Administración de Sistemas Avanzada

Eduardo Grosclaude

2014-08-11

[2014/11/27, 12:43:23- 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**.

Índice

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.
   - Heartbeat, DRBD, Clustering de aplicaciones

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.
- K. Kopper, The Linux Enterprise Cluster: build a highly available cluster with commodity hardware and free software. San Francisco: No Starch Press, 2005.
- R. Pollei, Debian 7 System Administration Best Practices. Birmingham: Packt Publishing, 2013.
- T. A. Limoncelli, C. J. Hogan, and S. R. Chalup, The practice of system and network administration. Upper Saddle River, N.J: Addison-Wesley, 2008.
- Chen, Peter M., et al., RAID: High-performance, reliable secondary storage. ACM Computing Surveys (CSUR), 1994.
- S. van Vugt, Pro Linux high availability clustering. 2014.

Biblioteca Virtual del MinCyT

Biblioteca Virtual del MinCyT: http://www.biblioteca.mincyt.gob.ar/libros.
Títulos accesibles desde la UNC:

- C. Wolf and E. M. Halter, Virtualization from the desktop to the enterprise. Berkeley, CA; New York, NY: Apress; Distributed in U.S. by Springer-Verlag New York, 2005; http://rd.springer.com/book/10.1007/978-1-4302-0027-7.
- K. Schmidt, High availability and disaster recovery concepts, design, implementation. Berlin; Springer, 2006; http://rd.springer.com/book/10.1007/3-540-34582-5.

5. Cronograma

Fecha	Semana	Unidad	Trabajo I	Trabajo II
11/8	1	1. Scripting Avanzado		
18/8	2			
25/8	3			
1/9	4	2. Configuraciones de Almacenamiento		
8/9	5			
15/9	6			
22/9	7			
29/9	8	Parcial I		
6/10	9	3. Estrategias de respaldo		
13/10	10			
20/10	11	4. Virtualización		
27/10	12			
3/11	13	5. Alta Disponibilidad		
10/11	14			
17/11	15			
24/11	16	Parcial II		

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, A)
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.

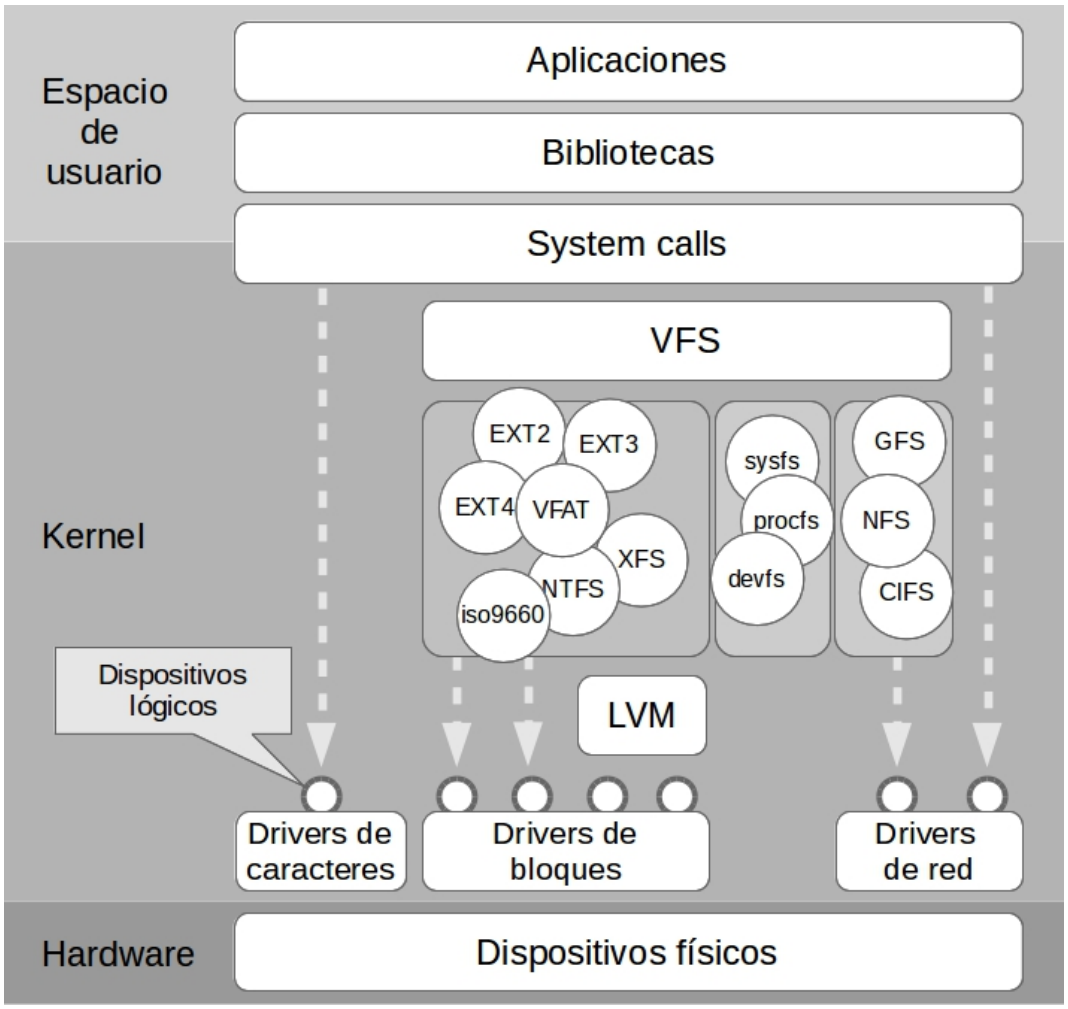

Figura 1: I/O y dispositivos

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` (Multiple Device) y por eso suelen recibir nombres como `md0`, `md1`, etc.

- Redundancia para tolerancia a fallos
- Mejoramiento de velocidad de acceso
- Hardware RAID, Fake RAID, Software RAID

- Niveles RAID
- 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.

Modos de operación

Create Creación de un array nuevo con superblocks por cada dispositivo.

Assemble Ensamblar dispositivos componentes previamente creados para conformar un dispositivo RAID activo. Los componentes pueden especificarse en el comando o ser identificados por scanning.

Follow o Monitor Monitorizar un dispositivo RAID y actuar en caso de eventos interesantes. No se aplica a RAID niveles 0 ni linear, ya que sus componentes no poseen estados (*failed*, *spare*, *missing*).

Build Construir un array sin superblocks por dispositivo (para expertos).

Grow Extender o reducir un array, cambiando el tamaño de los componentes activos (RAID 1, 4, 5 y 6) o cambiando el número de dispositivos activos en RAID1.

Manage Manipular componentes específicos de un array, como al agregar nuevos dispositivos *spare* o al eliminar dispositivos *faulty*.

Misc Toda otra clase de operaciones sobre arrays activos o sus componentes.

