Subject: [PATCH v2 09/13] memcg: Account for kmalloc in kernel memory accounting.
Posted by Suleiman Souhlal on Fri, 09 Mar 2012 20:39:12 GMT
View Forum Message <> Reply to Message

In order to do this, we have to create a kmalloc_no_account()
function that is used for kmalloc allocations that we do not
want to account, because the kernel memory accounting code has
to make some kmalloc allocations and is not allowed to recurse.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>
---
include/linux/slab.h | 8 ++++++++
mm/memcontrol.c | 2 +-
mm/slab.c | 42 ++---
3 files changed, 48 insertions(+), 4 deletions(-)

diff --git a/include/linux/slab.h b/include/linux/slab.h
index bc9f87f..906a015 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -377,6 +377,8 @@ struct kmem_cache *kmem_cache_create_memcg(struct kmem_cache *cachep,
char *name);
void kmem_cache_drop_ref(struct kmem_cache *cachep);

+void *kmalloc_no_account(size_t size, gfp_t flags);
+
#else /* !CONFIG_CGROUP_MEM_RES_CTLR_KMEM || !CONFIG_SLAB */

#define MAX_KMEM_CACHE_TYPES 0
@@ -392,6 +394,12 @@ static inline void
kmem_cache_drop_ref(struct kmem_cache *cachep)
{
}
+
+static inline void *kmalloc_no_account(size_t size, gfp_t flags)
+{
+ return kmalloc(size, flags);
+}
+
#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM && CONFIG_SLAB */

#endif /* _LINUX_SLAB_H */
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index a5593cf..72e83af 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c

```
@@ -5824,7 +5824,7 @@ memcg_create_cache_enqueue(struct mem_cgroup *memcg, struct
kmem_cache *cachep)
  if (!css_tryget(&memcg->css))
   return;

- cw = kmalloc(sizeof(struct create_work), GFP_NOWAIT);
+ cw = kmalloc_no_account(sizeof(struct create_work), GFP_NOWAIT);
  if (cw == NULL) {
   css_put(&memcg->css);
   return;
diff --git a/mm/slab.c b/mm/slab.c
index 95b024c..7af7384 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -1623,8 +1623,8 @@ void __init kmem_cache_init(void)
    sizes->cs_cachep = kmem_cache_create(names->name,
      sizes->cs_size,
      ARCH_KMALLOC_MINALIGN,
-     ARCH_KMALLOC_FLAGS|SLAB_PANIC,
-     NULL);
+     ARCH_KMALLOC_FLAGS | SLAB_PANIC |
+     SLAB_MEMCG_ACCT, NULL);
   }
 #ifdef CONFIG_ZONE_DMA
   sizes->cs_dmacachep = kmem_cache_create(
@@ -3936,11 +3936,16 @@ static __always_inline void *
 __do_kmalloc_node(size_t size, gfp_t flags, int node, void *caller)
 {
  struct kmem_cache *cachep;
+ void *ret;

  cachep = kmem_find_general_cachep(size, flags);
  if (unlikely(ZERO_OR_NULL_PTR(cachep)))
   return cachep;
- return kmem_cache_alloc_node_trace(size, cachep, flags, node);
+ cachep = mem_cgroup_get_kmem_cache(cachep, flags);
+ ret = kmem_cache_alloc_node_trace(size, cachep, flags, node);
+ mem_cgroup_put_kmem_cache(cachep);
+
+ return ret;
 }

 #if defined(CONFIG_DEBUG_SLAB) || defined(CONFIG_TRACING)
@@ -3986,14 +3991,31 @@ static __always_inline void *__do_kmalloc(size_t size, gfp_t flags,
  cachep = __find_general_cachep(size, flags);
  if (unlikely(ZERO_OR_NULL_PTR(cachep)))
   return cachep;
+ cachep = mem_cgroup_get_kmem_cache(cachep, flags);
```

```
  ret = __cache_alloc(cachep, flags, caller);

  trace_kmalloc((unsigned long) caller, ret,
         size, cachep->buffer_size, flags);
+ mem_cgroup_put_kmem_cache(cachep);

  return ret;
 }

+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;
+}

 #if defined(CONFIG_DEBUG_SLAB) || defined(CONFIG_TRACING)
 void *__kmalloc(size_t size, gfp_t flags)
@@ -4008,12 +4030,26 @@ void *__kmalloc_track_caller(size_t size, gfp_t flags, unsigned long
caller)
 }
 EXPORT_SYMBOL(__kmalloc_track_caller);

+void *
+kmalloc_no_account(size_t size, gfp_t flags)
+{
+ return __do_kmalloc_no_account(size, flags,
+      __builtin_return_address(0));
+}
+EXPORT_SYMBOL(kmalloc_no_account);
 #else
 void *__kmalloc(size_t size, gfp_t flags)
 {
  return __do_kmalloc(size, flags, NULL);
 }
 EXPORT_SYMBOL(__kmalloc);
+
+void *
+kmalloc_no_account(size_t size, gfp_t flags)
```

```
+{
+ return __do_kmalloc_no_account(size, flags, NULL);
+}
+EXPORT_SYMBOL(kmalloc_no_account);
 #endif

 /**
--
1.7.7.3
```