Sistemas de almacenamiento distribuido y redes

PID_00286210

Remo Suppi Boldrito

Tiempo mínimo de dedicación recomendado: 2 horas

Universitat Oberta de Catalunya

Remo Suppi Boldrito

Ingeniero de Telecomunicaciones. Doctor en Informática por la Universidad Autónoma de Barcelona (UAB). Profesor del Departamento de Arquitectura de Computadores y Sistemas Operativos en la UAB.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Josep Jorba Esteve

Primera edición: febrero 2022

Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Remo Suppi Boldrito
Producción: FUOC

Índice

Objetivos

Los objetivos principales de este módulo son los siguientes:

1. Conocer los diferentes aspectos de las redes de almacenamiento, su arquitectura, protocolos y herramientas.

2. Analizar diferentes casos de uso para el almacenamiento: redundancia (RAID), distribuido (GlusterFS) y lógico (LVM).

3. Estudiar diferentes aspectos de redes y su monitorización, así como el concepto de SDN, su arquitectura y pruebas y su configuración a través de simuladores.

4. Analizar diferentes tipos de software estándar para la creación y utilización de *switchs* virtuales.

En este módulo, el estudiante deberá centrar su atención en los siguientes conceptos fundamentales:

- Redes de almacenamiento.
- Arquitecturas y configuración de sistemas de almacenamiento.
- Redes de datos y SDN.
- Monitorización de redes y configuración de *switchs* virtuales.

Se recomienda analizar y realizar pruebas de instalación y configuración de las principales herramientas indicadas.

1. La importancia del almacenamiento: desde redes locales al *big data* y *cloud*

En este módulo, el estudiante comenzará leyendo el capítulo 1 de ***Network Storage***[1] **(James O'Reilly)** del repositorio de la UOC (O'Reilly), donde se presenta cómo está aumentando el impacto del almacenamiento en la tecnología de la información, a medida que la velocidad y la tasa de transacciones se ajustan a la ley de Moore. Con este concepto se busca justificar el inicio de una «explosión» de almacenamiento, tanto en capacidad como en la carga de la red, que cada año se incrementa en valores nunca alcanzados anteriormente. En el capítulo 2, el estudiante consolidará diferentes conceptos y realizará una revisión sobre el almacenamiento y su evolución desde la perspectiva de la red; continuará con los capítulos 3 y 4 para hacer una revisión del estado actual de las redes de almacenamiento, los diferentes protocolos y esquemas de conexión, cómo se diferencian en casos de uso y comportamiento operativo, y una breve historia de cada uno a medida que ha ido evolucionando, junto con sus fortalezas y debilidades, entre otros aspectos.

[(1)]Podéis encontrar el libro del repositorio UOC en el siguiente enlace: https://bit.ly/3r9VgMg

En el capítulo 8, el estudiante se aproximará al *big data* y cómo este afecta al almacenamiento por su falta de estructura y el modo como se utilizan estos datos, que hacen que sean necesarios nuevos enfoques. Finalmente, en los capítulos 9 y 10 la aproximación del almacenamiento será hacia el *high performace computing* (HPC) y hacia el *cloud*. Si bien el HPC es similar al *big data* en muchos aspectos, es mucho más sensible a la latencia de acceso y afecta a las redes, y de ahí el auge de redes como InfiniBand para llegar a los dispositivos de almacenamiento. En el *cloud*, el problema es totalmente diferente, ya que mover datos a internet significa otra dimensión para las redes de almacenamiento, con *gateways*, compresión y deduplicación, por un lado, y protocolos de transferencia eficientes, por el otro, lo cual implica nuevas formas y métodos para implementarlo de manera eficiente.

Como lectura recomendada se deja al estudiante los capítulos 11 y 12, en los que el autor reflexiona sobre aspectos no menos importantes en el almacenamiento, como son duplicación de la información (*data integrity*) y privacidad de la información (*data security*). Se debe tener en cuenta que la integridad de los datos es un problema importante en el almacenamiento, ya que las unidades y los controladores fallan, y en los discos aparecen fallos que provocan que los datos sean ilegibles. Los esquemas de protección de datos han evolucionado rápidamente, centrados en la replicación y la codificación de borrado, y han obtenido mejoras notables en la protección de los datos almacenados. En cuanto a seguridad, más allá de la pérdida o «secuestro» de los datos, un aspecto crucial es la privacidad, que tiene por objetivo proteger tanto el dato

almacenado como su tránsito por medio de técnicas de cifrado que permitan que los datos no puedan ser visualizados no solo cuando están en el disco sino también cuando viajan por la red.

