

Sommaire

| | |
|--|----------|
| Optimisez votre système Linux..... | 1 |
| Considérations générales..... | 1 |
| Parallélisme..... | 1 |
| Le tuning du système..... | 2 |
| La mémoire virtuelle..... | 2 |
| Le démon noyau bdflush..... | 2 |
| Pourcentage de la mémoire allouée pour le cache..... | 3 |
| Anticiper l'accès aux pages mémoires..... | 3 |
| Le système de fichiers..... | 3 |
| Optimiser les performances des disques IDE..... | 3 |
| Tuner le nombre maximum de fichiers et d'inodes..... | 3 |
| Tuner le nombre de superblocks..... | 4 |
| Optimisez votre partition de swap..... | 4 |
| Comment récupérer 5% d'espace disque sur une partition ext2..... | 4 |
| Changer la taille des blocs d'un système de fichiers..... | 4 |
| Ajuster le comportement du swap..... | 4 |
| Activer l'option noatime..... | 5 |
| Le réseau..... | 5 |
| Améliorer les performances NFS..... | 5 |
| Tuner les connexions TCP/IP..... | 5 |
| Tuner le serveur Samba..... | 5 |
| Tampon cache TCP/IP..... | 5 |
| Divers..... | 5 |
| Améliorer les performances d'un système SMP..... | 6 |
| Taille maximale d'un segment de mémoire partagé..... | 6 |
| Tuner les niveaux d'interruptions..... | 6 |
| Augmenter la stack size..... | 6 |
| Limite de processus par utilisateur..... | 6 |
| Nombre maximum de processus..... | 6 |
| Glossaire..... | 6 |
| Copyright..... | 8 |

Optimisez votre système Linux

Optimisez votre système Linux

par Lionel Tricon

Quelques recettes pour optimiser votre système Linux.

Considérations générales

La première étape, pour obtenir de meilleures performances sous Linux, consiste à vérifier s'il n'est pas possible d'améliorer certains composants matériels (*hardware*) du système. Augmenter la mémoire vive (un système de type Unix est toujours friand en mémoire), changer pour des disques dur SCSI, des disques IDE UltraDMA100, bien dimensionner la carte réseau (10 MB ? 100MB ? Plus encore ?), opter éventuellement pour du SMP et choisir un processeur performant et adapté : tout cela fait partie du processus qui va vous amener à bien configurer votre système.

Si les accès disques sont un élément déterminant pour la performance de vos applications, vous pourrez envisager d'utiliser du Raid logiciel afin de donner du tonus à vos entrées/sorties. Si vous êtes encore plus argenté, des solutions de Raid matériel existent aussi.

Le gain de performance passe aussi par une bonne compilation des applications. La majorité des logiciels disponibles par exemple pour les plateformes de type Intel sont en effet souvent compilés pour un mode x86 standard afin de rester compatibles avec tous les systèmes existants. De multiples optimisations peuvent pourtant être apportées lors de la compilation et notamment celles qui vont préciser l'architecture cible du binaire. Les gains peuvent être très importants si on prends le temps de jouer avec les *flags* d'optimisations des compilateurs (`-march=pentium`, `mcpu=pentium`, `-O4`, `-funroll-loops`, `-fno-exceptions`, etc).

Il ne faut pas aussi hésiter à utiliser d'autres compilateurs que GCC, G77 ou G++, qui sont moins performants (l'accent est mis sur la portabilité et non sur l'optimisation) que d'autres compilateurs plus commerciaux.

Pour le monde Intel, on trouve en effet :

- compilateurs C, C++, HPF, F90/95/77 de PGI ;
- compilateur C++ de Kai ;
- compilateur C++ de Borland (coming soon) ;
- compilateur F90/95/77 de NAG ;
- compilateurs F90/95/77 de Absoft.

