

Sintonització, optimització i alta disponibilitat

Remo Suppi Boldrito

PID_00239613

Índex

Introducció	5
Objectius	7
1. Sintonització, optimització i alta disponibilitat	9
1.1. Aspectes bàsics	9
1.1.1. Monitoratge sobre UNIX System V	10
1.2. Test de rendiment (<i>Benchmark</i>)	17
1.2.1. Optimització del sistema	19
1.2.2. Optimitzacions de caràcter general	22
1.2.3. Configuracions complementàries	23
1.2.4. Resum d'accions per a millorar un sistema	27
1.3. Monitoratge.....	30
1.3.1. Munin	30
1.3.2. Monit	31
1.3.3. SNMP + MRTG	33
1.3.4. Nagios	37
1.3.5. Ganglia	40
1.3.6. Cacti.....	42
1.3.7. Altres eines	44
1.4. Alta disponibilitat en Linux (<i>High-Availability Linux</i>)	46
1.4.1. Guia simple clúster amb Heartbeat per HA amb Apache	47
1.4.2. Clúster HA amb Corosync+Pacemaker y Nginx	49
1.4.3. DRBD	52
1.4.4. DRBD + Heartbeat com a NFS d'alta disponibilitat ...	54
Activitats	57
Bibliografia	58

Introducció

Un aspecte fonamental, una vegada que el sistema està instal·lat, és la configuració i adaptació del sistema a les necessitats de l'usuari i que les prestacions del sistema siguin el més adequades possible a les necessitats que ha de cobrir. 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, una vegada feta una instal·lació (o en alguns casos, una actualització), han de fer-se determinades configuracions vitals al sistema. Si bé el sistema "funciona", és necessari efectuar 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 uns altres (a més), per a qüestions de seguretat. Quan el sistema està en funcionament, és necessari monitorar-lo per a veure el seu comportament i actuar en conseqüència. Si bé és un aspecte fonamental, la sintonització d'un sistema operatiu moltes vegades es relega a l'opinió d'experts o gurus de la informàtica; però coneixent els paràmetres que afecten el rendiment, és possible arribar a les bones solucions fent un procés cíclic d'anàlisi, canvi de configuració, monitoratge i ajustos.

D'altra banda, amb les necessitats de serveis actuals, els usuaris són molt exigents amb la qualitat de servei que s'obté i és per això que un administrador ha de preveure les incidències que puguin ocórrer en el sistema abans que ocorrin. Per a això, és necessari que l'administrador "vigili" de manera continuada determinats paràmetres de comportament dels sistemes que el puguin ajudar en la presa de decisions i actuar per endavant evitant que es produeixi l'error, ja que si això passés podria suposar la possibilitat que el sistema deixés de funcionar, amb les possibles conseqüències econòmiques en alguns casos, però gairebé sempre amb la deterioració de la imatge de l'empresa/institució (a ningú no li agrada -o no li hauria d'agradar- que els usuaris informin d'una fallada en els sistemes d'informació).

En resum, un sistema de monitoratge ha d'ajudar l'administrador a reduir el MTTR (*Mean time to recovery*), que indica la mitjana de temps que un sistema triga a recuperar-se d'una fallada i que pot tenir un valor dins del contracte de qualitat de servei (normalment anomenat SLA, *Service Level Agreement*), per la qual cosa també pot tenir conseqüències econòmiques. Aquest valor de vegades es denomina "*mean time to replace/repair/recover/resolve*" en funció de quin sistema es tracti i de quin SLA s'hagi acordat, però en tots els casos estem parlant de la finestra de temps entre la qual es detecta un problema i les accions per a solucionar-lo. Amb el monitoratge, podrem tenir indicis d'aquests pro-

bles i executar les accions perquè no arribin a més i que si ocorren, tinguin el MTTR més baix possible.

En aquest mòdul es veuran les principals eines per a monitorar un sistema GNU/Linux, com són Munin, Monit, MRTG, Ganglia, Nagios, Cactis o Zabbix, i es donaran indicacions per a sintonitzar el sistema a partir de la informació obtinguda.

És important notar que si el MTTR és proper a zero, el sistema haurà de tenir redundància en els altres sistemes i és un aspecte important en l'actualitat per als servidors de sistemes de la informació, que es coneix com a *alta disponibilitat*.

L'alta disponibilitat (*high availability*) és un protocol de disseny del sistema, i la seva implementació associada 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 per a LVS -Linux Virtual Server-, OpenSAF, Piranha, UltraMonkey, Pacemaker+Corosync, o Kimberlite (obs:EOL), etc.), en aquest mòdul analitzarem com es desenvolupa una arquitectura redundat en serveis utilitzant Heartbeat+DRBD i Pacemaker + Corosync.

Vegeu també

La seguretat s'estudia en el mòdul "Administració de seguretat".

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 de monitoratge i la seva integració per a resoldre els problemes d'eficiències/disponibilitat.
- 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, s'ha enumerar les causes que poden afectar les prestacions d'un sistema operatiu [Maj96]. Entre aquestes, es poden esmentar:

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 ocorren, coneixent les seves limitacions teòriques i 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 millorarà les prestacions (*speedup*) en un 5%, la qual cosa pot significar un tremend esforç no compensat pels resultats.

3) **Estimació de l'*speedup*:** és necessari estimar quant millorarà les prestacions el sistema per a evitar esforços i costos innecessaris. Es pot utilitzar la llei d'Amdahl per a valorar si és necessària una inversió, en temps o econòmica, 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 enfront de quantitat de treball:** si es compta amb vint usuaris, millorar en la productivitat significarà que tots tindran més treball fet al mateix temps, però no millors respostes individualment; podria ser que el temps de resposta per a alguns fos millor que per a altres. Millorar el temps de resposta significa optimitzar el sistema perquè 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. S'ha d'anar amb compte quan es fan les proves pels efectes col·laterals dels mateixos programes de mesura.

8) **Importància de la mitjana i la variació:** s'han de tenir en compte els resultats, ja que si s'obté una mitjana d'utilització de CPU del 50% quan ha estat utilitzada 100, 0, 0, 100, es podria arribar a conclusions errònies. És important veure la variació sobre la mitjana.

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

10) **Coneixements bàsics sobre el sistema operatiu que cal 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, terminació), crides al sistema, *buffers* de cau, sistema d'arxius, administració de memòria i memòria virtual (paginació, *swap*) i taules del nucli (*kernel*).

1.1.1. Monitoratge sobre UNIX System V

El directori `/proc` el veurem com un directori, però en realitat és un sistema d'arxius fictici anomenat *procfs* (i que es munta en temps de *boot* de la màquina), 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 després serà utilitzada per *totes* les ordres que veurem a continuació. El `/proc` actua com a interfície en l'estructura de dades internes de nucli i pot ser utilitzat per a canviar certa informació del kernel en temps d'execució (`sysctl`). No hem de confondre **procfs** amb el **sysfs** ja que aquest últim exporta informació sobre els dispositius i els seus controladors des del model de dispositius del nucli cap a l'espai de l'usuari (i és utilitzat per algunes parts importants del sistema com l'`udev`, que és el que crea per compatibilitat el `/dev`) permetent obtenir paràmetres i configurar-ne algun (p. ex., saber la grandària d'un disc `cat /sys/block/sda/size` o quins dispositius tenim en `/sys/class`). Una vista del directori `/proc` és (pot diferir segons la instal·lació):

bus	locks	cgroups	meminfo
cmdline	misc	consoles	modules
cpuinfo	mounts	crypto	mtrr
devices	net	diskstats	pagetypeinfo
dma	partitions	dri	sched_debug
driver	self	execdomains	slabinfo
fb	softirqs	filesystems	stat
fs	swaps	interrupts	sys
iomem	sysrq-trigger	ioports	sysvipc
irq	timer_list	kallsyms	timer_stats
kcore	tty	keys	uptime

key-users	version	kmsg	vmallocinfo
acpi	kpagecount	vmstat	asound
kpageflags	zoneinfo	buddyinfo	loadavg

A més d'una sèrie de directoris numèrics que corresponen a cadascun dels processos del sistema.

En el directori `/proc` existeixen un conjunt d'arxius i directoris amb diferent informació. A continuació, en veurem alguns dels més interessants*:

- `/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 aquest moment.
- `/proc/filesystems`: sistemes d'arxius configurats en el nucli.
- `/proc/interrupts`: mostra quines interrupcions estan en ús i quantes se n'han processat.
- `/proc/ioports`: ídem amb els ports.
- `/proc/kcore`: imatge de la memòria física del sistema.
- `/proc/kmsg`: missatges generats pel nucli, que després 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/modulis`: 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 manera dinàmica cada vegada 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 homogeneïtat de presentar informació interna i pot ser utilitzada per a les diverses eines/ordres d'informació/sintonització/control que utilitzarem regularment. És interessant, per exemple, veure la sortida de l'ordre `mount` i el resultat de l'execució `more /proc/mounts` és totalment equivalent!

S'ha de tenir en compte que aquests arxius són visibles (text), però algunes vegades les dades estan "en brut" i són necessàries ordres per a interpretar-les, que seran les que veurem a continuació. Els sistemes compatibles UNIX SV utilitzen les ordres `sar` i `sadc` per a obtenir estadístiques del sistema. En Debian és `atsar` (i `atsadc`), que és totalment equivalent a les que hem esmentat 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) i `sadf` (mostra informació del `sar` en diversos formats). L'ordre `atsar` llegeix comptadors i estadístiques del fitxer `/proc` i els mostra per la sortida estàndard. La primera manera de cridar l'ordre és (executar-la com a `root` o agregar l'usuari a la categoria corresponent del `sudoers` per a executar amb el `sudo`):

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

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

Vegeu també

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

On mostra l'activitat en n vegades cada t segons amb una capçalera que mostra els comptadors d'activitat (el valor per defecte de $n = 1$). La segona manera de cridar-la és:

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

L'ordre extreu dades de l'arxiu especificat per `-f` (que per defecte és l'arxiu `/var/log/atsar/atsarxx`, essent `xx` el dia del mes) i que van ser prèviament guardades 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 ser utilitzat per a indicar el dia del mes i `-s`, `-i` l'hora d'inici i final, respectivament. Per a activar `atsadc`, per exemple, es podria incloure en `/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 guarda les dades cada 10 minuts amb el *shell script* `atsa1`, que crida el `atsadc`. En `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
l (i)	Nombre d'interrupcions
v	Utilització de taules en el nucli
y	Estadístiques d'utilització de ttys
p	Informació de paginació i activitat de <i>swap</i>
r	Memòria lliure i ocupació de <i>swap</i>
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 existeixen 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`, només pot haver-hi alguna diferència en la visualització de les dades) i el significat de la informació que genera.

Per instal·lar el paquet `sysstat`, farem `apt-get install sysstat`; amb això tindrem instal·lades les ordres `sar`, `iostat`, `mpstat` i `vmstat`, i podrem visualitzar algunes estadístiques, però no aquelles acumulades. Per fer això,

haurem d'editar l'arxiu `/etc/default/sysstat` i canviar `ENABLED="false"` per `ENABLED="true"`, i reiniciar el servei `service sysstat restart`. Així, si fem `sar -A 4 5`, veurem totes les estadístiques cada 4 segons, i un total de 5 repeticions, i, al final, ens donarà els valors mitjans acumulats.

Utilització de CPU:

```
sar -u 4 5
Linux 3.16.0-4-amd64 (srv) 14/07/16 _x86_64_ (4 CPU)

17:03:29      CPU      %user   %nice   %system   %iowait   %steal   %idle
17:03:33      all        0.88    0.00    0.06     0.00     0.00    99.06
17:03:37      all        5.39    0.00    3.01     0.06     0.00    91.54
17:03:41      all       16.53    0.00    8.52     0.00     0.00    74.95
17:03:45      all       16.04    0.00    8.90     0.00     0.00    75.06
17:03:49      all       16.58    0.00    8.42     0.00     0.00    75.00
Average:      all       11.09    0.00    5.78     0.01     0.00    83.12
```

`%usr` i `%system` 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). En el cas `idle=99,06%` significa que la CPU està ociosa, per la qual cosa no hi ha processos per a executar i la càrrega és baixa; si *idle proper al 0%* i el nombre de processos és elevat, hauria de pensar-se a optimitzar la CPU i també poca activitat d'entrada/sortida per la qual cosa representa que la càrrega de 25% de la 3a. a la 5a. línia solament és de CPU i no té impacte sobre l'E/S.

Nombre d'interrupcions per segon: `-I 1,15,16,19 4`

```
Linux 3.16.0-4-amd64 (srv) 14/07/16 _x86_64_ (4 CPU)

17:24:11      INTR      intr/s
17:24:15        1         0.25
17:24:15       15         1.00
17:24:15       16         0.50
17:24:15       19        26.25

...

Average:      INTR      intr/s
Average:        1         0.10
Average:       15         0.96
Average:       16         0.48
Average:       19        23.04
```

Aquesta ordre mostra les interrupcions (en aquest cas l'1, 15, 16 i 19 que són *mouse* i disc, *eth1*, *eth0*) i que es troben en el `/proc/interrupts`. És útil per determinar què dispositius estan interrompent el treball de la CPU i per analitzar la causa. Podem veure en l'exemple que hi ha molta activitat de xarxa sobre *eth0* i poca de la resta dels dispositius. Es pot posar la paraula `ALL` per veure les primeres 16 interrupcions i `XALL` per veure-les totes.

