Subject: [PATCH v3 23/28] slab: Track all the memcg children of a kmem_cache.
Posted by Glauber Costa on Fri, 25 May 2012 13:03:43 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.

Code is inspired by Suleiman's, with adaptations to
the patchset and simplifications by me.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>
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>
---
 include/linux/memcontrol.h | 1 +
 include/linux/slab.h | 1 +
 mm/memcontrol.c | 13 ++++++++-
 mm/slab.c | 61 ++-
 4 files changed, 73 insertions(+), 3 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index df049e1..63113de 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 25f073e..714aeab 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -172,6 +172,7 @@ struct mem_cgroup_cache_params {
  struct kmem_cache *parent;

```
 #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 e2576c5..08f3c3e 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -542,10 +542,11 @@ void mem_cgroup_register_cache(struct mem_cgroup *memcg,

  cachep->memcg_params.memcg = memcg;

- if (!memcg)
+ if (!memcg) {
   id = ida_simple_get(&cache_types, 0, MAX_KMEM_CACHE_TYPES,
         GFP_KERNEL);
- else
+ INIT_LIST_HEAD(&cachep->memcg_params.sibling_list);
+ } else
   INIT_LIST_HEAD(&cachep->memcg_params.destroyed_list);
  cachep->memcg_params.id = id;
 }
@@ -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 59f1027..cb409ae 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -2661,6 +2661,8 @@ struct kmem_cache *kmem_cache_dup(struct mem_cgroup *memcg,
   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))
@@ -2829,6 +2831,33 @@ int kmem_cache_shrink(struct kmem_cache *cachep)
 }
 EXPORT_SYMBOL(kmem_cache_shrink);

+static void kmem_cache_destroy_memcg_children(struct kmem_cache *cachep)
+{
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ struct kmem_cache *c;
+ struct mem_cgroup_cache_params *p, *tmp;
+ int id = cachep->memcg_params.id;
+
+ if (id == -1)
+  return;
+
+ 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 (WARN_ON(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 */
+}
+
 /**
  * kmem_cache_destroy - delete a cache
  * @cachep: the cache to destroy
@@ -2849,6 +2878,9 @@ void kmem_cache_destroy(struct kmem_cache *cachep)
 {
  BUG_ON(!cachep || in_interrupt());

+ /* Destroy all the children caches if we aren't a memcg cache */
+ kmem_cache_destroy_memcg_children(cachep);
+
  /* Find the cache in the chain of caches. */
  get_online_cpus();
```

```
  mutex_lock(&cache_chain_mutex);
@@ -2869,6 +2901,7 @@ void kmem_cache_destroy(struct kmem_cache *cachep)

  mem_cgroup_release_cache(cachep);
 #ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ list_del(&cachep->memcg_params.sibling_list);
  /* memcg cache: free the name string. Doing it here saves us
   * a pointer to it outside the slab code */
  if (cachep->memcg_params.id == -1)
@@ -4243,7 +4276,7 @@ static void do_ccupdate_local(void *info)
 }

 /* Always called with the cache_chain_mutex held */
-static int do_tune_cpucache(struct kmem_cache *cachep, int limit,
+static int __do_tune_cpucache(struct kmem_cache *cachep, int limit,
     int batchcount, int shared, gfp_t gfp)
 {
  struct ccupdate_struct *new;
@@ -4286,6 +4319,32 @@ static int do_tune_cpucache(struct kmem_cache *cachep, int limit,
  return alloc_kmemlist(cachep, gfp);
 }

+static int do_tune_cpucache(struct kmem_cache *cachep, int limit,
+    int batchcount, int shared, gfp_t gfp)
+{
+ int ret;
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ struct kmem_cache *c;
+ struct mem_cgroup_cache_params *p;
+#endif
+
+ ret = __do_tune_cpucache(cachep, limit, batchcount, shared, gfp);
+#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+ if (g_cpucache_up < FULL)
+  return ret;
+
+ if ((ret < 0) || (cachep->memcg_params.id == -1))
+  return ret;
+
+ list_for_each_entry(p, &cachep->memcg_params.sibling_list, sibling_list) {
+  c = container_of(p, struct kmem_cache, memcg_params);
+  /* return value determined by the parent cache only */
+  __do_tune_cpucache(c, limit, batchcount, shared, gfp);
+ }
+#endif
+ return ret;
+}
+
```

```
/* Called with cache_chain_mutex held always */
static int enable_cpucache(struct kmem_cache *cachep, gfp_t gfp)
{
```
--
1.7.7.6