Para los estudiantes que deseen ver cuáles son las tecnologías actuales, teniendo en cuenta que este libro es del 2016, se recomienda la lectura de los capítulos 13 y 14, en los que se muestran los principales avances en discos sólidos (tecnología NAND) y nuevas alternativas donde el almacenamiento se «aleja» del silicio.

2. Casos de uso: RAID, GlusterFS, LVM

2.1. RAID

Como se analizó en el capítulo 11, la pérdida de información por fallo en las unidades, controladores o discos es un punto importante en el almacenamiento. La solución actual más aplicada en el RAID (*redundant array of independent disks*).

El RAID es una tecnología de almacenamiento de datos que permite que los datos estén distribuidos en diferentes discos, lo que posibilita la redundancia de datos y mejora el rendimiento.

Los datos se distribuyen entre las unidades de varias formas, denominadas ***niveles RAID***, según el nivel requerido de redundancia y rendimiento. Los diferentes esquemas, o diseños de distribución de datos, se nombran con la palabra *RAID* seguida de un número, por ejemplo, RAID 0 o RAID 1 y cada esquema, o nivel de RAID, proporciona un equilibrio diferente entre los objetivos clave: confiabilidad, disponibilidad, rendimiento y capacidad. Los niveles de RAID superiores a RAID 0 brindan protección contra errores de lectura de sector irrecuperables, así como contra fallos de unidades físicas completas.

El concepto *RAID* tiene más de tres décadas (Patterson, Gibson y Katz, 1988), pero algunos de sus niveles originales y sus evoluciones siguen siendo útiles en nuestros días. Como se ha explicado anteriormente (en el capítulo 11), muchos niveles de RAID emplean un esquema de protección contra errores llamado ***paridad***, basado en un XOR simple, a diferencia de RAID 6, que utiliza dos paridades separadas basadas respectivamente en la suma y la multiplicación. Con el avance de los SSD, el RAID también puede proporcionar seguridad de datos con unidades de estado sólido (SSD) sin que todos los discos sean SSD, utilizando discos magnéticos (mecánicos), pero se debe utilizar un controlador apropiado que utilice el SSD para todas las operaciones de lectura (algunos fabricantes llaman a esta configuración *RAID híbrido*).

Originalmente, había cinco niveles estándar de RAID, pero han evolucionado y actualmente encontramos muchas variaciones, incluidos varios niveles anidados y muchos niveles no estándar (en su mayoría, propietarios). Los niveles de RAID y sus formatos de datos asociados están estandarizados por la Storage Networking Industry Association (SNIA) en el estándar *Common RAID Disk Drive Format* (DDF) como:

- **RAID 0**: consta de bandas (*striping*, división lógica de datos secuenciales), pero sin duplicación ni paridad. Su capacidad es la suma de las capacidades de los discos involucrados, pero al distribuir los datos de cada archivo entre todas las unidades del conjunto, no soluciona el problema de los fallos (y la tasa promedio de fallos del volumen aumenta, ya que hay más unidades conectadas). Dependiendo de cómo se hayan distribuido los contenidos, el rendimiento de las operaciones de lectura y escritura en cualquier archivo se puede multiplicar por el número de unidades.

- **RAID 1** (*mirror*): consiste en duplicación de datos y sin paridad, donde cualquier unidad del conjunto puede atender cualquier solicitud de lectura, lo que mejora el rendimiento. El espacio de almacenamiento es la mitad del total (con discos enteros).

- **RAID 2, 3 y 4**: incluyen paridad y mejoras, pero no se utilizan en la actualidad porque la funcionalidad y protección es mejor en RAID 5/6.

- **RAID 5**: consta de bandas en el nivel de bloque con paridad distribuida, lo que requiere que todas las unidades menos una estén presentes para funcionar. En caso de fallo de una sola unidad, las lecturas posteriores se pueden calcular a partir de la paridad distribuida, de manera que no se pierdan datos y por ello RAID 5 requiere al menos tres discos.

- **RAID 6**: consta de bandas en el nivel de bloque con paridad distribuida doble, lo que proporciona tolerancia a fallos hasta de dos unidades y provoca que los grupos RAID más grandes sean más prácticos, especialmente para los sistemas de alta disponibilidad, ya que las unidades de gran capacidad tardan más en restaurarse. RAID 6 requiere un mínimo de cuatro discos.

