
Subject: Re: [PATCH 16/23] slab: provide kcalloc_no_account
Posted by [KAMEZAWA Hiroyuki](#) on Wed, 25 Apr 2012 01:44:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

(2012/04/23 8:53), Glauber Costa wrote:

> Some allocations need to be accounted to the root memcg regardless
> of their context. One trivial example, is the allocations we do
> during the memcg slab cache creation themselves. Strictly speaking,
> they could go to the parent, but it is way easier to bill them to
> the root cgroup.
>
> Only generic kcalloc allocations are allowed to be bypassed.
>
> The function is not exported, because drivers code should always
> be accounted.
>
> This code is mosly written by Suleiman Souhlal.
>
> 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>

Seems reasonable.

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

Hmm...but can't we find the 'context' in automatic way ?

-Kame

> ---
> include/linux/slab_def.h | 1 +
> mm/slab.c | 23 ++++++
> 2 files changed, 24 insertions(+), 0 deletions(-)
>
> diff --git a/include/linux/slab_def.h b/include/linux/slab_def.h
> index 06e4a3e..54d25d7 100644
> --- a/include/linux/slab_def.h
> +++ b/include/linux/slab_def.h
> @@ -114,6 +114,7 @@ extern struct cache_sizes malloc_sizes[];
>
> void *kmem_cache_alloc(struct kmem_cache *, gfp_t);
> void *__kalloc(size_t size, gfp_t flags);

```

> +void *kmalloc_no_account(size_t size, gfp_t flags);
>
> #ifdef CONFIG_TRACING
> extern void *kmem_cache_alloc_trace(size_t size,
> diff --git a/mm/slab.c b/mm/slab.c
> index c4ef684..13948c3 100644
> --- a/mm/slab.c
> +++ b/mm/slab.c
> @@ -3960,6 +3960,29 @@ void *__kmalloc(size_t size, gfp_t flags)
> }
> EXPORT_SYMBOL(__kmalloc);
>
> +static __always_inline void *__do_kmalloc_no_account(size_t size, gfp_t flags,
> + void *caller)
> +{
> + struct kmem_cache *cachep;
> + void *ret;
> +
> + cachep = __find_general_cachep(size, flags);
> + if (unlikely(ZERO_OR_NULL_PTR(cachep)))
> + return cachep;
> +
> + ret = __cache_alloc(cachep, flags, caller);
> + trace_kmalloc((unsigned long)caller, ret, size,
> + cachep->buffer_size, flags);
> +
> + return ret;
> +}
> +
> +void *kmalloc_no_account(size_t size, gfp_t flags)
> +{
> + return __do_kmalloc_no_account(size, flags,
> + __builtin_return_address(0));
> +}
> +
> void *__kmalloc_track_caller(size_t size, gfp_t flags, unsigned long caller)
> {
> return __do_kmalloc(size, flags, (void *)caller);

```
