Capítulo 7
Planificación de la capacidad

1. Planificación de la capacidad

Este capítulo trata el monitoreo del uso de recursos del sistema y de la gestión provisional de la evolución de los recursos del sistema.

1.1 Medida y uso de los recursos y depuración

El objetivo de esta sección es de enseñarle a:

– medir el uso de los recursos materiales y del ancho de banda, identificar y resolver los problemas relativos a los recursos.

1.1.1 Competencias principales

– Medir el uso del procesador.
– Medir el consumo de la memoria.
– Medir las entradas/salidas de los discos.
– Medir las entradas/salidas de la red.
– Medir la capacidad de tratamiento del firewall y del enrutamiento.
– Evaluar el consumo de ancho de banda de los clientes.
– Asociar los síntomas y los problemas probables.
– Estimar la capacidad de tratamiento e identificar los cuellos de botella del sistema y de las redes.

1.1.2 Elementos empleados

- `iostat`
- `iotop`
- `vmstat`
- `netstat`
- `ss`
- `iptraf`
- `pstree`, `ps`
- `w`
- `lsof`
- `top`
- `htop`
- `uptime`
- `sar`
- `swap`
- Procesos bloqueados en entrada/salida.
- Bloques en entrada.
- Bloques en salida.

1.2 Planificación prospectiva de los recursos

El objetivo de esta sección es de enseñarle a:

- seguir la evolución del uso de los recursos para anticipar las necesidades en el futuro.

1.2.1 Competencias principales

- Saber utilizar las herramientas de supervisión y de toma de medidas para controlar el uso de la infraestructura informática.
- Determinar el umbral de saturación de una configuración.
- Seguir la progresión del consumo de los recursos.
- Mostrar gráficamente las tendencias de consumo de los recursos.
- Conocimiento básico de las soluciones de supervisión: Icinga2, Nagios, collectd, MRTG y Cacti.

1.2.2 Elementos empleados

- Diagnosticar los problemas de recursos.
- Anticipar el aumento del consumo.
- Prevenir el agotamiento de los recursos.

2. Medida de uso de los recursos y depuración

El administrador del sistema tiene que ser capaz de hacer un censo de los recursos que se encuentran a disposición del sistema y vigilar su uso por el sistema y por las aplicaciones.

2.1 Tipos de recursos

Existen cuatro tipos principales de recursos:

- El o los procesadores.
- La memoria viva.
- El espacio de almacenamiento.
- La red.

Cada una de estas categorías de recursos sirve para permitir que las aplicaciones se ejecuten correctamente, presentando un rendimiento y funcionalidades que puedan satisfacer a los usuarios. Si uno de estos aspectos no fuera suficiente, podría hacer que el funcionamiento del conjunto no fuera correcto, provocando un "cuello de botella".

2.2 Fuentes de información sobre los recursos

El administrador del sistema debe poder cuantificar cada tipo de recurso y vigilar en tiempo real su uso. Para ello, Linux ofrecen diferentes fuentes de información, como las interfaces de comunicación con el núcleo, los comandos o los archivos de registro.

2.2.1 Los pseudo-sistemas de archivos proc y sysfs

El núcleo gestiona los recursos de tipo material y los pone a disposición de las aplicaciones. Para ello, se ocupa de los recursos principales (memoria y procesador) y coordina los pilotos de dispositivos encargados de los otros recursos de tipo material. Por lo tanto, sigue en tiempo real los recursos disponibles y su uso.

Linux dispone de un mecanismo muy potente que permite una comunicación dinámica con el núcleo: los pseudo-sistemas de archivos proc y sysfs (también llamados procfs y sys). Se trata de una interfaz, bajo la forma de un arborescencia de directorios

y de archivos especiales, gestionada por el núcleo. Esto permite obtener información sobre el estado del sistema incluyendo los procesos activos, e incluso enviar información al núcleo con el objetivo de modificar dinámicamente algunos de sus parámetros.

Estos pseudo-sistemas de archivos son una fuente esencial de información sobre los recursos de la máquina así como sobre su uso.

Los elementos que ofrecen información general se encuentran directamente en el directorio de montaje de `proc`, o en su subdirectorio `sys`.

El pseudo-sistema de archivos sysfs permite la gestión unificada de los buses, controladores y dispositivos a partir de una arborescencia virtual. Sobre todo es usado por los controladores de los dispositivos y por aplicaciones especializadas.

En general, el pseudo-sistema de archivos proc está montado en `/proc`, el pseudo-sistema de archivos sysfs está montado en `/sys`, lo podemos comprobar gracias al comando `mount`.

