Configuració, sintonització i optimització

Remo Suppi Boldrito

P07/M2003/02289

UOC
www.uoc.edu

Índex

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 tan adequada com sigui possible a les necessitats que se'n demanen. GNU/Linux és un sistema operatiu-eficient que permet un grau de configuració excel·lent i una optimització molt delicada d'acord amb les necessitats de l'usuari. Per això, una vegada feta una instal·lació (o en alguns casos una actualització), s'han de fer determinades configuracions vitals en el 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/serveis que presta la màquina. Aquesta sintonització dependrà d'on es trobi funcionant la màquina, i es durà a terme, en alguns casos, per a millorar el rendiment del sistema, i en d'altres (a més), per qüestions seguretat (vegeu el mòdul 9, "Administració de seguretat"). Quan és en funcionament, és necessari fer el monitoratge del sistema per tal de veure el seu comportament i actuar en conseqüència. Si bé és un aspecte fonamental, la sintonització d'un operatiu moltes vegades es relega a l'opinió d'experts o *gurus* de la informàtica; però si es coneixen els paràmetres que afecten el rendiment, és possible arribar a bones solucions fent un procés cíclic d'anàlisi, canvi de configuració, monitoratge i ajustos.

1. Aspectes bàsics

Abans de conèixer quines són les tècniques d'optimització, és necessari enumerar les causes que poden afectar les prestacions d'un sistema operatiu [Maj96]. Entre aquestes, es poden esmentar:

a) 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 què se'ls sotmet. El primer pas per a optimitzar el sistema és trobar aquests colls d'ampolla i les seves causes, i conèixer les seves limitacions teòriques i pràctiques.

b) Llei d'Amdahl: segons aquesta llei, "hi ha un límit de quant 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 CPU i s'optimitza reduint la utilització en un factor de 2, el programa millorarà les seves prestacions (*speedup*) en un 5%, la qual cosa pot significar un esforç enorme no compensat amb els resultats.

c) Estimació de l'*speedup*: és necessari estimar com millorarà per a evitar esforços i costos innecessaris. Es pot utilitzar la llei anterior per a valorar si és necessària una inversió en temps o econòmica en el sistema.

d) 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.

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

f) Psicologia de l'usuari: dos paràmetres són fonamentals: 1) l'usuari estarà insatisfet generalment quan es produeixin variacions en el temps de resposta, i 2) l'usuari no detectarà millores en el temps d'execució inferiors al 20%.

g) 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 propis programes de mesura.

h) 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.

Nota

Per a optimitzar cal tenir en compte la saturació dels recursos. Llei d'Amdahl: relaciona els coneixements de programari i maquinari disponible, el temps de resposta i el nombre de treballs.

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

j) Coneixements bàsics sobre el sistema operatiu per 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 *threads* (creació, execució, estats, prioritats, acabament), crides al sistema, *buffers* de *caché*, sistema d'arxius, administració de memòria i memòria virtual (paginació, *swap*) i taules del nucli.

1.1. Monitoratge sobre UNIX System V

El /proc el veurem com un directori però en realitat és un sistema d'arxius fictici, és a dir, no hi és sobre el disc i el nucli el crea en memòria. Aquest s'utilitza per a proveir informació sobre el sistema (originalment sobre processos, d'aquí el nom) que després serà utilitzada per totes les ordres que veurem a continuació. A continuació veurem alguns arxius interessants (consulteu la pàgina del manual per a més informació):

/proc/1: un directori amb la informació del procés 1 (el número del directori és el PID del procés)

/proc/cpuinfo: informació sobre la CPU (tipus, marca, model, prestacions...)

/proc/devices: llista de dispositius configurats en el nucli

/proc/dma: canals de DMA utilitzats en aquell moment

/proc/filesystems: sistemes d'arxius configurats en el nucli

/proc/interrupts: mostra quines interrupcions són en ús i quantes d'elles s'han processat

/proc/ioports: *ídem* amb els ports

/proc/kcore: imatge de la memòria física del sistema

/proc/kmsg: missatges generats pel nucli que 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/modules: mòduls carregats pel nucli

/proc/net: informació sobre els protocols de xarxa

/proc/stat: estadístiques sobre el sistema

/proc/uptime: des de quan el sistema està funcionant

/proc/version: versió del nucli

S'ha de tenir en compte que aquests arxius són visibles (text), però algunes vegades les dades són en "cru" 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 FC inclosos dins del paquet sysstat que inclou a més `iostat` o `mpstat`). El seu equivalent en GNU/Linux Debian és `atsar` (i `atsadc`), que és totalment equivalent als que hem esmentat. L'ordre atsar llegeix comptadors i estadístiques del fitxer /proc i els mostra per la sortida estàndard. La primera manera de cridar a l'ordre és:

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

en què mostra l'activitat en *n* vegades cada *t* segons amb una capçalera que mostra els comptadors d'activitat (el valor per defecte de *n* és 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* (per defecte /var/log/atsar/atsarxx, amb *xx* el dia del mes) i que van ser prèviament desades per atsadc (s'utilitza per a recollir les dades, salvar-les i processar-les i a Debian és en /usr/lib/atsar). El paràmetre *-n* pot ser utilitzat per a indicar el dia del mes i *-s*, *-e* l'hora d'inici-final, respectivament. Per a activar atsadc, per exemple, es podria incloure en /etc/cron.d/atsar unes línies com la següents:

```
@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 crea l'arxiu després d'un reinici. La segona desa les dades cada 10 minuts amb el *shell script* atsa1, que crida a l'atsadc.

En atsar (o sar), les opcions s'utilitzen per a indicar quins comptadors cal mostrar; alguns d'ells són:

Opció	Descripció
u	Utilització CPU
d	Activitat de disc
I (i)	Nombre d'interrupcions/s
v	Utilització de taules en el nucli
y	Estadístiques d'utilització de *ttys*
p	Informació de paginació i activitat de *swap*
r	Memòria lliure i ocupació de *swap*
I (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 d'NFS
A	Totes les opcions

Nota

Monitorar amb atsar
- CPU: atsar -o
- Memòria: atsar -r
- Disc: atsar -d
- Paginació: atsar -p

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

1) Utilització de CPU

