



UNIVERSITÉ DE LA ROCHELLE

RAPPORT DE PROJET

Dossier d'Architecture Technique

Auteurs :

Joris BERTHELOT
Laurent LE MOINE

Superviseurs :

Philippe HARRAND
Bertrand VACHON

Master ICONE 2011-2012

Table des matières

Introduction	3
Postes de travail	3
Code source	3
I Infrastructure logicielle	4
0.1 Installation des paquets	5
0.1.1 Configuration du proxy	5
0.2 Réplication bas niveau	5
0.2.1 Installation	6
0.2.2 Configuration	6
0.3 Déploiement des services	9
0.3.1 Apache	9
0.3.1.1 Installation	9
0.3.1.2 Configuration	9
0.3.2 MySQL	10
0.3.2.1 Installation	10
0.3.2.2 Configuration	10
0.3.3 DNS	11
0.3.4 LDAP	13
0.4 Mise en place de Pacemaker et Corosync	15
0.4.1 Présentation	15
0.4.2 Installation	16
0.4.3 Configuration	16
0.4.3.1 Corosync	16
0.4.3.2 Pacemaker	17
0.4.4 Utilisation	18
II Tests	20
0.5 Serveur de noms	21
0.6 Serveur Apache	21
0.7 Serveur MySQL	21
0.8 Serveur LDAP	21

III	Conclusion	22
0.9	Difficultés rencontrés	23
0.10	Retours sur échec	23
0.11	Retours personnels	24
IV	Annexes	25
.1	Bind	26

Introduction

Dans le cadre de notre formation Master Ingénierie Informatique et de son Unité d'Enseignement Architecture : Conception et Gestion, nous avons réalisé un projet d'architecture réseau HA (High Availability) en 3 jours seulement.

Ce projet nous a permis de mettre en exergue nos connaissances récemment acquises lors des cours respectifs de la même UE mais aussi de reprendre et appliquer les concepts vus en TP la semaine auparavant.

Assigné comme table n°5, nous avons utilisé les postes suivants :

Joris Berthelot (sera la machine « JB » dans le reste du rapport)

- Adresse IP : 10.192.10.23
- Host : mamba13

Laurent Le Moine (sera la machine « LLM » dans le reste du rapport)

- Adresse IP : 10.192.10.24
- Host : mamba14

Etant donné que ce projet fut réalisé en équipe, le code source des différents scripts et fichiers de configuration sont disponible sur Google Code + Subversion. Ainsi, vous pouvez à tout moment récupérer notre travail (ainsi que le code L^AT_EX du document) comme ceci :

```
svn export http://ulr-acg.googlecode.com/svn/trunk/ ulr-acg-src
```

Première partie

Infrastructure logicielle

Avant toute choses, vous devez savoir que l'ensemble des opérations décrites dans cette section sont à réaliser avec l'utilisateur root. Si vous lancez les scripts livrés avec le rapport sans être root, vous aurez droit à un gentil message d'erreur.

Nous avons aussi par ailleurs vidé et désactivé les tables de pare-feu afin de laisser toutes nos applications travailler sans avoir de gêne dans un premier temps :

```
# Flushes iptables
/sbin/chkconfig --del iptables
# Disables firewall service
service iptables stop
# Enables Network Time Protocol to sync time between machines
service ntpd start
```

0.1 Installation des paquets

Avant de commencer à configurer et déployer les services, nous aurons besoin d'installer un certain nombre de paquets afin de pouvoir parvenir à nos fins. Aussi divers que variés, nous avons scripté cette installation afin de faciliter la tâche.

