Reverse-proxy & cache

> What's wrong with that ?
>
> Le dixième docteur

1 Définitions

- **reverse-proxy** : relaye les requêtes sur un ou plusieurs chemins vers un ou plusieurs autres serveurs définis, le plus souvent internes ;
- **forward-proxy** ou **serveur proxy** : sert d'intermédiaire entre deux hôtes (le plus souvent, entre des hôtes et Internet) pour faciliter (cache par ex.) ou surveiller (logs, proxy filtrant) les échanges. Nous ne traiterons pas ce point : d'autres logiciels spécialisés existent pour faire cela et sont plus simples et plus efficaces ;
- **cache** : mécanisme permettant de "sauvegarder" les résultats de requêtes pour, si le cache n'est pas trop ancien, les servir plus rapidement, sans relayer la requête à nouveau.

Si Squid est un des proxy-caches les plus connus et les plus utilisés en *forward-proxy*, Apache peut être utilisé à l'identique sur des configurations qui restent "simples" mais très fonctionnelles.

2 Reverse-proxy

2.1 Reverse-proxy avec Apache

L'utilisation la plus courante de `mod_proxy` se fait sur un serveur frontal [1] en répartissant les charges dynamiques et statiques sur des « fermes [2] ». Par exemple, on redirige `/images/` et `/media/` sur une ferme statique avec peut-être un autre serveur Web plus approprié (nginx par exemple).

Apache joue ainsi le rôle de proxy et distribue la charge sur plusieurs serveurs, l'appellation courante est *reverse-proxy*.

On peut coupler le rôle de proxy avec un mécanisme de cache afin d'améliorer les performances en réduisant le nombre d'appels aux serveurs sous-jacents.

Vous noterez dans la documentation du module [3] que sont évoqués plusieurs sous-modules du module de cache, correspondant au différents fournisseurs de stockage pour le cache [4] (disque [5], à base de cache d'objets partagés [6]) ainsi que des sous-modules pour les différents protocoles pouvant potentiellement profiter d'un système de cache (ftp [7], http [8], etc.).

La commande `ProxyPass` permet d'associer un chemin (ou `Location`) à un (groupe de) serveur(s) tiers, exemple :

```
ProxyRequests Off
ProxyPass / http://10.0.0.1/
ProxyPass /images/ http://10.0.0.10/
ProxyPass /medias/ http://10.0.0.20/
```

1. Un serveur qui encaisse les requêtes entrantes et les transmet à un ou plusieurs autres serveurs
2. groupes de serveurs applicatifs fournissant le même service de manière équivalente et distribuable
3. http://httpd.apache.org/docs/2.4/fr/mod/mod_proxy.html
4. http://httpd.apache.org/docs/2.4/fr/mod/mod_cache.html
5. http://httpd.apache.org/docs/2.4/mod/mod_cache_disk.html
6. http://httpd.apache.org/docs/2.4/mod/mod_cache_socache.html
7. http://httpd.apache.org/docs/2.4/fr/mod/mod_proxy_ftp.html
8. http://httpd.apache.org/docs/2.4/fr/mod/mod_proxy_http.html

La directive `ProxyRequests` sert à activer les fonctionnalités de *forward-proxy* d'Apache, ce dont nous n'avons pas besoin ici, ce qui explique sa désactivation en lui passant le paramètre `Off` ;

Les requêtes pour l'URL `/` seront relayées au serveur possédant l'IP 10.0.0.1, etc.

Ici, en utilisant le sous-module `mod_proxy_balancer`[9], on distribue sur deux fermes de deux machines les parties statiques :

```
ProxyRequests Off
ProxyPass / http://10.0.0.1/
<Proxy balancer://clusterImages>
    BalancerMember http://10.0.0.10:80
    BalancerMember http://10.0.0.11:80
</Proxy>
<Proxy balancer://clusterMedias>
    BalancerMember http://10.0.0.20:80
    BalancerMember http://10.0.0.21:80
</Proxy>
ProxyPass /images/ balancer://clusterImages/
ProxyPass /medias/ balancer://clusterMedias/
```

Pour une gestion distante du sous-module `mod_proxy_balancer` on peut rajouter (attention à la sécurité[10] !) :

```
<Location /balancer-manager>
    SetHandler balancer-manager
    Require ip 10.0.0.1
</Location>
```

2.2 Reverse-proxy avec Nginx

On se servira ici de la directive `proxy_pass`.

