Fonctionnement de Squid

La première activité de Squid est de relayer les requêtes clientes et de mettre en cache la réponse si celle-ci est éligible au cache. Pour savoir si la réponse peut être mise en cache, Squid analyse les données présentes dans l'en-tête HTTP.
Voici les règles qui sont suivies pour décider de la mise en cache :

- Si des instructions interdisant la mise en cache sont rencontrées (Cache-Control: private) alors la ressource n'est pas mise en cache.
- Si la connexion est sécurisée ou authentifiée alors la ressource n'est pas mise en cache.
- Si aucune directive de validation n'est présente (Last-Modified ou Etag[1]) alors la ressource n'est pas mise en cache.
- Sinon la ressource est considérée comme éligible au cache et Squid décide de la stocker.

Si Squid reçoit une requête qui a déjà donné lieu à la mise en cache de la ressource, celui-ci va valider la donnée afin de savoir si elle est assez récente et si elle peut être renvoyée à l'utilisateur en l'état ou bien si au contraire, elle est considérée comme trop ancienne et nécessite d'être redemandée au serveur d'origine.

Cette étape de validation s'appuie sur les directives d'expiration et de validation contenues dans l'en-tête HTTP (Expires, max-age, Last-Modified et Etag).
Le serveur procède de la sorte :

- Vérification de la validité de la ressource en cache à l'aide des directives d'expiration (Expires et max-age). Cette étape consiste en un calcul de date aboutissant à la péremption ou non d'un objet du cache. Si ces directives ne sont pas présentes, la réponse est considérée comme étant valide à tout moment.
- Si la ressource est considérée comme encore valide, elle est renvoyée au poste client sans contacter le serveur source.
- Si la réponse est périmée, une requête conditionnelle (If-Modified-Since ou If-None-Match) basée sur les directives Last-Modified ou Etag est envoyée au serveur source qui répond soit en envoyant la nouvelle ressource soit en envoyant un code HTTP 304-Not Modified.

La directive principale de cache dans la version 1.1 du protocole HTTP est la directive Cache-Control. Elle peut prendre plusieurs options qui influent directement sur le comportement du serveur cache. La directive de cache HTTP peut aussi bien être présente dans la requête que dans la réponse. Utilisée dans la requête, elle permet au poste client de piloter certains aspects du serveur cache en signifiant par exemple de ne retourner une réponse que si celle-ci est dans le cache. Dans la réponse, c'est le serveur source qui pilote le comportement du serveur cache.
Dans une requête adressée par un utilisateur, la directive Cache-Control peut prendre les valeurs suivantes :

- no-cache : cette option va forcer le serveur à recharger la page auprès du serveur d'origine.
- max-age : utilisée dans la requête du poste client, cette option permet de fournir une valeur (en secondes) correspondant à l'âge maximum accepté pour la ressource retournée par le proxy cache. Sauf si l'option max-stale est également fournie, cela signifie que le poste client ne désire pas recevoir de réponse périmée.
- max-stale : permet d'indiquer au serveur de cache que le poste client est prêt à accepter une réponse éventuellement périmée mais dont le délai d'expiration n'excède toutefois pas la valeur passée en paramètre. Si aucune valeur n'est passée, toute réponse périmée est acceptée par le poste client.
- min-fresh : permet d'indiquer au serveur de cache que le poste client est prêt à accepter toute réponse qui sera encore considérée comme valide dans la période passée en paramètre. A la réception de ce type de requête, si la ressource est présente dans le cache, le serveur additionne la valeur de l'option min-fresh à l'âge de la ressource et regarde si le total est inférieur à la valeur de l'option max-age. Si c'est le cas, la ressource est renvoyée au poste client, sinon, le serveur source est contacté.

