Administración de Sistemas Avanzada

Eduardo Grosclaude

2014-08-11

[2014/09/01, 12:03:48- Material en preparación, se ruega no imprimir mientras aparezca esta nota]

Resumen

En este escrito se presenta la descripción y material inicial de la asignatura **Administración de Sistemas Avanzada**, para la carrera de Tecnicatura Universitaria en Administración de Sistemas y Software Libre, de la Universidad Nacional del Comahue.

La materia es cuatrimestral en modalidad presencial y las clases son de carácter teórico-práctico, desarrolladas en forma colaborativa. Está preparada con los objetivos generales de capacitar al estudiante para **implementar configuraciones especiales de almacenamiento, aplicar programación avanzada a la automatización de tareas, y diseñar e implementar estrategias de respaldo y de tolerancia a fallos para servicios críticos**.

Página en blanco

Índice

Página en blanco

Parte I
La asignatura

1. Objetivos

De la carrera

Según el documento fundamental de la Tecnicatura, el Técnico Superior en Administración de Sistemas y Software Libre estará capacitado para:

- Desarrollar actividades de administración de infraestructura. Comprendiendo la administración de sistemas, redes y los distintos componentes que forman la infraestructura de tecnología de una institución, ya sea pública o privada.
- Aportar criterios básicos para la toma de decisiones relativas a la adopción de nuevas tecnologías libres.
- Desempeñarse como soporte técnico, solucionando problemas afines por medio de la comunicación con comunidades de Software Libre, empresas y desarrolladores de software.
- Realizar tareas de trabajo en modo colaborativo, intrínseco al uso de tecnologías libres.
- Comprender y adoptar el estado del arte local, nacional y regional en lo referente a implementación de tecnologías libres. Tanto en los aspectos técnicos como legales.

De la asignatura

- Saber implementar configuraciones especiales de almacenamiento
- Saber aplicar programación avanzada a la automatización de tareas
- Saber diseñar e implementar estrategias de respaldo
- Conocer formas de implementar estrategias de tolerancia a fallos para servicios críticos

2. Cursado

- Cuatrimestral de 16 semanas, 128 horas totales
- Clases teórico-prácticas presenciales
- Promocionable con trabajos prácticos

3. Contenidos

Contenidos mínimos

- Instalación sobre configuraciones de almacenamiento especiales.
- Scripting avanzado.
- Planificación de tareas.
- Virtualización.
- Alta Disponibilidad.

Programa

1. Scripting avanzado
   - Estructuras de programación
   - Scripting para tratamiento de archivos
   - Planificación de tareas
2. Configuraciones de almacenamiento
   - Arquitectura de E/S, Dispositivos de E/S, Filesystems
   - Diseños típicos de almacenamiento
   - Software RAID, instalación y mantenimiento niveles 0, 1, 10
   - LVM, instalación y mantenimiento
3. Estrategias de respaldo
   - Copiado y sincronización de archivos
   - Estrategias y herramientas de backup, LVM snapshots
   - Control de versiones
4. Virtualización
   - Formas de virtualización, herramientas. KVM, Proxmox, otras
   - Creación, instalación, migración de MV
   - Cloud. IaaS, PaaS, SaaS, etc.
5. Alta Disponibilidad
   - Clustering de LB, de HA, de HPC. Conceptos de HA.
   - Balance de Carga
   - Heartbeat, DRBD, Clustering de aplicaciones
   - Alta Disponibilidad en Redes. Bonding, STP

4. Bibliografía inicial

- Kemp, Juliet. Linux System Administration Recipes: A Problem-Solution Approach. Apress, 2009.
- Lakshman, Sarath. Linux Shell Scripting Cookbook Solve Real-World Shell Scripting Problems with over 110 Simple but Incredibly Effective Recipes. Birmingham, U.K.: Packt Pub., 2011.
- Parker, Steve. Shell Scripting Expert Recipes for Linux, Bash, and More. Hoboken, N.J.; Chichester: Wiley; John Wiley, 2011.
- Quigley, Ellie. UNIX Shells by Example. 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.
- W. Soyinka, Linux administration a beginners guide. New York, NY: McGraw-Hill Osborne Media, 2012.

5. Cronograma

Semana	Unidad	Trabajo I	Trabajo II
1 2 3	1. Scripting Avanzado		
4 5 6 7	2. Configuraciones de Almacenamiento		
8	Parcial I		
9 10	3. Estrategias de respaldo		
11 12	4. Virtualización		
13 14 15	5. Alta Disponibilidad		
16	Parcial I		

Parte II
Scripting Avanzado

1. Contenidos

1. Comandos básicos de archivos ls, cd, mkdir, cp, mv, rm, ln, patrones de nombres
2. Redirección y piping, comandos head, tail, more, less, grep
3. Variables, ambiente, aritmética
4. Sentencias de control if, for, while, case
5. Funciones
6. Arreglos
7. Expresiones regulares, uso de grep
8. Uso de sort, diff, comm, uniq, cut
9. Uso de cron
10. Otros intérpretes: sed, awk, Perl

2. Ejercitación básica

Redirección y piping

1. Crear un archivo conteniendo la salida del comando ls
2. Crear un archivo conteniendo la salida del comando ls -lR /tmp
3. Obtener las cinco primeras líneas del archivo anterior
4. Crear un archivo conteniendo las cinco primeras líneas y las cinco últimas del archivo generado en 2
5. Crear un archivo conteniendo las primeras cinco líneas de la salida del comando ls -lR /tmp
6. Usando el anterior, crear un archivo conteniendo esas líneas, numeradas
7. Crear un archivo conteniendo las últimas cinco líneas de la salida del comando ls -lR /tmp

Variables, ambiente

1. Asignar e imprimir el contenido de dos variables
2. Asignar dos variables, imprimir sus valores, intercambiar sus valores, imprimirlos
3. Crear un script que imprima un valor que será pasado como argumento
4. Crear un script que imprima dos valores que serán pasados como argumento
5. Crear un script que imprima todos los valores que le sean pasados como argumento

Sentencias de control

1. Imprimir cinco veces "Linux"
2. Imprimir cinco veces el contenido de una variable
3. Imprimir los números de 0 a 5
4. Imprimir los dígitos de -1 a 6
5. Imprimir los números de 0 a 99
6. Imprimir junto al nombre de cada archivo en el directorio actual, su tamaño y su fecha de modificación

7. Copiar los archivos terminados en .txt en archivos con igual nombre pero extensión .bak
8. Renombrar los archivos con extensión .tex que comienzan en ASA reemplazando la partícula ASA con RII
9. Para cada archivo modificado hace más de cinco días en un directorio, mostrar su cantidad de líneas
10. Obtener mediante un cliente de HTTP una lista de archivos cuyos nombres están dados por una expresión variable y controlada por un lazo
11. De un conjunto de archivos tar, encontrar aquellas versiones de un archivo dado, contenido en ellos, que hayan sido modificadas entre dos fechas dadas.

