Sistemes d'emmagatzematge distribuït i xarxes

PID_00286203

Remo Suppi Boldrito

Temps mínim de dedicació recomanat: 2 hores

Universitat Oberta de Catalunya

Remo Suppi Boldrito

Enginyer de Telecomunicacions. Doctor en Informàtica per la Universitat Autònoma de Barcelona (UAB). Professor del Departament d'Arquitectura de Computadors i Sistemes Operatius en la UAB.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats pel professor: Josep Jorba Esteve

Primera edició: febrer 2022

Av. Tibidabo, 39-43, 08035 Barcelona
Autoria: Remo Suppi Boldrito
Producció: FUOC

Índex

Objectius..... 5

1. La importància de l'emmagatzematge: des de xarxes locals al *big data* i *cloud*..... 7

2. Casos d'ús: RAID, GlusterFS, LVM..... 9
2.1. RAID 9
2.2. GlusterFS 12
2.3. LVM (*logical volume manager*) 14

3. Xarxes..... 17
3.1. SDN (*software defined networks*) 20
3.2. Proves de concepte 20

Bibliografia..... 23

Objectius

Els objectius principals d'aquest mòdul són els següents:

1. Conèixer els diferents aspectes de les xarxes d'emmagatzematge, l'arquitectura, els protocols i les eines.

2. Analitzar diferents casos d'ús per a l'emmagatzematge: redundància (RAID), distribuït (GlusterFS) i lògic (LVM).

3. Estudiar diferents aspectes de xarxes i el seu monitoratge, així com el concepte de SDN, la seva arquitectura i proves i la seva configuració a través de simuladors.

4. Analitzar diferents tipus de programari estàndard per a la creació i utilització de *switchs* virtuals.

En aquest mòdul, l'estudiant haurà de centrar la seva atenció en els següents conceptes fonamentals:

- Xarxes d'emmagatzematge.
- Arquitectures i configuració de sistemes d'emmagatzematge.
- Xarxes de dades i SDN.
- Monitoratge de xarxes i configuració de *switchs* virtuals.

Es recomana analitzar i realitzar proves d'instal·lació i configuració de les principals eines indicades.

1. La importància de l'emmagatzematge: des de xarxes locals al *big data* i *cloud*

En aquest mòdul, l'estudiant començarà llegint el capítol 1 de ***Network Storage***[1] **(James O'Reilly)** del repositori de la UOC (O'Reilly), on es presenta com està augmentant l'impacte de l'emmagatzematge en la tecnologia de la informació, a mesura que la velocitat i la taxa de transaccions s'ajusten a la llei de Moore. Amb aquest concepte es pretén justificar l'inici d'una «explosió» d'emmagatzematge, tant en capacitat com en la càrrega de la xarxa, que cada any s'incrementa en valors mai aconseguits anteriorment. En el capítol 2, l'estudiant consolidarà diferents conceptes i farà una revisió sobre l'emmagatzematge i la seva evolució des de la perspectiva de la xarxa; continuarà amb els capítols 3 i 4 per fer una revisió de l'estat actual de les xarxes d'emmagatzematge, els diferents protocols i esquemes de connexió, com es diferencien en casos d'ús i comportament operatiu, i una breu història de cadascun a mesura que ha anat evolucionant, juntament amb les seves fortaleses i febleses, entre altres aspectes.

[(1)]Podeu trobar el llibre del repositori UOC en el següent enllaç: https://bit.ly/3r9vgmg

En el capítol 8, l'estudiant s'aproximarà al *big data* i a com aquest afecta l'emmagatzematge per la seva falta d'estructura i la manera com s'utilitzen aquestes dades, que fan que siguin necessaris nous enfocaments. Finalment, en els capítols 9 i 10 l'aproximació de l'emmagatzematge serà cap al *high performance computing* (HPC) i cap al *cloud*. Si bé l'HPC és similar al *big data* en molts aspectes, és molt més sensible a la latència d'accés i afecta les xarxes, i d'aquí l'auge de xarxes com InfiniBand per arribar als dispositius d'emmagatzematge. En el *cloud*, el problema és totalment diferent, ja que moure dades a internet significa una altra dimensió per a les xarxes d'emmagatzematge, amb *gateways*, compressió i de duplicació, d'una banda, i protocols de transferència eficients, per l'altre, la qual cosa implica noves formes i mètodes per implementar-ho de manera eficient.