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
```

Temas de práctica

1. ¿Qué marcas, modelos y precios de tarjetas controladoras RAID puede encontrar? ¿Con qué capacidades?
2. ¿Qué diferencias hay entre Software RAID y Hardware RAID?
3. ¿Qué niveles RAID ofrecen redundancia? ¿Contra qué eventos protege esta redundancia? ¿Contra qué eventos *no* protege esta redundancia?
4. El uso de un dispositivo RAID, ¿es un reemplazo efectivo para las políticas y actividades de backup?
5. ¿Cuáles niveles RAID ofrecen mejor velocidad de trabajo? ¿De qué factores depende la ventaja en performance de los diferentes niveles RAID entre sí y con respecto al uso de una única unidad de disco?
6. ¿Cuál es la diferencia entre los niveles Linear RAID y RAID 0? ¿Qué clase de redundancia ofrece cada uno? ¿Contra qué eventos protege?
7. Preparar un arreglo linear RAID sobre dos dispositivos loop. Observe qué relación tiene el espacio disponible en el dispositivo con los archivos que soportan los dispositivos loop.
8. Preparar un arreglo linear RAID sobre dos discos extra agregados al equipo.
9. Preparar un arreglo RAID nivel 0 sobre dos discos extra agregados al equipo.
10. ¿Puede medir la diferencia en performance entre los dispositivos de los ejercicios anteriores, de linear RAID y de nivel 0? ¿Tiene sentido esta medición cuando el equipo es una máquina virtual?
11. Preparar un RAID nivel 1 sobre dos discos extra. Inyectar un fallo en uno de los discos. Agregar un nuevo disco e incorporarlo al RAID. Observar la sincronización del dispositivo.
12. Como antes, preparar un RAID nivel 1 sobre dos discos extra, pero con una unidad *spare*. Inyectar un fallo en uno de los discos y observar el comportamiento del dispositivo.
13. Retire el disco fallado y compruebe en qué estado queda el dispositivo.
14. Vuelva a agregar el disco y compruebe en qué estado queda el dispositivo.
15. ¿Con qué comandos se investiga el estado de un dispositivo RAID? ¿Cómo se sabe cuándo un dispositivo RAID está activo o en modo degradado? ¿Cómo se sabe cuándo un dispositivo está fallado, activo, sincronizando?

16. ¿Cuál es la forma de convertir en dispositivo RAID 1 un filesystem ya existente?
17. ¿Cómo se puede adaptar el comportamiento de un RAID 1 a una situación con discos asimétricos (uno más rápido que el otro)?

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).

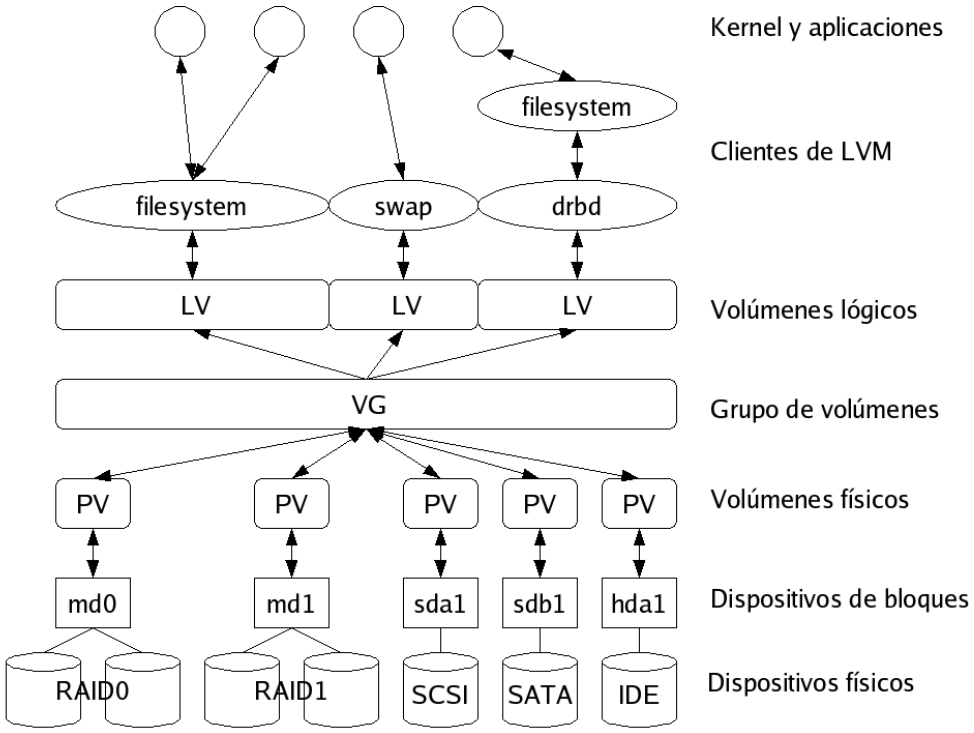

Figura 2: Jerarquía de componentes LVM

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 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.

Original	Comando	Final
50 GB	`lvresize -r -L 10G vg0/lv0`	10GB
	`lvresize -r -L -10G vg0/lv0`	40GB
	`lvreduce -r -L -10G vg0/lv0`	40GB
	`lvresize -r -L +10G vg0/lv0`	60GB

Cuadro 1: Tamaños con y sin signo

- 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) con los comandos `lvreduce` o `lvresize`. Si el LV redimensionado contuviera un filesystem, éste también debe ser redimensionado en forma acorde.

- Si un filesystem debe 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 (en caso contrario, el espacio extra se desperdicia).
- Si un LV va a ser reducido, primero debe reducirse el filesystem que contiene.
- Si un LV va a ser extendido, luego debe extenderse el fileystem que contiene (en caso contrario, el espacio extra se desperdicia).
- Si un LV que va a ser reducido está ocupado en un cierto porcentaje, la reducción del LV sólo puede llevarse a cabo en forma segura en dicho porcentaje.

Este redimensionamiento del filesystem puede ser hecho por el administrador con un comando separado, o automáticamente en el momento de redimensionar el LV.

Redimensionamiento automático

La herramienta `resize2fs`, que modifica el tamaño de un filesystem, es capaz de extender los filesystems sin necesidad de desmontar. Sin embargo, para reducir el tamaño de un filesystem, será necesario desmontarlo.

Aún más conveniente que `resize2fs` es la opción -r de los comandos lvreduce y lvresize. Esta opción tiene en cuenta el tamaño final del LV que se está redimensionando, y modifica el tamaño del filesystem contenido adecuadamente, todo en la misma secuencia de operación.

Indicación del nuevo tamaño

La opción -L (o –size) indica el nuevo tamaño del LV en bytes, con sufijos M (mega), G (giga), T (tera), etc.

En el comando `lvresize`, si el tamaño se indica con un prefijo de signo más o menos, significa que el tamaño debe aumentar o disminuir en la cantidad que se indica a continuación. En cambio, si el tamaño no lleva prefijo, significa que esa cantidad debe ser el tamaño final del LV. En el comando `lvreduce` sólo tiene sentido el signo menos (ejemplos en Cuadro 1).

Original	Comando	Final	
40 GB	`lvresize -r -l 1000 vg0/lv0`	4 GB	1000 extents = 1000 * 4 MB = 4 GB
	`lvresize -r -l +1000 vg0/lv0`	44 GB	40 GB + 4 GB = 44 GB
	`lvresize -r -l +100%LV vg0/lv0`	80 GB	Se duplica el tamaño del LV
	`lvresize -r -l -10%LV vg0/lv0`	36 GB	Se reduce el LV en un 10 %
	`lvresize -r -l +10%VG vg0/lv0`	50 GB	Se agrega un 10 % del total del VG
	`lvresize -r -l +10%FREE vg0/lv0`	46 GB	Se agrega un 10 % del espacio libre del VG
	`lvresize -r -l +10%ORIGIN vg0/snap`	14 GB	Si el snapshot era de 10 GB (un 25 % del LVO), lo extiende en un 10 % del LVO

Cuadro 2: Uso de extents

Bytes y extents

El nuevo tamaño para un LV puede darse en múltiplos del byte (MB, GB, TB...) o en *extents* (la unidad mínima de asignación de bloques de LVM, por defecto 4 MB). La opción -L (o –size) indica el tamaño deseado en bytes, y -l (o –extents) en extents.

Al usar extents, se puede opcionalmente indicar el tamaño deseado en términos relativos del tamaño del VG, del LV, o del espacio libre dentro del VG. Por ejemplo, supongamos que:

- el volumen lógico `vg0/lv0` mide 40 GB,
- el VG `vg0` que lo contiene mide 100 GB,
- el espacio del VG `vg0` no ocupado por el LV `vg0/lv0` está libre.

Bajo estas condiciones, diferentes comandos `lvresize` tienen los efectos que se ven en el Cuadro 2.

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.

Dimensionamiento del snapshot

Para la creación de un snapshot de un LVO se necesita contar con espacio extra disponible (no utilizado por ningún LV) dentro del mismo VG al cual pertenece el LVO. Este espacio extra no necesita ser del mismo tamaño que el LVO. Normalmente es suficiente un 15 % a 20 % del tamaño del LV original. Si el VG no tiene suficiente espacio, debe extenderse previamente.

No es fácil determinar con precisión el tamaño del snapshot, 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 de bloques 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).

En el Cuadro 2 hay un ejemplo de redimensionamiento del LV snapshot en función del tamaño del LVO.

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).

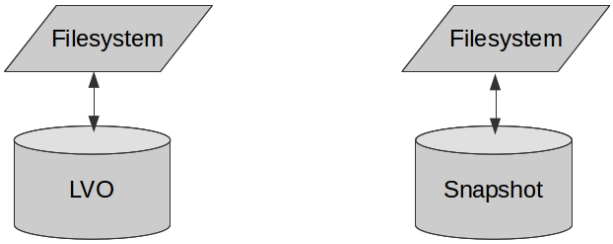

Figura 3: Un LVO y su snapshot

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).

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.

Creación de backups con snapshots

1. Se crea un snapshot del LV de interés.
2. Se monta el snapshot en modo R/O 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.

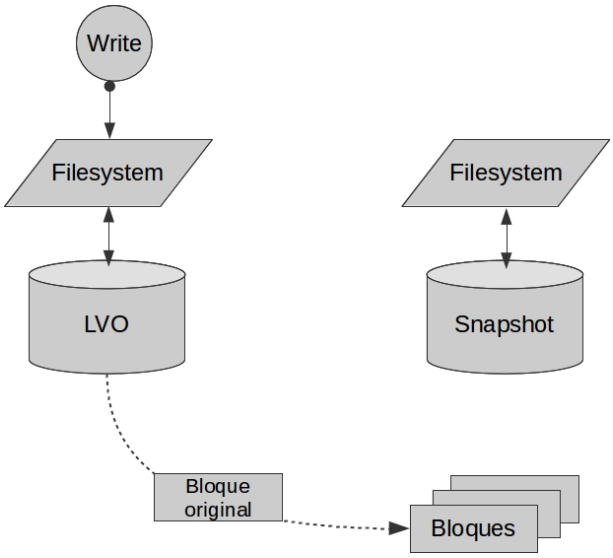

Figura 4: Escritura de un bloque del LVO

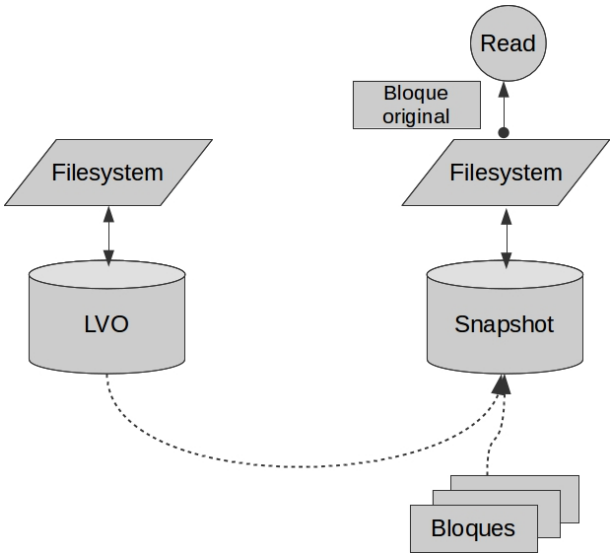

Figura 5: Lectura de un bloque desde el snapshot

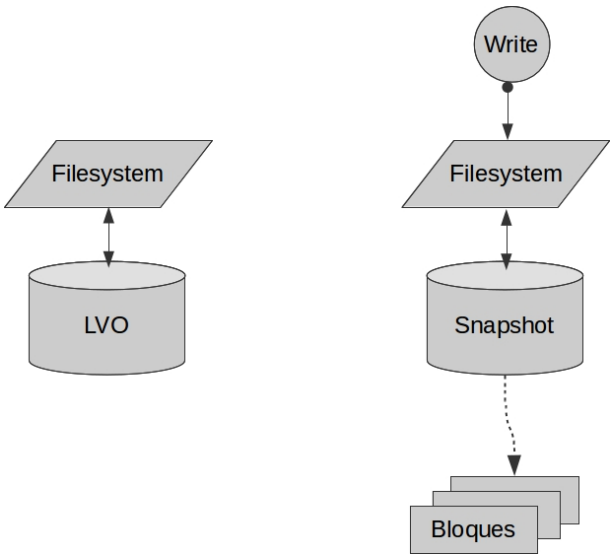

Figura 6: Escritura sobre el snapshot

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, tales como pruebas de instalación, modificación de configuraciones, etc.

Como se ha visto, el conjunto de datos 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, son bloques de archivos que fueron modificados mediante operaciones regulares de escritura sobre el filesystem localizado en el snapshot y montado en modo R/W (Fig. 6).

En ambos casos, este conjunto de bloques alojados en el snapshot puede volver a ser aplicado sobre el LVO (o *revertido*) con la opción `--merge` del comando `lvconvert`. Este comando restituye (*roll-back*) al LVO todos sus bloques originales, que fueron grabados temporariamente en el espacio de excepciones del snapshot, y *luego destruye el snapshot*.

Reversión de snapshots

Ahora bien, al ser revertido el snapshot sobre el LVO:

- Si no ha habido ninguna operación de escritura sobre el filesystem del snapshot, el LVO *vuelve al estado original* al momento de ser tomado el snapshot.
- Si, en cambio, existen archivos en el snapshot que han sido modificados, al revertir el snapshot el LVO *cambia*, asumiendo las modificaciones que se hayan hecho en el snapshot.

El resultado, para el LVO, será completamente diferente en uno y otro caso. Ambas propiedades pueden aprovecharse para recuperar el estado después de una modificación que no ha sido satisfactoria, con una de las dos técnicas siguientes.

Técnica 1 Con esta técnica, las modificaciones se realizan sobre el filesystem del LVO y en caso necesario se revierten usando el snapshot intacto.

1. Crear el snapshot del LVO, con espacio suficiente para registrar las modificaciones que se piensa hacer.
2. Ejecutar las operaciones críticas (instalar, actualizar, modificar) sobre el LVO.

3. Verificar el resultado.
   - Si el resultado fue satisfactorio, destruir el snapshot con `lvremove <ID snapshot>`.
   - Si el resultado no fue satisfactorio, recuperar el estado original del LVO con el comando `lvconvert --merge <ID del snapshot>`.
4. El snapshot ha sido eliminado.

Técnica 2 Con esta técnica se deja el LVO fuera de línea, intacto, y se trabaja sobre el snapshot montado en modo R/W.

1. Crear el snapshot del LVO, con espacio suficiente para registrar las modificaciones que se piensa hacer.
2. Desmontar el LVO y montar en su lugar el snapshot.
3. Ejecutar las operaciones críticas (instalar, actualizar, modificar) sobre el snapshot.
4. Verificar el resultado.
   - Si fue satisfactorio, se vuelcan las modificaciones al LVO con `lvconvert --merge <ID del snapshot>`.
   - Si no fue satisfactorio, se elimina el snapshot.
5. Finalmente, en uno u otro caso, el snapshot ha sido eliminado, y 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, poner 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.

5. SMART

La tecnología SMART (Self-Monitoring, Analysis and Reporting Technology) es un conjunto de funciones útiles para verificar la confiabilidad de los discos y predecir sus fallos. Estas funciones se incorporan en casi todos los discos rígidos electromecánicos ATA y SCSI actuales, así como en los de estado sólido. Los discos dotados de SMART pueden ejecutar varias clases de tests sobre sí mismos, y reportar sus resultados.

La interfaz SMART puede ser usada por el hardware o desde software adecuado. Normalmente el firmware BIOS puede hacer uso de SMART ejecutando un chequeo de salud de los discos al inicio del equipo y denunciando problemas potenciales o existentes. El paquete de software `smartmontools` permite utilizar las funciones de SMART de los discos desde varios sistemas operativos.

Atributos

La tecnología SMART incorpora sensores de ciertas variables físicas, y contadores de una cantidad de eventos relevantes, que describen la historia y el estado de los discos. El firmware SMART del disco almacena estos datos, llamados atributos, en memoria no volátil situada en el mismo disco, y los actualiza automáticamente. Cada disco, dependiendo del fabricante, del modelo y del nivel de la especificación ATA al cual adhiere, mantiene un conjunto de estos atributos. Se numeran entre 1 y 253, y tienen nombres específicos. Se dice que dos terceras partes de los fallos de los discos pueden ser predichos observando los valores de determinados atributos. También puede determinarse si un disco ha llegado al fin de su vida útil en base a estos valores.

Los atributos SMART tienen un "valor crudo" o *raw* y un "valor normalizado". El valor crudo de cada atributo tiene una interpretación dada en ciertas unidades. En el caso del atributo 12 (Fig. 7), llamado `power_cycle_count`, por ejemplo, el valor crudo es simplemente un entero que representa la cantidad de veces que el disco ha sido activado; para otros atributos, debe interpretarse como una temperatura en grados Celsius (atributo 194, `temperature_celsius`), una cantidad de horas (atributo 9, `power_on_hours`), etc. Estos valores de interpretación natural corren, lógicamente, sobre diferentes escalas. El "valor normalizado" de un atributo, por el contrario, está siempre entre 1 y 254, y se obtiene del valor crudo mediante una función matemática que es computada por el firmware del mismo disco. Estos valores normalizados, al ser comparados con determinados umbrales, denuncian la aparición de problemas. El fabricante del disco, que es quien conoce los detalles de construcción y funcionamiento del disco, provee el algoritmo para computar la función de normalización y el valor del umbral para cada atributo. SMART almacena el valor normalizado actual y el "peor valor" histórico (el menor valor normalizado obtenido desde la primera puesta en operación del disco hasta el momento actual) de cada atributo.

Diagnósticos

SMART agrupa los atributos en dos clases: *Old Age* y *Pre-failure*. Los primeros sirven para diagnosticar cuándo un disco ha llegado al término de su vida útil, y los segundos para diagnosticar fallos efectivos o inminentes de los discos. Si, en algún caso, el valor normalizado de un atributo resulta inferior o igual a su umbral, entonces el diagnóstico depende del tipo del atributo. Si un disco tiene un atributo de tipo Old Age cuyo valor normalizado es inferior al umbral, entonces se concluye que el disco ha superado su vida útil según el diseño del producto. Si un atributo es de tipo Pre-failure y su valor normalizado es inferior al umbral, entonces se anuncia un fallo inminente en 24 horas[1]. Si el "peor valor" de un atributo

[1] En un estudio publicado por Google (Eduardo Pinheiro et al., Failure Trends in a Large Disk Drive Population. USENIX V, 2007) se muestra evidencia, obtenida sobre una población muy grande, de que los fallos de discos están correlacionados

de tipo Pre-failure es menor que el umbral, es indicio seguro de que el disco ha presentado un fallo en algún momento anterior.

```
SMART Attributes Data Structure revision number: 16
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAG     VALUE WORST THRESH TYPE      UPDATED  WHEN_FAILED RAW_VALUE
  1 Raw_Read_Error_Rate     0x002f   100   100   051    Pre-fail  Always       -       9
  2 Throughput_Performance  0x0026   055   055   000    Old_age   Always       -       11745
  3 Spin_Up_Time            0x0023   089   089   025    Pre-fail  Always       -       3462
  4 Start_Stop_Count        0x0032   066   066   000    Old_age   Always       -       35133
  5 Reallocated_Sector_Ct   0x0033   252   252   010    Pre-fail  Always       -       0
  7 Seek_Error_Rate         0x002e   252   252   051    Old_age   Always       -       0
  8 Seek_Time_Performance   0x0024   252   252   015    Old_age   Offline      -       0
  9 Power_On_Hours          0x0032   100   100   000    Old_age   Always       -       12364
 10 Spin_Retry_Count        0x0032   252   252   051    Old_age   Always       -       0
 11 Calibration_Retry_Count 0x0032   100   100   000    Old_age   Always       -       149
 12 Power_Cycle_Count       0x0032   099   099   000    Old_age   Always       -       1114
191 G-Sense_Error_Rate      0x0022   100   100   000    Old_age   Always       -       119
192 Power-Off_Retract_Count 0x0022   252   252   000    Old_age   Always       -       0
194 Temperature_Celsius     0x0002   058   036   000    Old_age   Always       -       42 (Min/Max 15/64)
195 Hardware_ECC_Recovered  0x003a   100   100   000    Old_age   Always       -       0
196 Reallocated_Event_Count 0x0032   252   252   000    Old_age   Always       -       0
197 Current_Pending_Sector  0x0032   252   252   000    Old_age   Always       -       0
198 Offline_Uncorrectable   0x0030   252   252   000    Old_age   Offline      -       0
199 UDMA_CRC_Error_Count    0x0036   100   100   000    Old_age   Always       -       1
200 Multi_Zone_Error_Rate   0x002a   100   100   000    Old_age   Always       -       6290
223 Load_Retry_Count        0x0032   100   100   000    Old_age   Always       -       149
225 Load_Cycle_Count        0x0032   095   095   000    Old_age   Always       -       59834
```

Figura 7: Parte del informe SMART

Tests

SMART provee tres clases de tests, que pueden hacerse sin riesgo durante la operación normal del sistema y no causan errores de lectura ni de escritura. La primera categoría se llama *online testing*, y no tiene efecto sobre la performance del dispositivo. La segunda categoría se llama *offline testing*, y puede causar una degradación de la performance, aunque en la práctica esto es raro porque el disco suspende el test cada vez que hay actividad de lectura o escritura. Estas dos primeras categorías de tests deberían más bien llamarse "adquisición de datos", ya que simplemente se ocupan de actualizar constantemente los valores de los atributos. La tercera categoría es un test propiamente dicho, y se llama *self test*.

Paquete smartmontools

Comprende dos componentes: el daemon smartd y la interfaz de usuario smartctl. El daemon smartd habilita las funciones de SMART en los discos y consulta su estado cada cierto intervalo. Los parámetros de operación de smartd se indican en su archivo de configuración `/etc/smartd.conf`. La herramienta smartctl sirve para consultar el estado de los discos y ejecutar tests.

sobre todo con dos variables mantenidas por SMART: los errores de superficie (scan errors) y los eventos de reubicación de sectores (reallocation count).

Parte IV
Estrategias de Respaldo

1. Backups

Decisiones sobre los respaldos

- Propósitos y requerimientos legales, confidencialidad, etc.
- Archivos a respaldar
  - Archivos de sistema, de los usuarios, de las aplicaciones
  - Vuelcos de bases de datos, formatos portables
- Dinámica de los archivos
  - Estáticos o volátiles
  - Tasa de crecimiento
- Período de cobertura (ventana de seguridad)
- Esquema y política de respaldo
  - Quién es responsable
  - Frecuencia y alcance de cada operación de backup
  - Diarios, semanales
  - Completos, diferenciales, incrementales
- Almacenamiento del respaldo
  - Dónde se guarda
  - Quién y cómo accede
  - Recuperación de desastres, sitios remotos
- Medios de respaldo (cinta, disco, DVD...)
  - Costo en tiempo de respaldo
  - Costo en tiempo de restauración
  - Costo monetario por byte de almacenamiento
- Presupuesto en medios
- Herramienta y formato (ftp, sftp, scp, tar, cpio, rsync, dd...)
- Sistema de apoyo (AMANDA, Bacula, BackupPC, Mondo...)
- Procedimiento de restauración
- Procedimiento de verificación
- Eliminación del respaldo
  - Quién lo hace
  - El medio se destruye o se reutiliza
- ¿Cómo escala la solución? ¿Se adapta al crecimiento de los datos?

2. Sincronización de archivos

Herramienta rsync

Rsync es un comando de sincronización de directorios y archivos. Puede usarse básicamente como un reemplazo de los comandos rcp o scp, pero presenta gran cantidad de opciones interesantes. Entre otras cosas, utiliza un algoritmo propio para computar las diferencias entre los archivos origen y destino antes de iniciar la copia, y transfiere únicamente las modificaciones. Es decir, para aquellos archivos que

son diferentes entre origen y destino, únicamente copia aquellos bloques de datos que son efectivamente diferentes.

Las opciones de rsync son muy numerosas (ver Anexo B). Algunas de las más utilizadas:

```
rsync -av \                          # modo archive y modo verbose
  --compress \                       # comprimir al transferir
  --force \                          # borrar directorios aun si no vacios
  --delete \                         # borrar archivos no existentes
  --delete-excluded \                # borrar tambien los excluidos
  --ignore-errors \                  # borrar aun bajo error
  --exclude-from=exclude_file \      # excluir los archivos listados
  --backup \                         # modo backup
  --backup-dir=‘date +%Y-%m-%d‘ \    # directorio del modo backup
  origen destino
```

La opción -a funciona como sinónimo de un conjunto de otras opciones convenientes para las copias de respaldo en general. Esta opción equivale a -rlptgoD, que se traduce como r=recursivo; l=respetar los links simbólicos; p=preservar los permisos, t=los tiempos de acceso de los archivos, g=el grupo y o=el dueño; y D=recrear los pseudoarchivos de dispositivos en el lado destino. La compresión de archivos será conveniente cuando origen y destino se sitúen en equipos diferentes sobre la red.

Rsync puede usarse de muchas formas:

- Copia de archivos locales, cuando ninguno de los elementos origen y destino contiene un separador *dos puntos* (“:”).
- Copia entre host local y host remoto usando un shell remoto (por defecto, ssh) como transporte, cuando el destino contiene “:”.
- Copia desde un servidor rsync remoto al host local, cuando el origen contiene un separador “::” o es un URL que comienza en “rsync://”
- Copia entre el host local y un servidor rsync en el host remoto usando un shell como transporte, caso en que el origen contiene un separador “::” y se da la opción –rsh=COMMAND

Especificación del origen

La sintaxis de rsync tiene una particularidad: al especificar el directorio origen de la copia debe tenerse en cuenta que una barra al final (“/”) indica copiar solamente los contenidos del directorio origen, mientras que si no está presente la barra se entiende que el directorio también debe copiarse en el destino.