Aritmética

```
$ declare -i num
$ num="hola"
$ echo $num
  0
$ num=5 + 5
  bash: +: command not found
$ num=5+5
$ echo $num
  10
$ num=4*6
$ echo $num
  24
$ num="4 * 6"
$ echo $num
  24
$ num=6.5
  bash: num: 6.5: syntax error in expression (remainder of expression is ".5")
$ i=5; j=$i+1; echo $j
$ i=5; let j=$i+1; echo $j
$ let i=5
$ let i=i+1
$ echo $i
  6
$ let "i = i + 2"
$ echo $i
  8
$ let "i+=1"
$ echo $i
  9
$ i=3
$ (( i+=4 ))
$ echo $i
  7
$ (( i=i-2 ))
$ echo $i
  5
$ let b=2#101; echo $b
$ let h=16#ABCD; echo $h
```

Arreglos

```
$ A=(1 2 3 cuatro cinco)
$ echo ${!A[*]}
0 1 2 3 4
$ echo ${A[4]}
cinco
$ echo ${A[*]}
1 2 3 cuatro cinco
$ A[2]='banana'
$ echo ${A[*]}
1 2 banana cuatro cinco
```

Arreglos asociativos

```
$ declare -A B
$ B=([francia]='paris' [espana]='madrid' [argentina]='buenos aires')
$ echo ${!B[*]}
espana argentina francia
$ echo ${B[*]}
madrid buenos aires paris
$ echo ${B[francia]}
paris
```

Here-Documents

```
$ cat > texto.txt << END
> Hola
> Probando...
> END
$ cat texto.txt
```

Traps

```
# man 7 signal
# 1 = SIGHUP (Hangup of controlling terminal or death of parent)
# 2 = SIGINT (Interrupted by the keyboard)
# 3 = SIGQUIT (Quit signal from keyboard)
# 6 = SIGABRT (Aborted by abort(3))
# 9 = SIGKILL (Sent a kill command)

trap limpieza 1 2 3 6 9
function limpieza
{
   echo "Recibimos senal - desmantelando..."
   rm -f ${tempfiles}
   echo Finalizando
}
```

3. Casos de uso

Investigar el sistema

1. Modificar la salida del comando blkid para conocer el UUID, el nombre y tipo, y punto de montado, de cada dispositivo de bloques del sistema.
2. Analizar archivos de log buscando conocimiento: duración de sesiones ssh por usuario, mensajes de mail entre usuarios, con histograma por tamaños, etc. (ver iptables.log, **??**)
3. Detectar momentos en que la salida de vmstat muestra picos de I/O, procesos corriendo, procesos en espera, uso de swap, etc.

Recuperar espacio de almacenamiento

1. Encontrar los diez archivos más grandes en un directorio y sus hijos, imprimirlos junto con su tamaño de mayor a menor.
2. Encontrar los diez archivos más grandes en un directorio y sus hijos, moverlos a otro directorio (en otro filesystem).
3. Encontrar los diez archivos más grandes del sistema, imprimir el nombre de usuario dueño.
4. Agregar al script anterior el envío de notificación por mail al usuario responsable.
5. Encontrar archivos en directorios de usuario con la cadena "cache" en su nombre e imprimir el uso de disco de cada uno.
6. Idem, enviando nombres a un archivo y usándolo como lista para borrarlos, comprimirlos o moverlos.

Networking

1. Disparar un aviso cuando se pierde la conectividad a un conjunto dado de nodos de la red.
2. Analizar la salida del comando netstat para descubrir en qué momento aparece un nuevo port abierto y a qué aplicación corresponde.
3. Obtener un log de tráfico y obtener orígenes máximos y mínimos de tráfico, cantidades totales de bytes traficados por interfaz, etc.
4. Recoger estadísticas de espacio en disco, cantidad de procesos, carga de CPU, en diferentes nodos de la red, y centralizarlos en un nodo monitor que presente los resultados.

Seguridad

1. Detener el script si la identidad del proceso corresponde a root.
2. Solicitar información confidencial (como claves) con video inhibido.
3. Capturar señales para impedir la interrupción del script por BREAK o fallos de ejecución.
4. Utilizar MD5/SHAx para confirmar integridad de archivos.

Tratamiento de datos

1. Revisar el uso de los comandos cut, join, sort, uniq, comm.
2. Crear script que administra una base de datos en formato CSV.
3. Dado un archivo con una lista de direcciones IP, adjuntarles la resolución inversa de nombres correspondiente.
4. Crear un histograma de accesos por nombre de dominio, a partir de los paquetes registrados en un archivo de log generado por iptables.
5. Dada una base de datos CSV implementar búsqueda por expresiones regulares.

6. Dada una base de datos CSV implementar proyección sobre un conjunto de campos dados.
7. Convertir un listado de individuos PDF en archivo CSV.
8. Preparar un conjunto de scripts con un único punto de entrada para el administrador. Estos scripts mantendrán un conjunto de bases de datos en formato CSV:

```
alumnos: UID, Username, Apellido, Nombres, NoLegajo, Activo
materias: MID, Nombre, Carrera, Docente
cursadas: UID, MID, Ano, Cuatrimestre
```

   El dato Activo es booleano. Con estas bases de datos:

   - Listar todas las materias asignadas a un mismo docente.
   - Listar todas las materias cursadas por un alumno.
   - Listar todos los alumnos activos inscriptos en una materia.
   - Listar todos los alumnos que cursan una misma carrera dada durante un año dado.
   - Listar todos los alumnos, agrupados por materia cursada, dentro de cada año.
   - Listar todos los alumnos de un mismo docente.
   - Dado un alumno por su legajo, consultar su estado Activo/Inactivo.
   - Para aquellos alumnos que hace más de tres años que no se inscriben en ninguna cursada, pasar su dato Activo a falso (Inactivo).
   - Generar un par de archivos en el formato de /etc/passwd y /etc/shadow para todos los alumnos activos.
   - Generar un directorio /home/usuario para cada alumno activo, con UID correspondiente.