Es recomana a l'estudiant la lectura dels capítols 11 i 12, en els quals l'autor reflexiona sobre aspectes no menys importants en l'emmagatzematge, com són duplicació de la informació (*data integrity*) i privacitat de la informació (*data security*). S'ha de tenir en compte que la integritat de les dades és un problema important en l'emmagatzematge, ja que les unitats i els controladors fallen, i en els discos apareixen fallades que provoquen que les dades siguin il·legibles. Els esquemes de protecció de dades han evolucionat ràpidament, centrats en la replicació i la codificació d'esborrat, i han obtingut millores notables en la protecció de les dades emmagatzemades. Quant a seguretat, més enllà de la pèrdua o «segrest» de les dades, un aspecte crucial és la privacitat, que té per

objectiu protegir tant la dada emmagatzemada com el seu trànsit per mitjà de tècniques de xifratge que permetin que les dades no puguin ser visualitzades no només quan estan en el disc sinó també quan viatgen per la xarxa.

Per als estudiants que desitgin veure quines són les tecnologies actuals, tenint en compte que aquest llibre és del 2016, es recomana la lectura dels capítols 13 i 14, en els quals es mostren els principals avanços en discos sòlids (tecnologia NAND) i noves alternatives on l'emmagatzematge s'«allunya» del silici.

2. Casos d'ús: RAID, GlusterFS, LVM

2.1. RAID

Com es va analitzar en el capítol 11, la pèrdua d'informació per fallada en les unitats, controladors o discos és un punt important en l'emmagatzematge. La solució actual més aplicada és el RAID (*redundant array of independent disks*).

El RAID és una tecnologia d'emmagatzematge de dades que permet que les dades estiguin distribuïdes en diferents discos, la qual cosa possibilita la redundància de dades i millora el rendiment.

Les dades es distribueixen entre les unitats de diverses formes, denominades ***nivells RAID***, segons el nivell requerit de redundància i rendiment. Als diferents esquemes, o dissenys de distribució de dades, se'ls assigna la paraula *RAID* seguida d'un número, per exemple, RAID 0 o RAID 1 i cada esquema, o nivell de RAID, proporciona un equilibri diferent entre els objectius clau: confiabilitat, disponibilitat, rendiment i capacitat. Els nivells de RAID superiors a RAID 0 ofereixen protecció contra errors de lectura de sector irrecuperables, així com contra fallades d'unitats físiques completes.

El concepte *RAID* té més de tres dècades (Patterson, Gibson i Katz, 1988), però alguns dels seus nivells originals i evolucions continuen sent útils actualment. Com s'ha explicat anteriorment (en el capítol 11), molts nivells de RAID empren un esquema de protecció contra errors anomenat ***paritat***, basat en un XOR simple, a diferència de RAID 6, que utilitza dues paritats separades basades respectivament en la suma i la multiplicació. Amb l'avanç dels SSD, el RAID també pot proporcionar seguretat de dades amb unitats d'estat sòlid (SSD) sense que tots els discos siguin SSD, utilitzant discos magnètics (mecànics), però s'ha d'utilitzar un controlador apropiat que utilitzi el SSD per a totes les operacions de lectura (d'aquesta configuració, alguns fabricants en diuen *RAID híbrid*).

Originalment, hi havia cinc nivells estàndard de RAID, però han evolucionat i actualment en trobem moltes variacions, inclosos diversos nivells niats i molts nivells no estàndard (en la seva majoria, propietaris). Els nivells de RAID i els seus formats de dades associades estan estandarditzats per la Storage Networking Industry Association (SNIA) en l'estàndard *Common RAID Disk Drive Format* (DDF) com:

- **RAID 0**: consta de bandes (*striping*, divisió lògica de dades seqüencials), però sense duplicació ni paritat. La seva capacitat és la suma de les capacitats dels discos involucrats, però en distribuir les dades de cada arxiu entre totes les unitats del conjunt, no soluciona el problema de les fallades (i la taxa mitjana de fallades del volum augmenta, ja que hi ha més unitats connectades). En funció de com s'hagin distribuït els continguts, el rendiment de les operacions de lectura i escriptura en qualsevol arxiu es pot multiplicar pel nombre d'unitats.

