Systèmes de fichiers journalisés pour Linux

Linux Gazette numéro 68

Matteo Dell'Omodarme

matt@martine2.difi.unipi.it

1. Introduction

Un système de fichiers est le logiciel employé pour organiser et gérer l'information stockée sur disque ; il assure l'intégrité des données à la condition que les données écrites sur disque sont identiques quand elles sont relues plus tard. En plus des données contenues dans les fichiers, le système de fichier stocke et gère aussi d'autres informations au sujet des fichiers et de lui-même (i.e. informations de date et d'heure, propriétaire, permissions d'accès, taille du fichier, emplacement de stockage sur le disque, etc...) Ces informations sont communément appelées meta-données.

Puisque le système de fichiers essaie autant que possible de travailler de façon asynchrone afin d'éviter les goulots d'étranglement lors des accès au disque, une interruption soudaine de son activité amènerait des pertes de données. Par exemple, considérons le scénario suivant : qu'arriverait-il si votre machine plante quand vous travaillez sur un document qui réside sur le standard de Linux : Ext2FS ? Il y a plusieurs réponses :

- La machine plante après que vous ayez sauvegardé le fichier. C'est le meilleur scénario : vous n'avez rien perdu. Redémarrez simplement votre machine et continuez à travailler sur votre document.

- La machine plante avant que vous n'ayez sauvegardé le fichier. Vous avez perdu toutes les modifications que vous aviez faites, mais l'ancienne version du fichier est toujours sur disque.
- La machine plante au moment exact où vous sauvegardez votre fichier. C'est le plus mauvais cas : la nouvelle version du fichier recouvre physiquement l'ancienne version sur disque. Vous vous retrouvez avec un fichier à moitié nouveau mais aussi à moitié ancien. Si le fichier est un fichier binaire, vous ne pourrez plus le relire puisque le format interne sera endommagé et ne correspondra pas à ce que l'application s'attend à lire.

Dans ce dernier scénario, les choses peuvent être pires encore si les données écrites sur disques font partie des meta-données, tel le répertoire. Et maintenant, au lieu d'avoir un simple fichier corrompu, c'est un système de fichiers qui l'est, et vous pouvez y perdre un répertoire voire la partition de disque complète.

Le système de fichier standard de Linux, Ext2FS, essaie de prévenir et corriger les cas de corruption de meta-données en effectuant une analyse complète du système de fichier (**fsck**) lors du démarrage du système. Et comme Ext2FS inclut des copies redondantes des meta-données critiques, il est peu probable que les données soient complètement perdues. Le système s'emploie à savoir où se situent les meta-données corrompues, et soit répare les dommages en recopiant la version redondante, soit efface simplement le ou les fichiers dont les meta-données sont corrompues.

Bien évidemment, plus la taille du système de fichier à vérifier est importante, plus le temps de vérification sera long. Sur des espaces disque de plusieurs giga-octets, cela peut prendre un temps très important de vérifier les meta-données au démarrage.

Et comme Linux commence à être utilisé pour des applications de plus en plus complexes, sur des gros serveurs, et de moins en moins de tolérance pour les arrêts de production, le besoin d'un système de fichier protégeant données et meta-données se fait de plus en plus sentir.

Les systèmes de fichiers journalisés pour Linux sont la réponse à ce besoin.

2. Qu’est-ce qu’un système de fichiers journalisé ?

Nous ne ferons ici qu’une simple introduction aux systèmes de fichiers journalisés. Pour obtenir des informations plus avancées et techniques, reportez-vous à l’article de Juan I. Santos Florido dans la Linux-Gazette n°55 (http://www.linuxgazette.com/issue55/florido.html). D’autres informations sont disponibles dans l’article http://freshmeat.net/articles/view/212/.