Accesibilidad para usuarios finales

1. Preparar un script con interfaz gráfica para copiar archivos seleccionados a una carpeta preestablecida con el fin de obtener un backup periódico de todos sus contenidos.
2. Preparar un script con interfaz gráfica que presente los cinco directorios con mayor ocupación de almacenamiento dentro del home del usuario.
3. Agregar interfaz gráfica a los scripts de administración de bases de datos de alumnos y materias.

Parte III

Configuraciones de Almacenamiento

1. Contenidos

1. Arquitectura de E/S, Dispositivos de E/S, Filesystems
2. Software RAID, instalación y mantenimiento, niveles 0, 1, 10
3. LVM, instalación y mantenimiento
4. Diseños típicos de almacenamiento

2. Dispositivos y filesystems

Los dispositivos lógicos de bloques están asociados a algún medio de almacenamiento, real o virtual. Ejemplos de dispositivos de bloques que encontramos con frecuencia son `/dev/sda`, `/dev/sda1`, `/dev/dvd`, etc.

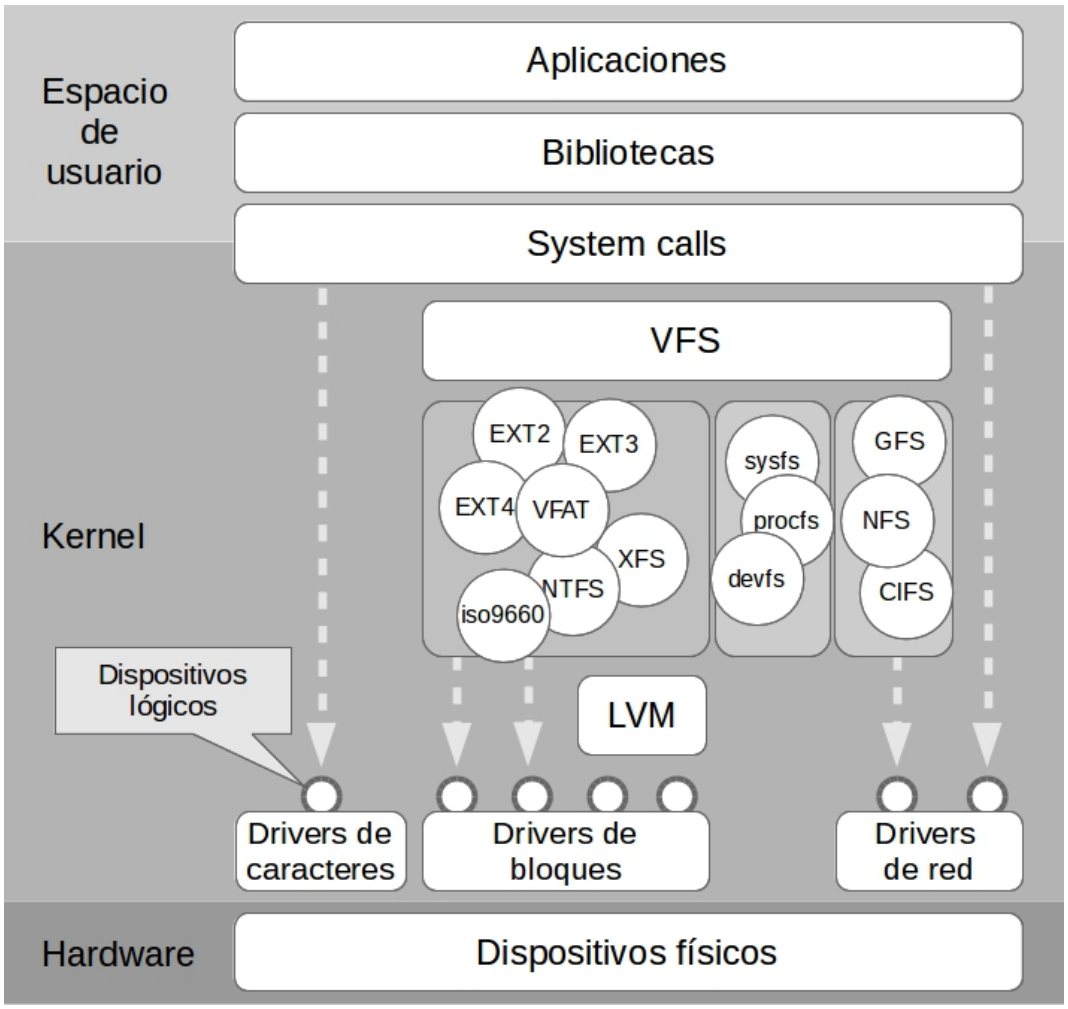

Figura 1: I/O y dispositivos

Presentan una interfaz que provee direccionamiento random o directo, es decir, sus bloques están numerados y se puede acceder a cualquier bloque con independencia de cuál haya sido accedido ante-

riormente (operación de *seek*). Pueden directamente contener un filesystem u ofrecer soporte a otros dispositivos virtuales, que los agrupan (como los dispositivos RAID) o en general los utilizan (como los dispositivos snapshot de LVM). Los típicos dispositivos de bloques con los que nos encontramos son los discos y las particiones, pero es interesante conocer otros dispositivos que están soportados por volúmenes lógicos, archivos, u otros, remotos, que se acceden por medio de la red.

Temas de práctica

1. Crear y destruir particiones con fdisk, parted, gparted.
2. Reconocer tipos de particiones. Comprender la estructura de la tabla de particiones, particiones primarias, extendidas y lógicas.
3. Comando dd, modificadores bs y count. Copia de dispositivos y archivos.
4. Dispositivos /dev/null y /dev/zero. Creación de archivos prealojados. Modificador seek.
5. Comando mkfs. Tipo de filesystem. Filesystems sobre una partición, sobre un archivo.
6. Loop devices. Comando losetup. Comando mount. Opciones ro, loop, offset. Montado de filesystems sobre una partición física, sobre un archivo, sobre una partición en una imagen de disco.
7. Redimensionamiento de filesystems. Comando dd y modificador conv=notrunc. Comando resize2fs. Opciones relacionadas con filesystems en parted.