Memòria i swap: `sar -r 1 1`

Linux 3.16.0-4-amd64 (srv) 14/07/16 _x86_64_ (4 CPU)

```
17:49:28 kbmemfree kbmemused %memused kbbuffers kbcached kbcommit %commit kbactive kbinact kbdirty
17:49:29 93324 930748 90.89 59824 361828 1521584 148.58 591008 172184 56
Average: 93324 930748 90.89 59824 361828 1521584 148.58 591008 172184 56
```

I on (en kilobytes) `kbmemfree` és la quantitat de memòria que hi ha disponible; `kbmemused`, la utilitzada; `%memused`, el percentatge usat; `kbbuffers`, la memòria en emmagatzematge intermedi (*buffers*); `kbcached`, la memòria caché; `kbcommit`, la memòria estimada sobre la quantitat necessària per a realitzar el treball (memòria + swap) –anomenat *workload*–; `%commit`, el percentatge del *workload*; `kbactive`, la memòria usada i no alliberada fins que no sigui necessari; `kbinact`, la memòria inactiva; i `kbdirty`, la memòria que espera ser transferida al disc. És també interessant l'ordre `free`, que permet veure la quantitat de memòria en una visió simplificada:

```
Mem: total used free shared buffers cached
-/+ 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 cau. A més, ens indica que el *swap* 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 la `sar -r`) per a 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 a 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 3.16.0-4-amd64 (srv) 07/10/16 _x86_64_ (4 CPU)

```
11:39:23 dentunusd file-nr inode-nr pty-nr
11:39:27 11465 1760 17228 1
11:39:31 11465 1760 17228 1
11:39:35 11465 1760 17228 1
11:39:39 11465 1760 17227 1
11:39:43 11465 1760 17227 1
Average: 11465 1760 17228 1
```

On `dentunusd` és el nombre d'entrades no utilitzades en la caché; `file-nr`, el nombre de descriptors d'arxius utilitzats pel sistema; `inode-nr`, el nombre de descriptors d'inodes utilitzats pel sistema; i `pty-nr` el nombre de pseudoterminals utilitzats (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 estat dels processos en el sistema. A continuació, es mostren dos exemples de totes dues 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 per a 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 cadascun 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 *daemon*); `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, 'zombi'); `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 actualització (configurable), es pot utilitzar l'ordre `top`, que mostra unes estadístiques generals (processos, estats, càrrega, etc.) i, després, informació de cadascuna similar a la `ps`, però s'actualitza cada 5 segons per defecte (en mode gràfic està `gnome-system-monitor`):

```
top - 15:09:08 up 21 min,  2 users,  load average: 0.16, 0.15, 0.12
Tasks: 184 total,  2 running, 182 sleeping,  0 stopped,  0 zombie
%Cpu(s):  0.3 us,  2.8 sy,  0.0 ni, 96.8 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  1509992 total,  846560 used,  663432 free,  117304 buffers
KiB Swap: 1087484 total,    0 used, 1087484 free,  374076 cached
  PID USER      PR  NI  VIRT  RES  SHR  S %CPU  %MEM    TIME+  COMMAND
 4144 root        20   0  201m  36m 8448 S   9.6   2.5   0:39.35 Xorg
 4694 adminp    20   0  980m  64m 29m S   6.7   4.4   0:26.22 gnome-shell
 4730 adminp    20   0  363m  16m 10m S   2.3   1.1   0:04.04 gnome-terminal
 4221 root        20   0 69796 1776 1140 S   0.3   0.1   0:01.62 nmbd
 4655 adminp    20   0  571m  26m 13m S   0.3   1.8   0:01.00 gnome-settings-
 6287 root        20   0 15080 1520 1072 R   0.3   0.1   0:00.10 top
    1 root        20   0 10648  812  676 S   0.0   0.1   0:01.21 init
    2 root        20   0    0    0  0 S   0.0   0.0   0:00.00 kthreadd
    3 root        20   0    0    0  0 S   0.0   0.0   0:00.93 ksoftirqd/0
```

Una ordre interessant i que presenta la informació d'una altra manera que pot servir per a tenir una panoràmica de tot el sistema és `atop` (s'ha d'instal·lar `apt-get install atop`). A continuació, unes línies d'aquesta ordre ens mostren la seva potencialitat:

```

ATOP - srv                2016/07/14 18:14:59                -----                10s elapsed
PRC | sys    0.14s | user  0.74s | #proc  124 | #tslpu  0 | #zombie  0 | #exit  0 |
CPU | sys    1% | user  5% | irq    0% | idle   395% | wait    0% | curscal  ?% |
cpu | sys    0% | user  2% | irq    0% | idle   98% | cpu003 w 0% | curscal  ?% |
cpu | sys    0% | user  1% | irq    0% | idle   99% | cpu001 w 0% | curscal  ?% |
cpu | sys    0% | user  2% | irq    0% | idle   98% | cpu000 w 0% | curscal  ?% |
CPL | avg1  0.05 | avg5  0.13 | avg15  0.13 | csw    8096 | intr   2215 | numcpu   4 |
MEM | tot   1.0G | free  142.5M | cache  333.9M | dirty  0.0M | buff   59.1M | slab  136.3M |
SWP | tot   0.0M | free  0.0M |                |                | vmcom  1.5G | vmlim  500.0M |

  PID RUID      THR  SYSCPU  USRCPU  VGROW  RGROW  RDDSK  WRDSK  ST  EXC  S  CPUNR  ACPU  CMD
  867 root       48   0.11s  0.60s   64K   1052K   0K     0K  --  -  R   0    7%  firefox-esr
  711 root        5   0.00s  0.14s    0K     0K    0K     0K  --  -  S   0    1%  Xorg
 1171 root        1   0.02s  0.00s    0K     0K    0K     0K  --  -  R   3    0%  atop
 1100 root        1   0.01s  0.00s    0K     0K    0K     0K  --  -  S   2    0%  kworker/2:1
   824 root        3   0.00s  0.00s    0K     0K    0K     0K  --  -  S   1    0%  lxterminal
   809 root        4   0.00s  0.00s    0K     0K    0K     0K  --  -  S   3    0%  lxpanel
     7 root        1   0.00s  0.00s    0K     0K    0K     0K  --  -  R   3    0%  rcu_sched
   202 root        1   0.00s  0.00s    0K     0K    0K     0K  --  -  S   3    0%  kworker/3:2

```

A més de les eines ja vistes del paquet `sysstat` com `sar` (*system activity reporter*), és interessant analitzar amb detall la potencialitat de `vmstat` (estadístiques de CPU, memòria i entrada/sortida), `iostat` (estadístiques de discos i CPU) i `uptime` (càrrega de CPU i estat general) i `mpstat` (estadístiques de cada *core*/processador).

Un resum de les ordres més interessants és:

Ordre	Descripció
<code>atop, top</code>	Activitat dels processos
<code>arpwatch</code>	monitor d'Ethernet/FDDI
<code>bmon, bwm-ng, nload</code>	monitor de l'amplada de banda
<code>downtimed</code>	Monitor del temps de caiguda, fora de servei
<code>free</code>	Utilització de memòria
<code>iostat, iotop</code>	Activitat de disc i E/S
<code>ip monitor, rtmon, iptotal, iptraf</code>	Monitor de dispositius de xarxa
<code>mpstat</code>	Estadístiques del processador
<code>netstat</code>	Estadística de la xarxa
<code>nfswatch</code>	Monitor de NFS
<code>ps, pstree, god</code>	Mostra estat i característiques dels processos
<code>/proc</code>	Sistema d'arxius virtual
<code>sysstat, atsar</code>	Recull informació del sistema
<code>stat, iwatch</code>	Estadístiques del sistema d'arxius
<code>strace</code>	Esdeveniments de les crides als sistemes i senyals
<code>tcpdump, etherape, sniffit</code>	Abocament / monitor de paquets de xarxa
<code>uptime, w</code>	Càrrega mitjana del sistema i temps des de l'inici
<code>vmstat</code>	Estadístiques de l'ús de memòria
<code>gnome-system-monitor, gkrellm, xosview, xwatch</code>	Monitors gràfics del sistema
<code>xconsole</code>	Monitor de missatges en l'escriptori

També hi ha una sèrie de programes que mesuren les prestacions del sistema (*benchmark*) o d'una part com per exemple: netperf, mbw (xarxa i amplada de banda), iozone (E/S), sysbench, globs, gtkperf, hpcc (general), bonie++ (disc). És important destacar que el *benchmark* hpcc inclou *High-Performance LINPACK* (HPL), *benchmark* que és l'utilitzat per a mesurar les prestacions i fer el rànquing de les màquines més potents del món*.

*<http://www.top500.org/>

1.2. Test de rendiment (*Benchmark*)

Com a programa de *Benchmarking* i per provar el rendiment de la nostra màquina, utilitzarem Phoronix Test Suite (PTS), que té llicència GPL3 i pot ser utilitzat amb finalitats personals i/o professionals, com ara *open source*. Aquest programa permet realitzar proves de manera automàtica amb una configuració molt simple (inclou més de 450 plantilles de test agrupades en 100 tipus diferents), que van des del consum de potència (CPU, GPU, memòria, disc o components de la placa base...), temperatura, ample de banda, fins al còmput, E/S, entre d'altres. A més, disposa d'una arquitectura molt flexible, de manera que, si no existeix un test, es pot afegir ràpidament i sense grans dificultats, utilitzant algunes de les plantilles disponibles.

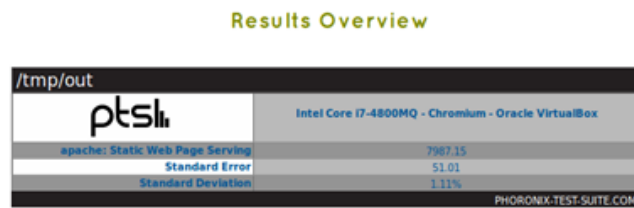
La seva instal·lació des dels repositoris de GitHub és molt senzilla:

Phoronix necessita *php-cli* i *php-xml* que estan dins del paquet *php5-cli*.

- 1) `apt-get install git php5-cli`
- 2) `git clone https://github.com/phoronix-test-suite/phoronix-test-suite.git`
- 3) `cd phoronix-test-suite/`
- 4) `./phoronix-test-suite` *Mostra totes les opcions*
- 5) `./phoronix-test-suite interactive` *S'executa en mode interactiu*
- 6) `./phoronix-test-suite gui` *Inicia la interfície gràfica*
- 7) `./phoronix-test-suite list-tests` *Llista tots els tests*
- 8) `./phoronix-test-suite info systester` *Dóna informació d'un test (per exemple, systester)*
- 9) `./phoronix-test-suite benchmark apache` *Realitza un test a Apache*

La figura 1 mostra el resultat de l'execució del test d'Apache sobre una màquina virtual i la figura 2, la imatge general de l'aplicació quan s'executa amb el paràmetre *gui*, i s'executa el test C-Ray de la llista dels més populars.

Figura 1



Test Results

Apache Benchmark Static Web Page Serving

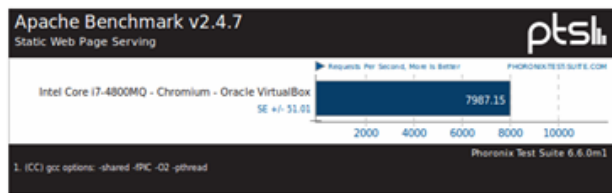
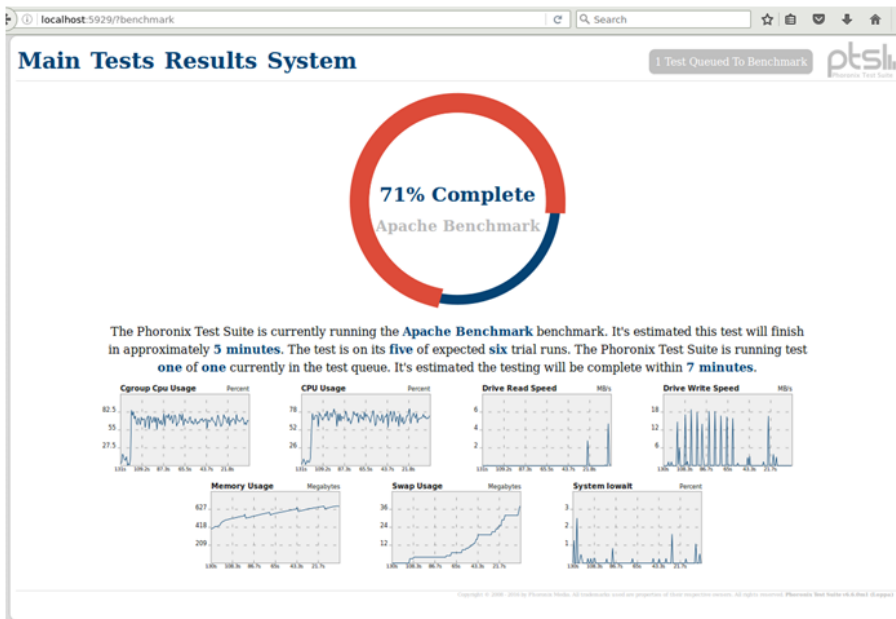


Figura 2



La Suite Phoronix permet una gran quantitat d'opcions i tests per avaluar tots els components del sistema. Funciona en mode local o remot o es pot instal·lar de manera definitiva sobre el servidor (vegeu l'script *install.sh* en el directori de la distribució). Podeu consultar les opcions en la documentació [PTS] o en el man *phoronix-test-suite* (una vegada instal·lat).

1.2.1. 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 al *swap*; però això significa que l'execució d'aquest procés es degradarà notablement. Si s'afegia memòria, el temps de resposta milloraria força. Per a això, s'ha de tenir en compte la grandària dels processos (*SIZE*) en estat *R* i afegir-hi la que usa 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 megaoctets) 1011/706/305/121/358 i on observem, en aquest cas, que només el 30% de la memòria està lliure i que en aquest moment no hi ha problemes, però una càrrega mínima del sistema pot significar un problema. Per aquest motiu, haurem d'analitzar si el sistema està limitat per la memòria (amb *sar -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 la seva grandària que emprar moltes hores per a guanyar un centenar de bytes en treure, ordenar o reduir requeriments dels processos en la seva execució. Reduir els requeriments pot fer-se reduint les taules del nucli, traient mòduls, limitant el nombre màxim d'usuaris, reduint els *buffers*, 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 això, s'hauran de monitorar els processos que estan dormint (zombis) i eliminar-los, o bé aquells que no progressen en la seva entrada/sortida (saber si són processos actius, quanta CPU han gastat i si els usuaris els estan esperant). Canviar la càrrega de treball és utilitzar planificació de cues perquè els processos que necessiten gran quantitat de memòria es puguin executar en hores de poca activitat (per exemple, a la nit, llançant-los amb l'ordre *at*).

2) Molta utilització de CPU: bàsicament ens la dona 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 posposar la seva execució, parar-los temporalment, canviar la seva prioritat (és la solució menys conflictiva de totes i per a això es pot utilitzar l'ordre *renice* prioritat *PID*), optimitzar el programa (per a la propera vegada) o canviar la CPU (o afegir-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 cert cas, “ajustades”, per a aconseguir prestacions diferents o millors.

Per a això, s'ha d'utilitzar l'ordre `sysctl -a` per a obtenir totes les variables del nucli i els seus valors en l'arxiu*. Altres ordres alternatives són `systemd` i `systemd-tunedump`, que permeten descarregar les variables en un arxiu i modificar-les, per a carregar-les novament en el `/proc` (l'ordre `systemd` guarda la configuració en `/etc/systemd.conf`). En aquest cas, per exemple, es podrien modificar (s'ha de procedir amb cura, perquè el nucli pot quedar fora de servei) les variables de la categoria `/proc/sys/vm` (memòria virtual) o `/proc/sys/kernel` (configuració del *core* del nucli).

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

Una altra optimització que es pot fer (encara que hauria de considerar-se com “l'últim recurs” i solament per a usuaris avançats) és canviar el temps del *quantum* de l'algorisme *round robin* d'assignació de CPU als processos (consulteu `man sched_rr_get_interval`). Des del nucli de Linux 3.9 s'han incorporat nous mecanismes per ajustar i veure el valor de `SCHED_RR quantum` que es permet ajustar a través de la variable `/proc/sys/kernel/sched_rr_timeslice_ms` expressada en mil·lisegons i que té un valor per defecte de 100 (si es posa a 0 s'adoptarà el valor per defecte). Si volem canviar el valor d'aquesta variable modifiquem l'arxiu indicat o usem

Lectura recomanada

Podeu consultar [TunK], que mostra una guia interessant per sintonitzar el nucli de Linux (si bé és per a la distribució SUSE i els kernels 2.6 o superiors, la majoria dels valors s'apliquen pels kernels 3.x).