- **RAID 1** (*mirror*): consisteix en duplicació de dades i sense paritat, on qualsevol unitat del conjunt pot atendre qualsevol sol·licitud de lectura, la qual cosa millora el rendiment. L'espai d'emmagatzematge és la meitat del total (amb discos sencers).

- **RAID 2, 3 i 4**: inclouen paritat i millores, però actualment no es fan servir perquè la funcionalitat i protecció és millor en RAID 5/6.

- **RAID 5**: consta de bandes en el nivell de bloc amb paritat distribuïda, la qual cosa requereix que totes les unitats menys una estiguin presents per funcionar. En cas de fallada d'una sola unitat, les lectures posteriors es poden calcular a partir de la paritat distribuïda, de manera que no es perdin dades i per això RAID 5 requereix almenys tres discos.

- **RAID 6**: consta de bandes en el nivell de bloc amb paritat distribuïda doble, la qual cosa proporciona tolerància a fallades fins de dues unitats i provoca que els grups RAID més grans siguin més pràctics, especialment per als sistemes d'alta disponibilitat, ja que les unitats de gran capacitat triguen més a restaurar-se. RAID 6 requereix un mínim de quatre discos.

- **Nivells RAID niats**: molts controladors d'emmagatzematge permeten niar nivells de RAID, on l'*array* final es coneix com *array* superior (per exemple, quan l'*array* superior és RAID 0 com en RAID 1 + 0 i RAID 5 + 0, encara que normalment s'elimina el + i s'indica amb RAID 10 i RAID 50, respectivament).

Per dissenyar un *array*, es recomana utilitzar un calculador de RAID dels molts que existeixen a internet.

Exemple: calculadors de RAID

Raid-calculator i Giga-Calculator.

Si bé existeixen controladores RAID maquinari (amb el consegüent increment de prestacions), es pot realitzar un RAID per programari a través de qualsevol distribució de Linux. Per crear un RAID programari, s'utilitza la ordre `mdadm` (el nom deriva de md, *múltiple device* + *admin*, i reemplaça l'anterior `mdctl`).

mdadm s'instal·la des del repositori (`apt install mdadm`) i com a primer pas s'han de preparar els discos (poden ser físics o virtuals) i crear una partició en cadascun d'aquests del **tipus 0xfd**. Per crear un RAID 5, per exemple, anomenat /dev/md0 amb tres discos (sdb1, sdc1, sdd1), fem:

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

Es pot modificar el *chunk size* (per exemple, agregant `--chunk=16384`), on valors grans del *chunk* són millor per a arxius grans (el valor per defecte és de 512). Recordeu també que s'ha de crear el *filesystem* (`mkfs.ext4 /dev/md0`) abans de muntar-lo.

Per muntar un *array* que no està en l'arxiu de configuració s'ha de fer el següent:

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

Per crear un arxiu de configuració i que l'*array* s'iniciï durant l'arrencada de SO fem:

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

Per agregar un nou disc i fer créixer l'*array* (una vegada afegit el disc, s'haurà d'executar `resize2fs /dev/md0` per a fer créixer el *filesystem*). Fem:

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

(aquest disc quedarà com *spare*)

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

(amb això es veurà que el disc passa de *spare* a actiu)

Per marcar un disc amb errors i treure'l de l'*array* (necessari quan es vol canviar un disc), fem:

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


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

Per parar l'*array* (recordeu que s'ha de desmuntar el *filesystem* primer), fem:

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

Per iniciar l'*array* (després es podrà muntar), fem:

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

Per obtenir informació de l'*array*, fem:

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

I per monitorar l'*array* i que enviï un correu electrònic (s'ha d'anar amb compte de no caure en un enviament reiterat –*spam*– de correus), fem:

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

2.2. GlusterFS

És un sistema d'arxius d'emmagatzematge distribuït i s'utilitza en *cloud*, serveis *streaming* o en xarxes de contingut (CDN).