Sur plateforme Alpha, ne pas hésiter à utiliser les compilateurs (C, C++ et F90) et bibliothèques (libm, cxml, cpml) de Compaq (gratuit pour une utilisation non commerciale) qui offrent des gains de performances étonnants.

Dans le même raisonnement, une application qui fera souvent appel à du calcul en virgule flottante verra ses performances grimper en utilisant le processeur Alpha. Toutefois si le calcul repose sur de l'entier, il sera plus intéressant de rester sur des processeurs de type x86.

Parallélisme

Si vous souhaitez améliorer encore et toujours les performances, regardez attentivement du côté du parallélisme, notion qui consiste à faire travailler collectivement un code sur plusieurs processeurs. On peut

faire du parallélisme sur du matériel SMP à l'aide généralement de processus légers (threads) ou bien sur des grappes de machines (cluster beowulf) en utilisant des bibliothèques de passage de messages (*message passing*). Cela demande cependant souvent d'écrire un code spécifique car il n'existe que très peu de compilateurs dans le monde capable de paralléliser automatiquement un code séquentiel (les gains sont souvent anecdotiques comparés à un code parallélisé par des mains expertes).

Le tuning du système

Votre système est désormais figé et votre applicatif compilé. Il reste maintenant à optimiser votre système en jouant sur le paramétrage (*tuning*) du noyau et du logiciel. Le tuning logiciel étant totalement spécifique pour chaque application, il nous reste à essayer d'optimiser la configuration du noyau.

Lorsque le noyau a été recompilé pour élaguer les options inutiles et pour l'adapter à vos besoins, on peut généralement agir sur plusieurs facteurs :

- la mémoire virtuelle ;
- le système de fichiers ;
- le réseau.

P.-S. Beaucoup d'optimisations utilisent le répertoire virtuel `/proc` pour discuter avec le noyau et modifier son paramétrage. Tapez `man sysctl` pour savoir comment conserver votre tuning à chaque démarrage.

La mémoire virtuelle

Le démon noyau *bdflush*

Vous pouvez essayer de tuner le fichier `/proc/sys/vm/bdflush`, qui contrôle le comportement du cache mémoire en lecture et écriture (par l'intermédiaire du démon noyau *bdflush*). On va se concentrer sur les variables les plus importantes du fichier : **nfract** **ndirty** **nrefill** **nref_dirt** **dummy1** **age_buffer** **age_super** **dummy2** **dummy3** :

| | |
|-------------------|--|
| nfract | Contient le pourcentage de buffers modifiés dans le cache mémoire qu'il faut atteindre ou dépasser pour physiquement écrire des données sur le disque. Plus la valeur est grande et plus le système prendra son temps pour flusher les données sur le disque et inversement. |
| ndirty | Quantité maximale de buffers modifiés qui peuvent être écrits sur le disque à un instant T pour libérer le cache mémoire. Une petite valeur ne va pas libérer le cache suffisamment vite et une grosse valeur peut amener à générer de nombreux I/O en peu de temps. |
| nrefill | Nombre de buffers qui peuvent être alloués à un instant T pour renflouer le cache mémoire. Plus ce nombre est important et plus la mémoire sera réservée en grande quantité (quitte à être sous utilisée). |
| age_buffer | Nombre de tip d'horloge (jiffies) avant qu'un buffer normal ne soit écrit sur le disque. |
| age_super | Même chose que pour <code>age_buffer</code> mais par rapport a un superbloc du système de fichier. |

Optimisation possible :

```
echo "100 1200 128 512 15 5000 500 1884 2" > /proc/sys/vm/bdflush
```

Pourcentage de la mémoire allouée pour le cache

Le fichier `/proc/sys/vm/buffermem` contient le pourcentage de la mémoire qui sera alloué au démarrage du système pour le cache. Ce fichier comporte trois champs mais seul le premier nous intéresse :

| | |
|--------------------|--|
| min_percent | Pourcentage minimum de la mémoire du système qui sera utilisé pour le cache mémoire. |
|--------------------|--|

Anticiper l'accès aux pages mémoires

Lorsque qu'une application fait une requête d'une page-mémoire et que celle-ci ne se trouve pas dans le cache, la couche système va optimiser les accès disques en chargeant non pas une seule mais plusieurs pages adjacentes à la page demandée. Le fichier `/proc/sys/vm/page-cluster` contient cette valeur mais il ne semble pas utile de dépasser la valeur 5 (la valeur par défaut est de 4).

