TP - Mise en place d'un cluster de stockage RozoFS

Journée Mathrice - Mardi 13 octobre 2015 - Orsay / IHES

**Intervenants [Rozo Systems](www.rozosystems.com) ** :

- Sylvain DAVID - *Ingénieur R&D* - <sylvain.david@rozosystems.com>
- Didier FERON - *Directeur Technique* - <didier.feron@rozosystems.com>

Introduction

But du TP

Le but de ce TP est de mettre en place un cluster de stockage RozoFS minimal sur 4 machines virtuelles via l'utilitaire [Vagrant](https://www.vagrantup.com) et de tester/explorer les fonctionnalités de RozoFS.

Pour rappel, [RozoFS](https://github.com/rozofs/rozofs) est :

- un système de fichier
- distribué
- tolérant à la panne
- scalable (horizontalement et verticalement)
- ...

Description de l'environnement de test utilisé

La plateforme de test est constitué de 4 machines virtuelles [VirtualBox](https://www.virtualbox.org/) interconnectées via un réseau virtuel (Host-Only Networking). Le système d'exploitation utilisé est Debian 8 ([jessie](https://www.debian.org/releases/stable/index.fr.html)).

L'adressage réseau utilisé est le suivant :

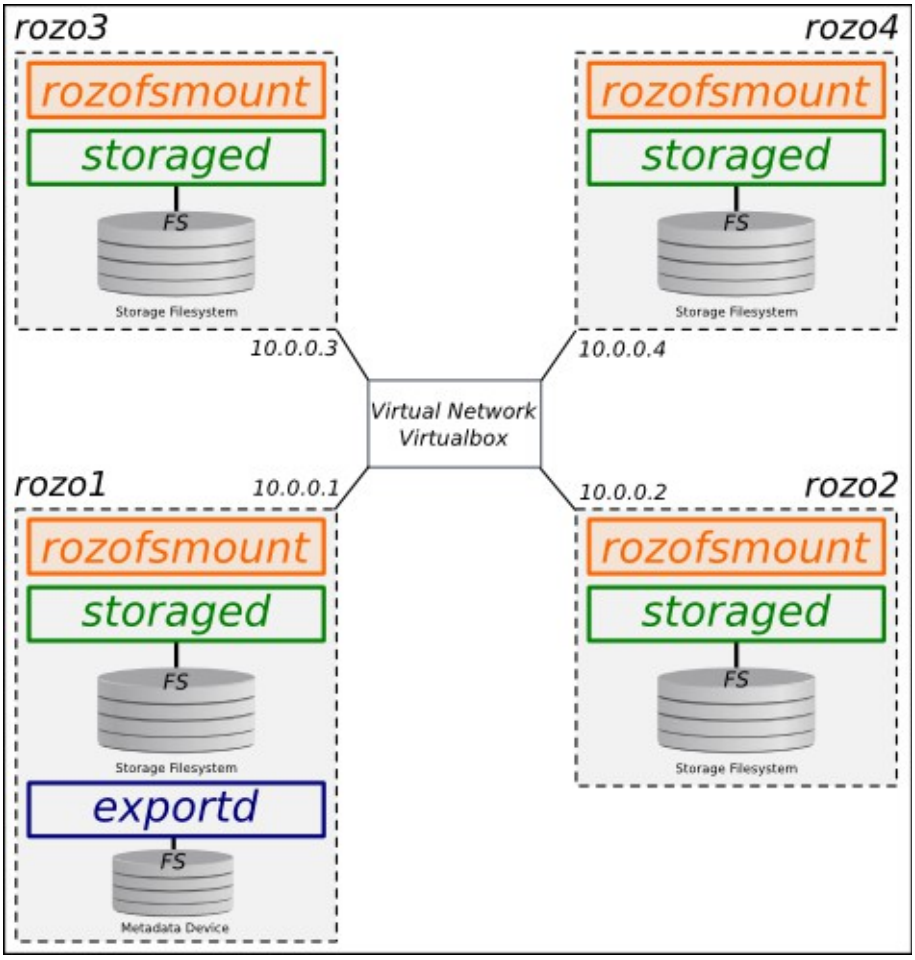
VM	@IP
rozo1	10.0.0.1/8
rozo2	10.0.0.2/8
rozo3	10.0.0.3/8
rozo4	10.0.0.4/8

