Coucou,

Ce qui suit va tenter d'exprimer ce que j'ai compris de l'allocateur de mémoire. Si quelque chose est erroné, prière de m'en informer. Je tiens absolument à comprendre de A à Z le fonctionnement d'où ce qui suit :

Lors de l'initialisation de l'allocateur, vous créez différentes choses :

kmem_free_range_list : Liste des régions libres.

kmem_used_range_list : Liste des régions utilisées.

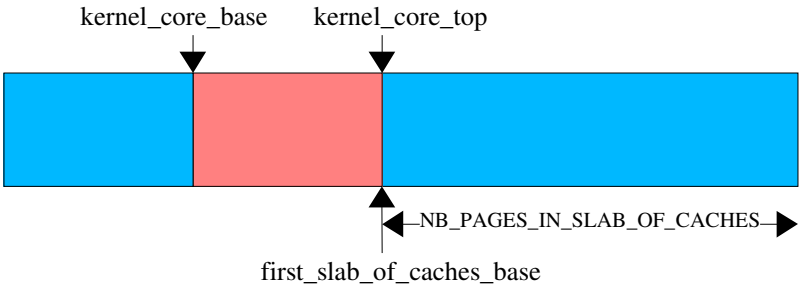
kmem_range_cache : Cache des régions mémoire. Je reviendrai dessus dans un instant.

cache_of_struct_kslab_cache : Création du premier cache des caches.

cache_of_struct_kslab : Création du premier cache des slabs.

kslab_cache_list : Liste des caches.

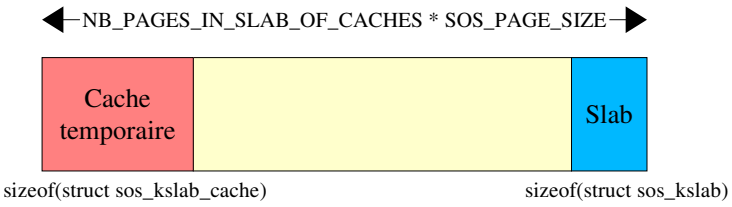
La fonction sos_kmem_cache_setup_prepare réalise les opérations suivantes :



first_slab_of_caches_base a pour adresse physique le dessus du code du noyau. Dans la première boucle for vous demandez une nouvelle page physique, puis, vous mappez cette page physique à l'adresse virtuelle first_slab_of_caches_base.
Une fois mappée, cette page est dé-référencée, elle est alors considérée comme libre, pourquoi ?
La page physique est donc consommée, mais pourtant, elle est libérée, il peut y avoir écrasement de donnée, ou sinon je ne comprend pas tout.

L'adresse virtuelle first_slab_of_caches_base est maintenant utilisée pour créer un cache grâce à la fonction create_cache_of_caches. Cette fonction va effectuer les opérations suivante :

- Initialisation (cache_initialize) d'un cache temporaire dans la pile.
- Mise à 0 de tous les octets composé par le cache à créer (memset).
- Initialisation d'un slab en fin de cache.

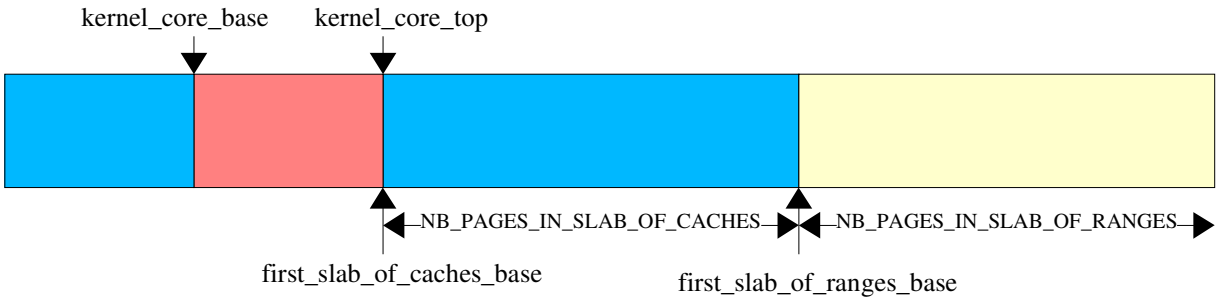


- Ajout de ce slab dans le cache temporaire. Le cache pointe alors sur ce slab et inversement, le slab pointe sur ce cache.

- Allocation d'un nouveau cache (sos_kmem_cache_alloc). Lors de l'initialisation, la liste des caches contient un élément dont le pointeur sur la liste des slabs libres est nul (champ free = NULL). Il y a donc appel à cache_grow qui va allouer un nouveau slab pour ce cache. Une région libre est alors demandée par la fonction sos_kmem_vmm_new_range. Le drapeau ON_SLAB étant levé, le slab est enregistré en fin de région puis la liste des slabs de ce cache est mise à jour. Le pointeur sur la région mémoire de ce slab est mis à jour par le pointeur sur la région allouée (new_slab->range = new_range;). Enfin, le descripteur de la page physique correspondant à l'adresse de la nouvelle région est mis à jour afin qu'il pointe sur le slab alloué. Désormais, le cache est alloué et un slab y est initialisé.
- Le descripteur de cache temporaire est alors copié dans le cache fraîchement alloué. Cette méthode permet de copier le contenu complet du descripteur de cache. L'inconvénient, est que le slab ne pointe plus sur le bon descripteur de cache. Il faut remettre ce champ à jour (slab_of_caches->cache = real_cache_of_caches;). Après cette opération, le slab est ajouté à la liste des slabs du cache. Un pointeur sur ce cache est alors retourné.