GlusterFS va ser desenvolupat originalment per Gluster, Inc. i després per Red Hat (adquisició el 2011 i el 2012 es va integrar a Red Hat Storage Server, que, quan el 2014 RH va comprar Inktank Storage –desenvolupador de Ceph–, li va canviar el nom per Red Hat Gluster Storage).

GlusterFS agrupa diversos servidors d'emmagatzematge a través d'interconnexió Ethernet o InfiniBand en un gran sistema d'arxius de xarxa. Es tracta de programari lliure amb llicència GPL/LGPL, el seu disseny està basat en un *stackable user space* i funciona en una modalitat de client i servidor. Normalment, els servidors proveeixen «els maons» (*bricks*) d'emmagatzematge a partir dels quals es creen volums virtuals (compostos de diversos servidors), mentre que el client es connecta als servidors amb un protocol personalitzat a través de TCP/IP, InfiniBand o Sockets Direct Protocol, per accedir als volums virtuals creats prèviament a partir dels *bricks*.

El client pot muntar el volum mitjançant un protocol nadiu de GlusterFS, utilitzant el mecanisme FUSE o per NFSv3. GlusterFS proporciona fiabilitat i disponibilitat de dades per mitjà de diversos tipus de rèplica: volums replicats i georeplicació, on els volums replicats garanteixen que hi hagi almenys una còpia de cada arxiu en diferents «maons», per la qual cosa si un falla, les dades continuaran emmagatzemades i accessibles. La georeplicació proporciona un model de rèplica *master/worker*, en el qual es copien els volums en ubicacions geogràficament diferents i això es fa de manera asíncrona, la qual cosa resulta útil per a la disponibilitat en cas que un centre de dades quedi fora de servei.

En la majoria de les distribucions de Linux GlusterFS es troba en el repositori de la distribució i s'ha d'instal·lar el servidor, però el client, si es vol, amb l'ordre `apt install glusterfs-client glusterfs-server` pot també muntar el volum distribuït.

Des de qualsevol dels servidors que disposem es necessita descobrir els servidors restants perquè iniciïn la seva sincronització, a través de la següent ordre `gluster peer probe IP_servidor_remot`. I es pot comprovar l'estat del sistema amb `gluster peer status`.

La creació d'un volum (en aquest exemple, anomenat *myvol*) passa per tenir definits i muntats els discos (o particions) que conformaran els *bricks* en cada servidor i després executar el següent:

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

A continuació, s'iniciarà el volum amb `gluster volume start myvol` i per poder obtenir informació del seu estat es pot executar la ordre `gluster volume info`.

Per crear dues rèpliques de cada arxiu, s'agrega el paràmetre `replica 2`:

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

Tot seguit, es podrà crear un punt de muntatge mitjançant, per exemple, l'ordre `mkdir -p /mnt/gluster` i muntar-ho, verificant el seu espai amb `dh -h`:

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

A més, si se'n vol automatitzar el muntatge a l'arrencada, es pot incloure en */etc/fstab*:

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

Una manera útil d'analitzar la seva funcionalitat és generant una gran quantitat d'arxius i mirant la seva distribució en els *bricks*:

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

A continuació, es pot verificar en cada *brick* com s'han distribuït els arxius (que es veuran tots junts en el volum */mnt/gluster*).

Altres ordres útils són les següents:

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

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

2.3. LVM (*logical volume manager*)

És un entorn que proporciona una gestió de volums per al nucli de Linux que separa la gestió física del disc de la lògica.

El seu creador (H. Mauelshagen) va escriure el codi original el 1998 basat en les idees del gestor de volums d'HP-UX. LVM s'utilitza per als següents propòsits:

- Creació de volums lògics individuals de diversos volums físics o discos sencers (similars a RAID 0), permetent canviar la grandària del volum de manera dinàmica.

