監視效能

執行效能是很重要的工作。當我們安裝、組態和穩定我們的系統，我們應該提供這最有效的系統效能。在這裏，我們提供一些基礎的工具來辨別和處理一些效能問題。快速的檢查可能問題區域簡化了效能解決的過程。當我們的系統效能突然低於平均應有的情況，可能是正在執行行程、記憶體的使用率、磁碟的效能、網路流量和 CPU 的飽合度。我們要解決這些問題來提高系統效能。一般的程序是決定整個系統的狀態，然後是檢查特定的子系統。有時後所有的狀態可能給我們可以給我們一些子系統的評估，但是最主要的還是我們需要去觀查這些子系統來查出真正的問題。當我們有經驗的使用這些系統後，我們將可以比較方便的查出影響效能的問題所在。

1-1 所有的系統狀態

我們可以使用 uptime 來得到整個系統的效能狀態。Uptime 指令得到這個系統已經開機多久了，有多少使用者登錄，系統在最近 1 分鐘、5 分鐘和 15 分鐘的平均負載。系統的平均負載顯示有多少工作在執行序列等待，行程準備執行但還未執行的有多少。下面狀態顯示最近 1 分鐘內系統的平均負載是 0.03，在最近 5 分鐘內系統的平均負載是 0.05，在最近的 15 分鐘內系統的平均負載是 0。我們也可以使用 cron 定時工作，每隔十五分鐘就記載其平均負載，這樣累計一天，我們就可以得到最進一天的平均負載。

```
[root@aasir chaiyen]# uptime
 10:26:57  up 4 days,  9:40,  3 users,  load average: 0.03, 0.05, 0.00
```

我們在 8 分鐘後使用 uptime 來觀看系統狀態，其平最近 1 分鐘和最近 5 分鐘的平均負載變成 0。因為 uptime 是快照這 15 分鐘以內的情況，所以只能看到最近系統的效能。

```
[root@aasir chaiyen]# uptime
 10:34:33  up 4 days,  9:48,  3 users,  load average: 0.00, 0.00, 0.00
```

系統活動報告(system activity reporter)我們簡稱為 sar，我們可以使用 sar 指令。這個指令指示 sar 來計數 count 次的範本其間隔為 secs 秒。

語法：

sar secs count

sar -u 1 10 這個指令指示 sar 每隔 1 秒總共 10 次來顯示 CPU 的使用率。-u 的選項告訴 sar 來報告 CPU 的使用率。%user 指示 CPU 花費執行使用者模式的百分比(使用在應用程式)，%nice 花費在非零 nice 優先權的使用者模式，%system 指示花費在核心(系統呼叫)，%idle 顯示 CPU 暫停的百分比。一般規則，假如%system 很高，應用程式可能使用系統呼叫。假如%idle 很高而 uptime 的報告為 5，這樣表示可能發生 I/O 或記憶體飽合的問題

```
[root@aasir chaiyen]# sar -u 1 10
Linux 2.4.20-8 (aasir.com)      西元2003年10月02日

10時41分15       CPU     %user     %nice   %system     %idle
10時41分16       all      0.00      0.00      2.00     98.00
10時41分17       all      0.00      0.00      0.00    100.00
10時41分18       all      0.00      0.00      1.00     99.00
10時41分19       all      0.00      0.00      0.00    100.00
10時41分20       all      0.00      0.00      1.00     99.00
10時41分21       all      0.00      0.00      0.00    100.00
10時41分22       all      0.00      0.00      1.00     99.00
10時41分23       all      0.00      0.00      2.00     98.00
10時41分24       all      0.00      0.00      0.00    100.00
10時41分25       all      0.00      0.00      0.00    100.00
Average:         all      0.00      0.00      0.70     99.30
```

sar 從/var/log/sa/sadd 讀取記錄檔的資料。/usr/lib/sa/sadc 這系統活動資料建立和維護這些記錄檔。然而大部份的 sar 實作包含兩個 shell 程式/usr/lib/sa/sa1 和 /usr/lib/sa/sa2。/etc/cron.d/systat 是 crontab 的系統進入檔，執行 sa1 每十分鐘一次，然後將它的輸出加到記錄檔後面。Sa1 以二元檔案格式集合和儲存 sar 所讀取的資料。

