會經過 MFS 做處理，因此多台虛擬機間互傳時，會有大量的頻寬在區網內傳輸所以我們由特定網卡負責專門傳輸 MFS 上的頻寬資料是有意義的(可參考 3.3.3)。

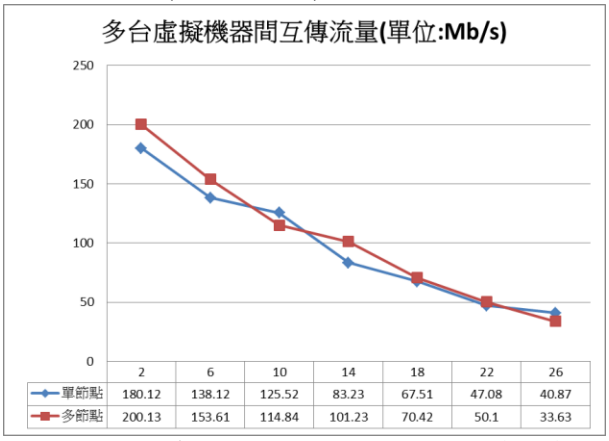

圖 23. 所有虛擬機間，兩兩成一對，同時互傳

5. 結論

本論文提出了一基於 IaaS Cloud 的資安攻防實驗平台，提供能達到同時開啟約 100 台虛擬機的能力，並且還使用分散式的 file system 架構來提高負載能力，也提高 fault tolerance 的能力；而比較特別的是，我們在網路方面，提供以組為單位的獨立網路，也就是說不同組的網路不會互相影響。此外，我們以反向的 Network IPS 阻擋了對 Internet 有攻擊性的封包。由於我們的平台規模並不大(只有 5 個 node)，目前只能提供本校內部相關課程的使用。然而透過本篇論文，我們將平台的設計以及一些運作上的經驗與各位同行先進分享。相信這對於有興趣搭設類似的攻防實驗平台的學校單位，會有一定的參考價值。

6. 參考文獻

[1] "Eucalyptus (computing) - Wikipedia, the free encyclopedia," Mar. 2011, Available at: http://en.wikipedia.org/wiki/Eucalyptus_(computing) [Accessed March 25, 2011].

[2] "Eucalyptus Community," Mar. 2011, Available at: http://open.eucalyptus.com/ [Accessed March 25, 2011].

[3] "Features, Architecture and Requirements :: MooseFS network file system - Moose FS," Mar. 2011, Available at: http://www.moosefs.org/ [Accessed March 25, 2011].

[4] "Snort IDS," Mar. 2011, Available at: http://www.snort.org/ [Accessed March 25, 2011].

[5] "Testbed@TWISC 啟用與服務說明," Available at: http://testbed.ncku.edu.tw/docfortestbed/Testbed@TWISC_1031_online_20071107.doc.

[6] Amazon.com, "AWS Customer Agreement," Available at: http://aws.amazon.com/agreement/ [Accessed March 25, 2011].

[7] D. Chisnall, E.M. Dow, T. Deshane, W. Hu, J. Bongio, P.F. Wilbur, and B. Johnson, *The Definitive Guide to the Xen Hypervisor*, Prentice Hall, 2007.

[8] J. Matthews, *Running Xen: A Hands-On Guide to the Art of Virtualization*, Prentice Hall, 2008.

[9] J. Murty, *Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB*, O'Reilly, 2008.

[10] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," IEEE CCGRID, 2009.

[11] C. Takemura, *The Book of Xen: A Practical Guide for the System Administrator*, No Starch Press, 2009.

[12] 蔡德明, "鳥哥的 Linux 私房菜 -- 利用 Xen 進行虛擬機的製作," Mar. 2011, Available at: http://linux.vbird.org/linux_enterprise/xen.php [Accessed March 25, 2011].