
Subject: Re: [PATCH v2 07/11] mm: Allocate kernel pages to the right memcg
Posted by [KAMEZAWA Hiroyuki](#) on Fri, 10 Aug 2012 17:33:06 GMT
[View Forum Message](#) <> [Reply to Message](#)

(2012/08/09 22:01), Glauber Costa wrote:

> When a process tries to allocate a page with the __GFP_KMEMCG flag, the
> page allocator will call the corresponding memcg functions to validate
> the allocation. Tasks in the root memcg can always proceed.
>
> To avoid adding markers to the page - and a kmem flag that would
> necessarily follow, as much as doing page_cgroup lookups for no reason,
> whoever is marking its allocations with __GFP_KMEMCG flag is responsible
> for telling the page allocator that this is such an allocation at
> free_pages() time. This is done by the invocation of
> __free_accounted_pages() and free_accounted_pages().
>
> Signed-off-by: Glauber Costa <glommer@parallels.com>
> CC: Christoph Lameter <cl@linux.com>
> CC: Pekka Enberg <penberg@cs.helsinki.fi>
> CC: Michal Hocko <mhocko@suse.cz>
> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> CC: Johannes Weiner <hannes@cmpxchg.org>
> CC: Suleiman Souhlal <suleiman@google.com>

Ah, ok. free_accounted_page() seems good.

Acked-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

I myself is okay with this. But...

Because you add a new hook to alloc_pages(), please get Ack from Mel
before requesting merge.

Thanks,
-Kame

> ---
> include/linux/gfp.h | 3 +++
> mm/page_alloc.c | 38 +++++++++++++++++++++++++++++++++++++
> 2 files changed, 41 insertions(+)
>
> diff --git a/include/linux/gfp.h b/include/linux/gfp.h
> index d8eae4d..029570f 100644
> --- a/include/linux/gfp.h
> +++ b/include/linux/gfp.h

```

> @@ -370,6 +370,9 @@ extern void free_pages(unsigned long addr, unsigned int order);
> extern void free_hot_cold_page(struct page *page, int cold);
> extern void free_hot_cold_page_list(struct list_head *list, int cold);
>
> +extern void __free_accounted_pages(struct page *page, unsigned int order);
> +extern void free_accounted_pages(unsigned long addr, unsigned int order);
> +
> #define __free_page(page) __free_pages((page), 0)
> #define free_page(addr) free_pages((addr), 0)
>
> diff --git a/mm/page_alloc.c b/mm/page_alloc.c
> index b956cec..da341dc 100644
> --- a/mm/page_alloc.c
> +++ b/mm/page_alloc.c
> @@ -2532,6 +2532,7 @@ __alloc_pages_nodemask(gfp_t gfp_mask, unsigned int order,
> struct page *page = NULL;
> int migratetype = allocflags_to_migratetype(gfp_mask);
> unsigned int cpuset_mems_cookie;
> + void *handle = NULL;
>
> gfp_mask &= gfp_allowed_mask;
>
> @@ -2543,6 +2544,13 @@ __alloc_pages_nodemask(gfp_t gfp_mask, unsigned int order,
> return NULL;
>
> /*
> + * Will only have any effect when __GFP_KMEMCG is set.
> + * This is verified in the (always inline) callee
> + */
> + if (!memcg_kmem_new_page(gfp_mask, &handle, order))
> + return NULL;
> +
> + /*
> * Check the zones suitable for the gfp_mask contain at least one
> * valid zone. It's possible to have an empty zonelist as a result
> * of GFP_THISNODE and a memoryless node
> @@ -2583,6 +2591,8 @@ out:
> if (unlikely(!put_mems_allowed(cpuset_mems_cookie) && !page))
> goto retry_cpuset;
>
> + memcg_kmem_commit_page(page, handle, order);
> +
> return page;
> }
> EXPORT_SYMBOL(__alloc_pages_nodemask);
> @@ -2635,6 +2645,34 @@ void free_pages(unsigned long addr, unsigned int order)
>
> EXPORT_SYMBOL(free_pages);

```

```

>
> +/*
> + * __free_accounted_pages and free_accounted_pages will free pages allocated
> + * with __GFP_KMEMCG.
> + *
> + * Those pages are accounted to a particular memcg, embedded in the
> + * corresponding page_cgroup. To avoid adding a hit in the allocator to search
> + * for that information only to find out that it is NULL for users who have no
> + * interest in that whatsoever, we provide these functions.
> + *
> + * The caller knows better which flags it relies on.
> + */
> +void __free_accounted_pages(struct page *page, unsigned int order)
> +{
> + memcg_kmem_free_page(page, order);
> + __free_pages(page, order);
> +}
> +EXPORT_SYMBOL(__free_accounted_pages);
> +
> +void free_accounted_pages(unsigned long addr, unsigned int order)
> +{
> + if (addr != 0) {
> + VM_BUG_ON(!virt_addr_valid((void *)addr));
> + memcg_kmem_free_page(virt_to_page((void *)addr), order);
> + __free_pages(virt_to_page((void *)addr), order);
> + }
> +}
> +EXPORT_SYMBOL(free_accounted_pages);
> +
> static void *make_alloc_exact(unsigned long addr, unsigned order, size_t size)
> {
> if (addr) {
>

```