0.1.1 Configuration du proxy

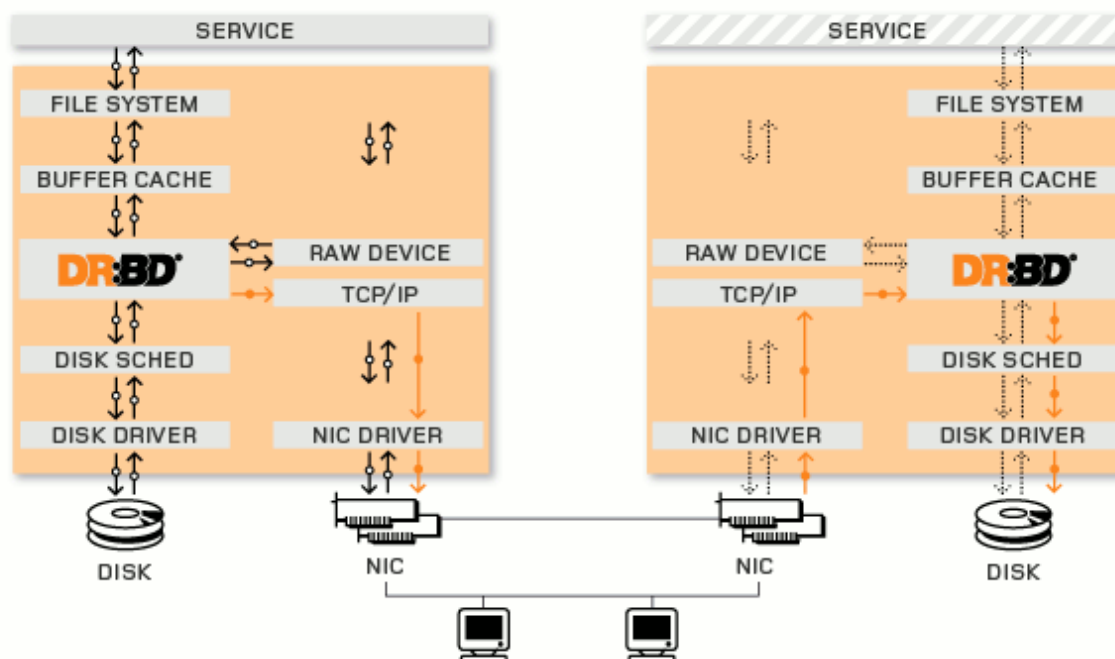
Il faudra auparavant configurer manuellement les paramètres du proxy si besoin afin de ne pas rendre le script d'installation inopérant. Pour se faire, veuillez à bien changer les paramètres dans les Serveurs mandataires (Système > Configuration > Serveurs mandataires) ainsi que rajouter les bons paramètres à Yum (proxy) :

```
# Configuration de Yum
vim /etc/yum.conf
```

0.2 Réplication bas niveau

La réplication bas niveau permet de créer non pas une redondance applicative mais directement sur le support des données applicatives (système de fichier). L'intérêt à cela est d'éviter de configurer chaque service pour sa propre réplication (si existant) et d'aller droit au but en répliquant directement le volume sur lequel repose les données.

Voici un petit schéma (issu du site de DRBD) afin d'imager le concept :



0.2.1 Installation

Pour la réplication bas niveau (système de fichiers), nous avons utilisé **DRBD** : un logiciel permettant de faire de la réplication de données au sein d'une architecture en grappe. DRBD est assez complexe et fastidieux à mettre en place car il demande quelques notions assez poussées sur les volumes, leur synchronisations, etc.

Ce logiciel ne fonctionnait autrefois qu'en mode maître/esclave mais depuis les dernières versions, on peut partir sur une configuration maître/maître afin que les données soient bien synchronisées de manière bidirectionnelle.

```
# Installation de drbd
yum install -y drbd drbd-pacemaker drbd-udev
```

0.2.2 Configuration

Avant de configurer DRBD, il faut choisir où seront stockées nos données et comment s'y prendre. Dans une configuration idéale, il aurait fallu stocker nos données sur des partitions logiques cryptées et répliquées avec en RAID 15 mais nous n'avons pas eu le temps de nous soucier de cela donc nous avons vu simple : une partition logique dans une groupe de volumes.

Pour se faire, nous avons utilisé le disque `/dev/sdb` vide par défaut et nous y avons créé une partitions de type Linux LVM :

```
fdisk /dev/sdb
> d
> <ENTER>
> n
> p
> <ENTER>
> <ENTER>
> <ENTER>
> t
> 8E
> w
# Création d'un volume physique à partir de /dev/sdb1
pvcreate /dev/sdb1
# Création d'un groupe de volumes
vgcreate ulr-acg /dev/sdb1
# Création d'un volume dans le groupe de volumes
lvcreate -n ulr-data -L1G ulr-acg
```

Maintenant que notre volume de données est prêts, nous décidont qu'il sera monté dans **/var/cluster** et toute les données applicatives seront dedans.

La configuration de DRBD peu s'avérer très simple mais permet un certain degré de complexité en fonction des architectures. La syntaxe est claire et s'apparente à celle du serveur de nom.

Voici la configuration que nous avons utilisé :


```
1 global {
2     # Enables statistics usage
3     usage-count yes;
4 }
5 common {
6     # Maner to sync data, it's flagged as completed when both disks has written
7     protocol C;
8 }
9 resource ulr-data {
10     # Stores data meta-data in the volume
11     meta-disk internal;
12     # DRBD device
13     device /dev/drbd1;
14     # Volume to work on
15     disk /dev/mapper/ulr--acg-ulr--data;
16     syncer {
17         # Hash method to check data integrity
18         verify-alg sha1;
19         # Network sync speed
20         rate 100M;
21     }
22     # Allows both machine are primary
23     net {
24         allow-two-primaries;
25     }
26     # JB's machine
27     on mamba13 {
28         address 10.0.0.23:7789;
29     }
30     # LLM's machine
31     on mamba14 {
32         address 10.0.0.24:7789;
33     }
34 }
```

DRBD implique qu'un lien réseau doit être établi en supplément des liens existants. Il est important de comprendre que le DRBD utilise un réseau à lui propre afin d'y transférer les données.