#vi /etc/crond/sysstat

```
# run system activity accounting tool every 10 minutes
*/10 * * * * root /usr/lib/sa/sa1 1 1
# generate a daily summary of process accounting at 23:53
53 23 * * * root /usr/lib/sa/sa2 -A
```

我們可以使用 xload 圖形程式來觀看系統的載入狀態。這個 xload 每秒更新一次，其尺寸大小為 1，前景顏色為黑藍。xload 程式顯示 uptime 指令的輸出。

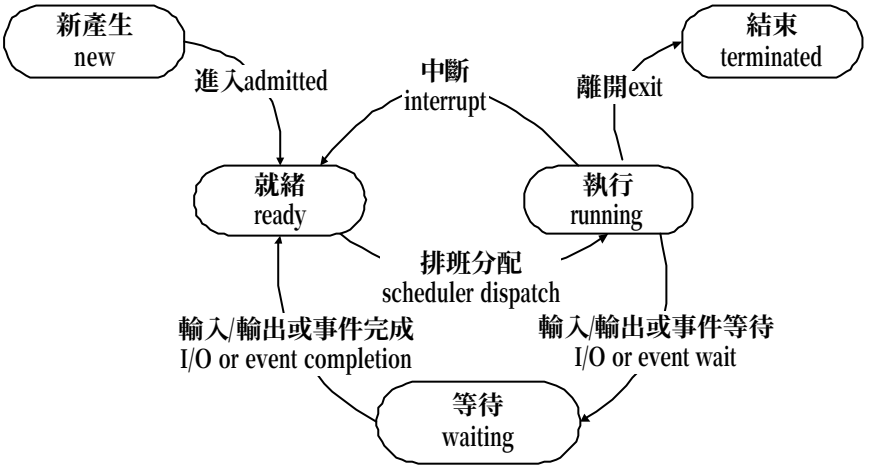
#xload –scale 1 –update 1 –fg darkblue –hl tan

xload 的選項與說明。

選項	說明
-scale count	在螢幕上顯示最少數量的線，每一條線代表 1 的平均負載。
-update secs	每一次更新的相隔時間多少 secs 秒。
-bg color	圖形背景的顏色。
-fg color	圖形前景的顏色。
-hl color	任何文字或尺寸的顏色。

1-2 監視執行行程

當我們執行程式時，電腦作業系統就會使用 new 幫我們產生新的行程，而當我們的行程就緒(ready)時，而核心又排班分配 CPU 中央處理器給他時，他就會開始執行 running，直到執行結束 terminated(或 Zombie)。當然在過程中也有可能發生像系統呼叫的 System call 而發生中斷，或者改成其它行程執行(被 preempt)，這時我們的行程就會回到就緒 ready 的狀態。當然在過程中也可能發生像輸入/輸出 I/O 或事件等待(sleep)，此時，CPU 是在閒置 waiting 的狀態，而當我們的輸入/輸出或事件完成時(被 wake up)，才會到就緒 ready，等待下一次排班分配 CPU 中央處理器，直到結束(zombie 或 terminated)。