Ejemplo

```
mount
[...]
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
[...]
```

Dos pseudo-sistemas de archivos están montados, en lectura y escritura: proc y sysfs.

El directorio de montaje del pseudo-sistema de archivos proc, `/proc`, contiene numerosos archivos y directorios.

Ejemplo

```
ls /proc
1     15    1904  23    asound       modules
10    1530  1906  24    buddyinfo    mounts
1004  1562  1910  25    bus          mpt
1036  1571  1914  26    cgroups      mtd
1038  1572  1917  27    cmdline      mtrr
1039  16    1928  270   cpuinfo      net
1040  162   1931  271   crypto       pagetypeinfo
1041  1620  1947  28    devices      partitions
1042  1621  2     29    diskstats    sched_debug
1043  1624  20    294   dma          schedstat
1046  1625  2017  295   driver       scsi
1048  163   2018  3     execdomains  self
1049  164   2020  30    fb           slabinfo
1053  1649  2023  303   filesystems  softirqs
1057  1667  2027  31    fs           stat
1059  17    2045  32    interrupts   swaps
```

```
11    1738  2047  37    iomem        sys
12    177   2049  386   ioports      sysrq-trigger
13    178   2056  39    ipmi         sysvipc
1334  179   2057  4     irq          timer_list
1353  18    2058  40    kallsyms     timer_stats
1360  1818  2059  5     kcore        tty
14    1824  2061  6     keys         uptime
141   1825  2062  7     key-users    version
1410  1842  2071  71    kmsg         vmallocinfo
142   1852  2075  8     kpagecount   vmstat
1425  1853  2079  850   kpageflags   zoneinfo
1436  1861  21    851   loadavg
1440  1872  2118  896   locks
1455  1884  2122  897   mdstat
1467  1899  2188  9     meminfo
1483  19    22    acpi  misc
```

Los directorios cuyo nombre es un número están asociados a los procesos activos del sistema, este número corresponde al PID del proceso.

Para acceder a una información, basta con leer el archivo correspondiente. No es un archivo de disco, sino un archivo especial gestionado por el núcleo. Una solicitud de lectura de este archivo envía un mensaje al núcleo y este responde bajo la forma de un conjunto de caracteres.

Ejemplo

Para obtener la versión del núcleo cargado en memoria, basta con leer el contenido del archivo `/proc/version`.

```
cat /proc/version
Linux version 4.18.0-348.23.1.el8_5.x86_64 (mock-build@x86-vm-
07.build.eng.bos.redhat.com) (gcc version 8.5.0 20210514 (Red Hat 8.5.0-4)
(GCC)) #1 SMP Tue Apr 12 11:20:32 EDT 2022
```

El directorio `/proc/sys` contiene especialmente, bajo la forma de pseudo-archivos, información sobre la configuración del núcleo, organizada en directorios por categoría de recursos. Algunos de estos pseudo-archivos pueden ser editados, para poder modificar dinámicamente la configuración del núcleo.

Ejemplos

```
ls /proc/sys
abi   crypto  debug  dev  fs  kernel  net  sunrpc  user  vm
```

Para saber si el enrutamiento IPv4 está activo:

```
cat /proc/sys/net/ipv4/ip_forward
0
```

El núcleo devuelve cero como respuesta a una solicitud de lectura: el enrutamiento no está autorizado. Para activarlo, basta con escribir `1` en el archivo:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
cat /proc/sys/net/ipv4/ip_forward
1
```

Observación

Esta modificación de la configuración es dinámica, pero como se hace directamente en la memoria viva, se perderá en el siguiente reinicio del sistema.

Los directorios dinámicos de procesos

Cada proceso está representado por un directorio en el pseudo-sistema de archivos proc, cuyo nombre es el PID del proceso. Dentro de este directorio se encuentran archivos especiales, gestionados en tiempo real por el núcleo y que dan información sobre las procesos.

En cuanto el proceso se termina, el sistema suprime este directorio.

Ejemplo

En su directorio de conexión, el usuario `becario` abre un archivo `archivo` con Vim. El proceso que ha sido creado tiene como PID 1989.

```
ls -lcd /proc/1989
dr-xr-xr-x. 9 stage stage 0  8 septiembre 13:42 /proc/1989
```

Vemos que el proceso ha sido creado el `08/09` a las `13:42` (fecha de creación del directorio).