Le système de fichiers

Optimiser les performances des disques IDE

Les distributions modernes optimisent automatiquement les performances des disques durs de type IDE (en activant le transfert 32 bits ou encore le canal DMA, par exemple). Cependant, il est souvent bien utile de savoir comment cela fonctionne. À utiliser avec précaution.

- Lister les paramètres : `cat /proc/ide/hd[a]/settings ;`
- activer les I/O 32 bits : `/sbin/hdparm -c 1 /dev/hd[a]` (-c 3 pour certain chipsets) ;
- activer le support du canal DMA : `/sbin/hdparm -d 1 /dev/hd[a]` ;
- activer le support du canal DMA mode 2 : `/sbin/hdparm -d 1 -X34 /dev/hd[a]` ;
- activer le support UltraDMA mode 2 : `/sbin/hdparm -d 1 -X66 /dev/hd[a]` ;
- pour tester les performances de votre disque : `/sbin/hdparm -[tT] /dev/hd[a]` ;
- changer le nombre de secteurs transférés à chaque interruption : `/sbin/hdparm -m [1-16] /dev/hd[a]` ;
- pour conserver la configuration après un reset IDE : `/sbin/hdparm -k 1 /dev/hd[a]`.

Tuner le nombre maximum de fichiers et d'inodes

Le noyau 2.2.x vous permet de modifier certaines limites à la volée, notamment celles concernant le nombre de file descriptors qu'il est possible d'ouvrir en parallèle (la valeur par défaut est de 4096 sur la série des 2.2.x) ainsi que le nombre d'inodes (un simple `/bin/cat` sur ces fichiers virtuels vous donnera la valeur actuelle) :

```
echo 32000 > /proc/sys/fs/file-max
echo 65000 > /proc/sys/fs/inode-max
```

Nota : il est préférable de mettre au moins trois ou quatre fois la valeur de `file-max` dans `inode-max` car les flux `stdin`, `stdout` et `stderr` ainsi que les sockets réseaux reposent sur un inode dans le noyau.

Tuner le nombre de superblocks

Si vous souhaitez augmenter le nombre de point de montage possible, vous devez jouer sur le paramètre `/proc/sys/vm/fs/super-max`, qui borne le nombre maximum de superblocks qu'il est possible d'allouer sur un système Linux.

Optimisez votre partition de swap

Il est préférable de déporter votre partition de swap sur un autre disque que le disque système afin d'optimiser les I/O (c'est une solution coûteuse mais efficace). De même, vous pouvez utiliser une particularité de Linux qui est d'autoriser l'utilisation de plusieurs partitions de swap et ce afin de stripper les I/O sur plusieurs disques. Mais cela demande de disposer d'au moins 2 disques afin de placer 2 partitions de swap de même priorité (**pri=1** dans les options de montage). Essayez de placer ces partitions en début de disque car l'accès aux données sera plus rapide.

Comment récupérer 5% d'espace disque sur une partition ext2

Il existe par défaut 5% d'espace réservé à root sur chaque partition ext2. Pour le réduire ou littéralement mettre à zéro cet espace, utiliser les commandes suivantes (bien lire le man d'abord) : `tune2fs -m pourcentage` ou, au moment de la création de la partition, `mke2fs -m pourcentage partition`.

