

0.1 Généralité sur les systèmes de fichiers

0.1.1 1. Structure du disque dur :<

Pour comprendre comment fonctionne un système de fichiers, il est important de bien distinguer, la partie physique (la partition de disque dur) et sa représentation (système de fichiers), les 2 étant bien-sûr liés. Un disque dur est périphérique de stockage de type bloc, c'est à dire qu'il n'est pas possible d'accéder à un octet et encore moins à un unique bit. Tous les échanges se font par ensemble blocs. Un disque dur comporte des pistes circulaires concentriques. Les pistes sont découpées en arcs de cercles appelés secteurs. Les secteurs sont repérés par un couple (n° piste, n° secteur) Un secteur est formé de plusieurs octets (128,256,512 octets ...). La modification d'un unique octet n'est pas faisable, pour lire un octet il faut récupérer le secteur en entier et pour l'écriture, idem. Dans les systèmes de fichiers traditionnels le disque dur se présente comme une succession de blocs (1 bloc peut valoir 2 secteurs ou moins ou même plus), le n° du bloc est appelé adresse physique. Les n° des blocs sont rangés dans une table. Un fichier est représenté par plusieurs blocs contigus ou non (successifs ou non), si les blocs sont trop gros le système de fichier fait potentiellement du gaspillage de place (ex : FAT 16). Si les blocs sont trop petits, le système de fichier fragmentera rapidement.

0.1.2 2. Système de fichiers traditionnels :

Les systèmes de fichiers traditionnels comme ext2,fat32 utilisent un système de cache, c'est à dire que les données que vous manipulez ne sont pas directement écrites sur le disque mais dans une mémoire plus rapide : le cache. L'avantage c'est que les accès disques s'en trouvent profondément améliorés. L'inconvénient c'est qu'en cas de coupure intempestive, les données dans le cache sont perdues. D'où le fsck (scandisk) au redémarrage du PC. Ces systèmes de fichiers utilisent une table pour repérer les blocs vides, ainsi lors de l'écriture d'un nouveau fichier, la table est parcourue et l'emplacement du fichier est déterminé en fonction de l'espace disponible. L'inconvénient c'est que plus le système de fichier est grand plus le parcours de cette table est long, donc coûteux. Lors de la modification d'un fichier par rajout, par exemple, la table des blocs est parcourue à la recherche d'un nombre de blocs contigus correspondants aux modifications à apporter. S'il s'agit d'un fichier d'une taille égale à un bloc, cela ne pose aucun problème, dans le cas contraire, plus les modifications sont importantes plus le ralentissement général du système est perceptible.

0.1.3 3. Amélioration du cache :

Sur les systèmes de fichiers modernes comme Reiserfs, ext3, XFS ... on rajoute la notion de journal. D'où le nom système de fichier journalisé. Les données sont toujours mises en cache mais le vidage du cache est contrôlé finement par le système de fichier. De plus le système de fichier tient un journal des données en cache, ainsi en cas de coupure intempestive, le fsck sera plus rapide car le système de fichier se contentera de lire le journal. Attention, on ne gagne pas forcément en performance car la journalisation a un coût mais on limite les pertes de données.

0.1.4 4. Les Balanced Trees :

Nous avons vu précédemment que pour connaître l'emplacement d'un fichier, il fallait parcourir la table des blocs et qu'en fonction de sa taille cela pouvait devenir très coûteux en ressource. Avec les Balanced Trees, cette table a la forme d'un arbre où les feuilles sont des pointeurs vers des fichiers. Ainsi pour retrouver un fichier il n'est plus nécessaire de parcourir toute la table mais juste les branches qui nous intéressent. Par la même occasion trouver de l'espace libre pour un fichier devient là aussi très facile.

0.1.5 5. Les extents :

Un extent est un référencement de groupes de blocs libres contigus. Il mentionne l'adresse du premier bloc, la taille de l'ensemble et l'offset. Ainsi sur un système de fichiers utilisant des extents, pour allouer de l'espace à un nouveau fichier ou rallonger un ancien, il devient inutile de parcourir la table des blocs vides, celle des extents suffit. Comme elle est plus petite, vous gagnez en efficacité. Ceci permet de lutter efficacement contre la fragmentation.

0.1.6 6. Les systèmes de fichiers :

- **Ext2** : c'est le système de fichiers 32 bits natifs sous linux. Très performant bien que limité, il ne fragmente quasiment pas.
- **Ext3** : c'est le successeur annoncé d'ext2, il permet de moderniser son système de fichier notamment par la journalisation, en douceur. Nous en parlons plus longuement [ICI](#)¹.
- **Reiserfs** : c'est le premier système de fichier disponible sous linux. Nous en parlons plus longuement [ICI](#)².
- **XFS** : C'est le projet développé par SGI (IRIX), il s'agit du portage du successeur d'EFS sous Linux. Il utilise les Balanced Trees, c'est un système de fichiers 64 bits, il supporte des fichiers de plus d'un million de To. Il est également très à l'aise en SMP. Il est aujourd'hui en version 1.1 disponible sous forme de noyau 2.4.18 pré-patché. Vous trouverez plus de renseignements à l'adresse suivante³
- **JFS** : C'est le projet développé par IBM (AIX), il utilise les extents et Balanced Trees. Il peut accepter des fichiers d'une taille maximale de 512 To, il supporte le LVM et l'allocation dynamique des i-nodes. Il est aujourd'hui en version 1.0.20 et est disponible sous forme de patch à appliquer à différents noyaux dont le 2.4.18. C'est également un 32 bits. Vous trouverez plus de renseignements à l'adresse suivante⁴

¹<http://www.truostonme.net/didactels/141.html>

²<http://www.truostonme.net/didactels/142.html>

³<http://oss.sgi.com/projects/xfst/>

⁴<http://oss.software.ibm.com/developerworks/opensource/jfs/>