```
sar -o 4 5

Linux 2.6.19-prep (localhost.localdomain)24/03/07
```

08:23:22	CPU	%user	%nice	%system	%iowait	%steal	%idle
08:23:26	all	0,25	0,00	0,50	0,00	0,00	99,25
08:23:30	all	0,00	0,00	0,00	0,00	0,00	100,00
08:23:34	all	0,00	0,00	0,00	0,00	0,00	100,00
08:23:38	all	0,25	0,00	0,00	0,00	0,00	99,75
08:23:42	all	0,00	0,00	0,00	0,00	0,00	100,00
Media:	all	0,10	0,00	0,10	0,00	0,00	99,80

- usr i system mostren el percentatge de temps de CPU en el mode usuari amb nice = 0 (normals) i en el mode nucli.

- nice, el mateix, però amb processos d'usuari amb nice > 0 (prioritat més petita que la mitjana).

- idle indica el temps no utilitzat de CPU pels processos en estat d'espera (no inclou espera de disc).

- iowait és el temps que la CPU ha estat lliure quan el sistema estava fent entrada o sortida de disc.

- steal és el temps gastat inútilment esperant per un CPU virtual (vàlid en entorns virtualitzats).

En aquest cas `idle= 100`, que significa que la CPU està ociosa per la qual cosa no hi ha processos per executar i la càrrega és baixa; si `idle = 10` i el nombre de processos és elevat, s'hauria de pensar a optimitzar la CPU, ja que podria ser el coll d'ampolla del sistema.

2) Nombre d'interrupcions per segon

```
sar -I 4 5
Linux 2.6.19-prep (localhost.localdomain) 24/03/07
08:24:01 INTR intr/s
08:24:06 4 0,00
Media: 4 0,00
```

Mostra la informació de la freqüència d'interrupcions dels nivells actius que es troben en /proc/interrupts. Ens és útil per a veure si hi ha algun dispositiu que està interrompent constantment el treball de la CPU.

3) Memòria i *swap*

```
sar -r 4 5
Linux 2.6.19-prep (localhost.localdomain) 24/03/07
```

08:24:20	kbmemfree	kbmemused	%memused	kbbuffers	kbcached	kbswpfree	kbswpused	%swpused	kbswpcad
08:24:24	296516	729700	71,11	24260	459972	963860	0	0,00	0
08:24:28	296500	729716	71,11	24268	459968	963860	0	0,00	0
08:24:32	296516	729700	71,11	24268	459976	963860	0	0,00	0
08:24:36	296516	729700	71,11	24276	459976	963860	0	0,00	0
08:24:40	296500	729716	71,11	24276	459976	963860	0	0,00	0
Media:	296510	729706	71,11	24270	459974	963860	0	0,00	0

En aquest cas kbmemfree és la memòria principal (MP) lliure; used la utilitzada, *buffers* és la MP utilitzada en *buffers*; cached és memòria principal utilitzada en *caché* de pàgines; swpfree/used l'espai lliure/ocupat de *swap*. És important tenir en compte que si no hi ha espai en MP, les pàgines dels processos aniran a parar el *swap*, on hi ha d'haver lloc. Això s'ha de contrastar amb la utilització

de CPU. També s'ha de controlar que la mida dels *buffers* sigui adequada i estigui en relació amb els processos que estan fent operacions d'entrada/sortida.

També és interessant l'ordre free (fc) que permet veure la quantitat de memòria en una visió simplificada:

```
   total used free shared buffers cached
Mem: 1026216 729716 296500 0 24324 459980
-/+ buffers/cache: 245412 780804
Swap: 963860 0 963860
```

Això indica que d'1 Gb gairebé tres quartes parts de la memòria està ocupada i que gairebé la meitat són de *caché*. 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 en volguéssim més detalls, hauríem d'utilitzar l'ordre `vmstat` (o `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:

------procs---------				------------memory---------					--swap--		----yo-----			system		-cpu
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
0	0	0	295896	24384	459984	0	0	321	56	1249	724	11	2	81	5	0
0	0	0	295896	24384	459984	0	0	0	28	1179	383	1	0	99	0	0
0	0	0	295896	24384	460012	0	0	0	0	1260	498	0	0	100	0	0
0	0	0	295896	24384	460012	0	0	0	0	1175	342	0	0	100	0	0
0	0	0	295896	24384	460012	0	0	0	0	1275	526	0	0	100	0	0
0	0	0	295896	24392	460004	0	0	0	72	1176	356	0	0	99	1	0
0	0	0	295896	24392	460012	0	0	0	0	1218	420	0	0	100	0	0
0	0	0	295896	24392	460012	0	0	0	0	1216	436	0	0	100	0	0
0	0	0	295896	24392	460012	0	0	0	0	1174	361	0	0	100	0	0
0	0	0	295896	24392	460012	0	0	0	0	1260	492	0	0	100	0	0

4) Utilització de les taules del nucli

```
 sar -v 4 5
Linux 2.6.19-prep (localhost.localdomain 24/03/07
```

08:24:48	dentunusd	file-sz	inode-sz	super-sz	%super-sz	dquot-sz	%dquot-sz	rtsig-sz	%rtsig-sz
08:24:52	19177	3904	15153	0	0,00	0	0,00	0	0,00
08:24:56	19177	3904	15153	0	0,00	0	0,00	0	0,00
08:25:00	19177	3904	15153	0	0,00	0	0,00	0	0,00
08:25:04	19177	3904	15153	0	0,00	0	0,00	0	0,00
08:25:08	19177	3904	15153	0	0,00	0	0,00	0	0,00
Media:	19177	3904	15153	0	0,00	0	0,00	0	0,00

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

ps -edaflm

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	STIME	TTY	TIME	CMD
4	-	root	1	0	0	-	-	-	508	-	08:01	?	00:00:00	init [5]
1	-	root	1927	7	0	-	-	-	0	-	08:02	?	00:00:00	[kondemand/0]
1	-	rpc	2523	1	0	-	-	-	424	-	08:02	?	00:00:00	syslogd -m 0
5	S	rpc	2566	1	0	-	-	-	444	-	08:02	?	00:00:00	portmap
5	-	root	-	0	78	0	-	-	-	-	08:02	-	00:00:00	-
5		root	2587	1	0	-	-	-	472	-	08:02	?	00:00:00	rpc.statd
5	S	root	-	-	0	81		0	-	-	08:02	-	00:00:00	-
1	-	root	2620		1	0	-	-	1232	-	08:02	?	00:00:00	rpc.idmapd
1	S	root	-	-	0	75		0	-	defaul	08:02	-	00:00:00	-
5	-	root	2804	1	0	-	-	-	1294	-	08:02	?	00:00:00	/usr/sbin/sshd
5	S	root	-	-	0	84	0	-	-	-	08:02	-	00:00:00	-
5	-	root	2910	1	0	-	-	-	551	-	08:02	?	00:00:00	/usr/sbin/atd
5	S	root	-	-	0	84	0	-	-	-	08:02	-	00:00:00	-
4	-	root	3066	1	0	-	-	-	407	-	08:02	tty1	00:00:00	/sbin/mingetty tty1
4		root	3305	1	0	-	-	-	21636	-	08:03	?	00:00:01	nautilus --no-default-window --sm-
4	-	root	3305	1	0	-	-	-	21636	-	08:03	?	00:00:01	client-id default3
0	-	root	3643	3541	0	-	-	-	1123	-	08:17	pts/1	00:00:00	bash
4	-	root	3701	3643	0	-	-	-	1054	-	08:27	pts/1	00:00:00	ps -edaflm

En què els paràmetres reflecteixen el valor indicat en la variable del nucli per a aquest procés, els més importants 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, *zombie*). PRI és la prioritat; NI és nice; STIME, el temps d'inici d'execució; TTY, des d'on s'ha executat; TIME, el temps de CPU; CMD, el programa que s'ha executat i els seus paràmetres. Si es vol sortida amb refresc (configurable), es pot utilitzar l'ordre `top`, que mostra unes estadístiques generals (processos, estats, càrrega, etc.), i, després informació de cadascun d'ells similar al `ps`, però s'actualitza cada 5 segons per defecte:

Nota

Consulteu l'ordre man ps o man top per a una descripció dels paràmetres i característiques.