Changer la taille des blocs d'un système de fichiers

Si vous utilisez la plupart du temps de gros fichiers, il sera probablement très profitable de formater vos partitions avec des blocs de taille plus importante. En effet, Linux utilise par défaut des taille de blocs de 1024 octets. Vous pouvez changer avec des tailles de 4096 avec la commande `mke2fs -b 4096 /dev/...`, qui utilise des blocs de 4k au lieu de 1k. Cela va notamment réduire la fragmentation et réduire le temps de vérification lors d'un `fsck`.

Ajuster le comportement du swap

Le fichier `freepages` présent sous le répertoire `/proc/sys/vm` permet de contrôler le comportement du noyau vis à vis du swap. L'écriture sur disque est toujours une opération très lourde en ressources : un bon tuning peut donc avoir un impact très fort sur les performances du système. Ce fichier contient trois valeurs (entières) : `freepages.min`, `freepages.low` et `freepages.high`.

| | |
|-----------------------|--|
| freepages.min | Lorsque le nombre de pages libre descend en dessous de cette valeur, le noyau sera seul habilité à allouer de la mémoire, |
| freepages.low | Lorsque le nombre de pages libre se retrouve en dessous de cette valeur, le noyau swappe (en grande quantité) des pages sur le disque, |
| freepages.high | Le système va essayer de maintenir le seuil de pages libre à cette valeur, quitte à swapper régulièrement des pages sur le disque. |

Sinon, vous pouvez aussi essayer de modifier le troisième paramètre du fichier `/proc/sys/vm/kswapd`, qui vous permet de changer le quantité de pages qui seront sauvées (en une passe) sur le disque par le démon noyau `kswapd`.

Activer l'option *noatime*

Pour améliorer les performances disques, on peut utiliser une option du mount appelée *noatime*. En effet, sous Linux, toute lecture d'un fichier provoque une mise à jour de l'information *atime* (c'est-à-dire le dernier accès) associée. Activer cette option peut amener des gains de performances appréciables. À noter que cela n'affectera pas l'écriture dans un fichier (qui mettra, malgré le *flag noatime*, cette information à jour). À n'utiliser qu'avec précaution car je ne connais pas les effets de bords exacts de ce flag (le *man* précise que c'est principalement utilisé pour améliorer les accès aux serveurs de news). Note de Jicé : j'utilise cette option depuis plus d'un an sans problème, mais comme disent les anglophones : YMMV !

Le réseau

Améliorer les performances NFS

Le fait d'utiliser NFS avec les options `rsize=8192,wsiz=8192` rendra la plupart du temps votre connexion plus rapide qu'avec la taille de tampon paramétrée par défaut (1024). Ceci est valable pour le noyau 2.2.x.

Tuner les connexions TCP/IP

Vous pouvez modifier plusieurs paramètres concernant la valeur de timeout de votre connexion Tcp/Ip ou encore réduire le temps que le système va mettre à tuer une connexion qui ne répond plus :

```
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 1800 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 0 > /proc/sys/net/ipv4/tcp_window_scaling
echo 0 > /proc/sys/net/ipv4/tcp_sack
echo 0 > /proc/sys/net/ipv4/tcp_timestamps
```

Tuner le serveur Samba

Vous pouvez améliorer les performances d'un serveur Samba en modifiant le paramétrage socket dans le fichier `smb.conf` : `socket options = SO_SNDBUF=4096 SO_RCVBUF=4096` .

Tampon cache TCP/IP

Un des facteurs les plus importants de la performance d'une connexion TCP/IP est la taille du tampon en entrée et en sortie. Vous pouvez modifier cela dans vos programmes avec la fonction `getsockopt()` en jouant sur les options `SO_RCVBUF` et `SO_SNDBUF`. Il ne faut pas oublier aussi de modifier ces valeurs dans le noyau (sinon cela n'aura aucun effet). En effet, la valeur par défaut est de 64 KB ($64 \times 1024 = 65536$) et cela ne convient pas si l'on souhaite utiliser du gigabit par exemple. Dans ce dernier cas, il est plus intéressant de configurer ces valeurs à 256 KB ($256 \times 1024 = 262144$) :

```
echo 262144 > /proc/sys/net/core/rmem_max
echo 262144 > /proc/sys/net/core/wmem_max
```

Divers

Améliorer les performances d'un système SMP

Vous pouvez améliorer l'allocation de pages sur un système SMP en jouant sur les valeurs du fichier `/proc/sys/vm/overcommit_memory` (le documentation n'est pas un exemple de clarté à ce sujet).