Loop devices

```
$ dd if=/dev/zero of=imagen.img bs=1024 count=1024
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.00223564 s, 469 MB/s
$ ls -l imagen.img
-rw-r--r-- 1 root root 1048576 Sep 1 11:54 imagen.img
$ losetup /dev/loop0 imagen.img
$ losetup -a
/dev/loop0: [0808]:2260385 (/tmp/imagen.img)
$ mkfs -t ext3 /dev/loop0
mke2fs 1.42.8 (20-Jun-2013)

Filesystem too small for a journal
Discarding device blocks:       done
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
128 inodes, 1024 blocks
51 blocks (4.98%) reserved for the super user
First data block=1
Maximum filesystem blocks=1048576
1 block group
8192 blocks per group, 8192 fragments per group
128 inodes per group

Allocating group tables: 0/1 done
Writing inode tables: 0/1 done
Writing superblocks and filesystem accounting information: 0/1 done
```

```
$ mkdir mnt
$ mount -o loop /dev/loop0 mnt
$ df -h mnt
Filesystem     Size  Used Avail Use% Mounted on
/dev/loop0    1003K   17K  915K   2% /tmp/mnt
$ ls -l mnt
total 12
drwx------ 2 root root 12288 Sep 1 11:54 lost+found
$ ls / > mnt/lista.txt
$ ls -l mnt
total 13
-rw-r--r-- 1 root root 167 Sep 1 11:54 lista.txt
drwx------ 2 root root 12288 Sep 1 11:54 lost+found
$ df -h mnt
Filesystem     Size  Used Avail Use% Mounted on
/dev/loop0    1003K   18K  914K   2% /tmp/mnt
$ dd if=/dev/zero of=imagen.img bs=1024 count=1024 oflag=append conv=notrunc
1024+0 records in
1024+0 records out
1048576 bytes (1.0 MB) copied, 0.00206669 s, 507 MB/s
$ ls -l imagen.img
-rw-r--r-- 1 root root 2097152 Sep 1 11:54 imagen.img
$ losetup -c /dev/loop0
$ losetup -a
/dev/loop0: [0808]:2260385 (/tmp/imagen.img)
/dev/loop1: [0005]:5178 (/dev/loop0)
$ umount mnt
$ e2fsck -fp /dev/loop0
/dev/loop0: 12/128 files (0.0% non-contiguous), 39/1024 blocks
$ resize2fs /dev/loop0
resize2fs 1.42.8 (20-Jun-2013)
Resizing the filesystem on /dev/loop0 to 2048 (1k) blocks.
The filesystem on /dev/loop0 is now 2048 blocks long.

$ mount -o loop /dev/loop0 mnt
$ df -h mnt/
Filesystem     Size  Used Avail Use% Mounted on
/dev/loop0     2.0M   18K  1.9M   1% /tmp/mnt
```

3. RAID

Los *arrays* RAID (Redundant Array of Independent Disks) son dispositivos virtuales creados como combinación de dos o más dispositivos físicos. El dispositivo virtual resultante puede contener un filesystem.

Los diferentes modos de combinación de dispositivos, llamados niveles RAID, ofrecen diferentes características de redundancia y performance. Un array RAID con redundancia ofrece protección contra fallos de dispositivos.

Los dispositivos Software RAID de Linux son creados y manejados por el driver `md` y por eso suelen recibir nombres como `md0`, `md1`, etc.

- Redundancia para tolerancia a fallos
- Mejoramiento de velocidad de acceso
- Niveles

- Hardware RAID, fake RAID, Software RAID
- Devices
- Spare disks, faulty disks

Niveles RAID

Linear mode Dos o más dispositivos concatenados. La escritura de datos ocupa los dispositivos en el orden en que son declarados. Sin redundancia. Mejora la performance cuando diferentes usuarios acceden a diferentes secciones del file system, soportadas en diferentes dispositivos.

RAID-0 Las operaciones son distribuidas (*striped*) entre los dispositivos, alternando circularmente entre ellos. Cada dispositivo se accede en paralelo, mejorando el rendimiento. Sin redundancia.

RAID-1 Dos o más dispositivos replicados (*mirrored*), con cero o más *spares*. Con redundancia. Los dispositivos deben ser del mismo tamaño. Si existen *spares*, en caso de falla o salida de servicio de un dispositivo, el sistema reconstruirá automáticamente una réplica de los datos sobre uno de ellos. En un RAID-1 de N dispositivos, pueden fallar o quitarse hasta $N-1$ de ellos sin afectar la disponibilidad de los datos. Si N es grande, el bus de I/O puede ser un cuello de botella (al contrario que en Hardware RAID-1). El scheduler de Software RAID en Linux asigna las lecturas a aquel dispositivo cuya cabeza lectora está más cerca de la posición buscada.

RAID-4 No se usa frecuentemente. Usado sobre tres o más dispositivos. Mantiene información de paridad sobre un dispositivo, y escribe sobre los restantes en la misma forma que RAID-0. El tamaño del array será $(N-1)*S$, donde S es el tamaño del dispositivo de menor capacidad en el array. Al fallar un dispositivo, los datos se reconstruirán automáticamente usando la información de paridad. El dispositivo que soporta la paridad se constituye en el cuello de botella del sistema.

RAID-5 Utilizado sobre tres o más dispositivos con cero o más *spares*. El tamaño del dispositivo RAID será $(N-1)*S$. La diferencia con RAID-4 es que la información de paridad se distribuye entre los dispositivos, eliminando el cuello de botella de RAID-4 y obteniendo mejor performance en lectura. Al fallar uno de los dispositivos, los datos siguen disponibles. Si existen *spares*, el sistema reconstruirá automáticamente el dispositivo faltante. Si se pierden dos o más dispositivos simultáneamente, o durante una reconstrucción, los datos se pierden definitivamente. RAID-5 sobrevive a la falla de un dispositivo, pero no de dos o más. La performance en lectura y escritura mejora con respecto a un solo dispositivo.

RAID-6 Usado sobre cuatro o más dispositivos con cero o más *spares*. La diferencia con RAID-5 es que existen dos diferentes bloques de información de paridad, distribuidos entre los dispositivos participantes, mejorando la robustez. El tamaño del dispositivo RAID-6 es $(N-2)*S$. Si fallan uno o dos de los dispositivos, los datos siguen disponibles. Si existen *spares*, el sistema reconstruirá automáticamente los dispositivos faltante. La performance en lectura es similar a RAID-5, pero la de escritura no es tan buena.