[1] La balise-entité ETag HTTP est une partie du protocole de communication HTTP utilisée pour la validation du cache.
Un ETag est un identifiant unique opaque assigné par le serveur web à chaque version d'une ressource accessible via une URL. Si la ressource accessible via cette URL change, un nouvel ETag différent du précédent sera assigné. Utilisés ainsi, les ETags sont similaires à des empreintes digitales, et peuvent être rapidement comparés pour vérifier si deux versions sont identiques, et ainsi savoir si une demande peut être honorée par un cache local ou pas (source Wikipédia).

- only-if-cached : cette option permet de récupérer dans le cache une ressource uniquement si celle-ci est déjà présente dans le cache. Si ce n'est pas le cas, une réponse HTTP 504-Gateway timeout est renvoyée au poste client.

Au sein d'une réponse serveur, les valeurs possibles sont les suivantes :

- public : permet d'indiquer aux serveurs de cache que la réponse est éligible à la mise en cache (valable aussi bien pour les caches partagés que les caches privés).
- private : cette option permet d'indiquer au serveur de cache que tout ou partie de la réponse est destiné à un unique utilisateur et que par conséquent la réponse ne doit pas être mise en cache.
- no-cache : indique que la ressource doit être revalidée chaque fois qu'elle est redemandée.
- no-store : cette option permet de ne pas stocker la réponse sur le disque. La réponse peut être mise en cache mais ne doit pas être enregistrée sur des volumes de stockage non volatiles.
- no-transform : cette option permet d'empêcher le serveur cache de modifier le paquet HTTP, que ce soit l'en-tête HTTP ou le contenu lui-même. Certains serveurs de cache convertissent ou compressent les données pour économiser de l'espace mémoire.
- must-revalidate : cette option peut être utilisée par les serveurs source pour forcer la validation des données mises en cache.
- proxy-revalidate : cette option permet de réaliser la même chose que l'option must-revalidate à ceci près qu'elle n'est utilisée que par les serveurs de cache.
- max-age : cette option sert de validateur et implique que la ressource est éligible au cache. Lorsque cette option a pour valeur « -1 », elle équivaut à l'option no-cache et indique au serveur qu'elle ne doit pas être stockée. Si la directive Expires (HTTP/1.0) est également présente dans l'en-tête HTTP, l'option max-age (HTTP/1.1) est utilisée en priorité.
- s-maxage : cette option permet de combiner les propriétés des options max-age et must-revalidate. Le paramètre fourni permet de contrôler la date d'expiration de la ressource envoyée par le serveur et de forcer la revalidation par le serveur cache. Cette option surcharge les options Expires et max-age.

Les **Etags** (Entity Tag) sont arrivés avec la version 1.1 du protocole HTTP. Ils ont été pensés afin d'améliorer le processus de validation qui était auparavant basé sur l'option Last-Modified pour envoyer une requête IMS (If-Modified-Since) au serveur source. Un Etag est en fait une empreinte (ou hash) qui est associée à une ressource et qui permet de l'identifier de manière plus ou moins unique. Celui-ci est généré par le serveur source et apporte un autre mécanisme de détection des modifications basé sur l'utilisation des directives HTTP If-Match et If-None-Match.

Remarque : Dans la version 3 de Squid, le support des Etags n'est pas encore complet.

Pour vérifier la validité d'une ressource en cache, Squid utilise trois types de variables : des variables correspondant à l'objet enregistré, une variable fournie par le client et des variables du fichier de configuration.
Les variables associées à l'objet enregistré sont calculées d'après les données relatives à l'enregistrement de l'objet dans le cache :

- AGE correspond au temps écoulé depuis son entrée dans le cache.
- LM_AGE correspond au temps écoulé entre la dernière modification de l'objet sur le serveur d'origine et son entrée dans le cache.
- LM_FACTOR est le rapport entre AGE et LM_AGE. Plus ce facteur est grand, plus l'objet a de chances d'être périmé.
- EXPIRES est la valeur éventuellement fournie par le serveur au moment de l'entrée de l'objet dans le cache. Comme son nom l'indique, cette variable contient la date d'expiration de l'objet.