```
$ ll xmms
total 2052
-rw-rw-r-- 1 oso oso 1973069 Aug 5 11:18 xmms-1.2.10-16.i386.rpm
-rw-rw-r-- 1 oso oso  36388 Aug 5 11:18 xmms-cdread-0.14-6.a.i386.rpm
-rw-rw-r-- 1 oso oso  79784 Aug 5 11:18 xmms-mp3-1.2.10-0.lvn.3.4.i386.rpm
```

El comando `rsync -avz xmms/ xmms2` copiará los tres archivos en un nuevo directorio llamado xmms2. En cambio, el comando `rsync -avz xmms xmms2` producirá lo siguiente.

```
$ rsync -avz xmms xmms2
building file list ... done
created directory xmms2
xmms/
xmms/xmms-1.2.10-16.i386.rpm
xmms/xmms-cdread-0.14-6.a.i386.rpm
xmms/xmms-mp3-1.2.10-0.lvn.3.4.i386.rpm
```

```
sent 2068345 bytes received 92 bytes 827374.80 bytes/sec
total size is 2089241 speedup is 1.01
$ ll xmms2
total 4
drwxrwxr-x 2 oso oso 4096 Aug 5 11:28 xmms
```

Modo backup

El modo backup de rsync hace que los archivos preexistentes en el destino sean renombrados cada vez que se reemplazan o se borran. La opción –backup-dir controla dónde serán creadas estas copias de backup con nuevos nombres. La opción –backup-suffix indica qué extensión tendrán estos archivos (por defecto se agrega un signo "~"). En el caso anterior, supongamos que luego de hacer la copia, se modifican los archivos presentes en xmms.

```
$ ll xmms2/xmms/
total 2052
-rw-rw-r-- 1 oso oso 1973069 Aug 5 11:18 xmms-1.2.10-16.i386.rpm
-rw-rw-r-- 1 oso oso  36388 Aug 5 11:18 xmms-cdread-0.14-6.a.i386.rpm
-rw-rw-r-- 1 oso oso  79784 Aug 5 11:18 xmms-mp3-1.2.10-0.lvn.3.4.i386.rpm
$ touch xmms/*
$ rsync -avz --backup xmms xmms2
building file list ... done
xmms/xmms-1.2.10-16.i386.rpm
xmms/xmms-cdread-0.14-6.a.i386.rpm
xmms/xmms-mp3-1.2.10-0.lvn.3.4.i386.rpm

sent 2068339 bytes received 86 bytes 827370.00 bytes/sec
total size is 2089241 speedup is 1.01
$ ll xmms2/xmms/
total 4104
-rw-rw-r-- 1 oso oso 1973069 Aug 5 11:30 xmms-1.2.10-16.i386.rpm
-rw-rw-r-- 1 oso oso 1973069 Aug 5 11:18 xmms-1.2.10-16.i386.rpm~
-rw-rw-r-- 1 oso oso  36388 Aug 5 11:30 xmms-cdread-0.14-6.a.i386.rpm
-rw-rw-r-- 1 oso oso  36388 Aug 5 11:18 xmms-cdread-0.14-6.a.i386.rpm~
-rw-rw-r-- 1 oso oso  79784 Aug 5 11:30 xmms-mp3-1.2.10-0.lvn.3.4.i386.rpm
-rw-rw-r-- 1 oso oso  79784 Aug 5 11:18 xmms-mp3-1.2.10-0.lvn.3.4.i386.rpm~
```

3. Replicación de datos

Más allá de las diferentes alternativas para realizar la copia periódica de archivos, con una u otra herramienta, está el concepto de replicación de información. Su propósito es diferente del de la copia de resguardo. En lugar de proteger los datos, manteniendo una historia de copias a través del tiempo, la replicación busca aumentar la disponibilidad de los datos, manteniendo dos imágenes iguales en dos lugares de almacenamiento.

Esta forma de tratamiento de la información no protege de pérdidas o errores, pero facilita el rápido regreso a la operación de un sistema ante fallas de hardware. La replicación puede ser asimétrica (es decir, una copia es master y la segunda recibe las modificaciones), o simétrica (ambas copias reciben las modificaciones de la otra); y puede ser continua (disparada por eventos de filesystem) o administrativa (accionada por el administrador). El regreso a la operación de un sistema con fallos puede ser un proceso manual, como en el caso de clonar sobre *bare metal* un sistema completo cuyo hardware ha quedado inutilizado, o automático, como en los clusters de Alta Disponibilidad.

Herramientas que permiten ejecutar replicación, con diferentes alcances y objetivos, son los utilitarios rsync, csync2, unison (que utilizan el algoritmo de rsync), los sistemas de control de versiones como

git o Subversion, filesystems distribuidos como Lustre o GlusterFS, los sistemas de almacenamiento en nube con clientes automáticos, como Dropbox o Owncloud, la clonación de máquinas virtuales, y el dispositivo de bloques replicado DRBD.

Replicación con rsync

El comando rsync siguiente tomará las acciones necesarias para obtener una réplica del directorio /datos en el servidor backupserver, directorio /backups. La opción -a copiará todos los archivos en todos los subdirectorios del directorio origen que no existan en el directorio destino, o que hayan sido modificados, preservando los permisos; e ignorará todos aquellos archivos que sean iguales en ambos directorios. Además, la opción –delete borrará los archivos en el directorio destino que no existan en el origen.

```
rsync -aE --delete /datos backupserver.ejemplo.com.ar:/backups
```

Modo *dry-run*

Por supuesto, la opción –delete es peligrosa. Para verificar cuál será el efecto de un comando rsync antes de ejecutarlo, se puede utilizar la opción -n o su sinónimo –dry-run.

Replicación de particiones

Una extensión de la idea de backup es la de crear imágenes de particiones completas de un sistema, de modo de poder recuperarlo ante fallas generalizadas en menos tiempo y con menos trabajo que una reinstalación. Esta técnica es conveniente para aquellas particiones que alojan filesystems "de sistema", es decir, donde los datos son mayormente binarios o archivos de configuración de sistema, que no son afectados por el trabajo cotidiano del usuario.

Comando dd

Una herramienta útil para el resguardo de particiones completas es el utilitario dd. Con él se puede obtener una imagen de una partición o dispositivo completo.

```
dd if=/dev/sda1 of=part1.dd; scp part1.dd serverbackup.ejemplo.com.ar:
dd if=/dev/fd0 of=diskette.img
```

Un uso alternativo es el resguardo de las tablas de particiones:

```
dd if=/dev/sda of=tpart.dd bs=1K count=1
```

Si se necesitara recuperar un conjunto de archivos de una imagen de una partición o dispositivo, puede hacerse montando la imagen sobre un directorio vacío, como si fuera un dispositivo físico. La opción de mount que permite esto es loop.

```
mount -t vfat -o ro,loop /backups/partwin.img /mnt/rescate
cp /mnt/rescate/* /datos
```

Comando partimage

La réplica de particiones con dd, sin embargo, carece de flexibilidad. Los backups son del mismo tamaño que la partición, y son difíciles de manipular. Un utilitario tal como partimage tiene conocimiento del filesystem, copia solamente los bloques en uso del dispositivo, y opcionalmente aplica compresión. Además, puede dividir el resguardo en múltiples archivos para facilitar el almacenamiento. Puede ser

usado en volúmenes de los diferentes sistemas de archivos de Linux, tanto como en FAT o NTFS. Tiene un modo interactivo y uno de línea de comandos.

Tanto partimage como dd tienen limitaciones. No se puede recuperar un backup sobre una partición de tamaño menor que la original. En caso de ser sobre una de tamaño mayor, el espacio sobrante se desaprovecha, salvo que se redimensione el filesystem con una herramienta tal como resize2fs. No es sencillo recuperar selectivamente un conjunto de archivos de una imagen.

Comando netcat

El comando nc (o netcat) permite crear una conexión entre dos equipos a través de la red y utilizar entrada y salida standard para transferir información. Esto puede aprovecharse para la replicación de particiones de forma muy simple. El siguiente ejemplo utiliza el comando dd para crear un flujo de bytes directamente de una partición del equipo origen, que se desea replicar.

