Comment tester la vitesse d'un réseau Ethernet

Alexandre BLANCKE

Il est souvent nécessaire de vérifier la configarution d'un réseau Ethernet. Pour ce faire, le meilleur moyen consiste à vérifier que la vitesse de transfert d'un fichier peut pratiquement atteindre la vitesse nominale du réseau, soit :

- 1 000 Kbytes/s, pour un réseau à 10 Mb,
- 10 000 Kbytes/s, pour un réseau à 100 Mb.

Procédure

Pour tester la vitesse réelle d'un réseau Ethernet, il faut tout d'abord s'assurer que la commande utilisée va bien tester la vitesse du réseau et non pas, par effet de bord, la vitesse du CPU ou celle d'un disque.

La commande à privilégier pour faire un test de vitesse est "**ftp**". En effet, elle est simple d'utilisation et disponible sur la plupart des systèmes.

Pour éviter la consommation de ressouces CPU due à la translation de caractères, il faut toujours se mettre en "mode binaire" dans ftp.

① Vérifier que le routage est fait correctement

Passer la commande :

```
# ping -R sp2n9g

PING sp2n9g.astaix.france.ibm.com (9.101.4.241): 56 data bytes
64 bytes from 9.101.4.241: icmp_seq=0 ttl=255 time=0 ms
RR:     sp2n9g.astaix.france.ibm.com (9.101.4.241)
       testh70g.astaix.france.ibm.com (9.101.4.242)
64 bytes from 9.101.4.241: icmp_seq=1 ttl=255 time=0 ms (same route)
```

② Lancer "ftp"

```
# ftp sp2n9g

Connected to sp2n9g.astaix.france.ibm.com.
220 sp2n9c FTP server (Version 4.1 Mon Jul 26 19:58:48 CDT 1999) ready.
Name (sp2n9g:root): root
331 Password required for root.
Password:
230 User root logged in.
ftp> bin
200 Type set to I.
```

③ Tester la vitesse du réseau

Première méthode

La première méthode consiste à envoyer un fichier, créé à partir du *pseudo-device* "**/dev/zero**" de la machine source, vers le *device* "**/dev/null**" de la machine de destination.
Ceci permet d'envoyer un fichier aussi grand que nécessaire en jouant avec le "**bs**" (*block_size*) et le "**count**" (nombre d'enregistrements de la taille du *block_size*) :

```
ftp> put «|dd if=/dev/zero bs=4096k count=200» /dev/null

200 PORT command successful.
150 Opening data connection for /dev/null.
200+0 records in
200+0 records out
226 Transfer complete.
838860800 bytes sent in 30.9 seconds (2.651e+04 Kbytes/s)
local: |dd if=/dev/zero bs=4096k count=200 remote: /dev/null
```

Ici, le résultat est relativement bon mais le débit est bloqué par la consommation de ressouces CPU, pour créer les "**0**", par la commande "**dd**" et le *pseudo device* "**/dev/zero**".

On peut le vérifier en lançant un "**vmstat**" pendant que le ftp tourne :

 # vmstat 5

```
kthr  memory              page              faults           cpu
----  ----------  ------------------------ ------------ -----------
r  b   avm    fre  re  pi  po  fr  sr  cy  in   sy    cs   us sy  id  wa
0  1  12748  3485  0   0   0   0   0   0   109  28    31   0   0  99  0
0  1  12748  3485  0   0   0   0   0   0   107  875   25   0   1  99  0
1  1  15839   382  0   0   0   0   0   0   1351 1106  1056 0  64  36  0
1  1  15839   382  0   0   0   0   0   0   2039 1712  1684 1  99   1  0
2  1  15839   382  0   0   0   0   0   0   2066 1731  1703 1  99   0  0
2  1  15839   380  0   0   0   0   0   0   2042 1710  1684 1  99   1  0
2  1  15839   380  0   0   0   0   0   0   2071 1734  1707 0  99   0  0
1  1  15839   380  0   0   0   0   0   0   2031 1762  1732 1  98   1  0
0  1  12748  3492  0   0   0   1   1   0   1112 910   888  0  50  50  0
```

⇐ 99% CPU système

Deuxième méthode

Dans le type de configuration ci-dessus, on s'aperçoit que le CPU n'est pas assez puissant pour alimenter le réseau avec les commandes système classiques...
On utilisera alors une autre méthode faisant appel à un fichier temporaire pour faire le test.

