Sintonització, optimització i alta disponibilitat

Remo Suppi Boldrito

PID_00174424

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

Un aspecte fonamental, un cop el sistema està instal·lat, és la configuració i l'adaptació del sistema a las necessitats de l'usuari i que les prestacions del sistema siguin tan adequades com sigui possible a les necessitats que se li demanen. GNU/Linux és un sistema operatiu eficient que permet un grau de configuració excel·lent i una optimització molt delicada d'acord amb les necessitats de l'usuari. És per això que, un cop feta una instal·lació (o en alguns casos una actualització), cal que es facin determinades configuracions vitals en el sistema. Tot i que el sistema "funciona", cal fer alguns canvis (adaptació a l'entorn o sintonització) per a permetre que estiguin cobertes totes les necessitats de l'usuari i dels serveis que presta la màquina. Aquesta sintonització dependrà d'on es trobi funcionant la màquina i en alguns casos es farà per a millorar el rendiment del sistema, mentre que en d'altres (a més a més), per qüestions de seguretat. Quan el sistema està en funcionament, és necessari monitorar-lo per a veure'n el comportament i actuar en conseqüència. Si bé és un aspecte fonamental, la sintonització d'un sistema operatiu molts cops es relega a la opinió d'experts o gurus de la informàtica; però coneixent els paràmetres que afecten el rendiment, és possible arribar a bones solucions fent un procés cíclic d'anàlisi, canvi de configuració, monitoratges i ajustos. En aquest mòdul es veuran les principals eines per a monitorar un sistema GNU/Linux com ara Munin i Monit MRTG, i es donaran indicacions de com s'ha de sintonitzar el sistema a partir de la informació obtinguda.

La seguretat s'estudia al mòdul "Administració de seguretat".

Un altre aspecte important en l'actualitat dels servidors de sistemes de la informació és l'alta disponibilitat.

L'alta disponibilitat (*high availability*) és un protocol de disseny del sistema i la seva implementació associada que assegura un cert grau absolut de continuïtat operacional durant períodes llargs de temps. El terme *disponibilitat* es refereix a l'habilitat de la comunitat d'usuaris per a accedir al sistema, enviar nous treballs, actualitzar o alterar treballs existents o recollir els resultats de treballs previs. Si un usuari no pot accedir al sistema es diu que està *no disponible*.

D'entre totes les eines que hi ha per a tractar aquests aspectes (Heartbeat, ldirectord i LVS –Linux Virtual Server–, Piranha, UltraMonkey, Kimberlite, etc.), en aquest mòdul n'analitzarem algunes que permeten gestionar aquesta infraestructura redundant, com Heartbeat o Kimberlite.

Objectius

En els materials didàctics d'aquest mòdul trobareu els continguts i les eines procedimentals per a aconseguir els objectius següents:

1. Analitzar i determinar les possibles pèrdues de prestacions d'un sistema.
2. Solucionar problemes de sintonització del sistema.
3. Instal·lar i analitzar les diferents eines i la seva integració per a resoldre els problemes d'eficiències.
4. Analitzar les eines que permeten tenir un sistema en alta disponibilitat.

1. Sintonització, optimització i alta disponibilitat

1.1. Aspectes bàsics

Abans de conèixer quines són les tècniques d'optimització, cal enumerar les causes que poden afectar les prestacions d'un sistema operatiu [9]. Entre aquestes, es poden mencionar:

1) Colls d'ampolla en els recursos: la conseqüència és que tot el sistema anirà més lent perquè hi ha recursos que no poden satisfer la demanda a la qual se'ls sotmet. El primer pas per a optimitzar el sistema és trobar aquests colls d'ampolla i determinar per què es produeixen, coneixent-ne les limitacions teòriques i les pràctiques.

2) Llei d'Amdahl: segons aquesta llei, "hi ha un límit de quant es pot millorar en velocitat una cosa si només se n'optimitza una part"; és a dir, si es té un programa que utilitza el 10% de la CPU i s'optimitza reduint la utilització en un factor 2, el programa en millorarà les prestacions (*speedup*) en un 5%, la qual cosa pot significar un tremend esforç no compensat pels resultats.

3) Estimació del *speedup*: és necessari estimar en quina mesura el sistema millorarà les prestacions per a evitar esforços i costos innecessaris. Es pot utilitzar la llei d'Amdahl per a valorar si cal una inversió, econòmica o en temps, en el sistema.

4) Efecte bombolla: sempre es té la sensació que quan es troba la solució a un problema, en sorgeix un altre. Una manifestació d'aquest problema és que el sistema es mou constantment entre problemes de CPU i problemes d'entrada/sortida, i viceversa.

5) Temps de resposta davant la quantitat de treball: si es tenen vint usuaris, millorar en la productivitat significarà que tots tindran més feina feta alhora, però no millors respostes individuals; podria ser que el temps de resposta per a alguns fos millor que per a d'altres. Millorar el temps de resposta significa optimitzar el sistema per tal que les tasques individuals triguin el menys possible.

6) Psicologia de l'usuari: dos paràmetres són fonamentals:

a) l'usuari generalment estarà insatisfet quan es produeixin variacions en el temps de resposta; i

b) l'usuari no detectarà millores en el temps d'execució menors del 20%.

7) Efecte prova: les mesures de monitoratge afecten les pròpies mesures. Cal anar amb compte quan es fan les proves pels efectes col·laterals dels propis programes de mesura.

8) Importància de la mitjana i la variació: cal tenir con compte els resultats, atès que si s'obté una mitjana d'utilització de CPU del 50% quan ha estat utilitzada 100, 0, 0, 100, es podrien extreure conclusions errònies. És important veure la variació sobre la mitjana.

9) Coneixements bàsics sobre el maquinari del sistema que s'ha d'optimitzar: per tal de millorar una cosa cal "conèixer" si és susceptible de millora. L'encarregat de l'optimització haurà de conèixer bàsicament el maquinari subjacent (CPU, memòries, busos, memòria cau, entrada/sortida, discos, vídeo, etc.) i la seva interconnexió per a poder determinar on són els problemes.

10) Coneixements bàsics sobre el sistema operatiu que s'ha d'optimitzar: de la mateixa manera que en el punt anterior, l'usuari haurà de conèixer aspectes mínims sobre el sistema operatiu que pretén optimitzar, entre els quals s'inclouen conceptes com processos i fils o *threads* (creació, execució, estats, prioritats i terminació), crides al sistema, *buffers* de memòria cau, sistema d'arxius, administració de memòria i memòria virtual (paginació, sistema d'intercanvi –*swap*–) i taules del nucli (*kernel*).

1.1.1. Monitoratge sobre UNIX System V

El `/proc` el veurem com un directori, però en realitat és un sistema d'arxius fictici, és a dir, no existeix sobre el disc i el nucli el crea en memòria. S'utilitza per a proveir d'informació sobre el sistema (originalment sobre processos, d'aquí el nom), informació que posteriorment serà utilitzada per totes les ordres que veurem a continuació. Una vista d'aquest directori és:

```
1     ...   asound       ioports       sched_debug
124   ...   buddyinfo    irq           scsi
125   ...   bus          kallsyms      self
1258  ...   cgroups      kcore         slabinfo
126   ...   cmdline      key-users     stat
127   ...   cpuinfo      kmsg          swaps
1601  ...   crypto       kpagecount    sys
1612  ...   devices      kpageflags    sysrq-trigger
1851  ...   diskstats    loadavg       sysvipc
1886  ...   dma          locks         timer_list
1930  ...   driver       meminfo       timer_stats
1941  ...   execdomains  misc          tty
1951  ...   fb           modules       uptime
1963  ...   filesystems  mounts        version
1964  ...   fs           mtrr          vmallocinfo
2     ...   ide          net           vmstat
2012  ...   interrupts   pagetypeinfo  zoneinfo
2049  ...   acpi         iomem         partitions
```

Hi ha un conjunt d'arxius i directoris. A continuació, en veurem alguns dels més interessants*:

***Consulteu la pàgina del manual per a obtenir més informació.**

`/proc/1`: un directori amb la informació del procés 1 (el número del directori és el PID del procés).
`/proc/cpuinfo`: informació sobre la CPU (tipus, marca, model, prestacions, etc.).
`/proc/devices`: llista de dispositius configurats en el nucli.
`/proc/dma`: canals de DMA utilitzats en aquell moment.
`/proc/filesystems`: sistemes d'arxius configurats en el nucli.
`/proc/interrupts`: mostra quines interrupcions estan en ús i quantes d'elles s'han processat.
`/proc/ioports`: ídem amb els ports.
`/proc/kcore`: imatge de la memòria física del sistema.
`/proc/kmsg`: missatges generats pel nucli, que posteriorment són enviats a syslog.
`/proc/ksyms`: taula de símbols del nucli.
`/proc/loadavg`: càrrega del sistema.
`/proc/meminfo`: informació sobre la utilització de memòria.
`/proc/modules`: mòduls carregats pel nucli.
`/proc/net`: informació sobre els protocols de xarxa.
`/proc/stat`: estadístiques sobre el sistema.
`/proc/uptime`: des de quan el sistema està funcionant.
`/proc/version`: versió del nucli.

Aquests arxius es construeixen de forma dinàmica cada cop que es visualitza el contingut i el nucli del sistema operatiu els proveeix en temps real. És per això que es denomina *sistema d'arxius virtual* i el contingut dels arxius i directoris va canviant en forma dinàmica amb les dades actualitzades. D'aquesta manera es pot considerar el `/proc/` com una interfície entre el nucli de Linux i l'usuari i és una forma sense ambigüitats i homogènia de presentar informació interna que pot ser utilitzada per les diverses eines/ordres d'informació/sintonització/control que utilitzarem regularment. És interessant, per exemple, veure la sortida d'ordre `mount` i el resultat de l'execució `more /proc/mounts`: és totalment equivalent!

En els subapartats següents s'ensenyarà com obtenir i modificar la informació del nucli de Linux treballant amb el sistema d'arxius `/proc`.