La plupart des systèmes de fichiers modernes utilisent des techniques de journalisation empruntées au monde des bases de données pour améliorer la reprise sur incident. Les transactions sont écrites séquentiellement sur une partie du disque appelée *journal* ou *log* avant de les écrire sur disque à leur place définitive dans le système de fichiers. Les mises en oeuvres varient dans le contenu écrit dans le journal. Certaines n’y mettent que les meta-données, alors que d’autres y enregistrent toutes les écritures[1]

Maintenant, si un plantage survient avant qu’une entrée de journal ne soit validée[2], les données originales sont toujours sur disque et vous avez uniquement perdu vos dernières modifications. Si le plantage intervient pendant l’**update**[3] (i.e. après la validation de l’entrée du journal), l’entrée du journal reflète ce qui aurait dû être fait. Et au ré-amorçage du système, il n’a qu’a rejouer les transactions du journal, et terminer l’**update** qui a été interrompu.

Dans les deux cas, vous avez des données valides, et non une partition inutilisable. Et comme le temps de reprise sur incident associé à cette approche journalisée, le système est utilisable dans les quelques secondes qui suivent.

Il est aussi important de noter que l’utilisation d’un système de fichiers jounalisé ne rend pas complètement obsolète l’emploi d’un programme de vérification d’intégrité (**fsck**). Des erreurs matérielles ou logicielles peuvent corrompre de façon aléatoire quelques blocs du système de fichiers, interdisant une récupération grâce au journal de transactions.

3. Systèmes de fichiers journalisés disponibles

Dans ce qui va suivre, nous allons considérer trois systèmes de fichiers journalisés.

Le premier est ext3. Développé par Stephen Tweedie, un des principaux développeurs du noyau Linux, ext3 ajoute la journalisation à ext2. Il est disponible en version alpha à ftp://ftp.linux.org.uk/pub/linux/sct/fs/jfs/

Namesys développe un système de fichiers journalisé du nom de ReiserFS. Il est disponible à http://www.namesys.com (http://www.namesys.com/)

SGI a sorti le 1er mai 2001 la version 1.0 de son système de fichiers XFS pour Linux. Vous pouvez le trouver à http://oss.sgi.com/projects/xfs/.

Dans cet article, ces trois solutions seront testées et mesurée au banc d'essai en utilisant deux programmes différents.

4. Installer ext3

Pour obtenir des détails techniques au sujet de ext3, se référer à la publication (ftp://ftp.kernel.org/pub/linux/kernel/people/sct/ext3/journal-design.ps.gz) et à la présentation (http://olstrans.sourceforge.net/release/OLS2000-ext3/OLS2000-ext3.html) du Dr Stephen Tweedie.

Le système de fichier ext3 est directement dérivé de son ancêtre ext2. Il a le gros avantage d'être complètement compatible avec ext2 puisqu'il s'agit d'un ext2 avec journalisation. Le désavantage le plus visible est que ext3 ne met pas en oeuvre la plupart des caractéristiques avancées des systèmes de fichiers modernes, qui améliorent manipulation des données et compacité de stockage.

ext3 est livré sous la forme d'un patch pour le noyau 2.2.19. Il vous faut donc trouver un noyau `linux-2.2.19` sur ftp://ftp.kernel.org/ ou depuis un site miroir (http://www.kernel.org/mirrors/). Le patch est disponible depuis ftp://ftp.linux.org.uk/pub/linux/sct/fs/jfs/ ou ftp://ftp.kernel.org/pub/linux/kernel/people/sct/ext3 ou depuis un miroir de ce site.

Vous avez besoin des fichiers suivants :

- `ext3-0.0.7a.tar.bz2` (ftp://ftp.kernel.org/pub/linux/kernel/people/sct/ext3/ext3-0.0.7a.tar.bz2) : le patch noyaui ;
- `e2fsprogs-1.21-WIP-0601.tar.bz2` (ftp://ftp.kernel.org/pub/linux/kernel/people/sct/ext3/e2fsprogs-1.21-WIP-0601.tar.bz2) : les programmes e2fsprogs avec le support pour ext3.