```
sysctl -w kernel.sched_rr_timeslice_ms=valor
```

(aquesta modificació serà temporal però si la hi vol fer permanent caldrà modificar l'arxiu `/etc/sysctl.conf`).

3) Reduir el nombre de crides: una altra pràctica adequada per a millorar les prestacions és reduir el nombre de crides al sistema de major cost en temps de CPU. Aquestes crides són les invocades (generalment) pel `shell fork()` i `exec()`. Una configuració inadequada de la variable `PATH` amb el directori actual (indicat per `.`) pot tenir una relació desfavorable d'execució (perquè la funció `exec()` no guarda res en cau i perjudica aquesta execució). Per a això, sempre caldrà configurar la variable `PATH` amb el directori actual com a última ruta. P. ex., en `$HOME/.bashrc` s'ha de fer `PATH=$PATH:.; export PATH` si el directori actual no és en el `path` o, si hi és, refer la variable `PATH` per a posar-lo com a última ruta.

S'ha de 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 (`sar -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 de dispositiu de xarxa per un altre de més intel·ligent o canviar l'estructura de comunicacions si detectem una activitat elevada de les interrupcions que genera o el canal de E/S al que es connecta.

4) Molta utilització de disc: després de la memòria, un temps de resposta baix pot estar provocat pel sistema de discos. En primer lloc, s'ha de 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 `sar -o` i `sar -d`. Les solucions passen per:

- En un sistema multidisc, planificar on es trobaran els arxius més utilitzats per a equilibrar el trànsit cap a aquests (per exemple `/home` en un disc i `/usr` en un altre) i que puguin utilitzar totes les capacitats d'entrada/sortida amb cau i concurrent de GNU/Linux (fins i tot, per exemple, planificar sobre quin *bus ide* es col·loquen). Comprovar després que hi ha un equilibri del trànsit amb `sar -d` (o amb `iostat`). En situacions crítiques, es pot considerar la compra d'un sistema de discos RAID que fan aquest ajust de manera automàtica.
- Tenir en compte que s'obtenen millors prestacions sobre dos discos petits que sobre un de la grandària dels dos anteriors.
- En sistemes amb un únic disc, generalment es fan, des del punt de vista de l'espai, quatre particions de la següent manera (des de fora cap a dins): `/`, `swap`, `/usr`, `/home`. Però això genera pèssimes respostes d'entrada/sortida perquè si, per exemple, un usuari compila des del seu directori `/home/user` i el compilador es troba en `/usr/bin`, el cap del disc es mourà al llarg de tota la seva longitud. En aquest cas, és millor unir les particions `/usr` i `/home` en una de sola (més gran), encara que pot representar alguns inconvenients quant a manteniment.
- Incrementar els *buffers* de cau d'entrada/sortida (podeu veure, per exemple, `/sys/block/sda...`).
- Si s'utilitza un `extfs`, es pot fer servir l'ordre `dumpe2fs -h /dev/sda1` per a obtenir informació sobre el disc i `tune2fs /dev/hdx` per a canviar alguns dels paràmetres configurables d'aquest.
- Òbviament, el canvi del disc per un de major velocitat (més rpm) sempre tindrà un impacte positiu en un sistema limitat per l'entrada/sortida de disc [Maj96].

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

- 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 aquests i els temps d'accés.

- Ajustar paràmetres del nucli (*/proc/sys/net/*). Per exemple, per a obtenir millores en el *throughput* hem d'executar la següent instrucció:

```
echo 600 > /proc/sys/net/core/netdev_max_backlog*.
```

*Mínim 300

6) Altres accions sobre paràmetres del nucli: hi ha altres paràmetres sobre el nucli que és possible sintonitzar per a obtenir millors prestacions, si bé, considerant el que hem tractat anteriorment, s'ha d'anar amb compte, ja que podríem causar l'efecte contrari o inutilitzar el sistema. Consulteu en la distribució del codi font en el directori *kernel-source-x.x/Documentation/sysctl* alguns arxius com per exemple *vm.txt*, *fs.txt* i *kernel.txt*. */proc/sys/vm* controla la memòria virtual del sistema (*swap*) i permet que els processos que no entren en la memòria principal siguin acceptats pel sistema però en el dispositiu de *swap*, per la qual cosa, el programador no té límit per a la grandària del seu programa (òbviament ha de ser menor que el dispositiu de *swap*). Els paràmetres susceptibles de sintonitzar es poden canviar molt fàcilment amb *sysctl*. */proc/sys/fs* conté paràmetres que poden ser ajustats de la interacció nucli-sistema de fitxers, tal com *file-max* (i exactament igual per a la resta d'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 primer llegir l'arxiu *readme* que hi ha al 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, resultarà més estable. Hi ha dues formes de construir un nucli: **monolític** (millors prestacions) o **modular** (basat en mòduls, que tindrà millor portabilitat si disposem d'un sistema molt heterogeni i no es desitja compilar un nucli per a cadascun). Per a compilar el seu propi nucli i adaptar-lo al seu maquinari i necessitats, cada distribució té les seves regles (si bé el procediment és similar).

1.2.2. Optimitzacions de caràcter general

Hi ha una sèrie d'optimitzacions d'índole 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 (*lib.a*), el codi de funció de la qual s'inclou en l'executable, o amb una de dinàmica (*lib.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

Enllaços d'interès

És interessant consultar els següents llibres/articles:
Linux Performance and Tuning Guidelines (https://lenovopress.com/redp4285?cm_mc_uid=63499174791814685798965&cm_mc_sid_50200000=1468579896.com/redpapers/pdfs/redp4285.pdf sobre Linux Performance and Tuning Guidelines, http://people.redhat.com/alikins/system_tuning.html sobre informació d'optimització de sistemes servidors Linux, i <http://www.linuxjournal.com/article/2396> sobre *Performance Monitoring Tools for Linux*. El primer és un llibre electrònic obert de la sèrie RedBooks d'IBM molt ben organitzat i amb gran quantitat de detalls sobre la sintonització de sistemes Linux, els dos restants són articles que si bé tenen un cert temps, els conceptes/metodologia i alguns procediments continuen vigents.

decidir quina és l'adequada per al seu programa incloent `-static` en les opcions del compilador (no posar-ho 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 a les anteriors. En gcc 4.X per defecte s'utilitzen biblioteques dinàmiques, però es pot forçar (no recomanat) l'ús d'estàtiques fins i tot per a la `libc` (opcions `-static -static-libgcc` en contraposició amb aquelles que hi ha 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 simplement

```
gcc -march=i686;
```

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

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

3) Optimització del disc: en l'actualitat, la majoria d'ordinadors inclou disc Sata II/III per defecte; no obstant això, en una gran quantitat de casos no estan optimitzats per a extreure les millors prestacions. Hi ha una eina (`hdparm`) que permet sintonitzar el nucli als paràmetres del disc tipus IDE i SATA (encara que aquests últims també disposen d'una utilitat específica anomenada `sdparm`). S'ha d'anar amb compte amb aquestes utilitats, sobretot en discos UltraDMA (cal verificar en el BIOS que els paràmetres per a suport per a DMA estan habilitats), ja que poden inutilitzar el disc. Consulteu les referències i la documentació ([Mou] 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`, `-o1`, `-W1`, `-k1`, `-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` o `/dev/sdX` (on X és el disc optimitzat), i la crida a `hdparm` amb tots els paràmetres es pot posar en `/etc/init.d` per a carregar-la en el *boot*. Per a consultar la informació del disc es pot fer, per exemple, `hdparm -i /dev/sdb`.

1.2.3. 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 següents accions [Mou]:

Es poden consultar totes les opcions a l'apartat *Linker Options* a <https://gcc.gnu.org/onlinedocs/gcc-4.9.3/gcc/option-Summary.html#Option-Summary>.

Paquets no-free

Recordeu que sobre Debian s'ha d'activar el repositori de paquets *no-free* per a instal·lar els paquets de documentació del compilador `gcc-doc` i `gcc-doc-base`.

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 s'ha d'inhibir des del BIOS de l'ordinador el *boot* per CD-ROM o USB i posar una contrasenya d'accés (recordeu la contrasenya del BIOS ja que, d'una altra manera, podria causar problemes quan es volgués canviar la configuració).

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

3) Modificar els arxius de `/etc/security`: d'acord amb les necessitats d'utilització i seguretat del sistema. En `access.conf` hi ha informació sobre qui pot fer un *login* al sistema; per exemple:

```
# Taula de control d'accés. Línies amb column1=# és un comentari.
# L'ordre de les línies és important.
# Format: permission : users : origins
# Deshabilitar tots els logins excepte root sobre tty1.
-:ALL EXCEPT root:tty1
# User "root" 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 consola.
-:ALL EXCEPT user1 user2 user3:console
```

També s'haurien, per exemple, de configurar els grups per a controlar com i a 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 de *root*: utilitzar com a mínim 8 caràcters, amb un, almenys, 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, així com 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). Hem d'utilitzar algorismes com ara SHA512 per a la configuració de `passwd` (en Debian ve configurat per defecte, vegeu `/etc/pam.d/common-password`).

5) No accedir al sistema com a *root*: 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'hi accedeix remotament, sempre s'haurà d'utilitzar `ssh` per a connectar-se al `sysadm` i, en cas de ser necessari, fer un `su -` per a treballar com a *root* 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 `TMOU`, per exemple a 360 (valor expressat en segons), que serà el temps màxim d'inactivitat que esperarà el *shell* abans de bloquejar-se; es pot posar en els arxius de configuració del *shell* (per exemple, `/etc/profile`, `.profile`, `$HOME/.bashrc`, etc.). En cas d'utilitzar entorns gràfics (KDE, Gnome, etc.), es pot activar l'estalvi de pantalles amb contrasenya, igual que el mode de suspensió o hibernació.

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

8) Evitar arrencades des del *bootloader* 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 perquè l'arrencada d'aquesta manera sempre sigui amb contrasenya. Per a això s'haurà de configurar per a cada entrada del menú un usuari (o conjunt d'ells) dins de l'etiqueta *superusers* per a indicar qui poden fer activitats considerades de reg per al sistema. Per configurar-ho haurem de:

a) Fer `vi /etc/grub.d/00_header` i inserim al final:

```
cat « EOF set superusers="adminp" password adminp passwd_desitjat EOF
```

b) Fer `vi /etc/grub.d/10_linux` (és recomanable fer prèviament una còpia d'aquest arxiu) i busquem l'entrada corresponent a la qual volem accedir amb *passwd*. Serà alguna cosa com

```
menuentry '$(tiro "$title" | grub_quote)' ${CLASS} \${menuentry_id} option ...
```

i li agregarem l'etiqueta *users* " (després d'*users* va un espai en blanc i dos caràcters ' seguits), quedant alguna cosa com

```
menuentry '$(tiro "$title grub_quote)' $CLASS -users " \${menuentry_id} option ...
```

Observació: si no es modifica una entrada però s'ha modificat *00_header*, s'aplicarà a totes les entrades indistintament.

c) Actualitzar Grub amb `update-grub` i provar l'arrencada (es pot verificar prèviament que `/boot/grub/grub.conf` estigui ben configurat en l'entrada corresponent).

d) Atès que el *passwd* va en clar en els arxius de configuració es pot utilitzar `grub-mkpasswd-pbkdf2` que ens donarà alguna cosa com: *PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.A21953DA...* On l'hash té 283 caràcters (aquí s'han posat solament els primers però en la configuració s'han d'inserir tots) i després modifiquem `/etc/grub.d/00_header` amb:

```
cat << EOF
set superusers="adminp"
password_pbkdf2 adminp grub.pbkdf2.sha512.10000.A21953DA...
EOF
```

Es pot provar el risc que això representa fent el següent (sempre que el 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 desitja arrencar i s'edita la línia pressionant la tecla *e* (*edit*),
- c) busquem la línia que comença per `kernel . . .` i al final de la línia esborrem el paràmetre `ro` i introduïm `rw init=/bin/bash` (la qual cosa indica accés directe a la consola). Pressionem F10.
- d) Amb això s'arrencarà el sistema i passarem directament a mode *root*, i gràcies a això es podrà canviar la contrasenya (inclosa la de *root*), editar el fitxer */etc/passwd* o el */etc/shadow* o també crear un nou usuari i tot el que desitgem (s'ha de muntar prèviament el / en manera *rw* ja que sinó els canvis no quedaran gravats).

9) Control de la combinació *Ctrl-Alt-Delete*: Anteriorment es modificava l'arxiu */etc/inittab* però en sistemes amb *systemd* haurem de fer servir l'ordre `systemctl mask ctrl-alt-del.target` i a continuació utilitzarem la instrucció `systemctl daemon-reload`.

10) Evitar peticions de serveis no oferts: s'ha de fer el bloqueig de l'arxiu */etc/services*, per a no admetre serveis no previstos, per mitjà de la instrucció `chattr +i /etc/services`.

11) Connexió del *root*: cal modificar l'arxiu */etc/securetty* que conté les TTY i VC (*virtual console*) en què es pot connectar el *root* deixant 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: s'han d'esborrar els usuaris o grups que no siguin necessaris, inclosos els que vénen per defecte (per exemple, *operator*, *shutdown*, *FTP*, *uucp*, *games*, etc.) i deixar només els necessaris (*root*, *bin*, *daemon*, *sync*, *nobody*, *sysadm*) i els que s'hagin 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*, */etc/gshadow* per a evitar la seva modificació (compte amb aquesta acció, perquè no permetrà canviar posteriorment les contrasenyes).

13) Muntar les particions en forma restrictiva: utilitzar en */etc/fstab* atributs per a les particions com ara *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). P. ex.: `/tmp /tmp ext2 defaults,nosuid,noexec 0 0`. També és aconsellable muntar el */boot* en una partició separada i amb atributs *ro*.

14) Proteccions diverses: es pot canviar a 700 les proteccions dels arxius de */etc/init.d* o */etc/systemd* (serveis del sistema) perquè només el *root* pugui modificar-los, arrencar-los o parar-los i modificar els arxius */etc/issue* i */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 (amb l'ordre `chmod a-s file`) a les ordres que no el necessiten. Aquests arxius poden buscar-se amb `find / -type f -perm -4000 -or -perm -2000`. S'ha de procedir amb cura respecte 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), "..^G" o equivalents. Per a això, caldrà utilitzar

```
find / -name=".*" -print | cat -v ,o si no, 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, tret 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 `hosts` que es podran connectar a través de XDM i evitar que qualsevol `host` pugui tenir una pantalla de `login` es pot modificar l'arxiu `/etc/X11/xdm/Xaccess`.

1.2.4. 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`; s'ha d'observar la jerarquia de processos i parer esment a com s'està utilitzant la CPU i la memòria i analitzar el temps d'execució dels processos per a trobar processos *zombis* mirant en la columna `STAT` aquells que tinguin l'identificador `Z` (els quals es podran eliminar sense problemes). També s'ha de prestar especial atenció a aquells que estiguin amb `D`, `S` (que estan fent entrada o sortida) i `W` (que estan utilitzant el *swap*). En aquests tres últims, utilitzeu `l'sar` 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 (generalment, per les seves necessitats; en aquest cas, no podem 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 detalladament per a descobrir on s'està gastant la memòria. Recordeu que tots els processos que s'han d'executar han d'estar en memòria principal i, si no n'hi ha, el procés pagarà en *swap* però amb la consegüent 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 detalladament, utilitzeu el `vmstat 2` (o `l'sar`), per exemple, i observeu les columnes `swpd`, que és la quantitat de memòria virtual (*swap*) 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 *swap* (`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, s'haurien d'eliminar processos de la memòria per a fer lloc o ampliar la memòria RAM del sistema si és que no es poden treure els processos que hi ha, sempre que el procés sota estudi no sigui d'execució ocasional.