Une fois configuré, nous pouvons lancer Corosync :

```
/etc/init.d/drbd start
```

Une fois le service DRBD démarré, nous devons initialiser le disque afin que le système de réplique fonctionne :

```
# Création du disque à partir du volume
drbdadm --force create-md ulr-data
# Activation du volume au sein de DRBD
drbdadm up ulr-data
# Déclaration de notre disque comme maître
drbdadm -- --overwrite-data-of-peer primary ulr-data
```

Pour vérifier que DRBD fonctionne bien, il suffit de voir son état en faisant la commande sui-

vante :

```
cat /proc/drbd
```

Si tout va bien, on peut créer le système de fichier sur le disque maître et le monter dans `/var/cluster` :

```
# Création d'un système de fichier de type EXT4
mkfs.ext4 /dev/drbd1
# Montage du système de fichier
mount /dev/drbd1 /var/cluster
```

Voici comment créer un volume logique contenant un système de fichier de type EXT4 réparti sur plusieurs machines avec DRBD.

0.3 Déploiement des services

Dans cette partie, il est important de comprendre que lors de la mise en place d'une architecture en grappe avec des noeuds répliqués, il faut toujours un noeud de référence, surtout dans une architecture active/passive comme celle que nous allons mettre en place. Suivant la machine sur laquelle vous allez lancer les scripts, il faudra ou non déployer les données.

0.3.1 Apache

0.3.1.1 Installation

Pour installer Apache, il vous faudra très simplement lancer son script d'installation dans le répertoire `scripts/apache.sh`. Ce script va essayer de stopper Apache, vérifier l'intégrité de son fichier de configuration et si il ne concorde pas avec le notre, il va le remplacer. Ensuite, selon si vous êtes le premier noeud de la grappe.

0.3.1.2 Configuration

La configuration d'Apache est très légèrement spécifique dû à notre application Web qui utilise Silex mais sinon rien de bien particulier ormis la configuration de l'URL `/server-status` qui doit être disponible pour Pacemaker.

Il faudra donc bien décommenter le bout de code suivant à la fin du fichier :

```
1 <Location /server-status>
2     SetHandler server-status
3     Order deny,allow
4     Deny from all
5     # Only from localhost where Pacemaker runs
6     Allow from 127.0.0.1
7 </Location>
```

Enfin, spécifier le `DocumentRoot` à `/var/cluster/www/org/tp/g1b5/web` et y ajouter les règles de réécritures suivantes afin de faire fonctionner notre application :

```

1 <IfModule mod_rewrite.c>
2     Options -MultiViews
3     RewriteEngine On
4     # All requests which are not a file
5     RewriteCond %{REQUEST_FILENAME} !-f
6     # Except this request
7     RewriteCond %{REQUEST_URI} !=/server-status
8     RewriteRule ^ index.php [L]
9 </IfModule>

```

Une fois configuré, nous pouvons lancer Apache :

```
/etc/init.d/corosync httpd
```

0.3.2 MySQL

0.3.2.1 Installation

L'installation de MySQL se fait très facilement grâce au script d'installation `scripts/mysql.sh`. Le script arrête le serveur MySQL, charge le fichier de configuration `confs/mysql/my.cnf` à la place de l'ancien. Il démarre ensuite le service `mysqld`, puis crée les utilisateurs et peuple la base si nous sommes sur le premier noeud de la grappe.

0.3.2.2 Configuration

Le fichier de configuration de MySQL est court, mais il est important dans notre cas de bien préciser le chemin du répertoire où seront stockées les bases, à savoir dans notre cluster, dans le répertoire `/var/cluster/mysql`. De même, il est aussi important de préciser que l'adresse utilisée par le serveur est celle de l'adresse du HAProxy : `10.192.10.50`.

```

1 [mysqld]
2 # On définit le répertoire où seront stockées les données sur le cluster
3 datadir=/var/cluster/mysql
4 # On définit l'adresse IP utilisée par le serveur MySQL
5 bind-address=10.192.10.50
6 # Chemin du fichier de socket pour les connexions locales
7 socket=/var/lib/mysql/mysql.sock
8 user=mysql
9 # On désactive les liens symboliques pour améliorer la sécurité du serveur
10 symbolic-links=0
11
12 [mysqld_safe]
13 log-error=/var/log/mysqld.log
14 pid-file=/var/run/mysqld/mysqld.pid

```

Nous devons aussi définir le mot de passe de l'utilisateur "root", qui est le même sur les deux machines. Cela se fait simplement grâce à la commande :

```
mysqladmin password <mot_de_passe>
```

Il faut ensuite créer un utilisateur “tpuser”, et lui allouer des droits de sélection sur tout les machines du réseau 10.192.10.0, afin de permettre l'accès à la base “projet_hd” depuis une machine distante.

```
1 CREATE USER 'tpuser'
2 IDENTIFIED BY 'tpuser';
3
4 GRANT ALL PRIVILEGES
5 ON *.* TO 'root'@'%';
6
7 CREATE DATABASE projet_hd;
8
9 GRANT SELECT PRIVILEGES
10 ON projet_hd.* TP 'tpuser'@'10.192.10.%';
```

Il faut aussi créer puis initialiser la table “product”, qui contient les données sur notre stock. Elle contient les champs “id”, “name”, “price” et “quantity”.