```
# en el host origen
dd if=/dev/sda5 | nc 10.0.0.2 3000

# en el host destino
nc -l -p 3000 | dd of=/dev/sda3
```

El flujo de bytes es emitido por la salida standard del comando dd y entubado a la entrada del comando nc que se conecta con un servidor nc en el host destino. En este host se recibe el flujo a través de la conexión y se entuba a la entrada standard de dd, que lo vuelca en la partición destino. El resultado es una transferencia *raw* de la partición, sin almacenamiento intermedio. En el ejemplo, el host origen tiene la dirección IP 10.0.0.1, y el destino, 10.0.0.2. El servidor nc en el destino atiende por el port 3000. La partición 5 del primer host se copia en la partición 3 del segundo, la que debe tener al menos el tamaño de la primera.

4. Temas de práctica

1. ¿Cómo determinar cuánto cambio hay en un filesystem? Diseñe un script que sea capaz de determinar qué archivos han sido modificados y cuánto ha crecido un filesystem entre dos fechas cualesquiera. Diseñe un script que sea capaz de presentar esta información en forma de tabla semanal.
2. Utilizando rsync, ejecute una transferencia a través de la red, de un archivo de tamaño considerable. Anote el tiempo que registra el programa. Verifique las opciones de compresión que está utilizando su comando.
3. Repita la experiencia utilizando scp. Verifique las opciones de compresión de modo que la comparación sea justa. Puede usar el comando time para obtener el tiempo real de ejecución. Compare los tiempos.
4. Repita una vez más la transferencia utilizando rsync, sin eliminar el archivo ya copiado en su lugar de destino. Repita con scp y compare los tiempos.
5. Prepare un directorio con uno o dos niveles de subdirectorios. Guarde archivos de tamaño considerable en esa estructura. Repita el experimento anterior con rsync y con scp, registrando tiempos. Luego modifique un único carácter de un único archivo de la jerarquía, repitiendo la transferencia con ambas herramientas y registrando tiempos.
6. Clasifique en simétrica/asimétrica, continua/administrativa, la replicación ejecutada por herramientas como rsync, csync2, unison, git, Subversion, Lustre, GlusterFS, Dropbox, Owncloud, la clonación de máquinas virtuales, y el dispositivo de bloques replicado DRBD. ¿De qué forma ayuda cada una a recuperar la operatividad de un equipo en caso de fallo?
7. Cuando rsync utiliza ssh como transporte, se adapta a la política de autenticación de usuarios que utiliza ssh. ¿Cómo se puede evitar que rsync pida password de usuario para ejecutar una transferencia entre diferentes hosts?

8. Prepare un script para replicar un directorio junto con sus subdirectorios, usando rsync, en otro host. Instale el script en crontab. Verifique su funcionamiento modificando los contenidos del directorio.
9. Modifique el script anterior para utilizar el modo backup de rsync en lugar de replicar el directorio.
10. ¿De qué forma, y en qué casos, puede ser de ayuda la herramienta `anacron` en lugar de `cron`?
11. Utilice el comando partimage para obtener una imagen de una partición de su sistema. Aplíquela sobre otro disco. ¿Cómo se puede verificar que la replicación ha sido correcta?

Parte V
Alta Disponibilidad

1. Conceptos de Disponibilidad

- Fallos
- Modos de fallo
  - Fallo silencioso: El sistema deja de cumplir su función sin notificarlo
  - Fallo bizantino: El sistema sigue cumpliendo su función en forma incorrecta
  - Modo Fail-stop: Al fallar, el sistema deja de funcionar
- Tolerancia a fallos: Propiedad que permite a un sistema continuar operando correctamente ante fallos de algunos de sus componentes
- Cada sistema existe sobre un continuo de tiempo dividido en Tiempo entre Fallas (TTF) y Tiempo de Reparación (TTR).
  - MTBF (*Mean Time Between Failures*), tiempo promedio entre fallos.
  - MTTR (*Mean Time To Repair*), tiempo promedio para la reparación.
- La proporción del tiempo en la cual realmente está operando el sistema mide la disponibilidad.
  - En promedio, $D = MTBF/(MTBF + MTTR)$.
  - La disponibilidad se elevará entonces aumentando el MTBF y/o reduciendo el MTTR.
  - La selección de sistemas confiables a nivel de hardware apunta a aumentar el MTBF.
  - La implementación de técnicas de Alta Disponibilidad pretende reducir lo más posible los períodos de carencia de los servicios o TTR, por lo tanto el MTTR.
  - En esta fórmula se consideran únicamente los tiempos de carencia del servicio *no planificados*.
- Paradas planificadas
  - Backups, instalación de patches, upgrades
- Paradas no planificadas
  - Recuperables: Error de la aplicación, error del operador, apagones, caídas de red, fallas de hardware
  - Desastres: Accidentes, robos, sabotajes, incendios, meteoros, catástrofes naturales
- La no disponibilidad tiene generalmente un costo para la organización, y puede controlarse con medidas especiales.

Alta Disponibilidad

Las técnicas de Alta Disponibilidad (*High Availability, HA*) buscan construir un sistema que dé soporte para la provisión de un servicio durante la mayor parte del tiempo que sea posible. Las medidas para lograr esta meta abarcan varios niveles de actividad, y van desde prácticas adecuadas de administración de sistemas, pasando por elecciones especiales de soporte de hardware, políticas de backup, implantación de sistemas especiales de HA con redundancia, hasta la preparación de estrategias de Recuperación de Desastres (*Disaster Recovery*). Las medidas se apoyan unas en otras, y deben ser implementadas en orden creciente de complejidad y costo (Fig. 8).

Alta disponibilidad **no es**:

- Servicio Continuo, donde el usuario no ve interrupciones del servicio.
- Tolerancia a Fallos, que es una característica del hardware redundante y altamente confiable.
- Balance de Carga, aunque algunos modelos de HA lo incluyen como beneficio adicional.
- Recuperación de Desastres, donde se supone que existe una pérdida de datos que se debe subsanar.

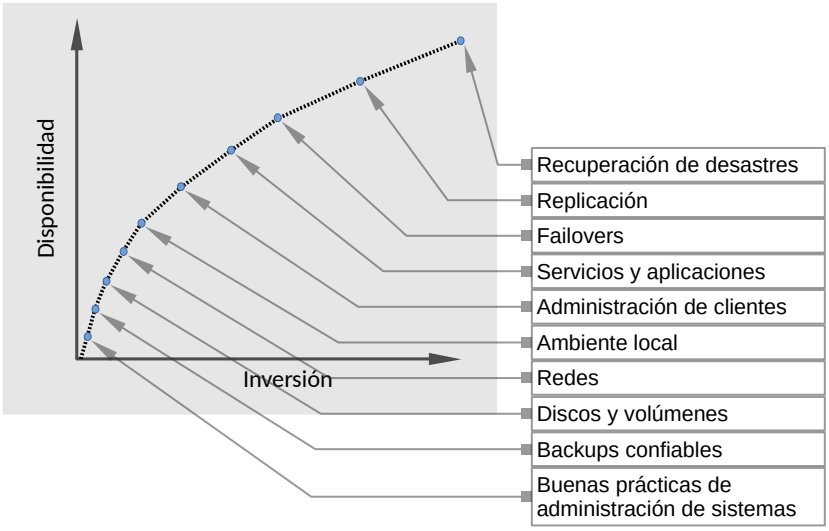

Figura 8: Jerarquía de medidas de disponibilidad

Al implementar medidas de Alta Disponibilidad no pretendemos servicio continuo. Sin embargo, aunque el usuario puede llegar a percibir interrupciones o anomalías en el servicio, normalmente se puede lograr un "tiempo de reparación" tolerable, y a veces imperceptible. En la práctica, ante un evento de impacto, esto se traduce en una conexión interrumpida con el servicio pero que se recupera dentro de las próximas acciones del cliente (siguientes reintentos de lectura por la aplicación, siguiente click del usuario, etc.).

Redundancia y disponibilidad

El principio fundamental de HA es la eliminación de los puntos únicos de falla (*Single Point of Failure, SPOF*) por duplicación o multiplicación de puntos de servicio. Esta técnica se llama en general *redundancia*.

Si un servicio tiene una disponibilidad de 90 % (o sea, una probabilidad de estar disponible igual a 0.9) sobre un período dado, su probabilidad de carencia o no disponibilidad en ese período es $1 - 0{,}9 = 0{,}1$. Si el recurso puede implementarse en forma redundante, y suponiendo eventos de fallo independientes, la probabilidad de que dos instancias del recurso estén no disponibles a la vez en ese mismo período es $0{,}1 * 0{,}1 = 0{,}01$; con lo cual la probabilidad de disponibilidad del servicio es ahora $1 - 0{,}01 = 0{,}99$, o sea 99 % para dicho período. Al agregarse la segunda instancia de recurso se ha eliminado el SPOF y se ha ganado un orden de magnitud en disponibilidad.

Para ilustración, cuando un servicio tiene una disponibilidad del 99 % por año, esto quiere decir que en promedio la no disponibilidad del servicio ocurrirá durante 3.65 días por año. En cambio, si la disponibilidad es de 99.99 %, el servicio quedará no disponible a lo sumo durante 52.5 min por año.

La suposición de que los eventos de fallo son independientes se basa en que no existan SPOFs de nivel superior, es decir, anteriores en la cadena de precondiciones para el servicio (por ejemplo, se supone que existen diferentes líneas de alimentación eléctrica). No es posible construir un sistema completo sin ningún SPOF, pero es posible delimitar ámbitos de responsabilidad donde no existan.

Métricas económicas

Probabilidad Cantidad de veces que se espera ocurra el evento durante la vida remanente del sistema. Un evento que puede ocurrir semanalmente durante los próximos cinco años tiene una probabilidad de 260[2].

Duración Cantidad de tiempo en que los usuarios carecerán del servicio a causa del evento. Incluye el tiempo necesario para el restablecimiento del servicio y para notificar a los usuarios dicho restablecimiento.

Impacto Porcentaje de la comunidad de usuarios que se verá afectada por el evento.

- Cálculo de riesgos para cada evento e_i en el sistema original
  - Efecto $E_b = Probabilidad * Duración * Impacto$
  - Riesgo $R_b = Costo(ND) * \sum E_b(e_i)$
- Luego de implementar HA
  - Efecto $E_a = Probabilidad_{HA} * Duración_{HA} * Impacto_{HA}$
  - Riesgo $R_a = Costo(ND) * \sum E_a(e_i) + Costo(HA)$
  - Ahorro $A = R_b - R_a$
- Retorno de inversión $ROI = A/Costo(HA)$

2. Clustering

Una arquitectura típica de Alta Disponibilidad es el cluster de HA (*High Availability Cluster, HAC*), donde dos o más nodos en una red ofrecen servicios en forma redundante. El HAC protege a los servicios de los fallos del nodo que los soporta.

Este tipo de cluster no debe confundirse con los clusters de Computación de Altas Prestaciones (*High Performance Computing, HPC*) cuyo objetivo es lograr alta capacidad de procesamiento dividiendo la ejecución de una tarea en varios procesadores, ni con los clusters de Balance de Carga, o Servidores Virtuales, donde simultáneamente se brinda el mismo servicio pero distribuyéndolo sobre varios nodos.

Existen diferentes configuraciones, pero la más sencilla es la del cluster de dos nodos, donde uno es el servidor activo o primario de un servicio, y el otro está normalmente en estado pasivo, secundario o *standby*. La función del secundario, durante la operación normal, es la monitorización del primario para detectar sus caídas o fallas y tomar el control de los servicios.

El proceso de migración de un servicio de un nodo a otro se llama *failover*. La acción de asumir el rol de primario para un servicio desde la situación de standby se llama *takeover*. Cuando un sistema pierde la característica de ser redundante, por fallo de todas las instancias menos una, se dice que trabaja en *modo degradado* (momento en que se hace crítica la puesta en servicio de las demás instancias).

Los servicios protegidos por el HAC pueden ser varios de entre los normalmente brindados por un nodo, como WWW, mail, LDAP, NFS, NIS, DHCP, SMB/CIFS, proxy/firewall, etc. y por supuesto el almacenamiento compartido o replicado. En particular existe un recurso sujeto a failover que es una dirección IP especial, de propiedad del cluster (*Virtual IP, VIP*). Es la dirección por donde se ofrecen los servicios protegidos. Esta dirección migra convenientemente al nodo que detente el rol primario en cada momento.

La decisión de si los servicios de un nodo, mientras su rol es secundario, están o no corriendo, y en tal caso, en qué modo, es una cuestión de diseño del HAC. Dependiendo de la implementación del servicio, puede ser suficiente el modo *cold standby* (los procesos servidores no están ejecutándose) o el modo *warm standby* (los servidores están ejecutándose pero sin dar servicio). Este último modo tiene una penalidad menor en tiempo de arranque de los servicios al momento del failover.

Un nodo puede ser a la vez primario para algunos servicios y secundario para otros, aunque en general no aporta nada a la solución del problema de disponibilidad el hecho de distribuir los servicios (si los equipos no son capaces de soportar la carga durante la caída del primario, no lograremos aumentar la disponibilidad de todos modos). En cambio, tiene sentido la distribución de servicios cuando además se

[2]Este concepto de probabilidad se llama originalmente *likelihood*, y claramente no se trata del mismo concepto de la probabilidad matemática, que únicamente puede tomar valores entre 0 y 1.

pretende balance de carga, aunque entonces no se habla con propiedad de nodos en primario/standby, ni de takeover.

En general, las aplicaciones responsables de ofrecer los servicios deben compartir alguna cantidad de información de estado o de configuración, disponible a todos los nodos del cluster, para lo cual se agrega un almacenamiento compartido o replicado. Si existe una sola instancia, compartida, de almacenamiento (p. ej. mediante NFS, o mediante una única unidad de almacenamiento *multihomed*), se introducirá un SPOF. La alternativa es la replicación del almacenamiento, que dependiendo de los objetivos del sistema HA puede hacerse en varias formas y modos.

La replicación puede realizarse a nivel de bloques (donde se interpone un módulo de software especial en el camino de I/O de los datos hacia el disco), de filesystem, o de archivos. Este último caso sería el de la replicación ejecutada por el administrador de sistemas a nivel de aplicación (con utilitarios como rsync o similares). Para los clusters construidos sobre redes locales (i.e. sobre redes de baja latencia), es conveniente ejecutar la replicación a nivel de bloques en modo sincrónico. Para clusters sobre WANs, o para los procedimientos de recuperación de desastres, es preferible la replicación a nivel de aplicación.

- Formas de clustering
  - de Alta Disponibilidad
  - de Almacenamiento
  - de Altas Prestaciones
  - de Balance de Carga
- Alta Disponibilidad ->High Availability ->HA
- Recursos ->servicios
- Modelo de capas o niveles (*clustering stack*)
  - Mensajería y pertenencia (*messaging/membership agent*): comunicaciones
  - Gestión de recursos (*cluster resource manager*, CRM): lógica de asignación de recursos a nodos
- Failover
  - VIP (Virtual IP)
  - Migración de recursos
  - Almacenamiento compartido o replicado (Fig. 9)
    - DAS (*Direct Attached Storage*)
      - Unidades de disco locales
      - Filesystem XFS, EXT4
    - NAS (*Network Attached Storage*)
      - Filesystem exportado por otro nodo, que se monta a través de la red
      - NFS, SMB/CIFS
    - SAN (*Storage Area Network*)
      - Dispositivo de bloques que aparenta ser local pero es de almacenamiento externo, exportado por otro nodo
      - Tecnologías iSCSI, Fibre Channel
  - *Heartbeats, Quorum, Fencing, Split Brain, STONITH*
    - Los nodos de un cluster comprueban que pueden alcanzar a sus peers usando mensajes cortos y periódicos llamados latidos (*heartbeats*).
    - Si un nodo sufriera un fallo de red, dejaría de alcanzar a un peer y podría llegar erróneamente a la conclusión de que ha caído. Para separar este falso positivo de falla, los nodos siguen protocolos distribuidos que comprueban que son capaces de alcanzar a otros, y además algunos protocolos consultan a otros nodos si pueden alcanzar a los terceros, sospechosos de fallo. Cuando existe una cantidad suficiente de nodos que están de acuerdo sobre el estado de pertenencia de los demás miembros del cluster, se dice que existe *quorum*, y sólo en ese caso se toman ciertas acciones.

- En un cluster de dos nodos, si éstos perdieran contacto entre sí pero no con el resto de la red, ambos se considerarían a sí mismos únicos miembros vivos del cluster. Esta condición se llama *split brain* o *cerebro dividido*. El cluster no es capaz de recuperarse automáticamente de esta condición. Para evitarla, se aplica redundancia en las vías de comunicación de heartbeats (normalmente, más de una interfaz de red, y a veces algún canal secundario separado del subsistema de red, tal como una línea serial).
- En clusters de más de dos nodos, al perderse contacto con un nodo, el resto del cluster cae en la incertidumbre sobre su estado. Cuando existe almacenamiento compartido, una falla del nodo podría afectar a los servicios del cluster. Para evitarlo, el cluster utiliza algún mecanismo de *fencing* (separar por la fuerza al nodo presuntamente fallado). Por lo general estos mecanismos se basan en dispositivos que controlan físicamente la alimentación de los nodos.
- Cuando se pierde comunicación con un nodo por una cantidad de tiempo determinada, el agente de comunicaciones, que vigila la pertenencia al cluster, debe accionar el dispositivo de fencing y provocar una acción de *STONITH (shoot the other node in the head)*, apagándolo. Cuando no existe un dispositivo, sino que es el administrador del cluster quien debe dar de baja manualmente al nodo fallado, se dice que el dispositivo es de tipo *meatware STONITH* (☺).

- Herramientas
  - Heartbeat v.1, OpenAIS, Corosync, CMAN
  - Heartbeat v.2.1.4+, Pacemaker, rgmanager
  - LVS
- Configuraciones
  - Activo/Pasivo
  - Activo/Activo con balance de carga
  - Activo/Pasivo con múltiples recursos

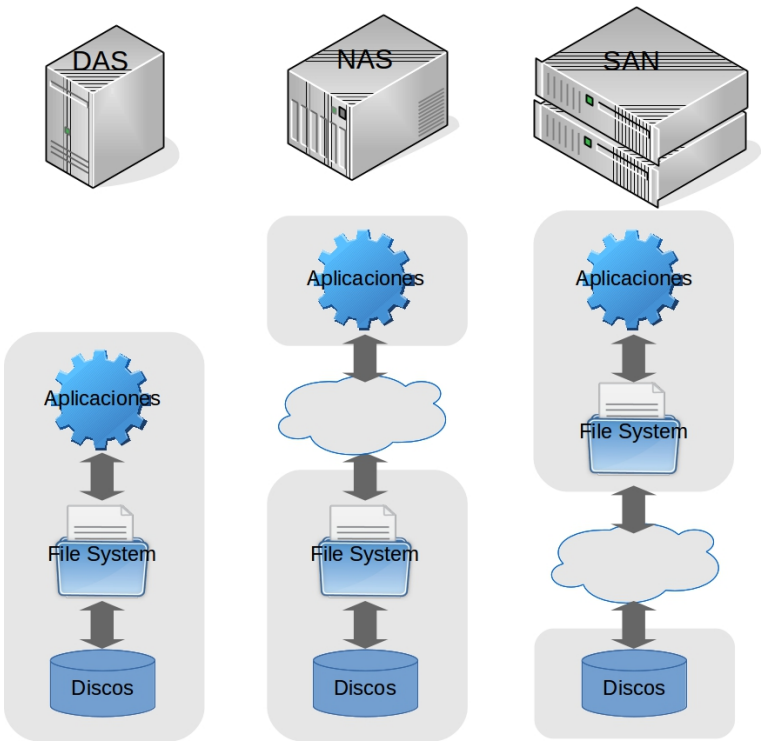

Figura 9: DAS, NAS, SAN

3. DRBD

DRBD (Distributed Replicated Block Device) es una tecnología implementada a nivel del kernel Linux que efectúa una replicación **automática, continua y asimétrica**, de bloques de disco. El modo de operación de DRBD es semejante a un conjunto RAID 1 donde ambos discos estuvieran en diferentes hosts de la red (Fig. 10).

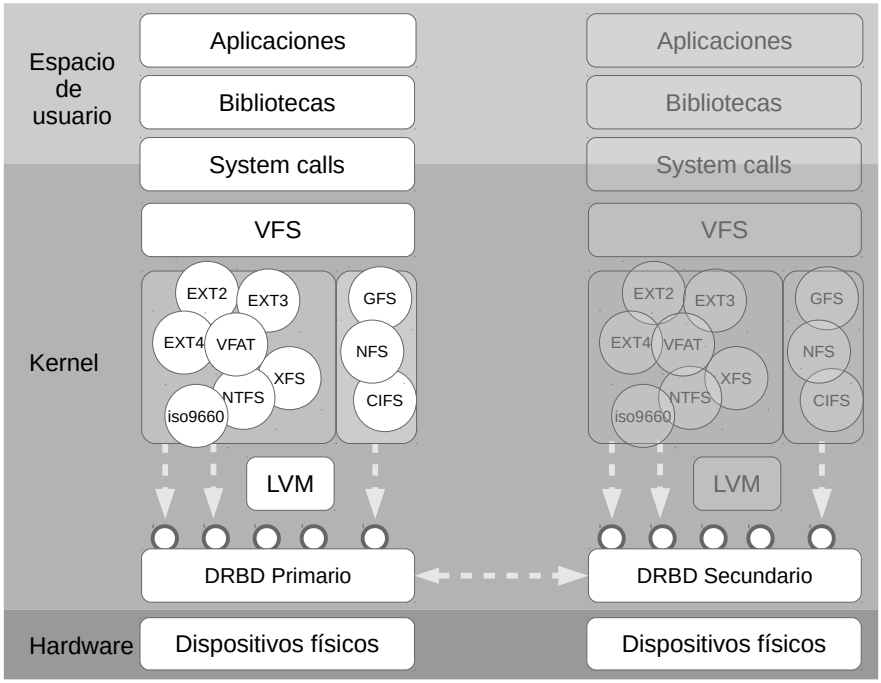

Figura 10: Localización de DRBD en el stack de I/O

El driver DRBD está incorporado al kernel Linux desde la versión 2.6.33. Se ubica lógicamente entre la cache de bloques del kernel y el almacenamiento físico, ofreciendo a los procesos usuarios la interfaz de un dispositivo de bloques. El usuario normalmente utiliza este dispositivo para, por ejemplo, crear un filesystem sobre él.

Para configurar un dispositivo DRBD debe dársele un soporte, es decir, un dispositivo de bloques subyacente que le provea almacenamiento (tal como una partición o un volumen lógico), y un canal de comunicación con su nodo *peer*.

En su configuración más habitual, DRBD impone roles primario y secundario a los nodos que intervienen. En este modo, únicamente las aplicaciones corriendo en el nodo primario pueden utilizar el espacio de almacenamiento ofrecido por el dispositivo DRBD. En particular, solamente el primario puede montar ese dispositivo[3].

Forma de operación

- Toda actividad de lectura o escritura sobre el dispositivo soporte (creación de filesystem, escritura de bloques de datos, borrado de archivos, etc.) pasa por el control del módulo DRBD.

[3]En realidad, es posible una configuración de doble primario; pero para poder hacer accesos concurrentes en forma segura, las aplicaciones que vayan a usar ese almacenamiento necesitan contar con un **lock manager distribuido**, que sólo está presente en los filesystems de cluster como GFS2, GlusterFS u OCFS2.

- Cada modificación, antes de ser efectivizada en el nodo primario, es comunicada al módulo *peer* secundario.
- Éste graba el bloque modificado en su soporte local y confirma la grabación.
- Recibida la confirmación, el nodo primario graba su copia del bloque en su propio soporte local.

Ante un evento de caída de uno u otro de los nodos, cuando se recupera el nodo perdido, ocurre una resincronización automática. Cuando el nodo primario cae y se recupera, puede volver a ese rol o no, dependiendo de una opción de configuración. El vínculo por el cual circulan los datos de bloques que deben actualizarse en el secundario, y por donde se realizan las sincronizaciones, es una conexión de red preferiblemente dedicada y diferente de la red de servicio, y de la mayor velocidad posible. Puede funcionar tanto en LAN como en WAN, utilizando diferentes protocolos de replicación para favorecer la performance en caso de alta latencia (como en WAN) o maximizar la expectativa de consistencia (en caso de LAN).

DRBD en su versión actual está restringido a un conjunto de dos nodos. Por lo tanto, los clusters de Alta Disponibilidad con almacenamiento replicado basado en DRBD tienen exactamente dos nodos (aunque cada nodo puede tener configurados diferentes recursos DRBD y pertenecer a varios clusters, o los recursos pueden ser *apilados (stacked)*).

Recursos DRBD

Un recurso (*resource*) DRBD es un conjunto replicado de unidades de almacenamiento. Puede alojar uno o múltiples volúmenes (*volume*). Cada volumen dentro de un recurso puede tener algunos atributos propios y diferentes de los demás en el mismo recurso. Cada volumen DRBD necesita un espacio de almacenamiento propio, y ofrecerá un dispositivo de bloques independiente. La administración de DRBD se realiza al nivel de recursos.

Configuración de DRBD

La configuración de DRBD puede alojarse en /etc/drbd.conf con la descripción de los recursos, o bien se pueden crear archivos de configuración independientes para cada recurso (dentro de `/etc/drbd.d/`, con la extensión `.res`). En `/usr/share/doc/drbd84-utils-8.9.1/drbd.conf.example` se encuentra un ejemplo bastante complejo de configuración con varios recursos (ver Apéndice C).

> La configuración de DRBD debe ser **idéntica en ambos nodos**. Sin embargo, no hay restricciones sobre los nombres o tipos de los dispositivos de soporte. Un volumen puede estar soportado por particiones de nombre diferente, por una partición en un nodo y por un LV en el otro, etc.

Ejemplos de configuración

- Un recurso con un único volumen. En ambos hosts, el almacenamiento es idéntico (se utilizan particiones con el mismo nombre en ambos nodos).

