Réseaux Datacenters/HPC
TP - Administration des réseaux Infiniband

CEA/ENSIIE – 2017-2018

15 mai 2018

Résumé

Ce TP a pour objectif de comprendre et de manipuler les éléments des réseaux Infiniband. Dans un premier temps, il sera vu les commandes nécessaires à l'administration des réseaux IB, puis dans un second temps, nous verrons comment réaliser du debug sur ce type très spécifique de réseau.

Table des matières

1 Rendu

Pour nous assurer que les notions du TP ont bien été acquises, un rendu devra être fait à la fin de ce TP.

1. La qualité rédactionnelle : forme et fond
2. La pertinence des réponses

Les rapports peuvent être faits à maximum 2. Tout nom manquant sur le rendu aura automatiquement 0. Si vous le faites à 2, il n'y a qu'un seul rendu attendu.

Vous devez copier non seulement la commande mais aussi son résultat. Si la commande génère beaucoup de lignes, copier uniquement les éléments pertinents.

1.1 Le rendu

Le rendu final sera une archive (zip) contenant les éléments suivants :

1. Le rapport
2. Les scripts écrits
3. Les graphiques (on limitera à maximum 2MB par graphique, si vous en avez des plus lourds, ne les mettez pas dans l'archive)

Pour rendre, vous devez envoyer par mail l'archive au format NOM1_NOM2.zip à l'adresse damien.gros@cea.fr.

Date finale de rendu : mardi 22 mai 2018 à 19h.

Tout retard dans le rendu entrainera la note de 0 aux personnes du groupe.

2 Mise en place de l'environnement

Pour réaliser ce TP, nous allons utiliser un machine virtuelle qui a été spécialement préparée. Elle est sur le calculateur dans le répertoire `/home/shared/grosd/tp-ib`. Elle a déjà été déployé dans vos home directory.

Cette machine virtuelle s'utilise avec l'hyperviseur PCOCC. Comme vous l'avez vu dans un précédent TP, pour utiliser les machines au travers de PCOCC, il est nécessaire de créer une entrée dans le fichier `~/.pcocc/templates.yaml`. Pensez à adapter le chemin vers l'image.

```
mycentos7.4-tp-ib:
    resource-set: ens-cluster
    image: /home/guestXX/.pcocc/images/test-centos74-tp-ib
    user-data: ~/.pcocc/cloud-user-data/test-centos74-tp-ib
```

Listing 1 – Extrait du contenu du fichier templates.yaml

La seconde manipulation est d'ajouter une clé SSH dans le fichier `~/.pcocc/cloud-init/centos.ci`. Pensez à modifier le nom de connexion.

```
#cloud-config

user: guestXX
ssh_authorized_keys:
 - ssh-rsa cle_ssh.pub
```

Listing 2 – Extrait du fichier centos.ci

Une fois ces deux fichiers renseignés et que vous avez copié la machine virtuelle dans votre home directory, vous pouvez la lancer avec la commande suivante.

```
\$ pcocc alloc -t 4:00:00 -c 4  mycentos74-tp-ib:1
```

Listing 3 – Lancement de la machine virtuelle

Petit rappel sur l'utilisation de la commande :
- alloc : permet de lancer une machine virtuelle en mode interactif
- -t : permet de définir la durée pendant laquelle la machine sera exécutée
- -c : définit le nombre de processeur alloué à la machine
- -p : permet de définir sur quelle partition s'exécute la machine virtuelle
- mycentos74-tp-ib :1 : on lance une unique instance de notre machine virtuelle

Une fois la machine lancée, pour vous connectez, il vous suffit de taper la commande suivante :

```
\$ pcocc ssh vm0
```

Listing 4 – Connexion à la machine virtuelle

Vous pouvez vous connecter en `root` avec le mot de passe `root` en utilisant la commande `su`.

Vous obtenez donc un shell dans la machine virtuelle. Pour la suite du TP, soit vous ouvrez autant de shell que nécessaire, soit vous utilisez la commande `screen`.

3 Les outils de gestion des réseaux IB

Comme pour le précédent TP, nous allons utiliser les outils suivants :
- ibsim : simulateur de réseau Infiniband
- opensm : Subnet Manager

Comme le but de ce TP n'est pas de travailler sur la partie topologie, nous vous fournissons une topologie de base.

Pour charger la topologie, il suffit de lancer la commande suivante :

```
ibsim -s net.txt
```

Listing 5 – Lancement du simulateur Infiniband

Ensuite, pour pouvoir se connecter au simulateur, il faut pré-charger la bibliothèque de ibsim.

```
export LD_PRELOAD=/usr/lib64/umad2sim/libumad2sim.so
```

Listing 6 – Chargement de la bibliothèque de bypass d'ibsim

3.1 Le subnet manager : opensm

Comme nous l'avons vu dans les différents cours, pour pouvoir communiquer, les nœuds ont besoin d'avoir un identifiant unique au sein d'un même réseau. Cet identifiant est donné par le subnet manager. Pour nous, ce sera `opensm`.