```
top - 08:26:52 up 25 min, 2 users, load average: 0.21, 0.25, 0.33
Tasks: 124 total, 1 running, 123 sleeping, 0 stopped, 0 zombie
Cpu(s):10.8%us, 2.1%sy, 0.0%ni, 82.0%id, 4.9%wa, 0.1%hi, 0.1%si, 0.0%st
Mem: 1026216k total, 731056k used, 295160k free, 24464k buffers
Swap: 963860k total, 0k used, 963860k free, 460208k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3541	root	15	0	42148	14m	981	S	1.9	1.5	0:00.76	gnome-terminal
3695	root	15	0	260	944	1650	R	1.9	0.1	0:00.02	top
1	root	RT	0	2032	680	580	S	0.0	0.1	0:00.85	init
2	root	34	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	RT	19	0	0	0	S	0.0	0.0	0:00.04	ksoftirqd/0
4	root	10	0	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	16	-5	0	0	0	S	0.0	0.0	0:00.00	events/0
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khelper
7	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kthread
53	root	11	-5	0	0	0	S	0.0	0.0	0:00.01	kblockd/0
54	root	15	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
177	root	18	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
178	root	18	-5	0	0	0	S	0.0	0.0	0:00.00	ksuspend_usbd
181	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khubd
183	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	kseriod
203	root	23	0	0	0	0	S	0.0	0.0	0:00.00	pdflush
204	root	15	0	0	0	0	S	0.0	0.0	0:00.03	pdflush

Debian Linux també inclou tot un conjunt d'eines de monitoratge equivalents a sar, però que són originàries de UNIX BSD i presenten una funcionalitat similar, encara que des de diferents ordres: `vmstat` (estadístiques de CPU, memòria i entrada/ sortida), `iostat` (estadístiques de discos i CPU), `uptime` (càrrega de CPU i estat general).

1.2. Optimització del sistema

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

1) Resoldre els problemes de memòria principal

S'ha de procurar que la memòria principal pugui acollir un percentatge elevat de processos en execució, ja que, si no és així, el sistema operatiu podrà paginar i anar al *swap*; però això significa que l'execució d'aquest procés es degradarà notablement. Si s'hi agrega memòria, el temps de resposta millorarà notablement. Per això, s'ha de tenir en compte la mida dels processos (SIZE) en estat *R* i agregar-li la que utilitza el nucli, que es pot obtenir amb l'ordre `dmesg`, que ens mostrarà (o amb `free`), per exemple:

Memory:
255048k/262080k available (1423k kernel core, 6644k reserved, 466k data, 240k init, Ok highmem

Després s'haurà de comparar amb la memòria física i analitzar si el sistema està limitat per la memòria (es veurà amb `atsar -r` i `-p` molta activitat de paginació).

Les solucions per a la memòria són òbvies: o s'incrementa la capacitat o es redueixen les necessitats. Pel cost actual de la memòria, és més adequat incrementar-ne la mida que fer servir moltes hores per a guanyar un centenar de bytes per treure, ordenar o reduir requisits dels processos en la seva execució. Reduir els requisits es pot fer 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 inoperatiu).

Nota

On cal mirar?
1. Memòria
2. CPU
3. Entrada/sortida
4. TCP/IP
5. Kernel
6. Diversos

Un altre aspecte que es pot reduir és la quantitat de memòria dels usuaris eliminant processos redundants i canviant la càrrega de treball. Per això, s'hauran de monitorar els processos que estan dormint (*zombies*) i eliminar-los, o aquells que no progressen en la seva entrada/sortida (saber si són processos actius, quanta CPU porten gastada i si els "usuaris estan pendents d'ells"). 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 dóna el temps idle (valors baixos). Amb ps o top s'han d'analitzar quins processos són els que "devoren CPU" i prendre decisions com: posposar la seva execució, parar-los temporalment, canviar la prioritat (menys conflictiu de tots, es pot utilitzar l'ordre renice prioritat PID), optimitzar el programa (per a la propera vegada) o canviar la CPU (o agregar-ne una altra). Com ja s'ha esmentat, GNU/Linux utilitza el directori /proc per a mantenir totes les variables de configuració del nucli que poden ser analitzades i, en algun cas concret, "ser ajustada", per a aconseguir prestacions diferents o millors.

Per això, s'ha d'utilitzar l'ordre `systune dump > /tmp/sysfile` per a obtenir totes les variables i els seus valors en l'arxiu /tmp/sysfile (en altres distribucions es pot fer amb `sysctl`). Aquest arxiu es pot editar, canviar la variable corresponent i després utilitzar l'ordre `systune -c /tmp/sysfile` per a carregar-les novament en el /proc. L'ordre systune també llegeix per defecte si no té l'opció -c de /etc/systune.conf. En aquest cas, per exemple, es podrien modificar (cal procedir amb compte, perquè el nucli pot quedar fora de servei) les variables de la categoria /proc/sys/vm (memòria virtual) o /proc/sys/kernel (configuració del *core* del nucli).

En aquest mateix sentit, també (per a experts o desesperats) es pot canviar el temps màxim (*slice*) que l'administrador de CPU (*scheduler*) del sistema operatiu dedica a cada procés de manera circular (si bé és aconsellable utilitzar *renice* com a pràctica). Però GNU/Linux, a diferència d'altres operatius, és un valor fix dins del codi, ja que està optimitzat per a diferents funcionalitats (però és possible tocar-lo). Es pot "jugar" (al vostre propi risc) amb un conjunt de variables que permeten tocar el *time slice* d'assignació de CPU (kernel-source-2.x.x/kernel/sched.c).