- **Niveles RAID anidados**: muchos controladores de almacenamiento permiten anidar niveles de RAID, donde el *array* final se conoce como *array* superior (por ejemplo, cuando el *array* superior es RAID 0 como en RAID 1 + 0 y RAID 5 + 0, aunque normalmente se elimina el + y se indica con RAID 10 y RAID 50, respectivamente).

Para diseñar un *array*, se recomienda utilizar un calculador de RAID de los muchos que existen en internet.

Ejemplo: calculadores de RAID

Raid-calculator y Giga-Calculator.

Si bien existen controladoras RAID hardware (con el consiguiente incremento de prestaciones), se puede realizar un RAID por software a través de cualquier distribución de Linux. Para crear un RAID software, se utiliza el comando `mdadm` (el nombre deriva de md, *múltiple device* + *admin*, y reemplaza la anterior `mdctl`).

mdadm se instala desde el repositorio (`apt install mdadm`) y como primer paso se deben preparar los discos (pueden ser físicos o virtuales) creando una partición en cada uno de estos del **tipo 0xfd**. Para crear un RAID 5, por ejemplo, llamado /dev/md0 con tres discos (sdb1, sdc1, sdd1), hacemos:

```
mdadm --create --level=5 --raid-devices=3 /dev/md0 /dev/sdb1 /dev/sdc1 /dev/sdd1
```

Se puede modificar el *chunk size* (por ejemplo, agregando `--chunk=16384`), donde valores grandes del *chunk* son mejor para archivos grandes (el valor por defecto es de 512). Recordad también que se debe crear el *filesystem* (`mkfs.ext4 /dev/md0`) antes de montarlo.

Para montar un *array* que no está en el archivo de configuración se debe hacer lo siguiente:

```
mdadm --assemble /dev/md0 /dev/sdb1 /dev/sdc1 /dev/sdd1
```

Para crear un archivo de configuración y que el *array* se inicie durante el arranque de SO hacemos:

```
mdadm --detail --scan >> /etc/mdadm/mdadm.conf
```

Para agregar un nuevo disco y hacer crecer el *array* (si se deriva en un aumento del espacio total, con `resize2fs /dev/md0` se deberá hacer crecer el *filesystem*). Hacemos:

```
mdadm --add /dev/md0 /dev/sde1
```

(este disco quedará como *spare*)

```
mdadm --grow /dev/md0 -raid-devices=4
```

(con ello se verá que el disco pasa de *spare* a activo)

Para marcar un disco con errores y sacarlo del *array* (necesario cuando se quiere cambiar un disco), hacemos:

```
mdadm --fail /dev/md0 /dev/sde1


mdadm --remove /dev/md0 /dev/sde1
```

Para parar el *array* (recordad que se debe desmontar el *filesystem* primero), hacemos:

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

Para iniciar el *array* (después se podrá montar), hacemos:

```
mdadm --run /dev/md0
```

Para obtener información del *array*, hacemos:

```
mdadm --detail /dev/md0
```

Y para monitorizar el *array* y que envíe un correo electrónico (se debe tener cuidado de no caer en un envío reiterado –*spam*– de correos), hacemos:

```
mdadm --monitor --scan --mail=[email address] --delay=1800 &
```

2.2. GlusterFS

Es un sistema de archivos de almacenamiento distribuido y se utiliza en *cloud*, servicios *streaming* o en redes de contenido (CDN).

GlusterFS fue desarrollado originalmente por Gluster, Inc. y después por Red Hat (adquisición en 2011 y en 2012 se integró en Red Hat Storage Server, que, cuando en 2014 RH compró Inktank Storage –desarrollador de Ceph–, le cambió el nombre a Red Hat Gluster Storage).

GlusterFS agrupa varios servidores de almacenamiento a través de interconexión Ethernet o InfiniBand en un gran sistema de archivos de red. Se trata de software libre con licencia GPL/LGPL, su diseño está basado en un *stackable user space* y funciona en una modalidad de cliente y servidor. Normalmente, los servidores proveen «los ladrillos» (*bricks*) de almacenamiento a partir de los cuales se crean volúmenes virtuales (compuestos de varios servidores), mientras que el cliente se conecta a los servidores con un protocolo personalizado a través de TCP/IP, InfiniBand o Sockets Direct Protocol, para acceder a los volúmenes virtuales creados previamente a partir de los *bricks*.

