Una Introducción a la Monitorización y Ajuste de Recursos en UNIX

(Versión 1.5)
Puedes descargar la última versión de este documento de:
http://blog.unlugarenelmundo.es/?page_id=127

José María Morales Vázquez

josemaria@morales-vazquez.com

Resumen: Una de las actividades más importantes a la hora de realizar pruebas de prestaciones es la monitorización de los servidores objeto de las pruebas con el fin de comprender que es lo que está pasando en ellos y cual es la causa de que devuelvan un determinado rendimiento en cada momento de la realización de dichas pruebas. Cuando realizamos pruebas sobre un servidor NT, las herramientas de prestaciones suelen 'conectar' directamente con el monitor del sistema operativo (***perfmon***) proporcionándonos de forma gráfica y/o analítica los mismos datos que este presenta a través de su utilidad gráfica de monitorización. Cuando realizamos pruebas de prestaciones sobre servidores UNIX, es decir en el 95% de los casos, los medios que nos proporcionan las herramientas de prestaciones suelen ser insuficientes y se limitan a un subconjunto de datos tomados de la ejecución remota de comandos que recogen estadísticas publicadas por el sistema con ***rmstat*** o ***rexec:vmstat*** *o* ***iostat***. No obstante, no tenemos porque conformarnos con estos datos cuando disponemos de toda una amplia colección de comandos que nos permiten recoger medidas directamente a través del sistema operativo. El principal problema al que nos enfrentamos es el desconocimiento de los mismos. Los comandos de identificación del hardware o el software de una máquina no suelen ser estándar y aquí aparecen los propios de las plataformas Solaris, los mas extendidos en ambientes de producción. Los demás comandos, bien hayan tenido su origen en plataformas UNIX o LINUX pueden encontrarse disponibles en la actualidad para la práctica totalidad de las plataformas existentes.

Índice

1. Introducción. 3
2. Algunos conceptos previos. 4
3. Identificación de recursos en un sistema UNIX. 6
3.1. Identificación del hardware en un sistema Sun Solaris. 6
3.2. Identificación del software en un sistema Sun Solaris. 9
3.3. Identificación de la configuración en un sistema Sun Solaris. 10
3.4. Identificación de usuarios en un sistema Sun Solaris. 11
4. Comandos de monitorización. 12
4.1. Comandos de monitorización polivalentes. 12
4.2. Monitorización de CPU's. 16
4.3. Monitorización de Procesos. 17
4.4. Monitorización de Discos. 18
4.5. Monitorización de la red. 21
4.6. Monitorización de la Memoria. 23
4.7. Visualizando los límites de nuestro sistema. 25
5. Planificación programada de la monitorización. 27
6. Ajustando algunos parámetros. 30
7. Bibliografía. 34

1. Introducción.

Así como en el mundo de los ordenadores domésticos y estaciones de trabajo más del 80% de los sistemas operativos pertenecen a la familia de **Windows**, en el mundo de los grandes servidores estas cifras se invierten y nos encontramos con que más del 80% de las máquinas corren algún tipo de **UNIX**: **Sun Solaris, IBM AIX, HP-UX**, etc.

La monitorización de recursos en sistemas operativos de Microsoft se basa fundamentalmente en el uso de la herramienta ***perfmon*** y la única complicación reside en elegir correctamente, de entre las miles de opciones posibles, los parámetros que nos van a reportar información verdaderamente útil.

En las máquinas **UNIX** tenemos dos grandes problemas. El primero se deriva de sus dimensiones: se trata casi siempre de grandes servidores de varios procesadores, con cantidades de memoria del orden de Gigabytes, multiples discos agrupados en volúmenes **RAID**, decenas de usuarios simultáneos y cientos de procesos corriendo en paralelo en cada instante de tiempo. El segundo problema suele ser nuestro gran desconocimiento del sistema operativo unido a que en **UNIX** las cosas no son tan sencillas como en los sistemas de la empresa de **Redmon**.

No obstante, casi todos los datos están ahí: tan disponibles como a través del monitor de prestaciones de **Windows**. Algunas veces mucho más. El sistema operativo recoge de forma continua exhaustivas mediciones de todo lo que pasa y las almacena en ficheros de texto o en tablas temporales. Tan sólo hay que conocer los comandos que nos devuelven estas mediciones y nos las presentan de forma adecuada.

Como digo, lo más importante es conocer que comando debemos de usar. Una vez hecho esto los sistemas UNIX ponen a nuestro servicio un potente manual (ejecutar ***man*** seguido del comando cuya información deseamos ampliar). Cuando un determinado comando no tiene entrada en el manual del sistema suele responder a la opción **–H** o **–h** para presentar ayuda en línea. Por todo ello, este documento no será un compendio exhaustivo de todas las opciones posibles de cada uno de los comandos presentados, sino que se limitará a introducirlos y a presentar las opciones más útiles de los mismos.

En este documento se proporciona información de todos los comandos necesarios para realizar una completa vigilancia en una máquina **UNIX**. No obstante, hay que reseñar que en **UNIX** hay decenas de formas de hacer lo mismo. Dada la gran diversidad de comandos he intentado elegir los que más información proporcionan, los que están más extendidos en las distintas versiones de **UNIX** o, simplemente, los que conozco. Si puedes aportar algún comando nuevo, más útil y/o que proporcione mayor o mejor información a los aquí expuestos comunícamelo y será incluido en las siguientes versiones de este documento.

2. Algunos conceptos previos.

Puesto que no debemos de olvidar que el objetivo final de nuestra monitorización es auxiliarnos en la realización de unas pruebas de prestaciones, creo necesario antes de empezar, la introducción y definición de algunos conceptos importantes.

Los cuatro conceptos que no debemos de olvidar a la hora de realizar las pruebas son latencia, rendimiento, utilización y eficiencia:

- **Latencia.** Básicamente, mide el tiempo transcurrido entre la realización de una petición y el comienzo de la visualización o ejecución de los resultados. Se mide en unidades de tiempo (segundos, milisegundos...)

- **Rendimiento.** Demanda de trabajo capaz de ser procesada satisfactoriamente por un sistema por unidad de tiempo. Se mide en hits por segundo, Kbytes por hora, Mbytes por día, etc.

- **Utilización.** La utilización mide la fracción de un componente o servicio que estamos usando realmente. Es uno de los parámetros mas comprometidos. Los administradores de sistemas se sienten seguros si la utilización es baja, pero esto limita el rendimiento. Tampoco podemos maximixar la utilización porque corremos el riesgo de bloquear el sistema ante un aumento inesperado de carga. Además, muchos componentes (utilización de CPU, por ejemplo) ofrecen sus mejores prestaciones cuando trabajan en torno al 70-80% de su utilización, presentando peor comportamiento por encima de esta cifra.

- **Eficiencia o eficacia.** Se define habitualmente como el cociente entre rendimiento y utilización.

En el terreno que nos ocupa, nuestro objetivo ha de ser disminuir la latencia y aumentar los otros tres parámetros: rendimiento, utilización y eficiencia.

- **Calidad de servicio.** En sentido amplio, se define como la satisfacción por parte del cliente de sus distintas necesidades de acuerdo con sus requisitos. En el campo que nos ocupa, son tres los parámetros que miden la calidad del servicio:

  - Tiempo de respuesta ante las peticiones.
  - Probabilidad de error, rechazo o pérdida de las peticiones.
  - Caídas o interrupciones de servicio.

- **Cuello de botella.** Elemento software o hardware de un sistema que limita su rendimiento y/o las prestaciones que este ofrece. Todo sistema, por muy potente que sea, posee cuellos de botella: son los eslabones más débiles que nos delimitan la verdadera fuerza de nuestro sistema. No obstante, no siempre tenemos la obligación de encontrarlos: basta con asegurar con que la calidad de servicio será aceptable de acuerdo a los requisitos exigidos.

- **Demanda de trabajo.** Peticiones generadas por los clientes y que deben de ser atendidas por el sistema.

- **Prestaciones del sistema.** Comportamiento del sistema frente a la demanda de trabajo.