3) Reduir el nombre de crides

Una altra pràctica adequada per a millorar les prestacions és reduir el nombre de crides al sistema de més cost en temps de CPU. Aquestes crides són les invocades (generalment) pel *shell* fork() i exec(). Una configuració inadequada de la variable PATH i a causa que la crida exec() no desa res en cau, el directori actual (indicat per . /), pot tenir una relació desfavorable d'execució. Per això, sempre caldrà configurar la variable PATH amb el directori actual com a última ruta. Per exemple, en *bash* (o en .bashrc) feu: export PATH = $PATH:. Si no és així, no hi és el directori actual, o si hi és, refeu 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 amb relació als processos que executa. Mitjançant el monitoratge (`atsar -I`) es pot mirar quin és la relació d'interrupcions per segon i prendre decisions respecte als dispositius que les causen. Per exemple, canviar de mòdem per un altre de més intel·ligent o canviar l'estructura de comunicacions si detectem una activitat elevada sobre el port sèrie al qual es troba connectat.

4) Molta utilització de disc

Després de la memòria, un temps de resposta baix pot ser a causa del sistema de discos. En primer lloc, s'ha de verificar que es disposa de temps de CPU (per exemple, idle > 20%) i que el número d'entrada/sortida sigui elevat (per exemple, > 30 entrada/sortida/s) utilitzant `atsar -u` i `atsar -d`. Les solucions passen per:

a) En un sistema multidisc, planificar on es trobaran els arxius més utilitzats per a equilibrar el trànsit cap a ells (per exemple /home en un disc i /usr sobre

un altre) i que puguin utilitzar totes les capacitats de l'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 `atsar -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.

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

c) En sistemes amb un sol disc, generalment es fan, des del punt de vista de l'espai, quatre particions de la manera següent (des de fora cap a dins): /, swap, /usr, /home; però que 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 per tota la seva longitud. En aquest cas és millor unir les particions /usr i /home en una sola (més gran) encara que pot representar alguns inconvenients quant a manteniment.

d) Incrementar els *buffers* de cau de l'entrada/sortida (vegeu, per exemple: /proc/ide/hd...).

e) Si s'utilitza un ext2fs, es pot utilitzar l'ordre: `dumpe2fs -h /dev/hd...` per a obtenir informació sobre el disc i `tune2fs /dev/hd...` per a canviar alguns dels paràmetres configurables del disc.

f) Òbviament, el canvi del disc per un de més velocitat (RPM) sempre tindrà un impacte positiu en un sistema limitat per entrada/sortida de disc. [Maj96]

5. Millorar aspectes de TCP/IP

Examinar la xarxa amb l'ordre atsar (o també amb `netstat -i` o amb `netstat -s | more`) per a analitzar si hi ha paquets fragmentats, errors, *drops*, *overflows*, etc., que puguin estar afectant les comunicacions i amb això el sistema (per exemple, en un servidor d'NFS, NIS, Ftp o Web). Si es detecten problemes, s'ha d'analitzar la xarxa per a considerar les actuacions següents:

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

b) Planificar on seran els servidors per a reduir el trànsit cap a ells i els temps d'accés.

c) Ajustar paràmetres del nucli (/proc/sys/net/), per exemple, per a obtenir millores en el *throughput* feu:

`echo 600 > /proc/sys/net/core/netdev_max_backlog` (per defecte 300).

6) Altres accions sobre paràmetres del nucli

Hi ha un altre conjunt de paràmetres sobre el nucli que és possible sintonitzar per a obtenir millors prestacions, si bé, tenint en compte el que hem tractat anteriorment, s'hi ha d'anar amb compte, ja que podríem causar l'efecte contrari o inutilitzar el sistema. Consulteu en la distribució del codi font en *kernel-source*-2.4.18/Documentation/sysctl els arxius vm.txt, fs.txt, kernel.txt i sunrpc.txt:

a) /proc/sys/vm: controla la memòria virtual (MV) del sistema. La memòria virtual permet que els processos que no entren en memòria principal siguin acceptats pel sistema però al dispositiu de *swap*, per la qual cosa, el programador no té límit per a la mida del seu programa (òbviament ha de ser inferior al dispositiu de *swap*). Els paràmetres susceptibles de sintonitzar es poden canviar molt fàcilment amb *gpowertweak*.

b) /proc/sys/fs: es poden ajustar paràmetres de la interacció kernel-FS com file-max.

c) I també sobre /proc/sys/kernel, /proc/sys/sunrpc

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 de /usr/src/linux. Una bona configuració del nucli permetrà que aquest s'executi més ràpid, que es disposi de més memòria per als processos d'usuari i a més resultarà més estable. Hi ha dues maneres de construir un nucli: monolític (millors prestacions), o modular (basat en mòduls tindrà millor portabilitat si tenim un sistema molt heterogeni i no es vol compilar un nucli per a cadascun d'ells). 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).

8) És interessant consultar els articles següents:

- http://people.redhat.com/alikins/system_tuning.html sobre informació d'optimització de sistemes servidors Linux.
- http://www.linuxjournal.com/article.php?sid=2396 Performance Monitoring Tools for Linux, si bé és un article antic i algunes opcions no estan disponibles, la metodologia és totalment vigent.

1.3. 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, pot aconseguir una biblioteca estàtica (libr.a), el codi de funció del qual s'inclou

en l'executable o amb una dinàmica (libr.so.xx.x), en el qual es carrega la biblioteca en el moment de l'execució. Si bé les primeres garanteixen codi portable i segur, consumeixen més memòria. El programador haurà de decidir quina és l'adequada per al seu programa incloent -static en les opcions del compilador (no posar-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 les anteriors.

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: `-march` (per exemple, `marchi686 o -marchk6`) en fer simplement `gcc - marchi686`, l'atribut d'optimització -O1,2,3 (-O3 generarà la versió més ràpida del programa, `gcc -O3 -march = i686`) i els atributs -f (consulteu la documentació per als diferents tipus).

3) Optimització del disc: en l'actualitat, la majoria d'ordinadors inclou disc UltraDMA (100) per defecte; tanmateix, en una gran quantitat de casos no estan optimitzats per a extreure les millors prestacions. Hi ha una eina (`hdparm`) que permet sintonitzar el nucli als paràmetres del disc tipus IDE. S'ha d'anar amb compte amb aquesta utilitat, sobretot en discos UltraDMA (verificar en el BIOS que els paràmetres per a suport per DMA estan habilitats), ja que poden inutilitzar el disc. Consulteu les referències i la documentació ([Mou01] i `man hdparm`) sobre quines són (i el risc que comporten) les optimitzacions més importants, per exemple: -c3, -d1, -X34, -X66, -X12, -X68, -mXX, -a16, -u1, -W1, -k1, -K1. Cada opció significa una optimització i algunes són d'altíssim risc, per la qual cosa caldrà conèixer molt bé el disc. Per a consultar els paràmetres optimitzats, es podria utilitzar `hdparm -vtT /dev/hdX` (en què *X* és el disc optimitzat) i la crida a hdparm amb tots els paràmetres es pot posar en /etc/init.d per a carregar-la en el *boot*.