Une fois configuré, nous pouvons lancer MySQL :

```
/etc/init.d/corosync mysqld
```

0.3.3 DNS

Pour commencer, il faut installer le paquet “bind”. Comme pour les autres services, nous avons fait un script qui configure le serveur bind. Il copie tout simplement les fichiers de configuration que nous avons écrit, puis lance le servic “named”.

Nous avons choisi une configuration maître/esclave classique plutôt que d'utiliser DRBD et PaceMaker, car il n'y a pas de données à répliqué comme avec l'annuaire LDAP ou MySQL, il y a seulement les fichiers de zones.

De même, pour les adresses des serveurs DNS, nous avons utilisé l'adresse des machines plutôt que celle du proxy. Le maître est donc la machine LLM, ayant l'adresse 10.192.10.24, et l'esclave la machine JB, 10.192.10.23.

Le fichier de configuration de bind est simple, et nous avons seulement rajouté la partie concernant les fichiers de zones. Dans les fichiers de zones, l'adresse des serveurs DNS est celle des machines correspondantes, mais l'adresse de “g1b5.tp.org” et celle des différents service est celle du proxy, 10.192.10.50.

Nous n'avons pas pu donner d'adresses IPv6 aux différents services car Corosync et PaceMaker ne permettent pas de définir plusieurs adresses (ou du moins nous n'avons pas trouvé) pour le proxy, les services ont donc seulement une adresse IPv4, seul les serveurs DNS ont une adresse IPv6.

Le fichier de configuration de Bind :

```
1 options {
2     listen-on port 53 { ANY; };
3     listen-on-v6 port 53 { ANY; };
4     directory      "/var/named";
5     dump-file       "/var/named/data/cache_dump.db";
6     statistics-file "/var/named/data/named_stats.txt";
7     memstatistics-file "/var/named/data/named_mem_stats.txt";
8     allow-query     { ANY; };
9     recursion yes;
10
11     dnssec-enable yes;
12     dnssec-validation yes;
13     dnssec-lookaside auto;
14
15     version "Try again! ;)";
16
17     bindkeys-file "/etc/named.iscdlv.key";
18
19     managed-keys-directory "/var/named/dynamic";
20 };
21
22 logging {
23     channel default_debug {
24         file "data/named.run";
25         severity dynamic;
26     };
27 };
28
29 zone "." IN {
30     type hint;
31     file "named.ca";
32 };
33
34 zone "glb5.tp.org" IN {
35     type master;
36     file "/etc/named/glb5.tp.org.zone";
37 };
38
39
40 zone "0.0.0.0.9.c.0.0.0.0.0.0.0.0.d.f.ip6.arpa" {
41     type master;
42     file "/etc/named/rv6.glb5.tp.org.zone";
43 };
44
45 zone "10.192.10.in-addr.arpa" {
46     type master;
47     file "/etc/named/rv4.glb5.tp.org.zone";
48 };
49
50 include "/etc/named.rfc1912.zones";
51 include "/etc/named.root.key";
```

Les autres fichiers de configuration (zones et zones inverses) sont disponibles dans l'annexe [.1](#).

0.3.4 LDAP

Il faut tout d'abord installer le paquets "openldap", puis lancer le script d'installation, qui remplace les fichiers de configuration présents sur la machine par les notres.

Pour configurer OpenLDAP, nous avons utilisé l'ancienne méthode, avec le fichier "slapd.conf", plutôt que la nouvelle méthode qui est très peu documentée. Il faut bien penser à supprimer le répertoire `/etc/openldap/slapd.d/` car sinon OpenLDAP ne prend pas en compte le fichier "slapd.conf".

Le fichier `/etc/openldap/ldap.conf` contient seulement le suffixe de notre base :

```
1 #/etc/openldap/ldap.conf
2 # LDAP Defaults
3 #
4
5 # See ldap.conf(5) for details
6 # This file should be world readable but not world writable.
7
8 BASE      dc=g1b5,dc=tp,dc=org
9 URI       ldap://localhost
```