Copiez le noyau Linux `linux-2.2.19.tar.bz2` et `ext3-0.0.7a.tar.bz2` dans le répertoire `/usr/src`, puis en extraire le contenu :

```
mv linux linux-old
tar -Ixvf linux-2.2.19.tar.bz2
tar -Ixvf ext3-0.0.7a.tar.bz2
cd linux
cat ../ext3-0.0.7a/linux-2.2.19.kdb.diff | patch -sp1
cat ../ext3-0.0.7a/linux-2.2.19.ext3.diff | patch -sp1
```

Le premier fichier **diff** est une copie du patch dévermineur noyau de SGI. Le second est le patch de ext3. Maintenant, configurez le noyau, en spécifiant YES à l'option « Enable Second extended fs development code » dans la section « filesystem », puis le construire.

Après la compilation et l'installation du noyau, vous devez construire et installer la suite e2fsprogs :

```
tar -Ixvf e2fsprogs-1.21-WIP-0601.tar.bz2
cd e2fsprogs-1.21
./configure
make
make check
make install
```

Et c'est tout. Le point suivant est de construire un système de fichiers ext3 dans une partition. Re-démarrez avec le nouveau noyau. Vous avez maintenant deux options : créer un nouveau système de fichiers, ou en journaliser un existant.

- Créer un nouveau système de fichiers. Utilisez simplement la commande **mke2fs** de la suite installée e2fsprogs, et en utilisation l'option `-j` en lançant **mke2fs** :

  ```
  mke2fs -j /dev/xxx
  ```

  où `/dev/xxx` est le périphérique où vous voulez créer le système de fichiers ext3. L'option `-j` permet de spécifier à **mke2fs** de créer un système ext3 avec un journal caché. Vous pouvez contrôler la taille du journal avec l'option `-Jsize=<n>` (n étant la taille en Mo du journal).

- Mettre à jour un système de fichiers ext2 en ext3. Utilisez simplement **tune2fs** :

  ```
  tune2fs -j /dev/xxx
  ```

  Vous pouvez lancer cette commande sur un système de fichier monté ou démonté. S'il est monté, un fichier `.journal` sera créé en sa racine ; s'il est démonté, un inode système et caché sera utilisé pour y mettre le journal. Ainsi, toutes les données du système de fichiers sont préservées.

Vous pouvez monter le système de fichier ext3 avec la commande :

```
mount -t ext3 /dev/xxx /mount_dir
```

Puisque ext3 est essentiellement un système de fichier ext2 avec journalisation, un système de fichier ext3 proprement démonté peut être remonté en tant que ext2 sans aucune autre commande.

5. Installer XFS

Pour un survol technique de XFS, référez-vous à la page linux-XFS du site SGI (http://linux-xfs.sgi.com/projects/xfs/) et aux pages des publications de SGI. (http://linux-xfs.sgi.com/projects/xfs/publications.html) Voir aussi la Foire Aux Questions (http://oss.sgi.com/projects/xfs/faq.html).

XFS est un système de fichiers journalisé pour Linux disponible chez SGI. La technologie est mature, ce qui a été prouvé sur les systèmes Irix en tant qye système de fichiers par défaut pour tous les clients de SGI. XFS est disponible sous LPG (GNU GPL).

XFS Linux 1.0 est disponible pour la version 2.4 du noyau Linux. J'ai essayé le patch pour la version 2.4.2. La première étape est donc d'obtenir le noyau linux-2.4.2 d'un des miroirs de kernel.org. Les patches sont disponibles à ftp://oss.sgi.com/projects/xfs/download/Release-1.0/patches/. Téléchargez-y :

- `linux-2.4-xfs-1.0.patch.gz` (ftp://oss.sgi.com/projects/xfs/download/Release-1.0/patches/linux-2.4-xfs-1.0.patch.gz)
- `linux-2.4.2-core-xfs-1.0.patch.gz` (ftp://oss.sgi.com/projects/xfs/download/Release-1.0/patches/linux-2.4.2-core-xfs-1.0.patch.gz)
- `linux-2.4.2-kdb-04112001.patch.gz` (ftp://oss.sgi.com/projects/xfs/download/Release-1.0/patches/linux-2.4.2-kdb-04112001.patch.gz)