Les packages RozoFS à utiliser sont les suivants :

Nom du package	Rôle
rozofs-exportd	Serveur de méta-données (localisation des projections, attributs des fichiers et arborescence).
rozofs-storaged	Serveur de stockage (gère les projections).
rozofs-rozofsmount	Client du système de fichiers (communique avec l'*exportd* et *storaged*). Il est responsable de la transformation des données
rozofs-rozodiag	Utilitaire permettant d'obtenir des statistiques sur les différents composants de RozoFS.
rozofs-manager-*	Utilitaire en commande ligne permettant de configurer les différents composants de RozoFS.

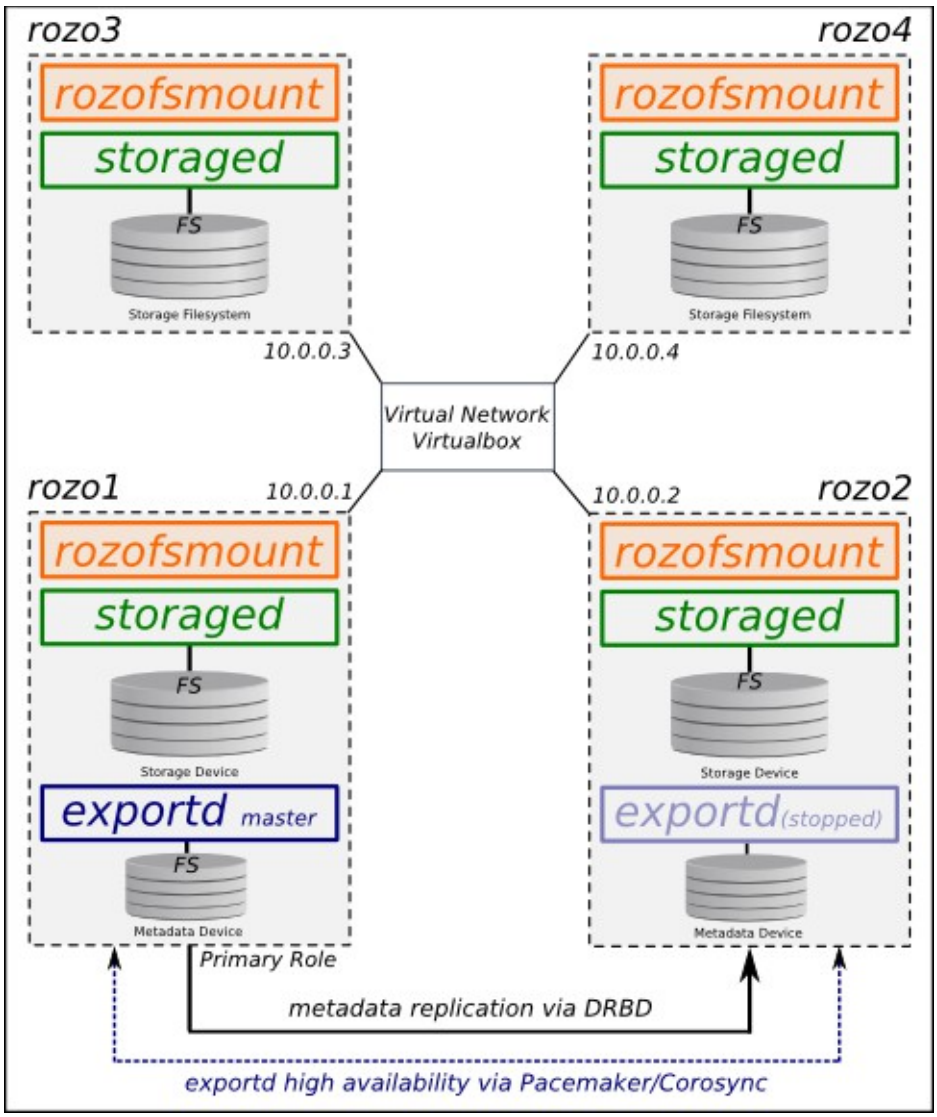
Les étapes du TP

1 - Mise en place d'un cluster RozoFS simple sans haute-disponibilité du serveur de méta-données comme décrit dans le schéma ci-dessous.