```
resource drbd0 {
        device /dev/drbd0;
        disk /dev/sdc2;
        meta-disk internal;
        on nodo1 {
                address 10.0.16.12:7780;
        }
        on nodo2 {
                address 10.0.16.13:7780;
        }
}
```

- Un recurso con dos volúmenes, con almacenamiento en volúmenes lógicos llamados lv0 y lv1 respectivamente. En ambos nodos, los nombres de los LV son idénticos. Cada volumen recibe un nombre de dispositivo propio (parámetro `device`).

```
resource drbd0 {
        net {
                protocol A;
        }
        volume 0 {
                device /dev/drbd0;
                disk /dev/vgdrbd/lv0;
                meta-disk internal;
        }
        volume 1 {
                device /dev/drbd1;
                disk /dev/vgdrbd/lv1;
                meta-disk internal;
        }
        on nodo1 {
                address 192.168.47.1:7780;
        }
        on nodo2 {
                address 192.168.47.2:7780;
        }
}
```

- Una configuración con un único volumen pero donde los nombres de las particiones de soporte no son iguales en ambos nodos.

```
resource drbd0 {
        on nodo1 {
                device /dev/drbd0;
                disk /dev/sdc1;
        meta-disk internal;
                address 10.0.16.17:7780;
        }
        on nodo2 {
                device /dev/drbd0;
        disk /dev/sdb2;
        meta-disk internal;
                address 10.0.16.16:7780;
        }
}
```

Parámetros de configuración

Discutiremos algunos parámetros importantes de la configuración.

Dispositivo (device) Es el nombre con el que se manejará el volumen usando los comandos administrativos.

Disco (disk) Es el nombre de dispositivo usual del almacenamiento para ese volumen.

Recursos y volúmenes (resource, volume) La unidad de replicación y de administración es el recurso, que puede ser dividido en volúmenes que se utilizan independientemente.

`drbdadm cstate`	Listar estado de conexiones
`drbdadm dstate`	Listar estado de datos
`drbdadm up`	Activar recurso
`drbdadm up all`	Activar todo
`drbdadm down all`	Desactivar todos los recursos
`drbdadm primary`	Asumir el rol primario sobre un recurso
`drbd-overview`	Estado de conexiones y recursos
`cat /proc/drbd`	Monitorear estado de dispositivos

Cuadro 3: Comandos de administración de DRBD

Direcciones (address) Se especificarán las direcciones de cada nodo sobre la red privada, dedicada a la replicación. La política de firewalling debe permitir la conexión entre los nodos mediante los ports declarados en la configuración. Como normalmente se trata de una red dedicada, suele ser conveniente desactivar completamente el firewall sobre esta red.

Metadatos (meta-disk) Los metadatos mantenidos por DRBD durante la operación incluyen datos vitales para la integridad del volumen y su recuperación. El parámetro de configuración DRBD correspondiente se llama *meta-disk*. Pueden ser alojados de dos maneras:

1. En los últimos sectores del mismo soporte del almacenamiento (opción llamada *meta-disk internal*)
2. Usando un soporte separado (como, por ejemplo, *meta-disk /dev/sdbX*).

La ventaja de la primera opción es una más fácil administrabilidad, ya que los metadatos acompañan a los datos cada vez que se recuperan. La ventaja de la segunda opción puede explotarse si se trata de particiones o volúmenes en discos separados, en cuyo caso la performance puede ser mejor que si datos y metadatos comparten un disco. En caso de altos niveles de actividad los metadatos pueden convertirse en un cuello de botella, y con la segunda opción esto puede evitarse.

Si se desea instalar DRBD para replicar un volumen de datos preexistente, la primera opción implica riesgo de corrupción de los datos, a menos que se reduzca el filesystem o se extienda el volumen. En cualquier otro caso, por simplicidad, la opción más recomendada es *meta-disk internal*.

Nombres de nodo (on) En la configuración aparecen los nombres de nodo arbitrarios *nodo1* y *nodo2*. Los nombres de los nodos son vitales tanto para que el software de dispositivos replicados interprete su configuración, como a los efectos de que los nodos puedan identificarse ante un manejador de pertenencia al cluster. Deben satisfacerse algunas condiciones indispensables sobre los nombres para lograr la conexión entre peers.

1. Ambos peers deben responder al comando `uname -n` con esos respectivos nombres. Para esto se puede usar temporariamente el comando `hostname`, pero la configuración persistente depende de la distribución[4].
2. Cada peer debe poder efectuar la resolución de la dirección del otro por su nombre. Se puede utilizar un servidor DNS, pero es suficiente con modificar la configuración del resolver local (`/etc/hosts`).

Protocolos (protocol) Dependiendo de las características de la red, es posible configurar el dispositivo replicado para que las operaciones de escritura sean confirmadas al llegar al otro nodo (protocolo A), al ingresar en la cache del sistema de E/S del otro nodo (protocolo B), o al ser grabadas físicamente (protocolo C). En redes locales se elige normalmente el protocolo de replicación C, que es el de mejor integridad de datos, ya que considera efectuada la operación de write solamente cuando el nodo remoto acusa haber grabado físicamente un sector replicado. En redes con alta latencia, en cambio, se preferiría el protocolo A.

La configuración por defecto del protocolo (C) se encuentra en el archivo de configuración global.

[4] En CentOS se consigue configurando `HOSTNAME=nombre` en `/etc/sysconfig/network`.

Ejecución de DRBD

Una vez creada la configuración, para la inicialización, puesta en marcha y administración de DRBD se recurre principalmente al comando `drbdadm` con sus subcomandos (Cuadro 3).

Inicializar el dispositivo

Esta operación crea en el soporte los metadatos para la replicación.

```
drbdadm create-md <recurso>
```

Activar el recurso

```
drbdadm up <recurso>
```

Establecer roles

Una vez configurado un recurso y activado por primera vez el dispositivo DRBD, los peers se comunican pero no pueden decidir por sí solos cuál de los dos será el primario. El administrador debe imponer los roles con una opción especial del comando drbdadm. Esta opción varía según las versiones de DRBD. En DRBD 8.4 en adelante, el rol primario se define **únicamente en el nodo que vaya a cumplir el rol primario** dando el comando `drbdadm primary --force <recurso>`. Esta novedad se comunica automáticamente al otro peer que se asumirá como secundario y quedará replicando.

Uso del dispositivo

Creación de filesystems

Una vez puesto en marcha exitosamente un dispositivo, ya puede ser utilizado (**en el primario**) aunque los peers no hayan acabado la sincronización. Normalmente se creará un filesystem sobre cada volumen DRBD. Por defecto, `mkfs` aprovechará todo el espacio en el volumen.

```
mkfs -t ext4 /dev/drbd0
```

> Notar que el dispositivo donde se crea el filesystem es el especificado como `device` en la configuración. Los volúmenes o particiones subyacentes **no deben volver a accederse** mientras esté activo el dispositivo, ya que son de uso exclusivo del driver DRBD.

Cada filesystem debe ser creado, no sobre el dispositivo de bloques soporte (la partición del disco `/dev/sdaX`, el dispositivo RAID `/dev/mdX`, el volumen `/dev/vgX/lvX`, etc.), sino sobre el dispositivo replicado `/dev/drbd0, /dev/drbd1`, etc., correspondiente, que existe una vez que se ha inicializado el recurso drbd. DRBD además ofrece un directorio `/dev/drbd/by-res` donde aparecen subdirectorios por cada recurso y pseudoarchivos por cada volumen. Alternativamente, se puede usar esa nomenclatura.

Tuning del filesystem

Normalmente un filesystem tiene definido un intervalo de tiempo entre filesystem checks y la cantidad máxima de veces que se montará el filesystem sin ser chequeado. Cada vez que sea montado el filesystem, el sistema verificará estas condiciones y eventualmente lanzará un chequeo que puede aumentar la latencia de la puesta en servicio del nodo. Al crear el filesystem sobre el dispositivo DRBD correspondiente (con el utilitario mkfs), puede ser de interés redefinir estos parámetros con el utilitario `tune2fs`. Los argumentos `-c0` y `-i0` eliminan esos controles.

```
tune2fs -c 0 -i 0 /dev/drbd0
```

Montado de los recursos

En cada nodo se necesitará preparar líneas en `/etc/fstab` para permitir el montado de los dispositivos de DRBD. Sin embargo, estos dispositivos **no deben ser montados automáticamente** (opción `noauto`) ya que es el servicio de *cluster resource manager* quien debe manejar su montado y desmontado en función del rol del nodo dentro del cluster en cada momento.

```
/dev/drbd0   /public       ext3      defaults,noauto     0 0
```

Failover manual

En ausencia de un gestor de recursos de cluster, y ante cualquier evento de fallo (o para realizar alguna tarea administrativa), para transferir el rol primario al otro nodo se deben seguir pasos en cierto orden.

1. En el primario, si está activo, detener las aplicaciones que usen el recurso y desmontarlo.
2. Degradarlo administrativamente al rol secundario. Momentáneamente ambos nodos quedarán en rol secundario.
3. Promover el otro nodo a primario.
4. Montar el recurso en el nuevo primario.

```
umount /dev/drbd/by-res/<recurso>
drbdadm secondary <recurso>
drbdadm primary <recurso>
mount /dev/drbd/by-res/<recurso> <punto de montado>
```

Monitoreo de DRBD

El estado de los recursos se visualiza rápidamente con `drbd-overview`.

```
# drbd-overview
 0:drbd0/0 Connected Primary/Secondary UpToDate/UpToDate /r0 ext4 19G 173M 18G 1%
 1:drbd0/1 Connected Primary/Secondary UpToDate/UpToDate /r1 ext4 13G 161M 12G 2%
```

La información de estado de DRBD incluye una terna de rótulos *(cs, ro, ds)* con varios posibles valores:

Connection status (cs) Indica si ambos nodos están conectados. Los valores posibles pueden indicar estados de esperando conexión, desconectado, en proceso de sincronización, etc. El valor para el estado correcto operativo es *Connected*.

Role (ro) Es un par de valores que indican los roles asumidos por ambos nodos en un momento dado. El primer valor corresponde al nodo donde se da el comando, y el segundo al nodo restante. Los valores para el estado correcto operativo son *Primary/Secondary* (cuando el comando se da en el nodo primario) o *Secondary/Primary* (cuando se da en el nodo secundario).

Disk Status (ds) Es un valor que indica si los contenidos del almacenamiento de ambos nodos están sincronizados. El primer valor corresponde al nodo donde se da el comando, y el segundo al nodo restante. El valor correcto operativo es *UpToDate* para ambos nodos.

Valores diferentes de los correctos operativos, no necesariamente indican un error, sino que puede tratarse de una condición transitoria. Inicialmente los dispositivos atravesarán estados de no conectado, inconsistente, desconocido, etc., hasta que se comuniquen los peers y comiencen a sincronizar sus contenidos. Otro tanto ocurre durante la recuperación del cluster al volver del modo degradado.

Eventualmente, las ternas de estado de DRBD en ambos nodos deben alcanzar respectivamente los valores (*Connected, Primary/Secondary, UpToDate/UpToDate*) y (*Connected, Secondary/Primary, UpToDate/UpToDate*).

Para vigilar la actividad de DRBD puede observarse el pseudoarchivo `/proc/drbd`. Con el comando `watch cat /proc/drbd` se puede visualizar constantemente el estado de los recursos. El resto de los datos que aparecen en la salida corresponde a niveles de performance e historia de actividad.

- Primario, con ambos recursos sincronizados

```
0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate A r-----
   ns:5066100 nr:0 dw:0 dr:5066764 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
       oos:0
1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate A r-----
   ns:0 nr:0 dw:0 dr:664 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

- Secundario, esperando conexión

```
0: cs:WFConnection ro:Secondary/Unknown ds:Inconsistent/DUnknown C r----s
   ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:20127092
1: cs:WFConnection ro:Secondary/Unknown ds:Inconsistent/DUnknown C r----s
   ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:13422144
```

- Secundario sincronizando el volumen 0 y habiendo sincronizado el volumen 1

```
0: cs:SyncTarget ro:Secondary/Primary ds:Inconsistent/UpToDate A r-----
   ns:0 nr:4030816 dw:4030816 dr:0 al:0 bm:0 lo:1 pe:7 ua:0 ap:0 ep:1 wo:f
       oos:1035284
       [==============>.....] sync'ed: 79.7% (1008/4944)M
       finish: 0:03:21 speed: 5,124 (5,380) want: 7,600 K/sec
1: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate A r-----
   ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

- Secundario con ambos recursos sincronizados

```
0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate A r-----
   ns:0 nr:5066100 dw:5066100 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f
       oos:0
1: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate A r-----
   ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

Restricciones importantes

- Bajo ningún concepto debe montarse un filesystem soportado por DRBD v.0.7.X en un nodo mientras su rol de DRBD es secundario, ni siquiera en modo read-only. El acceso simultáneo es causa de corrupción de datos.
- La restricción anterior deja de tener efecto a partir de la versión 8.X, que es capaz de funcionar con ambos nodos en rol primario. Sin embargo, su uso práctico no es trivial porque requiere 1) un filesystem de cluster y 2) aplicaciones tolerantes al acceso múltiple.