Pour créer ce fichier, il faut d'abord déterminer la taille mémoire que l'on peut allouer au *mapping* des fichiers.

En effet, si l'on veut tester la vitesse pure du réseau, il faut que le fichier à transférer puisse être lu directement en mémoire et non sur disque. Pour cela, on utilisera les commandes suivantes :

 # lsattr -El sys0|grep realm

```
realmem    131072     Amount of usable physical memory in Kbytes      False
```

 # /usr/samples/kernel/vmtune

```
vmtune:  current values:

  -p       -P        -r          -R        -f       -F        -N        -W
minperm  maxperm  minpgahead maxpgahead  minfree  maxfree  pd_npages maxrandwrt
 6344     25376       2           8        120      128      524288      0

  -M      -w      -k      -c       -b         -B         -u        -l       -d
maxpin npswarn npskill numclust numfsbufs hd_pbuf_cnt lvm_bufcnt lrubucket defps
26196    2048     512      1       93         64          9       131072     1

       -s              -n         -S           -h
sync_release_ilock  nokillroot  v_pinshm  strict_maxperm
       0                0          0           0

number of valid memory pages = 32744      maxperm=77.5% of real memory
maximum pinable=80.0% of real memory      minperm=19.4% of real memory
number of file memory pages = 20672       numperm=63.1% of real memory
```

Nous avons donc ici 132 Mb de mémoire et le **maxperm** (taille réservée au *mapping* des fichiers) en occupe 77.5%.

Pour être certains que notre fichier soit bien en mémoire, nous allons faire nos calculs sur 60% (environ 80 Mb) de la mémoire réelle.

- Créer le fichier :

```
# dd if=/dev/zero bs=1024k count=80 of=/home/testfile

80+0 records in
80+0 records out
```

- Puis transférer le fichier par **ftp** en mode binaire :

```
# ftp sp2n9g

Connected to sp2n9g.astaix.france.ibm.com.
220 sp2n9c FTP server (Version 4.1 Mon Jul 26 19:58:48 CDT 1999) ready.
Name (sp2n9g:root): root
331 Password required for root.
Password:
230 User root logged in.
ftp> bin
200 Type set to I.
ftp> put /home/testfile /dev/null
200 PORT command successful.
150 Opening data connection for /dev/null.
226 Transfer complete.
83886080 bytes sent in 8.564 seconds (9566 Kbytes/s) ⇐ vitesse du disque
local: /home/testfile remote: /dev/null
```

A partir du deuxième transfert, la commande "**vmstat**" ne doit pratiquement plus montrer de *waits* d'I/O, puisque le fichier réside en mémoire, et le débit du "ftp" doit correspondre à la vitesse maximum sur la connexion.

```
ftp> put /home/testfile /dev/null

200 PORT command successful.
150 Opening data connection for /dev/null.
226 Transfer complete.
83886080 bytes sent in 2.581 seconds (3.174e+04 Kbytes/s) ⇐ vitesse du réseau
local: /home/testfile remote: /dev/null
```

Problèmes fréquents

On rencontre régulièrement les problèmes suivants :

- Le débit du réseau est extrêmement faible (par exemple : 5 Kb/s pour un réseau à 10 ou à 100 Kb/s).

- Le débit est beaucoup plus rapide dans un sens de transfert que dans l'autre.

Outre les **câbles de mauvaise qualité** sur lesquels un ou plusieurs fils seraient coupés, il y a toutes les chances pour que ces problèmes soient dûs :

- à la gestion de l'**auto-négociation**,

ou

- à une **configuration** différente des **vitesses** ou des **duplex** des divers *ports* des unités de *switch* ou des *adapters* des systèmes.

Full duplex :
En *full duplex* on utilise 4 fils : 2 en émission et 2 en réception.

Half duplex :
En *Half duplex* on n'utilise que 2 fils qui servent à la fois en émission et en réception, ce qui explique que l'on puisse avoir des "collisions".

Si donc une carte du RS/6000 est configurée en "*100-Full duplex*" alors que le *port* de l'unité de *switch* a négocié ou est configuré en "*100-Half duplex*", le RS/6000 s'attend à recevoir les données sur ses 2 fils de réception alors que le *switch* les lui envoie sur les 2 d'émission du RS/6000. Il y a donc une énorme perte de temps due à la récupération des données (*recovery*).

La solution la plus efficace est de configurer tous les composants (cartes et *ports* des unités de *switch*) en "*100-Full duplex*". ■