Copiez le noyau Linux `linux-2.4.2.tar.bz2` dans `/usr/src`, renommez le répertoire `linux` existant en `linux-old` et extrayez le nouveau noyau :

```
mv linux linux-old
tar -Ixf inux-2.4.2.tar.bz2
```

Copiez chacun des patches cités c-dessus et appliquez-les :

```
zcat patchfile.gz | patch -p1
```

Configurez ensuite le noyau, en validant les options « XFS filesystem support » (CONFIG_XFS_FS) et « Page buffer support » (CONFIG_PAGE_BUF) dans la section « Filesystem ». Notez que vous aurez aussi besoin de mettre à jour les utilitaires systèmes dans les versions montrées ici ou les suivantes

- modutils-2.4.0 (http://www.kernel.org/pub/linux/utils/kernel/modutils/v2.4/)
- http://www.gnu.org/software/autoconf/autoconf.htmlautoconf-2.13
- e2fsprogs-devel-1.18 (http://e2fsprogs.sourceforge.net/)

Installez ensuite le nouveau noyau et redémarrez. Téléchargez maintenant les outils XFS (ftp://oss.sgi.com/projects/xfs/download/Release-1.0/cmd_tars/xfsprogs-1.2.0.src.tar.gz). Cette archive tar contient tout un jeu de commandes pour utiliser XFS, comme **mkfs.xfs**. Pour les construire :

```
tar -zxf  xfsprogs-1.2.0.src.tar.gz
cd xfsprogs-1.2.0
make configure
make
make install
```

Après avoir installé ces commandes, vous pouvez créer un nouveau système de fichiers avec la commande :

```
# mkfs -t xfs /dev/xxx
```

Une option importante dont vous aurez peut-être besoin est l'option `-f` qui forcera la création du nouveau système de fichiers s'il en existe déjà un sur la partition. Bien évidemment, cela détruira toute donnée présente sur cette partition :

```
mkfs -t xfs -f /dev/xxx
```

Vous pouvez ensuite monter le système de fichier avec la commande :

```
mount -t xfs /dev/xxx /mount_dir
```

6. Installer ReiserFS

Pour les détails techniques au sujet de ReiserFS, référez-vous au site de Namesys (http://www.namesys.com/) et la Foire aux Questions (http://www.namesys.com/faq.html).

ReiserFS est intégré au noyau Linux depuis la version 2.4.1-pre4. Vous devez néanmoins récupérer les outils (i.e. **mkreiserfs** pour initialiser le système de fichier sur une partition vide, le retailleur, etc.)

La toute dernière version de ReiserFS est disponible en tant que patch envers les noyaux 2.2.x et 2.4.x. J'ai testé la version pour le noyau Linux 2.2.19.

La première étape, comme d'habitude, est d'obtenir le noyau standard `linux-2.2.19.tar.bz2` depuis un des miroirs de kernel.org. Récupérez aussi le patch ReiserFS pour 2.2.19 (ftp://ftp.namesys.com/pub/reiserfs-for-2.2/)

Notez que si vous utilisez la version pour le noyau 2.4, vous aurez aussi besoin de l'archive des utilitaires `reiserfsprogs-3.x.0j.tar.gz` (ftp://ftp.namesys.com/pub/reiserfsprogs/reiserfsprogs-3.x.0j.tar.gz)[4]

Déballez maintenant l'archive du noyau et le patch. Copiez les archives dans `/usr/src` et renommez le répertoire `linux` en `linux-old` ; puis lancez les commandes :

```
tar -Ixf linux-2.2.19.tar.bz2
bzcat linux-2.2.19-reiserfs-3.5.33-patch.bz2 | patch -p0
```

Compilez le noyau en ayant validé le support de ReiserFS dans la section « Filesystem ». Compilez et installez les utilitaires :

```
cd /usr/src/linux/fs/reiserfs/utils
make
make install
```