- Gestió de granges de discos durs que permeten afegir i substituir discos sense temps d'inactivitat ni interrupció del servei, en combinació amb l'intercanvi en calent.

- En lloc d'haver d'estimar la grandària que pot necessitar una partició, LVM permet canviar la grandària dels sistemes d'arxius segons sigui necessari.

- Permet realitzar còpies de seguretat consistents prenent instantànies dels volums lògics.

- Permet xifrar diverses particions físiques amb una sola contrasenya.

- Organitza els discos primer en particions físiques (*physical volumes*, PV), grups de volums (*volume groups*, VG) sobre els PV i volums lògics (*logical volumes*, LV) sobre cada VG.

Per crear un LVM (alguns SO disposen d'un assistent, però es pot fer en CLI), s'han de fer els següents passos:

1) Definir i inicialitzar els discos creant una partició de tipus **8e** (amb *fdisk*).

2) Definir i inicialitzar cada *physical volumes* (PV):

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

3) Definir el *volume groups* agrupant els PV:

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

4) Inicialitzar els *logical volumes* sobre cada VG:

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

5) Es pot expandir (o reduir) un volum amb `pvcreate /dev/nou_dev` i després executant:

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

6) Per reduir un volum, fem:

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

Per instal·lar-ho dels repositoris, `apt install lvm2`. Per instal·lar la interfície gràfica, `apt install system-config-lvm`.