Le fichier `/etc/openldap/slapd.conf` est quand à lui plus fourni :

```
1  #/etc/openldap/slapd.conf
2  include      /etc/openldap/schema/core.schema
3  include      /etc/openldap/schema/cosine.schema
4  include      /etc/openldap/schema/inetorgperson.schema
5  include      /etc/openldap/schema/nis.schema
6
7  pidfile      /var/run/openldap/slapd.pid
8  argsfile     /var/run/openldap/slapd.args
9
10 # On définit une base de données bdb, ayant pour suffixe glb5.tp.org, et comme root Ma
11 # Le mot de passe du root est haché en md5
12 database      bdb
13 suffix        "dc=glb5,dc=tp,dc=org"
14 rootdn        "cn=Manager,dc=glb5,dc=tp,dc=org"
15 rootpw        {MD5}nsoz4g836Njg9I6ijrlC9w==
16
17 # On définit le chemin du répertoire qui contiendra les bases sur le cluster
18 directory     /var/cluster/ldap
19
20 # Les différents indices à maintenir dans la base de données
21 index default eq
22 index objectClass eq
23 index cn,name,surname,givenname eq,sub
24
25 # Définitions des access list, seul le manager peut écrire.
26 access to attrs=userPassword
27     by self write
28     by anonymous auth
29     by dn="cn=Manager,dc=glb5,dc=tp,dc=org" write
30     by * none
31
32 access to *
33     by self write
34     by dn="cn=Manager,dc=glb5,dc=tp,dc=org" write
35     by * read
```

Ensuite le script peuple la base à l'aide d'un fichier `.ldif`. Nous stockons dans la base nos employés, qui sont actuellement au nombre de deux, nous mêmes :

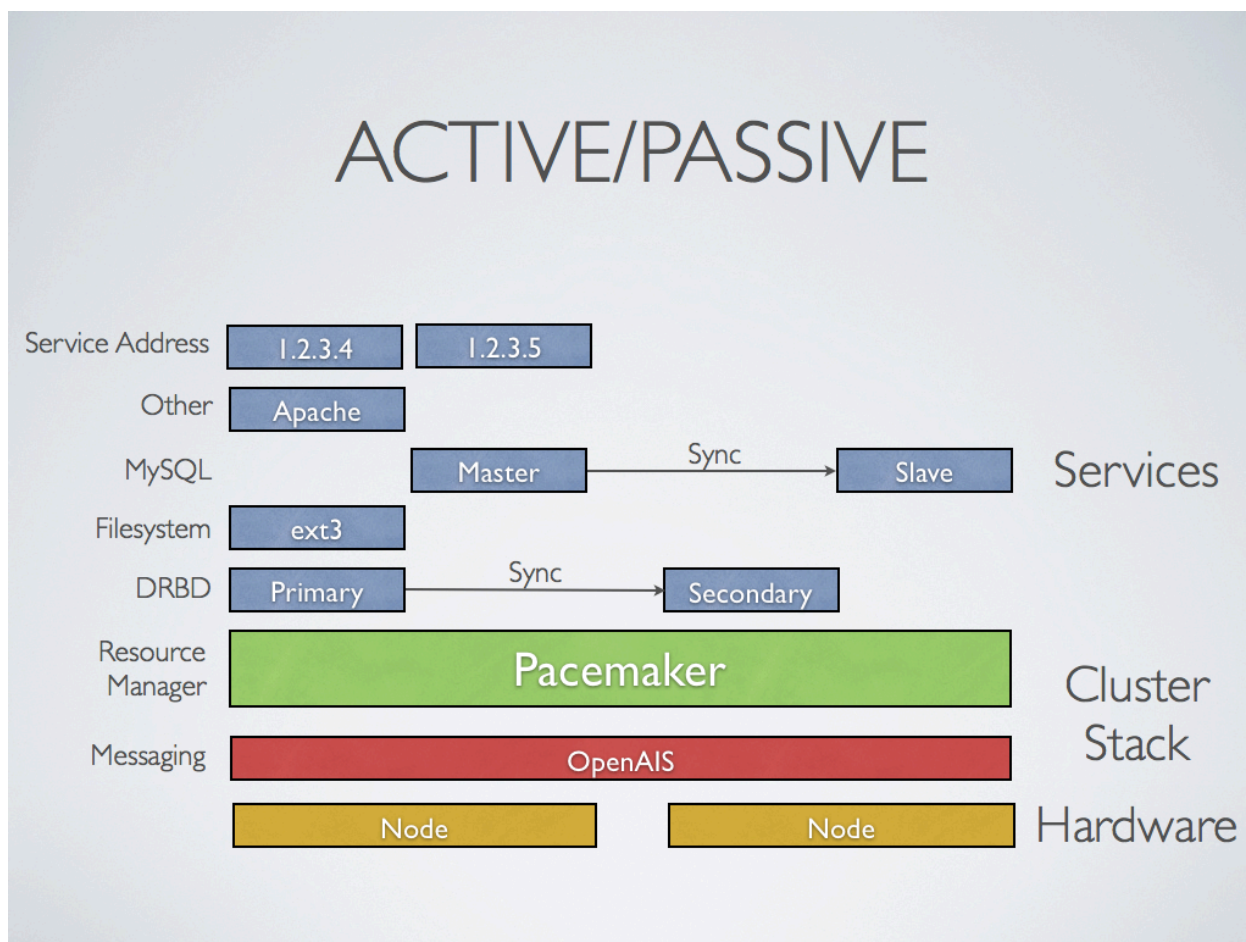
```
1 dn: ou=employee,dc=glb5,dc=tp,dc=org
2 ou: employee
3 objectClass: organizationalUnit
4 description: Employee of GLB5 Co LTD
5
6 dn: cn=Berthelot,ou=employee,dc=glb5,dc=tp,dc=org
7 objectClass: top
8 objectClass: person
9 objectClass: inetOrgPerson
10 objectClass: organizationalPerson
11 cn: Berthelot
12 sn: Berthelot
13 givenName: Joris
14
15 dn: cn=Le_Moine,ou=employee,dc=glb5,dc=tp,dc=org
16 objectClass: top
17 objectClass: person
18 objectClass: inetOrgPerson
19 objectClass: organizationalPerson
20 cn: Le_Moine
21 sn: Le_Moine
22 givenName: Laurent
```