RAID-10 Combinación de RAID-1 y RAID-0 completamente ejecutada por el kernel, más eficiente que aplicar dos niveles de RAID independientemente. Es capaz de aumentar la eficiencia en lectura de acuerdo a la cantidad de dispositivos, en lugar de la cantidad de pares RAID-1, ofreciendo un 95 % del rendimiento de RAID-0 con la misma cantidad de dispositivos. Los *spares* pueden ser compartidos entre todos los pares RAID-1.

FAULTY Nivel especial de RAID que sirve para debugging del array por inyección de fallos de lectura y escritura. Sólo permite un dispositivo. Simula fallos a bajo nivel, permitiendo analizar comportamiento en caso de fallos de sector en lugar de fallos de discos.

Construcción y uso de un array RAID-1

Crear particiones en ambos discos, tipo fd (Linux RAID autodetect)

```
fdisk /dev/sdb; fdisk /dev/sdc
```

Crear el array

```
mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
watch cat /proc/mdstat
```

Usar el array

```
mkfs -t ext3 /dev/md0
mkdir /datos
mount -t ext3 /dev/md0 /datos
cp /etc/hosts /datos
ll /datos
```

Examinar el array

```
cat /proc/mdstat
cat /proc/partitions
mdadm --examine --brief --scan --config=partitions
mdadm --examine /dev/sdc
mdadm --query --detail /dev/md0
```

Crear script de alerta

```
cat > /root/raidalert
#!/bin/bash
echo $(date) $* >> /root/alert
^D
chmod a+x /root/raidalert
```

Monitorear el arreglo con script de alerta

```
mdadm --monitor -1 --scan --config=partitions --program=/root/raidalert
```

Crear configuración

```
cat > /etc/mdadm.conf
DEVICE=/dev/sdb1 /dev/sdc1
ARRAY=/dev/md0 devices=/dev/sdb1,/dev/sdc1
PROGRAM=/root/raidalert
```

Establecer tarea periódica de monitoreo

```
crontab -e
MAILTO=""
*/2 * * * * /sbin/mdadm --monitor -1 --scan
```

Declarar un fallo

```
mdadm /dev/md0 -f /dev/sdb1
cat /root/alert
```

Quitar un disco del array

```
mdadm /dev/md0 -r /dev/sdb1
cat /root/alert
```

Reincorporar el disco al array

```
mdadm /dev/md0 -a /dev/sdb1
cat /proc/mdstat
cat /root/alert
```

Destruir el array

```
mdadm --stop /dev/md0
```

4. Administración de LVM

Introducción a LVM

El soporte habitual para los file systems de servidores son los discos magnéticos, particionados según un cierto diseño definido al momento de la instalación del sistema. Las particiones se definen a nivel del hardware. El conjunto de aplicaciones y servicios del sistema utiliza los filesystems que se instalan sobre estas particiones.

Las particiones de disco son un concepto de hardware, y dado que las unidades de almacenamiento se definen estáticamente al momento del particionamiento, presentan un problema de administración a la hora de modificar sus tamaños.

El diseño del particionamiento se prepara para distribuir adecuadamente el espacio de almacenamiento entre los diferentes destinos a los que se dedicará el sistema. Sin embargo, es frecuente que el patrón de uso del sistema vaya cambiando, y el almacenamiento se vuelva insuficiente o quede distribuido en forma inadecuada. La solución a este problema implica normalmente el reparticionamiento de los discos, operación que obliga a desmontar los filesystems y a interrumpir el servicio. Para redimensionar una partición, normalmente es necesario el reboot del equipo, con la consiguiente interrupción del servicio en producción.

La alternativa consiste en interponer una capa intermedia de software entre el hardware crudo, con sus particiones, y los filesystems sobre los que descansan los servicios. Esta capa intermedia está implementada por Logical Volume Manager (LVM). LVM es un subsistema orientado a flexibilizar la administración de almacenamiento, al interponer una capa de software que implementa dispositivos de bloques lógicos por encima de las particiones físicas.

Usando LVM, el almacenamiento queda estructurado en capas, y las unidades lógicas pueden crearse, redimensionarse, o destruirse, sin necesidad de reboot, desmontar ni detener el funcionamiento del sistema. Con LVM pueden definirse por software contenedores de filesystems, de límites flexibles, que admiten el redimensionamiento "en caliente", es decir sin salir de actividad, mejorando la disponibilidad general de los servicios.

Con LVM pueden agregarse unidades físicas mientras el hardware lo permita, extendiéndose dinámicamente las unidades lógicas y redistribuyendo el espacio disponible a conveniencia. Presenta también otras ventajas como la posibilidad de extraer *snapshots* o instantáneas de un filesystem en funcionamiento (para obtener backups consistentes a nivel de filesystem), y manipular con precisión el mapeo a unidades físicas para aprovechar características del sistema (como *striping* sobre diferentes discos).

Componentes de LVM

En la terminología LVM, los dispositivos de bloques entregados al sistema LVM se llaman PV (physical volumes). Cualquier dispositivo de bloques escribible puede convertirse en un PV de LVM. Esto incluye particiones de discos y dispositivos múltiples como conjuntos RAID. Los PVs se agrupan en VGs (volume groups) que funcionan como *pools* de almacenamiento físico. De cada pool pueden extraerse a discreción LVs (logical volumes), que se comportan nuevamente como dispositivos de bloques, y que pueden, por ejemplo, alojar filesystems. Estos serán los usuarios finales de la jerarquía (Fig. 2).