Installez le nouveau noyau et rebootez. Vouz pouvez maintenant créer le nouveau système de fichier ReiserFS, avec la commande suivante :

```
mkreiserfs /dev/xxxx
```

et le monter :

```
mount -t reiserfs /dev/xxx /mount_dir
```

7. Les systèmes de fichiers au banc d'essai

Pour le test, j'ai utilisé un Pentium III, avec 16 Mo de mémoire et un disque dur de 2 Go, installé avec la distribution Linux RedHat 6.2. Tous les systèmes de fichiers ont correctement fonctionné chez moi, ce qui m'a fait démarré un petit test de performances pour pouvoir les comparer. Le premier test fut d'arrêter brutalement l'alimentation électrique afin de vérifier la reprise grâce au journal. Tous les systèmes de fichiers ont passé cette phase avec succès, et la machine fut en ordre de marche dans les secondes qui suivirent pour chaque système de fichier.

L'étape suivante est l'analyse de performances en utilisant le programme bonnie++, disponible à http://www.coker.com.au/bonnie++/. Ce programme analyse les accès de type base de données dans un fichier, ainsi que les créations, lectures, et destructions de petits fichiers simulant ainsi les utilisations faites par des programmes tels Squid, INN, ou les programmes utilisant le format Maildir (qmail). La commande utilisée pour lancer le test est :

```
bonnie++ -d/work1 -s10 -r4 -u0
```

Elle lance le test en utilisant 10 Mo dans le système de fichiers monté sur le répertoire `/work1`. Les autres options spécifient la quantité de mémoire utilisée (`-r4`), et l'utilisateur (`-u0`, c'est-à-dire l'administrateur).

Les résultats apparaissent dans les tableaux ci-dessous :

		Caractère	F
SF	Espace de test		Ko/s
ext2	10 Mo		147
ext3	10 Mo		136
xfs	10 Mo		120
reiserfs	10 Mo		145

		Création	Le
SF	Nombre de fichiers		/ se
ext2	16		94
ext3	16		89
xfs	16		92
reiserfs	16		130

Deux résultats sont montrés par type de test : le débit du système de fichiers (en Ko par seconde) et le temps d'utilisation du processeur (en %). Plus les chiffres de débit sont élevés, meilleurs ils sont pour le système de fichiers. L'inverse est vrai pour le temps processeur.

Ainsi que vous pouvez le voir, ReiserFS remporte haut-la-main la victoire pour la gestion de fichiers (sections « Créations séquentielles » et « Créations aléatoires »), dépassant ses opposants d'un facteur supérieur à 10. De plus, il est quasiment aussi bon que les autres systèmes de fichiers dans les lectures et écritures séquentielles. Il n'y a pas de différence significative entre les autres systèmes de fichiers. XFS est aussi rapide que ext2, et ext3, comme prévu, un peu plus lent que ext2 (qui fait la même chose, la perte de temps dues aux appels systèmes pour la journalisation en moins.)

Le dernier test employé est mongo, un programme disponible sur la page benchmark pour ReiserFS de www.namesys.com (html://www.namesys.com/), programme que j'ai modifié pour tester les trois systèmes de fichiers journalisés. J'ai inséré dans le script perl mongo.pl les commandes pour monter les systèmes de fichiers XFS et ext3 et pour les formater. Puis j'ai lancé le banc de test.

Le script initialise une partition /dev/xxx, la monte, et fait tourner un certain nombre de processus dans chaque phase : création de fichier, copie, lien symbolique, lectures, état du fichier[5], renommage, et destruction du fichier. Mongo calcule aussi la fragmentation après les phases de création et de copie :

Fragmentation = nombre de fragments / nombre de fichiers

Vous trouverez les résultats du test sous différentes formes dans le répertoire `results` :

log	résultats bruts
log.tbl	résultats à injecter dans le programme **compare**
log_table	log

Les tests furent exécutés comme indiqué dans l'exemple suivant

```
mongo.pl ext3 /dev/hda3 /work1 logext3 1
```

où ext3 doit être remplacé par reiserfs ou xfs pour pouvoir tester les autres systèmes de fichiers. Les autres arguments sont le périphérique à monter où se situe le système de fichier à tester, le point de montage, le nom du fichier où les résultats seront conservés et le nombre de processus à lancer.