3) Considereu que $\%CPU + \%E/S + \%Idle = 100\%$ per la qual cosa veiem que l'E/S (I/O en `vmstat`) també afecta un procés.

```
root@srv:~# vmstat 2
procs -----memory----- --swap-- -----io----- -system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo       in   cs  us   sy  id  wa  st
 0  1     0 686100 30136 141168    0    0   105    2       25   50   0    0  99   1   0
 1  0     0 668456 43808 140900    0    0  6836   912      2077 5059   2   15  69  14   0
```

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

4) S'ha de 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 desitja 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 ens dóna

```
real 0m4.584s
user 0m0.096s
sys  0m0.156s
```

en canvi, si fem `time ls -R /usr` la sortida és

```
real 0m3.325s
user 0m0.232s
sys  0m0.248s
```

Com veiem, per a obtenir informació equivalent (llista d'arxius i directoris) l'ordre `ls` és més eficient a l'espai d'usuari que l'ordre `find` però no passa el mateix a l'espai de nucli, per la qual cosa és interessant analitzar quines ordres escollim per a fer el treball. Un altre aspecte interessant és que si executem, p. ex., `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 passat? El sistema ha emmagatzemat en les taules de cau la informació i les següents vegades ja no triga el mateix, sinó molt menys. Si es desitja utilitzar el `time` en el format avançat o estès, els usuaris que executin *bash* com *shell* hauran d'executar el `time` juntament amb el *path* on es trobi; per exemple `/usr/bin/time ls -R /usr`, per a obtenir els resultats desitjats (consulteu `man time` per a més informació).

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

```
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
int main(void)
{int x=1, y=2, z=3; long iter1=0,iter2=0;
struct timeval tv1,tv2;
gettimeofday(&tv1,NULL);
for(;;){
    x=(x*3+y*7+z*9)%11;
    y=(x*9+y*11+z*3)%29;
    z=(x*17+y*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("Iteracions: %ldM Resultat: %d %d %d\n",iter2,x,y,z);
    return 0;
}
```

El resultat de l'execució és

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

on es pot observar que els 3,240 s'han aconseguit per a 22 milions d'iteracions. En què es gasten els 3,240 s? Doncs a calcular l'hora en cada iteració, ja que són múltiples les 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 puja el nombre d'iteracions (135 milions), tenint en compte que hem passat de 22 milions d'execució de la funció `gettimeofday` a 135 milions de vegades. Quina n'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 fer servir 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`. [Die]

1.3. Monitoratge

Un aspecte important en el funcionament 24x7 d'un sistema és que l'administrador s'ha d'anticipar als problemes i és per això que o bé està contínuament mirant el seu funcionament (la qual cosa és pràcticament impossible tot el temps) o bé disposa d'eines adequades que puguin prevenir la situació, generar alertes i advertir el responsable que "alguna cosa està passant" perquè aquest pugui fer 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 altres que l'usuari desitgi. Les eines més comunes permeten, mitjançant una interfície web, conèixer de manera remota 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 de la situació l'administrador. A continuació, es descriuran algunes de les eines més representatives (però no són les úniques): Munin, Monit, MRTG, Nagios, Ganglia, Zabbix i Cacti.

1.3.1. Munin

Munin [Mun] 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 `sysdw.nteum.org` i que ja el tenim configurat amb aquest nom i amb el DocumentRoot d'Apache en `/var/www/html`. Per a instal·lar Munin sobre Debian fem, per exemple, `apt-get install munin munin-node`. Després hem de configurar Munin (`/etc/munin/munin.conf`) amb:

```
dbdir /var/lib/munin
htmldir /var/cache/munin/www
logdir /var/log/munin
rundir /var/run/munin

tmpldir /etc/munin/templates

[srv.nteum.org]
address 127.0.0.1
use_node_name yes
```

Per a configurar la interacció amb Apache i permetre que es pugui connectar no solament del `localhost`, podem accedir a `/etc/apache2/conf-enabled/munin.conf` i canviar `Require local` per

```
Require all granted
Options FollowSymLinks
SymLinksIfOwnerMatch
```

(per exemple, fent `#Require local` i afegint les dues línies anteriors en els dos llocs que està aquesta sentència). A continuació s'ha de reiniciar Apache2 amb

`service apache2 restart` i **Munin amb** `service munin-node restart`. Amb això podrem connectar-nos a la pàgina (<http://srv.nteum.org/munin/> o a <http://localhost/munin/>) i veurem (després d'uns minuts) els gràfics de Munin que ens mostren l'activitat dels serveis configurats.

Si es desitja mantenir la confidencialitat de les gràfiques, es pot reemplaçar en l'arxiu de configuració d'Apache2 (abans indicat) la sentència *Require all granted*, per exemple, posant un comentari amb:

```
AuthUserFile /etc/munin/htpasswd AuthName "Munin"
AuthType Basic Require valid-user
```

Després, s'ha de crear l'arxiu de contrasenya en `/etc/munin/htpasswd` amb l'ordre (com a *root*): `htpasswd -c /etc/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.

Munin ve amb un conjunt de *plugins* instal·lats però fàcilment se'n poden habilitar uns altres fent, per exemple per a monitorar MySQL:

```
cd /etc/munin/plugins
ln -s /usr/share/munin/plugins/mysql_mysql_
ln -s /usr/share/munin/plugins/mysql_bytes mysql_bytes
ln -s /usr/share/munin/plugins/mysql_innodb mysql_innodb
ln -s /usr/share/munin/plugins/mysql_isam_space_ mysql_isam_space_
ln -s /usr/share/munin/plugins/mysql_queries mysql_queries
ln -s /usr/share/munin/plugins/mysql_slowqueries mysql_slowqueries
ln -s /usr/share/munin/plugins/mysql_threads mysql_threads
```

Es pot utilitzar l'ordre `munin-node-configuri --suggest` per veure els mòduls que disposa l'eina i el seu estat o les observacions que hi ha en relació a cadascun d'ells: la columna *used* mostra si està habilitat i la de *Suggestions*, si el servidor té el servei que monitoriza aquest mòdul. Per activar un mòdul solament s'ha de crear un enllaç, per exemple, com s'ha mostrat anteriorment per activar els mòduls de MySQL (després no cal oblidar de reiniciar Munin amb `service munin-node restart`). La figura 3 mostra algunes de les gràfiques d'activitat de Munin a través del navegador a <http://srv.nteum.org/munin/>.

1.3.2. Monit

Monit [Mon] permet configurar i verificar la disponibilitat de serveis com ara Apache, MySQL o Postfix i fa diferents accions, 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 s'haurà de consultar la documentació per a obtenir més informació*. A continuació, presentem un exemple de configuració típic sobre alguns serveis `/etc/monit/monitrc` [MonD]:

```
# Monit control file example: /etc/monit/monitrc
# Només es mostren les línies canviades.
```

*<http://mmonit.com/monit>

Figura 3



```

set daemon 120 # Poll at 2-minute intervals
set logfile /var/log/monit.log
set alert adminp@srv.nteum.org
# S'utilitza el servidor intern que disposa monit per a controlar apache2 també
set httpd port 2812 and
  use address localhost # only accept connection from localhost
  allow admin:monit # require user 'admin' with password 'monit'
# Exemples de monitors
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 process apache with pidfile /var/run/apache2/apache2.pid
  group www-data
  start program = "/etc/init.d/apache2 start"
  stop program = "/etc/init.d/apache2 stop"
  if failed host srv.nteum.org port 80 protocol http
    and request "/monit/token" then restart
  if cpu is greater than 60% for 2 cycles then alert
  if cpu > 80% for 5 cycles then restart
check process ntpd with pidfile /var/run/ntpd.pid
  start program = "/etc/init.d/ntp start"
  stop program = "/etc/init.d/ntp stop"
  if failed host 127.0.0.1 port 123 type udp then restart
  if 5 restarts within 5 cycles then timeout

```

Enllaç d'interès

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

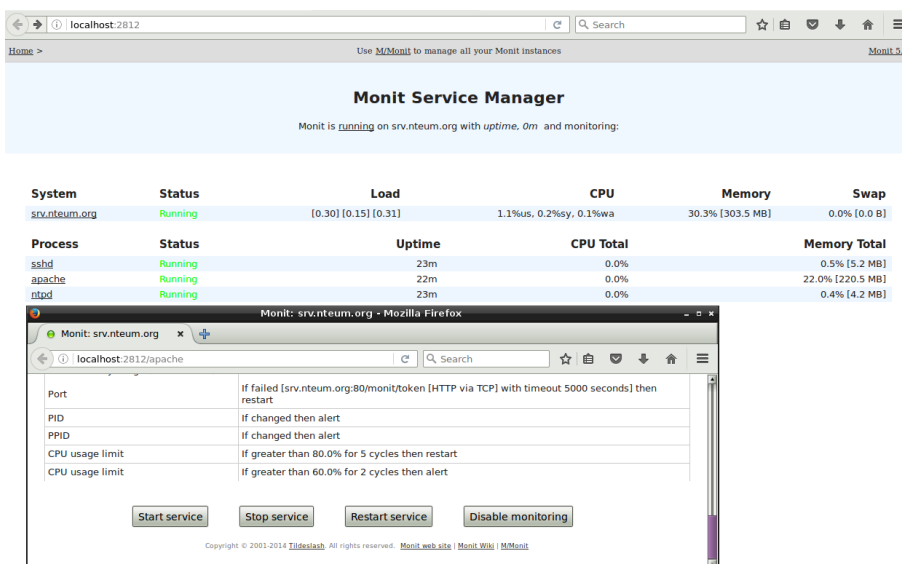
Per al monitoratge d'Apache, hem indicat que verifiqui un fitxer que haurà d'estar en `/var/www/monit/token`, per la qual cosa s'haurà de crear amb:

```
mkdir /var/www/monit; echo "hello" > /var/www/monit/token
```


Per a verificar que la sintaxi és correcta, executem `monit -t` i per a engegar-lo executem `monit`. A partir d'aquest moment, es pot consultar en l'adreça i port seleccionat en l'arxiu `/etc/monit/monitrc` (en el nostre cas en l'adreça `http://localhost:2812/`), que ens demanarà l'usuari i la contrasenya també introduïts en el mateix arxiu (`admin` i `monit` en el nostre cas). Es pot parar i arrencar el servei amb `service monit restart` (verifiquem el valor de la variable `START` en `/etc/default/monit`).

Monit permet gestionar els *daemons* d'acord a la configuració que se li ha indicat a `/etc/monit/monitrc` i això ho veurem com a botons dins de la informació de cada servei. La figura 4 mostra la pantalla principal de Monit i, com a exemple, els controls per a la gestió d'Apache2.

Figura 4



Enllaç d'interès

Podeu consultar la pàgina [MonitEx] per a obtenir exemples de configuració de serveis, dispositius, monitors HW, etc.

1.3.3. SNMP + MRTG

El MRTG (*Multi-Router Traffic Grapher*) [M] va ser creat per a mostrar informació gràfica sobre dades de xarxa, com es veurà en l'exemple que mostrarem a continuació, per a monitorar la xarxa Ethernet, però es poden usar altres dades per a visualitzar el comportament i per a generar les estadístiques de càrrega (*load average*) del servidor. En primer lloc, instal·larem SNMP o Protocol Simple d'Administració de Xarxa (*Simple Network Management Protocol*), que és un protocol de la capa d'aplicació que facilita l'obtenció d'informació entre dispositius de xarxa (p. ex., *routers*, *switches*, servidors, estacions de treball, impresores, etc.) i que permet als administradors supervisar el funcionament de la xarxa i buscar/resoldre els seus problemes. Per a això fem `apt-get install snmp snmpd`. Les variables accessibles a través de SNMP estan organitzades en jerarquies de metadades (tipus, descripció, etc.) i emmagatzemades en unes taules anomenades *Management Information Bases* (MIB). Per a instal·lar-les, hem d'afegir primer el repositori de Debian en non-free i després descarregar el paquet:

Afegim el repositori *non-free* de la distribució (al que tingui el *main*), per exemple, deb <http://ftp.uk.debian.org/debian/> jessie main contrib non-free

```
Després actualitzem els repositoris i instal·lem el paquet
apt-get update
apt-get install snmp snmpd snmp-mibs-downloader
download-mibs
```

Després hem de configurar el servei `snmpd` i per a això editem la configuració `/etc/snmp/snmpd.conf` (només hem deixat les línies més importants per a canviar):

```
agentAddress udp:127.0.0.1:161
rocommunity public
com2sec local localhost public
group MyRWGroup v1 local
group MyRWGroup v2c local
group MyRWGroup usm local
view all included .1 80
access MyRWGroup ""any noauth exact all all none
com2sec notConfigUser default mrtg
group notConfigGroup v1 notConfigUser
group notConfigGroup v2c notConfigUser
view systemview included .1.3.6.1.2.1.1
view systemview included .1.3.6.1.2.1.25.1.1
view systemview included .1 80
access notConfigGroup ""any noauth exact systemview none none
syslocation BCN
syscontact Adminp <adminp@srv.nteum.org>
```

A continuació, en l'arxiu `/etc/defaults/snmpd` hem modificat la línia `export MIBS=/usr/share/mibs` per a indicar-li on estaven les MIB i es reinicia el servei (`/etc/init.d/snmpd restart` o `systemctl restart snmpd`). Podem interrogar al servidor `snmpd` utilitzant l'ordre `snmpwalk`, per exemple:

```
snmpwalk -v1 -c public localhost          Donarà una llarga llista d'informació
snmpwalk -v 2c -c public localhost        Igual que l'anterior
```

O preguntar-li per una variable específica de la MIB:

```
snmpwalk -v1 -c mrtg localhost IP-MIB::ipAdEntIfIndex
IP-MIB::ipAdEntIfIndex.127.0.0.1 = INTEGER: 1
IP-MIB::ipAdEntIfIndex.158.109.65.67 = INTEGER: 2
```

Amb això, ja podem instal·lar MRTG fent

```
apt-get install mrtg mrtg-contrib mrtgutils mrtgutils-sensors
```

Després generem la configuració amb

```
mkdir /etc/mrtg
/usr/bin/cfgmaker --output=/etc/mrtg/traffic.cfg --ifdesc=ip --ifref=descr
--global "Options[_]: bits,growright" public@localhost
vi /etc/mrtg/traffic.cfg          Canviar la línia WorkDir per WorkDir: /var/www/html/mrtg
```

i hem de crear el directori i canviar les proteccions per al grup d'Apache:

```
mkdir /var/www/html/mrtg; chown www-data:www.data /var/www/html/mrtg.
```

Finalment, haurem de crear l'`index.html` amb:

```
indexmaker --title="srv.nteum.org" --output /var/www/html/mrtg/index.html
/etc/mrtg/traffic.cfg.
```

Per a executar `mrtg` inicialment i verificar si tot està bé, hauríem de fer: `env LANG=C /usr/bin/mrtg /etc/mrtg/traffic.cfg` (i repetint-la una sèrie de vegades). Podrem visualitzar l'activitat de xarxa accedint des del navegador a l'URL: <http://srv.nteum.org/mrtg/> i podrem veure que comença a mostrar l'activitat de les targetes de xarxa.