El cliente puede montar el volumen mediante un protocolo nativo de GlusterFS, utilizando el mecanismo FUSE o por NFSv3. GlusterFS proporciona fiabilidad y disponibilidad de datos por medio de varios tipos de réplica: volúmenes replicados y georreplicación, donde los volúmenes replicados garantizan que haya al menos una copia de cada archivo en diferentes «ladrillos», por lo que si uno falla, los datos continuarán almacenados y accesibles. La georreplicación proporciona un modelo de réplica *master/worker*, en el que se copian los volúmenes en ubicaciones geográficamente diferentes y esto se realiza de manera asíncrona, lo que resulta útil para la disponibilidad en caso de que un centro de datos quede fuera de servicio.

En la mayoría de las distribuciones de Linux GlusterFS se encuentra en el repositorio de la distribución y se debe instalar el servidor, pero también el cliente si se desea puede montar el volumen distribuido con el siguiente comando: `apt install glusterfs-client gusterfs-server`.

Desde cualquiera de los servidores que disponemos se necesita descubrir los restantes servidores, para que inicien su sincronización, a través del siguiente comando `gluster peer probe IP_servidor_remoto`. Y se puede comprobar el estado del sistema con `gluster peer status`.

La creación de un volumen (en este ejemplo, llamado *myvol*) pasa por tener definidos y montados los discos (o particiones) que conformarán los *bricks* en cada servidor y luego ejecutar lo siguiente:

```
gluster volume create myvol transport tcp server_A:/export/brick server_B:/export/brick
```

A continuación, se iniciará el volumen con `gluster volume start myvol` y para poder obtener información de su estado se puede ejecutar el comando `gluster volume info`.

Si se desean crear dos réplicas de cada archivo, se debe agregar el parámetro `replica 2`:

```
gluster volume create myvol replica 2 transport tcp server_A:/export/brick server_B:/export/brick
```

Acto seguido, se podrá crear un punto de montaje con, por ejemplo, el comando `mkdir -p /mnt/gluster` y montarlo, verificando su espacio con `dh -h`:

```
mount -t glusterfs server_A:/myvol /mnt/gluster
```

También, si se desea automatizar su montaje al arranque, se puede incluir en */etc/fstab*:

```
server_A:/myvol /mnt/gluster glusterfs rw,noauto 0 0
```

Una manera útil de analizar su funcionalidad es generando una gran cantidad de archivos y mirando su distribución en los *bricks*:

```
for i in `seq 1 25`; do cp -rp /var/log/messages /mnt/gluster/test-$i; done
```

A continuación, se puede verificar en cada *brick* cómo se han distribuido los archivos (que se verán todos juntos en el volumen */mnt/gluster*).

Otras órdenes útiles son las siguientes:

```
gluster volume stop nombre_vol
gluster volumen delete nommbre_vol
```

```
gluster peer detach server_A
gluster peer status
```

2.3. LVM (*logical volume manager*)

Es un entorno que proporciona una gestión de volúmenes para el núcleo de Linux que separa la gestión física del disco de la lógica.

Su creador (H. Mauelshagen) escribió el código original en 1998 basado en las ideas del gestor de volúmenes de HP-UX. LVM se utiliza para los siguientes propósitos:

- Creación de volúmenes lógicos individuales de varios volúmenes físicos o discos enteros (similares a RAID 0), permitiendo cambiar el tamaño del volumen de forma dinámica.

- Gestión de granjas de discos duros que permiten añadir y sustituir discos sin tiempo de inactividad ni interrupción del servicio, en combinación con el intercambio en caliente.

- En lugar de tener que estimar el tamaño que puede necesitar una partición, LVM permite cambiar el tamaño de los sistemas de archivos según sea necesario.

- Permite realizar copias de seguridad consistentes tomando instantáneas de los volúmenes lógicos.

- Permite cifrar varias particiones físicas con una sola contraseña.

- Organiza los discos primero en particiones físicas (*physical volumes*, PV), grupos de volúmenes (*volume groups*, VG) sobre los PV y volúmenes lógicos (*logical volumes*, LV) sobre cada VG.

Para crear un LVM (algunos SO disponen de un asistente, pero se puede hacer en CLI), se deben realizar los siguientes pasos:

1) Definir e inicializar los discos creando una partición de tipo **8e** (con *fdisk*).