1.4. Configuracions complementàries

Hi ha més configuracions complementàries des del punt de vista de la seguretat que de l'optimització, però són necessàries sobretot quan el sistema està connectat a una intranet o a Internet. Aquestes configuracions impliquen les accions següents [Mou01]:

a) 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-ne l'actual, per la qual cosa s'ha d'inhibir des del BIOS de l'ordinador el *boot* per *floppy* o CD-ROM 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).

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

c) Modificar els arxius de /etc/security, d'acord amb les necessitats d'utilització i seguretat del sistema. Per exemple, en access.conf sobre qui pot fer una connexió al sistema.

```
Format:
permisssion:users : origins
+o -  : usuarios : desde donde
-:ALL EXCEPT root: tty1                  Impedeix l'accés a tots no-root sobre tty1.
-:ALL EXCEPT user1 user2 user3:console   Impedeix l'accés excepte user1,2,3 però
                                         l'últim només des de consola.
-:user1:ALL EXCEPT LOCAL .uoc.edu 'group.conf':
```

També s'haurien de configurar, per exemple, els grups per a controlar què i com i també els límits màxims (limits.conf) per a establir els temps màxims d'utilització de CPU, E/S, etc. per tal d'evitar per exemple DoS.

d) Mantenir la seguretat de la contrasenya de *root*: utilitzar com a mínim 6 caràcters, amb un, almenys, en majúscules o algun caràcter '.-_,' que sigui no trivial; així mateix, és recomanable activar l'envelliment per a forçar a canviar-la periòdicament, així com limitar el nombre de vegades amb contrasenya incorrecta. Així mateix, caldrà canviar el paràmetre minx 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).

e) No accedir al sistema com a *root*: crear un compte com a sysadm i treballar-hi. Si s'hi accedeix remotament, sempre s'haurà d'utilitzar `shh` per a connectar-se al sysadm i, en cas de ser necessari, fer un `su –` per a treballar com a *root*.

f) Temps màxim d'inactivitat: inicialitzar la variable TMOUT, per exemple a 360 (valor expressat en segons), que serà el temps màxim d'inactivitat que esperarà l'intèrpret de dades abans de bloquejar-se; es pot posar als arxius de configuració de l'intèrpret de dades (per exemple /etc/etc/profile, /.bashrc...). En el cas d'utilitzar entorns gràfics (KDE, Gnome, etc.), activeu el salvapantalles amb contrasenya.

g) Configuració de l'NFS en forma restrictiva: en el /etc/etc/exports exporteu només el necessari, no utilitzeu comodins (*wildcards*), permeteu només l'accés de lectura i no permeteu l'accés d'escriptura per *root*, per exemple, amb */directori_exportat host.domain.com* (*ro*, *root_squash*).

h) Evitar arrencades des del lilo (o grub) amb paràmetres: es pot iniciar el sistema com a linux single, la qual cosa engegarà el sistema operatiu en mode d'usuari únic. Configureu el sistema perquè l'arrencada d'aquest mode sempre sigui amb contrasenya. Per això, en l'arxiu /etc/inittab verifiqueu que hi ha la línia: S:wait:/sbin/sulogin i que té habilitat el /bin/sulogin. A més, l'arxiu /etc/ lilo.conf ha de tenir els permisos adequats perquè ningú no el pugui modificar excepte el *root* (`chmod 600/etc/lilo.conf`). Per a prevenir canvis accidentals, canvieu l'atribut de bloqueig amb `chattr +i /etc/lilo.conf` (utilitzeu -i quan el vulgueu canviar). Aquest arxiu permet una sèrie d'opcions que és convenient considerar: timeout 0, si el sistema té només un sistema operatiu perquè faci el *boot* immediatament, *restricted*, per a evitar que puguin inserir ordres en el moment del *boot* com linux init = /bin/sh, i tenir accés com a *root* sense autorització; en aquest cas, ha d'anar acompanyat de `password = 'paraula-de-password';` si només es posa contrasenya, sol·licitarà la contrasenya per a carregar la imatge del nucli. Grub té opcions similars.

i) Control de la combinació Ctrl-Alt-Delete. Per a evitar que es pugui apagar la màquina des del teclat, inseriu un comentari (#) a la primera columna de la línia següent:

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

del archivo /etc/inittab. Activar los cambios con `telinit q`.

j) Evitar serveis no oferts: bloquegeu l'arxiu /etc/services per a no admetre serveis no previstos bloquejant l'arxiu amb *`chattr`* `+i /etc/services`.

k) Connexió del *root:* modifiqueu l'arxiu /etc/securetty que conté les TTY i VC (*virtual console*) al qual es pot connectar el *root* deixant-ne només una de cada, per exemple, *tty1* i *vc*/1, i si és necessari connecteu-vos-hi com a sysadm i feu un `su`.

l) Eliminar usuaris no utilitzats: esborreu els usuaris/grups que no siguin necessaris, inclosos els que vénen per defecte (per exemple, *operator, shutdown, ftp, uucp, games*...), i deixeu-hi 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, es podria considerar bloquejar (`chattr +i file`) els arxius /etc/passwd, /etc/shadow, /etc/group, /etc/gsahdow per a evitar la seva modificació (atenció amb aquesta acció, perquè no permetrà canviar posteriorment els passwd).

m) Muntar les particions de manera restrictiva: utilitzeu en /etc/fstab atributs per a les particions com *nosuid* (no permet suplantar l'usuari o el grup sobre la partició), *nodev* (no interpreta dispositius de caràcters o blocs sobre aquesta partició) i *noexec* (no permet l'execució d'arxius sobre aquesta partició). Per exemple:

```
/tmp /tmp ext2 defaults,nosuid,noexec 0 0
```

També és aconsellable muntar el /boot en una partició separada i amb atributs ro.

n) Proteccions diverses: canvieu a 700 les proteccions dels arxius de /etc/init.d (serveis del sistema) que només el *root* els pugui modificar, arrencar o aturar i modifiqueu els arxius /etc/issue i /etc/issue.net perquè no donin informació (sistema operatiu, versió...) quan algú es connecti per `telnet`, `ssh`, etc.