Finalment, si està tot bé haurem de configurar el `cron` perquè s'actualitzin les gràfiques cada 5', per a això editem el `crontab` -e inserint

```
*/5 * * * * root env LANG=C /usr/bin/mrtg /etc/mrtg/traffic.cfg.
```

MRTG és molt potent per a visualitzar gràfics de variables ja sigui SNMP o de *scripts* que podem incloure. Vegem uns exemples de com podem analitzar la càrrega de CPU, la memòria, *Swap* i *Round Trip* a través de diferents opcions. Per a això editarem cada arxiu a `/etc/mrtg`:

1) vi `/etc/mrtg/cpu.cfg` (la línia de `Target` ha d'anar tot en una línia)

```
WorkDir: /var/www/html/mrtg
Target[localhost.cpu]:ssCpuRawUser.0&ssCpuRawUser.0:public@localhost
    + ssCpuRawSystem.0&ssCpuRawSystem.0:public@localhost
    + ssCpuRawNice.0&ssCpuRawNice.0:public@localhost
RouterUptime[localhost.cpu]: public@localhost
MaxBytes[localhost.cpu]: 100
Title[localhost.cpu]: CPU Load
PageTop[localhost.cpu]: <h1>Active CPU Load %</h1>
Unscaled[localhost.cpu]: ymwd
ShortLegend[localhost.cpu]: %
YLegend[localhost.cpu]: CPU Utilization
Legend1[localhost.cpu]: Active CPU in % (Load)
Legend2[localhost.cpu]:
Legend3[localhost.cpu]:
Legend4[localhost.cpu]:
LegendI[localhost.cpu]: Active
LegendO[localhost.cpu]:
Options[localhost.cpu]: growright,nopercent
```

2) vi `/etc/mrtg/mem.cfg` (la línia de `Target` ha d'anar tot en una línia)

```
LoadMIBs: /usr/share/snmp/mibs/HOST-RESOURCES-MIB.txt
Target[localhost.mem]: .1.3.6.1.4.1.2021.4.11.0&.1.3.6.1.4.1.2021.4.11.0:public@localhost
```

```
PageTop[localhost.mem]: <H1>Free Memory </H1>
WorkDir: /var/www/html/mrtg
Options[localhost.mem]: nopercent,growright,gauge,noinfo
Title[localhost.mem]: Free Memory
MaxBytes[localhost.mem]: 1000000
kMG[localhost.mem]: k,M,G,T,P,X
YLegend[localhost.mem]: bytes
ShortLegend[localhost.mem]: bytes
LegendI[localhost.mem]: Free Memory:
LegendO[localhost.mem]:
Legend1[localhost.mem]: Free memory, not including swap, in bytes
```

3) vi /etc/mrtg/swap.cfg

```
LoadMIBs: /usr/share/snmp/mibs/UCD-SNMP-MIB.txt
Target[localhost.swap]: memAvailSwap.0&memAvailSwap.0:public@localhost
PageTop[localhost.swap]: <H1>Swap Memory</H1>
WorkDir: /var/www/html/mrtg
Options[localhost.swap]: nopercent,growright,gauge,noinfo
Title[localhost.swap]: Free Memory
MaxBytes[localhost.swap]: 1000000
kMG[localhost.swap]: k,M,G,T,P,X
YLegend[localhost.swap]: bytes
ShortLegend[localhost.swap]: bytes
LegendI[localhost.swap]: Free Memory:
LegendO[localhost.swap]:
Legend1[localhost.swap]: Swap memory avail, in bytes
```

4) vi /etc/mrtg/ping.cfg

```
WorkDir: /var/www/html/mrtg
Title[localhost.ping]: Round Trip Time
PageTop[localhost.ping]: <H1>Round Trip Time</H1>
Target[localhost.ping]: `/etc/mrtg/ping.sh`
MaxBytes[localhost.ping]: 2000
Options[localhost.ping]: growright,unknaszero,nopercent,gauge
LegendI[localhost.ping]: Pkt loss %
LegendO[localhost.ping]: Avg RTT
Legend1[localhost.ping]: Maximum Round Trip Time in ms
Legend2[localhost.ping]: Minimum Round Trip Time in ms
YLegend[localhost.ping]: RTT (ms)
```

5) vi /etc/mrtg/ping.sh

```
#!/bin/sh
PING="/bin/ping"
# Google, for example
ADDR="google.com"
```

```
DATA=`$PING -c10 -s500 $ADDR -q `
LOSS=`echo $DATA | awk '{print $18 }' | tr -d %`
echo $LOSS
if [ $LOSS = 100 ];
then
    echo 0
else
    echo $DATA | awk -F/ '{print $5 }'
fi
```

chmod 755 /etc/mrtg/ping.sh

Creem l'índex:

```
/usr/bin/indexmaker --output=/var/www/html/mrtg/index.html --title="srv.nteum.org"
--sort=name --enumerate /etc/mrtg/traffic.cfg /etc/mrtg/cpu.cfg
/etc/mrtg/mem.cfg /etc/mrtg/swap.cfg /etc/mrtg/ping.cfg
```

Es pot executar cada *script* individualment (unes quantes vegades) com (substituir *script.cfg* pel que es desitgi executar):

```
env LANG=C /usr/bin/mrtg /etc/mrtg/script.cfg,
```

I després podrem veure-ho a la pàgina <http://srv.nteum.org/mrtg/>. Finalment, per a que quedin en automàtic els haurem d'incloure a `crontab -e` com:

```
*/5 * * * * root env LANG=C /usr/bin/mrtg /etc/mrtg/traffic.cfg
*/5 * * * * root env LANG=C /usr/bin/mrtg /etc/mrtg/cpu.cfg
*/5 * * * * root env LANG=C /usr/bin/mrtg /etc/mrtg/mem.cfg
*/5 * * * * root env LANG=C /usr/bin/mrtg /etc/mrtg/swap.cfg
*/5 * * * * root env LANG=C /usr/bin/mrtg /etc/mrtg/ping.cfg
```

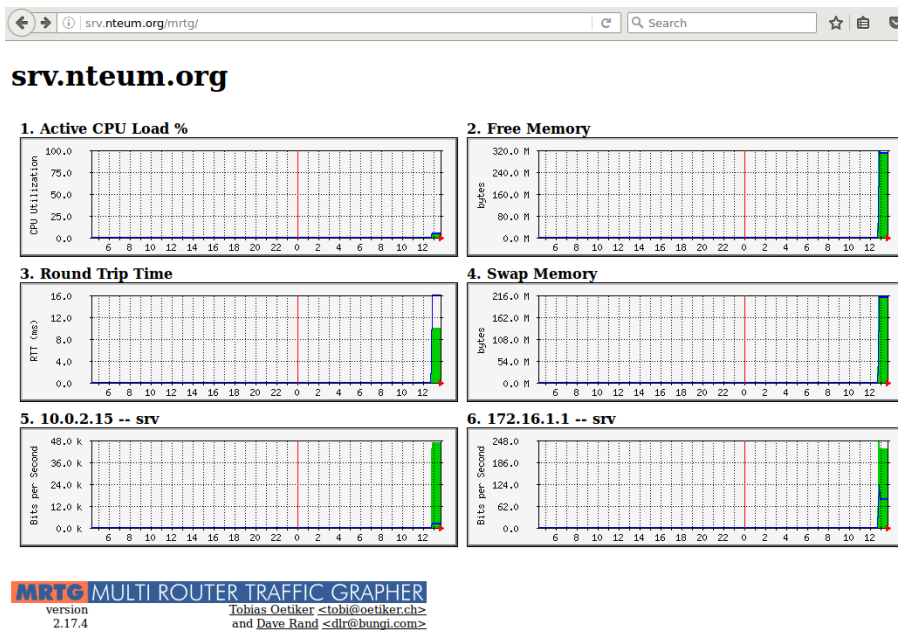
En les següents referències [SNMP, DV] es pot trobar més informació sobre SNMP (configuració, solució d'errors, etc).

La figura 5 mostra la sortida de MRTG amb el control de la càrrega de la CPU, la memòria, el *swap*, el *round trip* a Google.com i les dues targetes de xarxa que disposa el servidor. Si bé aquí s'han inclòs diferents gràfiques MRTG, s'utilitza habitualment per a dispositius de xarxa.

1.3.4. Nagios

Nagios és un sistema de monitoratge de xarxes i sistemes àmpliament utilitzat per la seva versatilitat i flexibilitat, sobretot a l'hora de comunicar alertes de comportaments o tendències dels sistemes monitorats. Pot monitorar serveis de xarxa (SMTP, HTTP, SNMP, etc.) i recursos del sistema (càrrega del procesador, ús dels discos, memòria, estat dels ports, etc.) tant locals com remots

Figura 5



a través d'un *plugin* (anomenat NRPE) i es pot estendre la seva funcionalitat mitjançant aquests *plugins*. El producte base de Nagios és anomenat **Nagios Core** el qual és *Open Source*, i serveix de base a altres productes comercials de Nagios com NagiosXI, IM i NA. Nagios té una comunitat molt activa (anomenada Nagios Exchange) i a la pàgina web de Nagios (<http://www.nagios.org/>) es pot accedir a les contribucions, *plugins* i documentació. Nagios permet consultar pràcticament qualsevol paràmetre d'interès d'un sistema, i genera alertes, que poden ser rebudes pels responsables corresponents mitjançant diferents canals com per exemple correu electrònic i missatges SMS. La instal·lació bàsica és molt simple, fent `apt-get install nagios3`, que instal·larà totes les biblioteques i *plugins* per a una primera configuració. Durant la instal·lació se'ns sol·licitarà un *passwd*, però també després es pot canviar fent `cd /etc/nagios3; htpasswd htpasswd.users nagiosadmin`. Després podrem recarregar novament Apache2 i amb el navegador connectant a l'URL <http://srv.nteum.org/nagios3/> se sol·licitarà l'accés com a *nagiosadmin* i el *passwd* que hem introduït i podrem veure l'estructura de Nagios i observar els serveis monitorats en cadascun dels apartats.

L'arxiu de configuració de Nagios està en `/etc/nagios3/nagios.cfg`, el qual inclou tots els arxius del directori `conf.d` del mateix directori. En aquests tindrem agrupats per arxius els diferents aspectes que cal monitorar, per exemple sistemes (`localhost_nagios2.cfg`) i serveis (`services_nagios2.cfg`). El fitxer més important probablement és `localhost_nagios2.cfg`, del qual farem una breu descripció:

```
# definició d'un host utilitzant un template (generic-host)
define host{
    use                generic-host
    host_name          localhost
    alias              localhost
```

```

        address                127.0.0.1
    }
# definició d'un servei -espai de disc- utilitzant un template
# (generic-service) executant un cmd i amb advertiment al 20% i error al 10%
# de l'espai lliure.
define service{
    use                generic-service
    host_name          localhost
    service_description Disk Space
    check_command      check_all_disks!20%!10%
}
# ídem per a usuaris: advertiment 20 usuaris, crític 50 usuaris
define service{
    use                generic-service
    host_name          localhost
    service_description Current Users
    check_command      check_users!20!50
}
#ídem processos:advertència 250, crític 400
define service{
    use                generic-service
    host_name          localhost
    service_description Total Processes
    check_command      check_procs!250!400
}
# Càrrega de CPU.
define service{
    use                generic-service
    host_name          localhost
    service_description Current Load
    check_command      check_load!5.0!4.0!3.0!10.0!6.0!4.0
}
    
```

Si volguéssim agregar un servei (per exemple ping al *localhost*), només l'hauríem d'agregar al final de l'arxiu:

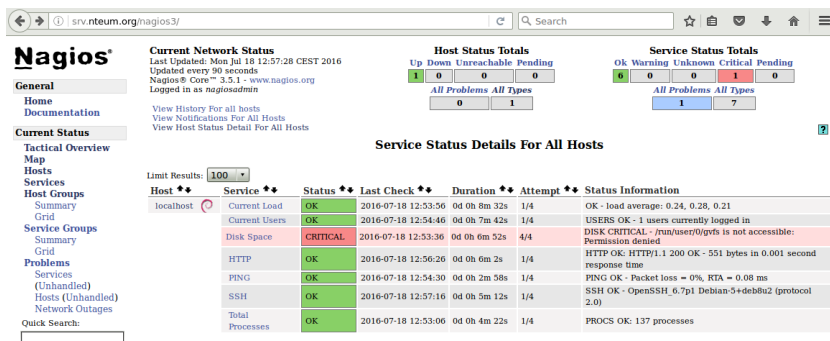
```

# Ping: advertiment quan el 20% sigui 100 ms, crític 60% sigui 500 ms.
define service{
    use                generic-service
    host               localhost
    service_description PING
    check_command      check_ping!100.0,20%!500.0,60%
}
    
```

Els *plugins* són incorporats per */etc/nagios3/nagios.cfg* i estan definits en l'arxiu */etc/nagios-plugins/config*, on es poden veure les diferents alternatives per a monitorar.

La figura 6 mostra els serveis monitoritzats sobre *localhost*.

Figura 6



Dos complements interessants per a Nagios són PNP4Nagios* i NagVis:

*<http://docs.pnp4nagios.org/pnp-0.6/start>

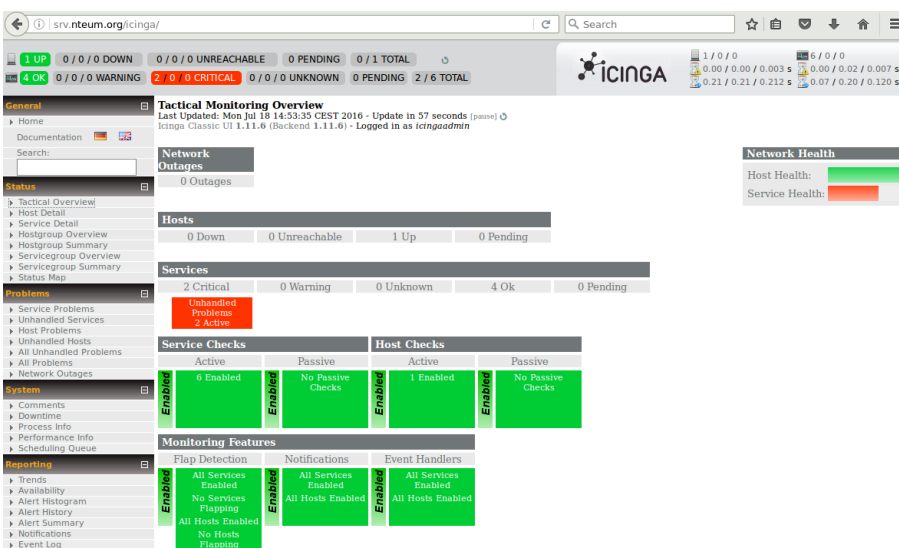
1) PNP4Nagios permet emmagatzemar les informacions que proveeixen els *plugins* en una base de dades anomenada *RRD-database* i cridar l'eina *RRD Tool* (<http://oss.oetiker.ch/rrdtool/>), que permet la visualització d'aquestes dades incrustades a la pàgina de Nagios. Per la qual cosa, a més del valor instantani de les variables que cal monitorar, també tindrem integrats aquests en el temps i podrem veure la seva evolució.

2) NavVis (<http://www.nagvis.org/>) permet dibuixar (secció map) la xarxa de diferents formes i aspectes per a tenir diverses visualitzacions de la xarxa monitorada en forma activa, és a dir, veient els valors de les variables seleccionades sobre el gràfic.

Finalment, és interessant considerar **Icinga** (<https://www.icinga.org/>), que és una bifurcació (*fork*) de Nagios (del 2009) i ha evolucionat molt últimament (per a alguns ja ha superat Nagios) i el seu objectiu és transformar-se en una eina de referència dins de l'*Open Source* i en l'àmbit del monitoratge de xarxes i sistemes. Icinga neix amb la idea de superar les deficiències en el procés de desenvolupament de Nagios i les seves polítiques, així com la voluntat de ser més dinàmica i fàcil per a agregar-hi noves característiques, com per exemple una interfície d'usuari d'estil Web 2.0, connectors de base de dades addicionals i una API REST que permeti als administradors integrar nombroses extensions sense complicades modificacions del nucli. Icinga està disponible en Debian i la seva configuració és pràcticament similar a Nagios.

La figura 7 mostra la pàgina del "Quadre de comandaments i Visió General" d'Icinga.