Nota : sur des systèmes mono-cpu, il est intéressant de mettre ces valeurs à zéro afin de pouvoir glaner quelques octets de mémoire vive.

Taille maximale d'un segment de mémoire partagé

Le fichier `/proc/sys/kernel/shmmax` contient la taille limite d'un segment de mémoire partagé qu'il est possible d'allouer sur votre système (très utile dans le cas de plateformes multi-processeurs). Des segments de mémoire allant jusqu'à 1GB peuvent ainsi être créés (noyaux 2.2.x).

Tuner les niveaux d'interruptions

Pour pouvez utiliser la commande `irqtune` pour changer la priorité de vos interruptions. Par exemple, si vous possédez une carte ethernet et une carte scsi, vous pourrez donner avantage à la carte réseau en tapant : `"irqtune 5 11"` ou 5 est l'interruption de la carte réseau et 11 de la carte scsi. La HomePage du logiciel est sur <http://www.best.com/~cae/irqtune/>.

Augmenter la stack size

Si vous avez des programmes qui déclarent de gros tableaux, vous pouvez changer la valeur de la stack size qui est de 8 MB par défaut (vous avez besoin d'être en root) :

```
# ulimit -s  
8192  
# ulimit -s 32768  
# ulimit -s  
32768
```

Limite de processus par utilisateur

Il faut utiliser la commande `ulimit` pour changer la configuration par défaut. Pour rendre cette limite caduque, tapez : `ulimit -u unlimited`.

Nombre maximum de processus

Pour augmenter le nombre de tâches qu'il est possible de lancer en parallèle sur votre système Linux, vous devez éditer le fichier `include/linux/tasks.h` dans les sources, modifier la variable `NR_TASKS` (jusqu'à 4092 sur x86) et générer un nouveau noyau. Ceci n'est valable que pour la série 2.2.x (la série 2.4.x élimine cette contrainte en l'associant à la taille de la mémoire du système).

Glossaire

SCSI

Small Computer Systems Interface. Très utilisé dans le monde industriel (essentiellement pour les disques durs) car il offre de très bonne performances et des caractéristiques intéressantes liées au protocole SCSI (hot-swap, nombre de devices supportés, ...).

SMP

Symmetric Multi Processing. Se dit d'un système qui possède plusieurs processeurs sur la même carte mère. Tous les processeurs partagent la même mémoire vive : on dit que c'est un système à mémoire partagée.

Thread

Un *thread* est une duplication d'un processus qui va partager la même mémoire mais, à la différence d'un *fork* classique, un thread est ce qu'on appelle un processus léger, ce qui signifie qu'il partage aussi le segment de code et le segment de donnée. Il n'y a que la pile qui est dupliquée entre le processus et son ou ses thread(s). L'avantage : les changements de contexte sont bien plus rapide avec un thread qu'avec un processus classique et l'occupation mémoire est moins importante.

Cette page est issue de la documentation « pré-wiki » de Léa a été convertie avec HTML::WikiConverter.
Elle a été créée par Lionel Tricon le 16/02/2001.

Copyright

Copyright © 16/02/2001, Lionel Tricon



*Ce document est publié sous licence Creative Commons
Attribution, Partage à l'identique, Contexte non commercial 2.0 :
<http://creativecommons.org/licenses/by-nc-sa/2.0/fr/>*