Cal tenir en compte que aquests arxius són visibles (text), però de vegades les dades estan "en cru" i es necessiten ordres per a interpretar-los, que seran les que veurem a continuació. Els sistemes compatibles UNIX SV utilitzen les ordres `sar` i `sadc` per a obtenir estadístiques del sistema. A Debian és `atsar` (i `atsadc`), que és totalment equivalent als que hem mencionat i posseeix un conjunt de paràmetres que ens permeten obtenir informació de tots els comptadors i informació sense processar del `/proc`. Debian també inclou el paquet **sysstat** que conté les ordres `sar` (informació general de l'activitat del sistema), `iostat` (utilització CPU i d'E/S), `mpstat` (informes globals per processador), `pidstat` (estadístiques de processos), `sadf` (mostra informació del `sar` en diversos formats). L'ordre `atsar` llegeix comptadors i estadístiques del fitxer `/proc` i les mostra per la sortida estàndard. La primera manera de cridar a l'ordre és (executar-la com a arrel o agregar l'usuari a la categoria corresponent del `sudoers` per a executar amb el `sudo`):

```
atsar opcions t [n]n
```

on mostra l'activitat en `n` vegades cada `t` segons amb un encapçalament que mostra els comptadors d'activitat (el valor per defecte de $n = 1$). La segona manera de cridar-lo és:

```
atsar -opciones -s time -e time -i sec -f file -n day#
```

L'ordre extreu dades de l'arxiu especificat per `-f` (que per defecte és el fitxer `/var/log/atsar/atsarxx`, sent `xx` el dia del mes) i que van ser desats prèviament per `atsadc` (s'utilitza per a recollir les dades, desar-les i processar-les i en Debian és a `/usr/lib/atsar`). El paràmetre `-n` pot utilitzar-se per a indicar el dia del mes i `-s`, `-e` l'hora d'inici i final, respectivament. Per a activar `atsadc`, per exemple, es podria incloure a `/etc/cron.d/atsar` una línia com la següent:

```
@reboot root test -x /usr/lib/atsadc && /usr/lib/atsar/atsadc /var/log/atsar/atsa'date +%d'
10,20,30,40,50 * * * * root test -x /usr/lib/atsar/atsa1 && /usr/lib/atsar/atsa1
```

La primera línia crea l'arxiu després d'un reinici i la segona, desa les dades cada 10 minuts amb l'*script* de l'intèrpret d'ordres `atsa1`, que crida al `atsadc`. A `atsar` (o `sar`), les opcions s'utilitzen per a indicar quins comptadors cal mostrar i alguns són:

Opcions	Descripció
u	Utilització de CPU
d	Activitat de disc
I (i)	Nombre d'interrupcions/s
v	Utilització de taules en el nucli
i	Estadístiques d'utilització de ttys
p	Informació de paginació i activitat del sistema d'intercanvi
r	Memòria lliure i ocupació de memòria d'intercanvi
l (L)	Estadístiques de xarxa
L	Informació d'errors de xarxa
w	Estadístiques de connexions IP
t	Estadístiques de TCP
U	Estadístiques d'UDP
m	Estadístiques d'ICMP
N	estadístiques de NFS
A	Totes les opcions

Entre `atsar` i `sar` només presenten algunes diferències quant a la manera de mostrar les dades i `sar` inclou unes quantes opcions més (o diferents). A continuació es veuran alguns exemples d'utilització de `sar` (exactament igual que amb `atsar`, solament hi pot haver alguna diferència en la visualització de les dades) i el significat de la informació que genera:

Utilització de CPU: `sar -u 4 5`

```
Linux  debian  2.6.26-2-686  #1 SMP Thu May 28 15:39:35 UTC 2009  i686  11/30/2010
05:32:54  cpu %usr   %nice   %sys %irq %softirq    %wait %idle            _cpu_
05:33:05  all    3     0      8    0        0       88     0
05:33:09  all    4     0     12    0        0       84     0
05:33:14  all   15     0     19    1        0       65     0
...
05:41:09  all    0     0      1    0        0        0    99
```

`%usr` i `%sys` mostren el percentatge de temps de CPU en el mode usuari amb `nice=0` (normals) i en el mode nucli. `idle` indica el temps no utilitzat de CPU

pels processos en estat d'espera (no inclou espera de disc). `wait` és el temps que la CPU ha estat lliure quan el sistema estava fent entrada o sortida (per exemple, de disc). `irq` i `sofirq` és el temps que la CPU ha dedicat a gestionar les interrupcions, que és un mecanisme de sincronització entre el que fa la CPU i els dispositius d'entrada i de sortida. En el cas `idle=99%` significa que la CPU està ociosa, per la qual cosa no hi ha processos per executar i la càrrega és baixa; si `idle` $\simeq$ i el nombre de processos és elevat, caldria pensar a optimitzar la CPU, atès que podria ser el coll d'ampolla del sistema. En l'exemple podem veure que s'utilitza poc CPU i molt entrada i sortida, per la qual cosa es pot verificar que en aquest cas la càrrega del sistema l'està generant el disc (per a l'exemple s'havien obert 5 còpies del programa OpenOffice Writer).

Nombre d'interrupcions per segon: `sar -I 4 5`

```
Linux  debian  2.6.26-2-686  #1 SMP Thu May 28 15:39:35 UTC 2009  i686  11/30/2010
05:46:30  cpu  iq00 iq01 iq05 iq08 iq09 iq10 iq11 iq12 iq14 iq15       _intr/s_
05:46:34  all     0    0    0    0   33    0    0  134    4    5
05:46:37  all     0    0    0    0   54    1    0  227   38   13
05:46:42  all     0    0    0    0   41    0    0  167   10    8
```

Mostra la informació de la freqüència d'interrupcions dels nivells actius que es troben a `/proc/interrupts`. Ens és útil per a veure si existeix algun dispositiu que està interrompent constantment la feina de la CPU. Consultant aquest arxiu veurem que en l'exemple les més actives són la 9 (acpi), 12 (teclat), 14-15 (ide) i molt poc la 10 (usb).

Memòria i memòria d'intercanvi: `sar -r 4 5`

```
Linux  debian  2.6.26-2-686  #1 SMP Thu May 28 15:39:35 UTC 2009  i686  11/30/2010
05:57:10  memtot memfree buffers   cached  slabmem      swptot swpfree  _mem_
05:57:17   1011M    317M    121M     350M      30M        729M    729M
05:57:21   1011M    306M    121M     351M      30M        729M    729M
05:57:25   1011M    300M    121M     351M      30M        729M    729M
```

En aquest cas `memtot` indica la memòria total lliure i `memfree`, la memòria lliure. La resta d'indicadors és la memòria utilitzada com a intermèdia, la utilitzada en memòria cau (de dades), `slabmem` és la memòria dinàmica del nucli i `swptot/free` és l'espai total/lliure de memòria d'intercanvi. És important tenir en compte que si `memfree` $\simeq$ 0 (no hi ha espai), les pàgines dels processos aniran a parar a la memòria d'intercanvi, on deu haver-hi lloc tenint en compte que això permetrà l'execució, però tot anirà més lent. Això s'ha de contrastar amb l'ús de CPU. També cal controlar que la mida de les memòries intermèdies sigui adequada i estigui en relació amb els processos que estan fent operacions d'entrada i de sortida. També és interessant l'ordre `free`, que permet veure la quantitat de memòria en una visió simplificada:

```
             total       used       free     shared    buffers     cached
Mem:       1036092     711940     324152          0     124256     359748
-/+ buffers/cache:     227936     808156
Swap:       746980          0     746980
```

Això indica que d'1 Gb gairebé les 3/4 parts de la memòria estan ocupades i que aproximadament 1/3 són de memòria cau. A més a més, ens indica que la memòria d'intercanvi no s'està utilitzant per a res, per la qual cosa podem concloure que el sistema està bé. Si volguéssim més detalls hauríem d'utilitzar l'ordre `vmstat` (amb més detalls que el `sar -r`) per tal d'analitzar què és el que està causant problemes o qui està consumint tanta memòria. A continuació, es mostra una sortida de `vmstat 1 10`*:

***Consulteu el manual per obtenir una descripció de les columnes.**

```
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 1  1      0 324820 124256 359796    0    0    23    11   20  112  0  0 99  1
 0  0      0 324696 124256 359816    0    0     0    88    4   96  1  1 98  0
 0  0      0 324716 124256 359816    0    0     0     0  106  304  0  0 100  0
 0  0      0 324716 124256 359816    0    0     0     0  150  475  1  2 97  0
...
```

Utilització de les taules del nucli: `sar -v 4 5`

```
Linux  debian  2.6.26-2-686  #1 SMP Thu May 28 15:39:35 UTC 2009  i686  11/30/2010
06:14:02  superb-sz inode-sz     file-sz    dquota-sz    flock-sz      _curmax_
06:14:06     0/0     32968/36     3616/101976      0/0        13/0
06:14:10     0/0     32968/36     3616/101976      0/0        13/0
06:14:13     0/0     32984/36     3616/101976      0/0        13/0
06:14:17     0/0     32984/36     3616/101976      0/0        13/0
06:14:22     0/0     33057/36     3680/101976      0/0        13/0
```

En aquest cas, `superb-sz` és el nombre actual-màxim de *superblocks* mantingut pel nucli per als sistemes d'arxius muntats; `inode-sz` és el nombre actual-màxim d'*incore-inodes* en el nucli necessari, que és d'un per disc com a mínim; `file-sz` és el nombre actual-màxim d'arxius oberts, `dquota-sz` és l'ocupació actual-màxima d'entrades de quotes (per a més informació, consulteu `man sar` –o `atsar`–). Aquest monitoratge es pot completar amb l'ordre `ps -Af` (*process status*) i l'ordre `top`, que mostraran l'activitat i l'estat dels processos en el sistema. A continuació, es mostren dos exemples d'ambdues ordres (només algunes línies):

```
debian:/proc# ps -Alw
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 -   525 -      ?        00:00:01 init
5 S     0     2     0  0  75  -5 -     0 -      ?        00:00:00 kthreadd
1 S     0     3     2  0 -40   - -     0 -      ?        00:00:00 migration/0
...
5 S     1  1601     1  0  80   0 -   473 -      ?        00:00:00 portmap
5 S   102  1612     1  0  80   0 -   489 -      ?        00:00:00 rpc.statd
...
4 S   113  2049  2012  0  80   0 - 31939 -      ?        00:00:03 mysqld
...
4 S     0  2654  2650  0  80   0 -  6134 -      tty7     00:00:49 Xorg
1 S     0  2726     1  0  80   0 -  6369 -      ?        00:00:00 apache2
0 S     0  2746     1  0  80   0 -   441 -      tty1     00:00:00 getty
...
```