Figura 7



1.3.5. Ganglia

Ganglia [Ga] és una eina que permet monitorar de manera escalable i distribuïda l'estat d'un conjunt de màquines agrupades sota diferents criteris (xarxa, serveis, etc.) o simplement sota una mateixa identificació que anomenarem *clúster*. L'aplicació mostra a l'usuari les estadístiques de forma remota (per exemple, les mitjanes de càrrega de la CPU o la utilització de la xarxa) de totes les màquines que conformen aquest clúster basant-se en un disseny jeràrquic i utilitza comunicacions punt a punt o *multicast* per a l'intercanvi d'informació entre els diferents nodes que formen el clúster. Ganglia utilitza XML per a la representació de dades, XDR per al transport compacte i portàtil de dades i RRDtool (<http://oss.oetiker.ch/rrdtool/>) per a emmagatzematge de dades i visualització. El sistema es compon de dos *daemons* (**gmond** i **gmetad**), una pàgina de visualització (**ganglia-webfrontend**) basada en PHP.

Gmond és un *daemon* multifil que s'executa en cada node del clúster que es desitja supervisar (no és necessari tenir un sistema d'arxius NFS o base de dades ni mantenir comptes especials dels arxius de configuració). Les tasques de Gmond són monitorar els canvis d'estat en el *host*, enviar els canvis pertinents, escoltar l'estat d'altres nodes (a través d'un canal *unicast* o *multicast*) i respondre les peticions d'un XML de l'estat del clúster. La federació dels nodes es porta a terme amb un arbre de connexions punt a punt entre els nodes determinats (representatius) del clúster per a agregar l'estat dels restants nodes. En cada node de l'arbre, s'executa el *daemon* **Gmetad** que periòdicament sol·licita les dades dels restants nodes, analitza el XML, guarda tots els paràmetres numèrics i exporta el XML agregat per un *socket* TCP. Les fonts de dades poden ser *daemons* gmond, en representació de determinats grups, o altres *daemons* gmetad, en representació de conjunts de grups. Finalment, la web de Ganglia proporciona una vista de la informació recollida dels nodes del clúster en temps real. Per exemple, es pot veure la utilització de la CPU durant l'última hora, dia, setmana, mes o any i mostra gràfics similars per a l'ús de memòria, ús de disc, estadístiques de la xarxa, nombre de processos en execució i tots els altres indicadors de Ganglia.

Per a la instal·lació de Ganglia sobre Debian (és similar per a altres distribucions):

1) Cal fer la instal·lació dels paquets de Ganglia sobre el servidor web:

```
apt-get install ganglia-monitor gmetad ganglia-webfrontend.
```

2) Sobre tots els altres nodes, només es necessita tenir instal·lat el paquet `ganglia-monitor`.

3) Els arxius de configuració estan en `/etc/ganglia` i en `gmond.conf`, la línia més important és `data_source "my cluster" IP_nodo_Gmetad`, on indica quin serà el nom del clúster (per a agregar totes les màquines sota el mateix *tag*) i on es recolliran les dades (en aquest cas, en l'adreça IP de la màquina on es troba el `gmetad` que recollirà les dades). En `gmond.conf` tenim les

configuracions generals i de noms a més dels canals de comunicació que per defecte són *multicast*. Si desitgem que siguin *unicast* haurem de modificar les seccions `udp_send|recv` on la IP de host serà la del servidor (*gmetad*) i comentar les adreces de *multicast*:

```
udp_send_channel {
# mcast_join = 239.2.11.71
  host = IP_node_gmetad
  port = 8649
  ttl = 1
}

udp_recv_channel {
# mcast_join = 239.2.11.71
  port = 8649
# bind = 239.2.11.71
}
```

A continuació s'hauran de reiniciar els serveis com `service gmetad restart` i `service ganglia-monitor restart`

(o `systemctl restart ganglia-monitor; systemctl restart gmetad` als sistemes que disposin de *systemd*).

4) Finalment, hem de crear l'enllaç entre la configuració de Ganglia-frontend i Apache fent

`ln -s /etc/ganglia-webfrontend /etc/apache2/conf-enabled/ganglia.conf,`

reiniciem Apache (`service apache2 restart`) i després ens connectem a l'URL <http://srv.nteum.org/ganglia/> per a visualitzar una panoràmica global dels sistemes monitorats i fent clic en cada imatge/botó podem obtenir més informació sobre cadascun dels recursos monitorats.

La figura 8 mostra la interfície de Ganglia i l'estat del servidor a la pantalla principal.

Figura 8



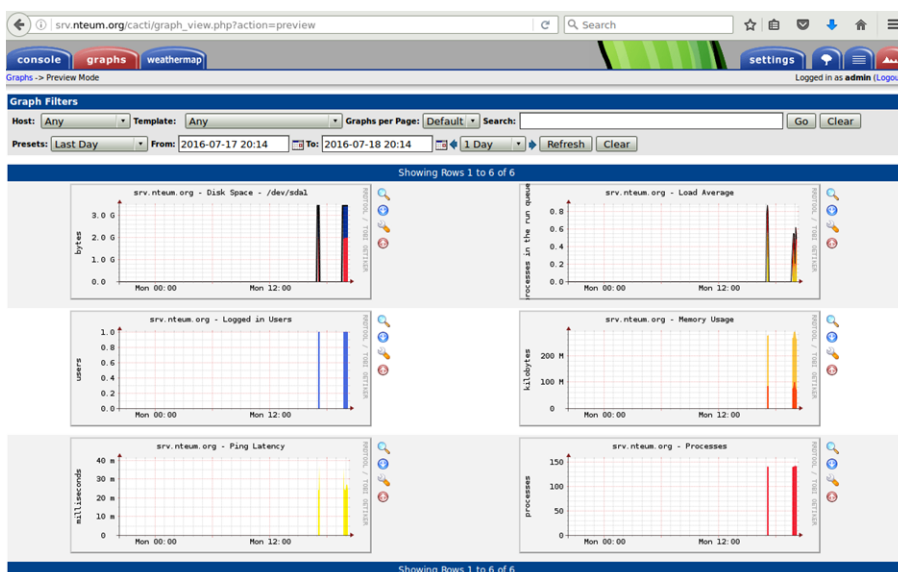
1.3.6. Cacti

Cacti [Cacti] és una solució per a la visualització d'estadístiques de xarxa. Va ser dissenyada per aprofitar el poder d'emmagatzematge i per la funcionalitat de generar gràfiques dinàmiques que posseeix RRDtool (utilitzades en Ganglia també). Aquesta eina, desenvolupada en PHP, proveeix diferents maneres de visualització i de gràfics avançats, i disposa d'una interfície d'usuari fàcil d'utilitzar. Tot això la fa interessant tant per a xarxes LAN com per a xarxes complexes amb centenars de dispositius.

La seva instal·lació és simple i requereix tenir instal·lat prèviament MySQL. Després, fent `apt-get install cacti` s'instal·larà el paquet. Al final, ens demanarà si volem instal·lar la interfície amb la base de dades, i ens sol·licitarà el nom d'usuari i la contrasenya de la mateixa base i la contrasenya de l'usuari *admin* de la interfície de Cacti*. Després, instal·larà la configuració en el lloc d'Apache2 (*/etc/apache2/conf-avaliable/cacti.conf*) i reiniciarà els servidors. Ara ja podem connectar-nos a l'URL <http://srv.nteum.org/cacti/>. Les primeres tasques que cal dur a terme és fer els ajustos pertinents de la instal·lació (ens mostrarà un *checklist*) i canviar la contrasenya per *admin*. A continuació, ens mostrarà una pantalla amb les pestanyes de gràfics i de consola on trobarem les diferents opcions per configurar i posar en marxa el monitoratge més adequat per al nostre lloc. Seguint el procediment indicat a http://docs.cacti.net/manual:088:2_basics.1_first_graph, serà molt fàcil incorporar els gràfics més adequats per monitoritzar la nostra instal·lació. És interessant incorporar a Cacti un *plugin* anomenat *weathermap* (<http://www.networkweathermap.com/>) que ens permetrà veure un diagrama de la infraestructura en temps real i els elements monitoritzats amb dades dinàmiques sobre el seu estat. La figura 9 mostra una sèrie de gràfics (espai de disc, càrrega, usuaris, memòria, latència i processos) generats en Cacti per al node actual.

*Si no la hi hem donat, contindrà la que defineix per defecte, que és usuari *admin* i *passwd admin*.

Figura 9



1.3.7. Altres eines

Altres paquets interessants que cal tenir en compte per a monitorar un sistema són els següents:

- **Zabbix [Za]** és un sistema de monitoratge (*Open Source*) que permet recollir l'estat de diferents serveis de xarxa, servidors i maquinari de xarxa. Aquest programa utilitza una base de dades (MySQL, PostgreSQL, SQLite, etc.) per a millorar les prestacions i permet la instal·lació d'agents Zabbix sobre diferents màquines per a monitorar aspectes interns com per exemple càrrega de CPU, utilització de xarxa, espai en disc, etc. També és possible fer aquest monitoratge mitjançant diferents protocols com SNMP, TCP i ICMP, IPMI, etc. i suporta una varietat de mecanismes de notificació en temps real, incloent XMPP. La seva instal·lació no està disponible en els paquets de Debian (per diferències en les polítiques de Debian i Zabbix des del 2012) però es pot obtenir el paquet (.deb) des del web de Zabbix i seguir la guia d'instal·lació*.
- **Collectd [Coll]**. És un *daemon* que recopila estadístiques sobre el sistema i les emmagatzema per a la seva posterior visualització. Aquestes estadístiques es poden utilitzar per trobar els colls d'ampolla de rendiment actuals i predir la càrrega del sistema futur. Collectd està escrit en C pel que el seu rendiment i portabilitat és molt alt i inclou optimitzacions i característiques per manejar grans volums de dades. A més incorpora més de 90 *plugins* i és un projecte ben documentat, actiu a la comunitat i amb ampli suport.
- **Shinken [Shi]**. Sistema de monitoratge (compatible amb Nagios) que permet gestionar alertes, visualitzar càrregues i serveis, controlar l'estat de diferents dispositius i analitzar el rendiment. Ofereix una arquitectura senzilla amb un equilibri de càrrega i suporta alta disponibilitat. L'administrador solament ha de gestionar una única configuració i el sistema la recompondrà automàticament i recaptarà tota la informació indicada de cada node supervisat. Va ser escrit com a prova de concepte d'una nova arquitectura de Nagios4 (més ràpid i més flexible), però va ser rebutjada pels autors de Nagios, per la qual cosa Shinken es va convertir en una aplicació de programari de monitoratge de xarxa independent i compatible amb Nagios.
- **XYMon [XY]**. Aquesta aplicació permet monitoritzar servidors i xarxes (la seva inspiració prové del *Big Brother monitoring system*). Utilitza una arquitectura molt eficient i permet reduir en forma considerable la càrrega tant si es tracta d'una petita xarxa de servidors com si es desitja analitzar una xarxa amb milers de servidors i serveis. La seva primera implementació va ser com un conjunt d'eines (*bbgen*) i està disponible des del 2002 com *add-on* de *Big Brother*. No obstant això, la seva evolució ha va permetre

*https://www.zabbix.com/documentation/2.0/manual/installation/install_from_packages

concebre-ho com una eina amb identitat pròpia moment en el qual se li va canviar el nom per Hobbit i, posteriorment en 2009, per el seu nom definitiu com Xymon.

- **PandoraFMS [Pa]**. Pandora FMS és un paquet de monitoratge per a la gestió integral de la infraestructura tecnològica d'una institució/empresa. Permet monitoritzar servidors i serveis de diferents característiques i sistemes operatius, així com infraestructures virtuals i dispositius que una organització pugui tenir. Pandora FMS està orientat a grans entorns i pot utilitzar-se amb o sense agents. És eficient perquè pugui ser utilitzat en clústers o grans agrupacions d'ordinadors permetent analitzar quan un servei no està operatiu, quan existeix un dispositiu que té problemes de memòria o el moviment de valors de llindar de determinades variables crítiques d'un conjunt de servidors. Pandora FMS està publicat sota llicència GPL2 i és *open source* encara que disposa d'una versió específica per a empreses, sota el model conegut com *OpenCore*.

Hi ha un conjunt addicional d'eines no menys interessants (no s'inclouen les ja esmentades) que incorpora GNU/Linux per al monitoratge de diferents aspectes del sistema (es recomana veure el `man` de cada eina per a més informació):

- **isag**: *Interactive System Activity Grapher*, per a l'auditoria de recursos hw/sw.
- **mon**: 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 (fsapy, gamin)**: *file alteration monitor*, monitor d'alteració de fitxers.
- **ksensors (lm-sensors)**: monitor de la placa base (temperatura, alimentació, ventiladors, etc.).
- **sysctl**: utilitat per a retirar capacitats assignades al nucli en el fitxer `/proc/sys/kernel`.
- **swatch**: monitor per a l'activitat del sistema mitjançant 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.

1.4. 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, a causa de 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 Red Hat), UltraMonkey (solució de VA Linux), o OpenAIS+Corosync+Pacemaker.

El projecte Linux-HA (Linux d'alta disponibilitat) [HA] és una solució clúster d'alta disponibilitat per a Linux i altres sistemes operatius, com FreeBSD, OpenBSD, Solaris i MacOSX i que proporciona fiabilitat, disponibilitat i prescripció discontinua de serveis. El producte principal del projecte és **Heartbeat**, l'objectiu principal del qual és la gestió de clústers amb l'objectiu d'obtenir alta disponibilitat. Les seves característiques més importants són les següents: il·limitat nombre 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 a poder ser útils als usuaris, el *daemon* de Heartbeat ha de combinar-se amb un administrador de recursos de clúster (CRM), que té la tasca d'iniciar i aturar els serveis (adreces IP, servidors web, etc.) la qual cosa 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|Corosync + Pacemaker [Pa].

1.4.1. Guia simple clúster amb Heartbeat per HA amb Apache

Un clúster HA per a un servei com Apache es pot construir molt fàcilment utilitzant només Heartbeat i sense utilitzar un CRM (després veurem com construir un clúster HA amb Pacemaker). És interessant, encara que una mica antic, l'article [30] i la documentació del lloc web de Linux-HA [19]. Els passos que cal seguir són:

- 1) Per a aquesta prova de concepte hem utilitzat dues màquines virtuals amb dos adaptadors cadascuna: *eth1* amb NAT per accedir a Internet i instal·lar-hi els paquets, i *eth0*, configurada com a *Host-Only* per tenir una xarxa compartida amb el *host*. Per crear una xarxa *Host-Only* sobre Virtualbox, anem a les preferències de VirtualBox i, en l'apartat *Networks*, creem un *host-only network* (en el nostre cas amb IP 172.16.1.100 i màscara 255.255.255.0), cosa que generarà una targeta de xarxa sobre el *host* que compartirà amb ell (des d'on cri-

darem el clúster d'HA) i les màquines virtuals (proveïdores del servei Apache). És important tenir en compte que, si Virtualbox (per a algunes versions) està sobre Windows 10, quan les màquines virtuals s'iniciïn sobre la xarxa *Host-Only*, donaran un error. Això se soluciona modificant des de Windows 10 -> *Network & Sharing Center* -> *adaptin Settings* l'adaptador creat (alguna cosa com ara *VirtualBox Host-Only Network #1*), accedint a les propietats de l'adaptador i activant el *VirtualBox NDIS6 Bridged Network Drive*.