行程狀態圖

我們使用 ps -el 的選項來觀看執行行程的資訊，-e 的選項會列出每一個正在執行的行程，-l 的選項可以產生較長的列表來顯示資訊。

```
[root@flash chaiyen]# ps -ellmore
F S   UID   PID  PPID  C PRI  NI ADDR    SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  75   0    -   342 schedu ?        00:00:04 init
1 S     0     2     1  0  75   0    -     0 contex ?        00:00:00 keventd
1 S     0     3     1  0  75   0    -     0 schedu ?        00:00:00 kapmd
1 S     0     4     1  0  94  19    -     0 ksofti ?        00:00:00 ksoftirqd_C
PU0
1 S     0     9     1  0  75   0    -     0 bdflus ?        00:00:00 bdflush
1 S     0     5     1  0  75   0    -     0 schedu ?        00:00:04 kswapd
1 S     0     6     1  0  75   0    -     0 schedu ?        00:00:00 kscand/DMA
1 S     0     7     1  0  75   0    -     0 schedu ?        00:04:58 kscand/Norm
al
1 S     0     8     1  0  75   0    -     0 schedu ?        00:00:00 kscand/High
Mem
1 S     0    10     1  0  75   0    -     0 schedu ?        00:00:00 kupdated
1 S     0    11     1  0  85   0    -     0 md_thr ?        00:00:00 mdrecoveryd
1 S     0    15     1  0  75   0    -     0 end    ?        00:00:01 kjournald
1 S     0    73     1  0  85   0    -     0 end    ?        00:00:00 khubd
```

欄位	說明
PID	為行程的編號，每一個行程都有它自己唯一的行程編號。
TTY	行程執行時的終端機。
STAT	行程的狀態
TIME	行程已經執行的時間。
CMD	行程被執行的指令名稱
USER	行程的執行用者
%CPU	所用 CPU 與所花費的時間的比率
%MEM	記憶體的使用率
SZ	VIRTUAL SIZE，行程在虛擬記憶體中的大小。
RSS	行程在實體記憶體中所佔的大小，單位 KBYTES
START	開使執行行程的時間
F	顯示目前行程的狀態，以數值表示。
S	顯示目前行程的狀態。
UID	行程執行者的使用者 ID
PRI	行程被排班的優先權
PPID	父行程的行程 ID
NI	Nice 的值，Nice 為降低優先權
WCHAN	等待頻道，當為 Null 空時，表示行程正在執行，當行程在就緒時為 Waiting for

這是第二欄位 S 的狀態 status 與說明。

行程狀態 state	說明
D	非中斷式 sleep(通常為 I/O 輸入/輸出)
N	低優先權行程(被 Nice 過的行程)
R	被排在執行佇列中，隨時會被執行的行程
S	Sleeping ，正在睡眠中
T	Traced 或 stopped 追蹤或停止
Z	Zombie，已經被停止的行程
W	被 swapped 到硬碟的行程

1-3 監視記憶體使用情況

如果我們實體記憶體太少，就會增加 swapping 的次數，效能就會減低。前面使用 ps –el 來觀看行程，其 sz 欄位就是檢查使用需擬記憶體的大小。在 Linux 上的 Swap 空間是在實體記憶體(RAM)用完時才會使用到，假如系統需要更多的記憶體資源，而實體記憶體已經用完，記憶體上不活動的頁面將會被移到 swap 。swap 空間可以幫助系統增加一小部份容量的 RAM，不過不能將它當作更多記憶體的替代品。Swap 空間是位於硬碟上，它的存取速度比起實體記憶體慢了很多。Swap 就是使用硬碟來當實體記憶體的替代品。Swap 空間可以是一個既定的 swap 分割區、一個 swap 檔案，或為 swap 分割區與 swap 檔案的結合。swap 空間的大小必須是我們電腦記憶體兩倍大的空間或者至少為 32MB，不過不能大於 2048MB(2GB)，因為太大會降低整體電腦的效能。

我們使用 vmstat 指令來檢查虛擬記憶體的子系統。vmstat 報告有關虛擬記憶體、CPU 負載、磁碟和 CPU 的活動等各項統計。Vmstat 每隔 secs 秒就會顯示它的標準輸出，總共 count 次。

指令：

vmstat secs count

```
[root@flash chaiyen]# vmstat 5 5
   procs                  memory      swap      io     system      cpu
 r  b  w   swpd   free   buff  cache   si   so   bi   bo   in    cs us sy id
 0  0  0   3936  19284 136868 183328    0    0    1    6   33   101  1  0 99
 1  0  0   3936  19284 136868 183328    0    0    0    6  102   215  0  0 100
 1  0  0   3936  19284 136868 183328    0    0    0    0  101   214  0  0 100
 0  0  0   3936  19284 136868 183328    0    0    0    0  103   216  0  0 100
 1  0  0   3936  19284 136868 183328    0    0    0    5  102   216  0  0 100
```