0.4 Mise en place de Pacemaker et Corosync

Dans une architecture HA, le système doit pouvoir constamment suivre son état et celui de son entourage afin de décider si il doit basculer (= "failover") ou non vers un noeud de secours. Pour cela, avec Fedora, nous avons choisi d'utiliser le service Pacemaker couplé à Corosync car ils nous ont semblé très complets et très professionnels.

0.4.1 Présentation

Pour notre projet, nous avons choisi de mettre en place une architecture en grappes avec un système actif/passif. Ci-dessous un schéma de l'architecture :



Les « node » correspondent à nos machines (JB et LLM), la couche de messagerie OpenAIS correspondra au service Corosync qui est un projet dérivé d'OpenAIS. Pour les couches supérieures à Pacemaker, nous les avons vu précédemment.

Corosync est un moteur de clustering mettant en oeuvre une messagerie de service sur réseau en utilisant une adresse et un port multicast.

0.4.2 Installation

```
# Installation de pacemaker et de corosync
yum install -y pacemaker corosync
```

0.4.3 Configuration

0.4.3.1 Corosync

Avant toute chose, voici les adresses IP nécessaires :

Adresse IP multicast : 239.0.0.1

Port multicast : 6800

Réseau de la grappe : 10.192.10.0

Adresse IP de la grappe : 10.192.10.50

On va commencer par la configuration de Corosync qui est simpliste. Pour cela, on prends le fichier de configuration par défaut et on va simplement y changer les adresses IP comme ceci :

```
CONF="/etc/corosync/corosync.conf"
cp -f $CONF.example $CONF
sed -i.bak "s/.*mcastaddr:./mcastaddr:\ 239.0.0.1/g" $CONF
sed -i.bak "s/.*mcastport:./mcastport:\ 6800/g" $CONF
sed -i.bak "s/.*bindnetaddr:./bindnetaddr:\ 10.192.10.0/g" \ $CONF
```

Ensuite, on va signaler à Corosync de charger le plugin Pacemaker afin qu'il puisse "travailler" avec :

```
cat << EOT > /etc/corosync/service.d/pcmk
service {
    # Load the Pacemaker Cluster Resource Manager
    name: pacemaker
    ver: 1
}
EOT
```

Une fois configuré, nous pouvons lancer Corosync :

```
/etc/init.d/corosync start
```

0.4.3.2 Pacemaker

Pacemaker offre une invite de commande propre à lui-même (comme les switch Cisco) que nous pouvons lancer en tapant `crm`. Il est intéressant de savoir qu'en interne, la configuration de Pacemaker est formatée en XML car suite à quelques erreurs de notre part, nous avons pu constater des erreurs de validation d'entrée contre des Schémas XML.

Afin de vous épargner la configuration à la main de Pacemaker, nous pouvons exporter et injecter directement une configuration donnée :

```
crm configure load replace /path/to/conf/file
```

Mais avant cela, vous devez démarrer Pacemaker et vider sa configuration actuelle (si existante) :

```
/etc/init.d/pacemaker start
cibadmin -E --force
```

Voici donc notre fichier de configuration pour Pacemaker (la version initiale qui marchait correctement) :

```

1 primitive ClusterIP ocf:heartbeat:IPaddr2 params ip="10.192.10.50" \
2   cidr_netmask="24" op monitor interval="5s"
3 primitive FS ocf:heartbeat:Filesystem params \
4   device="/dev/drbd/by-res/ulr-data" \
5   directory="/var/cluster" fstype="ext4"
6 primitive UlrData ocf:linbit:drbd params drbd_resource="ulr-data" \
7   op monitor interval="30s"
8 primitive WebApp ocf:heartbeat:apache params \
9   configfile="/etc/httpd/conf/httpd.conf" op monitor interval="10s"
10 primitive MySQL lsb:mysql
11 primitive LDAP lsb:slapd
12 ms UlrDataClone UlrData meta master-max="1" master-node-max="1" \
13   clone-max="2" clone-node-max="1" notify="true"
14 colocation WebApp-with-FS inf: WebApp FS
15 colocation FS-with-UlrDataClone inf: FS UlrDataClone:Master
16 colocation WebApp-with-ClusterIP inf: WebApp ClusterIP
17 colocation MySQL-with-ClusterIP inf: MySQL ClusterIP
18 colocation LDAP-with-ClusterIP inf: LDAP ClusterIP
19 order MySQL-after-FS inf: FS MySQL
20 order LDAP-after-FS inf: FS LDAP
21 order WebApp-after-FS inf: FS WebApp
22 order WebApp-after-ClusterIP inf: ClusterIP WebApp
23 order FS-after-UlrDataClone inf: UlrDataClone:promote FS:start
24 property $id="cib-bootstrap-options" \
25   dc-version="1.1.6-1.fc14-b379478e0a66af52708f56d0302f50b6f13322bd" \
26   cluster-infrastructure="openais" \
27   expected-quorum-votes="2" \
28   stonith-enabled="false" \
29   no-quorum-policy="ignore"