Dans la configuration de votre *virtualhost* :

```
location / {
    include proxy_params;
    proxy_pass http://127.0.0.1;
}
```

Pour distribuer la charge sur plusieurs machines, on utilise la directive `upstream` **en dehors** de la configuration du *virtualhost*, donc du `server {...}` :

```
upstream loadbalancing {
    server 10.0.0.1;
    server 10.0.0.2;
}
```

Et on utilisera l'*upstream* ainsi défini dans le *virtualhost* :

```
location / {
    include proxy_params;
    proxy_pass http://loadbalancing;
}
```

9. http://httpd.apache.org/docs/2.4/fr/mod/mod_proxy_balancer.html
10. ne laissez pas tout le monde accéder à quelque chose qui peut modifier le comportement de votre serveur

3 Cache

3.1 Apache

On l'a vu plus haut, il existe différents sous-modules de cache. Il vous faudra en activer un pour bénéficier des mécanismes de cache d'Apache.

Prenons l'exemple de `mod_cache_disk` :

```
a2enmod cache_disk
systemctl restart apache2
systemctl start apache-htcacheclean
```

Dans votre *virtualhost* :

```
CacheQuickHandler off
<Location />
    CacheEnable disk
</Location>
```

La directive `CacheQuickHandler` est généralement à positionner à `off`. En effet, si elle est à `on`, sa valeur par défaut, les directives de restrictions d'accès (`Require`) ne seront pas appliquées.

On peut aussi spécifier l'URL pour laquelle il faut activer le cache directement en argument de la directive `CacheEnable` :

```
CacheEnable disk /foo/
```

3.2 Nginx

NB : Nginx ne peut cacher que sur disque, pas en mémoire. On peut néanmoins tricher en le faisant cacher sur un montage *tmpfs*[11] ou en mamaillant avec `memcached`.

Définissez un espace de cache en **dehors** de votre *virtualhost*, donc du `server {...}` :

```
proxy_cache_path /var/cache/nginx keys_zone=my_zone:10m inactive=7d max_size=700m;
```

De nombreux arguments[12] sont utilisables avec cette directive mais nous n'en verrons que quelques-uns (la documentation du module vous tend les bras).

Notons d'abord les indispensables :

- `/var/cache/nginx` : il s'agit du chemin du répertoire de cache (donc bien évidemment complètement modifiable) ;
- `keys_zone=name:size` : le nom de la zone de cache. Nginx permet donc de définir différents caches, réutilisables à loisir dans différents endroits de la configuration. La taille de cet argument n'est pas la taille du cache, mais la taille de la mémoire allouée pour conserver les métadonnées des données cachées.

Les arguments optionnels :

- `inactive=time` : les données cachées non consultées durant ce délai seront supprimées du cache ;
- `max_size=size` : il s'agit de la taille maximale du dossier de cache. Lorsque cette taille est dépassée, les données les plus anciennes sont supprimées pour faire de la place aux nouvelles.

11. https://fr.wikipedia.org/wiki/Tmpfs
12. http://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_cache_path

On utilisera ensuite cette zone de cache ainsi dans un *virtualhost* (ou juste dans le location, voire en dehors du *virtualhost*) :

```
location / {
    proxy_cache my_zone;
    ...
}
```

3.3 En-têtes

Il n'y a pas que le cache côté serveur qui améliore les performances, il y a aussi le cache côté client !

Si certaines applications se chargent de renvoyer toutes seules des en-têtes contenant des instructions pour le cache client, il est parfois necéssaire de les mettre soi-même.

Avec Apache, on pourra utiliser indifféremment le module `mod_expires` [13] ou le module `mod_headers` [14] :

```
<Directory /var/www/site/css/>
    ExpiresActive On
    ExpiresDefault "access plus 1 month"
</Directory>

<Directory /var/www/site/js>
    Header add Cache-Control "public, max-age=2678400"
</Directory>
```

Avec Nginx, c'est semblable, on utilisera la directive `expires` ou `add_header` :

```
location /css {
    expires 1M;
}
location /js {
    add_header Cache-Control "public, max-age=2678400";
}
```

4 Varnish

Varnish est un reverse-proxy cachant dont les performances sont impressionnantes [15] et dont la mise en œuvre est fort simple pour une utilisation basique.