Nous allons commencer par lancer le subnet manager sans aucun paramètre ou presque. Nous allons uniquement donner l'information de topologie.

```
opensm -R tree -f stdout
```

Listing 7 – Lancement OpenSM

Ex. 1 — Information de topologie

1. Quelles sont les informations fournies par OpenSM ? D'un point de vu topologique, mais aussi les informations sur la fabric

Relancez le subnet manager en mode verbeux.

Ex. 2 — OpenSM en mode verbeux

1. Quelle option faut-il donner à OpenSM pour qu'il soit en mode verbeux ?

Une fois OpenSM un peu plus loquace pour avoir les informations du routage et de la topologie.

Ex. 3 — Informations détaillées d'OpenSM

1. Quelles sont les informations fournies par OpenSM dans ce nouveau mode ? Qu'est-ce que cela nous apprend sur la fabric ?

On vérifie que les nœuds et les switchs déclarés dans le fichier de topologie sont bien vus :

```
ibnetdiscover (-l)
```

Listing 8 – IbNetDiscover

3.2 Interrogation de la fabric

Ensuite, le but va être d'interroger les différentes cartes grâce aux trames MAD. Pour connaitre ces propres informations, on commence par :

```
ibstat
```

Listing 9 – ibstat

Ex. 4 — Informations données par ibstat

1. Sur quel nœud de la fabric est réalisé la commande ibstat ?

2. Détaillez les informations fournies par la commande ibstat

Cette première commande nous permet d'obtenir un certain nombre d'information sur la carte/les cartes locales. Nous allons maintenant voir comment obtenir des informations détaillées au travers des autres commandes.

Pour obtenir des informations plus détaillées sur notre nœud, nous allons utiliser la commande `smpquery` comme nous l'avons vu en cours.

Voici un premier d'exemple d'utilisation de la commande `smpquery` :

```
1 smpquery nodeinfo lid info_routage
```

Listing 10 – SMPQuery

Pour l'utiliser, il vous faut donc connaître votre LID.

Si votre LID est 1, alors la commande s'utilise de cette manière :

```
1 smpquery nodeinfo 1 1
```

Listing 11 – SMPQuery

— lid : lid du port voulu

— info_routage : chemin direct

Si vous ne connaissez pas le LID mais que vous connaissez le GUID, vous pouvez spécifier de la manière suivante : -G guid

Ex. 5 — Informations données par smpquery

1. Quelle commande vous a permis de connaître votre LID ?
2. Détaillez (en fonction de ce qui a été présenté en cours) les informations fournies par la commande smpquery

Une autre commande permet de récupérer de l'information sur les éléments de notre fabric, toujours basée sur le LID du nœud. A la différence de la commande précédente, elle prend en argument optionnel le LID du nœud.

```
1 saquery (sans lid)
saquery lid
```

Listing 12 – SAQuery

Si le LID de votre nœud est 1, alors la commande saquery s'utilise de la manière suivante :

```
saquery 1
```

Listing 13 – SAQuery

Ex. 6 — Informations données par saquery

1. Quel est le comportement de la commande squery sans argument ?
2. En comparant avec les résultats donnés par la smpquery, quelles sont les différences (ou les informations supplémentaires) avec la commande saquery ?
3. Quelles sont les options de smpquery à utiliser pour obtenir les mêmes informations que la commande saquery ?

Nous avons obtenu un certain nombre d'information de la part de ces différentes commandes. Ces informations ont été récupérées uniquement sur un nœud en locale. Refaite les mêmes manipulations mais en modifiant le LID cible.

Ex. 7 — Informations données de manière distante

1. Quelles sont les commandes qui peuvent être utilisées pour récupérer les LID de toutes la fabric ?
2. Quel outil contient aussi ces informations ?
3. Quelles sont les commandes qui peuvent être faites uniquement en locale sur le nœud ?
4. Quelles sont les commandes qui peuvent être faites pour interroger les nœds distants ?

Nous avons fait réaliser ces manipulations pour 2 nœuds, un local et un connu. Maintenant, l'objectif est de le faire pour l'ensemble de la fabric.

Dans un script bash, récupérez l'ensemble des informations déjà vues et affichez le résultat. (attention à bien faire le LD_PRELOAD pour que les commandes soient passées dans le simulateur).

Si une carte à plusieurs ports, on peut vouloir récupérer uniquement les informations d'un seul port. Pour cela, on va utiliser la commande suivante :

```
ibportstate lid query
```

Listing 14 – IbPortState

Ex. 8 — Informations données par ibportstate

1. Quelles sont les informations fournies par la commande ibportstate ?

Ajoutez cette commande au script précédent.

On veut aussi avoir une vision plus globale de notre fabric. Par exemple, on va vouloir connaitre l'état de tous les liens et savoir s'ils négocient bien à la bonne vitesse. Pour cela, il suffit de taper la commande suivante :

```
iblinkinfo check
```

Listing 15 – iblinkinfo

Ex. 9 — Informations données par ibportstate

1. Quelles sont les informations fournies par la commande iblinkinfo ?

3.3 Debug de la fabric

On peut savoir comment se comporte le routage et quel est le chemin emprunté par un message entre deux nœuds. La commande `ibtracert` va nous permettre de voir cela ; (on peut la rapprocher de la commande `traceroot`)

Il est nécessaire de connaitre les lid. Nous avons plusieurs solutions pour cela. Vous avez déjà plusieurs solutions dans les précédentes questions, en voici une donnée par le simulateur.

