Chapitre 3

Le Buffer Cache

3.1 Introduction au buffer cache

Le buffer cache est un ensemble de structures de données et d'algorithmes qui permettent de minimiser le nombre des accès disque.

Ce qui est très important car les disques sont très lents relativement au CPU et un noyau qui se chargerait de toutes les entrées/sorties serait d'une grande lenteur et l'unité de traitement ne serait effectivement utilisée qu'à un faible pourcentage (voir Historique).
Deux idées pour réduire le nombre des accès disques :

1. bufferiser les différentes commandes d'écriture et de lecture de façon à faire un accès disque uniquement pour une quantité de données de taille raisonnable (un bloc disque).
2. Eviter des écritures inutiles quand les données peuvent encore être changées (écriture différées).

3.1.1 Avantages et désavantages du buffer cache

- Un accès uniforme au disque. Le noyau n'a pas à connaître la raison de l'entrée-sortie. Il copie les données depuis et vers des tampons (que ce soient des données, des inodes ou le superbloc). Ce mécanisme est modulaire et s'intègre facilement à l'ensemble du système qu'il rend plus facile à écrire.
- Rend l'utilisation des entrées-sorties plus simple pour l'utilisateur qui n'a pas à se soucier des problèmes d'alignement, il rend les programmes portables sur d'autres UNIX [1].
- Il réduit le trafic disque et de ce fait augmente la capacité du système. Attention : le nombre de tampons ne doit pas trop réduire la mémoire centrale utilisable.
- L'implémentation du buffer cache protège contre certaines écritures "concurrentes"
- L'écriture différée pose un problème dans le cas d'un crash du système. En effet si votre machine s'arrête (coupure de courant) et que un (ou plusieurs) blocs sont marqués "à écrire" ils n'ont donc pas étés sauvegardés physiquement. L'intégrité des données n'est donc pas assurée en cas de crash.
- Le buffer cache nécessite que l'on effectue une recopie (interne à la mémoire, de la zone utilisateur au cache ou inversement) pour toute entrée-sortie. Dans le cas de transferts nombreux ceci ralentit les entrées-sorties .

3.2 Le buffer cache, structures de données.

Le statut d'un bloc cache est une combinaison des états suivants :

verrouillé l'accès est reservé à un processus.

[1] Les problèmes d'alignement existent toujours quand on transfère des données, cf. protocoles XDR,RPC

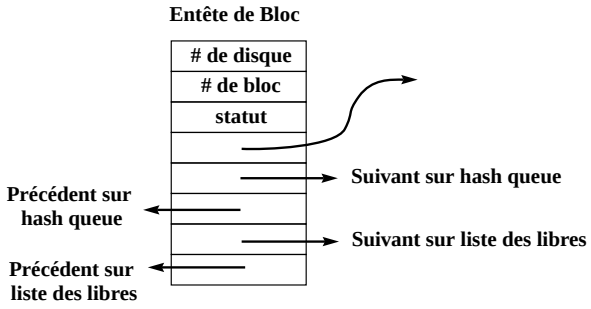

FIG. 3.1 – Structure des entêtes de Bloc du Buffer Cache

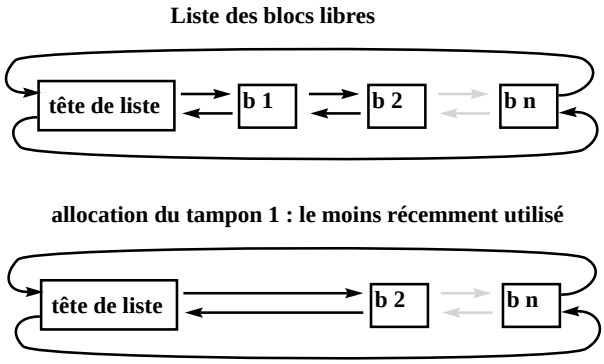

FIG. 3.2 – La liste des tampons libres.

valide (les données contenues dans le bloc sont valides).

"à écrire" les données du bloc doivent être écrites sur disque avant de réallouer le bloc (c'est de l'écriture retardée).

actif le noyau est en train d'écrire/lire le bloc sur le disque.

attendu un processus attend la libération du bloc.