3. Identificación de recursos en un sistema UNIX.

Todos los sistemas **UNIX** poseen un juego de comandos para identificar el hardware instalado. Desgraciadamente estos comandos son, en la mayoría de los casos, dependientes del fabricante del sistema operativo. Nos centraremos aquí en la plataforma más importantes y extendida: **Sun Solaris.** Dividiremos esta información en cuatro apartados aunque, a veces, la información que ofrecen se solapa:

- Identificación del hardware.
- Identificación del software.
- Identificación de la configuración.
- Identificación de usuarios.

3.1. Identificación del hardware en un sistema Sun Solaris.

- Información proporcionada: Plataforma hardware.
  comando: **`/usr/bin/uname -i`**
  salida de ejemplo:

```
SUNW, Ultra-Enterprise
```

- Información proporcionada: Procesadores.
  `psrinfo` (*processors information*).
  comando: **`/usr/sbin/psrinfo -v`**
  salida de ejemplo:

```
Status of processor 0 as of: 01/17/02 12:05:24
Processor has been on-line since 01/05/02 20:35:04
the sparc processor operates at 400 MHz, and has a sparc floating point
processor.
Status of processor 1 as of: 01/17/02 12:05:24
Processor has been on-line since 01/05/02 20:35:07
the sparc processor operates at 400 MHz, and has a sparc floating point
processor.
Status of processor 4 as of: 01/17/02 12:05:24
Processor has been on-line since 01/05/02 20:35:07
the sparc processor operates at 400 MHz, and has a sparc floating point
processor.
Status of processor 5 as of: 01/17/02 12:05:24
Processor has been on-line since 01/05/02 20:35:07
the sparc processor operates at 400 MHz, and has a sparc floating point
processor.
```

- Información proporcionada: Cantidad de Memoria RAM disponible.
  `prtconf` (*print system configuration*).
  comando: **`/usr/sbin/prtconf | grep Memory`**
  salida de ejemplo:

```
Memory size: 4096 Megabytes
```

- Información proporcionada: Tamaño, en bytes, de las páginas de memoria.
  comando: **`/usr/bin/pagesize`**
  salida de ejemplo:

```
8192
```

- Información proporcionada: Información del subsistema de discos
  comando: **`/usr/sbin/vxprint -l`**
  salida de ejemplo:

```
Disk group: rootdg

Group:   rootdg
info:    dgid=941616176.1025.sun25
copies:  nconfig=default nlog=default
minors:  >=0

Disk:    c0t0d0
info:    diskid=941616199.1048.sun25
assoc:   device=c0t0d0s2 type=sliced
flags:   autoconfig
device:  pubpath=/dev/vx/dmp/c0t0d0s6 privpath=/dev/vx/dmp/c0t0d0s7
devinfo: publen=17678493 privlen=3590

Disk:    c0t1d0
info:    diskid=941616177.1031.sun25
assoc:   device=c0t0d0s2 type=sliced
flags:   autoconfig
device:  pubpath=/dev/vx/dmp/c0t0d0s4 privpath=/dev/vx/dmp/c0t0d0s3
devinfo: publen=17678493 privlen=3590

Subdisk: c0t0d0-02
info:    disk=c0t0d0 offset=0 len=1027025
assoc:   vol=rootvol plex=rootvol-01 (offset=1)
flags:   enable busy
device:  device=c0t0d0s2 path=/dev/vx/dmp/c0t0d0s6 diskdev=155/6
...
...
Plex:  home-01
info:  len=104139
type:  layout=CONCAT
state: state=ACTIVE kernel=ENABLED io=read-write
assoc: vol=home sd=c0t0d0-06
flags: busy complete
...
...
```

```
Volume:   opt
info:     len=5120766
type:     usetype=fsgen
state:    state=ACTIVE kernel=ENABLED
assoc:    plexes=opt-01,opt-02
policies: read=ROUND exceptions=GEN_DET_SPARSE
flags:    open writeback
logging:  type=REGION loglen=0 serial=0/0 (disabled)
apprecov: seqno=0
device:   minor=8 bdev=156/8 cdev=156/8 path=/dev/vx/dsk/rootdg/opt
perms:    user=root group=root mode=0600
...
...
```

- Información proporcionada: Diagnóstico del hardware
  `prt diag` (*print diagnostic information*).
  comando: **`/usr/platform/<ver nota>/sbin/prtdiag`**
  nota: tenemos un directorio distinto por cada plataforma de hardware diferente. El nombre de la plataforma se puede obtener con **`uname -i`** o **`uname -a`**).
  salida de ejemplo:

```
System Configuration:  Sun Microsystems  sun4u
Sun Enterprise 450 (2 X UltraSPARC-II 296MHz)
System clock frequency: 99 MHz
Memory size: 512 Megabytes

========================= CPUs =========================

                     Run   Ecache   CPU    CPU
Brd  CPU   Module   MHz     MB    Impl.   Mask
---  ---  -------  -----  ------  ------  ----
SYS     1     1      296     2.0   US-II    2.0
SYS     3     3      296     2.0   US-II    2.0

========================= Memory =========================

       Interlv.  Socket   Size
Bank    Group     Name    (MB)  Status
----    -----    ------   ----  ------
  0      none      1901   128      OK
  0      none      1902   128      OK
  0      none      1903   128      OK
  0      none      1904   128      OK

========================= IO Cards =========================

No failures found in System
===========================
```

3.2. Identificación del software en un sistema Sun Solaris

- Información proporcionada: Versión del Sistema Operativo.
  comando: **`/usr/bin/uname -r`**
  salida de ejemplo:

```
5.6
```

- Información proporcionada: Lista de todos los paquetes instalados.
  `pkginfo` (*package information*).
  comando: **`/usr/bin/pkginfo -l`**
  salida de ejemplo:

```
      PKGINST:  SMClsof
         NAME:  lsof
     CATEGORY:  application
         ARCH:  sparc
      VERSION:  4.45
      BASEDIR:  /usr/local
       VENDOR:  Vic Abell
       PSTAMP:  Steve Christensen
     INSTDATE:  Aug 23 2001 11:23
        EMAIL:  steve@smc.vnet.net
       STATUS:  completely installed
        FILES:     23 installed pathnames
                   22 shared pathnames
                    5 directories
                    4 executables
                 1448 blocks used (approx)

      PKGINST:  SUNWarc
         NAME:  Archive Libraries
     CATEGORY:  system
         ARCH:  sparc
      VERSION:  11.6.0,REV=1997.07.15.21.46
      BASEDIR:  /
       VENDOR:  Sun Microsystems, Inc.
           DESC:   system libraries in archive (ar) format for software
development of
     statically linked executables
       PSTAMP:  on297-patchm16195639
     INSTDATE:  Oct 18 2001 14:35
      HOTLINE:  Please contact your local service provider
       STATUS:  completely installed
        FILES:    115 installed pathnames
                    7 shared pathnames
                    2 linked files
                   10 directories
                19245 blocks used (approx)
...
...
```

3.3. Identificación de la configuración en un sistema Sun Solaris.

- Información proporcionada: Nombre de la máquina.
  comando: **`/usr/bin/uname -n`**
  salida de ejemplo:

```
goliath
```

- Información proporcionada: Información básica del sistema.
  comando: **`/usr/bin/uname -a`**
  salida de ejemplo:

```
SunOS sun25 5.6 Generic_105181-26 sun4u sparc SUNW,Ultra-4
```

- Información proporcionada: Información extendida del sistema.
  comando: **`/usr/bin/uname -X`**
  salida de ejemplo:

```
System = SunOS
Node = goliath
Release = 5.6
KernelID = Generic_105181-26
Machine = sun4u
BusType = <unknown>
Serial = <unknown>
Users = <unknown>
OEM# = 0
Origin# = 1
NumCPU = 2
```

- Información proporcionada: Dirección(es) IP de la máquina.
  `ifconfig` (*interface configuration*).
  comando: **`/usr/sbin/ifconfig -a`**
  salida de ejemplo:

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 192.168.2.150 netmask ffffff00 broadcast 192.168.2.255
```

3.4. Identificación de usuarios en un sistema Sun Solaris.

- Información proporcionada: Usuarios conectados al sistema.
  comando: **`/usr/bin/who`**
  salida de ejemplo:

```
jmmv       pts/15       Jan 17 08:18    (192.168.2.128)
fpe        pts/17       Jan 17 12:52    (192.168.2.102)
httpusr    pts/16       Jan 17 12:10    (192.168.2.130)
```

- Información proporcionada: Información sobre los últimos logins y logouts.
  comando: **`/usr/bin/last`**
  salida de ejemplo:

```
jmmv       pts/6  192.168.2.38   Thu Jan 17 12:58 - 13:09  (00:10)
fpe        pts/17 192.168.2.102  Thu Jan 17 12:52   still logged in
httpusr    pts/6  192.168.2.38   Thu Jan 17 12:46 - 12:55  (00:09)
httpusr    pts/19 192.168.2.38   Thu Jan 17 12:29 - 12:36  (00:07)
jmmv       pts/17 192.168.2.102  Thu Jan 17 12:15 - 12:52  (00:36)
httpusr    pts/16 192.168.2.130  Thu Jan 17 12:10   still logged in
jgarrid1   pts/16 goliath        Thu Jan 17 10:35 - 10:58  (00:22)
lsc        pts/17 crispin        Thu Jan 17 10:07 - 10:11  (00:03)
lsc        pts/18 192.168.2.31   Thu Jan 17 10:04 - 12:38  (02:34)
httpusr    pts/6  192.168.2.31   Thu Jan 17 10:00 - 12:33  (02:32)
jmmv       pts/17 192.168.2.217  Thu Jan 17 09:59 - 10:06  (00:07)
httpusr    pts/16 192.168.2.52   Thu Jan 17 09:27 - 10:14  (00:47)
jmmv       pts/6  pedrin         Thu Jan 17 09:25 - 09:53  (00:27)
jmmv       pts/6  pedrin         Thu Jan 17 09:17 - 09:21  (00:04)
...
...
```

- Información proporcionada: Lista de procesos pertenecientes a un usuario.
  `ps` (*process status*).
  comando: **`/usr/bin/ps –fu "userid"`**
  salida de ejemplo:

```
UID        PID    PPID   C      STIME          TTY      TIME CMD
jmmv       16982  16980  0      12:52:39       pts/17   0:00 -ksh
lsc        16763  16761  0      08:17:49       pts/15   0:00 -ksh
```

4. Comandos de monitorización.

Al igual que nos ocurría con los comandos de identificación, los comandos de monitorización son difícilmente 'encasillables' en una categoría concreta de información. No obstante intentaremos presentarlos divididos en cinco grupos:

- Comandos de monitorización polivalentes (memoria, CPU, discos, etc.)
- Monitorización de CPU's
- Monitorización de procesos.
- Monitorización de discos.
- Monitorización de la red.

4.1. Comandos de monitorización polivalentes.

- **`vmstat`** **(***Virtual Memory Statistic*). A pesar de su nombre, este comando ofrece mucho más que un análisis de la memoria del sistema. A través de el podemos obtener datos del estado de los procesos, la utilización de la CPU, la utilización de memoria, el tráfico en los discos, etc. Veremos, a continuación, un breve resumen de la poderosa funcionalidad de este comando. la sintaxis de la llamada es la siguiente:

  `vmstat <opciones> <intervalo de medición> <n° de muestras>`.

  La primera línea es una media de los valores tomados desde el último arranque del sistema y, generalmente, no debe de tenerse en cuenta. Vemos un listado típico tras la ejecución de **`vmstat 5 5`**

```
 procs      memory            page            disk          faults      cpu
 r b w    swap  free  re  mf pi po fr de sr f0 s0 s1 s2   in   sy   cs us sy id
 0 0 0   26112 20472   4  45  5 39 40  0  1  0  1  1  9  534 2617  971  3  3 95
 0 0 0 1397288 8216  12  10  3 65 64  0  0  0  5  5 16  701 6627 1063  3  5 92
 0 0 0 1397280 7976  11   9  1 62 80  0 26  0  0  0  6  515 4089 1039  1  1 98
 0 0 0 1397288 7904  15  13  3104 112 0 14  0  0  0 11  525 3361 1017  0  0 99
 0 0 0 1397272 8016   7   6  1 81 91  0 17  0  1  0  8  525 4817 1053  2  1 97
```

  Veamos algunos de los datos más importantes que nos ofrece el comando:

  - **Procesos:** La columna r (*ready*) indica el número de procesos en cola de listos para su ejecución. La columna b (*blocked for resources*) indica los procesos en espera de una operación de i/o . Por último, la columna w (*swapped*) sumariza los procesos que han sido llevados al area de *swapping* en los últimos segundos.
    Un sistema 'saludable' debería de cumplir las siguientes condiciones: el número de procesos en la columna r debería de ser permanentemente inferior a cuatro veces el número de CPU's del sistema. Lo contrario podría indicar saturación en el

tiempo de proceso. Las columna b debería de ser 0 casi siempre. valores continuamente por encima de este valor indican un cuello de botella en algún disco, interfaz de red, etc. la columna w también debería de valer, casi siempre, 0. Otros valores pueden indicar falta de memoria física.

- **Memoria:** las columna ***swap*** y ***free*** nos indican la memoria disponible para ***swapping*** y la memoria física disponible respectivamente. Ambas están expresadas en kbytes.

- **Disco:** reporta información sobre el número de operaciones por segundo realizadas sobre lo cuatro primeros discos del sistema. No es una medición muy útil: ver mejor el comando *sar* para estos datos.

- **CPU:** Muestra el porcentaje de dedicación de CPU en tres columnas: usuario (us), sistema (sy) y desocupado o en espera (id).

- **sr:** Nos muestra la actividad del demonio encargado de la paginación medido en páginas por segundo. Es el indicador más crítico que podemos tener de falta de memoria. En versiones anteriores a Solaris 8, si presenta valores por encima de 200 páginas por segundo durante un periodo razonable de tiempo, es señal de que necesitamos más memoria. En Solaris 8 podemos decir lo mismo si esta columna es superior a cero. En **Linux** la ejecución de **`vmstat`** es ligeramente diferente. En este sistema, nuestros indicadores más valiosos de falta de memoria son dos columnas etiquetadas como **so** y **si** que indican la cantidad de *swap-out* y *swap-in* respectivamente. Valores altos durante largos periodos en estas columnas nos indican que estamos cortos de memoria.

- **`sar`** (*System Activity Reporter*). Es, al igual que **`vmstat`**, un comando polivalente que nos permite obtener datos sobre datos de disco, utilización de CPU, utilización de memoria, etc. La sintaxis de la llamada es:

  `sar <opciones> <intervalo de medición> <n° de mediciones>.`

  Ejecutado sin opciones de intervalo y número de ejecuciones nos muestra un resumen de las últimas 24 horas. Veremos a continuación algunas de las opciones más útiles:

  - **`sar –u`** Utilización de la CPU. ejemplo: **`sar –u 10 3`**

```
SunOS goliath 5.6 Generic_105181-26 sun4u    01/21/02

12:09:15    %usr    %sys    %wio   %idle
12:09:25       4       3       1      92
12:09:35       9       5       1      86
12:09:45       6       4       2      88
```

```
Average        6       3       1      89
```

Una gran aportación frente a **vmstat** es que, en el listado anterior, nos separa en dos columnas independientes los datos correspondientes a la espera de i/o (wio) y la cpu desocupada (*idle*) que en una ejecución de **vmstat** aparecen sumadas bajo la columna (id).

- **sar -r** Utilización de memoria. Ejemplo: **sar -r 5 10**

```
SunOS pedrin 5.6 Generic_105181-26 sun4u    01/21/02

12:15:05 freemem freeswap
12:15:10    1023  2802502
12:15:15    1044  2796793
12:15:20    1055  2796717
12:15:25    1044  2796516
12:15:30    1072  2769474
12:15:35    1127  2792083
12:15:40    1132  2792242
12:15:45    1128  2803357
12:15:50    1076  2792118
12:15:55    1002  2803328

Average     1070  2794481
```

Atención: Las medidas están expresadas en páginas libres (ver comando ***pagesize***.)

- **sar -d** Utilización de los discos.