o) SUID i SGID: un usuari podrà executar com a propietari una ordre si té el bit SUID o SGID activat, la qual cosa es reflecteix com una 's' SUID `(-rwsr-xr-x)` i SGID (-r-xr-sr-x). Per tant, és necessari treure el bit (`chmod a-s file`) en les ordres que no el necessiten. Aquests arxius poden ser buscats amb:

```
find / -type f -perm -4000 o -perm -2000 -print
```

S'ha de procedir amb compte respecte als arxius que treu el SUID- GUID perquè l'ordre podria quedar inútil.

p) Arxius sospitosos: busqueu periòdicament arxius amb noms no usuals, ocults o sense un uid/gid vàlid com '...' (tres punts), '.. ' (punt punt espai), '..^G', per 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 -nogroup` (atenció, perquè algunes instal·lacions aconsegueixen un usuari que després no està definit i que l'administrador ha de canviar).

q) Connexió sense contrasenya: no 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).

r) X Display manager: modifiqueu l'arxiu /etc/X11/xdm/Xaccess per a indicar els *hosts* que es podran connectar mitjançamt XDM i eviteu que qualsevol *host* pugui tenir una pantalla de *connexió*.

1.5. Monitoratge

Hi ha dues eines molt interessants per al monitoratge del sistema: Munin i Monit. Munin produeix gràfics sobre diferents paràmetres del servidor (load average, memory usage, CPU usage, MySQL throughput, eth0 traffic, etc.) sense configuracions excessives, mentre que monit verifica la disponibilitat de serveis com Apache, MySQL, Postfix, i pren diferents accions com per exemple reactivar-lo si el servei no és present. La combinació ofereix gràfics importants per a reconèixer on i què està generant problemes.

Considerem que el nostre sistema s'anomena pirulo.org i tenim la nostra pàgina configurada com a www.pirulo.org amb els documents en /var/www/pi-

rulo.org/web. Per a instal·lar Munin sobre Debian Sarge, 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/www/www.pirulo.org/web/monitoring
logdir /var/log/munin
rundir /var/run/munin
tmpldir /etc/munin/templates
[pirulo.org]
 address 127.0.0.1
 use_node_name yes
```

A continuació, es crea el directori, es canvien els permisos i es reinicia el servei.

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

Després d'uns minuts es podran veure els primers resultats en l'adressa http://www.pirulo.org/monitoring/ al navegador. Per exemple dues gràfiques (càrrega i memòria) es mostren a continuació.

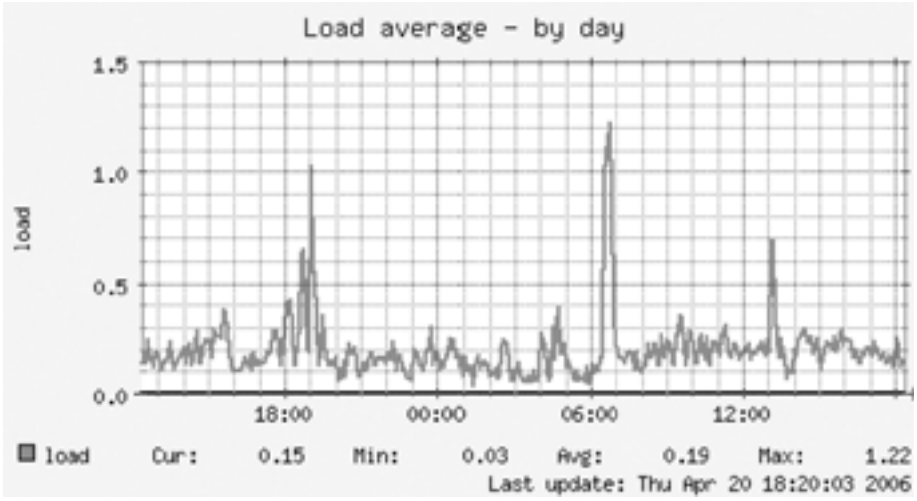


Figura 1

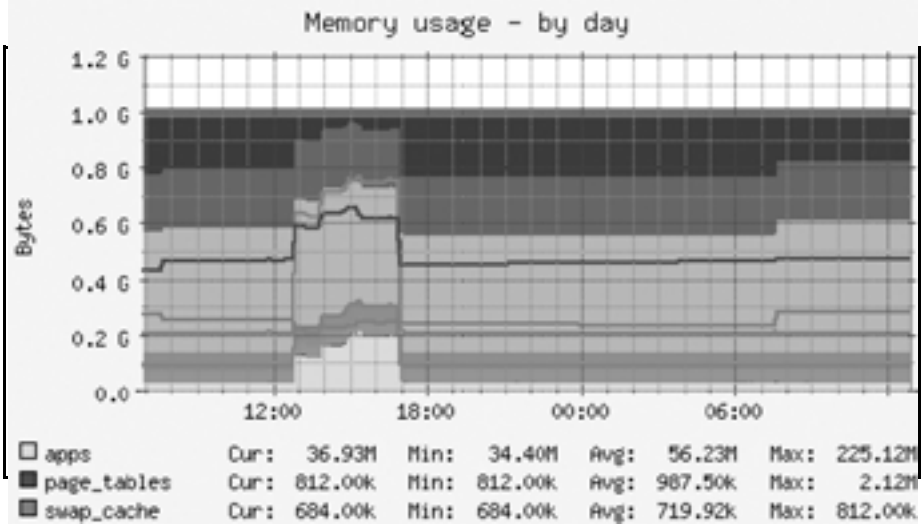


Figura 2

Si es vol mantenir la privacitat de les gràfiques, n'hi ha prou amb posar un passwd a l'accés amb Apache en el directori. Per exemple, es posa en el directori /var/www/pirulo.org/web/monitoring l'arxiu .htaccess amb el contingut següent:

```
AuthType Basic
AuthName "Members Only"
AuthUserFile /var/www/pirulo.org/.htpasswd
<limit GET PUT POST>
require valid-user
</limit>
```

Després s'ha de crear l'arxiu de passwd en /var/www/pirulo.org/.htpasswd amb l'ordre (com a *root*):