Installez Varnish, configurez-le pour qu'il utilise votre serveur Apache (voir dans le fichier `/etc/varnish/default.vcl`, changez le port du backend pour le port `80`) et redémarrez-le.

Alourdissez la page servie par votre serveur sur `http://localhost/` avec la commande `truncate` :

```
truncate -s 500M /var/www/html/index.html
```

Lancez les commandes suivantes :

```
wget http://localhost/ -O /tmp/index.html
wget http://localhost:6081/ -O /tmp/index.html
wget http://localhost:6081/ -O /tmp/index.html
```

13. http://httpd.apache.org/docs/current/mod/mod_expires.html
14. http://httpd.apache.org/docs/current/mod/mod_headers.html
15. true story

Vous constaterez aisément que *Varnish* (qui écoute par défaut sur le port 6081) améliore grandement les performances tout en réduisant l'utilisation des ressources de votre machine, sauf bien évidemment lors de sa première requête puisqu'il n'a pas encore le fichier en cache.

Pro tips :

— pour vérifier la configuration de Varnish avant de le relancer, exécutez `varnishd -Cf /etc/varnish/default.vcl` ;
— Varnish peut aussi faire du *load balancing* : https://varnish-cache.org/docs/trunk/users-guide/vcl-backends.html#directors

NB : Varnish cache par défaut en mémoire, mais il peut aussi cacher sur disque.

5 Miscellanées

— Ce n'est parce que Nginx est plus véloce dans notre cas, même sans cache, qu'il faut l'employer partout. Le fait de pouvoir étendre les capacités d'Apache de façon simple et dynamique grâce aux modules en font un allié précieux dans bien des cas. « Tout ressemble à un clou pour qui ne possède qu'un marteau ». Vous avez plusieurs outils à votre disposition, faites-en bon usage pour la bonne raison.
Un montage pertinent et une configuration idoine vous permettront d'encaisser des charges bien plus importantes, voire carrément impressionnantes [16] !
— Varnish déchire pour les performances, c'est un fait. Mais ça n'est qu'un reverse-proxy. Il ne fera pas de php, de FastCGI, etc. De plus, sa configuration n'est pas forcément la plus simple à prendre en main pour des usages avancés.
— Le *load balancing* ou l'envoi de requêtes sur un serveur plus performant n'est pas la seule raison d'utilisation d'un proxy. Certaines applications, comme celles écrites avec le framework Perl Mojolicious [17] comme Lstu [18] sont servies par un serveur web dédié à ce genre d'application. La façon la plus simple de les utiliser est de les proxyfier.
— Un proxy cache n'est pas forcément destiné à servir de serveur frontal. Ainsi, l'application Lutim, sur son instance officielle https://lut.im était, avant le développement d'un système de cache interne, servie de la façon suivante :
Nginx -> Varnish -> Hypnotoad (le serveur web dédié qui fournit Lutim)
Ceci afin de mettre en cache en mémoire les images renvoyées par Lutim, mais certaines URLs étaient fournies directement par Nginx (css, js, etc).
— Certaines pages sont tellement dynamiques qu'il ne sert à rien de vouloir les mettre en cache (cours de la bourse, etc.).
— Nginx est un très bon *load balancer*
— Attention aux logs ! Le serveur de la ferme loguera l'IP du frontal si vous n'adaptez pas la configuration. Différents modules ou techniques sont à votre disposition pour régler ce problème :
  — `mod_remoteip` [19] pour Apache 2.4 et `mod_rpaf` pour Apache 2.2 [20] ;
  — l'utilisation de l'en-tête `X-FORWARDED-FOR` ;
  — le module `ngx_http_realip_module` pour Nginx [21].

16. il ne faut pas non plus négliger les possibilités de caching des applications servies. Un bon plugin sur Wordpress peut tout changer
17. https://mojolicious.org
18. raccourcisseur d'URLs, https://lstu.fr
19. https://httpd.apache.org/docs/2.4/fr/mod/mod_remoteip.html
20. https://github.com/gnif/mod_rpaf
21. http://nginx.org/en/docs/http/ngx_http_realip_module.html