Subject: [PATCH v5 07/14] mm: Allocate kernel pages to the right memcg
Posted by Glauber Costa on Tue, 16 Oct 2012 10:16:44 GMT
View Forum Message <> Reply to Message

When a process tries to allocate a page with the __GFP_KMEMCG flag, the
page allocator will call the corresponding memcg functions to validate
the allocation. Tasks in the root memcg can always proceed.

To avoid adding markers to the page - and a kmem flag that would
necessarily follow, as much as doing page_cgroup lookups for no reason,
whoever is marking its allocations with __GFP_KMEMCG flag is responsible
for telling the page allocator that this is such an allocation at
free_pages() time. This is done by the invocation of
__free_accounted_pages() and free_accounted_pages().

[v2: inverted test order to avoid a memcg_get leak,
  free_accounted_pages simplification]
[v4: test for TIF_MEMDIE at newpage_charge]

Signed-off-by: Glauber Costa <glommer@parallels.com>
Acked-by: Michal Hocko <mhocko@suse.cz>
Acked-by: Mel Gorman <mgorman@suse.de>
Acked-by: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: Johannes Weiner <hannes@cmpxchg.org>
CC: Suleiman Souhlal <suleiman@google.com>
CC: Tejun Heo <tj@kernel.org>
---
 include/linux/gfp.h | 3 +++
 mm/page_alloc.c | 35 +++++++++++++++++++++++++++++++++++
 2 files changed, 38 insertions(+)

diff --git a/include/linux/gfp.h b/include/linux/gfp.h
index 9289d46..8f6fe34 100644
--- a/include/linux/gfp.h
+++ b/include/linux/gfp.h
@@ -362,6 +362,9 @@ extern void free_pages(unsigned long addr, unsigned int order);
 extern void free_hot_cold_page(struct page *page, int cold);
 extern void free_hot_cold_page_list(struct list_head *list, int cold);

+extern void __free_accounted_pages(struct page *page, unsigned int order);
+extern void free_accounted_pages(unsigned long addr, unsigned int order);
+
 #define __free_page(page) __free_pages((page), 0)
 #define free_page(addr) free_pages((addr), 0)

diff --git a/mm/page_alloc.c b/mm/page_alloc.c

```
index feddc7f..dcf33ad 100644
--- a/mm/page_alloc.c
+++ b/mm/page_alloc.c
@@ -2595,6 +2595,7 @@ __alloc_pages_nodemask(gfp_t gfp_mask, unsigned int order,
  int migratetype = allocflags_to_migratetype(gfp_mask);
  unsigned int cpuset_mems_cookie;
  int alloc_flags = ALLOC_WMARK_LOW|ALLOC_CPUSET;
+ struct mem_cgroup *memcg = NULL;

  gfp_mask &= gfp_allowed_mask;

@@ -2613,6 +2614,13 @@ __alloc_pages_nodemask(gfp_t gfp_mask, unsigned int order,
  if (unlikely(!zonelist->_zonerefs->zone))
   return NULL;

+ /*
+  * Will only have any effect when __GFP_KMEMCG is set.  This is
+  * verified in the (always inline) callee
+  */
+ if (!memcg_kmem_newpage_charge(gfp_mask, &memcg, order))
+  return NULL;
+
 retry_cpuset:
  cpuset_mems_cookie = get_mems_allowed();

@@ -2648,6 +2656,8 @@ out:
  if (unlikely(!put_mems_allowed(cpuset_mems_cookie) && !page))
   goto retry_cpuset;

+ memcg_kmem_commit_charge(page, memcg, order);
+
  return page;
 }
 EXPORT_SYMBOL(__alloc_pages_nodemask);
@@ -2700,6 +2710,31 @@ void free_pages(unsigned long addr, unsigned int order)

 EXPORT_SYMBOL(free_pages);

+/*
+ * __free_accounted_pages and free_accounted_pages will free pages allocated
+ * with __GFP_KMEMCG.
+ *
+ * Those pages are accounted to a particular memcg, embedded in the
+ * corresponding page_cgroup. To avoid adding a hit in the allocator to search
+ * for that information only to find out that it is NULL for users who have no
+ * interest in that whatsoever, we provide these functions.
+ *
+ * The caller knows better which flags it relies on.
```

```
+ */
+void __free_accounted_pages(struct page *page, unsigned int order)
+{
+ memcg_kmem_uncharge_page(page, order);
+ __free_pages(page, order);
+}
+
+void free_accounted_pages(unsigned long addr, unsigned int order)
+{
+ if (addr != 0) {
+  VM_BUG_ON(!virt_addr_valid((void *)addr));
+  __free_accounted_pages(virt_to_page((void *)addr), order);
+ }
+}
+
 static void *make_alloc_exact(unsigned long addr, unsigned order, size_t size)
 {
  if (addr) {
--
1.7.11.7
```