1. Taper la commande `Dump` dans `ibsim`

Ensuite, il suffit de taper la commande comme suit (pour la suite, prenez deux lid éloignés) :

```
ibtracert src dst
```

Listing 16 – IbTracert

Pour avoir un résultat "concluant", prenez deux nœds qui ne sont pas sur la même leaf.

Ex. 10 — Informations données par ibtracert

1. En vous basant sur les résultats d'`ibtracert`, commentez le trajet emprunté par le message entre les nœuds

D'autres commandes peuvent être utilisées pour débugger la fabric.

```
ibroute
ibaddr
```

Listing 17 – Autres commandes

Une autre commande permet de diagnostiquer l'ensemble de fabric. Cependant, elle n'est pas installée par défaut. Pour l'installer, rendez-vous dans le dossier /home/gros/MLNX_OFED_Linux-4.2-1.2.0.0-rhel7.4-x86_64\T1\textsectionRMPS puis tapez la commande suivante :

```
rpm -iv ./ibutils2-2.1.1-0.92.MLNX20170822.g2c10d6f.42120.x86_64.rpm
```

Une fois installée, il vous suffit de taper la commande

```
ibdiagnet
```

Cette commande va générer un certain nombres de fichier dans le dossier /var/tmp/ibdiagnet2/

Ex. 11 — Informations données par ibdiagnet

1. En vous basant sur les résultats d'`ibdiagnet`, commentez les résultats obtenus.

Attention : la commande ibdiagnet crée un segmentation fault d'opensm (c'est à cause du simulateur), pensez à relancer l'opensm avant de passer à la suite de ce TP.

4 Partitions

Comme on l'a vu pendant le cours, il est possible de segmenter le réseau Infiniband pour n'autoriser que certains nœud a communiqué entre eux. Pour cela, il est nécessaire de définir une (ou plusieurs) partition(s). Ces partition sont définies dans le fichier (par défaut)/etc/opensm/partitions.conf.

Dans un premier temps, il est nécessaire de récupérer le **PortGuid** du nœud voulu.

```
smpquery nodeinfo 11
```

Listing 18 – SmpQuery

Une fois le **PortGuid** obtenu, on l'ajoute dans le fichier partitions.conf avec la syntaxe suivante.

```
lustre=0x0020, ipoib, mtu=5, sl=0 : 0x0000000000100011=full;
```

Listing 19 – Lancement OpenSM

Il est nécessaire de relancer l'opensm pour qu'il prenne en compte les modifications. Une fois cela fait, on interroge le nœud concerné et on dumpe sa table de `partition keys`.

```
smpquery pkeys 11
```

Listing 20 – Lancement OpenSM

On voit dans le résultat que le pkey a bien été appliquée.

```
0: 0x7fff 0x8020 0x0000 0x0000 0x0000 0x0000 0x0000 0x0000
```

Listing 21 – Lancement OpenSM

Créez 5 partitions (partition lustre incluse) avec 5 pkeys différentes et attribuez-les aux différents nœuds de votre fabric. Certains nœuds vont avoir plusieurs pkeys.

1. Lustre_OSS_cluster_1 : 0x020
2. Lustre_OSS_cluster_2 : 0x040
3. Luste_MDS : 0x060
4. Services : 0x080

Ex. 12 — Partitions

1. Donnez le contenu de votre fichier partitions.conf

5 Sécurité

La sécurité du réseau d'interconnect est indispensable.

Ex. 13 — Sécurité

1. Dans une petite étude, basé sur les éléments vus dans les différents cours, proposez des éléments de sécurisation du réseau.

6 Qualité de service

Même si elle ne peut être utilisée dans le simulateur, nous allons voir comment créer des politiques de QoS.

Ex. 14 — Qualité de Service

1. En vous basant sur le cours, proposez une QoS en fonction des pkeys précédement définies. Mettez votre configuration dans le rapport.

7 Exercices pratiques

Nous avons le besoin suivant :
— 1800 nœuds
— un radix de 36

— HCA vers LA : cuivre : coût 0
— L1 vers L2 : fibre : coût 1 (même chose quelque soit le niveau)
— Switch : 10

Ex. 15 — XGFT non pruné

1. En vous basant sur un XGFT non pruné, donnez en détails la topologie : nombre de switchs par niveau, nombre de niveau, etc.
2. D'après les coûts donnés, estimez le coût global de ce réseau d'interconnect.

Ex. 16 — XGFT pruné

1. En vous basant sur un XGFT pruné à 50% à partir du L1, donnez en détails la topologie : nombre de switchs par niveau, nombre de niveau, etc.
2. D'après les coûts donnés, estimez le coût global de ce réseau d'interconnect.

Ex. 17 — Autre topologie

1. En vous basant sur les exemples du cours, proposez une nouvelle topologie pour ce réseau d'interconnect avec son coût