Comprimer les données qu'envoie le serveur HTTP Apache

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 mai 2012. Dernière mise à jour le 11 juin 2012

http://www.bortzmeyer.org/gzip-compression-apache.html

Faut-il comprimer les fichiers qu'Apache envoie au navigateur Web ? Il y a du pour et du contre et la réponse dépend de pas mal de facteurs. En tout cas, après quelques tests, j'ai augmenté l'usage de la compression sur ce blog. Voici pourquoi et comment.

Le protocole HTTP permet à un client HTTP d'indiquer qu'il sait gérer les données comprimées, avec l'en-tête `Accept-Encoding:` (RFC 7231 [1], section 5.3.4). Lorsque le serveur HTTP reçoit cet en-tête, il peut comprimer les données avant de les envoyer.

Mais est-ce rentable ? Il y a des données qui sont déjà comprimées (les images, en général) et il y a des cas où la capacité du réseau n'est pas le principal **goulet d'étranglement**, c'est souvent la latence qui est le facteur limitant.

Parmi les excellents outils de l'excellent service Google Webmasters Tools <http://www.google.com/webmasters/>, on trouve, dans la rubrique "*Labs*" -¿ "*Site Performance*", le conseil de comprimer. Si on ne le fait pas, Google dit « "*Enable gzip compression \ Compressing the following resources with gzip could reduce their transfer size by 22.7 KB :*" » et menace de rétrograder le site dans son classement. (Sur l'ensemble des conseils techniques de Google, vous pouvez évidemment consulter leur site, ou bien, en français, « Les critères de performance web pour le référencement Google <http://www.1ere-position.fr/blog/les-criteres-de-performance-web-pour-le-referencement-google> ».)

Bon, avant de faire comme le demande Google, on va mesurer. Prenons curl et lançons-le sur une page de grande taille (779 481 octets sur le disque). On utilise l'excellente option `--write-out` de curl qui permet de n'afficher que les informations qu'on choisit :

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc7231.txt`

```
% curl --silent \
    --write-out "size = %{size_download} bytes time = %{time_total} seconds speed = %{speed_download} bytes/
    --output /dev/null http://www-remote.example.org/feed-full.atom
size = 779481 bytes time = 1.142 seconds speed = 682361.000 bytes/second
```

On a bien récupéré tous nos octets. Activons maintenant la compression sur le serveur HTTP (un Apache ; j'explique plus tard dans l'article comment c'était fait). Et disons à curl de la demander, grâce à l'option `--header` pour demander l'ajout d'un `Accept-Encoding:` à la requête. La valeur de ce champ est le nom du ou des protocoles de compression acceptés, ici gzip et deflate :

```
% curl --header 'Accept-Encoding: gzip,deflate' --silent \
    --write-out "size = %{size_download} bytes time = %{time_total} seconds speed = %{speed_download} bytes/
    --output /dev/null http://www-remote.example.org/feed-full.atom
size = 238471 bytes time = 0.744 seconds speed = 320385.000 bytes/second
```

On a réduit la quantité de données de 69 %. La vitesse de transfert a **diminué** (la compression et la décompression prennent du temps) mais c'est compensé par la réduction de taille : le transfert prend 34 % de temps en moins, ce qui était le but. (Dans la réalité, il faut répéter la mesure plusieurs fois, sur une longue période, pour s'assurer qu'il n'y a pas d'autres phénomènes en jeu, comme la charge du réseau.)

Notez que, parmi les variables que curl permet d'afficher, `time_total` inclut des temps qui ne dépendent pas de la compression comme la résolution de noms. Il n'y a pas de variable `time_transfer` qui donnerait le temps qui nous intéresse directement. Il faudrait sans doute afficher `time_total` et `time_starttransfer` puis faire la soustraction.