2 - Utilisation/Test du cluster et exploration de l'outil *rozodiag*

3 - Mise en place d'un cluster RozoFS simple avec haute-disponibilité du serveur de méta-données comme décrit dans le schéma ci-dessous.



Mise en place d'un cluster RozoFS simple

Les étapes nécessaires pour mettre en place un cluster RozoFS sont :
- la céation d'un volume RozoFS sur les 4 machines virtuelles.
- l'ajout d'un système de fichier exporté utilisant le volume précédemment créé.
- Le montage des clients RozoFS sur les 4 machines virtuelles.

Configuration d'un volume RozoFS

Rappel de quelques définitions

storaged node : un serveur stockant des projections sur n emplcemant logique de stockage (*storage*).
storage : emplacement logique de stockage identifié par un `SID`, un *storage* peut utiliser plusieurs systèmes de fichiers sous-jacents.
device : un système de fichiers contenant les données des fichiers stockés dans RozoFS.
cluster : ensemble de *storage* (`host`, `SID`) utilisé pour stocker les données d'un fichier (identifiant : `CID`)
volume : un ensemble de *cluster* utilisé pour socker les données de tous les fichiers d'un même système de fichiers. Un *volume* utilise une configuration de redondance particulière (*layout*).
layout : configuration de redondance utilisée pour transformer et stocker les blocs de données.

Configuration du volume souhaité

Ici, on va mettre en place un *volume* (`vid=1`) contenant un seul *cluster* (`cid=1`). Le *cluster* contiendra 4 *storages* (`sid=[1-4]`) localisés sur les VMs rozo[1-4]. Chaque *storage* sera constitué que d'un *device*.

Tâches à accomplir :

- Installer les packages RozoFS nécessaires (`rozofs-rozolauncher` et `rozofs-storaged`) sur les noeuds de stockage.
- Mise en place du système de fichier nécessaire au stockage des données (projections).
- Configuration du démon storaged (ajout d'un *storage* dans le fichier `storage.conf`).
- Installer le package `rozofs-exportd` sur *rozo1*.
- Configuration du démon exportd (ajout d'un *volume* dans le fichier `export.conf`).

Installation du composant storaged

Installation via l'utilitaire *dpkg* :

```
$ dpkg -i rozofs-rozolauncher-<version>.deb
$ dpkg -i rozofs-storaged-<version>.deb
```

Mise en place des systèmes de fichiers (stockage projection)

Cette étape consiste à identifier le *device block* à utiliser, initialiser le système de fichier (*xfs* ou *ext4*) et à le monter.

Exemple avec le device `/dev/sdb` :

```
$ mkfs.xfs /dev/sdb
$ mkdir -p /srv/rozofs/storage_<cid>_<sid>/<device_idx>
$ mount /dev/sdb /srv/rozofs/storage_<cid>_<sid>/<device_idx>
```

Configuration des démons de stockage (*storaged*)

La déclaration et la prise en compte d'un nouveau *storage* par le démon storaged consiste à ajouter le nouveau *storage* dans le fichier de configuration (`storage.conf`) puis à redémarrer le démon.

