Subject: [PATCH v2 25/29] memcg: Track all the memcg children of a kmem_cache.
Posted by Glauber Costa on Fri, 11 May 2012 17:44:27 GMT
View Forum Message <> Reply to Message

From: Suleiman Souhlal <ssouhlal@FreeBSD.org>

This enables us to remove all the children of a kmem_cache being
destroyed, if for example the kernel module it's being used in
gets unloaded. Otherwise, the children will still point to the
destroyed parent.

We also use this to propagate /proc/slabinfo settings to all
the children of a cache, when, for example, changing its
batchsize.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>
---
 include/linux/memcontrol.h | 1 +
 include/linux/slab.h | 1 +
 mm/memcontrol.c | 9 +++++++
 mm/slab.c | 52 ++---
 4 files changed, 59 insertions(+), 4 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index 3e03f26..5a5cd42 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -465,6 +465,7 @@ extern struct static_key mem_cgroup_kmem_enabled_key;
 #define mem_cgroup_kmem_on static_key_false(&mem_cgroup_kmem_enabled_key)

 void mem_cgroup_destroy_cache(struct kmem_cache *cachep);
+void mem_cgroup_remove_child_kmem_cache(struct kmem_cache *cachep, int id);
 #else
 static inline void mem_cgroup_register_cache(struct mem_cgroup *memcg,
          struct kmem_cache *s)
diff --git a/include/linux/slab.h b/include/linux/slab.h
index d03637e..876783b 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -169,6 +169,7 @@ struct mem_cgroup_cache_params {

 #endif
  struct list_head destroyed_list; /* Used when deleting memcg cache */
+ struct list_head sibling_list;
 };
 #endif

```
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index c3772dc..933edf1 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -548,6 +548,7 @@ void mem_cgroup_register_cache(struct mem_cgroup *memcg,
  else
   INIT_LIST_HEAD(&cachep->memcg_params.destroyed_list);
  cachep->memcg_params.id = id;
+ INIT_LIST_HEAD(&cachep->memcg_params.sibling_list);
 }

 void mem_cgroup_release_cache(struct kmem_cache *cachep)
@@ -845,6 +846,14 @@ struct kmem_cache *__mem_cgroup_get_kmem_cache(struct
kmem_cache *cachep,
 }
 EXPORT_SYMBOL(__mem_cgroup_get_kmem_cache);

+void mem_cgroup_remove_child_kmem_cache(struct kmem_cache *cachep, int id)
+{
+ mutex_lock(&memcg_cache_mutex);
+ cachep->memcg_params.memcg->slabs[id] = NULL;
+ mutex_unlock(&memcg_cache_mutex);
+ mem_cgroup_put(cachep->memcg_params.memcg);
+}
+
 bool __mem_cgroup_new_kmem_page(struct page *page, gfp_t gfp)
 {
  struct mem_cgroup *memcg;
diff --git a/mm/slab.c b/mm/slab.c
index a6fd82e..cd4600d 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -2638,6 +2638,8 @@ struct kmem_cache *kmem_cache_dup(struct mem_cgroup *memcg,
  if (new == NULL)
   goto out;

+ list_add(&new->memcg_params.sibling_list,
+     &cachep->memcg_params.sibling_list);
  if ((cachep->limit != new->limit) ||
      (cachep->batchcount != new->batchcount) ||
      (cachep->shared != new->shared))
@@ -2853,6 +2855,29 @@ void kmem_cache_destroy(struct kmem_cache *cachep)
 {
  BUG_ON(!cachep || in_interrupt());

+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ /* Destroy all the children caches if we aren't a memcg cache */
+ if (cachep->memcg_params.id != -1) {
```

```
+  struct kmem_cache *c;
+  struct mem_cgroup_cache_params *p, *tmp;
+  int id = cachep->memcg_params.id;
+
+  mutex_lock(&cache_chain_mutex);
+  list_for_each_entry_safe(p, tmp,
+      &cachep->memcg_params.sibling_list, sibling_list) {
+   c = container_of(p, struct kmem_cache, memcg_params);
+   if (c == cachep)
+    continue;
+   mutex_unlock(&cache_chain_mutex);
+   BUG_ON(c->memcg_params.id != -1);
+   mem_cgroup_remove_child_kmem_cache(c, id);
+   kmem_cache_destroy(c);
+   mutex_lock(&cache_chain_mutex);
+  }
+  mutex_unlock(&cache_chain_mutex);
+ }
+#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
+
  /* Find the cache in the chain of caches. */
  get_online_cpus();
  mutex_lock(&cache_chain_mutex);
@@ -2860,6 +2885,9 @@ void kmem_cache_destroy(struct kmem_cache *cachep)
   * the chain is never empty, cache_cache is never destroyed
   */
  list_del(&cachep->next);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ list_del(&cachep->memcg_params.sibling_list);
+#endif
  if (__cache_shrink(cachep)) {
   slab_error(cachep, "Can't free all objects");
   list_add(&cachep->next, &cache_chain);
@@ -4650,11 +4678,27 @@ static ssize_t slabinfo_write(struct file *file, const char __user
*buffer,
    if (limit < 1 || batchcount < 1 ||
      batchcount > limit || shared < 0) {
     res = 0;
-   } else {
-    res = do_tune_cpucache(cachep, limit,
-            batchcount, shared,
-            GFP_KERNEL);
+    break;
    }
+
+   res = do_tune_cpucache(cachep, limit, batchcount,
+       shared, GFP_KERNEL);
+
```

```
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+   {
+    struct kmem_cache *c;
+    struct mem_cgroup_cache_params *p;
+
+    list_for_each_entry(p,
+         &cachep->memcg_params.sibling_list,
+         sibling_list) {
+     c = container_of(p, struct kmem_cache,
+          memcg_params);
+     do_tune_cpucache(c, limit, batchcount,
+          shared, GFP_KERNEL);
+    }
+   }
+#endif
    break;
  }
 }
--
1.7.7.6
```