2) Definir e inicializar cada *physical volumes* (PV):

```
pvcreate /dev/device [/dev/device]
```

3) Definir el *volume groups* agrupando los PV:

```
vgcreate vg-name /dev/device [/dev/device]
```

4) Inicializar los *logical volumes* sobre cada VG:

```
lvcreate -L size -n lv-name vg-name
```

5) Se puede expandir (o reducir) un volumen con `pvcreate /dev/nou_dev` y luego ejecutando:

```
vgextend vg-name /dev/nou_dev
```

6) Para reducir un volumen, hacemos:

```
vgreduce vg-name /dev/dev
```

Para instalarlo de los repositorios, `apt install lvm2`. Para instalar la interfaz gráfica, `apt install system-config-lvm`.

Para crear, por ejemplo, un volumen lógico (*mylogvol*), primero se debe crear un grupo (*myvol*), con las particiones */dev/sdd1* y */dev/sde1*, un tamaño de 10 Gbytes y montarlo en */mnt/lvm*. Se deberá ejecutar como *root*:

```
pvcreate /dev/sdd1 /dev/sde1
vgcreate myvol /dev/sdd1 /dev/sde1
vgdisplay
lvcreate -n mylogvol -L 10g myvol
mkfs.ext4 /dev/myvol/mylogvol
mkdir /mnt/lvm
mount /dev/myvol/mylogvol /mnt/lvm
df -h
lvdisplay
```

Con el comando `lvremove /dev/myvol/mylogvol` podemos eliminar el LVM creado.

Otros comandos útiles son los siguientes:

- `lvchange`: cambiar los atributos de un *logical volume*.
- `lvconvert`: convertir a *logical volume* a *mirror* o *snapshot*.
- `lvcreate`: crear un *logical volume* en un existente *volume group*.
- `lvdisplay`: mostrar los atributos de un *logical volume*.
- `lvextend`: extender el tamaño de un *logical volume*.

- `lvreduce`: reducir el tamaño de un *logical volume*.
- `lvremove`: eliminar un *logical volume*.
- `lvrename`: renombrar un *logical volume*.
- `lvresize`: redimensionar un *logical volume*.
- `lvs`: obtener información de los *logical volumes*.

3. Redes

En una red de una infraestructura para el procesamiento de datos, es importante conocer los niveles de rendimiento (*throughput*) y su ancho de banda, ya que de estos parámetros dependerá la eficiencia del sistema. El **rendimiento** indica cuántos datos se han transferido desde una fuente en un momento dado, mientras que el **ancho de banda** indica cuántos datos se podrían transferir teóricamente desde una fuente en cada momento. El administrador/gestor de la infraestructura deberá saber en cada momento cuál es el rendimiento y el ancho de banda tanto interno como de los proveedores externos (si las aplicaciones están externamente interconectadas, por ejemplo, al *cloud*) para actuar en consecuencia.

Por ello, se debe diferenciar claramente cuál es el rendimiento y el ancho de banda, cuál es la diferencia hay entre estos y por qué importa. La respuesta breve es la **velocidad**.

> La velocidad es uno de los parámetros más importantes que se utilizan para medir el rendimiento de la red. La velocidad con la que viajan los paquetes desde la fuente hasta el destino es la que determina la cantidad de información que se puede enviar en un intervalo considerado.

Una velocidad de red lenta es igual a bajo rendimiento de las aplicaciones de procesamiento de datos y esto equivale a aplicaciones condicionadas en su ejecución por este parámetro que actúa como cuello de botella.

El rendimiento (*throughout*) de red mide cuántos paquetes llegan a sus destinos con éxito y generalmente se mide en bits por segundo. Cuando se utilizan aplicaciones en *big data* se quiere que los paquetes de datos «viajen» a la máxima velocidad, y cuando los paquetes se pierden y deben ser reenviados esto supone un rendimiento de las aplicaciones deficiente o lento.

Las magnitudes principales que pueden dar información sobre los problemas de una red son la pérdida de paquetes, la latencia y la fluctuación, ya que están relacionadas con la velocidad de transmisión. La **latencia** (*latency*) es la cantidad de tiempo que tarda un paquete desde el origen hasta el destino, mientras que la **fluctuación** (*jitter*) hace referencia a la diferencia de retraso de cada paquete. La minimización de estos factores es fundamental para aumentar la velocidad e incrementar las prestaciones de las aplicaciones distribuidas de *big data*. Un aspecto importante que debe gestionar un administrador de redes es

minimizar la latencia de la red y lograr que la fluctuación sea constante; con ello, logrará que la red incremente su rendimiento y proporcione la máxima velocidad a las aplicaciones de los usuarios.