Per crear, per exemple, un volum lògic (*mylogvol*), primer s'ha de crear un grup (*myvol*), amb les particions */dev/sdd1* i */dev/sde1*, una grandària de 10 Gbytes i muntar-lo en */mnt/lvm*. S'haurà d'executar com a *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
```

Amb l'ordre `lvremove /dev/myvol/mylogvol` podem eliminar el LVM creat.

Altres ordres útils són les següents:

- `lvchange`: canviar els atributs d'un *logical volume*.
- `lvconvert`: convertir a *logical volume* a *mirror* o *snapshot*.
- `lvcreate`: crear un *logical volume* en un existent *volume group*.
- `lvdisplay`: mostrar els atributs d'un *logical volume*.
- `lvextend`: estendre la grandària d'un *logical volume*.
- `lvreduce`: reduir la grandària d'un *logical volume*.

- `lvremove`: eliminar un *logical volume*.
- `lvrename`: canviar de nom un *logical volume*.
- `lvresize`: redimensionar un *logical volume*.
- `lvs`: obtenir informació dels *logical volumes*.

3. Xarxes

En una xarxa d'una infraestructura per al processament de dades, és important conèixer els nivells de rendiment (*throughput*) i la seva amplada de banda, ja que d'aquests paràmetres dependrà l'eficiència del sistema. El **rendiment** indica quantes dades s'han transferit des d'una font en un moment donat, mentre que l'**amplada de banda** indica quantes dades es podrien transferir teòricament des d'una font a cada moment. L'administrador/gestor de la infraestructura haurà de saber en cada moment quin és el rendiment i l'amplada de banda tant intern com dels proveïdors externs (si les aplicacions estan externament interconnectades, per exemple, al *cloud*) per actuar en conseqüència.

Per això, s'ha de diferenciar clarament quin és el rendiment i l'amplada de banda, quina és la diferència hi ha entre aquests i per què importa. La resposta breu és la **velocitat**.

> La velocitat és un dels paràmetres més importants que s'utilitzen per mesurar el rendiment de la xarxa. La velocitat amb la qual viatgen els paquets des de la font fins a la destinació és la que determina la quantitat d'informació que es pot enviar en un interval considerat.

Una velocitat de xarxa lenta és igual a baix rendiment de les aplicacions de processament de dades i això equival a aplicacions condicionades en la seva execució per aquest paràmetre que actua com a coll d'ampolla.

El rendiment (*throughout*) de xarxa mesura quants paquets arriben a les seves destinacions amb èxit i generalment es mesura en bits per segon. Quan s'utilitzen aplicacions en *big data* es vol que els paquets de dades «viatgin» a la màxima velocitat, i quan els paquets es perden i han de ser reexpedits això suposa un rendiment de les aplicacions deficient o lent.

Les magnituds principals que poden donar informació sobre els problemes d'una xarxa són la pèrdua de paquets, la latència i la fluctuació, ja que estan relacionades amb la velocitat de transmissió. La **latència** (*latency*) és la quantitat de temps que triga un paquet des de l'origen fins a la destinació, mentre que la **fluctuació** (*jitter*) fa referència a la diferència de retard de cada paquet. La minimització d'aquests factors és fonamental per augmentar la velocitat i incrementar les prestacions de les aplicacions distribuïdes de *big data*. Un aspecte important que ha de gestionar un administrador de xarxes és minimit-

zar la latència de la xarxa i aconseguir que la fluctuació sigui constant; amb això, aconseguirà que la xarxa incrementi el seu rendiment i proporcioni la màxima velocitat a les aplicacions dels usuaris.

La causa més freqüent de latència és que hi hagi massa aplicacions que intentin utilitzar una xarxa alhora, i per això s'haurà de comptar amb aplicacions de monitoratge que comprovin l'ús dels punts d'interconnexió (tant extrems com intermedis) de la connexió per veure quines aplicacions provoquen latència. Entre les solucions possibles trobem les següents:

- Analitzar els encaminadors i reduir el nombre de nodes que utilitza la xarxa.

- Implementar *bonding* en els punts conflictius dels servidors (més canals de sortida de xarxa del mateix servidor).

- Verificar les connexions per cables (*wired*) i desestimar o reduir aquelles que es produeixen per aire (wifi).

- Reemplaçar, sempre que sigui possible, les xarxes de coure per xarxes de fibra òptica, encara que això suposi un increment en la inversió.

- Analitzar les aplicacions que estan utilitzant l'amplada de banda (per exemple, NAS) sense oblidar que les connexions de xarxa tenen una amplada de banda limitada i, si se n'utilitza més, la latència augmentarà.

- Analitzar els tallafocs i si són necessaris entre l'origen i destinació dels paquets, ja que aquests processen tot el trànsit de xarxa entrant i sortint i, per tant, inclouen un retard per si sols; també cal considerar la vida útil dels dispositius de comunicació per trobar defectes, ja que a vegades l'envelliment pot causar increments de latència.

- Analitzar el rendiment real amb la SLA (*service level agreement*) dels proveïdors externs per exigir compliment del servei contractat.

Si amb totes aquestes contramesures el rendiment encara no és l'adequat, serà necessari fer un redisseny i una replanificació de la xarxa per aconseguir els objectius proposats.

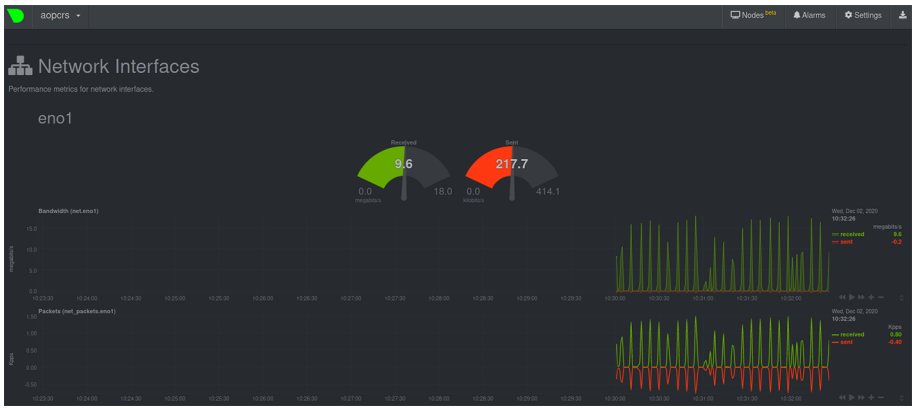
Com a eines principals per a la gestió del rendiment de xarxa, l'administrador pot comptar amb les següents:

1) Iperf: és una eina molt simple, però que brinda molta informació sobre el rendiment d'una xarxa i que es troba en gairebé tots els repositoris de Linux (`apt install iperf3`). Requereix que s'executi en manera client en un node (`iperf3 -c IP_servidor`) i en manera servidora (`iperf3 -s`). Aquesta eina utilitza un port per defecte (5201), però es pot canviar.