Conviene tener en mente la jerarquía de los siguientes elementos:

Volumen físico o PV (physical volume) Es un contenedor físico que ha sido agregado al sistema

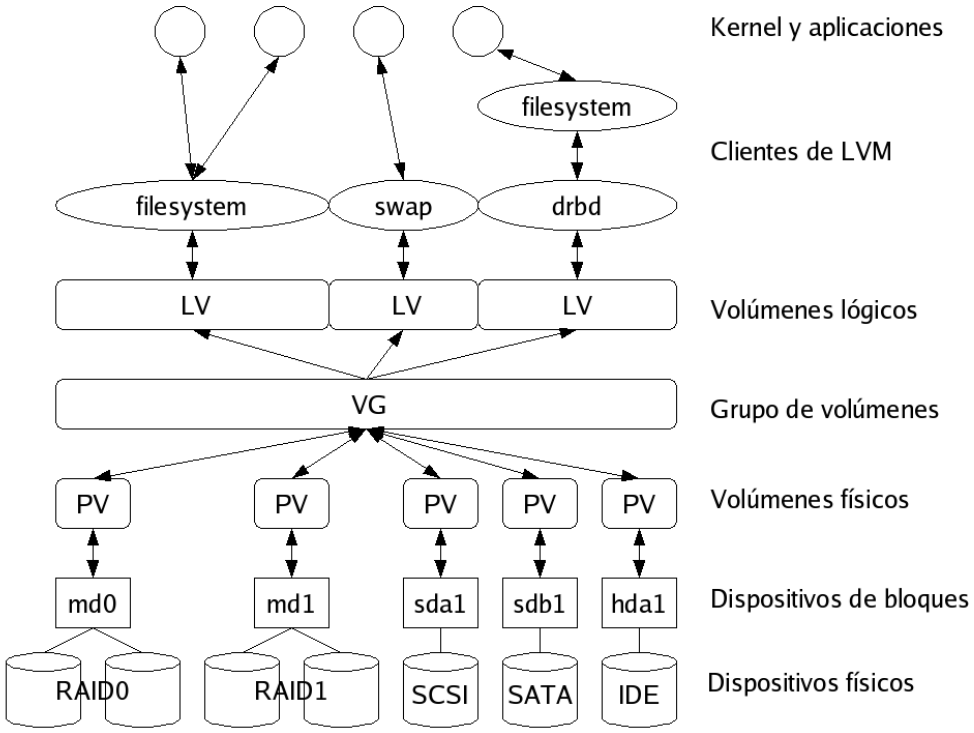

Figura 2: Jerarquía de componentes LVM

LVM. Puede ser una partición u otro dispositivo de bloques adecuado.

Grupo de volúmenes o VG (volume group) Es un pool o repositorio de espacio conformado por uno o varios PVs. Un VG ofrece un espacio de almacenamiento virtualmente continuo, cuyo tamaño corresponde aproximadamente a la suma de los PVs que lo constituyen. Los límites entre los PVs que conforman un VG son transparentes.

Volumen lógico o LV (logical volume) Es una zona de un VG que ha sido delimitada para ser usada por otro software, como por ejemplo un filesystem. Los tamaños de los LVs dentro de un VG no necesariamente coinciden con los de los PVs que los soportan.

Uso de LVM

Los pasos lógicos para utilizar almacenamiento bajo LVM son:

- Crear uno o más PVs a partir de particiones u otros dispositivos.
- Reunir los PVs en un VG con lo cual sus límites virtualmente desaparecen.
- Particionar lógicamente el VG en uno o más LVs y utilizarlos como normalmente se usan las particiones.

El sistema LVM incluye comandos para realizar estas tareas y en general administrar todas estas unidades. Con ellos se puede, dinámicamente:

- Redimensionar LVs de modo de ocupar más o menos espacio dentro del VG.
- Aumentar la capacidad de los VGs con nuevos PVs sin detener el sistema.
- Mover LVs a nuevos PVs, más rápidos, sin detener el sistema.
- Usar *striping* entre PVs de un mismo VG para mejorar las prestaciones.
- Tomar una instantánea o *snapshot* de un LV para hacer un backup del filesystem contenido en el LV.
- Tomar una instantánea como medida preventiva antes de una actualización o modificación.

Los comandos tienen nombres con los prefijos `pv`, `vg`, `lv`, etc. Además, el comando `lvm` ofrece una consola donde se pueden dar esos comandos y pedir ayuda.

Redimensionamiento de volúmenes

Una vez creado un LV, su capacidad puede ser reducida o aumentada (siempre que exista espacio extra en el VG que lo contiene). Si el LV redimensionado contuviera un filesystem, éste también debe ser redimensionado en forma acorde.

- Si un filesystem va a ser extendido, primero debe extenderse el LV que lo contiene.
- Si un filesystem va a ser reducido, luego debe reducirse el LV que lo contiene.
- Si un LV que va a ser reducido está ocupado en un porcentaje, la reducción del LV sólo puede llevarse a cabo en forma segura en dicho porcentaje.

Los filesystems ext3 y ext4 cuentan con una herramienta, `resize2fs`, que es capaz de redimensionarlos sin detener la operación.

Snapshots y backups

Un snapshot es un LV virtual, especialmente preparado, asociado a un LV original cuyo estado se necesita "congelar" para cualquier propósito de mantenimiento. Una vez creado el snapshot, mediante un mecanismo de *copy-on-write (o COW)*, LVM provee una instantánea o vista inmutable del filesystem original, aunque éste se actualice. Una vez creado el LV virtual de snapshot, sus contenidos son estáticos y permanentemente iguales al LV original. Puede ser montado y usado como un filesystem corriente. El snapshot es temporario y una vez utilizado se descarta.

La motivación principal del mecanismo de snapshots es la extracción de copias de respaldo. Durante la operación del sistema, las aplicaciones y el kernel leen y escriben sobre archivos, y por lo tanto el filesystem pasa por una sucesión de estados. Una operación de backup que se desarrolle concurrentemente con la actividad del filesystem no garantiza la consistencia de la imagen obtenida, ya que archivos diferentes pueden ser copiados en diferentes momentos, bajo diferentes estados del filesystem. Como consecuencia, la imagen grabada no necesariamente representa un estado concreto de la aplicación; y esto puede dar lugar a problemas al momento de la recuperación del backup.

Hay muchos escenarios posibles de inconsistencia. Algunos ejemplos son:

- Una aplicación que mantiene un archivo temporario en disco con modificaciones automáticas y periódicas (por ejemplo, `vi`).
- Aplicaciones que mantienen conjuntos de archivos fuertemente acoplados (bases de datos compuestas por tablas e índices en archivos separados).
- Una instalación de un paquete de software, que suele afectar muchos directorios del sistema.

Una solución consiste en "congelar" de alguna forma el estado del filesystem durante la operación de copia (por ejemplo, desmontándolo). Con LVM, gracias al mecanismo de *COW*, esta instantánea puede ser obtenida sin detener la operación del LV original, o sea sin afectar la disponibilidad del servicio. La operación con el filesystem del LV origen (LVO) no se interrumpe, ni modifica en nada la conducta de las aplicaciones que lo estén usando.