La causa más frecuente de latencia es que haya demasiadas aplicaciones que intenten utilizar una red a la vez, y por ello se deberá contar con aplicaciones de monitorización que comprueben el uso de los puntos de interconexión (tanto extremos como intermedios) de la conexión para ver qué aplicaciones provocan latencia. Entre las soluciones posibles encontramos las siguientes:

- Analizar los enrutadores y reducir el número de nodos que utiliza la red.
- Implementar *bonding* en los puntos conflictivos de los servidores (más canales de salida de red del mismo servidor).
- Verificar las conexiones por cables (*wired*) y desestimar o reducir aquellas que se producen por aire (wifi).
- Reemplazar, siempre que sea posible, las redes de cobre por redes de fibra óptica, aunque ello suponga un incremento en la inversión.
- Analizar las aplicaciones que están utilizando el ancho de banda (por ejemplo, NAS) recordando que las conexiones de red tienen un ancho de banda limitado y, si utiliza más, la latencia aumentará.
- Analizar los cortafuegos y si son necesarios entre el origen y destino de los paquetes, ya que estos procesan todo el tráfico de red entrante y saliente y, por lo tanto, incluyen un retraso de por sí; también hay que considerar la vida útil de los dispositivos de comunicación para encontrar defectos, ya que en ocasiones el envejecimiento puede causar incrementos de latencia.
- Analizar el rendimiento real con la SLA (*service level agreement*) de los proveedores externos para exigir cumplimiento del servicio contratado.

Si con todas estas contramedidas todavía se tiene un rendimiento no adecuado, será necesario realizar un rediseño y replanificación de la red para lograr los objetivos propuestos.

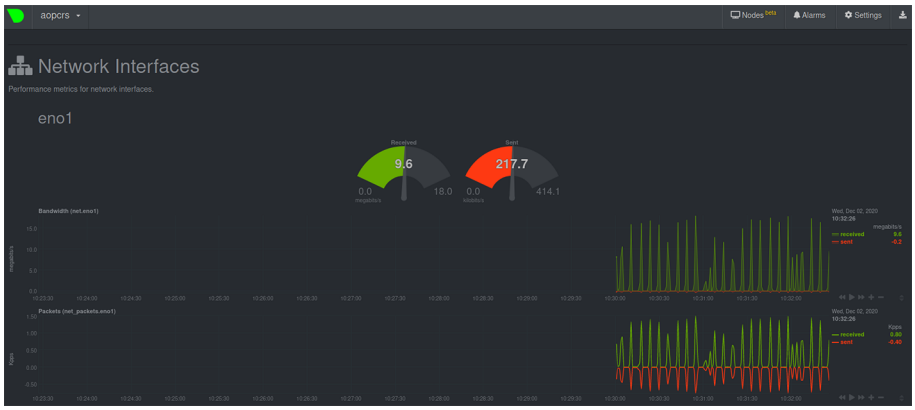
Como herramientas principales para la gestión del rendimiento de red, el administrador puede contar con las siguientes:

1) Iperf: es una herramienta muy simple pero que brinda mucha información sobre el rendimiento de una red y que se encuentra en casi todos los repositorios de Linux (`apt install iperf3`). Requiere que se ejecute en modo cliente en un nodo (`iperf3 -c IP_servidor`) y en modo servidor (`iperf3 -s`). Esta herramienta utiliza un puerto por defecto (5201), pero puede cambiarse.

```
root@mybi:~# iperf3 -c 20.20.21.14
Connecting to host 20.20.21.14, port 5201
[  4] local 20.20.21.20 port 48480 connected to 20.20.21.14 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.01   sec  92.2 MBytes   770 Mbits/sec   96    100 KBytes
[  4]   1.01-2.00   sec   109 MBytes   920 Mbits/sec  106    195 KBytes
[  4]   2.00-3.00   sec   114 MBytes   954 Mbits/sec  207   97.6 KBytes
[  4]   3.00-4.00   sec   108 MBytes   903 Mbits/sec  108    140 KBytes
[  4]   4.00-5.00   sec   109 MBytes   918 Mbits/sec  189    163 KBytes
[  4]   5.00-6.00   sec   104 MBytes   868 Mbits/sec  283    143 KBytes
[  4]   6.00-7.00   sec   107 MBytes   893 Mbits/sec  199    236 KBytes
[  4]   7.00-8.02   sec  98.9 MBytes   816 Mbits/sec  217    168 KBytes
[  4]   8.02-9.00   sec  96.6 MBytes   825 Mbits/sec   64    187 KBytes
[  4]   9.00-10.00  sec   110 MBytes   926 Mbits/sec  305    107 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.02 GBytes   879 Mbits/sec  1774             sender
[  4]   0.00-10.00  sec  1.02 GBytes   877 Mbits/sec                  receiver

iperf Done.
```

2) Netdata: es una herramienta muy interesante y visual, pero se debe considerar que permite ver el rendimiento de la red de la máquina analizada y no de la red en general; no obstante, con estas medidas el administrador puede intuir el estado de la red en general.