```
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: és una eina molt interessant i visual, però s'ha de considerar que permet veure el rendiment de la xarxa de la màquina analitzada i no de la xarxa en general; no obstant això, amb aquestes mesures l'administrador pot intuir l'estat de la xarxa en general.



3) ping / hping3: com a eina universal, sempre que es tingui accés a un servidor remot, on la velocitat ve donada per la grandària *ping* × 8 bits/byte / RTT (*round trip time*), per la qual cosa si s'envien 1.000 *pings* amb grandària de 5.000 bytes i s'obtenen un RTT mitjà de 100 mseg, es pot dir que la velocitat serà de 5.000 × 8 / 0,1 = 400.000 bps.

4) Ntopng: nova versió del conegut programa Ntop, mesura el trànsit de xarxa a través de la llibreria *libpcap/PF_RING* i es troba en gairebé tots els repositoris de Linux (`apt install ntopng`). Utilitza un navegador per visualitzar la interfície posant com URL *localhost:3000*.

Enllaç recomanat

Versió en línia de Ntopng per mesurar el rendiment d'una xarxa tenint en compte un servidor i diversos llocs del món: https://www.dotcom-tools.com/website-speed-test.aspx

5) Netperf: en els sistemes Linux (`apt install netperf`) és una altra eina habitual i s'executa, com: `netperf -H IP_node -l 10 -t TCP_RR -v.2`.

Enllaç recomanat

A https://bit.ly/3JTuPS5 hi trobareu una anàlisi de com mesurar la latència i les seves consideracions amb eines habituals.

6) Eina de monitoratge de tipus general, que inclou un apartat de rendiment de xarxes disponible en els repositoris de les distribucions de Linux més importants, entre d'altres: Cacti, Ganglia, Icinga, Nagios, Collectd, Munin i Zabbix.

Vegeu també

Algunes eines de monitoratge de tipus general s'analitzaran en el mòdul «Infraestructura com a codi (IaC). Monitoratge i seguretat».

3.1. SDN (*software defined networks*)

Per a aquest apartat, l'alumnat haurà d'utilitzar el llibre ***SDN: Software Defined Network***[2] ***(T. Nadeau i K. Gray)*** del repositori de la UOC (O'Reilly), en el qual es veuran les definicions, els protocols i els estàndards emergents per a SDN (xarxes programables, controlades per programari i definides per programari). Les SDN són independents del proveïdor i en el llibre s'utilitzen diversos casos d'ús de SDN, inclosa la programació i manipulació de l'amplada de banda, el trànsit d'entrada i les accions desencadenades, així com alguns casos d'ús interessants entorn de *big data*, superposicions de centres de dades i virtualització de funcions de xarxa.

(2)Podeu trobar el llibre del repositori UOC en el següent enllaç: https://bit.ly/3uhxdyc

També s'analitza el model *OpenFlow* i el control de xarxa centralitzat, així com les dues visions del control (control central i control distribuït), inclosa la generació de plans de dades i també l'estructura i les capacitats dels controladors comercials i de codi obert.

L'alumne haurà de començar pel capítol 1, en el qual es presenta una introducció al perquè de les SDN, i el capítol 2, on es veu la relació entre xarxes amb control centralitzat o distribuït i els plans de dades (*data plans*). A continuació, seguirà amb el model *OpenFlow* en el capítol 3 i amb els controladors en el capítol 4, i com es defineix el seu desplegament en el capítol 5.

3.2. Proves de concepte

La forma més simple d'experimentar amb *OpenFlow* és utilitzar el simulador Mininet emulator, que es troba disponible en gairebé totes les distribucions de Linux (`apt install mininet`) i on es pot utilitzar un *RYU Controller* que proporciona components de programari amb una API «ben definida» per facilitar que els desenvolupadors puguin provar la creació de noves aplicacions de control i administració de xarxes.