```
htpasswd -c /var/www/pirulo.org/.htpasswd admin
```

Quan ens connectem al www.pirulo.org, no demanarà l'usuari (admin) i el passwd que hem introduït després de l'ordre anterior.

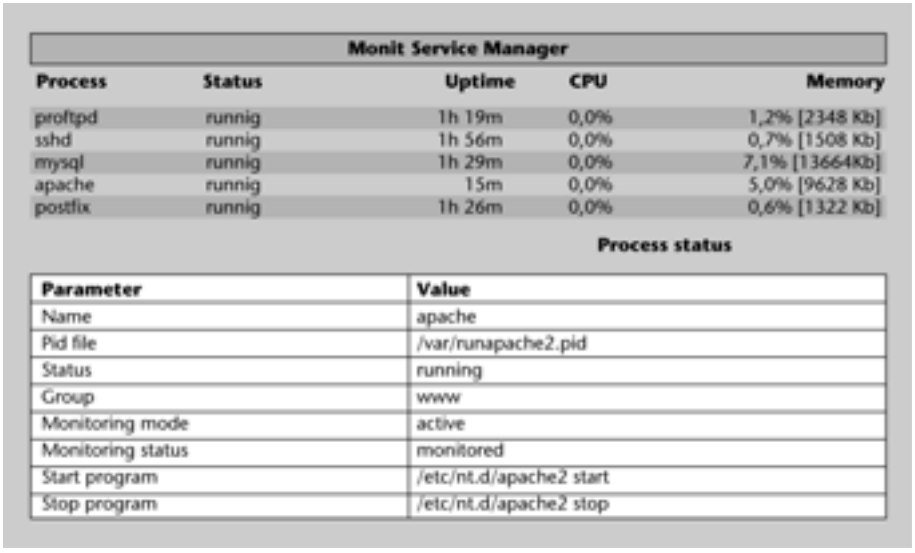
Per a instal·lar-hi monit, fem `apt-get install monit` i editem /etc/monit/monitrc. L'arxiu per defecte inclou un conjunt d'exemples, però es poden obtenir més des de http://www.tildeslash.com/monit/doc/examples.php. Si volem, per exemple, fer el monitoratge de proftpd, sshd, mysql, apache, i postfix, habilitant la interfície web de monit sobre el port 3333 podem fer sobre monitrc:

```
set daemon 60
set logfile syslog facility log_daemon
set mailserver localhost
set mail-format { from: monit@pirulo.org }
set alert root@localhost
set httpd port 3333 and
 allow admin:test
check process proftpd with pidfile /var/run/proftpd.pid
 start program = "/etc/init.d/proftpd start"
 stop program = "/etc/init.d/proftpd stop"
 if failed port 21 protocol ftp then restart
 if 5 restarts within 5 cycles then timeout

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.pid
 group www
 start program = "/etc/init.d/apache2 start"
 stop program = "/etc/init.d/apache2 stop"
 if failed host www.pirulo.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
  if totalmem > 500 MB for 5 cycles then restart
  if children > 250 then restart
  if loadavg(5min) greater than 10 for 8 cycles then stop
  if 3 restarts within 5 cycles then timeout

check process postfix with pidfile /var/spool/postfix/pid/master.pid
  group mail
  start program = "/etc/init.d/postfix start"
  stop program = "/etc/init.d/postfix stop"
  if failed port 25 protocol smtp then restart
  if 5 restarts within 5 cycles then timeout
```

Per a més detalls, consulteu: http://www.tildeslash.com/monit/doc/manual.php. Per a verificar que el servidor Apache funciona Monit, hem posat en la configuració que accedeixi a *if failed hostl www.pirulo.org port 80 protocol http and request " /monit/token" then restart*. Si no hi pot accedir, significa que Apache no funciona, per la qual cosa aquest arxiu hi haurà de ser (mkdir /var/www/pirulo.org/web/monit; echo "pirulo" > /var/www/pirulo.org/web/monit/token). També es pot configurar monit perquè funcioni sobre SSL (vegeu www.howtoforge.com/server_monitoring_monit_munin_p2).

Finalment, s'ha de modificar /etc/default/monit per a habilitar a monit i canviar startup=1 i CHECK_INTERVALS=60 per exemple (en segons). Si posem en marxa monit (`/etc/init.d/monit start`) i ens connectem http://www.pirulo.org:3333, es veurà alguna pantalla similar a:



Monit Service Manager

Process	Status	Uptime	CPU	Memory
proftpd	runnig	1h 19m	0,0%	1,2% [2348 Kb]
sshd	runnig	1h 56m	0,0%	0,7% [1508 Kb]
mysql	runnig	1h 29m	0,0%	7,1% [13664Kb]
apache	runnig	15m	0,0%	5,0% [9628 Kb]
postfix	runnig	1h 26m	0,0%	0,6% [1322 Kb]

Process status

Parameter	Value
Name	apache
Pid file	/var/runapache2.pid
Status	running
Group	www
Monitoring mode	active
Monitoring status	monitored
Start program	/etc/nt.d/apache2 start
Stop program	/etc/nt.d/apache2 stop

Figura 3

Hi ha eines més sofisticades per al monitoratge de xarxa i serveis de xarxa utilitzant SNMP (*simple network management protocol*) i MRTG (*multirouter traffic grapher*) per exemple. Més informació sobre això es pot trobar a:

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance

L'MRTG (http://oss.oetiker.ch/mrtg/) va ser creat bàsicament per a fer gràfiques de dades de xarxa, però es poden utilitzar altres dades per a visualitzar el seu comportament, per exemple, per a generar les estadístiques de càrrega (*load average*) del servidor. Per això, fem ús dels paquets mrtg i atsar. Una vegada instal·lats, configurem el fitxer `/etc/mrtg.cfg`:

```
WorkDir: /var/www/mrtg
Target[average]: `/usr/local/bin/cpu-load/average`
MaxBytes[average]: 1000
Options[average]: gauge, nopercent, growright, integer
YLegend[average]: Load average
kMG[average]: ,,
ShortLegend[average]:
Legend1[average]: Load average x 100
LegendI[average]: load:
LegendO[average]:
Title[average]: Load average x 100 for pirulo.org
PageTop[average]: <H1>Load average x 100 for pirulo.org</H1>
 <TABLE>
 <TR><TD>System:</TD>
 <TD>pirulo.org</TD></TR>
 <TR><TD>Maintainer:</TD> <TD>webmaster@pirulo.org</TD></TR>
 <TR><TD>Max used:</TD> <TD>1000</TD></TR>
 </TABLE>
```

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