這是 vmstat 指令輸出欄位的說明。

Procs 欄位	說明
r	準備執行的行程數量
b	停止等待資源時的行程數量
w	被 swapped out 當等待資源時的行程數量

Memory 欄位	說明
swpd	Swap 空間的大小(KB)
free	未分配記憶體的大小(KB)
buff	緩衝記憶體的大小(KB)
cache	快取記憶體的大小(KB)

Swap 欄位	說明
si	從磁碟 swap in 到記憶體的數量
so	從記憶體 swap out 到磁碟的數量

io 欄位	說明
bi	寫到區塊裝置的區塊數量
bo	從區塊裝置讀取區塊的數量

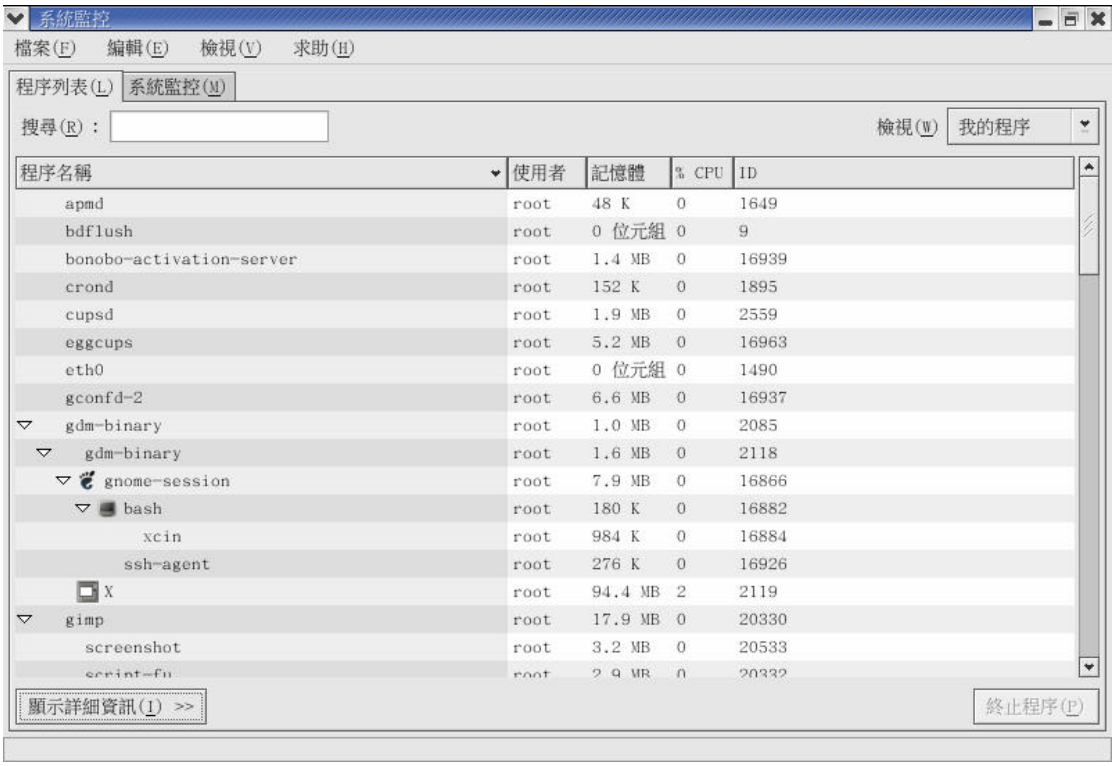
System 欄位	說明
in	裝置終段的數量
cs	CPU 行程切換的數量

CPU 欄位	說明
us	執行使用者端程式所花費 CPU 時間的百分比。
sy	執行核心/系統程式所花費 CPU 時間的百分比。
id	暫停所花費 CPU 時間的百分比。