```

Dans ce fichier, on peut y entrevoir des déclarations de ressources comme une IP “flottante” (10.192.10.50) sur laquelle le monde extérieur va se connecter aux services que contient la grappe, des ressources de battement de cœurs pour un système de fichier, les serveurs Apache, MySQL, LDAP et enfin la ressource de réplication bas niveau DRBD.

Mais ce n'est que la déclaration des ressources, ensuite viennent les dépendances de “colocation” qui obligent deux ressources données à toujours être de paire et enfin les ordres de démarrage des ressources en fonction des dépendances. Par exemple, Apache ne peut pas démarrer avant que le système de fichier soit opérationnel.

0.4.4 Utilisation

Pour ~~administrer~~ surveiller le bon fonctionnement de Pacemaker, il y a la commande `crm_mon`.

En ce qui concerne la gestion des nœuds ou des ressources, il faut utiliser la commande `crm`. Voici quelques exemples :

```
# Vérifier le status d'une ressource
crm resource status Apache
# Redémarrer une ressource
crm resource restart MySQL
# Déplacer une ressource sur un autre noeud
crm ressource move FS mambal4
# Voir le status d'un noeud
crm node status mambal3
# Mettre un noeud en standby (simulation du failover)
crm node standby mamb14
# Et le rendre disponible à nouveau
crm node online mambal4
```

Deuxième partie

Tests

0.5 Serveur de noms

0.6 Serveur Apache

0.7 Serveur MySQL

0.8 Serveur LDAP

Troisième partie

Conclusion

0.9 Difficultés rencontrés

Durant ce projet, la contrainte majeure était de réaliser une telle architecture en 3 jours seulement sans avoir aucune expérience dans le domaine. Nous avons souhaité nous orienter vers des choix fonctionnels et matures plutôt qu'universitaires et peu professionnels.

Le premier facteur de difficulté aura été l'absence d'indications concernant le choix des technologies car nous avons réellement été soumis à un environnement pragmatique où nous jouons le rôle des responsables SI vers qui on vient se conseiller suite une problématique bien présente. Le choix et la décision des technologies est une chose mais ce qui est le plus chronophage est de regarder les produits et les solutions existantes sur le marché.

Une fois les choix réalisés, il a fallu se documenter pour savoir implémenter au mieux possible l'infrastructure logicielle. Nous ne vous cacherons pas que nous avons lu 90% de documentation en anglais durant ces 3 jours. De l'anglais pas toujours facile à comprendre de part ses termes techniques mais surtout de part la notion nouvelle que nous découvrons sur le tas.

La difficulté suivante fut liée aux machines elles-même : impossible de prévoir que tout va fonctionner du premier coup, le fait que les machines aient besoin d'un serveur mandataire pour accéder au Web rajoute quelques contraintes sur les configurations mais surtout le manque de maîtrise complet sur les distributions Fedora nous aura fait perdre un peu de temps non négligeable.

Malgré tout cela, nous sommes parvenus automatiser de manière relativement complète l'installation, la configuration et le déploiement de notre projet à l'aide de Subversion et de scripts Bash.

0.10 Retours sur échec

Après 3 jours de recherche, de développement et de tests acharnés sur les différentes briques de notre architecture, nous avons échoué lors de la présentation de notre projet. En voici les raisons majeures :

1. Manque d'expérience avec les technologies utilisées
2. Manque d'organisation et de rigueur dans les procédures
3. Manque de temps (implique du stress)
4. Analyses sur tests défaillants pas assez complètes

D'une certaine manière, tous ces facteurs sont liés car ils s'entre-croisent et s'encouragent et nous le savions mais pas c'est pas toujours évident d'avoir le temps de prendre le recul nécessaire afin de mieux repartir.

Après des heures intensives de travail, nous avons réussi à faire tourner notre architecture active/passive en simulant des failovers et des failback avec succès mais sans tester tous les cas possibles et surtout les cas d'évaluation. Suite à ce succès précipité, nous avons voulu optimiser la configuration de Pacemaker afin de la simplifier mais nous sommes rendu compte après analyse que cette nouvelle configuration ne répondait plus aux contraintes d'ordre de basculement des services lors du failover.