Alguns aspectes interessants de veure són la dependència dels processos (PPID = procés pare) i, per exemple, que per a saber l'estat dels processos es pot executar amb `ps -Alw` i en la segona columna ens mostrarà com es troba cada

un dels processos. Aquests paràmetres reflecteixen el valor indicat en la variable del nucli per a aquest procés, els més importants dels quals, des del punt de vista del monitoratge, són: `F` *flags* (en aquest cas 1 és amb superprivilegis, 4 creat des de l'inici dimoni), `S` és l'estat (`D`: no interrompible dormint entrada/sortida, `R`: executable o en cua, `S`: dormint, `T`: en traça o aturat, `Z`: mort en vida, 'zombie'). `PRI` és la prioritat; `NI` és *nice*; `TTY`, des d'on s'ha executat; `TIME`, el temps de CPU; `CMD`, el programa que s'ha executat i els seus paràmetres. Si es vol sortida amb refresc (configurable), es pot utilitzar l'ordre `top`, que mostra unes estadístiques generals (processos, estats, càrrega, etc.) i, després, informació de cada un d'ells similar al `ps`, però s'actualitza cada 5 segons per defecte:

```
top - 06:40:37 up  5:45,  3 users,  load average: 0.00, 0.00, 0.00
Tasks: 118 total,   2 running, 116 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.3%us,  4.1%sy,  0.0%ni, 93.2%id,  0.4%wa,  0.1%hi,  0.0%si,  0.0%st
Mem:   1036092k total,   722484k used,   313608k free,   124396k buffers
Swap:   746980k total,        0k used,   746980k free,   367408k cached
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2654 root      20   0 41368  16m 8928 S  3.5  1.7   0:51.86 Xorg
 3093 remo      20   0 38764  15m 9532 R  1.8  1.6   0:22.24 gnome-terminal
 2987 remo      20   0 36924  19m  11m S  1.1  1.9   0:05.14 gnome-panel
 2984 remo      20   0 20604  11m 7252 S  0.7  1.2   0:03.86 metacity
 2980 remo      20   0 15464 5344 4336 S  0.6  0.5   0:05.34 gnome-screensav
 2988 remo      20   0 79820  28m  13m S  0.5  2.8   0:07.12 nautilus
 5263 remo      20   0  6036 3204 1312 S  0.4  0.3   0:00.08 bash
 2547 root      20   0  2052  868  736 S  0.1  0.1   0:01.16 dhcdbd
 2592 root      20   0  3388 1032  900 S  0.1  0.1   0:09.36 hald-addon-stor
```

També es poden utilitzar les eines del paquet `systat` per a conèixer l'estat dels recursos, com per exemple `vmstat` (estadístiques de CPU, memòria i entrada/sortida), `iostat` (estadístiques de discos i CPU) i `uptime` (càrrega de CPU i estat general).

1.1.2. Optimització del sistema

A continuació veurem algunes recomanacions per a optimitzar el sistema en funció de les dades obtingudes.

1) Resoldre els problemes de memòria principal: s'ha de procurar que la memòria principal pugui acollir un percentatge elevat de processos en execució, ja que si no és així, el sistema operatiu podrà paginar i anar a la memòria d'intercanvi; però això significa que l'execució d'aquell procés es degradarà notablement. Si s'agrega memòria, el temps de resposta millorarà notablement. Per a fer-ho, cal tenir en compte la mida dels processos (`SIZE`) en estat `R` i agregar-hi la que utilitza el nucli. Les quantitats de memòria es poden obtenir amb l'ordre `free`, que ens mostrarà (o amb `dmesg`), per exemple (`total/used/free/buffers/cached`):

```
1036092/723324/312768/124396/367472,
```

que és equivalent a (en megabytes) `1011/706/305/121/358` i on observem, en aquest cas, que només el 30% de la memòria està lliure i que en aquest mo-

ment no hi ha problemes, però una càrrega mínima del sistema pot significar un problema. És per això que haurem d'analitzar si el sistema està limitat per la memòria (amb `atsar -r` i `-p` es veurà molta activitat de paginació).

Les solucions per a la memòria són òbvies: o s'incrementa la capacitat o es redueixen les necessitats. Pel cost actual de la memòria, és més adequat incrementar-ne la mida que emprar moltes hores per a guanyar un centenar de bytes en treure, ordenar o reduir requeriments dels processos en la seva execució. Es poden reduir els requisits reduint les taules del nucli, traient mòduls, limitant el nombre màxim d'usuaris, reduint la memòria intermèdia, etc.; tot això degradarà el sistema (efecte bombolla) i les prestacions seran pitjors (en alguns casos, el sistema pot quedar totalment no operatiu).

Un altre aspecte que es pot reduir és la quantitat de memòria dels usuaris gràcies a l'eliminació de processos redundants i canviant la càrrega de treball. Per a fer-ho, caldrà monitorar els processos que estan dormint (zombies) i eliminar-los, o bé aquells que no progressin en la seva entrada/sortida (saber si són processos actius, quanta CPU han gastat i si els "usuaris els estan esperant per ells"). Canviar la càrrega de treball és utilitzar planificació de cues per tal que els processos que necessiten gran quantitat de memòria es puguin executar en hores de poca activitat (per exemple, a la nit, llençant-los amb l'ordre `at`).

2) Molta utilització de CPU: bàsicament ens la dóna el temps *idle* (valors baixos). Amb `ps` o `top` s'ha d'analitzar quins processos són els que "devoren CPU" i prendre decisions, com ara postposar-ne l'execució, aturar-los temporalment, canviar-ne la prioritat (és la solució menys conflictiva de totes i per això es pot utilitzar l'ordre `renice prioritat PID`), optimitzar el programa (per a la propera vegada) o canviar la CPU (o agregar-ne una altra). Com ja s'ha esmentat, GNU/Linux utilitza el directori `/proc` per a mantenir totes les variables de configuració del nucli que poden ser analitzades i, en determinades ocasions, "ajustades", per a aconseguir prestacions diferents o millors.

Per a fer-ho, cal utilitzar l'ordre `sysctl -a` per a obtenir totes les variables del nucli i els seus valors en l'arxiu*. Altres ordres alternatives són la `systune` i `systunedump`, que permeten descarregar les variables en un arxiu i modificar-les, per a carregar-les novament en el `/proc` (l'ordre `systune` desa la configuració a `/etc/systune.conf`). En aquest cas, per exemple, es podrien modificar (cal procedir amb compte, perquè el nucli pot quedar fora de servei) les variables de la categoria `/proc/sys/vm` (memòria virtual) o `/proc/sys/kernel` (configuració del l'ànima del nucli).

***Consulteu el manual per canviar els valors i l'arxiu de configuració** `/etc/sysctl.conf`

En aquest mateix sentit, també (per a experts o desesperats) es pot canviar el temps màxim (*slice*) que l'administrador de CPU (*scheduler*) del sistema operatiu dedica a cada procés en forma circular (si bé és aconsellable utilitzar `renice` com a pràctica). Però a GNU/Linux, a diferència d'altres sistemes operatius, és un valor fix dins del codi, ja que està optimitzat per a diferents funcionalitats (però és possible tocar-lo). Es pot "jugar" (a risc seu) amb un

conjunt de variables que permeten tocar el temps màxim d'assignació de CPU (`kernel-source-2.x.x/kernel/sched.c`).

3) Reduir el nombre de crides: una altra pràctica adequada per a millorar les prestacions és reduir el nombre de crides al sistema de més cost en temps de CPU. Aquestes crides són les invocades (generalment) pel `shell fork()` i `exec()`. Una configuració inadequada de la variable `PATH`, i a causa del fet que la crida `exec()` no desa res en la memòria cau, el directori actual (indicat per `./`) pot tenir una relació desfavorable d'execució. Per això, sempre caldrà configurar la variable `PATH` amb el directori actual com a última ruta. Per exemple, a `$HOME/.bashrc` feu: `PATH=$PATH:.; export PATH` si el directori actual no està en el camí o, si està, refeu la variable `PATH` per a posar-lo com a última ruta.

Cal tenir en compte que una alta activitat d'interrupcions pot afectar les prestacions de la CPU en relació amb els processos que executa. Mitjançant monitoratge (`atsar -I`) es pot mirar quina és la relació d'interrupcions per segon i prendre decisions pel que fa als dispositius que les causen. Per exemple, canviar el mòdem per un altre de més intel·ligent o canviar l'estructura de comunicacions si detectem una activitat elevada sobre el port en sèrie on es troba connectat.

4) Molta utilització de disc: després de la memòria, un temps de resposta baix pot ser a causa del sistema de discos. En primer lloc, cal verificar que es disposi de temps de CPU (per exemple, `idle` > 20%) i que el nombre d'entrades/sortides sigui elevat (per exemple, superior a 30 entrades/sortides) utilitzant `atsar -u` i `atsar -d`. Les solucions passen per:

- En un sistema de més d'un disc, planificar on es trobaran els arxius més utilitzats per a equilibrar el trànsit cap a ells (per exemple `/home` en un disc i `/usr` en un altre) i que puguin utilitzar totes les capacitats d'entrada/sortida amb memòria cau i concurrent de GNU/Linux (fins i tot, per exemple, planificar sobre quin *bus ide* es col·loquen). Comprovar després que el trànsit està equilibrat amb `atsar -d` (o amb `iostat`). En situacions crítiques, es pot considerar l'opció de comprar un sistema de discos RAID que facin aquest ajust de manera automàtica.

- Tenir en compte que s'obtenen millors prestacions sobre dos discos petits que sobre un de gran de la mida dels dos anteriors.

- En sistemes amb un sol disc, generalment es creen, des del punt de vista de l'espai, quatre particions de la manera següent (des de fora cap a dins): `/, swap, /usr, /home`, però això genera respostes pèssimes d'entrada/sortida perquè si, per exemple, un usuari compila des del seu directori `/home/user` i el compilador es troba a `/usr/bin`, el capçal del disc es mourà longitudinalment. En aquest cas, és millor unir les particions `/usr` i `/home` en una de sola (més gran), tot i que pot representar alguns inconvenients quant al manteniment.

- Incrementar les memòries intermèdies de cau d'entrada/sortida (consulteu, per exemple, `/proc/ide/hd`...).