2) Indicarem a cada màquina virtual que el segon adaptador (*eth1*) serà NAT i el primer (*eth0*), *Host-Only*, i seleccionarem la targeta (virtual) que hem creat anteriorment. Després, quan s'iniciïn, els canviarem el nom a */etc/hostname* (nosaltres hem posat *ha1* i *ha2* respectivament), i les IP d'*eth0* (*eth1* les deixem per DHCP) amb una IP estàtica dins de la xarxa que hem creat abans. En el nostre cas *ha1 = 172.16.1.1* i *ha2 = 172.16.1.2*.

3) Modifiquem el */etc/hosts* de les dues màquines perquè quedi:

```
127.0.0.1    localhost
172.16.1.1  ha1.nteum.org  ha1
172.16.1.2  ha2.nteum.org  ha2
172.16.1.10 ha.nteum.org  ha
```

L'IP 172.16.1.10 serà l'"IP flotant" on es prestarà el servei que anirà o bé al node *ha1* o bé al node *ha2*.

Canviem a */etc/ssh/sshd.conf* el valor *PermitRootLogin yes* per, després, poder copiar arxius entre els nodes com a *root*. Reiniciem i verifiquem que podem fer *ping* entre les dues màquines virtuals i des del *host*. Cada màquina ha de tenir instal·lat Apache i SSH-Server. Si no ho estan, cal instal·lar-los i verificar que ens podem connectar a aquests serveis individualment (és recomanable canviar el */var/www/html/index.html* de cada màquina virtual per posar un identificador que ens permeti saber si estem a *ha1* o *ha2*).

4) Sobre cada màquina virtual aturem Apache (`service apache2 stop`), modifiquem la configuració d'Apache afegint a */etc/apache2/ports.conf* la línia *NameVirtualHost 172.16.1.10:80* (deixant la resta com estan), que serà la IP virtual (flotant) que atindrà el servei donat per Heartbeat. Després, per tal que Apache no estigui iniciat des del principi, sinó que el controli Heartbeat, hem de treure'l de l'inici amb

```
update-rc.d apache2 remove o bien systemctl disable apache2.
```

5) Instal·lem Heratbeat i `chkconfig` en cada node (per fer-ho executem la instrucció `apt-get install heartbeat chkconfig`).

6) Sobre *ha1* editem l'arxiu `vi /etc/ha.d/ha.cf` i inserim el següent contingut:

```
logfile /var/log/cluster.log
logfacility local0
warntime 2
deadtime 30
initdead 120
keepalive 2
bcast eth1
udpport 694
```

```
auto_failback on
node ha1
node ha2
```

Aquí, *logfile* i *logfacility* indiquen on estarà l'arxiu de *log* i el nivell de missatges que volem; *wartime* és el temps que transcorrerà abans que ens avisi; *deadtime*, el temps després del qual es confirmarà que un node ha caigut; *initdead*, el temps màxim que esperarà que un node arrenqui; *keepalive*, l'interval de temps per comprovar-ne la disponibilitat; *BCAST*, la manera de comunicació (*broadcast*) i la interfície, i *node*, els nodes que formen el nostre clúster HA.

7) Editem l'arxiu `vi /etc/ha.d/authkeys` i inserim:

```
auth 2
2 sha1 passwd
```

És l'arxiu d'autenticació entre la comunicació dels nodes (podem reemplaçar *passwd* amb un de desitjat) i ha de ser només propietari el *root* (executarem `chmod 600 /etc/ha.d/authkeys`).

8) Després haurem d'editar `vi /etc/ha.d/haresources`:

```
ha1 172.16.1.10/24/eth0 apache2
```

On indiquem el node primari, la IP flotant, la màscara i la interfície, així com el servei que cal posar en marxa (Apache2 en el nostre cas).

9) Finalment, haurem de propagar aquests arxius a cada node del clúster executant `/usr/share/heartbeat/ha_propagate`, i també haurem de copiar en cadascun d'ells el fitxer *haresources* (p. ex., `scp /etc/ha.d/haresources ha2: /etc/ha.d/haresources`). Reiniciem tots els nodes i verifiquem que no tenim errors a `/var/log/cluster.log`.

10) Per provar el servei podem anar al *host* i, des d'un navegador, posem com a URL 172.16.1.10. Ens haurà de sortir *index.html* del node primari (*ha1*). Després, si hi apliquem `ifdown eth0` i reiniciem la pàgina, ens haurà de sortir l'*index.html* del secundari (*ha2*). L'ordre `cl_status` ens donarà informació sobre els nodes i l'estat d'aquest clúster HA (simple).

1.4.2. Clúster HA amb Corosync+Pacemaker y Nginx

A la pàgina de Linux-HA [HPac] es pot veure com integrar Heartbeat i Pacemaker com a CRM, però, com s'indica a Debian-HA [D-HA], els esforços estan orientats a evolucionar amb *Pacemaker / Corosync HA cluster stack* (tant per les evolucions d'aquests paquets com per l'obsolescència de Heartbeat). Malauradament, aquests paquets no estan integrats en el repositori de Debian Jessie (ja que no es va arribar a temps amb el desenvolupament per a aquesta versió), però es troben en el repositori de *backports*, tal com s'explica a [D-CFS].

Pacemaker és un gestor de recursos molt potent que utilitza CoroSyc (o Heartbeat) i que no té les limitacions dels antics CRM. La infraestructura i les seves prestacions es poden obtenir de la pàgina principal del projecte [Pacem]. Com a prova de concepte per treballar amb l'estructura de CoroSyc

+ Pacemaker a Debianm, utilitzarem dues màquines (iguals que aquelles de què disposàvem anteriorment i connectades a una xarxa 172.16.1.0, i amb les mateixes IP-noms que en el subapartat anterior). És convenient, si s'utilitzen les mateixes màquines del subapartat anterior, desinstal·lar tots els paquets (`apt-get remove -purge hearbeat`) per evitar incompatibilitats amb algunes llibreries.

- 1) Afegim `deb http://http.debian.net/debian Jessie-backports main` i realitzem un `apt-get update` (recordeu que els paquets són al repositori *backports*).
- 2) Instal·lem en tots dos nodes `apt-get install -t Jessie-backports Pacemaker crmsh` (pot donar alguns errors, per exemple, *openhpid*, *corosync*, *Pacemaker...*, ja que encara no està configurat).
- 3) Instal·lem en tots dos nodes `apt-get install Nginx`, canviant la pàgina principal per saber quin servidor presta el servei per deshabilitar amb `systemctl disable Nginx` (el nostre objectiu és que el controli Pacemaker).
- 4) Editem `vi /etc/corosync/corosync.conf` amb un contingut similar al següent:

```
totem {
    version: 2
    cluster_name: debian
    token: 3000
    token_retransmits_before_loss_const: 10
    clear_node_high_bit: yes
    crypto_cipher: none
    crypto_hash: none
    interface {
        member {
            memberaddr: 172.16.1.1
        }
        member {
            memberaddr: 172.16.1.2
        }
        ringnumber: 0
        bindnetaddr: 172.16.1.0
        mcastaddr: 239.255.1.1
        mcastport: 5405
        ttl: 1
    }
}
logging {
    fileline: off
    to_stderr: no
    to_logfile: no
    to_syslog: yes
    syslog_facility: daemon
    debug: off
    timestamp: on
    logger_subsys {
        subsys: QUORUM
        debug: off
    }
}
quorum {
    provider: corosync_votequorum
    two_node: 1
    expected_votes: 2
}
```

Copiem l'altre node en la mateixa posició. Després, com es mostra a la figura 10, podrem reiniciar els serveis i veure l'estat amb

```
service corosync restart; service pacemaker restart; crm status
```

Figura 10

```
root@ha1:/etc/corosync# crm status
Last updated: Tue Jul 19 18:00:25 2016          Last change: Tue Jul 19 17:31:14
2016 by root via crm_resource on ha2
Stack: corosync
Current DC: ha2 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 0 resources configured

Online: [ ha1 ha2 ]
```

crm

És interessant veure les opcions de l'ordre `crm`, que es pot executar de manera interactiva i amb un ajut explícit dins la mateixa ordre.

5) Per configurar el servei, executarem l'ordre `crm configure i`, de manera interactiva, crearem els recursos i les propietats que ens posi el `prompt crm (live) configure#`:

```
property stonith-enabled=no
property no-quorum-policy=ignore
property default-resource-stickiness=100
primitive IP-nginx ocf:heartbeat:IPAddr2 \
    params ip="172.16.1.10" nic="eth0" cidr_netmask="24" \
    meta migration-threshold=2 \
    op monitor interval=20 timeout=60 on-fail=restart
primitive Nginx-rsc ocf:heartbeat:nginx \
    meta migration-threshold=2 \
    op monitor interval=20 timeout=60 on-fail=restart
colocation lb-loc inf: IP-nginx Nginx-rsc
order lb-ord inf: IP-nginx Nginx-rsc
commit
up
bye
```

La sortida a `crm status` serà ara la que es mostra a la figura 11.

Figura 11

```
root@ha1:/etc/corosync# crm status
Last updated: Tue Jul 19 18:11:49 2016          Last change: Tue Jul 19 17:31:14
2016 by root via crm_resource on ha2
Stack: corosync
Current DC: ha2 (version 1.1.14-70404b0) - partition with quorum
2 nodes and 2 resources configured

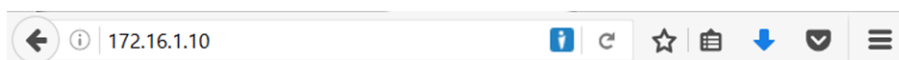
Online: [ ha1 ha2 ]

Full list of resources:

IP-nginx      (ocf::heartbeat:IPAddr2):      Started ha1
Nginx-rsc     (ocf::heartbeat:nginx):       Started ha1
```

Amb això, ja podem connectar-nos a una màquina dins de la mateixa xarxa (per exemple, si fem servir una xarxa *Host-Only* podrà ser el *host* i, en posar com a URL la IP virtual 172.16.1.10, veurem la pàgina de *ha1* tal com es mostra a la figura 12).

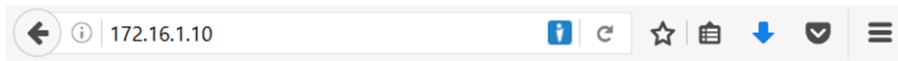
Figura 12



Welcome to nginx on Debian! HA1

Si fem `crm resource status IP-nginx`, ens donarà *resource IP-nginx is running on: ha1*. Si entrem a `crm resource` i fem `migrate IP-nginx` i, després, `status`, ens dirà que *resource IP-nginx is running on: ha2*. I, si tornem a carregar la pàgina, obtindrem el que es mostra a la figura 13.

Figura 13



Welcome to nginx on Debian! HA2

Per retornar a *ha1*, executarem dins de `crm resource:umigrate IP-nginx` per treure la restricció que el servei torni a *ha1*, en el cas que pugui ser inestable i, després, `migrate IP-nginx`. Si fem `status`, veurem que ha tornat a *ha1*, i carregarem la pàgina del primer servidor.

En aquesta configuració mínima no hem tingut en compte el Fence (consulteu [D-CfS]) que servirà per posar els nodes en un estat conegut (aspecte molt important d'un clúster), i per fer altres proves (que es poden seguir des de la mateixa referència) sobre caigudes del servei, caigudes del node i altres ordres útils per gestionar el clúster de HA. Una altra referència interessant per manejar recursos i gestionar-los és [GeekP], on presenten diferents opcions, funcionalitats i configuracions amb Pacemaker i crmsh.

1.4.3. DRBD

El **Distributed Replicated Block Device** (DRBD) és un emmagatzematge distribuït sobre múltiples *hosts* on, com en RAID1, les dades són replicades sobre el sistema d'arxiu sobre els altres *hosts* i sobre TCP/IP.[drbd] En aquesta prova de concepte, utilitzarem les mateixes màquines fetes servir anteriorment a les quals hem addicionat una unitat de disc més (*/dev/sdb*) i hem creat una partició sobre aquestes (*/dev/sdb1*). Per a la configuració de DRBD haurem de fer:

- 1) Instal·lar a les dues màquines DRBD `apt-get install drbd8-utils` on els arxius de configuració estaran en */etc/drbd.d/* (hi ha un arxiu */etc/drbd.conf* però que inclou els arxius del directori esmentat).
- 2) Crearem la configuració del recurs com */etc/drbd.d/demo.res* amb el contingut següent:

```
resource drbddemo {
    meta-disk internal;
    device /dev/drbd1;
    syncer {
        verify-alg sha1;
    }
}
```

```

    }
    net {
        allow-two-primaries;
    }
    on ha1 {
        disk /dev/sdb1;
        address 172.16.1.1:7789;
    }
    on ha2 {
        disk /dev/sdb1;
        address 172.16.1.2:7789;
    }
}

```

3) Copiem l'arxiu:

```
scp /etc/drbd.d/demo.res ha2:/etc/drbd.d/demo.res
```

4) Inicialitzem en els dos nodes el dispositiu executant `drbdadm create-md drbddemo`

5) Sobre ha1 (s'ha de verificar amb `uname -n`) fem `modprobe drbd` per a carregar el mòdul de kernel, després `drbdadm up drbddemo` per a aixecar el dispositiu i el podrem mirar amb `cat /proc/drbd` que no indicarà (entre altres missatges) que està:

```
cs:WfConnection ro:Secondary/Unknown ds:Inconsistent/DUnknown C r—
```

6) Sobre ha2 fem servir l'ordre `modprobe drbd` per a carregar el mòdul del kernel, `drbdadm up drbddemo` per a inicialitzar el dispositiu i

```
drbdadm ---overwrite-data-of-peer primary drbddemo
```

per a configurar-lo com a primari. Si mirem la configuració amb la instrucció `cat /proc/drbd`, veurem entre altres missatges

```
1: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r— i més a sota
[>.....] sync'ed: 0.4
```

7) A les dues màquines veurem el dispositiu `/dev/drbd1`, però només des d'aquell que és primari el podrem muntar (no des del secundari). Per a això, instal·lem les eines per a crear un *XFS filesystem* amb l'ordre `apt-get install xfsprogs` i fem sobre el primari (ha2) `mkfs.xfs /dev/drbd1` i després el podrem muntar amb `mount /dev/drbd1 /mnt` i podrem copiar un arxiu com `cp /var/log/messages /mnt/test`.

8) Per a executar uns petits tests, podem fer sobre ha2: `umount /mnt` i després `drbdadm secondary drbddemo` i sobre ha1 primer hem d'instal·lar XFS `apt-get install xfsprogs`, després `drbdadm primary drbddemo` i finalment `mount -t xfs /dev/drbd1 /mnt`. Podem comprovar el contingut amb `ls /mnt` i veurem els arxius que copiem en ha2. Amb l'ordre `cat /proc/drbd` a ambdues màquines veurem l'intercanvi de rols primari per secundari i viceversa.

9) Un segon test que podem fer és apagar el node ha2 (secundari en aquest moment) per a simular una fallada i veurem amb `cat /proc/drbd` que l'altre node no està disponible (`0: cs:WfConnection ro:Primary/Unknown`). Si copiem, fem `cp /mnt/test /mnt/test1` i posem en marxa ha2, quan faci el *boot*

sincronitzarà les dades i veurem sobre ha1 0: *cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r*— i sobre ha2 estaran els arxius sincronitzats.

10) Si provoquem una fallada sobre el primari (apagant-lo o desconnectant la xarxa) i tenim muntat el directori (és el que coneixem com a *shutdown* no ordenat) no hi haurà problemes per a anar a l'altre node, canviar-lo a primari i muntar-lo. Si recuperem el node primari que va fallar, veurem que tots dos queden com a secundaris quan torna a estar actiu, per la qual cosa l'haurem de posar com a primari i el podrem muntar novament, i en el secundari veurem que s'ha sincronitzat i després en uns segons tornarà al seu rol de secundari/primari.