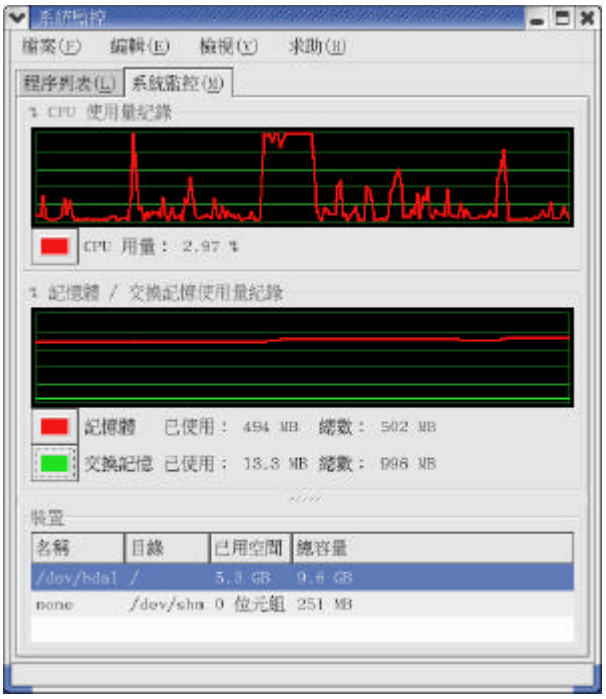
這 free 指令顯示記憶體的使用率，Swap 使用率和容量。Free 指令可以讓我們了解系統記憶體的使用情況。這所有都是以 kb 為單位。這裏第一行顯示了所有 total 記憶的容量，核心 used 分配多少記憶體，目前有多少記憶體是未使用 free，有多少是共享 shared 記憶體，有多少記憶體是用在緩衝和快取 cached。
第二行顯示記憶體的緩充使用調整。Free 未使用記憶體加上緩充 cached 和快取記憶體 buffers(19284+136868+183336=339488)，所以我們可以得到更精準的核心未使用量。Used 減去 buffers 和 cached(494520-136868+183336=174316)，所以我們可以得到更精準的核心使用量。
第三行顯示可獲得 swap 空間的數量，swap 使用的數量和 swap 未使用的數量。

```
[root@flash chaiyen]# free
             total       used       free     shared    buffers     cached
Mem:        513804     494520      19284          0     136868     183336
-/+ buffers/cache:     174316     339488
Swap:      1020116       3936    1016180
```

我們可以選取選單→系統工具→系統監控。
這是顯示每個行程使用記憶體的情況。



這是 CPU 和記憶體的使用記錄。



1-4 監視磁碟使用情況

我們可以使用 df 和 du 指令來觀看磁碟的使用情況。df 報告未使用的磁碟空間，du 報告已經使用的磁碟空間。

df 指令選項	說明
-a	在報告中包含空的檔案系統
-h	使用可讀單位來顯示報告
-k	使用 1k 的區塊
-l	只報告本地端的檔案系統
-m	使用 1024k 的區塊

我們使用 df –k 來觀看磁碟空間的使用。

```
[root@flash chaiyen]# df -k
檔案系統              1K-區段      已用      可用 已用% 掛載點
/dev/hda1            10080488   5531688   4036732  58% /
none                   256900         0    256900   0% /dev/shm
```

我們使用 df –h 來觀看磁碟空間的使用。

```
[root@flash chaiyen]# df -h
檔案系統              容量  已用  可用 已用% 掛載點
/dev/hda1             9.7G  5.3G  3.9G  58% /
none                  251M     0  251M   0% /dev/shm
```