Exemple de fichier `storage.conf` sur *rozo1* :

```
$ cat /etc/rozofs/storage.conf
crc32c_check = true;
crc32c_generate = true;
crc32c_hw_forced = true;
listen = (
      {
            addr = "*";
            port = 41001;
      } );
storages = (
      {
            cid = 1;
            sid = 1;
            root = "/srv/rozofs/storages/storage_1_1";
            device-total = 1;
            device-mapper = 1;
```

```
            device-redundancy = 1;
      } );
```

Redémarrage du démon rozofs-storaged :
```
$ /etc/init.d/rozofs-storaged restart
[ ok ] Restarting rozofs-storaged (via systemctl): rozofs-storaged.service.
```

Notes :
- Consulter les pages man (`storage.conf`, `storaged`) pour avoir plus d'informations sur la configuration d'un noeud de stockage RozoFS.
- Attention, le paramètre `SID` ne doit pas être identique sur les 4 VMs constituant le cluster.
- Il est possible d'utiliser l'outil `rozo` (bêta) pour configurer les 4 VMs en une seule commande (`man rozo`).

Installation du composant exportd

Installation via l'utilitaire *dpkg* :

```
$ dpkg -i rozofs-exportd-<version>.deb
```

Configuration du démon de méta-données (*exportd*)

Une fois les 4 démons de stockage configurés il convient d'ajouter un *volume* constitué du *cluster* des 4 storages configurés précédemment dans le fichier de configuration (`export.conf`) puis de recharger le démon.

Exemple de fichier `export.conf` sur rozo1 :

```
$ cat /etc/rozofs/export.conf
layout = 0;
volumes = (
      {
            vid = 1;
            cids = (
                  {
                        cid = 1;
                        sids = (
                              {
                                    sid = 1;
                                    host = "10.0.0.1";
                              },
                              {
                                    sid = 2;
                                    host = "10.0.0.2";
                              },
                              {
                                    sid = 3;
                                    host = "10.0.0.3";
                              },
                              {
                                    sid = 4;
                                    host = "10.0.0.4";
                              } );
                  } );
      } );
exports = ( );
```

Redémarrage du démon rozofs-exportd :
```
$ /etc/init.d/rozofs-exportd reload
[ ok ] Reloading rozofs-exportd configuration (via systemctl): rozofs-
exportd.service.
```

Notes :
- Consulter les pages man (`export.conf`, `exportd`) pour avoir plus d'informations sur la configuration d'un noeud méta-données RozoFS.
- Il est possible d'utiliser l'outil `rozo` (bêta) pour ajouter un *volume* en une seule commande (modifications des fichiers `storage.conf` et `export.conf` et redémmarage des démons)(voir `man rozo`).

Ajout d'un système de fichiers exporté RozoFS (*export*)

Définition

export : un système de fichier exporté RozoFS a pour but d'être utilisé sur *n* machines. Un *export* est associé à un seul *volume* donc les données des fichiers de cet *export* sont stockées dans les différents *storages* constituant le *volume*. Les méta-données du système de fichiers sont stockées dans un système de fichiers local au serveur.

**Configuration de l'export souhaité **

Le but est de créer un système de fichiers (`eid=1`) qui utilisera le *volume* configuré précédemment (`vid=1`).

Tâches à accomplir :

- Mise en place du système de fichier nécessaire au stockage des méta-données (projections)
- Configuration du démon exportd (ajout d'un *export* dans le fichier `export.conf`)

Mise en place du système de fichiers (stockage des méta-données)

Cette étape consiste à identifier le *device block* à utiliser, initialiser le système de fichier *ext4* et à le monter.

Exemple :
```
$ mkfs.ext4 /dev/sdc
$ mkdir  -p /srv/rozofs/exports/
$ mount /dev/sdb /srv/rozofs/exports
$ mkdir /srv/rozofs/exports/export_1
```

Ajout d'un export

La déclaration et la prise en compte d'un nouvel *export* par le démon exportd consiste à ajouter un *export* dans le fichier de configuration (`export.conf`) puis à rédémarrer le démon.