```
SunOS pedrin 5.6 Generic_105181-26 sun4u    01/21/02

12:18:32   device      %busy   avque   r+w/s  blks/s  avwait  avserv

12:18:37   fd0           0     0.0       0       0     0.0     0.0
           nfs1          0     0.0       0       0     0.0     0.0
           sd0           0     0.0       0       0     0.0     0.0
           sd0,a         0     0.0       0       0     0.0     0.0
           sd0,b         0     0.0       0       0     0.0     0.0
           sd0,c         0     0.0       0       0     0.0     0.0
           sd0,e         0     0.0       0       0     0.0     0.0
           sd0,f         0     0.0       0       0     0.0     0.0
           sd0,g         0     0.0       0       0     0.0     0.0
           sd0,h         0     0.0       0       0     0.0     0.0
           sd1           0     0.0       0       0     0.0     0.0
           sd1,a         0     0.0       0       0     0.0     0.0
           sd1,c         0     0.0       0       0     0.0     0.0
           sd1,d         0     0.0       0       0     0.0     0.0
           sd1,e         0     0.0       0       0     0.0     0.0
           sd2           4     0.4       8     141     0.0    48.2
           sd2,c         0     0.0       0       0     0.0     0.0
           sd2,d         0     0.0       0       0     0.0     0.0
           sd2,e         4     0.4       8     141     0.0    48.2
           sd3           1     0.0       1      38     0.0    12.4
           sd3,c         0     0.0       0       0     0.0     0.0
           sd3,d         0     0.0       0       0     0.0     0.0
```

```
sd3,e          1     0.0       1      38     0.0    12.4
sd21           0     0.0       0       0     0.0     0.0
st12           0     0.0       0       0     0.0     0.0
```

Aquí vemos la segunda gran aportación de este comando frente a ***vmstat***: nos muestra el porcentaje de utilización de cada una de las particiones del disco (*%busy*) .

Otras opciones interesantes de este comando son:

- **-b.** Actividad de la caché.
- **-c.** Llamadas al sistema.
- **-g.** Información de *pageout*.
- **-p.** Información de *pagein*.
- **-q.** Estadísticas sobre la cola de procesos en espera de CPU.
- **-w.** Estadísticas de actividad de *swapping*.

- **top.** El comando *top* da una buena idea general de la salud de un sistema UNIX. Se trata de una herramienta de libre distribución que, dada su gran utilidad y sencillez, está disponible en la mayoría de las plataformas. La salida típica de un comando *top* es la siguiente:

```
last pid: 29746;  load averages:  0.27,  0.34,  0.40                    17:42:01
659 processes: 658 sleeping, 1 on cpu
CPU states: 86.4% idle,  7.4% user,  4.6% kernel,  1.6% iowait,  0.0% swap
Memory: 512M real, 8008K free, 1469M swap in use, 1388M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
 4344 httpusr   26  58    0   32M   23M sleep  246:28  1.51% ns-slapd
  168 httpusr   22  49    0   45M   34M sleep  174:59  1.03% ns-slapd
29746 jmmv       1  53    0 2320K 1808K cpu/3    0:00  0.65% top
28350 httpusr    3  58    0 7480K 3560K sleep    0:03  0.21% ns-proxy
17590 httpusr   23  59    0   71M   29M sleep   40:50  0.20% java
29097 httpusr    3  48    0 7456K 3416K sleep    0:02  0.20% ns-proxy
29690 httpusr    3  58    0 7208K 3192K sleep    0:00  0.18% ns-proxy
29112 httpusr    3  58    0 7448K 3424K sleep    0:02  0.17% ns-proxy
28859 httpusr    3  58    0 7392K 3472K sleep    0:02  0.16% ns-proxy
28285 httpusr    3  58    0 7360K 3368K sleep    0:02  0.16% ns-proxy
28500 httpusr    3  58    0 7384K 3464K sleep    0:02  0.16% ns-proxy
  555 root       1  58    0 5152K 2744K sleep   47:08  0.15% jre
  573 root       1  58    0 5664K 2944K sleep   47:51  0.15% jre
29095 httpusr    3  58    0 6720K 3224K sleep    0:01  0.10% ns-proxy
10806 httpusr   21  58    0   58M   20M sleep    1:13  0.09% java
```

Pulsando la tecla ‘h’ durante la ejecución del comando nos aparece un panel de ayuda de las opciones disponibles en el modo interactivo, desde el cual podemos enviar una señal de kill a un proceso, decirle que no nos muestre los procesos en estado ‘idle’, cambiar el número de procesos que se visualizan o el campo por el que estos se muestran ordenados.

- Para los fanáticos de X11 existen dos comandos mucho menos funcionales que estos pero que nos pueden mostrar algunos indicadores del sistema de forma más amigable e intuitiva: **`xload`** y **`xosview`**.

4.2. Monitorización de CPU's.

- **`mpstat`** (*Multi Processor Statistics*). En sistemas multiprocesador, muestra un informe de actividad desglosado por procesador. La ejecución va acompañada, al igual que otros comandos similares, del intervalo de medición y el número de muestras tomadas.

  Ejemplo: **`mpstat 5 50`**

```
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   67   0  768   317  107  349   14   52   16    0  1168    5   3   2  90
  1   64   0 1304   140   24  369   15   55   13    0  1220    5   5   2  88
  4   61   0 1224   123   10  370   14   56   14    0  1217    4   4   2  89
  5   62   0  479   137   24  409   16   56   22    0  1305    4   2   2  91
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   10   0  420   316  104  360   14   64   16    0   979    0   0   2  98
  1   50   0  797   130   19  373   11   76   11    0  1108    1   2   2  96
  4    8   0   32   117    5  380   12   71   15    0  1212    1   0   1  97
  5    0   0   20   124    9  402   16   72   19    0  1086    0   0   1  99
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    2   0  372   321  111  385   11   74   17    0   924    0   0   1  99
  1   36   0 2948   127   11  412   15   73   15    0  1137    2   6  12  79
  4   10   0   23   119   13  305    7   55   10    0   781    0  27   3  70
  5   17   0   29   180   66  452   16   81   23    0  1221    1   1   1  98
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0    0   0  379   315  106  370   11   65   16    0  1080    2   0   4  94
  1    0   0   49   133   20  419   14   73   12    0  1151    1   0   3  96
  4   22   0 1271   116    8  300    8   64   10    0   875    0  21   1  78
  5   21   0   42   130   18  425   12   71   19    0  1183    1   0   3  96
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0  150   0  434   316  103  364   15   69   16    0   990    1   2   1  96
  1  168   0  125   193   76  428   16   79   13    0  1379    2   3   1  94
  4  150   0  556   119    3  386   16   72   11    0  1343    2   3   0  95
  5  210   0  313   126   12  461   14   70   20    0  1593    2   4   1  93
CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0   33   0  363   313  103  343   12   64   14    0  1018    0   0   1  99
  1    0   0   48   138   25  395   13   68   16    0  1068    0   0   1  99
  4    0   0   39   114    5  368   10   71   10    0  1071    0   0   1  99
  5   10   0  252   122   10  377   13   64   16    0  1071    1   1   0  97
```

Al igual que hace el comando ***sar***, separa en dos mediciones independientes el tiempo de CPU desocupada (idl) y el de espera de procesos de i/o (wt).

4.3. Monitorización de Procesos.

- **`uptime.`** Muestra un breve resumen del estado del sistema. Un ejemplo de la salida de este comando es la siguiente:

```
5:27pm  up 9 day(s), 20:55,  2 users,  load average: 0.31, 0.39, 0.48
```

Los tres últimos datos mostrados son la carga media (*load average*) del sistema en los últimos periodos de 1, 5 y 10 minutos respectivamente. La carga media del sistema se mide realizando la media de los procesos encolados a la espera de procesador y es un dato muy significativo del estado del sistema. Un sistema que mantiene permanentemente por encima de 4 veces su número de procesadores su '*load average*' del último minuto presenta, normalmente, síntomas de sobrecarga.
Más información útil proporcionada pro este comando: el tiempo transcurrido desde la última vez que se reinició el sistema y el número de usuarios conectados en este momento contra el.

- **`ps` (*Process Status*).** Muestra información sobre los procesos activos, su estado y varias medidas de alto interés referidas a ellos como el porcentaje de tiempo de procesador invertido en ellos, la memoria que consumen, tiempo acumulado de ejecución, propietario, etc.

Usándolo con la opción *aux* nos muestra los procesos ordenados por porcentaje consumido de CPU. Puesto que la lista en un entorno cargado puede estar llena de innumerables procesos insignificantes, conviene realizar un filtro con el comando ***head***, por ejemplo para ver los 10 procesos que más CPU consumen en un momento dado ejecutamos lo siguiente:

```
/usr/ucb/ps aux | head –10
```

La salida así obtenida es la siguiente:

```
USER     PID   %CPU %MEM SZ    RSS   T S  START   TIME  COMMAND

httpusr 4344  4.3  4.5  33216 22824 ? R  Jan 09  243:12  ./ns-slapd -f /var
httpusr 168   0.9  7.1  45816 35680 ? S  Jan 08  173:01  ./ns-slapd -f /var
httpusr 26691 0.3  0.7  6656  3160  ? S  16:33:12  0:00  ./ns-proxy -d /var
root    3     0.3  0.0  0     0     ? S  Jan 07  143:32  fsflush
httpusr 25997 0.3  0.7  6952  3496  ? S  16:24:28  0:02  ./ns-proxy -d /var
httpusr 27337 0.3  0.7  6776  3320  ? S  16:48:12  0:01  ./ns-proxy -d /var
httpusr 25958 0.3  0.7  6992  3504  ? S  16:21:41  0:03  ./ns-proxy -d /var
httpusr 27345 0.2  0.7  6624  3128  ? S  16:48:56  0:00  ./ns-proxy -d /var
httpusr 25944 0.2  0.7  7008  3568  ? S  16:20:52  0:03  ./ns-proxy -d /var
httpusr 27376 0.2  0.7  6600  3136  ? S  16:50:03  0:00  ./ns-proxy -d /var
```

Si deseamos ordenar los procesos por el porcentaje de memoria física que ocupan, la opción a usar en lugar de *aux* es *avx*.
La columna etiquetada con la letra **S** muestra es estado del proceso. Dicho estado puede ser uno de los siguientes:

- **R** – Ejecutándose o a la espera de CPU.
- **<num>** - Idem que el anterior. En algunos sistemas multiprocesador se indica, en lugar de la R el número del procesador que ejecuta el proceso.
- **S** – Proceso dormido durante un tiempo inferior a 20 segundos.
- **I** – Proceso dormido durante más de 20 segundos.
- **W** – Proceso en el área de *swap*.
- **D** – Proceso dormido a la espera de entrada/salida.
- **T** – Proceso dormido por que ha sido suspendido por un usuario.
- **N** – Proceso con prioridad positiva
- **<** - Proceso con prioridad negativa.
- **>** - Proceso que ha excedido su límite de memoria.
- **Z** – Proceso *zombie* a la espera de que su proceso padre o alguna operación de entrada/salida finalicen antes de que acaben con el.
- **P** – Proceso dormido a la espera de una petición de página en disco.

Si lo que queremos son los datos concretos de un proceso cuyo pid conocemos, el comando a ejecutar es el siguiente:

```
ps –fp <pid>
```

o bien:

```
/usr/ucb/ps | grep <pid>
```

4.4. Monitorización de Discos.

- **`iostat` (*Input/Output Statistics*). Aunque podríamos calificarlo también como un comando polivalente, (sirve para obtener estadísticas de i/o**, actividad en los terminales y dispositivos y actividad en la CPU**) nosotros lo usaremos exclusivamente para obtener datos sobre la utilización de los discos** mediante la opción **–xnp**. Un ejemplo de ejecución sería **`iostat –xnp`**

```
                    extended device statistics
  r/s  w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
  2.0  2.0   16.2   33.8  0.3  0.1   66.4   20.9   1   3 c1t0d0
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    7.6   0   0 c1t0d0s0
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    1.0   0   0 c1t0d0s1
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t0d0s2
  2.0  2.0   16.2   33.8  0.3  0.1   66.4   20.9   1   3 c1t0d0s3
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    5.1   0   0 c1t0d0s4
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t0d0s5
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t0d0s6
  0.1  0.5    7.0    3.5  0.0  0.0    0.7    7.0   0   0 c1t1d0
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t1d0s2
  0.0  0.0    0.0    0.0  0.0  0.0    0.0    3.8   0   0 c1t1d0s3
```

```
0.1  0.5    7.0    3.4  0.0  0.0    0.7    7.0   0   0 c1t1d0s4
1.8  1.8   16.8   30.2  0.2  0.1   61.6   20.3   0   2 c1t8d0
0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t8d0s0
0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t8d0s1
0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t8d0s2
0.0  0.0    0.0    0.0  0.0  0.0    0.0    4.7   0   0 c1t8d0s3
1.8  1.8   16.8   30.2  0.2  0.1   61.6   20.3   0   2 c1t8d0s4
0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t8d0s5
0.0  0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c1t8d0s6
```

- **`df` (*Disk Free*).** Visualiza información sobre la cantidad de espacio libre en el disco. Algunos ejemplos de ejecución comentados:

- **`df -k`** Muestra información de la ocupación, en kbytes, de todos los filesystems

```
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/vx/dsk/rootvol   290065   38795  222264    15%    /
/dev/vx/dsk/usr      1272749 1198185   23655    99%    /usr
/proc                      0       0       0     0%    /proc
fd                         0       0       0     0%    /dev/fd
/dev/vx/dsk/var      1527116 1094257  371775    75%    /var
/dev/vx/dsk/opt      1404887 1207174  141518    90%    /opt
swap                  153600   38088  115512    25%    /tmp
```

- **`df -k <file system>`** Muestra información de ocupación unicamente del sistema donde reside el directorio especificado, por ejemplo**:** **`df -k /var/tmp/prestaciones`**

```
Filesystem           kbytes    used   avail capacity  Mounted on
/dev/vx/dsk/var      1527116 1094266  371766    75%    /var
```

- **`df -k .`** Muestra información de ocupación del sistema donde reside el directorio desde el que se ejecuta el comando:

```
Filesystem              kbytes  used  avail capacity Mounted on
/dev/vx/dsk/tresdg/home  120175  103459      4699     96%   /home
```

- **`du`** **(*Disk Usage*).** Sirve para obtener el tamaño de los ficheros. La opción más interesante es **`-ka`**, la cual muestra el tamaño en kbytes de todos los ficheros y directorios de forma recursiva desde el directorio de ejecución del comando. Un listado de ejemplo:

```
1       ./.profile
1       ./.cshrc
1       ./.login
3       ./.sh_history
1       ./scripts/foto.sh
1       ./scripts/sar.awk
1       ./scripts/sar.sh
4       ./scripts
11      .
```

- **bonnie.** Quizás no tan extendida como el resto de las herramientas vistas en este documento, bonnie es un excelente instrumento para determinar la capacidad en lectura y escritura del sistema de ficheros de nuestras máquinas. Lo siguiente es un ejemplo de ejecución:

```
File './Bonnie.2831', size: 104857600
Writing with putc()...done
Rewriting...done
Writing intelligently...done
Reading with getc()...done
Reading intelligently...done
Seeker 1...Seeker 2...Seeker 3...start 'em...done...done...done...
             -------Sequential Output-------- ---Sequential Input-- --Random--
             -Per Char- --Block--- -Rewrite-- -Per Char- --Block--- --Seeks---
Machine    MB K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU  /sec %CPU
          100  1650 65.0  1791 12.2  1141 14.1  2379 88.3  3285 20.4  62.5  4.9
```

4.5. Monitorización de la red.

- **ping.** La utilidad ping envía paquetes ICMP con eco al ***host*** deseado. Se trata de uno de los comandos más conocidos y simples que nos puede aportar fácilmente una idea de la latencia de nuestra red. Usándolo con la opción **-s** tenemos una ejecución continua hasta que es interrumpida (por ejemplo pulsando **Control+C**). Al final de la ejecución obtendremos una breve estadística de resultados:

```
ping -s sun10

PING goliath.unix.enterprise: 56 data bytes
64 bytes from goliath.unix.enterprise (192.168.2.4): icmp_seq=0. time=0. ms
64 bytes from goliath.unix.enterprise (192.168.2.4): icmp_seq=1. time=3. ms
64 bytes from goliath.unix.enterprise (192.168.2.4): icmp_seq=2. time=5. ms
64 bytes from goliath.unix.enterprise (192.168.2.4): icmp_seq=3. time=0. ms
64 bytes from goliath.unix.enterprise (192.168.2.4): icmp_seq=4. time=0. ms
64 bytes from goliath.unix.enterprise (192.168.2.4): icmp_seq=5. time=0. ms
^C
----goliath.unix.enterprise PING Statistics----
6 packets transmitted, 6 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/1/5
```

Los tiempos típicos en una red *Ethernet* poco congestionada varían alrededor de menos de 1 y 3 milisegundos. El *round-trip* o viaje de ida y vuelta mostrado al final es, quizás, el parámetro más importante. Nos muestra como vemos tres datos: el mínimo, la media y el máximo de entre todos los *pings* efectuados. Tiempos muy altos o con grandes diferencias entre estos tres parámetros suelen ser síntomas de una red congestionada o en la que, de una forma u otra, tenemos pérdida de paquetes en algún punto de la misma.

- **traceroute.** Nos proporciona, al igual que el comando ping, la latencia del sistema, pero en este caso desglosada por cada uno de los 'saltos' que siguen nuestros paquetes TCP/IP. En sistemas windows tenemos un comando similar llamado **tracert.** Un ejemplo de ejecución:

```
traceroute enterprise088

1   <10 ms    10 ms   <10 ms  mad-rout1-1.enterprise [192.168.2.1]
2   <10 ms   <10 ms   <10 ms  nbxf91.enterprise [192.168.227.51]
3   <10 ms   <10 ms   <10 ms  192.168.113.55
4   <10 ms   <10 ms   <10 ms  enterprise088 [192.168.1.40]
```

Como vemos, en cada 'salto' el comando hace una consulta inversa al servidor DNS y esto retrasa la visualización de los resultados. Para eliminar esta consulta ejecutamos el comando con la opción **-n** (**-d** en windows).

```
traceroute -n enterprise088

1   <10 ms    10 ms   <10 ms  192.168.14.1
2   <10 ms   <10 ms   <10 ms  192.168.227.51
3   <10 ms   <10 ms   <10 ms  192.168.113.55
4   <10 ms   <10 ms   <10 ms  192.168.1.40
```

- **time.** Otro comando que nos puede proporcionar una medida de la latencia de la red. El comando time nos ofece, entre otras cosas, el tiempo entre la ejecución de un comando y su finalización. Si queremos averiguar, por ejemplo, el tiempo que nos tarda en llegar el contenido de una página web sin el coste de interpretación de su contenido, podemos ejecutar lo siguiente:

```
time more /home/jmmv/http/pru/index.html > /dev/null

real    0m0.02s
user    0m0.00s
sys     0m0.02s
```

- **netstat (*Network Status*).** Herramienta de uso muy generalizado que sirve para diagnosticar problemas en la red y/o monitorizar la actividad en la misma.

  La ejecución del comando con la opción **-i** da la siguiente salida:

```
Name Mtu  Net/Dest  Address   Ipkts     Ierrs    Opkts     Oerrs       Collis Queue
lo0  8232 loopback  localhost 426042133 0        426042133 0           0      0
hme0 1500 goliath   goliath   38962209  0        37671802  0           0      0
hme1 1500 goliath-1 goliath-1 9505822   0        19696393  0           0      0
```

En la ejecución se presenta una línea por cada interfaz de red disponible. Brevemente, el significado de algunos de los distintos campos es el siguiente:

- **Name.** Nombre del interfaz de red. El interfaz lo0 es especial y define un loopback con el propio sistema.
- **Ipkts.** Número de paquetes recibidos.
- **Ierrs.** Número de errores en los paquetes recibidos.
- **Opkts.** Número de paquetes totales transmitidos.
- **Oerrs.** Número de errores en los paquetes transmitidos.
- **Collis.** Número de colisiones
- **Queue.** Paquetes perdidos

Otra opción interesante puede ser la de obtención de estadísticas de un determinado protocolo de la familia TCP/IP. Esto lo obtenemos con la opción **`-sP`** seguida del nombre del protocolo deseado en minúsculas. Un ejemplo: **`netstat -sP tcp.`**

```
TCP  tcpRtoAlgorithm     =     4     tcpRtoMin           =   400
     tcpRtoMax           = 60000     tcpMaxConn          =    -1
     tcpActiveOpens      =798360     tcpPassiveOpens     =1040884
     tcpAttemptFails     = 42017     tcpEstabResets      = 19144
     tcpCurrEstab        =   700     tcpOutSegs          =471170713
     tcpOutDataSegs      =459350873  tcpOutDataBytes     =816801070
     tcpRetransSegs      =346699     tcpRetransBytes     =418662056
     tcpOutAck           =11393043   tcpOutAckDelayed    =2959680
     tcpOutUrg           =   116     tcpOutWinUpdate     = 33453
     tcpOutWinProbe      = 42150     tcpOutControl       =3854035
     tcpOutRsts          =235171     tcpOutFastRetrans   =157753
     tcpInSegs           =461046478
     tcpInAckSegs        =439767626  tcpInAckBytes       =825885847
     tcpInDupAck         =3147607    tcpInAckUnsent      =     0
     tcpInInorderSegs    =430475147  tcpInInorderBytes   =3736431854
     tcpInUnorderSegs    = 34656     tcpInUnorderBytes   =32449519
     tcpInDupSegs        = 58431     tcpInDupBytes       =6980169
     tcpInPartDupSegs    =  3195     tcpInPartDupBytes   =1778045
     tcpInPastWinSegs    =     0     tcpInPastWinBytes   =     0
     tcpInWinProbe       =   579     tcpInWinUpdate      = 17916
     tcpInClosed         = 11459     tcpRttNoUpdate      =242544
     tcpRttUpdate        =438131623  tcpTimRetrans       =343309
     tcpTimRetransDrop   =   120     tcpTimKeepalive     = 48116
     tcpTimKeepaliveProbe= 17764     tcpTimKeepaliveDrop =    67
     tcpListenDrop       =    18     tcpListenDropQ0     =     0
     tcpHalfOpenDrop     =     0
```

- **`ntop.`** Utilísima herramienta para monitorizar la actividad general en la red pero que, desgraciadamente, no está tan extendida como su prima (**`top`**) entre sistemas no LINUX.

4.6. Monitorización de la Memoria.

Antes de comenzar este punto es necesario diferenciar entre dos conceptos bien diferentes pero que a menudo se confunden y se usan de forma equivocada: ***paging*** y ***swapping***. Un

sistema que está haciendo *paging* (paginando) está escribiendo paginas de uso infrecuente de memoria a disco. mientras que un sistema que está haciendo *swapping* , está copiando procesos completos en ejecución de memoria a disco. El paginado no es un síntoma de que nuestro sistema esté corto de memoria, mientras que el *swapping* frecuente si suele ser indicativo de falta de memoria.

- **memstat (*Memory Statistic*).** Solaris, a partir de su versión 7, introduce nuevas estadísticas para la medición de la utilización de la memoria que pueden visualizarse con la herramienta **memstat**. Un ejemplo de ejecución de este parámetro presenta la siguiente información:

```
memory ------- paging--------executable- -anonymous---filesys - ---cpu ---
free re mf pi po fr de sr epi epo epf api apo apf fpi fpo fpf us sy wt  id
49584 0  1 5  0  0   0  0  0   0   0   0   0   0   5   0   0   1  1  1  97
56944 0  1 0  0  0   0  0  0   0   0   0   0   0   0   0   0   0  0  0 100
```

Al igual que ocurría en **vmstat**, la primera línea es una media desde el último rearranque de la máquina y es desechable. memstat divide la actividad de paginación en tres categorías diferentes: ejecutable, anónima y por operaciones del sistema de ficheros. Los sistemas desahogados de memoria deberían de tener valores de pequeña actividad en las columnas **epf** y **apo.** Valores elevados en estas columnas indican que nuestro sistema está corto de memoria.

- **uax**. Herramienta habitual en los sistemas Solaris, aunque no pertenece al sistema operativo. Nos da un report de la utilización de la memoria tal y como el que sigue a continuación:

```
USER  PID  %CPU %MEM  SZ   RSS  TT   S  START   TIME COMMAND
root 16755  0.1  1.0 1448 1208 pts/0 O 17:33:35 0:00 /usr/ucb/ps uax
root     3  0.1  0.0    0    0  ?    S   May 24 6:19 fsflush
root     1  0.1  0.6 2232  680  ?    S   May 24 3:10 /etc/init -
root   167  0.1  1.3 3288 1536  ?    S   May 24 1:04 /usr/sbin/syslog
root     0  0.0  0.0    0    0  ?    T   May 24 0:16 sched
root     2  0.0  0.0    0    0  ?    S   May 24 0:00 pageout
jmmv 14485  0.0  0.9 1424 1088 pts/0 S 16:17:57 0:00 -csh
```

- **pmap**. Posiblemente es el mejor comando del que disponemos para conocer la utilización que hace de la memoria un proceso determinado en sistemas Solaris. la ejecución y ejemplo de salida es como sigue:

```
/usr/proc/bin/pmap –x process-id

Ejemplo:

/usr/proc/bin/pmap –x 14485

14485: -csh
Address   Kbytes Resident Shared Private Permissions     Mapped File
```

```
00010000     144     144       8     136  read/exec       csh
00042000      16      16       -      16  read/write/exec csh
00046000     136     112       -     112  read/write/exec [ heap ]
FF200000     648     608     536      72  read/exec       libc.so.1
FF2B0000      40      40       -      40  read/write/exec libc.so.1
FF300000      16      16      16       -  read/exec       libc_psr.so.1
FF320000       8       8       -       8  read/exec       libmapmalloc.so.1
FF330000       8       8       -       8  read/write/exec libmapmalloc.so.1
FF340000     168     136       -     136  read/exec       libcurses.so.1
FF378000      40      40       -      40  read/write/exec libcurses.so.1
FF382000       8       -       -       -  read/write/exec [ anon ]
FF390000       8       8       8       -  read/exec       libdl.so.1
FF3A0000       8       8       -       8  read/write/exec [ anon ]
FF3B0000     120     120     120       -  read/exec       ld.so.1
FF3DC000       8       8       -       8  read/write/exec ld.so.1
FFBE4000      48      48       -      48  read/write      [ stack ]
--------   ------  ------ ------  ------
total Kb    1424    1320     688     632
```

- **`free`.** No se suele ver mucho fuera de los sistemas LINUX. Permote ver un resumen del estado de la memoria de la máquina:

```
             total       used       free     shared    buffers     cached
Mem:       1036152    1033560       2592       8596      84848     932080
-/+ buffers/cache:      16632    1019520
Swap:       265064        380     264684
```

4.7. Visualizando los límites de nuestro sistema.

- **`ulimit`.** Permite visualizar y, en algunos casos, modificar los límites de nuestro sistema. con la opción **`-aS`** lista todos los límites 'soft' actuales mientras que con la opción **`-aH`** lista los límites 'hard' (los de todo el sistema).

  Los límites soft son límites por defecto digamos como barreras de peligro que el propio usuario puede modificar mediante la utilidad **`ulimit`**. Los límites hard son los límites absolutos del sistema y nadie puede traspasarlos. Es preciso tener privilegios de **`root`** para realizar cualquier modificación sobre ellos..

  Veamos unos ejemplos de ejecución:

  ```
  ulimit -aS

  time(seconds)        unlimited
  file(blocks)         unlimited
  data(kbytes)         unlimited
  stack(kbytes)        8192
  coredump(blocks)     unlimited
  nofiles(descriptors) 64
  vmemory(kbytes)      unlimited
  ```

```
ulimit -aH

time(seconds)        unlimited
file(blocks)         unlimited
data(kbytes)         unlimited
stack(kbytes)        unlimited
coredump(blocks)     unlimited
nofiles(descriptors) 1024
vmemory(kbytes)      unlimited
```

5. Planificación programada de la monitorización.

En todos los sistemas UNIX existe un 'demonio' llamado *cron* encargado de ejecutar tareas en determinados momentos con una gran flexibilidad en la planificación del momento de la ejecución. Se maneja mediante el comando ***crontab*.** Para poder usar dicho comando es preciso que estemos autorizados para ello.

Estamos autorizados para usar ***crontab*** si:

- Nuestro identificador de usuario aparece en el fichero **`/etc/cron.d/cron.allow`**
- o si el fichero **`/etc/cron.d/cron.allow`** no existe y nuestro identificador de usuario no aparece en el fichero **`/etc/cron.d/cron.deny`**

La forma de usar ***crontab*** es la siguiente:

- **`crontab -e.`** Edita el fichero de planificación del usuario con el editor por defecto, tipicamente **vi**.
- **`crontab -l.`** Viualiza el contenido del fichero de planificación del usuario.

Un fichero de planificación consta de cero, una o más líneas de texto. En cada línea deben de aparecer cinco columnas de información separadas por uno o más espacios. Un ejemplo válido de fichero de planificación es el siguiente:

```
0   4,5,10  10  12  *  /usr/home/jmmv/bat1.sh
30  12-17   *   12  0  /usr/home/jmmv/bat2.sh
```

- Primera columna: minuto de ejecución. Rángo válido: 0 a 59.
- Segunda columna: hora de ejecución. Rango válido: 0 a 23
- Tercera columna: día de ejecución. Rango válido: 1 a 31
- Cuarta columna: mes de ejecución. Rango válido: 1 a 12
- Quinta columna: dia de la semana de ejecución. Rango válido: 0 a 6 (0 es el domingo).
- Sexta columna: comando a ejecutar. Debe de ser un comando válido, con su ***path*** totalmente especificado (desde el raíz en absoluto o desde el directorio $HOME en relativo) y sobre el que nuestro usuario tenga permiso de ejecución.

Cualquiera de las cinco primeras columnas puede ser sustituida por:

- Una lista de valores válidos separados por coma, indicando que el comando se ejecutará cuando se cumpla cualquiera de ellos.
- Dos valores válidos separados por un signo menos, indicando que el comando se ejecutará para todos los valores delimitados por ese rango, incluyendo los extremos.
- Un asterisco, significando que el comando se ejecutará para todos los valores válidos del rango de la columna.

En nuestro ejemplo. la primera ejecución:

```
0 4,5,10  10  12  *  /usr/home/jmmv/bat1.sh
```

se realizará a las 4, las 5 y las 10 en punto del día 10 de diciembre, independientemente del día de la semana que sea este.

La segunda ejecución:

```
30  12-17   *   12  0  /usr/home/jmmv/bat2.sh
```

se realizará a las 12.30, 13.30, 14.30, 15.30, 16.30 y 17.30 de todos los domingos del mes de diciembre.

6. Ajustando algunos parámetros.

Las principales plataformas de sistemas UNIX poseen páginas web con información acerca de como ajustar el rendimiento de sus sistemas:

- AIX: http://www.rs6000.ibm.com/resource/technology/sizing.html
- IRIX: http://www.sgi.com/tech/web
- SUN: http://www.sun.com/sun-on-net/performance/book2ref.html

Las plataformas AIX y HP-UX tienen sendas utilidades para ajustar los parámetros del kernel que afectan al rendimiento del sistema. Se llaman **`smit`** y **`sam`** respectivamente.

Los parámetros que afectan al rendimiento del protocolo TCP pueden consultarse y/o modificarse mediante la utilidad **`ndd`**. Su sintaxis es la siguiente:

```
ndd [-set] driver parametro [valor]
```

Ejemplos:

- Para consultar el valor de un parámetro:
  ```
  ndd /dev/tcp tcp_close_wait_interval
  ```

- Para modificar el valor de ese parámetro:
  ```
  ndd –set /dev/tcp tcp_close_wait_interval 60000
  ```

- Para mostrar una lista con todos los parámetros válidos para un driver:
  ```
  ndd /dev/tcp ?
  ```

El significado de muchos de los parámetros que pueden modificarse es oscuro y puede que sólo tenga sentido para nosotros con un amplio conocimiento del protocolo y su funcionamiento. En las páginas web del servidor de SUN y Netscape, iPlanet, aparece una lista con algunos de los parámetros susceptibles de ser modificados para incrementar el rendimiento de nuestros servidores con unos sucintos y a todas veces insuficientes comentarios acerca de su significado. No obstante, constituyen un buen punto de partida para comenzar. Un consejo: no hay que olvidar anotar los valores iniciales antes de modificarlos por si es preciso dar una marcha atrás.

Para máquinas con sistema operativo Solaris, los ajustes recomendados son los siguientes:

Parámetro	Valor por defecto	Valor aconsejado	Comentarios
tcp-close_wait_interval	240000	60000	
tcp_time_wait_interval	240000	60000	Sólo en Solaris 7
tcp_conn_req_max_q	128	1024	
tcp_conn_req_max_q0	1024	4096	
tcp_ip_abort_interval	480000	60000	
tcp_keepalive_interval	7200000	900000	Ajustar este parémetro para sedes web con elevadas tasas de tráfico
tcp_rexmit_interval_initial	3000	-	Incrementar este valor si las retransmisiones superan el 30% o 40%
tcp_rexmit_interval_max	240000	10000	
tcp_rexmit_interval_min	200	3000	
tcp_smallest_anon_port	32768	1024	
tcp_slow_start_initial	1	2	Mejora levemente las transmisiones de pequeñas cantidades de datos
tcp_xmit_hiwat	8192	32768	Para incrementar el buffer de transmisión
tcp_recv_hiwat	8192	32768	para incrementar el buffer de recepción.

En el fichero **`/etc/system`** tenemos otros tres parámetros que afectan muy directamente el rendimiento de nuestra máquina. Si dichos parámetros no aparecen se toman sus valores por defecto.

- **`rlim_fd_max`**. Límite hard (o máximo absoluto) para los descriptores de ficheros. Su valor por defecto es de 1024. Podemos mejorar el rendimiento de sedes web con elevado tráfico ajustándolo a 8192.
- **`rlim_fd_cur`**. Límite soft (o *current*, actual) de para los descriptores de ficheros Su valor por defecto es de 64. Podemos mejorar el rendimiento subiéndolo, igualmente a 8192. El límite soft nunca puede superar al hard.

Existe una estrecha relación entre la pila TCP/IP y el tamaño máximo de los posibles descriptores de ficheros disponibles en el sistema. Un servidor no debería de aceptar ninguna conexión si no tiene descriptores de ficheros disponibles para atenderla, por tanto el valor de los parámetros **`tcp_conn_req_max_q0`** y **`tcp_conn_req_max_q0`** nunca deberían de superar el de los parámetros **`rlim_fd_max`** y **`rlim_fd_cur`**.

- **`sq_max_size`**. Controla el tamaño de la colas para los drivers del ssitema. Ajustándolo a 0 la hacemos infinita, es decir, el rendimiento quedará limitado unicamente por el espacio disponible en el buffer. Por defecto este parámetro tiene un valor de 2.
- **`maxuprc.`** Número máximo de procesos por identificador de usuario
- **`max_nprocs.`** Número máximo de procesos concurrentes.

Para modificar cualquiera de estos parámetros tenemos que editar el fichero antes mencionado e incluir (o modificar la existente) una línea con la siguiente sintaxis:

```
set maxuprc=2641
```

Para máquinas con HP-UX, la lista de ajustes que iPlanet proporciona para la configuración de TCP es la siguiente:

Parámetro	Valor por defecto	Valor aconsejado
tcp_time_wait_interval	60000	60000
tcp_conn_req_max	20	1024
tcp_ip_abort_interval	600000	60000
tcp_keepalive_interval	72000000	900000
tcp_rexmit_interval_initial	1500	1500
tcp_rexmit_interval_max	60000	60000
tcp_rexmit_interval_min	500	500
tcp_xmit_hiwater_def	32768	32768
tcp_recv_hiwater_def	32768	32768

Para cambiar los límites en los descriptores de ficheros hard y soft que pueden ser abiertos por cada proceso, los parámetros a ajustar en este sistema operativo se llaman **`maxfiles`** y **`maxfiles_lim`** y se encuentran en el fichero **`/stand/system`**. El valor por defecto de cada uno de estos parámetros es de 2048 y es recomendable que sea ajustado a 4096 para sistemas web con elevadas tasas de tráfico. Los valores correspondientes al máximo número de procesos por identificador de usuario y concurrentes se ajustan, respectivamente, mediante los parámetros **`maxuprc`** y **`nproc`**. El máximo número de ficheros abiertos se configura con el parámetro **`nfile`**.

AIX no proporciona mecanismos para configurar directamente la mayoría de los parámetros del kernel. El propio sistema operativo modifica dinámicamente la gestión de recursos hasta

unos límites prefijados y no parametrizables. Podemos ver una lista de los valores actuales de la mayoría de estos parámetros mediante el siguiente comando:

```
/etc/lsattr –E –l sys0
```

para ver el valor asignado a algún parámetro concreto, por ejemplo maxuproc, ejecutamos lo siguiente:

```
/etc/lsattr –E –l sys0 –a maxuproc
```

Para cambiar el valor del numero máximo de procesos por identificador de usuario (uno de los escasos configurables por el usuario en este sistema) ejecutamos lo siguiente:

```
/etc/chdev –l sys0 –a maxuproc=200
```

Los límites, establecidos por el kernel e invariables para otros parámetros importantes son los siguientes:

- Límite soft de descriptores de ficheros por proceso: 2000
- Límite hard de descriptores de ficheros: 2000
- Máximo número de procesos concurrentes por identificador de usuario: configurable mediante el parámetro visto sin que pueda excederse de 131072.
- Máximo número de procesos concurrentes en el sistema: 131072.
- Máximo número de ficheros abiertos: 200000.

7. Bibliografía.

1. A. Cockcroft and R. Pettit. *Sun Performance and Tunning (Java and the Internet). 2ª edición.* Sun Microsystems Press.
2. A. Cockcroft and B. Walker. *Capacity Planning for Internet Services.* Sun Mycrosystems Press.
3. P. Killelea and L. Mui *Web Performance Tuning: speeding up the web. 2ª edición.* O'Reilly.
4. D. A. Menasce and V. A. F. Almeida. *Capacity Planning for Web Performance: Metrics, models and Methods.* Prentice Hall.
5. G. D. Musumeci. *System Performance Tuning. 2ª edición* O'Reilly.
6. J.R. Fink, M. Sherer, K. Wall and M.D. Sherer. *Linux performance Tuning and capacity Planning.* Sams.
7. *Unix performance Tuning.* CMP Books.
8. *iPlanet Web Server 4.1 Performance, Tuning, Sizing and Scaling.* http://docs.iplanet.com/docs/manuals/enterprise/50/tuning/perf6.0.pdf