Referencias

- http://www.drbd.org/users-guide-8.4/drbd-users-guide.html

Notas de instalación

En la familia RedHat/CentOS es necesaria la instalación previa de un repositorio adicional. Actualmente están disponibles paquetes para dos versiones diferentes de DRBD.

```
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
rpm -Uvh http://www.elrepo.org/elrepo-release-6-6.el6.elrepo.noarch.rpm
yum install drbd84-utils.x86_64 kmod-drbd84.x86_64
```

Temas de práctica

1. Prepare dos equipos en las condiciones necesarias para formar parte de un cluster de almacenamiento replicado DRBD (nombres de nodos, direcciones secundarias, resolución de nombres, partición libre en un disco, recurso DRBD, /etc/fstab, punto de montado). Busque la configuración DRBD más simple posible, con un único volumen. Luego de activar el recurso, cree un filesystem sobre él y observe la sincronización mientras ocurre. Monte y utilice el filesystem. Observe la conducta del nodo peer cuando se baja administrativamente el recurso, cuando se baja el nodo primario, cuando se simula una caída del sistema primario, cuando se migra administrativamente el rol primario. Intente acceder a los archivos creados en el otro nodo. ¿Qué ocurre al recuperarse el nodo anteriormente perdido? Tenga en cuenta que, en este ejercicio, las acciones de montar y desmontar los filesystems son manuales.
2. Idem anterior, pero creando un LV sobre el recurso DRBD.
3. Investigue la opción de configuración *dual-primary* de DRBD. Con esta opción, ¿qué ocurrirá si se monta el recurso en ambos nodos, sin contar con un filesystem de cluster? ¿Qué ocurrirá si se interrumpe la comunicación por la red entre los nodos?
4. En un cluster *dual-primary*, ¿puede utilizar el mecanismo de snapshots de LVM para acceder a los contenidos del LV desde el otro nodo sin migrar el primario?
5. Discuta qué mecanismos serían necesarios para que las operaciones de asumir el rol primario, y montar/desmontar los recursos DRBD, fueran automáticas, y cómo se podrían implementar.
6. La robustez del sistema DRBD depende de la estabilidad de las comunicaciones a través de la red. Investigue sobre el fenómeno de *split brain*, qué significa, sus causas, su gravedad, y las formas de resolverlo.
7. ¿Cómo se puede producir una situación de *split brain* en el cluster de práctica? ¿Cuál es la conducta del sistema ante este evento?

4. Heartbeat

Conceptos

Heartbeat es un agente de comunicaciones de cluster, cuya función es ofrecer un mecanismo para enviar mensajes entre los nodos en forma confiable, determinar qué nodos pertenecen al cluster en cada momento, y notificar a un gestor de recursos los eventos de cambio de pertenencia al cluster (ingreso o salida de nodos).

Hasta la versión 2.1.3, la misión de Heartbeat era, además de gestionar las comunicaciones, controlar la adquisición y entrega de los recursos servidos por el cluster en función de la pertenencia de los nodos (es decir, era además un CRM, *Cluster Resource Manager*). A partir de la versión 2.1.4, Heartbeat es solamente un agente de comunicaciones, y las funciones de CRM se han desplazado a un proyecto diferente, **Pacemaker**. A su vez, Pacemaker es un CRM que puede funcionar sobre varios agentes de comunicaciones, como Heartbeat, Corosync, u OpenAIS. Diferentes distribuciones adoptan diferentes stacks de Alta Disponibilidad, pero en general las configuraciones recomendadas actualmente son Pacemaker+Heartbeat, o preferiblemente, Pacemaker+Corosync.

`logfacility daemon`	Se usará para logging el facility daemon
`node nodo1 nodo2`	Nodos que participan del cluster
`keepalive 2`	Intervalo en segundos entre latidos
`deadtime 10`	Declarar nodo muerto luego de n segundos
`bcast eth0 eth1`	Por dónde emitir latidos en broadcast
`ping_group routers 172.16.20.1 10.0.0.2`	Pseudomiembros del cluster para comprobar que la red sigue viva
`auto_failback yes`	Intentar mantener los recursos en su responsable nominal
`respawn hacluster /usr/lib/heartbeat/ipfail`	Failover en caso de falla de red

Cuadro 4: Parámetros más importantes para ha.cf

Sin embargo, de todas formas Heartbeat sigue ofreciendo un CRM minimal para recursos locales que es sumamente sencillo de configurar. Aquí usaremos este CRM (configuración llamada "V1" o "*haresources*") para clusters **de dos nodos**.

- Para determinar la pertenencia al cluster, Heartbeat establece un tráfico periódico de latidos (*heartbeats*) en ambos sentidos entre los nodos. Mientras cada nodo escuche los latidos del otro, el cluster permanecerá en estado completo.
- Al caer un nodo, el sobreviviente dejará de escuchar latidos por un intervalo de tiempo mayor que un umbral dado, y así declarará muerto al nodo *peer*, luego de lo cual asumirá la prestación de determinados servicios. El cluster pasará a *modo degradado*, lo que significa que seguirá prestando servicios, pero sin redundancia, hasta la recuperación del nodo afectado.
- Los eventos posibles que afectan el estado de los recursos son la caída de un nodo, o bien el regreso de un nodo al cluster. Las acciones ante cada evento dependen de la configuración de los servicios que serán alojados en cada nodo, y consistirán en asumir o entregar recursos en un orden preestablecido.

Configuración

Heartbeat se configura con tres archivos de control, situados dentro de `/etc/ha.d` en la instalación default, y que son `ha.cf` (de configuración general), `haresources` (de recursos) y `authkeys` (de autenticación).

La configuración de Heartbeat es simétrica, es decir, los tres archivos deben ser **idénticos** en los dos nodos. El hostname de cada nodo se utiliza para asumir la pertenencia al cluster en la directiva node del archivo ha.cf y para ligar el nodo a su rol en haresources. Lo que identifica el rol del nodo es únicamente su hostname.

Archivo de configuración general

El archivo **/etc/ha.d/ha.cf** establece algunos parámetros de configuración general de heartbeat. Los únicos parámetros realmente imprescindibles en este archivo son **node**, **auto_failback** y uno entre **ucast, mcast** o **bcast** (respectivamente, unicast, multicast o broadcast). El Cuadro 4 muestra ejemplos de algunos parámetros de configuración.

node Define los nodos que integrarán el cluster. En particular es importante que esté correctamente consignado el nombre de cada nodo tal como es entregado por el comando **uname -n**.

auto_failback La característica de **auto_failback yes** es opcional, e implica que luego de un incidente ocurrido al primario de un servicio, y al recuperarse éste, el nodo secundario devolverá el recurso al responsable nominal.

keepalive El tiempo de **keepalive** es el intervalo entre latidos. Debe tenerse en cuenta que, si se propaga el latido por la red de servicio, un tiempo de keepalive muy bajo significará tráfico espurio en la red.

deadtime Este parámetro define el tiempo luego del cual se declara muerto a un nodo que no ha respondido un latido. Un **deadtime** muy bajo, si bien permite una recuperación del servicio más rápida, implica mayor peligro de ingresar en la condición de *cerebro dividido*, lo que puede ocasionar graves problemas (como el acceso simultáneo al almacenamiento) y requiere intervención humana para ser corregida. El deadtime debe afinarse en función de la capacidad de respuesta del nodo que debe contestar el latido. Si los nodos están sujetos a variaciones importantes en la carga de trabajo, es preferible adoptar una posición conservadora y evitar los problemas de cerebro dividido a costa de un mayor tiempo de failover.

bcast, ucast, mcast Define las interfaces que emitirán heartbeats. Puede utilizarse la red privada, que es confiable, para propagar el latido. Es preferible no utilizar la red de servicio.

serial y baud Para obtener redundancia se recomienda utilizar además un enlace serial null-modem. Se agrega una vía de heartbeat alternativa con la directiva serial ttyS0.

ping, ping_group Con estas directivas se identifican un host de control o un grupo de control de hosts que se supone están siempre activos en la red y pueden contestar pings. Este grupo de control sirve para validar que cada nodo puede acceder a la red, y determinar que una pérdida de latidos realmente se trata de un miembro que no contesta.

respawn Esta directiva establece procesos que serán monitoreados por heartbeat y reiniciados en caso de fallo. En el caso normal, el programa que se desea monitorear es ipfail, que vigila si la conectividad del nodo sufre algún inconveniente y en caso necesario entrega los recursos al otro miembro. En la línea de la directiva respawn, el primer argumento es la identidad del usuario con la cual correrá el proceso monitoreado.

Archivo de recursos

El archivo /etc/ha.d/haresources define los recursos que serán propios de cada nodo. Junto al nombre de cada nodo se especifican los recursos que adquirirá ese nodo mientras el cluster opere en modo normal. En modo degradado, el nodo superviviente adquirirá los recursos que sean propios del otro ("los recursos migrarán" de un nodo al otro).

Los recursos que Heartbeat es capaz de administrar pueden ser:

1. Direcciones IP de servicio, que son asumidas dinámicamente por los nodos.
2. Servicios del sistema, que pueden ser lanzados por un script de control habitual (de los que se alojan en /etc/init.d o /etc/rc.d/init.d).
3. Scripts especiales de Heartbeat (*resource agents*), localizados en /etc/ha.d/resource.d, que se comunican con otros servicios con interfaz más compleja.

El gestor de recursos minimal de Heartbeat inicializará cada recurso al arranque del servicio heartbeat, y con ellos manejará el arranque y detención de esos servicios al producirse eventos de pertenencia.

> Los recursos **se adquieren de izquierda a derecha** tal como están enumerados en haresources y se liberan de derecha a izquierda cuando el nodo sale del rol primario.

- Ejemplo sencillo de archivo **haresources** para un cluster activo-standby:

```
nodo1  httpd 172.16.20.101
```

  Aquí se especifican:

  - El servicio httpd, invocado por el script localizado en /etc/rc.d/init.d o /etc/init.d, que será lanzado al inicio o al momento de failover.
  - La dirección de servicio que será asumida por una interfaz secundaria, creada dinámicamente por Heartbeat.
  - El nodo que es el preferido para asumir los recursos "servicio httpd" y "dirección IP".

- Ejemplo de archivo **haresources** para un cluster activo-activo sencillo:

```
nodo1  postfix 172.16.20.101
nodo2  httpd 172.16.20.102
```

Aquí se especifican:

- Los servicios postfix y httpd, dependientes de /etc/rc.d/init.d o /etc/init.d.
- Las direcciones de los dos servicios principales del cluster, que serán asumidas por interfaces secundarias, creadas dinámicamente por Heartbeat.
- Los nodos que son preferidos para cada servicio y dirección IP.

- Ejemplo de archivo **haresources** para un cluster activo-activo un poco más complejo:

```
nodo1  drbddisk::drbd0 \
              Filesystem::/dev/drbd0::/mnt/mail::ext3 \
              postfix \
              172.16.20.101
nodo2  drbddisk::drbd1 \
              Filesystem::/dev/drbd1::/mnt/web::ext3 \
              httpd \
              172.16.20.102
```

Aquí se especifican:

- Los *resource agents* `drbddisk` y `Filesystem`. Los argumentos para esos scripts se indican con "::".
- Los recursos DRBD drbd0 y drbd1 definidos en el archivo de configuración /etc/drbd.conf, con sus nombres tal como figuran en dicho archivo.
- Los filesystems ubicados sobre los recursos DRBD, con su dispositivo, punto de montado y tipo de filesystem.
- Los servicios, dependientes de /etc/rc.d/init.d o /etc/init.d, que utilizarán esos filesystems.
- Las direcciones de esos dos servicios principales del cluster, que serán asumidas por interfaces secundarias, creadas dinámicamente por Heartbeat.

Archivo de autenticación

El archivo /etc/ha.d/authkeys permite la autenticación de un nodo frente al otro por uno de tres métodos posibles. Debe tener permisos restrictivos (0600). Puede ser generado con un script tal como el siguiente.

```
DATE=$(date)
cat <<-!AUTH >/etc/ha.d/authkeys
  # Generado automaticamente $DATE
  auth 1
  1 sha1 $(dd if=/dev/urandom count=4 2>/dev/null | md5sum | cut -c1-32)
!AUTH
chown root:root /etc/ha.d/authkeys
chmod 0600 /etc/ha.d/authkeys
```

Entrega de los servicios a Heartbeat

- Heartbeat requiere el control del arranque y parada de los servicios que componen los grupos de recursos.
- Los recursos del cluster no deben ser lanzados automáticamente a la inicialización del sistema, es decir, se debe deshabilitar el lanzamiento de los servicios que deben quedar bajo control de Heartbeat (en el ejemplo anterior, httpd y postfix).

- La deshabilitación debe hacerse para todos los runlevels bajo los cuales se piense hacer correr al sistema (típicamente 3 y 5).
- Habilitar el arranque de Heartbeat al inicio.

Monitoreo de Heartbeat

La salida de logging de Heartbeat depende de la configuración. Cuando la salida se dirige al log del sistema host, el comando `tail -f /var/log/messages | grep heartbeat` muestra la actividad en forma continua. Otros archivos de logging pueden ser `/var/log/ha-log`, `/var/log/ha-debug`.

Testing del cluster

Es imprescindible llevar a cabo algunos tests para verificar la corrección de la configuración.

1. **Desconectar la alimentación del servidor primario**
   En el servidor secundario, heartbeat debe detectar la pérdida de latidos e iniciar un failover. Deberán ser iniciados los scripts de recursos correspondientes. El secundario debe enviar broadcasts ARP gratuitos para notificar al resto de la red que la dirección MAC correspondiente al IP de servicio ha cambiado.
2. **Comprobar el efecto del comando hb_standby**
   Verificar que el comando hb_standby en el primario fuerza la migración de los recursos al secundario. Nuevamente verificar que el mismo comando, dado en el secundario, devuelve los servicios al primario.
3. **Desconectar la conexión a la red de producción del primario**
   El servicio ipfail debe detectar esta condición y los recursos deben recaer sobre el secundario.
4. **Desconectar uno de los caminos de heartbeat entre los dos servidores**
   Debe existir dos o más caminos de heartbeat entre los servidores para evitar los falsos positivos. Al eliminar uno de estos caminos, la operación no debe sufrir ningún cambio.
5. **Desconectar todos los caminos de heartbeat entre los dos servidores**
   Si se está usando algún dispositivo de Stonith, el secundario debe suponer que el primario ha salido de servicio, iniciar un evento de Stonith, y asumir los recursos. Lo que siga dependerá de cómo se ha configurado Stonith y de si se está usando o no la opción auto_failback. Con Stonith y auto_failback, ambos servidores comenzarán cíclicamente a sacarse de servicio mutuamente. Para evitarlo, deshabilitar auto_failback.
6. **Matar el daemon heartbeat en el primario**
   El secundario debe ejecutar Stonith sobre el primario antes de asumir los servicios para evitar un escenario de split-brain.
7. **Matar los daemons de servicios en el server primario**
   Si se están usando cl_status y/o cl_respawn, o la aplicación de monitoreo Mon, el cluster debe tomar la acción correspondiente (reiniciar los servicios, alertar al administrador).
8. **Rebootear ambos servidores**
   Los servidores deben arrancar correctamente y dejar los servicios activos en su primario. Esta acción de tuning puede sugerir un cambio en la configuración del tiempo initdead si el secundario trata de capturar los servicios antes de que el primario termine de bootear.

Notas de instalación

- En RedHat/CentOS: `yum install epel-release; yum update`
- Configuraciones ejemplo en `/usr/share/doc/heartbeat-X.XX`
- Scripts auxiliares en `/usr/share/heartbeat`.