我們可以使用 du 指令來觀看目前有多少磁碟空間已被使用。檔案可以是任何的檔案系統、裝置、目錄或檔案。

語法：

du 選項　檔案

du 指令選項	說明
-d	只報告特定檔案系統
-h	使用可讀單位來顯示報告
-k	使用 k 單位來顯示報告
-m	使用 MB 單位來顯示報告
-s	只報告總合

如果 du 沒有使用任何選項，它只會顯示該檔案的使用統計。

```
[root@flash chaiyen]# du /tmp
4       /tmp/orbit-root
4       /tmp/.font-unix
4       /tmp/.X11-unix
4       /tmp/.iroha_unix
8       /tmp/orbit-chaiyen
32      /tmp
```

這只報告/var 的磁碟使用率總合。

```
[root@flash chaiyen]# du -s /var
351212  /var
```

這是使用可讀的方式報告/var 的總合磁碟使用率。

```
[root@flash chaiyen]# du -smh /var
343M    /var
```

磁碟裝置是系統上最慢的一部份，主記憶體和 cpu 的速度比磁碟速度快上千萬倍。現在的 linux 核心和現代的磁碟使用軟體和硬體的快取換緩衝區來降低這些裝置的速度差異。有關磁碟的速度因素包括平均搜詢時間、最大搜詢時間、最小搜詢時間、查詢延遲、旋轉延遲、搜詢延遲和搜詢時間。

我們可以使用 iostat 來顯示 I/O 執行的問題。間隔秒數是報告的間隔秒數。總次數是報告的總次數。Device 是以 devm-n 來顯示實體裝置的分割，其中 m 是顯示主要編號，n 是顯示次要編號。

iostat 間隔秒數 總次數

```
[root@flash chaiyen]# iostat
Linux 2.4.20-8 (flash.aasir.com)        西元2003年10月02日

avg-cpu:  %user   %nice    %sys   %idle
           0.55    0.02    0.48   98.95

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
dev3-0            0.74         2.88        12.02    1087906    4539520
```

欄位	說明
tps	每秒傳輸到磁碟的數量
Blk_read/s	每秒從裝置讀取的區塊數量
Blk_wrtn/s	每秒寫入裝置的區塊數量
Blk_read	讀取區塊的總數
Blk_wrtn	寫入區塊的總數

我們使用 iostat –d –x /dev/had **來觀看詳細的** I/O **情況，** -x **指令報告延伸的** I/O **效能統計**。

```
[root@flash chaiyen]# iostat -d -x /dev/hda
Linux 2.4.20-8 (flash.aasir.com)        西元2003年10月02日

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await  svctm  %util
/dev/hda     0.18   0.94  0.18  0.54    2.88   12.02     1.44     6.01    20.63
    0.30   34.35 124.50   9.00
```

欄位	說明
rrqm/s	每秒合併讀取需求的數量
wrqm/s	每秒合併寫入需求的數量
r/s	每秒讀取需求的數量
w/s	秒秒寫入需求的數量
rsec/s	每秒讀取區段的數量
wsec/s	秒秒寫入區段的數量
avgrq-sz	需求的平均大小區段
avgqu-sz	需求的平均佇列長度
await	等待 I/O 平均的時間(milliseconds)
svctm	I/O 需求完成的平均時間
%util	被 I/O 需求消耗的 CPU 百分比

1-5 監視 CPU 使用情況

我們可以使用 top 指令來觀看 CPU 的使用情況。top 是即時的監視系統，它每五秒就會更新一次。我們可以使用 top 指令的選項，也可以使用 top 的交談模式。

top [-cCiqsS] [-d delay] [-p pid] [-n num]

top 選項	說明
-c	顯示用來行使行程的指令列參數
-C	顯示總數在 SMP 主機的 CPU 統計
-d delay	在更新時，暫停 delay 秒數
-i	忽略 idle 或 zombie 行程
-n num	刷新 num 次顯示，然後離開
-p pid	使用行程編號 pid 來監視行程
-q	在更新時沒有延遲的經常更新
-s	以安全模式執行，取消可能危險的交談性指令
-S	顯示每個行程和其子行程的總合統計

