Subject: [PATCH v2 10/13] memcg: Track all the memcg children of a kmem_cache.
Posted by Suleiman Souhlal on Fri, 09 Mar 2012 20:39:13 GMT
View Forum Message <> Reply to Message

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/slab.h | 1 +
 mm/slab.c | 53 ++---
 2 files changed, 50 insertions(+), 4 deletions(-)

diff --git a/include/linux/slab.h b/include/linux/slab.h
index 906a015..fd1d2ba 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -166,6 +166,7 @@ struct mem_cgroup_cache_params {
  struct kmem_cache *orig_cache;

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

diff --git a/mm/slab.c b/mm/slab.c
index 7af7384..02239ed 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -2511,6 +2511,7 @@ __kmem_cache_create(const char *name, size_t size, size_t align,
  cachep->memcg_params.orig_cache = NULL;
  atomic_set(&cachep->refcnt, 1);
  INIT_LIST_HEAD(&cachep->memcg_params.destroyed_list);
+ INIT_LIST_HEAD(&cachep->memcg_params.sibling_list);

  if (!memcg) {
   int id;
@@ -2582,6 +2583,8 @@ kmem_cache_create_memcg(struct kmem_cache *cachep, char
*name)
  }
  new->memcg_params.orig_cache = cachep;

```
+ 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))
@@ -2769,6 +2772,29 @@ 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;
+
+  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,
+       cachep->memcg_params.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);
@@ -2776,6 +2802,9 @@ 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);
@@ -4654,11 +4683,27 @@ 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;
   }
  }
--
1.7.7.3
```