3.2.1 La liste doublement chaînée des blocs libres

Les tampons libres appartiennent simultanément à deux listes doublement chaînées : la liste des blocs libres et la hash-liste correspondant au dernier bloc ayant été contenu dans ce tampon.

L'insertion dans la liste des tampons libres se fait en fin de liste, la suppression (allocation du tampon à un bloc donné) se fait en début de liste, ainsi le tampon alloué est le plus vieux tampon libéré[2]. Ceci permet une réponse immédiate si le bloc correspondant est réutilisé avant que le tampon ne soit alloué à un autre bloc.

3.3 L'algorithme de la primitive `getblk`

```
Algorithme getblk (allocation d'un tampon)
entree : # disque logique , # de block
sortie : un tampon verrouille utilisable pour manipuler bloc
{
    while (tampon non trouve)
```

[2]ordre fifo : first in first out

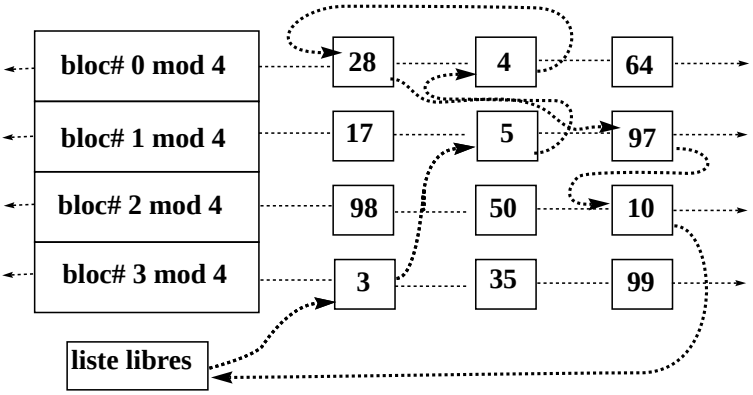

FIG. 3.3 – Etat du buffer cache avant les scénarios 1, 2 et 3.

```
    {
        if (tampon dans sa hash liste)
        {
                if (tampon actif )
                {
           [5]          sleep attente de la liberation du tampon
                        continuer
                }
           [1]  verrouiller le tampon
                retirer le tampon de la liste des tampons libres
                retourner le tampon
        }
        else  /* n'est pas dans la hash liste */
        {
                if (aucun tampon libre )
                {
            [4]         sleep attente de la liberation d'un tampon
                        continuer
                }
                retirer le tampon de la liste libre
            [3] if (le tampon est a ecrire)
                {
                    lancer la sauvegarde sur disque
                    continuer
                }
            [2] retirer le buffer de son ancienne liste
                 de hashage, le placer sur la nouvelle
                retourner le tampon
        }
    }
}
```

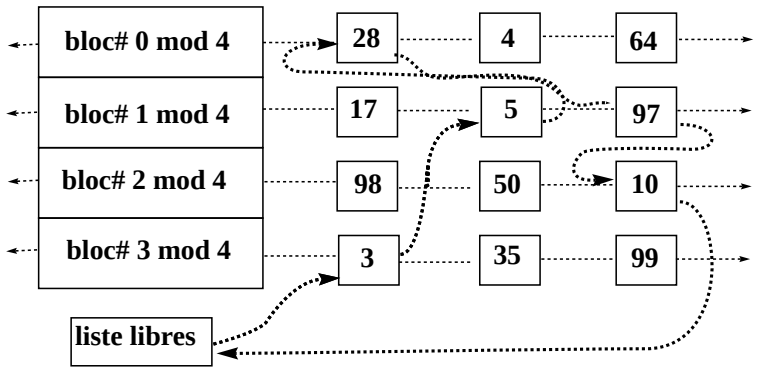

FIG. 3.4 – Scénario 1- Demande d'un tampon pour le bloc-disque 4.

