Subject: Re: [PATCH 06/10] sl[au]b: Allocate objects from memcg cache
Posted by Glauber Costa on Mon, 30 Jul 2012 13:11:06 GMT
View Forum Message <> Reply to Message

On 07/30/2012 04:58 PM, Kirill A. Shutemov wrote:
> On Wed, Jul 25, 2012 at 06:38:17PM +0400, Glauber Costa wrote:
>> We are able to match a cache allocation to a particular memcg. If the
>> task doesn't change groups during the allocation itself - a rare event,
>> this will give us a good picture about who is the first group to touch a
>> cache page.
>>
>> This patch uses the now available infrastructure by calling
>> memcg_kmem_get_cache() before all the cache allocations.
>>
>> 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>
>> ---
>> include/linux/slub_def.h | 18 ++++++++++++-----
>> mm/memcontrol.c | 2 ++
>> mm/slab.c | 4 ++++
>> mm/slub.c | 1 +
>> 4 files changed, 20 insertions(+), 5 deletions(-)
>>
>> diff --git a/include/linux/slub_def.h b/include/linux/slub_def.h
>> index 8bb8ad2..148000a 100644
>> --- a/include/linux/slub_def.h
>> +++ b/include/linux/slub_def.h
>> @@ -13,6 +13,8 @@
>> #include <linux/kobject.h>
>>
>> #include <linux/kmemleak.h>
>> +#include <linux/memcontrol.h>
>> +#include <linux/mm.h>
>>
>> enum stat_item {
>> ALLOC_FASTPATH, /* Allocation from cpu slab */
>> @@ -209,14 +211,14 @@ static __always_inline int kmalloc_index(size_t size)
>> * This ought to end up with a global pointer to the right cache
>> * in kmalloc_caches.
>> */
>> -static __always_inline struct kmem_cache *kmalloc_slab(size_t size)
>> +static __always_inline struct kmem_cache *kmalloc_slab(gfp_t flags, size_t size)
>> {

```
>>   int index = kmalloc_index(size);
>>
>>   if (index == 0)
>>    return NULL;
>>
>> - return kmalloc_caches[index];
>> + return memcg_kmem_get_cache(kmalloc_caches[index], flags);
>>  }
>>
>>  void *kmem_cache_alloc(struct kmem_cache *, gfp_t);
>> @@ -225,7 +227,13 @@ void *__kmalloc(size_t size, gfp_t flags);
>>  static __always_inline void *
>>  kmalloc_order(size_t size, gfp_t flags, unsigned int order)
>>  {
>> - void *ret = (void *) __get_free_pages(flags | __GFP_COMP, order);
>> + void *ret;
>> +
>> + flags = __GFP_COMP;
>> +#ifdef CONFIG_MEMCG_KMEM
>> + flags |= __GFP_KMEMCG;
>> +#endif
>
> Em.. I don't see where __GFP_KMEMCG is defined.
> It should be 0 for !CONFIG_MEMCG_KMEM.
>
```
It is not, sorry.

As I said, this is dependent on another patch series.
My main goal while sending this was to get the slab part - that will eventually come ontop of that - discussed. Because they are both quite complex, I believe they benefit from being discussed separately.

You can find the latest version of that here:

https://lkml.org/lkml/2012/6/25/251