Para la creación de un snapshot de un LVO se necesita contar con espacio extra disponible dentro del mismo VG al cual pertenece el LVO. Este espacio extra no necesita ser del mismo tamaño que el LVO[1]. Normalmente es suficiente un 15 % a 20 % del tamaño del LV original. Si el VG no tiene suficiente espacio, puede extenderse.

[1] No es fácil determinar con precisión este tamaño, ya que debe ser suficiente para contener todos los bloques modificados en el LVO durante el tiempo en que se use el snapshot; y este conjunto puede ser variable, dependiendo del patrón de uso del LVO y del medio hacia el cual se pretende hacer el backup (otro disco local, un servidor en la red local, un servidor en una red remota, una unidad de cinta, tienen diferente ancho de banda y diferente demora de grabación).

Creación de snapshots

Crear un snapshot es preparar un nuevo LV, virtual, con un filesystem virtualmente propio, que se monta en un punto de montado diferente del original (Fig. 3).

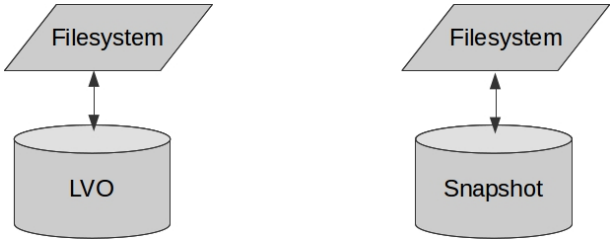

Figura 3: Un LVO y su snapshot

A partir de este momento se pueden hacer operaciones de lectura y escritura en ambos filesystems separadamente, con efectos distintos en cada caso.

Lectura y escritura del LVO

Los snapshots son creados tomando un espacio de bloques de datos (la tabla de excepciones) dentro del mismo VG del LVO. Mientras un bloque no sea modificado, las operaciones de lectura lo recuperarán del LVO. Pero, cada vez que se modifique un bloque del LVO, la versión original, sin modificar, de dicho bloque, será copiada en el snapshot (Fig. 4).

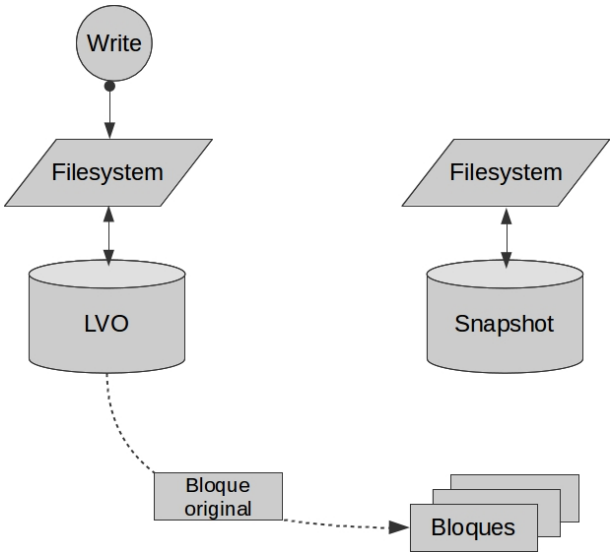

Figura 4: Escritura de un bloque del LVO

Lectura y escritura del snapshot

De esta manera, el snapshot muestra siempre los contenidos originales del LVO (Fig. 5), salvo que se modifiquen por alguna operación de escritura en el snapshot.

Los LVs pueden declararse R/O o R/W. En LVM2, los snapshots son R/W por defecto. Al escribir sobre un filesystem de un LV snapshot R/W, se grabará el bloque modificado en el espacio privado del snapshot sin afectar el LVO (Fig. 6). Al eliminar el snapshot, todas las modificaciones hechas sobre el mismo desaparecen.

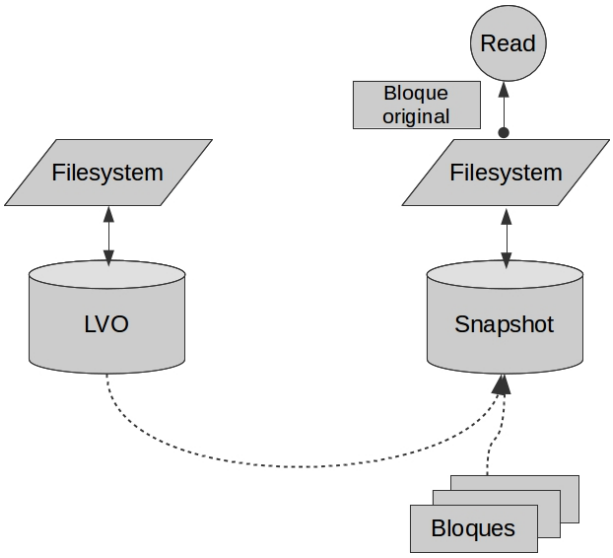

Figura 5: Lectura de un bloque desde el snapshot

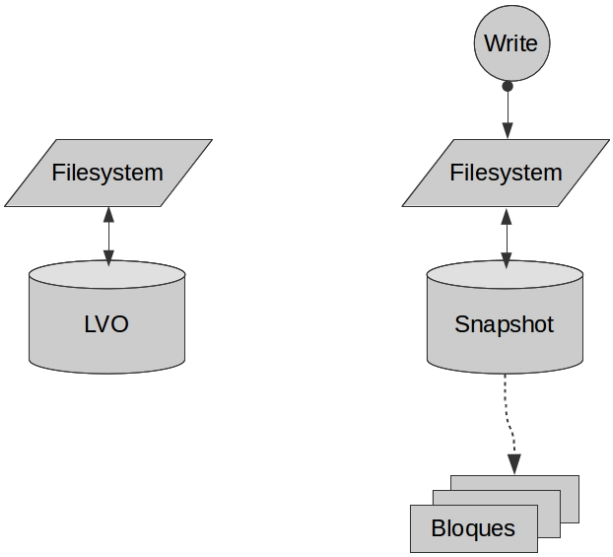

Figura 6: Escritura sobre el snapshot

Creación de backups con snapshots

1. Se crea un snapshot del LV de interés.
2. Se monta el snapshot en modo RO sobre un punto de montado conocido.
3. Se copian los archivos del snapshot con la técnica de backup que se desee.
4. Se verifica la integridad del backup.
5. Si la verificación no fue satisfactoria, se repite el backup.
6. En caso satisfactorio, el snapshot se destruye con `lvremove <ID del snapshot>`.