Extrait du fichier `export.conf` sur *rozo1* :
```
$ tail -n 9 /etc/rozofs/export.conf
exports = (
      {
            eid = 1;
            vid = 1;
            root = "/srv/rozofs/exports/export_1";
```

```
            md5 = "";
            squota = "";
            hquota = "";
      } );
```

Création du répertoire stockant les méta-données de l'export (`eid=1`):

```
$ mkdir -p /srv/rozofs/exports/export_1
```

Rechargement du démon rozofs-exportd :
```
$ /etc/init.d/rozofs-exportd reload
[ ok ] Reloading rozofs-exportd configuration (via systemctl): rozofs-
exportd.service.
```

Montage du système de fichiers RozoFS (client) sur les VMs

Définition

Client rozoFS : partie cliente d'un système de fichiers RozoFS, un client a besoin de savoir (paramètres) l'adresse IP du serveur de méta-données ainsi que l'*export* à utiliser. Il communique avec le serveurs de méta-données, avec les serveurs de stockage et est de responsable de la transformation des données.

**Configuration de l'export souhaité **

Le but est de monter le système de fichiers (`eid=1`) sur les 4 VMs constituants la plate-forme.

Tâches à accomplir :

- Installation du client RozoFS (package *rozofs-rozofsmount*)
- Configuration et montage d'un client RozoFS

Installation du client RozoFS (*rozofsmount*)

Installation via l'utilitaire *dpkg* :
```
$ dpkg -i rozofs-rozofsmount-<version>.deb
```

Configuration et montage d'un client RozoFS

Cette étape consiste à créer le point de montage, le configurer via le fichier `/etc/fstab` et de monter le système de fichier RozoFS sur les 4 VMs.

Création du point de montage RozoFS :

```
$ mkdir -p /mnt/rozofs/export_1
```

Extrait du fichier `/etc/fstab` sur *rozo1* :
```
$ tail -n 1 /etc/fstab
rozofsmount /mnt/rozofs/export_1    rozofs

exporthost=rozo1,exportpath=/srv/rozofs/exports/export_1,instance=0,_netdev  0
      0
```

Montage du système de fichier :
```
$ mount /mnt/rozofs/export_1/
```

Note :

- il est possible d'utiliser l'utilitaire `rozo` (version bêta) afin de configurer et monter l'ensemble des points de montage en une seule commande.

Utilisation/Test du cluster et exploration de l'outil *rozodiag*

Le but des manipulations suivantes est de mettre en évidence le fonctionnement de RozoFS via des tests d'écriture/lecture et d'analyser les sorties de l'utilitaire `rozodiag`.

Tests d'écriture/lecture

Une fois le cluster RozoFS monté, on peut commencer à écrire et lire des données sur les différents points de montage de la plate-forme.

```
$ cd /mnt/rozofs/export_1
$ dpkg-query -l > test-`uname -n`.txt
$ cat test-`uname -n`.txt
```

Utilisation de l'utilitaire `rozodiag`

RozoFS possède une multitude de compteurs utilisés afin d'observer l'utilisation du système de fichier par les applications/utilisateurs mais aussi pour diagnostiquer plus aisement les problèmes éventuels. L'outil `rozodiag` permet de consulter les valeurs des compteurs définis dans les différents composants logiciels de RozoFS.

Les exemples suivants permettent de consulter le topic 'profiler' sur les différents composants logiciels :
```
# rozofsmount component
$ rozodiag -i localhost -T mount:0 -c profiler <reset>
# storcli component
$ rozodiag -i localhost -T mount:0:1 -c profiler <reset>
# storaged component
$ rozodiag -i localhost -T storaged -c profiler <reset>
# storio component
$ rozodiag -i localhost -T storio:1 -c profiler <reset>
# exportd component
$ rozodiag -i localhost -T export:1 -c profiler <reset>
```

En utilisant les commandes ci-dessus, essayer d'observer :
- les opérations VFS effectuées lors de l'utilisation de la commande ``dpkg-query -l > test-`uname -n`.txt``.
- les opérations VFS effectuées lors de l'utilisation de la commande ``cat test-`uname -/n`.txt``.
- le temps pris pour envoyer les projections du fichier.
- le nombre d'opérations effectuées concernant les métadonnées.
- les machines virtuelles utilisées pour stocker les projections de ce fichier.