```
cd /proc/1989
ls
attr                fdinfo      oom_adj         stack
autogroup           gid_map     oom_score       stat
auxv                io          oom_score_adj   statm
cgroup              limits      pagemap         status
clear_refs          loginuid    patch_state     syscall
cmdline             map_files   personality     task
comm                maps        projid_map      timens_offsets
coredump_filter     mem         root            timers
cpu_resctrl_groups  mountinfo   sched           timerslack_ns
cpuset              mounts      schedstat       uid_map
cwd                 mountstats  sessionid       wchan
environ             net         setgroups
exe                 ns          smaps
fd                  numa_maps   smaps_rollup
```

Todos los archivos y directorios proporcionan información en tiempo real sobre el proceso.

```
cat cmdline
vim archivo
```

El comando ejecutado es: `vim archivo`.

```
cat environ
[...]
PWD=/home/stage
[...]
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpassHOME=/home/
stageSSH_CLIENT=192.168.1.141 65470
22SELINUX_LEVEL_REQUESTED=XDG_DATA_
DIRS=/home/stage/.local/share/flatpak/exports/share:/var/lib/flatpak/
exports/share:/usr/local/share:/usr/shareSSH_TTY=/dev/pts/0MAIL=/var/
spool/mail/stageTERM=xtermSHELL=/bin/bash
```

Este archivo especial da acceso (en solo lectura) a todas las variables de entorno del proceso. Vemos, por ejemplo, el contenido de la variable `PWD: /home/becario`.

```
ls fd
0  1  2  3  4
cat fd/4
3210#"! Utad hello a todos
```

El directorio `fd` contiene la información sobre los archivos abiertos del proceso: las tres entradas/salidas estándares (`0`, `1` y `2`), así como un archivo abierto por vim. El comando `cat` muestra el "contenido" de este archivo abierto, el archivo temporal de vim.

```
cat status
Name:   vim
Umask:  0002
State:  S (sleeping)
Tgid:   1989
Ngid:   0
Pid:    1989
PPid:   1910
TracerPid:      0
Uid:    1001    1001    1001    1001
Gid:    1001    1001    1001    1001
FDSize: 256
Groups: 1001
[...)
```

El archivo `status` muestra la información sobre el estado del proceso.

Cuando el usuario cierra vim, el pseudo-directorio se suprime automáticamente:

```
pwd
/proc/1989
ls
ls: no se puede acceder a .: No existe el fichero o el directorio
```

2.2.2 Los registros del sistema

El sistema y las aplicaciones usan los servicios de un daemon de gestión de registros `rsyslogd` (o de otro equivalente). Se puede encontrar mucha información acerca del uso de los recursos y de los problemas que pueden ser ocasionados consultando los archivos de registro generados por este daemon.

Durante el inicio del sistema, el núcleo debe determinar los recursos materiales de los que dispone. Para ello, activa un componente de software encargado de esta detección, `udev`. Después mostrará en consola los mensajes relativos a estas detecciones y los almacena también en memoria viva, porque el daemon de gestión de registros todavía no está activo y los sistemas de archivos no están posiblemente accesibles en escritura todavía.

Esta información de inicialización podrá ser leída posteriormente, usando el comando `dmesg`. Sin embargo, como el buffer en el cual están almacenados los mensajes es circular, es posible que los mensajes iniciales hayan desaparecido, si el sistema funciona desde hace mucho tiempo o si un controlador de un periférico señala errores de manera recurrente.

Observación

La mayoría de estos mensajes son recuperados de hecho por `rsyslogd` a partir del momento en el que es iniciado y son escritos en el archivo de registro principal (generalmente `/var/log/messages`).

Ejemplo

```
dmesg | more
[    0.000000] microcode: microcode updated early to revision 0x2f,
date = 2019-02-17
[   0.000000] Linux version 4.18.0-348.23.1.el8_5.x86_64 (mock-build@x86-vm-07.build.
eng.bos.redhat.com) (gcc version 8.5.0 20210514 (Red Hat 8.5.0-4) (GCC)) #1 SMP Tue A
pr 12 11:20:32 EDT 2022
[   0.000000] Command line: BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-348.23.1.el8_5.x8
6_64 root=/dev/mapper/cl-root ro crashkernel=auto resume=/dev/mapper/cl-swap rd.lvm.l
v=cl/root rd.lvm.lv=cl/swap rhgb quiet
[   0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers
'
[   0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[   0.000000] x86/fpu: Enabled xstate features 0x3, context size is 576 bytes, using
 'standard' format.
[   0.000000] BIOS-provided physical RAM map:
[...]
[   0.000000] Memory: 2589412K/4127088K available (12293K kernel code, 2261K
rwdata,7880K rodata, 2492K init, 13944K bss, 434124K reserved, 0K cma-reserved)
[...]
```