Parte VI
Virtualización

1. Formas de virtualización

- *Hosts* o Anfitriones
- *Guests* o Huéspedes
- Virtualización completa o Emulación
  - Reproducción, mediante software, de la conducta de todo el hardware. Es necesario escribir emuladores para cada dispositivo, lo cual es laborioso, costoso y generalmente difícil; pero presenta la ventaja de que la máquina completamente emulada permite correr cualquier sistema operativo sin modificaciones y sin que ese sistema operativo, ni las aplicaciones, perciban que están corriendo sobre hardware emulado. Esta forma de virtualización presenta la mayor fidelidad y transparencia, pero performance limitada.
- Virtualización asistida por hardware
  - Apoyo provisto por el hardware para funciones de multiplexado de entrada/salida y manejo especial de memoria. Las generaciones recientes de procesadores (familia Core 2 de Intel en adelante) contienen la funcionalidad necesaria para asistir en la virtualización. Las ventajas consisten en buena performance y el hecho de poder correr en forma virtualizada cualquier sistema operativo sin necesidad de modificarlo. La desventaja es, por supuesto, que se necesita contar con la capacidad necesaria en el hardware.
  - VMWare, QEMU, KVM, Xen
- Paravirtualización
  - Ejecución de un SO modificado
  - Reescritura de los drivers del sistema operativo host y de los guests. Los drivers se construyen según el modelo de *split drivers*, o drivers divididos. El sistema operativo host es quien sigue actuando, con la parte inferior de los drivers, sobre los dispositivos físicos. Las máquinas virtuales, al efectuarse un requerimiento de entrada/salida, activan la mitad superior del driver y provocan un trap al monitor de virtualización, que administra los pedidos y los deriva a la mitad inferior del driver. Los nuevos drivers permiten multiplexar entrada/salida entre los dispositivos físicos y una cantidad de dispositivos virtuales asociados. La ventaja principal es la gran performance lograda con respecto a la emulación o virtualización completa. Para determinadas cargas de trabajo, especialmente para programas acotados por CPU, la eficiencia de un conjunto de máquinas paravirtualizadas es muy cercana al óptimo. La principal desventaja es que claramente se necesita modificar o instrumentar el código, tanto del sistema operativo host como de los guests. Sin embargo las aplicaciones siguen funcionando sin modificaciones.
  - Xen, UML, VirtualBox en modo software
- Virtualización a nivel del SO
  - Creación de múltiples espacios de usuario independientes en lugar de uno solo (llamados, según la tecnología específica, contenedores, *virtual engines* o VEs, *virtual private servers* o VPS, o *jails*). Permiten al usuario y a las aplicaciones obtener la misma vista que si se tratara de un server real. Implementación avanzada del mecanismo de *chroot*. El kernel ofrece características de administración de recursos para garantizar el uso equitativo de los mismos entre los contenedores.
  - VServer, OpenVZ

Dominios o anillos de protección

- Orden de privilegios impuesto por el hardware

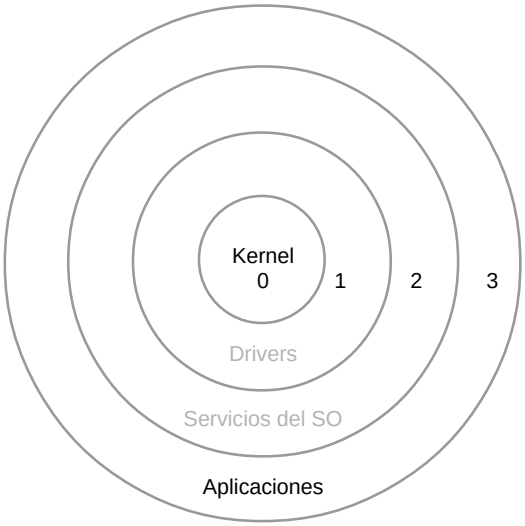

Figura 11: Anillos de protección, normalmente 1 y 2 sin uso

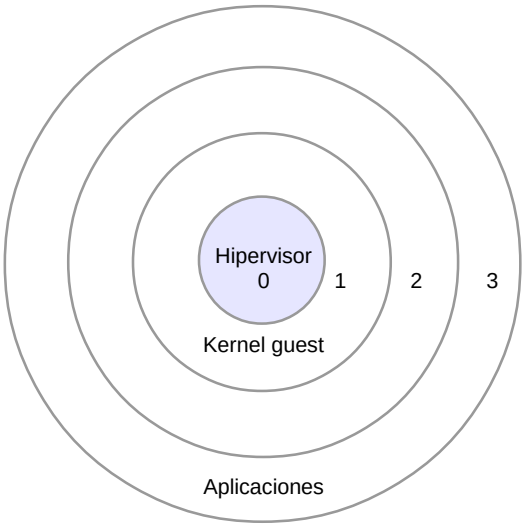

Figura 12: Hipervisor de tipo 1 y sistema guest

- Kernel, anillo 0
- Drivers, anillos 1 y 2
- Aplicaciones, procesos de usuario, anillo 3

- Posibilita el mecanismo de los system calls
  - El código que se ejecuta en un anillo más exterior no puede ejecutar instrucciones de los anillos más interiores ni acceder a memoria no asignada
  - El salto a los anillos más interiores se hace por puntos de acceso predefinidos (las llamadas al sistema)
  - Normalmente los kernels monolíticos corren en modo supervisor o kernel anillo 0, y las aplicaciones en modo usuario en el anillo 3 (Fig. 11)
- Hardware de hipervisor (VTx, AMD-V)
  - Característica de algunos procesadores desde 2005
  - Crea un anillo modo 1 donde ejecutar el código privilegiado de las máquinas virtuales (Fig. 12)
- Hipervisor o VMM
  - Tipo 1
    - Corre directamente sobre el hardware (Bare Metal)
    - XenServer, VMware ESX/ESXi, Hyper-V, Oracle VM Server
  - Tipo 2
    - Corre sobre un sistema operativo dado
    - VMware Workstation, VirtualBox, KVM

2. Aplicaciones

- Para el usuario final
  - Revolución del multicore
    - Muchos equipos en uno
  - Probar nuevas distribuciones u otro software
  - Probar actualizaciones
- Para el administrador de sistemas
  - Independencia del hardware
  - Sistemas *legacy*
  - *Provisioning*
  - *Live Migration* y balance de carga
  - Validación de backups, *staging*
  - Hosting de servicios
  - Consolidación de servidores
    - Menos espacio
    - Menos consumo de energía
    - Menos calor disipado
  - Aumentar la disponibilidad

3. Proxmox

- PVE - Proxmox Virtual Environment
- Infraestructura para administración de recursos de virtualización
- Virtualización

- KVM
  - Incorporado al kernel Linux
  - Soporta múltiples SOs
  - Almacenamiento en archivos o volúmenes
  - Raw o QCOW2
  - Compatible con otros SO pero costoso en CPU
- Consolas de acceso a VMs
  - SSH
  - VNC
  - QXL/Spice
  - HTML5

- Containers
  - OpenVZ
    - Mismo kernel que el host
    - Cada contenedor tiene sus propios archivos, bibliotecas, aplicaciones, /proc, /sys, locks, usuarios, grupos, árbol de procesos, red, dispositivos, instancias de IPC
    - Puede dárseles acceso a dispositivos reales
    - Soporta únicamente Linux
    - Almacenamiento en directorio del host
    - Mejor performance
    - Turnkey Linux aporta *appliances*

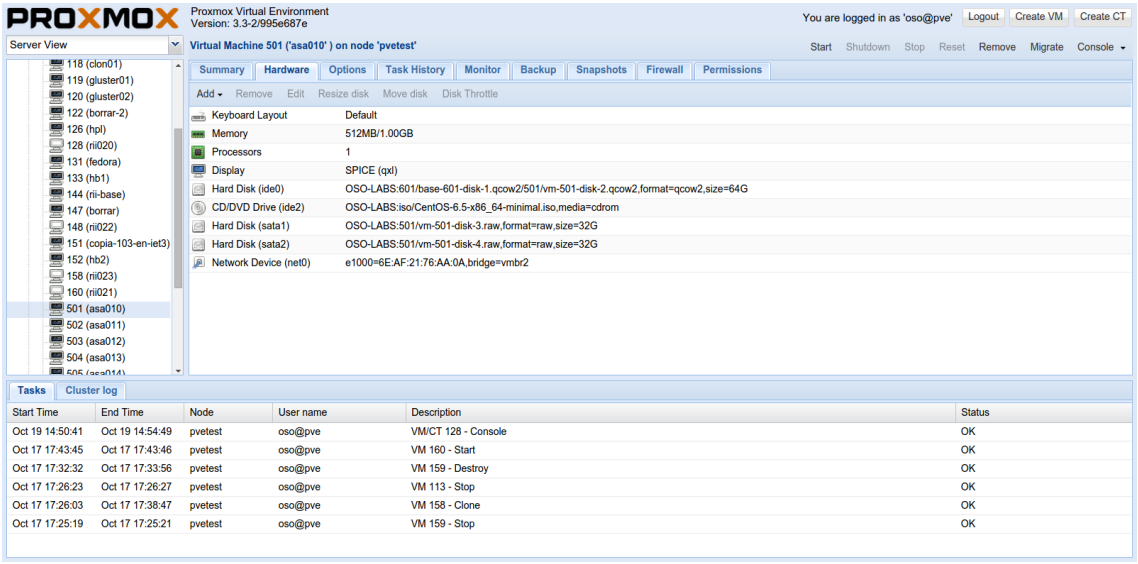

Figura 13: Cliente web de Proxmox

Línea de comandos

Comando qm

```
qm start <vmID>              start vm
qm stop <vmID>               kill vm (immediate stop)
qm shutdown <vmID>           gracefully stop vm (send poweroff)
qm reboot <vmID>             reboot vm (shutdown, start)
```

Utilizar API	pvesh
Estudiar performance	pveperf
Administrar el cluster de PVEs	pvecm
Administrar contenedores OpenVZ	vzctl
Administrar VMs	qm
Administrar templates	pveam

Cuadro 5: Herramientas de administración Proxmox

```
qm reset <vmID>              reset vm (stop, start)
qm suspend <vmID>            suspend vm
qm resume <vmID>             resume vm
qm destroy <vmID>            destroy vm (delete all files)
qm startall                  start all virtual machines
qm stopall [timeout]         stop all virtual machines
```

Script para agregar discos

```
#!/bin/bash

for ((m=501; m<=510; m+=1))
do
       for d in 1 2; do qm set $m -sata$d volume=OSO-LABS:32; done
       qm config $m
done
```

```
root@pvetest:~# qm config 501
balloon: 512
bootdisk: ide0
cores: 1
description: TUASSL-asa010
ide0: OSO-LABS:601/base-601-disk-1.qcow2/501/vm-501
    disk-2.qcow2,format=qcow2,size=64G
ide2: OSO-LABS:iso/CentOS-6.5-x86_64-minimal.iso,media=cdrom
memory: 1024
name: asa010
net0: e1000=6E:AF:21:76:AA:0A,bridge=vmbr2
ostype: l26
sata1: OSO-LABS:501/vm-501-disk-3.raw,format=raw,size=32G
sata2: OSO-LABS:501/vm-501-disk-4.raw,format=raw,size=32G
sockets: 1
vga: qxl
```

Otras características

- Almacenamiento local o de red iSCSI, NFS, etc
- Importación/exportación a VMWare
- Cliente Web, login Linux/PVE
- Clustering
  - Simétrico, sin nodo central de administración
  - Cluster File System, importante para preservar la disponibilidad

- Backup + Restore
  - Scheduling, compresión, Live Snapshots
  - Migración en vivo
- Administración de recursos
  - Pools de almacenamiento asignables a usuarios

4. Cloud Computing

- Las Clouds son grandes reservorios o *pools* de recursos virtualizados, fácilmente usables y accesibles (tales como hardware, plataformas de desarrollo y/o servicios).
- Estos recursos pueden ser dinámicamente reconfigurados para ajustarse a una carga variable, permitiendo también una utilización óptima de recursos.
- Este reservorio de recursos es típicamente explotado a través de un modelo de pago por uso, en el cual los proveedores de infraestructura ofrecen garantías mediante Acuerdos de Nivel de Servicio (SLAs) adaptados a los usuarios.

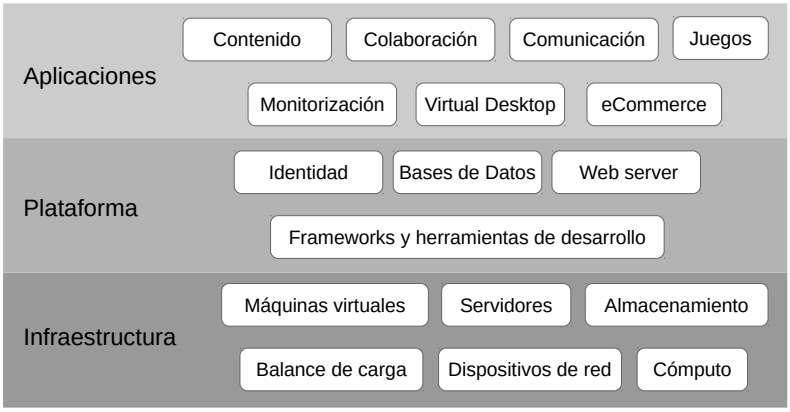

Figura 14: Modelos de servicio y recursos en Cloud Computing

Propiedades de Cloud Computing

- Autoservicio a demanda (*On-Demand Self-Service*)
- Amplio acceso a través de la red
- *Pooling* de recursos
- Elasticidad rápida
- Servicio mensurado

	IaaS	PaaS	SaaS	Open Source
Amazon AWS	✓	✓		
Apache CloudStack	✓			✓
Docker	✓			✓
Dropbox	✓			
Eucalyptus	✓			✓
Google App Engine		✓		
Google Apps		✓	✓	
MS Azure	✓	✓		
OpenNebula	✓			✓
OpenStack	✓			✓
OwnCloud	✓		✓	✓

Cuadro 6: Ejemplos de servicios Cloud

Modelos de servicio en Cloud Computing

IaaS, Infrastructure-as-a-Service Máquinas físicas o virtuales, y otros recursos. Bibliotecas de imágenes de máquinas virtuales, almacenamiento de bloques, almacenamiento de archivos o de objetos. Firewalls, balanceadores de carga, direcciones IP, VLANs, paquetes de software. Los usuarios instalan sistemas operativos sobre la infraestructura provista por la Cloud.

PaaS, Platform-as-a-Service El proveedor ofrece una plataforma de computación, generalmente incluyendo sistema operativo, lenguajes de programación, ambiente de ejecución de programas, bases de datos, web servers. Los desarrolladores de aplicaciones escriben y corren sus soluciones de software sobre la infraestructura provista evitando los costos de comprar y administrar el hardware y software necesario. La Cloud suele adaptar los recursos necesarios a la demanda del usuario.

SaaS, Software-as-a-Service Los usuarios reciben acceso a software y bases de datos. Los proveedores instalan y operan software de aplicación en la Cloud. Los usuarios no lo administran, ni tienen conocimiento de la infraestructura (generalmente elástica) que soporta esas aplicaciones.

Objetos gestionados

1. Aplicaciones
2. Datos
3. Runtime
4. Middleware
5. OS
6. Virtualización
7. Servers
8. Storage
9. Red

Según el modelo de servicio ofrecido, el usuario final gestiona los objetos:

- Software empaquetado: 1-9
- IaaS: 1-5
- PaaS: 1-2
- SaaS: Nada

Parte VII
Anexos

A. iptables.log

```
Logged 539 packets on interface eth1
  From 0000:0000:1011:1213:0100:0000:0000:0000 - 3 packets to icmpv6(130)
  From 0000:0000:0000:859e:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0000:0023:ff53:4d42:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0000:0000:0000:3433:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0000:0000:0000:3132:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0000:0000:0000:7d3a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0000:0000:0000:6e63:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0000:0000:0000:937f:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0000:0000:0000:0000:0000:0000:0000:0000 - 2 packets to icmpv6(130)
  From 0000:0000:0000:0000:0100:0000:0000:0000 - 40 packets to icmpv6(130)
  From 0000:0000:0000:6569:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 000e:175f:531c:580e:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0011:11db:a2d4:0a00:0100:0000:0000:0000 - 2 packets to icmpv6(130)
  From 002c:799a:0694:8c26:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0100:0000:0600:0000:0100:0000:0000:0000 - 2 packets to icmpv6(130)
  From 0101:080a:00c6:0621:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:00c6:f565:0007:994d:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:00c7:39ad:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:00c7:3e49:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:00c7:5bd1:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:1551:7183:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:1a9a:1543:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4553:94db:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4557:126c:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4559:6ffd:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4559:e9ac:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:455a:3fd8:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:455a:cc78:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:455c:c658:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:455d:4147:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:455d:bbfc:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:455e:3351:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4567:104c:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4575:8fa3:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4575:fcd3:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4584:d388:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:458c:f368:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0101:080a:4594:8809:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0102:0417:0000:0000:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0102:a16d:0a00:00c8:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 0204:05b4:0402:080a:0100:0000:0000:0000 - 2 packets to icmpv6(130)
  From 0300:0000:0400:0000:0100:0000:0000:0000 - 3 packets to icmpv6(130)
  From 036b:696d:0675:6e63:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 10d4:d5cb:f80c:14d5:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 1400:0300:9709:0000:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 2269:6422:3a20:2232:0100:0000:0000:0000 - 1 packet to icmpv6(130)
```

```
  From 3037:3337:3431:3832:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3135:3332:3a38:3439:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3234:3238:323a:3834:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3333:3238:323a:3834:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3335:3839:323a:3834:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3336:3532:323a:3834:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3336:3837:3039:3132:0100:0000:0000:0000 - 8 packets to icmpv6(130)
  From 3430:3734:3330:3532:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3432:3230:3239:3a33:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3432:3635:3239:3a33:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3433:3230:3332:3a38:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3433:3534:3039:3a33:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3734:3237:3339:313a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3734:3330:3238:313a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3734:3330:3636:313a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3734:3330:3930:323a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3734:3334:3533:313a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 3734:3334:3736:393a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 616a:6f72:223a:2031:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 6d65:6e74:7322:3a7b:0100:0000:0000:0000 - 9 packets to icmpv6(130)
  From 6f20:3134:3037:3433:0100:0000:0000:0000 - 2 packets to icmpv6(130)
  From 7473:223a:7b7d:7d00:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 7473:223a:7b7d:7d32:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From 7473:223a:7b7d:7d3a:0100:0000:0000:0000 - 4 packets to icmpv6(130)
  From 7473:223a:7b7d:7d7b:0100:0000:0000:0000 - 3 packets to icmpv6(130)
  From 756e:6e69:6e67:223a:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From bf1d:f661:984e:fcb0:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From c011:fdda:43fe:4d59:65fe:e1aa:c7d5:683a - 1 packet to icmpv6(130)
  From c012:aa5c:173c:261c:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From c012:cfa3:e641:3b5f:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From c013:4b15:1c15:3311:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From c013:9389:bcf8:f7d4:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From c014:ff7d:0000:0000:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From e063:837f:0000:0000:0100:0000:0000:0000 - 3 packets to icmpv6(130)
  From e063:837f:2077:937f:0100:0000:0000:0000 - 1 packet to icmpv6(130)
  From e063:837f:301f:937f:0100:0000:0000:0000 - 3 packets to icmpv6(130)
  From 0.0.0.0 - 154 packets to igmp(0)
  From 10.0.3.4 - 154 packets to igmp(0)
  From 10.0.3.21 - 75 packets to igmp(0)
  From 10.0.3.209 - 2 packets to igmp(0)

Listed by source hosts:
Logged 18 packets on interface virbr0
  From fe80:0000:0000:0000:5054:00ff:feed:8246 - 18 packets to udp(5353)

Listed by source hosts:
Logged 50 packets on interface wlan0
  From 10.0.4.1 - 2 packets to udp(68)
  From 64.233.186.188 - 15 packets to tcp(44421,53418)
  From 64.235.151.8 - 5 packets to tcp(56214)
  From 173.194.42.0 - 1 packet to tcp(56677)
  From 173.194.42.21 - 3 packets to tcp(58597)
  From 173.194.42.22 - 7 packets to tcp(35536)
  From 173.194.42.75 - 3 packets to tcp(48585)
  From 173.194.42.85 - 4 packets to tcp(44550)
```

```
  From 173.194.42.86 - 1 packet to tcp(51940)
  From 192.168.1.1 - 2 packets to udp(68)
  From 192.168.2.1 - 2 packets to udp(68)
  From 195.135.221.134 - 1 packet to tcp(51756)
  From 195.154.174.66 - 1 packet to tcp(58047)
  From 200.42.136.212 - 3 packets to tcp(59351,59361)
```

