
Subject: [PATCH 06/10] sl[au]b: Allocate objects from memcg cache
Posted by [Glauber Costa](#) on Wed, 25 Jul 2012 14:38:17 GMT
[View Forum Message](#) <> [Reply to Message](#)

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/slub.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 kmem_cache[index];
+ return memcg_kmem_get_cache(kmem_cache[index], flags);
}

void *kmem_cache_alloc(struct kmem_cache *, gfp_t);
@@ -225,7 +227,13 @@ void *__kmemalloc(size_t size, gfp_t flags);
static __always_inline void *
kmemalloc_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
+ ret = (void *) __get_free_pages(flags, order);
    kmemleak_alloc(ret, size, 1, flags);
    return ret;
}
@@ -274,7 +282,7 @@ static __always_inline void *kmemalloc(size_t size, gfp_t flags)
    return kmemalloc_large(size, flags);

    if (!(flags & SLUB_DMA)) {
- struct kmem_cache *s = kmemalloc_slab(size);
+ struct kmem_cache *s = kmemalloc_slab(flags, size);

    if (!s)
        return ZERO_SIZE_PTR;
@@ -307,7 +315,7 @@ static __always_inline void *kmemalloc_node(size_t size, gfp_t flags, int
node)
{
    if (__builtin_constant_p(size) &&
        size <= SLUB_MAX_SIZE && !(flags & SLUB_DMA)) {
- struct kmem_cache *s = kmemalloc_slab(size);
+ struct kmem_cache *s = kmemalloc_slab(flags, size);

    if (!s)
        return ZERO_SIZE_PTR;
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index b933474..2cc3acf 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -586,6 +586,8 @@ static struct kmem_cache *kmem_cache_dup(struct mem_cgroup
*memcg,

```

```

    new = kmem_cache_create_memcg(memcg, name, s->object_size, s->align,
        (s->flags & ~SLAB_PANIC), s->ctor, s);
+ if (new)
+ new->allocflags |= __GFP_KMEMCG;

    kfree(name);
    return new;
diff --git a/mm/slab.c b/mm/slab.c
index 76bc98f..ddc60a4 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -3316,6 +3316,8 @@ __cache_alloc_node(struct kmem_cache *cachep, gfp_t flags, int
nodeid,
    if (slab_should_failslab(cachep, flags))
        return NULL;

+ cachep = memcg_kmem_get_cache(cachep, flags);
+
    cache_alloc_debugcheck_before(cachep, flags);
    local_irq_save(save_flags);

@@ -3401,6 +3403,8 @@ __cache_alloc(struct kmem_cache *cachep, gfp_t flags, void *caller)
    if (slab_should_failslab(cachep, flags))
        return NULL;

+ cachep = memcg_kmem_get_cache(cachep, flags);
+
    cache_alloc_debugcheck_before(cachep, flags);
    local_irq_save(save_flags);
    objp = __do_cache_alloc(cachep, flags);
diff --git a/mm/slub.c b/mm/slub.c
index 417a806..6175a72 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -2292,6 +2292,7 @@ static __always_inline void *slab_alloc(struct kmem_cache *s,
    if (slab_pre_alloc_hook(s, gfpflags))
        return NULL;

+ s = memcg_kmem_get_cache(s, gfpflags);
redo:

/*
--
1.7.10.4

```