Enllaç recomanat

Podeu ampliar aquests conceptes a la documentació següent: https://bit.ly/3zEj7Go

Una altra opció és disposar d'un *Openflow enabled Switch* o del programari Open VSwitch (`apt install openvswitch-switch`), i es pot utilitzar com a controlador OpenDaylight (versió actual OpenDaylight Phosphorus).

Open vSwitch (també conegut com OVS) és una implementació de codi obert d'un *switch* multicapa virtual distribuït amb l'objectiu de proveir un dispositiu de comunicació per a entorns de virtualització de maquinari amb múltiples protocols i estàndards utilitzats en xarxes. Pot ser desplegat com *switch* de xarxa virtual entre servidors i distribuït de manera transparent a diversos servidors físics, al mateix temps que admet interfícies i protocols de gestió estàndard, com OpenFlow, NetFlow, sFlow, SPAN i RSPAN, entre d'altres.

Aquesta «distribució transparent» permet la creació de *switchs* entre servidors realitzant una abstracció de l'arquitectura de xarxa similar al *VMware vNetwork vswitch* o Cisco Nexus 1000V. OVS pot funcionar com *switch* de xarxa basat en programari que s'executa dins d'un hipervisor i com *stack* de control per al maquinari de commutació dedicat. Per això, s'ha implementat en múltiples plataformes de virtualització, *chipsets* de commutació i acceleradors de maquinari de xarxa. Pot treballar amb Linux KVM, Proxmox VEU i VirtualBox, o a través d'un port dedicat en Hyper-V i s'ha integrat en diverses plataformes i sistemes de gestió de *clouds*, com OpenStack, openQRM, OpenNebula i oVirt.

Si es necessita un *virtual bridge* en Linux sense les complexitats d'OVS, es pot utilitzar un *Linux Ethernet bridge*, que, a través del paquet *bridge-utils*, permet configurar un *bridge* per connectar diversos dispositius Ethernet en forma transparent (els *hosts* connectats a un dispositiu Ethernet veuen els *hosts* connectats als altres dispositius Ethernet directament), la qual cosa resulta un mètode útil per compartir una xarxa (per exemple, la connexió a internet) entre dos o més *hosts* (o MV) o utilitzar-lo com a pont per proporcionar capacitat de xarxa redundant (per exemple, la utilització de dues interfícies de xarxes per connectar-se a dos encaminadors externs) o si es necessiten gestionar les connexions de MV o contenidors que estan dins d'un *host* (com ocorre amb KVM, Docker, OpenNebula, Promox o VirtualBox).

La seva instal·lació es pot realitzar per mitjà dels repositoris de la distribució (`apt install bridge-utils`) i permet la seva configuració a través d'una nova interfície virtual que normalment encaminarà a través d'una interfície física en forma transparent.

Per inicialitzar aquesta nova interfície, s'ha d'executar `brctl addbr br0`, on el nom pot ser qualsevol en aquest format i després es podrà observar amb `ip address` que està en estat UP.

Com que la configuració amb **brctl** no és permanent, es pot realitzar a través de la seva configuració en */etc/network/interfícies* (en aquest exemple, s'associarà la interfície externa *ens3 al bridge br0*).

```
...

 # s'inicialitza de manera manual per evitar conflictes, p. ex. amb 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 es desitja la connexió a internet a través d'una connexió wifi, s'han de realitzar les configuracions addicionals esmentades en el mòdul 2, ja que el protocol wifi, per seguretat, normalment no accepta connexions des de la mateixa IP però amb diferents MAC.

Enllaços recomanats

Es pot obtenir informació complementària a Bridge Networking Connections (Debian) o Network Connections: Bridge (Ubuntu).

Bibliografia

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

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

O'Reilly, J. (2016). *Network Storage*. Cambridge: Morgan Kaufmann. Dins dels llibres d'O'Reilly a la biblioteca de la UOC.

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

Enllaços (als quals s'ha accedit per última vegada el desembre 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