- Si s'utilitza un `extfs`, es pot utilitzar l'ordre `dumpe2fs -h /dev/hdx` per a obtenir informació sobre el disc i `tune2fs /dev/hdx` per a canviar alguns dels paràmetres configurables del disc.

- Òbviament, canviar el disc per un altre de més velocitat (més RPM) sempre tindrà un impacte positiu en un sistema limitat per l'entrada/sortida de disc [9].

5) Millorar aspectes de TCP/IP: examinar la xarxa amb l'ordre `atsar` (o també amb `netstat -i` o amb `netstat -s | more`) per a analitzar si hi ha paquets fragmentats, errors, caigudes, desbordaments, etc. que puguin estar afectant les comunicacions i, amb això, el sistema (per exemple, en un servidor d'NFS, NIS, ftp o web). Si s'hi detecten problemes, caldrà analitzar la xarxa per a considerar les actuacions següents:

- Fragmentar la xarxa mitjançant elements actius que descartin paquets amb problemes o que no siguin per a màquines del segment.

- Planificar on estaran els servidors per a reduir el trànsit cap a ells i els temps d'accés.

- Ajustar paràmetres del nucli (`/proc/sys/net/`). Per exemple, per obtenir millores en la velocitat de transferència de dades haurem d'executar la instrucció `echo 600 > /proc/sys/net/core/netdev_max_backlog`*.

*Mínim 300.

6) Altres accions sobre paràmetres del nucli: existeix un altre conjunt de paràmetres sobre el nucli que és possible sintonitzar per a obtenir millors prestacions, si bé, tenint en compte el que hem tractat anteriorment, cal anar amb compte, ja que podríem causar l'efecte contrari o inutilitzar el sistema. Consulteu en el directori `kernel-source-2.x/Documentation/sysctl` de la distribució del codi font alguns arxius com per exemple `vm.txt`, `fs.txt` i `kernel.txt`. `/proc/sys/vm` controla la memòria virtual del sistema d'intercanvi) i permet que els processos que no entren en la memòria principal siguin acceptats pel sistema però en el dispositiu d'intercanvi, amb la qual cosa el programador no té límit per a la mida del seu programa (òbviament ha de ser més petit que el dispositiu d'intercanvi). Els paràmetres susceptibles de ser sintonitzats es poden canviar molt fàcilment amb `sysctl` (o també amb `gpowertweak`). `/proc/sys/fs` conté paràmetres que poden ser ajustats de la interacció nucli-sistema de fitxers, com `file-max` (i exactament igual per a la resta dels arxius d'aquest directori).

7) Generar el nucli adequat a les nostres necessitats: l'optimització del nucli significa escollir els paràmetres de compilació d'acord amb les nostres necessitats. És molt important llegir primer l'arxiu `readme` dins del directori `/usr/src/linux`.

Una bona configuració del nucli permetrà que s'executi més ràpid, que es disposi de més memòria per als processos d'usuari i, a més a més, resultarà més estable. Hi ha dues maneres de construir un nucli: **monolític** (millors prestacions) o **modular** (basat en mòduls, que tindrà millor portabilitat si tenim un sistema molt heterogeni i no es vol compilar un nucli per a cada un d'ells). Per compilar el seu propi nucli i adaptar-lo al seu maquinari i a les seves necessitats, cada distribució té les seves regles (si bé el procediment és similar).

Enllaços d'interès

És interessant consultar els articles següents: http://people.redhat.com/alikins/system_tuning.html sobre informació d'optimització de sistemes servidors Linux i http://www.linuxjournal.com/article/2396, *Performance Monitoring Tools for Linux*. El primer ja té més de quatre anys, però la majoria dels procediments segueixen essent vigents i el segon és un article antic i algunes opcions no estan disponibles, però la metodologia segueix essent la mateixa.

1.1.3. Optimitzacions de caràcter general

Hi ha una sèrie d'optimitzacions de caire general que poden millorar les prestacions del sistema:

1) Biblioteques estàtiques o dinàmiques: quan es compila un programa, es pot fer amb una biblioteca estàtica (`libr.a`), el codi de funció de la qual s'inclou en l'executable, o amb una de dinàmica (`libr.so.xx.x`), on es carrega la biblioteca en el moment de l'execució. Si bé les primeres garanteixen codi portable i segur, consumeixen més memòria. El programador haurà de decidir quina és l'adequada per al seu programa incloent `-static` en les opcions del compilador (no posar-lo significa dinàmiques) o `-disable-shared`, quan s'utilitza l'ordre `configure`. És recomanable utilitzar (gairebé totes les distribucions noves ho fan) la biblioteca estàndard `libc.a` i `libc.so` de versions 2.2.x o superiors (coneguda com a Libc6) que reemplaça les anteriors. En gcc 4.X s'utilitzen per defecte biblioteques dinàmiques, però es pot forçar (no recomanat) a utilitzar estàtiques fins i tot per a la `libc` (opcions `-static -static-libgcc` en contraposició amb les que s'usen per defecte `-shared -shared-libgcc`).

2) Selecció del processador adequat: generar codi executable per a l'arquitectura sobre la qual correran les aplicacions. Alguns dels paràmetres més influents del compilador són:

a) `-march` (per exemple, `-march=core2` per al suport de CPU Intel Core2 CPU 64-bit amb extensions MMX, SSE, SSE2, SSE3/SSSE3, o `-march=k8` per a CPU AMD K8 Core amb suport x86-64) fent `gcc -march=i686`;

b) l'atribut d'optimització `-O1,2,3` (`-O3` generarà la versió més ràpida del programa, `gcc -O3 -march = i686`), i

c) els atributs `-f` (consulteu la documentació per als diferents tipus).

Paquets `no-free`

Recordeu que sobre Debian cal activar el dipòsit de paquets `no-free` per a poder instal·lar els paquets de documentació del compilador `gcc-doc` i `gcc-doc-base`.

3) Optimització del disc: en l'actualitat, la majoria d'ordinadors inclou disc UltraDMA (100) per defecte; tanmateix, en una gran quantitat de casos no estan optimitzats per a extraure'n les millors prestacions. Hi ha una eina (hdparm) que permet sintonitzar el nucli als paràmetres del disc tipus IDE i SATA

(tot i que aquests últims també disposen d'una utilitat específica denominada sdparm). Cal anar amb compte amb aquestes utilitats, sobretot en discos UltraDMA (cal verificar en el BIOS que els paràmetres per a suport per DMA estan habilitats), ja que poden inutilitzar el disc. Consulteu les referències i la documentació ([14] i man hdparm/sdparm) sobre quines són (i el risc que comporten) les optimitzacions més importants, per exemple: `-c3`, `-d1`, `-X34`, `-X66`, `-X12`, `-X68`, `-mXX`, `-a16`, `-u1`, `-W1`, `-k1` i `-K1`. Cada opció significa una optimització i algunes són d'altíssim risc, per la qual cosa caldrà conèixer molt bé el disc. Per a consultar els paràmetres optimitzats, es podria utilitzar `hdparm -vtT /dev/hdX` (en què `X` és el disc optimitzat) i la crida a hdparm amb tots els paràmetres es pot posar en `/etc/init.d` per carregar-la en l'arrencada. Per a consultar la informació del disc es pot fer, per exemple, `hdparm -i /dev/sdb`

1.1.4. Configuracions complementàries

Hi ha més configuracions complementàries des del punt de vista de la seguretat que de l'optimització, però són necessàries sobretot quan el sistema està connectat a una intranet o a Internet. Aquestes configuracions impliquen les accions següents [14]:

1) Impedir que es pugui arrencar un altre sistema operatiu: si algú té accés físic a la màquina, podria arrencar amb un altre sistema operatiu preconfigurat i modificar l'actual, per la qual cosa cal inhibir des del BIOS de l'ordinador l'arrencada per CD-ROM o USB i posar una contrasenya d'accés (recordeu la contrasenya del BIOS, ja que no fer-ho podria causar problemes quan se'n volgués canviar la configuració).

2) Configuració i xarxa: és recomanable desconnectar la xarxa sempre que es vulguin fer ajustos en el sistema. Es pot treure el cable o deshabilitar el dispositiu amb `/etc/init.d/networking stop` (`start` per a activar-la de nou)o amb `ifdown eth0` (`ifup eth0` per a habilitar-la) per a un dispositiu en concret.

3) Modificar els arxius de `/etc/security`: d'acord amb les necessitats d'utilització i seguretat del sistema. A `access.conf` hi ha informació sobre qui pot iniciar sessió en el sistema; per exemple:

```
# Taula de control d'accés. Línies amb clomuna1=# és un comentari.
# L'ordre de les línies és important
# Format: permission : users : origins
# Deshabilitar tots els inicis de sessió excepte arrel sobre tty1
-:ALL EXCEPT root:tty1
# L'usuari primari té permès connectar-se des d'aquestes adreces
+ : root : 192.168.200.1 192.168.200.4 192.168.200.9
+ : root : 127.0.0.1
# O des de la xarxa
+ : root : 192.168.201.
# Impedeix l'accés excepte user1,2,3 però l'últim només des de la consola.
-:ALL EXCEPT user1 user2 user3:console
```

També caldria, per exemple, configurar els grups per a controlar com i on poden accedir i també els límits màxims (`limits.conf`) per a establir els temps màxims d'utilització de CPU, E/S, etc. i així evitar atacs per denegació de servei (DoS).

4) Mantenir la seguretat de la contrasenya d'usuari principal: utilitzar com a mínim 6 caràcters, amb almenys un en majúscules o algun caràcter que sigui no trivial, com "-", ".", ",", etc.; així mateix, és recomanable activar l'envelliment per a forçar a canviar-lo periòdicament, i també limitar el nombre de vegades amb contrasenya incorrecta. També es pot canviar el paràmetre `min=x` de l'entrada en `/etc/pam.d/passwd` per a indicar el nombre mínim de caràcters que s'utilitzaran en la contrasenya (`x` és el nombre de caràcters).

