

---

Subject: Re: [PATCH 6/7] BC: kernel memory (core)  
Posted by [Balbir Singh](#) on Tue, 29 Aug 2006 18:41:46 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

Kirill Korotaev wrote:

> Introduce BC\_KMEMSIZE resource which accounts kernel  
> objects allocated by task's request.  
>  
> Reference to BC is kept on struct page or slab object.  
> For slabs each struct slab contains a set of pointers  
> corresponding objects are charged to.  
>  
> Allocation charge rules:  
> 1. Pages - if allocation is performed with \_\_GFP\_BC flag - page  
> is charged to current's exec\_bc.  
> 2. Slabs - kmem\_cache may be created with SLAB\_BC flag - in this  
> case each allocation is charged. Caches used by kmalloc are  
> created with SLAB\_BC | SLAB\_BC\_NOCHARGE flags. In this case  
> only \_\_GFP\_BC allocations are charged.  
>

<snip>

> +#define \_\_GFP\_BC\_LIMIT ((\_\_force gfp\_t)0x100000u) /\* Charge against BC  
> limit \*/  
>

What's \_\_GFP\_BC\_LIMIT for, could you add the description for that flag?  
The comment is not very clear

> +ifdef CONFIG\_BEANCOUNTERS  
> + union {  
> + struct beancounter \*page\_bc;  
> + } bc;  
> +endif  
> };  
>  
> +#define page\_bc(page) ((page)->bc.page\_bc)

Minor comment - page->(bc).page\_bc has too many repetitions of page and bc - see  
the Practice of Programming by Kernighan and Pike

I missed the part of why you wanted to have a union (in struct page for bc)?

> const char \*bc\_rnames[] = {  
> + "kmemsiz", /\* 0 \*/  
> };  
>

```

> static struct hlist_head bc_hash[BC_HASH_SIZE];
> @@ -221,6 +222,8 @@ static void init_beancounter_syslimits(s
> {
>     int k;
>
> +    bc->bc_parms[BC_KMEMSIZE].limit = 32 * 1024 * 1024;
> +

```

Can't this be configurable CONFIG\_XXX or a #defined constant?

```

> --- ./mm/mempool.c.bckmem 2006-04-21 11:59:36.000000000 +0400
> +++ ./mm/mempool.c 2006-08-28 12:59:28.000000000 +0400
> @@ -119,6 +119,7 @@ int mempool_resize(mempool_t *pool, int unsigned
> long flags;
>
>     BUG_ON(new_min_nr <= 0);
> +    gfp_mask &= ~__GFP_BC;
>
>     spin_lock_irqsave(&pool->lock, flags);
>     if (new_min_nr <= pool->min_nr) {
> @@ -212,6 +213,7 @@ void * mempool_alloc(mempool_t *pool, gf
>     gfp_mask |= __GFP_NOMEMALLOC; /* don't allocate emergency
> reserves */
>     gfp_mask |= __GFP_NORETRY; /* don't loop in __alloc_pages */
>     gfp_mask |= __GFP_NOWARN; /* failures are OK */
> +    gfp_mask &= ~__GFP_BC; /* do not charge */
>
>     gfp_temp = gfp_mask & (~__GFP_WAIT|__GFP_IO);
>

```

Is there any reason why mempool\_xxxx() functions are not charged? Is it because mempool functions are mostly used from the I/O path?

```

> --- ./mm/page_alloc.c.bckmem 2006-08-28 12:20:13.000000000 +0400
> +++ ./mm/page_alloc.c 2006-08-28 12:59:28.000000000 +0400
> @@ -40,6 +40,8 @@
> #include <linux/sort.h>
> #include <linux/pfn.h>
>
> +#include <bc/kmem.h>
> +
> #include <asm/tlbflush.h>
> #include <asm/div64.h>
> #include "internal.h"
> @@ -516,6 +518,8 @@ static void __free_pages_ok(struct page  if
> (reserved)
>     return;
>

```

```

> + bc_page_uncharge(page, order);
> +
>   kernel_map_pages(page, 1 << order, 0);
>   local_irq_save(flags);
>   __count_vm_events(PGFREE, 1 << order);
> @@ -799,6 +803,8 @@ static void fastcall free_hot_cold_page(
>     if (free_pages_check(page))
>         return;
>
> + bc_page_uncharge(page, 0);
> +
>   kernel_map_pages(page, 1, 0);
>
>   pcp = &zone_pcp(zone, get_cpu())->pcp[cold];
> @@ -1188,6 +1194,11 @@ nopage:
>     show_mem();
>   }
> got_pg:
> + if ((gfp_mask & __GFP_BC) &&
> +     bc_page_charge(page, order, gfp_mask)) {

```

I wonder if bc\_page\_charge() should be called bc\_page\_charge\_failed()?  
 Does it make sense to atleast partially start reclamation here? I know with  
 bean counters we cannot reclaim from a particular container, but for now  
 we could kick off kswapd() or call shrink\_all\_memory() inline (Dave's patches do  
 this to shrink memory from the particular cpuset). Or do you want to leave this  
 slot open for later?

```

> +     __free_pages(page, order);
> +     page = NULL;
> + }

```

--

Balbir Singh,  
 Linux Technology Center,  
 IBM Software Labs