B. Opciones de rsync

```
$ rsync --help
rsync version 3.1.0 protocol version 31
Copyright (C) 1996-2013 by Andrew Tridgell, Wayne Davison, and others.
Web site: http://rsync.samba.org/
Capabilities:
    64-bit files, 64-bit inums, 32-bit timestamps, 64-bit long ints,
    socketpairs, hardlinks, symlinks, IPv6, batchfiles, inplace,
    append, ACLs, xattrs, iconv, symtimes, prealloc

rsync comes with ABSOLUTELY NO WARRANTY. This is free software, and you
are welcome to redistribute it under certain conditions. See the GNU
General Public Licence for details.

rsync is a file transfer program capable of efficient remote update
via a fast differencing algorithm.

Usage: rsync [OPTION]... SRC [SRC]... DEST
  or   rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST
  or   rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST
  or   rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST
  or   rsync [OPTION]... [USER@]HOST:SRC [DEST]
  or   rsync [OPTION]... [USER@]HOST::SRC [DEST]
  or   rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]
The ':' usages connect via remote shell, while '::' & 'rsync://' usages connect
to an rsync daemon, and require SRC or DEST to start with a module name.

Options
 -v, --verbose             increase verbosity
     --info=FLAGS          fine-grained informational verbosity
     --debug=FLAGS         fine-grained debug verbosity
     --msgs2stderr         special output handling for debugging
 -q, --quiet               suppress non-error messages
     --no-motd             suppress daemon-mode MOTD (see manpage caveat)
 -c, --checksum            skip based on checksum, not mod-time & size
 -a, --archive             archive mode; equals -rlptgoD (no -H,-A,-X)
     --no-OPTION           turn off an implied OPTION (e.g. --no-D)
 -r, --recursive           recurse into directories
 -R, --relative            use relative path names
     --no-implied-dirs     don't send implied dirs with --relative
 -b, --backup              make backups (see --suffix & --backup-dir)
     --backup-dir=DIR      make backups into hierarchy based in DIR
     --suffix=SUFFIX       set backup suffix (default ~ w/o --backup-dir)
 -u, --update              skip files that are newer on the receiver
```

```
    --inplace             update destination files in-place (SEE MAN PAGE)
    --append              append data onto shorter files
    --append-verify       like --append, but with old data in file checksum
-d, --dirs                transfer directories without recursing
-l, --links               copy symlinks as symlinks
-L, --copy-links          transform symlink into referent file/dir
    --copy-unsafe-links   only "unsafe" symlinks are transformed
    --safe-links          ignore symlinks that point outside the source tree
    --munge-links         munge symlinks to make them safer (but unusable)
-k, --copy-dirlinks       transform symlink to a dir into referent dir
-K, --keep-dirlinks       treat symlinked dir on receiver as dir
-H, --hard-links          preserve hard links
-p, --perms               preserve permissions
-E, --executability       preserve the file's executability
    --chmod=CHMOD         affect file and/or directory permissions
-A, --acls                preserve ACLs (implies --perms)
-X, --xattrs              preserve extended attributes
-o, --owner               preserve owner (super-user only)
-g, --group               preserve group
    --devices             preserve device files (super-user only)
    --specials            preserve special files
-D                        same as --devices --specials
-t, --times               preserve modification times
-O, --omit-dir-times      omit directories from --times
-J, --omit-link-times     omit symlinks from --times
    --super               receiver attempts super-user activities
    --fake-super          store/recover privileged attrs using xattrs
-S, --sparse              handle sparse files efficiently
    --preallocate         allocate dest files before writing them
-n, --dry-run             perform a trial run with no changes made
-W, --whole-file          copy files whole (without delta-xfer algorithm)
-x, --one-file-system     don't cross filesystem boundaries
-B, --block-size=SIZE     force a fixed checksum block-size
-e, --rsh=COMMAND         specify the remote shell to use
    --rsync-path=PROGRAM  specify the rsync to run on the remote machine
    --existing            skip creating new files on receiver
    --ignore-existing     skip updating files that already exist on receiver
    --remove-source-files sender removes synchronized files (non-dirs)
    --del                 an alias for --delete-during
    --delete              delete extraneous files from destination dirs
    --delete-before       receiver deletes before transfer, not during
    --delete-during       receiver deletes during the transfer
    --delete-delay        find deletions during, delete after
    --delete-after        receiver deletes after transfer, not during
    --delete-excluded     also delete excluded files from destination dirs
    --ignore-missing-args ignore missing source args without error
    --delete-missing-args delete missing source args from destination
    --ignore-errors       delete even if there are I/O errors
    --force               force deletion of directories even if not empty
    --max-delete=NUM      don't delete more than NUM files
    --max-size=SIZE       don't transfer any file larger than SIZE
    --min-size=SIZE       don't transfer any file smaller than SIZE
    --partial             keep partially transferred files
    --partial-dir=DIR     put a partially transferred file into DIR
    --delay-updates       put all updated files into place at transfer's end
```

```
-m, --prune-empty-dirs   prune empty directory chains from the file-list
    --numeric-ids        don't map uid/gid values by user/group name
    --usermap=STRING     custom username mapping
    --groupmap=STRING    custom groupname mapping
    --chown=USER:GROUP   simple username/groupname mapping
    --timeout=SECONDS    set I/O timeout in seconds
    --contimeout=SECONDS set daemon connection timeout in seconds
-I, --ignore-times       don't skip files that match in size and mod-time
-M, --remote-option=OPTION send OPTION to the remote side only
    --size-only          skip files that match in size
    --modify-window=NUM  compare mod-times with reduced accuracy
-T, --temp-dir=DIR       create temporary files in directory DIR
-y, --fuzzy              find similar file for basis if no dest file
    --compare-dest=DIR   also compare destination files relative to DIR
    --copy-dest=DIR      ... and include copies of unchanged files
    --link-dest=DIR      hardlink to files in DIR when unchanged
-z, --compress           compress file data during the transfer
    --compress-level=NUM explicitly set compression level
    --skip-compress=LIST skip compressing files with a suffix in LIST
-C, --cvs-exclude        auto-ignore files the same way CVS does
-f, --filter=RULE        add a file-filtering RULE
-F                       same as --filter='dir-merge /.rsync-filter'
                         repeated: --filter='- .rsync-filter'
    --exclude=PATTERN    exclude files matching PATTERN
    --exclude-from=FILE  read exclude patterns from FILE
    --include=PATTERN    don't exclude files matching PATTERN
    --include-from=FILE  read include patterns from FILE
    --files-from=FILE    read list of source-file names from FILE
-0, --from0              all *-from/filter files are delimited by 0s
-s, --protect-args       no space-splitting; only wildcard special-chars
    --address=ADDRESS    bind address for outgoing socket to daemon
    --port=PORT          specify double-colon alternate port number
    --sockopts=OPTIONS   specify custom TCP options
    --blocking-io        use blocking I/O for the remote shell
    --stats              give some file-transfer stats
-8, --8-bit-output       leave high-bit chars unescaped in output
-h, --human-readable     output numbers in a human-readable format
    --progress           show progress during transfer
-P                       same as --partial --progress
-i, --itemize-changes    output a change-summary for all updates
    --out-format=FORMAT  output updates using the specified FORMAT
    --log-file=FILE      log what we're doing to the specified FILE
    --log-file-format=FMT log updates using the specified FMT
    --password-file=FILE read daemon-access password from FILE
    --list-only          list the files instead of copying them
    --bwlimit=RATE       limit socket I/O bandwidth
    --outbuf=N|L|B       set output buffering to None, Line, or Block
    --write-batch=FILE   write a batched update to FILE
    --only-write-batch=FILE like --write-batch but w/o updating destination
    --read-batch=FILE    read a batched update from FILE
    --protocol=NUM       force an older protocol version to be used
    --iconv=CONVERT_SPEC request charset conversion of filenames
    --checksum-seed=NUM  set block/file checksum seed (advanced)
-4, --ipv4               prefer IPv4
-6, --ipv6               prefer IPv6
```

```
    --version             print version number
(-h) --help               show this help (-h is --help only if used alone)

Use "rsync --daemon --help" to see the daemon-mode command-line options.
Please see the rsync(1) and rsyncd.conf(5) man pages for full documentation.
See http://rsync.samba.org/ for updates, bug reports, and answers
```

C. Ejemplo completo de configuración DRBD

```
resource example {
       options {
              on-no-data-accessible suspend-io;
       }

       net {
              cram-hmac-alg "sha1";
              shared-secret "secret_string";
       }

       # The disk section is possible on resource level and in each
       # volume section
       disk {
              # If you have a resonable RAID controller
              # with non volatile write cache (BBWC, flash)
              disk-flushes no;
              disk-barrier no;
              md-flushes no;
       }

       # volume sections on resource level, are inherited to all node
       # sections. Place it here if the backing devices have the same
       # device names on all your nodes.
       volume 1 {
              device minor 1;
              disk /dev/sdb1;
              meta-disk internal;

              disk {
                     resync-after example/0;
              }
       }

       on wurzel {
              address 192.168.47.1:7780;

              volume 0 {
                     device minor 0;
                     disk /dev/vg_wurzel/lg_example;
                     meta-disk /dev/vg_wurzel/lv_example_md;
              }
       }
       on sepp {
```

```
                address 192.168.47.2:7780;

                volume 0 {
                        device minor 0;
                        disk /dev/vg_sepp/lg_example;
                        meta-disk /dev/vg_sepp/lv_example_md;
                }
        }
}

resource "ipv6_example_res" {
        net {
                cram-hmac-alg "sha1";
                shared-secret "ieho4CiiUmaes6Ai";
        }

        volume 2 {
                device "/dev/drbd_fancy_name" minor 0;
                disk   /dev/vg0/example2;
                meta-disk internal;
        }

        on amd {
                # Here is an example of ipv6.
                # If you want to use ipv4 in ipv6 i.e. something like
                    [::ffff:192.168.22.11]
                # you have to set disable-ip-verification in the global section.
                address ipv6 [fd0c:39f4:f135:305:230:48ff:fe63:5c9a]:7789;
        }

        on alf {
                address ipv6 [fd0c:39f4:f135:305:230:48ff:fe63:5ebe]:7789;
        }
}


#
# A two volume setup with a node for disaster recovery in an off-site location.
#

resource alpha-bravo {
        net {
                cram-hmac-alg "sha1";
                shared-secret "Gei6mahcui4Ai0Oh";
        }

        on alpha {
                volume 0 {
                        device minor 0;
                        disk /dev/foo;
                        meta-disk /dev/bar;
                }
                volume 1 {
                        device minor 1;
                        disk /dev/foo1;
```

```
                        meta-disk /dev/bar1;
                }
                address 192.168.23.21:7780;
        }
        on bravo {
                volume 0 {
                        device minor 0;
                        disk /dev/foo;
                        meta-disk /dev/bar;
                }
                volume 1 {
                        device minor 1;
                        disk /dev/foo1;
                        meta-disk /dev/bar1;
                }
                address 192.168.23.22:7780;
        }
}

resource stacked_multi_volume {
        net {
                protocol A;

                on-congestion pull-ahead;
                congestion-fill 400M;
                congestion-extents 1000;
        }

        disk {
                c-fill-target 10M;
        }

        volume 0 { device minor 10; }
        volume 1 { device minor 11; }

        proxy {
                memlimit 500M;
                plugin {
                        lzma contexts 4 level 9;
                }
        }

        stacked-on-top-of alpha-bravo {
                address 192.168.23.23:7780;

                proxy on charly {
                        # In the regular production site, there is a dedicated host
                            to run
                        # DRBD-proxy
                        inside   192.168.23.24:7780; # for connections to DRBD
                        outside  172.16.17.18:7780; # for connections over the WAN
                            or VPN
                        options {
                                memlimit 1G; # Additional proxy options are
                                    possible here
```

```
                        }
                }
        }
        on delta {
                volume 0 {
                        device minor 0;
                        disk /dev/foo;
                        meta-disk /dev/bar;
                }
                volume 1 {
                        device minor 1;
                        disk /dev/foo1;
                        meta-disk /dev/bar1;
                }
                address 127.0.0.2:7780;

                proxy on delta {
                        # In the DR-site the proxy runs on the machine that stores
                            the data
                        inside 127.0.0.1:7780;
                        outside 172.16.17.19:7780;
                }
        }
}

resource drbd_9_two_connection {
        volume 0 {
               device minor 10;
               disk /dev/foo/bar;
               meta-disk internal;
        }

        on alpha {
                node-id 0;
                address 192.168.31.1:7800;
        }
        on bravo {
                node-id 1;
                address 192.168.31.2:7800;
        }
        on charlie {
                node-id 2;
                address 192.168.31.3:7800;
        }

        net {
                ko-count 3;
        }

        connection "optional name" {
                host alpha;
                host bravo;
                net { protocol C; }
        }
```

```
        connection {
                host alpha address 127.0.0.1:7800 via proxy on alpha {
                        inside 127.0.0.2:7800;
                        outside 192.168.31.1:7801;
                }
                host charlie address 127.0.0.1:7800 via proxy on charlie {
                        inside 127.0.0.2:7800;
                        outside 192.168.31.3:7800;
                }
                net { protocol A; }
        }

        connection {
                host bravo address 127.0.0.1:7800 via proxy on bravo {
                        inside 127.0.0.2:7800;
                        outside 192.168.31.2:7801;
                }
                host charlie address 127.0.0.1:7800 via proxy on charlie {
                        inside 127.0.0.2:7800;
                        outside 192.168.31.3:7800;
                }
                net { protocol A; }
        }
}

resource drbd_9_mesh {
        volume 0 {
               device minor 11;
               disk /dev/foo/bar2;
               meta-disk internal;
        }

        on alpha {
                node-id 0;
                address 192.168.31.1:7900;
        }
        on bravo {
                node-id 1;
                address 192.168.31.2:7900;
        }
        on charlie {
                node-id 2;
                address 192.168.31.3:7900;
        }

        connection-mesh {
                hosts alpha bravo charlie;
                net {
                        protocol C;
                }
        }
}
```