5) No accedir al sistema com a usuari principal: si bé moltes distribucions ja incorporen un mecanisme d'aquest estil (per exemple, Ubuntu), es pot crear un compte com sysadm i treballar-hi. Si s'accedeix remotament, caldrà utilitzar sempre `shh` per a connectar-se al sysadm i, si fos necessari, fer un `su -` per a treballar com a usuari principal o activar el `sudoers` per a treballar amb l'ordre `sudo` (consulteu la documentació per a les opcions de l'ordre i la seva edició)

6) Temps màxim d'inactivitat: inicialitzar la variable `TMOUT`, per exemple a 360 (valor expressat en segons), que serà el temps màxim d'inactivitat que esperarà l'intèrpret d'ordres abans de bloquejar-se; es pot posar en els arxius de configuració de l'intèrpret d'ordres (per exemple, `/etc/profile`, `.profile`, `$HOME/.bashrc`, etc.). En cas d'utilitzar entorns gràfics (KDE, Gnome, etc.), es pot activar l'estalvi de pantalla amb contrasenya, igual que el mode de suspensió o hibernació.

7) Configuració de l'NFS en forma restrictiva: en el `/etc/exports`, exportar només el necessari, no utilitzar comodins (*wildcards*), permetre només l'accés de lectura i no permetre l'accés d'escriptura per usuari principal, per exemple, amb `/directori_exportat host.domain.com (ro, root_squash)`.

8) Evitar arrencades des del carregador de l'arrencada amb paràmetres: es pot iniciar el sistema com a *linux single*, la qual cosa arrencarà el sistema operatiu en mode d'usuari únic. Cal configurar el sistema per tal que l'arrencada d'aquest mode sigui sempre amb contrasenya. Per a fer-ho, en l'arxiu `/etc/inittab` s'ha de verificar que existeix la línia `S:wait:/sbin/sulogin` i que té habilitat el `/bin/sulogin`. Per exemple, si treballem amb Lilo com a carregador de l'arrencada, l'arxiu `/etc/ lilo.conf` ha de tenir els permisos adequats per tal que ningú el pugui modificar excepte l'usuari principal (`chmod 600 /etc/lilo.conf`). Per a prevenir canvis accidentals, cal canviar l'atribut de bloqueig amb `chattr +i /etc/lilo.conf` (utilitzeu `-i` quan es vulgui canviar). Aquest arxiu permet una sèrie d'opcions que convé tenir en compte: `timeout=0` si el sistema només té un sistema operatiu perquè faci l'arrencada immediatament; `restricted` per a evitar que es puguin inserir ordres en el moment de l'arrencada com a `linux init = /bin/sh` i tenir accés com a usuari principal sense autorització; en aquest cas, s'ha d'acompanyar de `password=paraula-de-contrasenya`; si només es posa `password`,

sol·licitarà la contrasenya per carregar la imatge del nucli. Consulteu Grub(2), que té opcions similars.

Es pot provar el risc que això representa fent el següent (sempre que Grub/Lilo no tingui contrasenya), que pot ser útil per a entrar en el sistema quan no es recordi la contrasenya d'usuari, però representa un gran perill de seguretat quan és un sistema i es té accés a la consola i el teclat:

a) s'arrenca l'ordinador fins que es mostri el menú d'arrencada,

b) se selecciona el nucli que es vol arrencar i s'edita la línia tot pressionant la tecla *e* (edit),

c) busquem la línia que comença por `kernel...` i pressionem novament la tecla *e*,

d) al final de la línia esborrem el paràmetre `ro` i introduïm `rw init=/bin/bash` (la qual cosa indica accés directe a la consola),

e) pressionem *enter* i finalment *b* (boot). Arrenquem el sistema i passarem directament a mode usuari principal; gràcies a això es podrà canviar la contrasenya (inclosa la de l'usuari principal), editar el fitxer `/etc/passwd` o el `/etc/shadow` o també crear un nou usuari i tot el que vulguem.

9) Control de la combinació *Ctrl-Alt-Delete*: per a evitar que s'apagui la màquina des del teclat, cal inserir un comentari (#) en la primera columna de la línia següent: `ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now` de l'arxiu `/etc/inittab`. Els canvis s'activen amb l'ordre `telinit q`.

10) Evitar peticions de serveis no oferts: cal bloquejar `/etc/services`, per a no admetre serveis no previstos, mitjançant `chattr +i /etc/services`.

11) Connexió de l'usuari principal: cal modificar l'arxiu `/etc/securetty` que conté les TTY i VC (*virtual console*) en què es pot connectar l'usuari principal deixant-ne només una de cada, per exemple, `tty1` i `vc/1` i, si és necessari, cal connectar-se com a sysadm i fer un `su`.

12) Eliminar usuaris no utilitzats: cal esborrar els usuaris o grups que no siguin necessaris, fins i tot els que venen per defecte (per exemple, operator, shutdown, ftp, uucp, games, etc.) i deixar només els necessaris (usuari principal, bin, dimoni, sync, nobody, sysadm) i els que s'hagi creat amb la instal·lació de paquets o per ordres (el mateix amb `/etc/group`). Si el sistema és crític, podria considerar-se el bloqueig (`chattr +i file`) dels arxius `/etc/passwd`, `/etc/shadow`, `/etc/group` i `/etc/gsahdow` per a evitar-ne la modificació (compte amb aquesta acció, perquè no permetrà canviar les contrasenyes posteriorment).

13) Muntar les particions en forma restrictiva: usar en `/etc/fstab` atributs per a les particions com `nosuid` (que impedeix suplantar l'usuari o grup sobre la partició), `nodev` (que no interpreta dispositius de caràcters o blocs sobre aquesta partició) i `noexec` (que no permet l'execució d'arxius sobre aquesta partició). Per exemple: `/tmp /tmp ext2 defaults,nosuid,noexec 0 0`. També és aconsellable muntar l'arrencada en una partició separada i amb atributs `ro`.

14) Proteccions vàries: es pot canviar a 700 les proteccions dels arxius de `/etc/init.d` (serveis del sistema) perquè només l'usuari principal pugui mo-

dificar-los, arrencar-los o aturar-los i modificar l'arxiu `/etc/issue` i l'arxiu `/etc/issue.net` perquè no donin informació (sistema operatiu, versió, etc.) quan algú es connecta per telnet, ssh, etc.

15) SUID i SGID: un usuari podrà executar com a propietari una ordre si té el bit `SUID` o `SGID` activat, la qual cosa es reflecteix com una 's' SUID (-rwsr-xr-x) i SGID (-r-xr-sr-x). Per tant, és necessari treure el bit (`chmod a-s file`) a les ordres que no el necessiten. Aquests arxius poden buscar-se amb: `find / -type f -prem -4000 o -perm -2000 -print`. Cal procedir amb cura pel que fa als arxius en què es tregui el SUID-GUID, perquè l'ordre podria quedar inutilitzada.

16) Arxius sospitosos: cal buscar periòdicament arxius amb noms no usuals, ocults o sense un uid/gid vàlid, com "..." (tres punts), ".. " (punt punt espai), "..Ĝ" o equivalents. Per a això, caldrà utilitzar: `find / -name=".*" -print | cat -v` o sino `find / -name ".." -print`.

Per a buscar uid/gid no vàlids, utilitzeu `find / -nouser` (o utilitzeu també `-nogroup` (compte, perquè algunes instal·lacions es fan amb un usuari que després no està definit i que l'administrador ha de canviar).

17) Connexió sense contrasenya: no s'ha de permetre l'arxiu `.rhosts` en cap usuari, a no ser que sigui estrictament necessari (es recomana utilitzar `ssh` amb clau pública en lloc de mètodes basats en `.rhosts`).

18) X Display manager: per a indicar els amfitrions que es podran connectar a través d'XDM i evitar que qualsevol amfitrió pugui tenir una pantalla d'inici de sessió es pot modificar l'arxiu `/etc/X11/xdm/Xaccess`.

1.1.5. Resum d'accions per a millorar un sistema

1) Observar l'estat del sistema i analitzar els processos que consumeixen molta CPU utilitzant, per exemple, l'ordre `ps auxS -H` (o l'ordre `top`) i mirant les columnes `%CPU`, `%MEM` i `TIME`; cal observar la jerarquia de processos i prestar atenció a com s'està utilitzant la CPU i la memòria i analitzar el temps d'execució dels processos per a trobar processos *zombies* mirant en la columna `STAT` aquells que tinguin l'identificador `Z` (que es podran eliminar sense problemes). També cal prestar especial atenció als que estiguin amb `D`, `S` (que estan fent entrada o sortida) i `W` (que estan utilitzant el sistema d'intercanvi). En aquests tres últims utilitzeu atsar i free (sar o vmstat) per a verificar la càrrega d'entrada i sortida, ja que pot ser que aquests processos estiguin fent que les prestacions del sistema baixin notablement (en general per les seves necessitats, en aquest cas no podrem fer gran cosa, però en altres casos pot ser que el codi no estigui optimitzat o ben escrit).

2) Analitzar l'estat de la memòria amb detall per a descobrir on s'està gastant la memòria. Recordeu que tots els processos que cal executar han d'estar en memòria principal i, si no n'hi ha, el procés paginarà en memòria d'intercanvi però amb la conseqüent pèrdua de prestacions, ja que ha d'anar al disc i portar

zona de memòria principal. És vital que els processos més actius tinguin memòria principal i això es pot aconseguir canviant l'ordre d'execució o fent un canvi de prioritats (ordre `renice`). Per a observar l'estat de la memòria amb detall utilitzeu vmstat 2 (o atsar), per exemple, i observeu les columnes `swpd`, que és la quantitat de memòria virtual d'intercanvi utilitzada, `free`, la quantitat de memòria principal lliure (l'ocupada s'obté de la total menys la lliure) i `si/so`, la quantitat de memòria virtual en lectura o escriptura utilitzada. Si tenim un procés que utilitza gran quantitat de memòria d'intercanvi (`si/so`), aquest procés estarà gastant molt de temps en gestió, retardarà el conjunt i veurem que la CPU té, per exemple, valors d'utilització baixos. En aquest cas caldria eliminar processos de la memòria per a fer espai o ampliar la memòria RAM del sistema si és que no es poden treure els processos que hi ha, sempre que el procés en estudi no sigui d'execució ocasional.