3) ping / hping3: como herramienta universal, siempre que se tenga acceso a un servidor remoto, donde la velocidad viene dada por el tamaño *ping* × 8 bits/byte / RTT (*round trip time*), por lo que si se envían 1.000 *pings* con tamaño de 5.000 bytes y se obtienen un RTT medio de 100 mseg, se puede decir que la velocidad será de 5.000 × 8 / 0,1 = 400.000 bps.

4) Ntopng: nueva versión del conocido programa Ntop, mide el tránsito de red a través de la librería *libpcap/PF_RING* y se encuentra en casi todos los repositorios de Linux (`apt install ntopng`). Utiliza un navegador para visualizar la interfaz poniendo como URL *localhost:3000*.

5) Netperf: en los sistemas Linux (`apt install netperf`) es otra herramienta habitual y se ejecuta, con: `netperf -H IP_nodo -l 10 -t TCP_RR -v.2`.

Enlace recomendado

Versión en línea de Ntopng para medir el rendimiento de una red teniendo en cuenta un servidor y diversos sitios en el mundo: https://www.dotcom-tools.com/website-speed-test.aspx

Enlace recomendado

En https://bit.ly/3JTuPS5 podéis encontrar un análisis de cómo medir la latencia y sus consideraciones utilizando herramientas habituales.

6) Herramienta de monitorización de tipo general, que incluye un apartado de rendimiento de redes disponible en los repositorios de las distribuciones de Linux más importantes, entre otras: Cacti, Ganglia, Icinga, Nagios, Collectd, Munin y Zabbix.

Ved también

Algunas herramientas de monitorización de tipo general serán analizadas en el módulo «Infraestructura como código (IaC). Monitorización y seguridad».

3.1. SDN (*software defined networks*)

Para este apartado, el alumnado deberá utilizar el libro ***SDN: Software Defined Network***[2] ***(T. Nadeau y K. Gray)*** del repositorio de la UOC (O'Reilly), en el que se verán las definiciones, los protocolos y los estándares emergentes para SDN (redes programables, controladas por software y definidas por software). Las SDN son independientes del proveedor y en el libro se utilizan varios casos de uso de SDN, incluida la programación y manipulación del ancho de banda, el tráfico de entrada y las acciones desencadenadas, así como algunos casos de uso interesantes en torno a *big data*, superposiciones de centros de datos y virtualización de funciones de red.

(2)Podéis encontrar el libro del repositorio UOC en el siguiente enlace: https://bit.ly/3uhXDyc

También se analiza el modelo *OpenFlow* y el control de red centralizado, así como las dos visiones del control (control central y control distribuido), incluida la generación de planos de datos y también la estructura y las capacidades de los controladores comerciales y de código abierto.

El alumno deberá comenzar por el capítulo 1, en el que se presenta una introducción al porqué de las SDN, y el capítulo 2, donde se ve la relación entre redes con control centralizado o distribuido y los planos de datos (*data planes*). A continuación, seguirá con el modelo *OpenFlow* en el capítulo 3 y con los controladores en el capítulo 4, y cómo se define su despliegue en el capítulo 5.

3.2. Pruebas de concepto

La forma más simple de experimentar con *OpenFlow* es utilizar el simulador Mininet emulator, que se encuentra disponible en casi todas las distribuciones de Linux (`apt install mininet`) y donde se puede utilizar un *RYU Controller* que proporciona componentes de software con una API «bien definida» para facilitar que los desarrolladores puedan probar la creación de nuevas aplicaciones de control y administración de redes.

Enlace recomendado

Podéis ampliar estos conceptos en la siguiente documentación: https://bit.ly/3zEj7Go

Otra opción es disponer de un *Openflow enabled Switch* o del software Open VSwitch (`apt install openvswitch-switch`), y se puede utilizar como controlador OpenDaylight (versión actual OpenDaylight Phosphorus).