Dans le cas précédent, le serveur HTTP était sur un autre continent que la machine qui lançait curl. Avec un serveur proche, l'écart est moins net :

```
# Sans compression
% curl --silent \
    --write-out "size = %{size_download} bytes time = %{time_total} seconds speed = %{speed_download} bytes/
    --output /dev/null http://www-close.example.org/feed-full.atom
size = 779481 bytes time = 0.215 seconds speed = 3626353.000 bytes/second

# Avec compression
% curl --header 'Accept-Encoding: gzip,deflate' --silent \
    --write-out "size = %{size_download} bytes time = %{time_total} seconds speed = %{speed_download} bytes/
    --output /dev/null http://www-close.example.org/feed-full.atom
size = 238471 bytes time = 0.162 seconds speed = 1472215.000 bytes/second
```

Bref, la compression est utile mais ne fait pas de miracles.

À noter que la compression peut présenter un autre avantage, celui de diminuer le débit sur le réseau, ce qui peut être utile, par exemple si vous êtes facturé au volume, ou tout simplement si vous voulez épargner une ressource partagée. Voici, sur ce blog, le résultat de l'activation (le mercredi au milieu du graphe) de la compression des flux de syndication : Deux semaines après, on a une vision plus large, voici la baisse du trafic réseau (semaine 21) : Et, logiquement, l'augmentation de la consommation de processeur (eh oui, on n'a rien sans rien) :

Florian Maury fait remarquer à juste titre que les tests ci-dessus avec curl ont une limite : ils ne testent pas le temps de décompression chez le client. Peut-être faudrait-il faire :

---

http://www.bortzmeyer.org/gzip-compression-apache.html

```
% curl --header 'Accept-Encoding: gzip,deflate' $URL | gunzip -
```

Pensez aussi que, pour tester la différence, il faut un client HTTP qui accepte la compression. C'est pour cela que je n'ai pas utilisé un programme comme echoping <http://echoping.sourceforge.net/>, qui ne permet pas d'envoyer l'en-tête `Accept-Encoding:`.

Sinon, si on préfère tester avec le classique outil de "*benchmark*" "*ApacheBench*" (`ab`), il faut lui dire explicitement d'activer la compression avec l'option `-H "Accept-Encoding: gzip,deflate"`. (Voir aussi un article plus détaillé sur ApacheBench <http://www.ffnn.nl/pages/articles/linux/apache-2-mod_deflate-benchmark.php>.)

Bon, et comment j'ai configuré Apache ? Le module le plus courant est le module standard mod_deflate <http://httpd.apache.org/docs/2.2/mod/mod_deflate.html> et est très bien documenté. Sur une Debian, il suffit de `a2enmod deflate && /etc/init.d/apache2 reload` pour l'activer. Toutefois, sa configuration par défaut n'inclut pas tous les fichiers, notamment pour éviter de comprimer les images (qui le sont déjà). Par défaut, il comprime le HTML mais j'ai ajouté Atom, les flux de syndication étant souvent de grande taille. Voici mon `deflate.conf` :

```
AddOutputFilterByType DEFLATE text/html text/plain text/xml
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE application/x-javascript application/javascript application/ecmascript
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/atom+xml

# Maximum compression. Not many changes noticed when changing it.
DeflateCompressionLevel 9
```

Ah, au fait, faut-il utiliser deflate ou gzip ? Comme le rappelle le RFC 2616, « *"[deflate is] The "zlib" format defined in in combination with the "deflate" compression mechanism described in ."* » gzip est simplement l'algorithme de deflate avec des métadonnées en plus. Il ne devrait donc pas y avoir de différences sensibles de performance.

Une alternative à mod_deflate, que je n'ai pas testée, est le mod_pagespeed <http://googlewebmastercentral.blogspot.com/2010/11/make-your-websites-run-faster.html> de Google (qui fait bien d'autres choses que comprimer). Autre alternative (signalée par Patrick Mevzek), « dans le cas des sites assez statiques et automatisés, générer à la fois le `.html` et le `.html.gz`. On perd en espace disque forcément, mais zéro temps de compression par Apache, il ne fait que servir selon le `Accept-Encoding:` et pas besoin de module spécifique. [Pour les] sites qui sont générés à partir d'un Makefile qui se charge donc de créer les versions compressées en même temps. »

Lectures supplémentaires ? Un court article sur l'utilisation de curl <http://www.if-not-true-then-false.com/2010/curl-tip-check-that-the-apache-compression-gzip-deflate-is-working/> pour tester la compression.