Dans le stress et la précipitation, nous avons voulu essayer une architecture active/active car nous pensions que le type d'architecture active/passive ne répondait pas à la demande de l'évaluation (analyse d'échec pas assez poussée) mais près quelques heures d'installation et de configuration, cette nouvelle monture n'a pas été un succès à cause d'un soucis de synchronisation de volume avec DRBD bien que la nouvelle configuration active/active de Pacemaker était déjà prête.

0.11 Retours personnels

Quatrième partie

Annexes

.1 Bind

Fichier de configuration du serveur esclave :

```
1 options {
2     listen-on port 53 { ANY; };
3     listen-on-v6 port 53 { ANY; };
4     directory      "/var/named";
5     dump-file       "/var/named/data/cache_dump.db";
6     statistics-file "/var/named/data/named_stats.txt";
7     memstatistics-file "/var/named/data/named_mem_stats.txt";
8     allow-query     { ANY; };
9     recursion yes;
10
11     dnssec-enable yes;
12     dnssec-validation yes;
13     dnssec-lookaside auto;
14
15     version "Try again! ;)";
16
17     bindkeys-file "/etc/named.iscdlv.key";
18
19     managed-keys-directory "/var/named/dynamic";
20 };
21
22 logging {
23     channel default_debug {
24         file "data/named.run";
25         severity dynamic;
26     };
27 };
28
29 zone "." IN {
30     type hint;
31     file "named.ca";
32 };
33
34 zone "g1b5.tp.org" IN {
35     type slave;
36     file "g1b5.tp.org.zone";
37     masters { 10.0.0.24; };
38 };
39
40
41 zone "0.0.0.0.9.c.0.0.0.0.0.0.0.0.0.d.f.ip6.arpa" {
42     type slave;
43     file "rv6.g1b5.tp.org.zone";
44     masters { 10.0.0.24; };
45 };
46
47 zone "10.192.10.in-addr.arpa" {
48     type slave;
49     file "rv4.g1b5.tp.org.zone";
50     masters { 10.0.0.24; };
51 };
52
53 include "/etc/named.rfc1912.zones";
54 include "/etc/named.root.key";
```

Fichier de configuration de zone :

```

1 $ttl 3600
2 @      IN      SOA      ns.glb5.tp.org. admin.glb5.tp.org. (
3                               2011102001
4                               10800
5                               3600
6                               604800
7                               38400 )
8 ;
9 @      IN      NS       ns.glb5.tp.org.
10 @      IN      NS       ns2.glb5.tp.org.
11 ;
12 @      IN      A        10.192.10.50
13 ;
14 ns      IN      A        10.192.10.24
15 ns      IN      AAAA     fd00:0:c9::501
16 ;
17 ns2     IN      A        10.192.10.23
18 ns2     IN      AAAA     fd00:0:ca:4::502
19 ;
20 www     IN      A        10.192.10.50
21 ;
22 sql     IN      A        10.192.10.50
23 ;
24 ldap    IN      A        10.192.10.50

```

Fichier de configuration de zone inverse IPv4 :

```

1 ; 10.192.10.0/24
2 $TTL 3600
3 $ORIGIN 10.192.10.in-addr.arpa.
4 @      IN SOA @ admin.glb5.tp.org. (
5         2011102002      ; Serial number (YYYYMMdd)
6         10800           ; Refresh time
7         3600            ; Retry time
8         604800          ; Expire time
9         38400           ; Default TTL (bind 8 ignores this, bind 9 needs it)
10 )
11
12 ; Name server entries
13     IN      NS         ns.glb5.tp.org.
14     IN      NS         ns2.glb5.tp.org.
15 ; IPv4 PTR entries
16
17 ; Subnet #1
18 $ORIGIN 10.192.10.in-addr.arpa.
19
20 24     IN      PTR      ns.glb5.tp.org.
21 23     IN      PTR      ns2.glb5.tp.org.
22 50     IN      PTR      glb5.tp.org.
23 50     IN      PTR      sql.glb5.tp.org.
24 50     IN      PTR      ldap.glb5.tp.org.
25 50     IN      PTR      www.glb5.tp.org.

```

Fichier de configuration de zone inverse IPv6 :

```
1 ; fd00:0:c9::/64
2 $ORIGIN 0.0.0.0.9.c.0.0.0.0.0.0.d.f.ip6.arpa
3 $TTL 3600
4 @      IN SOA @ admin.glb5.tp.org. (
5         2011102003      ; Serial number (YYYYMMdd)
6         10800           ; Refresh time
7         3600            ; Retry time
8         604800          ; Expire time
9         38400           ; Default TTL (bind 8 ignores this, bind 9 needs it)
10 )
11
12 ; Name server entries
13      IN      NS      ns.glb5.tp.org.
14      IN      NS      ns2.glb5.tp.org.
15 ; IPv6 PTR entries
16 501      IN   PTR      ns.glb5.tp.org.
17 502      IN   PTR      ns2.glb5.tp.org.
```