Système d'exploitation : Projet 2

Raphael Javaux et Marien Bourguignon

Avril-Mai 2013

1 Architecture

1.1 Enregistrement d'une page dans le ZRLE cache

Notre implémentation intercepte les pages allant être swappées dans la fonction `swap_writepage()`. Chaque page est analysée pour calculer sa taille une fois compressée avec l'algorithme ZRLE.

Si la taille de la page une fois compressée est inférieure à 80% à la taille d'une page non compressée, alors nous parcourons l'ensemble des secteurs libres (listés sous la forme d'un chaînage) de nos "bins" à la recherche du premier secteur suffisamment grand pour accueillir la page compressée. Si un tel espace existe, la page est enregistrée dans cette zone.

Un nœud est ensuite ajouté dans un `radix tree` du kernel pour enregistrer le correspondance entre une `swp_entry_t` et la page compressée. A cet instant, la page est considérée comme étant présente dans le ZRLE cache.

Etant donné que les pages sont interceptées dans la fonction `swap_writepage()`, une `swp_entry_t` a été déjà allouée pour chacune d'elle. Malheureusement, cela implique qu'un espace a été alloué sur la `swap device` alors qu'il est inutilisé. Cependant, il s'agit de la solution la plus simple que nous avons pu trouver. Nous avons d'abord envisagé d'intercepter la page dans `shrink_page_list()`, avant qu'un secteur de la swap ne soit alloué, mais cela s'avère beaucoup trop complexe (notamment, tout `pte_t` pourrait pointer vers une entrée du cache en plus d'une page physique ou d'une entrée d'une `swap device`, et ce cas devrait être envisagé à chaque endroit où un `pte_t` est accédé).

Si la page ne peut pas être suffisamment compressée, ou si aucun espace n'est disponible dans le cache, la page est transmise à la `swap device` comme dans le noyau non modifié.

1.2 Recopie des bins du cache sur le disque

Nous ne sommes malheureusement pas parvenus à implémenter le transfert des "bins" remplis sur le disque.

1.3 Récupération d'une page

Notre implémentation intercepte les requêtes de récupération des pages de swap dans la fonction `swap_writepage()`. Pour chaque requête, nous recherchons si une correspondance à la `swp_entry_t` existe dans l'arbre de recherche du cache.

Si une telle correspondance existe, le contenu compressé est restauré dans la page, le contenu du cache désalloué et l'accès à la `swap device` évité.

Si aucune correspondance n'existe, alors la page se trouve sur la `swap device` et le noyau reprend son cours normal d'exécution.

1.4 Structure du cache

Nous avons organisé notre cache de manière relativement efficace.

Chaque "bin" se comporte comme un allocateur dynamique en utilisant une structuration similaire à celle décrite au cours de cet article :

http://jamesgolick.com/2013/5/15/memory-allocators-101.html

Notre implémentation permet notamment de réduire la fragmentation induite lorsque des pages compressées sont supprimées du cache de manière rapide et efficace.

Comme dit plus haut, chaque page compressée dans le cache est répertoriée dans un arbre de recherche à partir de son `swp_entry_t`, pour rendre les interceptions des pages compressées rapides.

Pour maintenir la cohérence de notre cache, nous avons envisagé de protéger les opérations sur celui-ci à l'aide d'un `spin_lock`. Malheureusement, pour une raison qui nous est inconnue, celui-ci semblait provoquer des dead-locks.