Vous trouverez dans les tables suivantes les résultats de l'analyse. Les données correspondent au temps d'exécution, en secondes, la valeur la moins élevée étant la meilleure. Dans le premier tableau, la taille moyenne d'un fichier est de 100 octets, dans le second, 1000 octets, et dans le dernier, 10000 octets.

	ext3 files=68952 size=100 bytes dirs=242	XFS files=68952 size=100 bytes dirs=241	reiserFS files=68952 size=100 bytes dirs=241
Création	90,07	267.86	53,05
Fragmentation	1,32	1,02	1,00
Copie	239,02	744,51	126,97
Fragmentation	1,32	1,03	1,80
Liens symboliques	0	203,54	105,71
Lecture	782,75	1543,93	562,53
Etat des fichiers (stat)	108,65	262,25	225,32
Renommage	67,26	205,18	70,72
Destruction	23,80	389,79	85,51

	ext3 files=11248 size=1000 bytes dirs=44	XFS files=11616 size=1000 bytes dirs=43	ReiserFS files=11616 size=1000 bytes dirs=43
Création	30,68	57,94	36,38
Fragmentation	1,38	1,01	1,03
Copie	75,21	149,49	84,02

Fragmentation	1,38	1,01	1,43
Liens symboliques	16,68	29,59	19,29
Lecture	225,74	348,99	409,45
Etat des fichiers (stat)	25,60	46,41	89,23
Renommage	16,11	33,57	20,69
Destruction	6,04	64,90	18,21

	ext3 files=2274 size=10000 bytes dirs=32	XFS files=2292 size=10000 bytes dirs=31	reiserFS files=2292 size=10000 bytes dirs=31
Création	27,13	25,99	22,27
Fragmentation	1,44	1,02	1,05
Copie	55,27	55,73	43,24
Fragmentation	1,44	1,02	1,12
Liens symboliques	1,33	2,51	1,43
Lecture	40,51	50,20	56,34
Etat des fichiers (stat)	2,34	1,99	3,52
Renommage	0,99	1,10	1,25
Destruction	3,40	8,99	1,84

De ces tableaux, vous pouvez voir que ext3 est souvent plus rapide pour les opérations de type stat, destruction et renommage de fichier, alors que ReiserFS gagne pour les opération de création et de copie. Notez également que les temps d’exécution de ReiserFS sont meilleurs dans le premier cas (petits fichiers) comme prévu dans sa documentation technique.

8. Conclusions

Il existe actuellement deux systèmes de fichiers journalisés robustes et fiables pour Linux (i.e. XFS et ReiserFS) qui peuvent être utilisés sans crainte. ext3 est toujours en version alpha et peut subir quelque panne. J'ai eu quelques problèmes en utilisant bonnie++ sur ce système de fichiers : le système a enregistré quelques erreurs de la part de la mémoire virtuelle et a détruit l'interpréteur de commandes interactif que j'utilisais.

Au vu des résultats de ce banc d'essai, mon conseil est d'installer ReiserFS à l'avenir (je le ferai sûrement).

Paru dans le numéro 68 de la Linux Gazette, juillet 2001.

Traduction française par Jérôme Fenal <`jerome@fenal.org`>.

Notes

1. NdT : et donc les données contenues dans les fichiers.
2. NdT : en franglais : *committée*
3. NdT : on gardera le terme anglais puisqu'il s'agit du daemon (en mode noyau - pour Linux - ou utilisateur, suivant les Unix) du même nom
4. NdT : En effet, les utilitaires dans la version 3.5 de ReiserFS pour noyau 2.2 sont dans le patch pour des questions de réutilisation du code. Cela n'empêche en rien le fait de récupérer les derniers outils (les mêmes que pour le noyau 2.4, en particulier pour **reiserfsck**.
5. NdT : de l'appel système stat