我們使用 top 指令來顯示情況。top 第一行顯示系統更新的時間，就像 uptime 指令的輸出。下面兩行是顯示 CPU 使用率的總合，一個是顯示啟動、sleeping、running、zombied 和停止的行程，一個是顯示 CPU 使用的百分比。第五行和第六行總合目前的記憶體使用率。

```
[root@flash chaiyen]# top
 09:30:41  up 4 days,  8:55,  1 user,  load average: 0.00, 0.01, 0.00
85 processes: 84 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:   0.5% user   0.4% system   0.0% nice   0.0% iowait  98.9% idle
Mem:    513804k av,  495120k used,   18684k free,       0k shrd,  135848k buff
                     323720k actv,       0k in_d,    7152k in_c
Swap: 1020116k av,    4136k used, 1015980k free                   183668k cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
16563 root      19   0  1124 1124   856 R     0.9  0.2   0:00   0 top
    1 root      15   0   104   80    56 S     0.0  0.0   0:04   0 init
    2 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 keventd
    3 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kapmd
    4 root      34  19     0    0     0 SWN   0.0  0.0   0:00   0 ksoftirqd_CPU
    9 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 bdflush
    5 root      15   0     0    0     0 SW    0.0  0.0   0:04   0 kswapd
    6 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kscand/DMA
    7 root      15   0     0    0     0 SW    0.0  0.0   4:59   0 kscand/Normal
    8 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kscand/HighMe
   10 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kupdated
   11 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 mdrecoveryd
   15 root      15   0     0    0     0 SW    0.0  0.0   0:01   0 kjournald
   73 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 khubd
 1490 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 eth0
```

欄位	說明
PID	行程的編號(process ID)
USER	啟動行程的使用者
PRI	行程的執行優先權 priority
NI	通常修改行程的執行優先權數值
SIZE	行程需求記憶體的大小
RSS	行程消耗的實體記憶體數量
SHARE	行程使用的分享記憶體數量
STAT	行程目前的 CPU 狀態
%CPU	行程消耗的 CPU 時間百分比
%MEM	行程使用實體記憶體的百分比
TIME	行程以經執行的 CPU 時間總量
COMMAND	啟動的行程，假如被 swapped out 則會被括號起來。

當 top 在執行時，我們可以按下一些選項；A 是行程排序根據 AGE；M 是行程排序根據實體記憶體的消費；N 是行程排序按照行程編號；P 是行程排序按照CPU的使用率；T是行程排序按照CPU時間的消費，下圖是 T行程排序按照CPU時間的消費。

```
 15:57:33  up 5 days, 15:22,  3 users,  load average: 0.18, 0.18, 0.12
113 processes: 110 sleeping, 3 running, 0 zombie, 0 stopped
CPU states:   9.1% user   0.9% system   0.0% nice   0.0% iowait  89.8% idle
Mem:    513804k av,  496856k used,   16948k free,       0k shrd,  120536k buff
                     380612k actv,       0k in_d,   10328k in_c
Swap: 1020116k av,   16928k used, 1003188k free                   197096k cached
Sort by time
  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
20315 root      16   0 12316  12M  8272 S     5.3  2.3  23:55   0 gnome-system-
20339 root      16   0 15320  14M 11180 S     1.7  2.9  10:43   0 ksysguard
 2119 root      15   0 96676  13M  1772 R     1.7  2.7   7:53   0 X
    7 root      15   0     0    0     0 SW    0.0  0.0   6:54   0 kscand/Normal
16961 root      16   0  3752 3008  2696 S     0.3  0.5   6:19   0 magicdev
20403 root      15   0   848  848   628 S     0.5  0.1   4:37   0 ksysguardd
16967 root      15   0 10336 9660  5568 S     0.0  1.8   0:12   0 rhn-applet-gu
16957 root      15   0 14400  13M  8472 S     0.0  2.7   0:06   0 gnome-panel
16975 root      15   0  9724 9720  5680 S     0.0  1.8   0:06   0 gnome-termina
16953 root      15   0  8612 8612  4684 S     0.0  1.6   0:04   0 metacity
    5 root      15   0     0    0     0 SW    0.0  0.0   0:04   0 kswapd
    1 root      15   0   104   80    56 S     0.0  0.0   0:04   0 init
 2035 xfs       15   0  7556 7556   304 S     0.0  1.4   0:03   0 xfs
20350 root      15   0 10756  10M  8404 S     0.0  2.0   0:03   0 kdeinit
 1696 root      15   0   656  104    52 R     0.0  0.0   0:02   0 snmptrapd
```