11) Si hi ha canvis durant el temps que un node ha estat desconnectat com a primari, obtindrem un missatge de “split-brain” i per a recuperar-lo haurem d'executar sobre el secundari

```
drbdadm secondary drbddemo i drbdadm --discard-my-data connect drbddemo
```

i sobre el primari `drbdadm connect drbddemo` per a connectar els nodes i resoldre el conflicte. [drbdt]

1.4.4. DRBD + Heartbeat com a NFS d'alta disponibilitat

L'objectiu d'aquest apartat és mostrar com utilitzar DRBD i Heartbeat* per a generar un clúster NFS que permeti tenir una còpia del NFS i que entri en servei quan el servidor primari deixi de funcionar. Per a això, utilitzarem la mateixa instal·lació de l'apartat anterior però sobre un altre disc en cada node (sdc) sobre el qual hem creat una partició (*/dev/sdc1*), i és important verificar que tenim els mòduls de DRBD instal·lats (`lsmod | grep drbd`) [Rem]. A continuació, seguim aquests passos:

*Existeix una gran quantitat de guies de com realitzar el mateix que estem fent aquí amb DRBD i Heartbeat sobre Corosync + Pacemaker, per exemple [HA-NFS], que és sobre SUSE però la configuració del CRM és igual, si bé requereix més detall i temps en la seva configuració.

1) Crearem un arxiu */etc/drbd.d/demo2.res* amb el següent contingut:

```
resource myrs {
    protocol C;
    startup { wfc-timeout 0; degr-wfc-timeout 120; }
    disk { on-io-error detach; }
    on ha1 {
        device /dev/drbd2;
        disk /dev/sdc1;
        meta-disk internal;
        address 172.16.1.1:7788;
    }
    on ha2 {
        device /dev/drbd2;
        disk /dev/sdc1;
        meta-disk internal;
        address 172.16.1.2:7788;
    }
}
```

On utilitzem la partició de cada disc sobre cada node i generarem el dispositiu */etc/drbd2*. És important que els noms de les màquines siguin exactament els

que ens dona `uname -n` i modificar `/etc/hosts`, `/etc/resolv.conf` i `/etc/hostname` perquè les màquines tinguin connectivitat entre elles (i amb l'exterior) mitjançant el nom i la IP. Això s'ha de portar a terme sobre les dues màquines, incloent-hi la còpia de l'arxiu anterior.

2) També sobre les dues màquines haurem d'executar `drbdadm create-md myrs` per a inicialitzar el dispositiu, `drbdadm up myrs` per a activar-lo. Podrem veure el resultat amb `cat /proc/drbd`, que ens mostrarà els dispositius com a "Connected" i inicialitzats.

3) Sobre el servidor primari executem

```
drbdadm drbdadm ---overwrite-data-of-peer primary myrs
```

per a indicar-li que sigui primari, i visualitzant l'arxiu `/proc/drbd` veurem el resultat.

4) Finalment, executem/reiniciem el servei `service drbd start|restart` amb la qual cosa tindrem un dispositiu `/dev/drbd2` preparat per a configurar el sistema d'arxiu.

5) En aquest cas, utilitzarem LVM, ja que permet més flexibilitat per a gestionar les particions, però es podria utilitzar `/dev/drbd2` com a dispositiu de blocs simplement. Instal·lem LVM (`apt-get install lvm2`) i executem:

```
pvcreate /dev/drbd2          Creem la partició LVM física
pvdisplay                   Visualitzem
vgcreate myrs /dev/drbd2    Creem el grup anomenat myrs
lvcreate -L 20M -n web_files myrs  Creem una partició lògica web_files
lvcreate -L 20M -n data_files myrs  Creem una altra partició lògica data_files
lvdisplay                   Visualitzem
```

Amb això, haurem de tenir disponibles les particions en `/dev/myrs/web_files` i `/dev/myrs/data_files`.

Amb això, crearem el sistema d'arxius i el muntem:

```
mkfs.ext4 /dev/myrs/web_files
mkfs.ext4 /dev/myrs/data_files
mkdir /data/web-files          Creem els punts de muntatge
mkdir /data/data-files
mount /dev/myrs/web_files /data/web-files  Muntem les particions
mount /dev/myrs/data_files /data/data-files
```

6) Ara haurem d'instal·lar i configurar el servidor NFS (sobre els dos servidors) perquè pugui ser gestionat per Heartbeat i exportar-lo als clients. Per a això, executem l'ordre `apt-get install nfs-kernel-server` i editem l'arxiu `/etc/exports` amb el següent contingut:

```
/data/web-files 172.16.1.0/24(rw,async,no_root_squash,no_subtree_check,fsid=1)
/data/data-files 172.16.1.0/24(rw,async,no_root_squash,no_subtree_check,fsid=2)
```

És important el valor del paràmetre `fsid` ja que amb aquest els clients de sistema d'arxiu sobre el servidor primari sabran que són els mateixos que en el servidor secundari, i si el primari queda fora no es bloquejaran esperant que torni a estar actiu sinó que continuaran treballant amb el secundari. Com que deixarem que Heartbeat gestioni el NFS, l'hem de treure de `boot` amb (és equivalent a fer-ho amb `systemd` a través de `systemctl disable service`).

```
update-rc.d -f nfs-common remove i update-rc.d -f nfs-kernel-server
remove.
```

7) Finament, hem de configurar Heartbeat (si no es té instal·lat hem d'executar sobre cada node `apt-get install heartbeat`) i per a això modifiquem l'arxiu `/etc/ha.d/ha.cf` amb:

```

autojoin none
auto_failback off
keepalive 2
warntime 5
deadtime 10
initdead 20
bcast eth0
node ha1
node ha2
logfile /var/log/ha-log
debugfile /var/log/had-log

```

S'ha indicat `auto_failback = off`, ja que no desitgem que torni a l'original quan el primari retorni (la qual cosa podria ser desitjable si el hw del primari fos millor que el del secundari, i en aquest cas s'hauria de posar a *on*). `deadtime` indica que considera el servidor fora de servei després de 10 segons, i cada 2 segons preguntarà si estan vius. Deixem el fitxer `/etc/ha.d/authkeys` com ja el teníem definit i executem `/usr/share/heartbeat/ha_propagate` per a copiar els arxius a l'altre servidor (també es podria fer manualment).

8) Per a indicar una adreça virtual als clients NFS, utilitzarem 172.16.1.200 i així sempre tindran aquesta IP com a referent, de manera independent de qui els estigui prestant el servei. El següent és modificar l'arxiu `/etc/ha.d/haresources` per a indicar a Heartbeat quins són els serveis que cal gestionar (IPV, DRBD, LVM2, Filesystems i NFS), i que haurem d'introduir en l'ordre en què es necessiten:

```

ha1 \
IPaddr::172.16.1.200/24/eth0 \
drbddisk::myrs \
lvm2 \
Filesystem::/dev/myrs/web_files::/data/web-files::ext4::nosuid,usrquota,noatime \
Filesystem::/dev/myrs/data_files::/data/data-files::ext4::nosuid,usrquota,noatime \
nfs-common \
nfs-kernel-server

```

S'ha de copiar aquest arxiu en els dos servidors (sense modificar), ja que aquest indica que ha1 és el servidor primari i quan falli serà ha2, tal com ho hem explicat en `ha.cf`.

9) Finalment, podrem iniciar/reiniciar Heartbeat (amb `service heartbeat start|restart`) i provar a muntar el servidor en la mateixa xarxa i fer les proves de fallada corresponents.

Si bé es pot comprovar que és totalment funcional, es recomana, per a un servei estable i de millors prestacions, que es desplegui la infraestructura d'HA amb Corosync i Pacemaker com s'indiquen a les referències [HA-NFS].

Activitats

1. Feu un monitoratge complet del sistema amb les eines que considereu adequades i efectueu un diagnòstic de la utilització de recursos i colls d'ampolla que podrien existir al sistema. Heu de simular la càrrega al sistema del codi de `sumdis.c` que hem donat en el mòdul "Clúster, Cloud i DevOps". Per exemple, utilitzeu `sumdis 1 2000000`
2. Canvieu els paràmetres del nucli i del compilador i executeu el codi esmentat en l'activitat anterior (`sumdis.c`) amb, per exemple: `time ./sumdis 1 1000000`.
3. Amb l'execució de les dues activitats anteriors, traiu conclusions sobre els resultats.
4. Amb el programa d'iteracions indicat en aquest mòdul, implementeu una manera d'acabar l'execució en 5 segons, independent de les crides al sistema excepte si només és una vegada.
5. Monitoreu tots els programes anteriors amb Munin i Ganglia traient conclusions sobre la seva funcionalitat i prestacions.
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 configureu Cacti per tenir gràfiques similars a les de Munin/Ganglia i realitzeu una comparació sobre prestacions, facilitat en la configuració i prestacions de cada eina.
9. Instal·leu i experimenteu amb els sistemes descrits d'alta disponibilitat (Heartbeat, Pacemaker, DRBD).
10. Amb Heartbeat + DRBD, podeu crear un sistema d'arxius NFS redundat i fer les proves de fallada de xarxa sobre el servidor primari (deshabilitant/habilitant la xarxa) perquè el servidor secundari adquireixi el control i després retorni el control a aquest quan el primari recuperi la xarxa.

Bibliografia

Enllaços accedits per última vegada al juliol de 2016.

- [Cacti] Cacti: a complete network graphing solution. <<http://www.cacti.net/>>
- [Coll] Collectd: The system statistics collection daemon <<https://collectd.org/>>
- [D-CfS] Cluster from Scratch. <<https://wiki.debian.org/Debian-HA/ClustersFromScratch>>
- [D-HA] Debian-HA <<https://wiki.debian.org/Debian-HA>>
- [drbd] DRBD.
<<http://www.drbd.org/home/what-is-drbd/>>
- [drbdt] DRBD tests.
<<https://wiki.ubuntu.com/Testing/Cases/UbuntuServer-drbd>>
- [Die] **Bravo E., D.** (2006). *Mejorando la Performance en Sistemas Linux/Unix*. GbuFDL1.2. <di-egobravoestrada@hotmail.com>
<<http://es.tldp.org/Tutoriales/doc-tut-performance/perf.pdf>>
- [DV] **D. Valdez** *Configuración rápida de MRTG*.
<<http://sysnotas.blogspot.com.es/2013/06/mrtg-configuracion-rapida-para-debian.html>>
- [DV2] *Net-SNMP - MIBs*.
<<http://www.net-snmp.org/docs/readmefiles.html>>
- [Edu] **Eduardo Ciliendo, Takechika Kunimasa** (2007). *Linux Performance and Tuning Guidelines*.
<https://lenovopress.com/redp4285?cm_mc_uid=13203986654414690087751&cm_mc_sid_50200000=1469008775>
- [Edu2] **Nate Wiger** *Linux Network Tuning for 2013*.
<<http://www.nateware.com/linux-network-tuning-for-2013.html>>
- [Ga] *Ganglia Monitoring System*.
<<http://ganglia.sourceforge.net/>>
- [GeekP] Linux Cluster Part 2 - Adding and Deleting Cluster Resources.
<<https://geekpeek.net/linux-cluster-resources/>>
- [HA] *Linux-HA*.
<http://linux-ha.org/wiki/Main_Page>
- [HA-NFS] Highly Available NFS Storage with DRBD and Pacemaker (sobre SuSE).
<https://www.suse.com/documentation/sle_ha/singlehtml/book_sleha_techguides/book_sleha_techguides.html>
- [HAD] *Documentación de Linux-HA*.
<<http://www.linux-ha.org/doc/users-guide/users-guide.html>>
- [HPac] Linux-HA Heartbeat + Pacemaker Linux-HA (http://www.linux-ha.org/doc/users-guide/_heartbeat_as_a_cluster_messaging_layer.html)
- [Leu] **Leung, C. T.** *Building a Two-Node Linux Cluster with Heartbeat*.
<<http://www.linuxjournal.com/article/5862>>
- [M] MRTG.
<<http://oss.oetiker.ch/mrtg/>>
- [M2] Howto *sobre instalación de MRTG en Debian*.
<<http://preguntaslinux.org/-howto-instalacion-de-mrtg-monitoreo-debian-t-3061.html>>
- [Maj96] **Majidimehr, A.** (1996). *Optimizing UNIX for Performance*. Prentice Hall.
- [MM] *Monitorització amb Munin i monit*.
<http://www.howtoforge.com/server_monitoring_monit_munin>
- [Mon] *Monit*.
<<http://mmonit.com/monit/>>
- [MonD] *Monitor Debian servers with monit*.
<<http://www.debian-administration.org/articles/269>>

- [MonitEx] Monit: Real-world configuration examples
<<https://mmonit.com/wiki/Monit/ConfigurationExamples>>
- [Mou] **Mourani, G.** (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.
- [MRTG] *Monitorització amb SNMP i MRTG*.
<http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance>
- [Mun] *Munin*.
<<http://munin-monitoring.org/>>
- [Pa] *Pacemaker*.
<http://clusterlabs.org/doc/en-US/Pacemaker/1.0/html/Pacemaker_Explained/s-intro-pacemaker.html>
- [PaceM] *Pacemaker*. <http://clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/_what_is_emphasis_pacemaker_emphasis.html>
- [Pan] *PandoraFMS*. <<http://pandorafms.org/en/>>
- [Pef] *Performance Monitoring Tools for Linux*.
<<http://www.linuxjournal.com/article.php?sid=2396>>
- [PM1] *Pacemaker documentation*.
<<http://clusterlabs.org/doc/>>
- [PM2] *Pacemaker Cluster From Scratch*.
<http://clusterlabs.org/doc/en-US/Pacemaker/1.0/pdf/Clusters_from_Scratch/Pacemaker-1.0-Clusters_from_Scratch-en-US.pdf>
- [PM3] **I. Mora Perez**. *Configuring a failover cluster with heartbeat + pacemaker*.
<<http://opentodo.net/2012/04/configuring-a-failover-cluster-with-heartbeat-pacemaker/>>
- [PM4] **F. Diaz**. *Alta Disponibilidad con Apache2 y Heartbeat en Debian Squeeze*.
<<http://www.muspells.net/blog/2011/04/alta-disponibilidad-con-apache2-y-heartbeat-en-debian-squeeze/>>
- [PTS] *Phoronix Test Suite* <<http://www.phoronix-test-suite.com>>
<<https://wiki.ubuntu.com/PhoronixTestSuite>>
- [Red] *Optimització de servidors Linux*.
<http://people.redhat.com/alikins/system_tuning.html>
- [Rem] **R. Bergsma**. *Redundant NFS using DRBD+Heartbeat*.
<<http://blog.remibergsma.com/2012/09/09/building-a-redundant-pair-of-linux-storage-servers-using-drbd-and-heartbeat/>>
- [Shi] *Shinken: monitoring framework* <<http://www.shinken-monitoring.org>>
- [SNMP] *SNMP - Debian*.
<<https://wiki.debian.org/SNMP>>
- [TunK] *Completely Fair Scheduler*.
<http://www.suse.com/documentation/sles-12/book_sle_tuning/data/sec_tuning_taskscheduler_cfs.html>
- [XY] *XYMon Monitoring System*. <<http://xymon.sourceforge.net/>>
- [Za] *The Enterprise-class Monitoring Solution for Everyone*.
<<http://zabbix.com>>