3) Teniu en compte que %CPU + % E/S + %Idle = 100%, per la qual cosa veurem que l'E/S (I/O en vmstat) també afecta un procés.

```
  debian:/home/remo# vmstat 2
procs ----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 ...
 0  0      0 623624  29400 231596    0    0  7184     0  393 1389 10  8 10 73
 0  0      0 623596  29400 231612    0    0     0     0  416  800  0  2 98  0
 1  0      0 622540  29408 231604    0    0     0   276  212  549  2  2 94  2
 0  0      0 613544  29536 240620    0    0  4538     0  464 1597 10  8 29 54
 0  0      0 612552  29560 240824    0    0   112     0  412  850  1  2 85 12
```

En aquest cas podem observar que hi ha una utilització molt gran d'I/O (E/S) però 0 de memòria d'intercanvi un i alt valor de CPU tant en `wa` (*waiting*) com en `id` (*idle*), cosa que vol dir que la CPU està esperant que alguna cosa que és en E/S acabi (en aquest cas és l'execució d'unes quantes instàncies d'OpenOffice; això significa lectura de disc i càrrega d'un executable en memòria principal). Si aquesta situació es repeteix o és constant, s'hauria d'analitzar com utilitzen la memòria els processos en espera d'execució i com reduir-la (per exemple, posant un disc més ràpid o amb més memòria intermèdia d'E/S).

4) Cal tenir en compte que el %CPU està constituït per la suma de dos valors: "us" (User Time) i "sy" (System Time). Aquests valors representen el temps emprat executant codi de l'usuari (*non-kernel code*) i el temps gastat executant codi del nucli, respectivament, i poden ser útils quan es vol optimitzar el codi d'un programa amb la finalitat que consumeixi menys temps de CPU. Utilitzarem l'ordre `time`, que ens dóna el temps gastat en cada tipus de codi, fent, per exemple, `time find /usr`, de manera que tindrem que ens dóna `real 1m41.010s, user 0m0.076s, sys 0m2.404s`; en canvi, si fem `time ls -R /usr` la sortida és `real 0m5.530s user 0m0.160s sys 0m0.068s`. Com veiem, per a obtenir informació equivalent (llista d'arxius i directoris) una ordre ha emprat 2,404 s en espai de nucli i l'altra 0,06 s, per la qual cosa és interessant analitzar quines ordres escollim per a fer el treball. Un altre aspecte interessant és que si executem, per exemple, `time find /var > /dev/null` (per a no veure la sortida), la primera vegada obtenim `real 0m23.900s, user 0m0.000s, sys 0m0.484s` però una segona vegada obtenim `real 0m0.074s, user 0m0.036s, sys 0m0.036s`. Què ha succeït?

El sistema ha emmagatzemat en les taules de memòria cau la informació i les següents vegades ja no triga el mateix, sinó molt menys. Si es vol utilitzar `time` en el format avançat o estès, els usuaris que executin *bash* com a intèrpret d'ordres hauran d'executar `time` juntament amb el camí on es trobi; per exemple `/usr/bin/time ls -R /usr`, per a obtenir els resultats desitjats (consulteu `man time` per obtenir més informació).

5) És interessant veure quines optimitzacions podem generar en un codi amb modificacions simples. Per exemple, observem el codi elaborat per Bravo [1]:

```
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
int main(void)
{int x=1, i=2, z=3; long iter1=0,iter2=0;
struct timeval tv1,tv2;
gettimeofday(&tv1,NULL);
for(;;){
    x=(x*3+i*7+z*9)%11;
    i=(x*9+i*11+z*3)%29;
    z=(x*17+i*13+z*11)%37;
    iter1++;
    if(iter1==1000000){ iter2++; iter1=0;}
    gettimeofday(&tv2,NULL);
    if(tv2.tv_sec==tv1.tv_sec+5 && tv2.tv_usec>=tv1.tv_usec || tv2.tv_sec>tv1.tv_sec+5)
    break;}
printf("Iteraciones: %ldM Resultado: %d %d %d\n",iter2,x,i,z);
return 0;}
```

El resultat de l'execució és

```
time ./c: Iteracions: 22M real 0m5.001s, user 0m1.756s, sys 0m3.240s
```

S'hi pot observar que els 3,240 s han arribat per a 22 milions d'iteracions. En què es gasten els 3,240 s? Doncs en calcular l'hora en cada iteració, ja que són múltiples crides al nucli. Si millorem el codi i només calculem el `gettimeofday` cada milió d'iteracions, obtenim `Iteracions: 135M real 0m5.025s, user 0m4.968s, sys 0m0.056s` i veiem que es redueix notablement el temps `sys` i obtenim més temps per a executar el codi de l'usuari, per la qual cosa augmenta el nombre d'iteracions (135 milions), tenint en compte que hem passat de 22 milions d'execucions de la crida `gettimeofday` a 135 milions de vegades. Quina és la conseqüència? Que la finalització d'execució d'aquest exemple s'obté per comparació de 5 s amb el temps absolut, per la qual cosa en calcular el temps absolut menys vegades s'obté certa "imprecisió" en determinar quan finalitza l'execució (`real 0m5.001s` en el primer cas, mentre que en el segon `0m5.025s`, una diferència de 24 mil·lèsimes). Una solució optimitzada per a aquest cas seria no utilitzar aquest tipus de crides al sistema per a determinar quan ha de finalitzar un procés i buscar alternatives, per exemple, amb `alarm` i una crida a `signal`. [1]

1.2. Monitoratge

Un aspecte important en el funcionament 24x7 d'un sistema és que l'administrador s'ha d'anticipar als problemes i per això o bé està contínuament mirant

el seu funcionament (és pràcticament impossible que ho faci tot el temps), o bé es disposa d'eines adequades que puguin prevenir la situació, generar alertes i advertir al responsable del fet que "alguna cosa està passant" i que aquest pugui emprendre amb antelació les accions correctives per a evitar la fallada, disfunció o situació de fora de servei del sistema o recurs. Les eines que compleixen aquesta funció s'emmarquen dins del grup d'eines de monitoratge i permeten també obtenir informació del sistema amb finalitats estadístiques, comptables o d'un altre tipus que l'usuari vulgui. Les eines més comunes permeten, mitjançant una interfície web, conèixer remotament els cinc factors principals (ús de CPU, memòria, E/S, xarxa, processos/serveis) que donen indicis que "alguna cosa pot estar passant"; les més sofisticades generen alarmes per SMS per a advertir l'administrador de la situació. A continuació, es descriuran algunes de les eines més representatives (però no són les úniques): Munin, Monit, MRTG i Cacti.

Cacti es descriu en el mòdul "Clusterització", juntament amb Ganglia, ja que estan més orientades a clústers d'ordinadors.

1.2.1. Munin

Munin [16] produeix gràfics sobre diferents paràmetres del servidor (load average, memory usage, CPU usage, MySQL throughput, eth0 traffic, etc.) sense excessives configuracions i presenta gràfics importants per a reconèixer on i què està generant problemes. Considerem que el nostre sistema es diu `debian.nteum.org` i tenim la pàgina configurada com a `debian.nteum.org` amb els documents a `/var/www/`. Per a instal·lar Munin sobre Debian fem, per exemple, `apt-get install munin munin-node`. Després haurem de configurar Munin (`/etc/munin/munin.conf`) amb:

```
dbdir /var/lib/munin
htmldir /var/www/munin
logdir /var/log/munin
rundir /var/run/munin
tmpldir /etc/munin/templates
[debian.nteum.org]
    address 127.0.0.1
    use_node_name yes
```

Després es crea el directori, es canvien els permisos i es reinicia el servei (en cas de no existir).

```
mkdir -p /var/www/munin
chown munin:munin /var/www/munin
/etc/init.d/munin-node restart
```

Al cap d'uns minuts es podran veure els primers resultats en l'adreça web http://debian.nteum.org/munin o a `http://localhost/munin` en el navegador. Si es vol mantenir la privacitat dels gràfics n'hi ha prou de posar una contrasenya per a accedir amb Apache al directori. Per exemple, es posa en el directori `/var/www/munin/` l'arxiu `.htaccess` amb el contingut següent:

```
AuthType Basic
AuthName “Members Only”
AuthUserFile /var/www/munin/.htpasswd
<limit GET PUT POST>
   require valid-user
</limit>
```

Després cal crear l'arxiu de contrasenya a `/var/www/munin/.htpasswd` amb l'ordre (sent usuari principal): `htpasswd -c /var/www/munin/.htpasswd admin`. Quan ens connectem al `http://localhost/munin/` ens demanarà l'usuari (admin) i la contrasenya que hem introduït després de l'ordre anterior.

1.2.2. Monit

Monit [10] permet configurar i verificar la disponibilitat de serveis com Apache, MySQL o Postfix i pren accions diferents, com per exemple reactivar-los si no estan presents. Per a instal·lar Monit fem `apt-get install monit` i editem `/etc/monit/monitrc`. L'arxiu per defecte inclou un conjunt d'exemples, però caldrà consultar la documentació per a obtenir més informació*. A continuació presentem un exemple típic de configuració sobre alguns serveis `/etc/monit/monitrc` [11]:

*http://mmonit.com/monit

```
# Monit control file example: /etc/monit/monitrc
#
    set daemon 120 # Poll at 2-minute intervals
    set logfile syslog facility log_daemon
    set alert root@debian.nteum.org
# Set a default mail from-address for all alert messages emitted by monit.
# All alert mail will be sent to below mail address.
    set mail-format { from: root@debian.nteum.org }
    set alert root@debian.nteum.org
# Make monit start its web-server. So you can acccess it from webrowser.
    set httpd port 2812 and
    use address localhost
##Monit web-server ACL.
    allow localhost       # allow localhost to connect to the server and
#allow 192.168.1.2    # allow 192.168.1.2 to connect to the server,
                      # You can give only one per entry
    allow admin:monit     # user name and password for authentication.
    allow hardik:hardik   # set multiple user to access through browser.
# Monitoring the apache2 web services. It will check process apache2 with given pid file.
# If process name or pidfile path is wrong then monit will
# give the error of failed. tough apache2 is running.
    check process apache2 with pidfile /var/run/apache2.pid
#Below is actions taken by monit when service got stuck.
    start program = “/etc/init.d/apache2 start”
    stop program = “/etc/init.d/apache2 stop”
# Admin will notify by mail if below of the condition satisfied.
    if cpu is greater than 60% for 2 cycles then alert
    if cpu > 80% for 5 cycles then restart
    if totalmem > 200.0 MB for 5 cycles then restart
    if children > 250 then restart
    if loadavg(5min) greater than 10 for 8 cycles then stop
    if 3 restarts within 5 cycles then timeout
    group server
    check process sshd with pidfile /var/run/sshd.pid
    start program = “/etc/init.d/ssh start”
    stop program = “/etc/init.d/ssh stop”
    if failed port 22 protocol ssh then restart
    if 5 restarts within 5 cycles then timeout
```

```
    check process mysql with pidfile /var/run/mysqld/mysqld.pid
    group database
    start program = "/etc/init.d/mysql start"
    stop program = "/etc/init.d/mysql stop"
    if failed host 127.0.0.1 port 3306 then restart
    if 5 restarts within 5 cycles then timeout

#Check host for which services up/down on particular port.
    check host debian.nteum.org with address 10.0.2.15
    if failed icmp type echo with timeout 4 seconds then alert
    if failed port 21 then alert
    if failed port 80 protocol http then alert
#   if failed port 389 type tcp with timeout 15 seconds then alert

#include /etc/monit/default.monitrc
#include /etc/monit/mysql.monitrc
```

Enllaç d'interès

Consulteu el manual per obtenir més detalls a http://mmonit.com/monit.