Open vSwitch (también conocido como OVS) es una implementación de código abierto de un *switch* multicapa virtual distribuido con el objetivo de proveer un dispositivo de comunicación para entornos de virtualización de hardware con múltiples protocolos y estándares utilizados en redes. Puede ser desplegado como *switch* de red virtual entre servidores y distribuido de manera

transparente a varios servidores físicos, al tiempo que admite interfaces y protocolos de gestión estándar, como OpenFlow, NetFlow, sFlow, SPAN y RSPAN, entre otros.

Esta «distribución transparente» permite la creación de *switchs* entre servidores realizando una abstracción de la arquitectura de red similar al *VMware vNetwork vswitch* o Cisco Nexus 1000V. OVS puede funcionar como *switch* de red basado en software que se ejecuta dentro de un hipervisor y como *stack* de control para el hardware de conmutación dedicado. Por ello, se ha implementado en múltiples plataformas de virtualización, *chipsets* de conmutación y aceleradores de hardware de red. Puede trabajar con Linux KVM, Proxmox VE y VirtualBox, o a través de un puerto dedicado en Hyper-V y se ha integrado en diversas plataformas y sistemas de gestión de *clouds*, como OpenStack, openQRM, OpenNebula y oVirt.

Si se necesita un *virtual bridge* en Linux sin las complejidades de OVS, se puede utilizar un *Linux Ethernet bridge*, que, a través del paquete *bridge-utils*, permite configurar un *bridge* para conectar varios dispositivos Ethernet en forma transparente (los *hosts* conectados a un dispositivo Ethernet ven los *hosts* conectados a los otros dispositivos Ethernet directamente), lo cual resulta un método útil para compartir una red (por ejemplo, la conexión a internet) entre dos o más *hosts* (o MV) o utilizarlo como puente para proporcionar capacidad de red redundante (por ejemplo, la utilización de dos interfaces de redes para conectarse a dos enrutadores externos) o si se necesitan gestionar las conexiones de MV o contenedores que están dentro de un *host* (como ocurre con KVM, Docker, OpenNebula, Promox o VirtualBox).

Su instalación se puede realizar por medio de los repositorios de la distribución (`apt install bridge-utils`) y permite su configuración a través de una nueva interfaz virtual que normalmente encaminará a través de una interfaz física en forma transparente.

Para inicializar esta nueva interfaz, se debe ejecutar `brctl addbr br0`, donde el nombre puede ser cualquiera en este formato y luego se podrá observar con `ip address` que está en estado UP.

Como la configuración con **brctl** no es permanente, se puede realizar a través de su configuración en */etc/network/interfaces* (en este ejemplo, se asociará la interfaz externa *ens3 al bridge br0*).

```
...

 # se inicializa de forma manual para evitar conflictos p.ej. con el network manager
 iface ens3 inet manual

 # Bridge setup
 auto br0
```

```
iface br0 inet static
      address 10.10.10.198/24
      gateway 10.10.10.1
      bridged_ports ens3
      bridge_stp off       # disable Spanning Tree Protocol
      bridge_waitport 0    # no delay before a port becomes available
      bridge_fd 0          # no forwarding delay
```

Si se desea la conexión a internet a través de una conexión wifi, se deben realizar las configuraciones adicionales mencionadas en el módulo 2, ya que el protocolo wifi, por seguridad, normalmente no acepta conexiones desde la misma IP pero con diferentes MAC.

Enlaces recomendados

Se puede obtener información complementaria en Bridge Networking Connections (Debian) o Network Connections: Bridge (Ubuntu).

Bibliografía

Alapati, S. R. (2017). *Expert Hadoop Administration: Managing, Tuning, and Securing Spark, YARN, and HDFS*. EE. UU.: Addison-Wesley, Pearson Education.

Nadeau, T. D.; Gray, K. (2013). *SDN: Software Defined Networks*. Sebastopol: O'Reilly.

O'Reilly, J. (2016). *Network Storage*. Cambridge: Morgan Kaufmann. Dentro de los libros de O'Reilly en la biblioteca de la UOC.

Vacca, J. R. (2001). *The Essential Guide to Storage Area Networks*. Prentice Hall.

Enlaces (accedidos por última vez en diciembre de 2021)

Debian. «BridgeNetworkConnections». https://wiki.debian.org/BridgeNetworkConnections

Gluster. «GlusterFS». https://www.gluster.org/

Gluster. «Gluster Docs». https://docs.gluster.org/en/latest/

Mininet. «Mininet emulator». http://mininet.org/walkthrough/

Ubuntu. «Network Connection Bridge».https://help.ubuntu.com/community/NetworkConnectionBridge