1-6 監視網路流量

現在經由網路犯罪的人越來越多，所以對網路的安全我們要更加注重。當同時有多人進入我站，將會造成流量擁塞，因此如何分散流量也是非常重要的項目。我們可以使用 netstat –i **指令來檢查封包流量，這是根據乙太網路界面。這個輸出顯示有兩個乙太網路界面，** lo **和** eth0**。最好錯誤的封包數量少** 1%。

```
# netstat -i
Kernel Interface table
Iface      MTU Met    RX-OK  RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR  Flg
eth0      1500   0 2296581 2094489 791488   8651   148350      0       0       1   BMRU
lo       16436   0 1169442       0       0      0 1169442      0       0       0   LRU
```

欄位	說明
Iface	網路界面卡的名稱
MTU	界面的最大傳輸單位或封包大小
RX-OK	從界面啟動開始，已經進入的封包數量
RX-ERR	進入封包錯誤的數量
RX-OVR	進入封包超過輸入緩衝的數量
RX-DRP	進入封包遺失的數量
TX-OVR	輸出封包超過輸出緩衝的數量
TX-OK	從界面啟動開始，已經輸出的封包數量
TX-ERR	從界面最後啟動開始，輸出封包錯誤的數量
TX-DRP	輸出封包遺失的數量

我們可以使用 netstat 指令得到網路的情況。Netstat 指令顯示所有網路上的活動。我們可以使用-t 和-u 選項來減輕 Unix sockets。我們可以使用-c 選項來得到持續的更新。在下列欄位中，State 是狀態，ESTABLISHED 是已經建立，從外部的 61.218.29.5 連接到本地端的 flash.aasir.com。

```
[root@flash chaiyen]# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      2 flash.aasir.com:telnet  61-218-29-5.HINET-:1189 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  14     [ ]         DGRAM                    1590   /dev/log
unix  2      [ ]         DGRAM                    95375
unix  2      [ ]         DGRAM                    79340
unix  2      [ ]         DGRAM                    77417
unix  3      [ ]         STREAM     CONNECTED     3038   /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     3037
unix  3      [ ]         STREAM     CONNECTED     3034   /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     3033
unix  3      [ ]         STREAM     CONNECTED     3018   /tmp/.font-unix/fs7100
unix  3      [ ]         STREAM     CONNECTED     3017
unix  3      [ ]         STREAM     CONNECTED     3020   /tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     3012
unix  2      [ ]         DGRAM                    2718
unix  2      [ ]         DGRAM                    2616
unix  2      [ ]         DGRAM                    2309
```

我們可以使用 netstat –nr 查出我們網路的情況。

```
[root@flash chaiyen]# netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
61.218.29.0     0.0.0.0         255.255.255.248 U         0 0          0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U         0 0          0 lo
0.0.0.0         61.218.29.1     0.0.0.0         UG        0 0          0 eth0
```

我們可以使用 ping 指令來查看遠端到本地端網路相通的情況。

```
[root@flash chaiyen]# ping tw.yahoo.com
PING tw.yahoo.com (202.1.237.21) 56(84) bytes of data.
64 bytes from tw.yahoo.com (202.1.237.21): icmp_seq=1 ttl=249 time=43.2 ms
64 bytes from tw.yahoo.com (202.1.237.21): icmp_seq=2 ttl=249 time=43.8 ms

--- tw.yahoo.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 43.258/43.575/43.893/0.379 ms
```