Per verificar que la sintaxi és correcta executem `monit -t` i per posar-lo en marxa executem `monit`. A partir d'aquest moment es pot consultar en l'arxiu `/etc/monit/monitrc` de l'adreça i el port seleccionats (en el nostre cas `http://localhost:2812/`), que ens demanarà l'usuari i la contrasenya també introduïts en el mateix arxiu (admin i monit en el nostre cas). El servei es pot aturar i arrencar amb `/etc/init.d/monit stop|start`, encara que perquè ens permeti fer l'arrencada, haurem de canviar a `/etc/default/monit` la variable `startup=1`.

1.2.3. MRTG

L'MRTG (*Multi-Router Traffic Grapher*) [15] va ser creat per a mostrar informació gràfica sobre dades de xarxa, com es veurà en l'exemple que mostrem a continuació, per a monitorar la xarxa Ethernet, però es poden utilitzar altres dades per a visualitzar el comportament i per a generar les estadístiques de càrrega (*load average*) del servidor. Per a això utilitzem els paquets mrtg (`apt-get install mrtg mrtg-contrib mrtgutils`) i atsar. Un cop instal·lats, configurem el fitxer `/etc/mrtg.cfg` [5].

```
# Multi Router Traffic Grapher -- Sample Configuration File
# This file is for use with mrtg-2.5.4c

# Global configuration
    WorkDir: /var/www/mrtg
    WriteExpires: Yes
    Title[^]: Traffic Analysis for

# 128K leased line
# ----------------
#Title[leased]: a 128K leased line
#PageTop[leased]: <H1>Our 128K link to the outside world</H1>
#Target[leased]: 1:public@router.localnet
#MaxBytes[leased]: 16000

    Target[eth1]: '/usr/bin/mrtg-ip-acct eth1'
    MaxBytes1[eth1]: 32000
    MaxBytes2[eth1]: 16000
    Title[eth1]: Packets from eth1
    YLegend[eth1]: Traffic
    PageTop[eth1]: <H3>Analysis of total Internet traffic</H3>
```

```
Target[average]: '/usr/bin/cpu-load-mrtg'
MaxBytes[average]: 1000
Options[average]: gauge, nopercent, growright, integer
YLegend[average]: Load average
kMG[average]: ,,
ShortLegend[average]:
Legend1[average]: Load average x 100
LegendI[average]: load:
LegendO[average]:
Title[average]: Load average x 100 for debian.nteum.org
PageTop[average]: <H3>Load average x 100 for debian.nteum.org</H3>
<TABLE>
    <TR><TD>System:</TD>
        <TD>debian.nteum.org</TD></TR>
    <TR><TD>Maintainer:</TD> <TD>webmaster@debian.nteum.org</TD></TR>
    <TR><TD>Max used:</TD> <TD>1000</TD></TR>
</TABLE>
```

Per generar les dades amb atsar (o sar) creem un *script* en el directori següent `/usr/local/bin/cpu-load/average` (que tingui permisos d'execució per a tots) i que passarà les dades a mrtg:

```
#!/bin/sh
load='/usr/bin/atsar -u 1 | tail -n 1 | awk -F" " '{print $9}''
echo "($load -100) * 100 " | bc | awk -F"." '{print $1}'
```

Haurem de crear i canviar els permisos del directori `/var/www/mrtg`. Per defecte, mrtg s'executa en el `cron`, però si després volem executar-lo, podem executar l'ordre `mrtg /etc/mrtg.cfg` i això generarà els gràfics en l'arxiu `/var/www/mrtg/average.html`, que podrem visualitzar amb un navegador des d'http://debian.nteum.org/mrtg/averange.html o des de l'adreça `http://localhost/mrtg/averange.html`. D'altra banda, el gràfic de càrrega de CPU o el de càrrega de xarxa els podrem consultar en l'adreça següent: http://debian.nteum.org/mrtg/eth1.html o també des de aquesta altra adreça: `http://localhost/mrtg/eth1.html`. A continuació teniu una possible execució de mrtg a través del cron en l'arxiu `/etc/cron.d/mrtg`:

```
*/5 * * * * root if [ -x /usr/bin/mrtg ] && [ -r /etc/mrtg.cfg ];
then env LANG=C /usr/bin/mrtg /etc/mrtg.cfg 2>&1 | tee -a /var/log/mrtg/mrtg.log ; fi
```

Enllaç d'interès

Hi ha eines més sofisticades per al monitoratge de xarxa i serveis de xarxa utilitzant SNMP (*Simple Network Management Protocol*) i MRTG (*Multi-Router Traffic Grapher*), per exemple. En trobareu més informació en l'adreça següent: http://www.linuxhomenetworking.com /wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance.

1.2.4. Altres eines

Altres paquets interessants que s'han de tenir en compte per a monitorar un sistema són els següents:

- **Nagios***: Nagios és un sistema de monitoratge de xarxes de codi obert àmpliament utilitzat que permet monitorar tant el maquinari com els serveis

*http://www.nagios.org

que s'especifiquin i genera alertes quan el comportament no sigui el desitjat. Permet el monitoratge de serveis de xarxa (SMTP, POP3, HTTP, SNMP, etc.), de recursos de sistemes de maquinari (càrrega del processador, ús dels discos, memòria, estat dels ports, etc.), té possibilitat de monitoratge remota mitjançant túnels SSL xifrats o SSH i la possibilitat de programar connectors (*plug-ins*) específics per a nous sistemes.

Enllaç d'interès

Podeu consultar la guia d'instal·lació ràpida de Nagios per a Debian a: http://debianclusters.org/index.php/Nagios_Installation_and_Configuration [6].

- **Cacti***: Cacti és una solució gràfica dissenyada per a treballar conjuntament amb dades d'RRDTool. Cacti proveeix diferents formes de gràfics, mètodes d'adquisició i característiques que pot controlar l'usuari molt fàcilment i és una solució que s'adapta des d'una màquina a un entorn complex de màquines, xarxes i servidors.

*http://cacti.net

Cacti es descriu en el mòdul "Clusterització".

- **Frysk***: l'objectiu del projecte Frysk és crear un sistema de monitoratge distribuït i intel·ligent per a monitorar processos i fils.

*http://sources.redhat.com/frysk

A continuació es descriuran altres eines no menys interessants (per ordre alfabètic) que incorpora GNU/Linux (por exemple, Debian) per al monitoratge de sistema. No és una llista exhaustiva, sinó una selecció de les més utilitzades (es recomana veure el manual de cada eina per a més informació):

- **atsar**, **ac**, **sac**, **sysstat**, **isag**: eines d'auditoria com ac, last, accton, sa, atsar o isag (Interactive System Activity Grapher) per a l'auditoria de recursos hw i sw.
- **arpwatch**; **mon**: monitor d'activitat Ethernet/FDDI que indica si hi ha canvis en taules MACIP; monitor de serveis de xarxa.
- **diffmon**, **fcheck**: generació d'informes sobre canvis en la configuració del sistema i monitoratge dels sistemes de fitxers per a detectar intrusions.
- **fam**: file alteration monitor, monitor d'alteració de fitxers.
- **genpower**: monitor per a gestionar les fallades d'alimentació.
- **gkrellm**: monitoratge gràfic de CPU, processos (memòria), sistemes de fitxers i usuaris, disc, xarxa, Internet, memòria d'intercanvi, etc.
- **ksensors (lm-sensors)**: monitor de la placa base (temperatura, alimentació, ventiladors, etc.).
- **lcap**, **systune**: eina per a retirar capacitats assignades al nucli en el fitxer `/proc/sys/kernel` i s'adapta segons les necessitats amb systune.
- **powertweak**: monitoratge i modificació de diferents paràmetres del maquinari, nucli, xarxa, VFS o VM (permet modificar alguns dels paràmetres mostrats anteriorment sobre `/proc`).
- **gps**, **gtop**, **lavaps (de més a menys amigable)**: monitors de processos de diversos tipus (generalment utilitzen informació de `/proc`) i permeten veure recursos, sòcols (*sockets*), arxius, entorn i altres tipus d'informació que aquests utilitzen, i també administrar-ne els recursos i els estats.
- **swatch**: monitor per a l'activitat del sistema a través d'arxius de registre.
- **vtgrab**: monitoratge de màquines remotes (similar a VNC).
- **whowatch**: eina en temps real per al monitoratge d'usuaris.
- **wmnd**: monitor de trànsit de xarxa i monitoratge d'un clúster per xarxa.
- **xosview**: monitor de recursos gràfic.

1.3. Alta disponibilitat en Linux (*high-availability Linux*)

Actualment Linux és conegut com un sistema operatiu estable; els problemes es generen quan el maquinari falla. En els casos en què una fallada de maquinari provoca greus conseqüències per la naturalesa del servei (aplicacions crítiques), s'implementen sistemes tolerants a fallades (*fault tolerant*, FT) amb els quals es garanteix, amb una determinada probabilitat (molt alta), que el servei estigui sempre actiu. El problema d'aquests sistemes és que són extremadament cars, solen ser solucions tancades, totalment dependents de la solució integrada. Els sistemes d'alta disponibilitat (*high availability*, HA) intenten obtenir prestacions properes a la tolerància a fallades, però a costos accessibles. L'alta disponibilitat està basada en la replicació d'elements, per la qual cosa deixarem de tenir un servidor i necessitarem tenir un clúster d'alta disponibilitat. Existeixen per a Linux diferents solucions, com per exemple Heartbeat (element principal del Linux-HA), ldirectord i LVS (Linux Virtual Server), Piranha (solució basada en LVS de xarxa Hat), UltraMonkey (solució de VA Linux), Kimberlite (de Mission Critical Linux) o OpenAIS+Corosync+Pacemaker.