Uso preventivo de snapshots

El mecanismo de snapshots ofrece la posibilidad de recuperar el estado original del LVO al efectuar operaciones que pueden afectar críticamente la integridad u operatividad del sistema, como pruebas de instalación, etc.

Como se ha visto, el conjunto de bloques almacenados en un snapshot está formado por bloques que, o bien provienen del LVO original, en su estado anterior a ser modificados (Fig. 4), o bien, fueron modificados en el snapshot creado y montado en modo R/W (Fig. 6).

En ambos casos, este conjunto de bloques puede volver a ser aplicado sobre el LVO (o revertido) con la opción `--merge` del comando `lvconvert`. El resultado, sin embargo, será sumamente diferente en uno y otro caso.

Si se aplica el snapshot *sin modificaciones*, el LVO vuelve al estado original al momento de ser tomado el snapshot. Si los bloques *han sido modificados*, al revertir el snapshot el LVO cambia, asumiendo las modificaciones que se hayan hecho en el snapshot. Ambas propiedades pueden ser útiles para recuperar el estado después de una modificación que no ha sido satisfactoria, pero la forma de aplicar las operaciones es distinta.

Una técnica consiste en:

1. Crear el snapshot del LVO, con espacio suficiente para registrar las modificaciones que se piensa hacer.
2. Ejecutar sobre el LVO las operaciones críticas (instalar, actualizar, modificar).
3. Verificar el resultado.
4. Si el resultado fue satisfactorio, destruir el snapshot con `lvremove <ID del snapshot>`.
5. Si el resultado no fue satisfactorio, recuperar el estado original del LVO con el comando `lvconvert --merge <ID del snapshot>`

. Este comando restituye (*roll-back*) al LVO todos sus bloques originales, que fueron grabados en el espacio de excepciones del snapshot, y luego destruye el snapshot.

Con la técnica anterior, las modificaciones se realizan sobre el filesystem del LVO y en caso necesario se revierten. Una segunda posibilidad es dejar el LVO fuera de línea y trabajar sobre el snapshot R/W.

Para esto se crea el snapshot, se desmonta el LVO, se monta en su lugar el snapshot, se realizan las modificaciones, y se verifica el resultado. Si fue satisfactorio, se vuelcan las modificaciones al LVO con `lvconvert --merge <ID del snapshot>`. Si no, se elimina el snapshot. Finalmente, en uno u otro caso, se vuelve a montar el LVO en su lugar.

Eliminación del snapshot

El snapshot debe ser destruido al finalizar el backup o terminar de usarlo, ya que, al obligar a copiar cada bloque del LVO que se modifica, representa un costo en performance del sistema de I/O. Por lo demás, el snapshot se define con una cierta capacidad, que al ser excedida hace inutilizable el snapshot completo.

Ejemplos LVM

Creación de Physical Volumes (PV)

```
fdisk /dev/sdb
pvcreate /dev/sdb1 /dev/sdb2
pvdisplay
```

Creación de Volume Groups (VG)

```
vgcreate vg0 /dev/sdb1 /dev/sdb2
vgdisplay
```

Creación de Logical Volumes (LV)

```
lvcreate --size 512M vg0 -n lvol0
lvcreate -l 50%VG vg0 -n lvol1
lvcreate -l 50%FREE vg0 -n lvol2
lvdisplay vg0
```

Examinar LVM

```
pvs
vgs
lvs
pvscan
vgscan
lvscan
```

Uso de volúmenes

```
mkfs -t ext3 /dev/vg0/lvol0
mkdir volumen
mount /dev/vg0/lvol0 volumen
cp *.gz volumen
ls -l volumen
```

Extensión de un volumen

```
umount /dev/vg0/lvol0
lvextend --size +1G vg0/lvol0
mount /dev/vg0/lvol0 volumen
resize2fs /dev/vg0/lvol0
```

Agregar un disco al sistema

```
fdisk /dev/hdd
pvcreate /dev/hdd1
vgextend vg0 /dev/hdd1
lvextend --size +1G vg0/lvol0
ext2online /dev/vg0/lvol0
```

Snapshot de un volumen

```
lvcreate -s -n snap --size 100M vg0/lvol0
ls -l /dev/vg0
mkdir volumen-snap
```

```
mount /dev/vg0/snap volumen-snap
ls -l volumen-snap/
rm volumen/archivo1.tar.gz volumen/archivo2.tar.gz
ls -l volumen-snap/
```

Destruir un snapshot

```
lvremove vg0/snap
```

Temas de práctica

1. Crear una partición, convertirla en PV, crear un VG y definir un LV `lv0` dentro del mismo dejando un 25 % del espacio libre. Crear un filesystem sobre el LV, montarlo y utilizarlo para administrar archivos.
2. Definir un nuevo LV `lv1` en el mismo VG creado anteriormente, ocupando la totalidad del espacio del VG.
3. Crear otra partición en el mismo u otro medio de almacenamiento, convertirla en PV y adjuntarla al VG del ejercicio anterior. Examinar el resultado de las operaciones con los comandos de revisión correspondientes.
4. Extender el LV `lv1` para ocupar nuevamente la totalidad del espacio del VG extendido. Crear un filesystem sobre el LV, montarlo y utilizarlo para administrar archivos.
5. Modificar los tamaños de ambos LVs, extendiendo uno y reduciendo el otro. Recordar que al reducir un LV se debe primero reducir el filesystem alojado, y que para extender un filesystem se debe primero extender el LV que lo aloja. Comprobar que los filesystems alojados siguen siendo funcionales.
6. Supongamos que, al querer crear un snapshot de un LV, el administrador recibe un mensaje de error diciendo que el VG no cuenta con espacio disponible. Sugiera un método para enfrentar este problema usando LVM.
7. Dado un LV, ponga en práctica las técnicas de creación de snapshot para a) obtener un backup, y b) realizar modificaciones sobre el LV volviendo después al estado original.

Parte IV

Estrategias de Respaldo

Parte V

Virtualización

Parte VI

Alta Disponibilidad