Simulation de pannes

Il est possible de connaître la localisation (par défaut) des différentes projections des fichiers stockés sur RozoFS en consultant l'attribut étendu

rozofs comme dans la commande ci-dessous :
```
$ attr -g rozofs test-rozo1.txt
Attribute "rozofs" had a 399 byte value for test-rozo1.txt:
EID     : 1
VID     : 1
LAYOUT  : 0
BSIZE   : 0
NAME    : test-rozo1.txt
FID     : 00000000-0000-4000-1800-000000000810
SPLIT   : vers=0 fid_high=0 opcode=0 exp_id=0 eid=1 usr_id=24 file_id=0 idx=8
key=2 (REG)
HASH1/2 : 6508cd81/5e49b899 (000000d0)
UID/GID : 0/0
CREATE  : Mon Oct 12 08:55:54 2015
MODE    : REGULAR FILE
CLUSTER : 1
STORAGE : 004-002-003-001
NLINK   : 1
SIZE    : 49928
LOCK    : 0

```

Maintenant que la distribution du fichier est connue, essayer d'arrêter ou supprimer un *storage* utilisé pour stocker le fichier précédemment créé et tenter de relire/re-écrire ce fichier.

Vous pouvez aussi essayer de lire un fichier alors que 2 storages sont indisponibles (mais pas n'importe lesquelles).

Essayer ces opérations pendant une **lecture/écriture**.

Reconstruction d'un noeud de stockage RozoFS

RozoFS permet de reconstruire les données/projections d'un noeud de stockage en utilisant les informations présentes sur les autres noeuds.

Exemple de reconstruction d'un noeud :

```
$ storage_rebuild -r 10.0.0.1 -p 1
Start rebuild process (cid=1;sid=3).
2 files to rebuild by 1 processes
   -> /tmp/rbs.2336/cid1_sid3/storage.0 rebuild start
  <-  /tmp/rbs.2336/cid1_sid3/storage.0 rebuild success of 2 files
cid 1 sid 3 Rebuild success.
# This file was generated by storage_rebuild(8) version: %s.
# All changes to this file will be lost.
2.0.~alpha45
starting  : Mon Oct 12 09:24:49 2015

command   : storage_rebuild -r 10.0.0.1 -p 1
parallel  : 1
status    : completed
loop      : 1
* cid/sid : 1/3
            - layout   : 0
            - nb files : 2/2
            - written  : 29008
                         . nominal : 29008
                         . spare   : 0
            - read     : 58240
                         . nominal : 58240
```

```
                          . spare   : 0
* total   :
            - nb files : 2/2
            - written  : 29.0 KB (29.0 KB/s)
                         . nominal : 29.0 KB
                         . spare   : 0 Bytes
            - read     : 58.2 KB (58.2 KB/s)
                         . nominal : 58.2 KB
                         . spare   : 0 Bytes
delay     : 1 sec
```

A l'aide de la commande précédente, essayer de supprimer des fichiers projections et de les reconstruire.

Mise en place de la HA (DRBD/Corosync/Pacemaker)

Vous avez pu observer qu'en utilisant la redondance induite par la transformée mathématique Mojette, RozoFS permet de passer outre la panne de *n* serveurs. Cependant le serveur de méta-données utilisé par RozoFS reste un point de défaillance unique (SPOF). Pour cela il est nécessaire de mettre en place une stratégie de haute-disponibilité pour ce service.

Pour mettre en place la sécurisation du service *rozofs-exportd*, il est possible d'utiliser les outils suivants :

- *[DRBD](http://drbd.linbit.com/)* (synchronizes data at the block device)
- *[Corosync](https://corosync.github.io/corosync/)* (Totem single-ring ordering and membership protocol)
- *[Pacemaker](http://clusterlabs.org/)* (High Availability and load balancing stack for the Linux platform)

Mettez en place l'architecture suivante en utilisant les ressources contenues dans le wiki de [RozoFS] (http://rozofs.github.io/rozofs/develop/SettingUpRozoFS.html#preparing-nodes) et sur le site de ces différents outills.

Une fois cette architecture mise en place, il est possible de tester :

- La bonne migration du service *rozofs-expord*
- la continuité de service dans le cluster RozoFS