El projecte Linux-HA (Linux d'alta disponibilitat) [8] és una solució clúster d'alta disponibilitat per a Linux i altres sistemes operatius, com per exemple Solaris, FreeBSD, OpenBSD i MacOSX, i que ofereix fiabilitat, disponibilitat i prestació discontínua de serveis. El producte principal del projecte és **Heartbeat**, l'objectiu principal del qual és la gestió de grups de sectors amb la finalitat d'obtenir alta disponibilitat. Les seves característiques més importants són: nombre il·limitat de nodes (útil tant per a petits clústers com per a mides grans), monitoratge de recursos (aquests es poden reiniciar o desplaçar a un altre node en cas de fallada), mecanisme de cerca per a eliminar nodes amb fallades del clúster, gestió de recursos basada en directives o regles amb possibilitat d'incloure el temps, gestió preconfigurada de recursos (Apache, DB2, Oracle, PostgreSQL, etc.) i interfície gràfica de configuració. Per tal de poder ser útils als usuaris, el dimoni de Heartbeat s'ha de combinar amb un administrador de recursos de clúster (CRM), que té la tasca d'iniciar i detenir els serveis (adreces IP, servidors web, etc.) i proporcionarà l'alta disponibilitat. Des de la versió 2.1.3 de Heartbeat s'ha substituït el codi del gestor de recursos del clúster (CRM) pel component Pacemaker. Pacemaker aconsegueix la màxima disponibilitat dels seus serveis de clúster mitjançant la detecció i recuperació de nodes i les fallades de nivell de servei. Això s'aconsegueix mitjançant la utilització de les capacitats de missatgeria i la pertinença a la infraestructura proporcionada OpenAIS o Pacemaker.

1.3.1. Guia breu d'instal·lació de Heartbeat a Debian

En aquest subapartat es donarà una breu descripció de com construir un clúster d'alta disponibilitat de dos nodes amb Heartbeat*. És interessant (tot i que una mica antic) l'article [7] i la documentació del lloc web de Linux-HA [3]. Com a punt inicial disposem de dos ordinadors similars (no és necessari, però

***La informació detallada es pot consultar a** `file:///usr/share/doc/heartbeat/GettingStarted.html`.

si un ha d'ocupar el lloc de l'altre, és aconsellable). Als servidors els anomenarem NteumA (primari) i NeteumB (secundari), amb dos interfícies de xarxa cada un: la primera perquè es comuniquin entre sí i l'altra servirà a cadascun d'ells per a connectar-se a Internet, per exemple. D'aquesta forma NteumA: eth0 = 10.0.2.20 i NteumB: eth0 = 10.0.2.30. Aquestes dues eth0 es connecten a un encaminador l'adreça privada del qual és 10.0.2.1. Per a instal·lar Heartbeat a Debian executem `apt-get install heartbeat heartbeat-dev`; també podríem instal·lar `heartbeat-gui`, però no ho utilitzarem en aquesta guia. Aquesta instal·lació crearà el directori `/etc/ha.d` per a configurar el clúster de dos nodes d'HA i haurà de contenir tres arxius `ha.cf` (*main configuration file*) `haresources` (*resource configuration file*) i `authkeys` (*authentication information*). El fitxer `/etc/ha.d/ha.cf` (se'n pot obtenir un exemple des de `/usr/share/doc/heartbeat/`) és el principal fitxer de Heartbeat i hauria de tenir les opcions necessàries per tal que Heartbeat funcioni correctament (és necessari comentar que aquestes configuracions són vàlides per a ambdós nodes).

```
# Interfície de xarxa on s'aixecarà heartbeat
bcast eth0
# Pings a la interfície de l'encaminador per a validar que hi ha connexió a Internet
ping 10.0.2.1
# Pings cada 2 segons a l'altre node
keepalive 2
# L'absència de resposta de 5 segons fa que aquest node estigui en alerta
warntime 5
# Absència de resposta de 10 segons, l'altre node es considera fora de servei
deadtime 10
# Port UDP per defecte de Heartbeat segons IANA
udpport 694
auto_failback on
# Declaració dels dos nodes
node NeteumA
node NeteumB
```

Arxiu `/etc/ha.d/haresources` on s'identifica el node primari com a [MasterHost] [IP_Virtual] [serveis]

```
NteumA 10.0.2.15 httpd smb
```

La IP_Virtual és la que aixecarà Heartbeat per a referir-se a l'*aliàsing* de xarxa, és a dir, el que utilitzarà tant si NteumA (primari) cau com si no, de tal manera que si NteumA funciona realment anirà a l'adreça 10.0.2.20 i, en cas contrari, a la 10.0.2.30 (NteumB). Els serveis disponibles s'anomenen tal com apareixen a `/etc/init.d`, atès que és on Heartbeat els gestionarà.

L'arxiu `/etc/ha.d/authkeys` proporciona una mesura de seguretat per a evitar que hi hagi suplantació d'identitat tot permetent autenticació a sha, md5 o crc (menys segur). Aquest arxiu hauria de tenir atributs de 600 (`chmod 600 /etc/ha.d/authkeys`) i el seu contingut serà semblant a:

```
auth 1
1 sha1 key-for-sha1-cualquier-texto-que-se-desee
```

No menys important és configurar el `/etc/hosts`, per exemple:

```
127.0.0.1 localhost
10.0.2.15 router
10.0.2.20 NteumA
10.0.2.30 NteumB
```

Atès que Heartbeat utilitza el trànsit ICMP per a comunicar-se entre nodes i en el nostre cas l'encaminador, caldrà modificar el tallafoc per tal que permeti aquest tipus de trànsit (és important també considerar que es generaran paquets ARP i que algun programa com Snort o altres IDS poden considerar això com a atacs de falsejament d'ARP).

Per provar-ne la funcionalitat, primer assignarem les adreces manualment (o por DHCP) a les eth0 dels servidors:

```
ifconfig eth0 down ; ifconfig eth0 up 10.0.2.20 netmask 255.255.255.0 (NteumA)
ifconfig eth0 down ; ifconfig eth0 up 10.0.2.30 netmask 255.255.255.0 (NteumB)
```

Executant `/etc/init.d/heartbeat start` posem en marxa Heartbeat en ambdós nodes (es recomana posar en marxa primer el node primari i després, el secundari). Per provar-ne la funcionalitat, pressuposem que ambdós nodes estan funcionant, però només el node primari (NteumA) està donant el servei i el Heartbeat del node secundari (NteumB) està enviant contínuament *pings* per comprovar que efectivament NteumA segueix fent el seu treball. Per a això es pot intentar fer una connexió http a l'adreça 10.0.2.15 (la del `haresources`) i la resposta serà que s'està a NteumA (primari); si apaguem (desconnectem el cable de xarxa d'eth0) el node primari i es torna a entrar 10.0.2.15 serà NteumB qui haurà de donar la resposta (si tot ha funcionat correctament, si no, desplegarà un missatge de *Not Found* o *Forbidden*). Per a més detalls, consulteu la documentació a `/usr/share/doc/heartbeat`.

Activitats

1. Feu un monitoratge complet del sistema amb les eines que considereu adequades i feu un diagnòstic de la utilització de recursos i colls d'ampolla que hi podria haver en el sistema. Simuleu la càrrega en el sistema del codi de `sumdis.c` donat en el mòdul "Clusterització". Per exemple, utilitzeu: `sumdis 1 2000000`.

2. Canvieu els paràmetres del nucli i del compilador i executeu el codi mencionat en l'activitat anterior (`sumdis.c`) amb, per exemple: `time ./sumdis 1 1000000`.

3. Amb l'execució de les dues activitats anteriors, extraieu conclusions sobre els resultats.

4. Amb el programa d'iteracions indicat en aquest mòdul, implementeu una manera de finalitzar l'execució en 5 s, independent de les crides al sistema excepte que sigui només un cop.

5. Monitoreu tots els programes anteriors amb Munin.

6. Registreu quatre serveis amb Monin i feu la gestió i seguiment per mitjà del programa.

7. Instal·leu MRTG i monitoreu la CPU de l'execució dels programes anteriors.

8. Instal·leu i experimenteu amb els sistemes descrits d'alta disponibilitat.

Bibliografia

[1] **Bravo E., D.** (2006). *Mejorando la Performance en sistemas Linux/Unix*. GbuFDL1.2. <diegobravoestrada@hotmail.com>

[2] *Cacti*.
<http://cacti.net/>

[3] *Documentació de Linux-HA*.
<http://www.linux-ha.org/doc/users-guide/users-guide.html>

[4] *Frysk*.
<http://sources.redhat.com/frysk/>

[5] Howto *sobre instal·lació de MRTG en Debian*.
<http://preguntaslinux.org/-howto-instalacion-de-mrtg-monitoreo-debian-t-3061.html>

[6] *Instal·lació de Nagios sobre Debian*.
<http://debianclusters.org/index.php/Nagios_Installation_and_Configuration>
<http://debianclusters.org/index.php/Nagios>

[7] **Leung, C. T**. *Building a Two-Node Linux Cluster with Heartbeat*.
<http://www.linuxjournal.com/article/5862;>

[8] *Linux-HA*.
<http://linux-ha.org/wiki/Main_Page>

[9] **Majidimehr, A.** (1996). *Optimizing UNIX for Performance*. Prentice Hall.

[10] *Monit*.
<http://mmonit.com/monit/>

[11] *Monitor Debian servers with monit*.
<http://www.debian-administration.org/articles/269>

[12] *Monitoratge amb Munin i monit*.
<http://www.howtoforge.com/server_monitoring_monit_munin>

[13] *Monitoratge amb SNMP i MRTG*.
<http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance>

[14] **Mourani, G.** (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.

[15] *MRTG*.
<http://oss.oetiker.ch/mrtg/>

[16] *Munin*.
<http://munin.projects.linpro.no/>

[17] *Optimització de servidors Linux*.
<http://people.redhat.com/alikins/system_tuning.html>

[18] *Pacemaker*.
<http://clusterlabs.org/wiki/Main_Page>

[19] *Performance Monitoring Tools for Linux*.
<http://www.linuxjournal.com/article.php?sid=2396>