```
#!/bin/sh
load= `/usr/bin/atsar -u 1 | tail -n 1 | awk -F" " '{print $10}'`
echo "$load * 100" | bc | awk -F"." '{print $1}'
```

Haurem de crear i canviar els permisos del directori /var/www/mrtg. Per defecte, mrtg s'executa en el cron, però si després el volem executar, podem fer mrtg /etc/mrtg.cfg i això generarà les gràfiques en /var/www/mrtg/average.html que podrem visualitzar amb un navegador des d' http://www.pirulo.org/mrtg/averange.html.

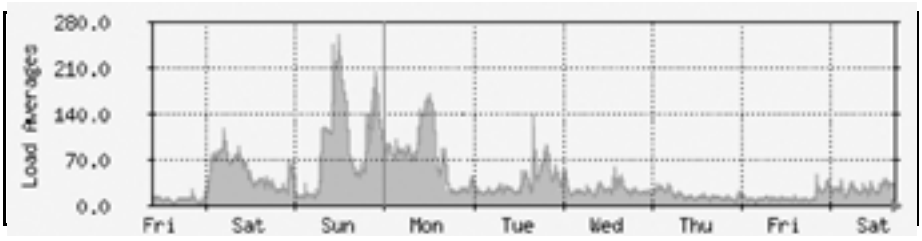


Figura 4

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

- Frysk (http://sources.redhat.com/frysk/): l'objectiu del projecte frysk és crear un sistema de monitoratge distribuït i intel·ligent per a monitorar processos i *threads*.

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

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

- atsar, ac, sac, sysstat, isag: eines d'auditoria com ac, last, accton, sa, atsar o isag (Interactive System Activity Grapher) per a l'auditoria de recursos hw i sw.

- arpwatch; mon: monitor d'activitat ethernet/FDDI que indica si hi ha canvis en taules MACIP; monitor de serveis de xarxa.

- diffmon, fcheck: generació d'informes sobre canvis en la configuració del sistema i monitoratge dels sistemes de fitxers per a detectar-hi intrusions.

- fam: file alteration monitor.

- genpower: monitor per a gestionar els errors d'alimentació.

- gkrellm: monitoratge gràfic de CPU, processos (memòria), sistemes de fitxers i usuaris, disc, xarxa, Internet, *swap*, etc.

- ksensors: (lm-sensors): monitor de placa base (temperatura, alimentació, ventiladors, etc.)

- .lcap, systune: retira capacitats assignades al nucli en el fitxer /proc/sys/kernel i adapta segons les necessitats amb systune.

- logwatcher: analitzador de *logs*.

- Munin i monit: monitoratge gràfic del sistema.

- powertweak i gpowertweak: monitoratge i modificació de diferents paràmetres del maquinari, nucli, xarxa, VFS o VM (permet modificar alguns dels paràmetres mostrats anteriorment sobre /proc).

- gps, gtop, tkps, lavaps (de més a menys amigable): monitors de processos de diversos tipus (generalment utilitzen informació de /proc) i permeten veure recursos, *sockets*, arxius, entorn i una altra informació que aquests utilitzen, així com administrar els seus recursos/estats.

- *swatch*: monitor per a l'activitat del sistema per mitjà d'arxius de *log*.
- vtgrab: monitoratge de màquines remotes (similar a VNC).
- whowatch: eina en temps real per al monitoratge d'usuaris.
- wmnd, dmachinemon: monitor de trànsit de xarxa i monitoratge d'un *clúster* per xarxa.
- xosview, si: monitor de recursos gràfic i System Information.

La figura següent mostra les interfícies de ksensors, gkrellm i xosview, que presenta els resultats del monitoratge en temps real que estan fent.

Figura 5

A continuació, es mostren les interfícies gràfiques `d'isag` i `gtop`. La interfície isag obté la informació generada per `systat` en /etc/cron.d/, `sysstat` per mitjà de l'ordre `sa1` i `sa2` en aquest cas, i és l'acumulatiu del dia; mentre que `gtop` mostra una de les seves possibles vistes amb la ubicació procés, memòria i informació complementària de CPU.

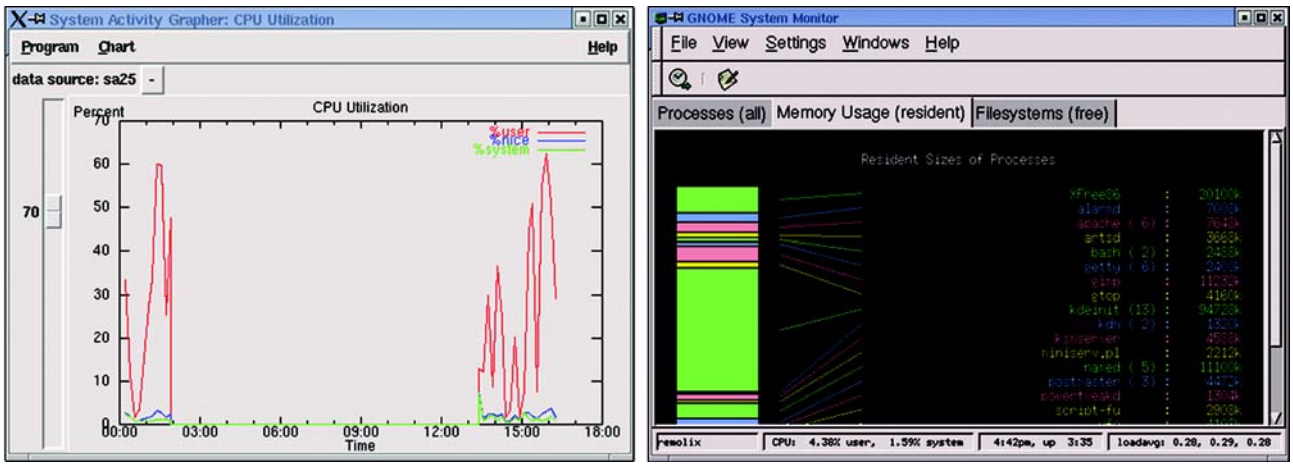


Figura 6

Activitats

1. Feu un monitoratge complet del sistema amb les eines que considereu adequades i feu un diagnòstic de la utilització de recursos i el coll d'ampolla que hi podria haver en el sistema. Simuleu la càrrega en el sistema del codi de sumdis.c donat en la unitat que tracta sobre els *clústers*. Per exemple, utilitzeu:

```
sumdis 1 2000000
```

2. Canvieu els paràmetres del nucli i del compilador i executeu el codi esmentat en el punt anterior (sumdis.c) amb, per exemple:

```
time ./sumdis 1 1000000
```

3. També amb ambdós nuclis i extraieu conclusions sobre els resultats.

Altres fonts de referència i informació

[Debc, Ibi]

Optimització de servidors Linux: http://people.redhat.com/alikins/system_tuning.html

Performance Monitoring Tools for Linux:

http://www.linuxjournal.com/article.php?sid=2396

Munin: http://munin.projects.linpro.no/

Monit: http://www.tildeslash.com/monit/

Monitoratge amb Munin i monit:

http://www.howtoforge.com/server_monitoring_monit_munin

Monitoratge amb SNMP i MRTG:

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch22_:_Monitoring_Server_Performance

MRTG: http://oss.oetiker.ch/mrtg/

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

Cacti: http://cacti.net/