Squid tient également compte de la variable de contrôle de cache CLIENT_MAX_AGE éventuellement fournie par le client, s'il utilise la version 1.1 d'HTTP. Cette variable indique l'âge maximal de l'objet accepté par le client.

Les variables du fichier de configuration sont fixées par la directive refresh_pattern qui permet gérer le rafraichissement du cache. Sa syntaxe est la suivante :

```
refresh_pattern [-i] <url_regexp> MIN_AGE PERCENT MAX_AGE <options>
```

Le drapeau optionnel -i permet de ne pas tenir compte de la casse des caractères dans l'expression régulière de description de l'URL. Il peut y avoir plusieurs directives refresh_pattern dans le chier de configuration.

Les variables qui permettent de régler le comportement du rafraichissement du cache sont :

- MIN_AGE indique la durée en minutes pour laquelle un objet sans expiration explicite doit être conservé
- PERCENT représente le pourcentage de l'objet (temps depuis sa dernière modification)
- MAX_AGE indique la durée maximale en minutes d'un objet sans expiration explicite

Extrait du fichier squid.conf :

```
...
refresh_pattern ^ftp:        1440    20%     10080
refresh_pattern ^gopher:     1440    0%      1440
refresh_pattern -i (/cgi-bin/|\?) 0 0%      0
refresh_pattern (Release|Packages(.gz)*)$ 0       20%     2880
refresh_pattern .            0       20%     4320
```

Ici, les valeurs de MAX_AGE, PERCENT et MIN_AGE pour les requêtes HTTP sont donc respectivement 0, 20% et 4320 (unité de temps en minutes).

La variable MAX_AGE éventuellement fournie par le client prévaut sur celle du fichier de configuration. Par contre, MIN_AGE du fichier de configuration est prioritaire devant la variable EXPIRES éventuellement fourni par le serveur.

Une fois les variables nécessaires déterminées, Squid observe les règles suivantes pour déterminer la fraîcheur de l'objet dans le cache :

- Si MAX_AGE est défini et que AGE > MAX_AGE, l'objet est déclaré « périmé »
- Si EXPIRES est défini et dépassé, l'objet est déclaré « périmé »
- Si AGE <= à MIN_AGE, l'objet est déclaré « frais »
- Si EXPIRES est défini mais pas encore dépassé, l'objet est déclaré « frais »
- Si LM_FACTOR est inférieur à PERCENT, l'objet est déclaré « frais »
- Sinon celui-ci est déclaré « périmé ».

```
// Exemple pour les pages web : une fois par jour pendant 7 jours
refresh_pattern ^http 1440 20% 10080
// ou pour les pages web (html) :
refresh_pattern -i \.html$ 1440 20% 10080
```

Squid effectue des requêtes DNS qui sont « bloquantes ». Il démarre ainsi un certain nombre de processus pour répondre à ces requêtes. On spécifie leur nombre avec la directive dns_children. Si cette requête réussit, elle est placée en cache pendant le temps précisé par la directive positive_dns. Dans le cas contraire, le temps sera celui fixé par la directive negative_dns.

```
# cat /etc/squid3/squid.conf.bak | grep "dns_"
positive_dns_ttl 6 hours
negative_dns_ttl 1 minutes
dns_children 5
```

Il est possible d'ajouter avec la directive dns_nameservers des serveurs de noms DNS qui seront utilisés par Squid pour résoudre les noms de domaine correspondant à des demandes émanant des clients :

```
// Exemple :
# cat /etc/squid3/squid.conf.bak | grep "dns_nameservers"
dns_nameservers 10.0.0.1 192.172.0.4
```

Sources : RFC 2616 http://abcdrfc.free.fr/rfc-vf/pdf/rfc2616.pdf

Squid – Edition ENI
http://www.squid-cache.org/Doc/