En fonction de la liste des caches (cache_of_struct_kslab_cache), un pointeur sur le premier slab créé est alors demandé (first_struct_slab_of_caches). Ce pointeur est utilisé comme sortie de la fonction.

La deuxième boucle for permet d'initialiser les plages mémoires :



Vous effectuez le même principe référencement, mappage et dé-référencement que je ne comprend toujours pas. Bref, continuons !

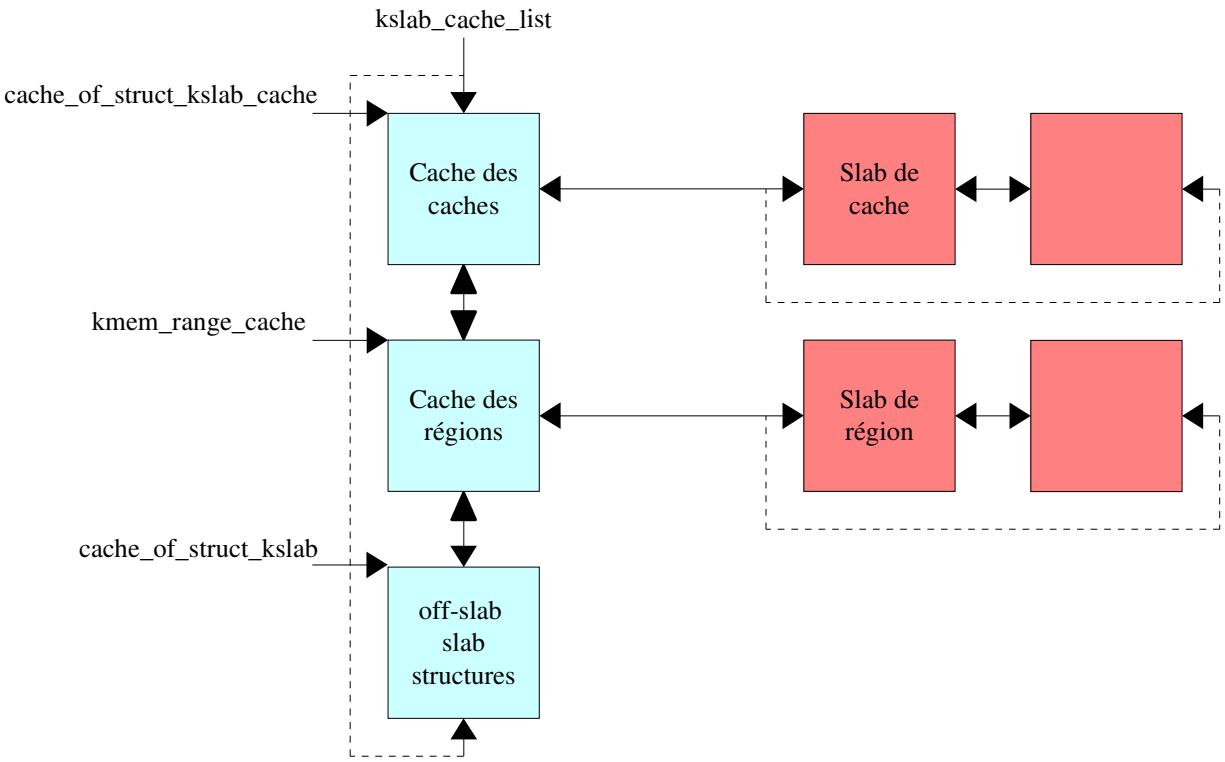
Maintenant, il y a création d'un cache de régions. Il est construit sur le même modèle que le cache précédent. Une fois ce cache alloué, il est ajouté à la liste des caches.

Le pointeur first_struct_slab_of_ranges pointe sur le slab de région créé.

Vous créez ensuite un cache appelé "off-slab slab structures" dont le pointeur sur ce cache est mémorisé dans la variable cache_of_struct_kslab.

La fonction d'initialisation des caches retourne ensuite un pointeur sur le descripteurs de cache des régions.

La représentation de l'initialisation de tout cela pourrait-être la suivante :



Dans la suite de la définition, les régions sont triées et rangées dans une des listes de régions (libre ou occupées).

Les différents pointeurs sur les descripteurs de caches (kmem_range_cache, cache_of_struct_kslab, kslab_cache_list, cache_of_struct_kslab_cache) permettent de retrouver rapidement toutes les informations concernant un cache.

Conclusion :

De rédiger ce document m'a aider à comprendre une partie de la complexité du principe mis en oeuvre. Bien entendu tout n'est pas clair, surtout concernant les slabs. J'ai encore des efforts à faire.

Concernant la liste cache_of_struct_kslab, je ne vois pas pourquoi un slab peut être OFF_SLAB ou ON_SLAB.

Je me permets d'insister sur le référencement, mappage puis dé-référencement des pages dans la fonction sos_kmem_cache_setup_prepare. Pourquoi ?

J'espère avoir été clair, Cyril du 02.