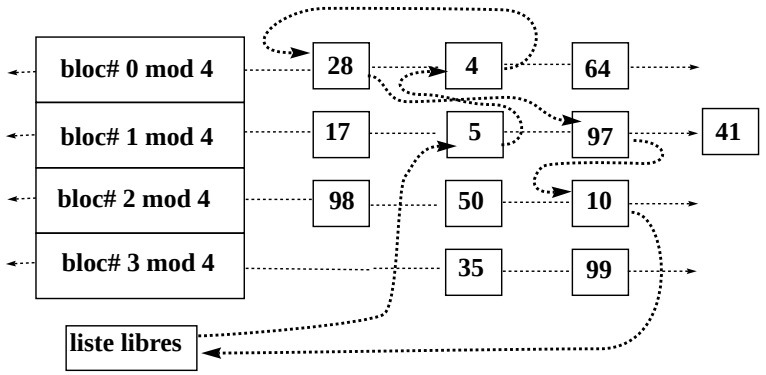

FIG. 3.5 – Scénario 2- Demande d'un tampon pour le bloc-disque 41.

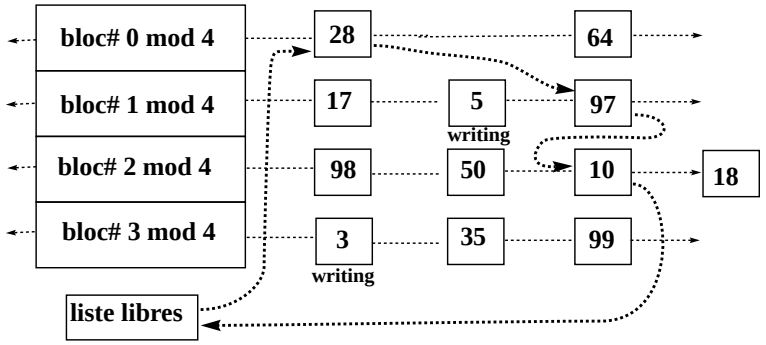

FIG. 3.6 – Scénario 3- Demande pour le bloc 18 (3 & 5 marqués à écrire).

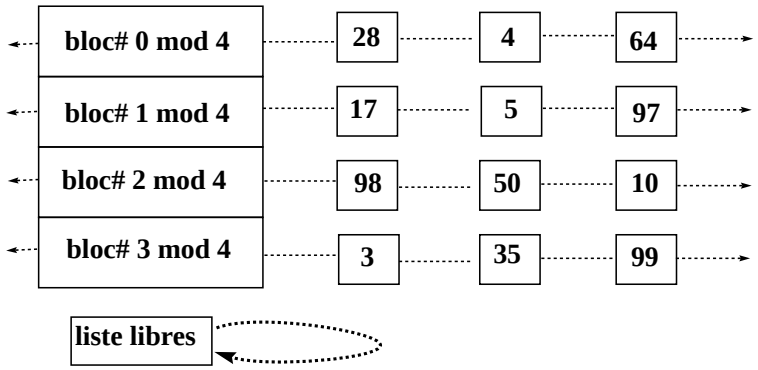

FIG. 3.7 – Scénario 4- Plus de blocs libres.

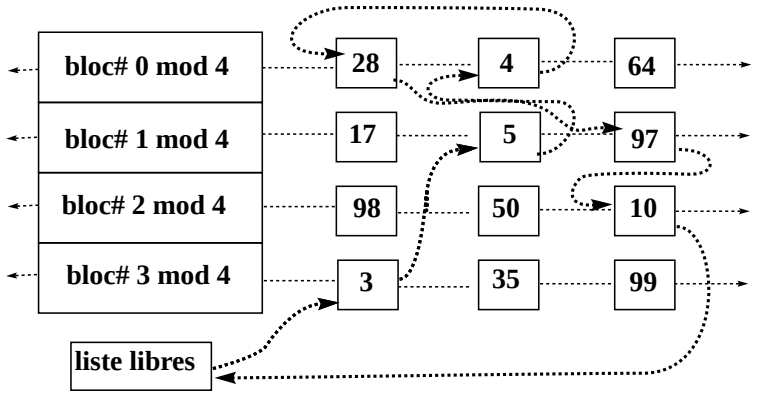

FIG. 3.8 – Scénario 5- Demande pour le bloc 17 qui est déjà utilisé.