Subject: [PATCH v5 15/18] Aggregate memcg cache values in slabinfo
Posted by Glauber Costa on Fri, 19 Oct 2012 14:20:39 GMT
View Forum Message <> Reply to Message

When we create caches in memcgs, we need to display their usage
information somewhere. We'll adopt a scheme similar to /proc/meminfo,
with aggregate totals shown in the global file, and per-group
information stored in the group itself.

For the time being, only reads are allowed in the per-group cache.

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>
CC: Suleiman Souhlal <suleiman@google.com>
CC: Tejun Heo <tj@kernel.org>
---
 include/linux/memcontrol.h | 8 ++++++++
 include/linux/slab.h | 4 ++++
 mm/memcontrol.c | 27 +++++++++++++++++++++++++++-
 mm/slab.h | 27 +++++++++++++++++++++++++++
 mm/slab_common.c | 42 ++----
 5 files changed, 103 insertions(+), 5 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index 5c1d234..14def0b 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -404,6 +404,11 @@ void sock_update_memcg(struct sock *sk);
 void sock_release_memcg(struct sock *sk);

 extern struct static_key memcg_kmem_enabled_key;
+
+extern int memcg_limited_groups_array_size;
+#define for_each_memcg_cache_index(_idx) \
+ for ((_idx) = 0; i < memcg_limited_groups_array_size; (_idx)++)
+
 static inline bool memcg_kmem_enabled(void)
 {
  return static_key_false(&memcg_kmem_enabled_key);
@@ -537,6 +542,9 @@ static inline void sock_release_memcg(struct sock *sk)
 {
 }

+#define for_each_memcg_cache_index(_idx) \

```
+ for (; NULL; )
+
 static inline bool memcg_kmem_enabled(void)
 {
  return false;
diff --git a/include/linux/slab.h b/include/linux/slab.h
index 4a3a749..b521426 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -219,6 +219,10 @@ struct memcg_cache_params {

 int memcg_update_all_caches(int num_memcgs);

+struct seq_file;
+int cache_show(struct kmem_cache *s, struct seq_file *m);
+void print_slabinfo_header(struct seq_file *m);
+
 /*
  * Common kmalloc functions provided by all allocators
  */
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index c7732fa..e7f3458 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -551,7 +551,7 @@ static void disarm_sock_keys(struct mem_cgroup *memcg)
 #endif

 #ifdef CONFIG_MEMCG_KMEM
-static int memcg_limited_groups_array_size;
+int memcg_limited_groups_array_size;
 #define MEMCG_CACHES_MIN_SIZE 64
 /*
  * MAX_SIZE should be as large as the number of css_ids. Ideally, we could get
@@ -2724,6 +2724,27 @@ static inline bool memcg_can_account_kmem(struct mem_cgroup
*memcg)
   (memcg->kmem_account_flags & KMEM_ACCOUNTED_MASK);
 }

+
+static int mem_cgroup_slabinfo_read(struct cgroup *cont, struct cftype *cft,
+     struct seq_file *m)
+{
+ struct mem_cgroup *memcg = mem_cgroup_from_cont(cont);
+ struct kmem_cache *s;
+
+ if (!memcg_can_account_kmem(memcg))
+  return -EIO;
+
```

```
+ print_slabinfo_header(m);
+
+ mutex_lock(&memcg->slab_caches_mutex);
+ list_for_each_entry(s, &memcg->memcg_slab_caches, list)
+  cache_show(s, m);
+
+ mutex_unlock(&memcg->slab_caches_mutex);
+
+ return 0;
+}
+
 static int memcg_charge_kmem(struct mem_cgroup *memcg, gfp_t gfp, u64 size)
 {
  struct res_counter *fail_res;
@@ -5687,6 +5708,10 @@ static struct cftype mem_cgroup_files[] = {
   .trigger = mem_cgroup_reset,
   .read = mem_cgroup_read,
  },
+ {
+  .name = "kmem.slabinfo",
+  .read_seq_string = mem_cgroup_slabinfo_read,
+ },
 #endif
  { }, /* terminate */
 };
diff --git a/mm/slab.h b/mm/slab.h
index ab57462..c60f649 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -131,6 +131,23 @@ static inline bool slab_equal_or_root(struct kmem_cache *s,
  return (p == s) ||
   (s->memcg_params && (p == s->memcg_params->root_cache));
 }
+
+/*
+ * We use suffixes to the name in memcg because we can't have caches
+ * created in the system with the same name. But when we print them
+ * locally, better refer to them with the base name
+ */
+static inline const char *cache_name(struct kmem_cache *s)
+{
+ if (!is_root_cache(s))
+  return s->memcg_params->root_cache->name;
+ return s->name;
+}
+
+static inline struct kmem_cache *cache_from_memcg(struct kmem_cache *s, int idx)
+{
```

```
+ return s->memcg_params->memcg_caches[idx];
+}
 #else
 static inline bool is_root_cache(struct kmem_cache *s)
 {
@@ -156,5 +173,15 @@ static inline bool slab_equal_or_root(struct kmem_cache *s,
 {
  return true;
 }
+
+static inline const char *cache_name(struct kmem_cache *s)
+{
+ return s->name;
+}
+
+static inline struct kmem_cache *cache_from_memcg(struct kmem_cache *s, int idx)
+{
+ return NULL;
+}
 #endif
 #endif
diff --git a/mm/slab_common.c b/mm/slab_common.c
index c02faf5..7e6d503 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -247,7 +247,7 @@ int slab_is_available(void)
 }

 #ifdef CONFIG_SLABINFO
-static void print_slabinfo_header(struct seq_file *m)
+void print_slabinfo_header(struct seq_file *m)
 {
  /*
   * Output format version, so at least we can change it
@@ -291,16 +291,43 @@ static void s_stop(struct seq_file *m, void *p)
  mutex_unlock(&slab_mutex);
 }

-static int s_show(struct seq_file *m, void *p)
+static void
+memcg_accumulate_slabinfo(struct kmem_cache *s, struct slabinfo *info)
+{
+ struct kmem_cache *c;
+ struct slabinfo sinfo;
+ int i;
+
+ if (!is_root_cache(s))
+  return;
```

```
+
+ for_each_memcg_cache_index(i) {
+  c = cache_from_memcg(s, i);
+  if (!c)
+   continue;
+
+  memset(&sinfo, 0, sizeof(sinfo));
+  get_slabinfo(c, &sinfo);
+
+  info->active_slabs += sinfo.active_slabs;
+  info->num_slabs += sinfo.num_slabs;
+  info->shared_avail += sinfo.shared_avail;
+  info->active_objs += sinfo.active_objs;
+  info->num_objs += sinfo.num_objs;
+ }
+}
+
+int cache_show(struct kmem_cache *s, struct seq_file *m)
 {
- struct kmem_cache *s = list_entry(p, struct kmem_cache, list);
  struct slabinfo sinfo;

  memset(&sinfo, 0, sizeof(sinfo));
  get_slabinfo(s, &sinfo);

+ memcg_accumulate_slabinfo(s, &sinfo);
+
  seq_printf(m, "%-17s %6lu %6lu %6u %4u %4d",
-     s->name, sinfo.active_objs, sinfo.num_objs, s->size,
+     cache_name(s), sinfo.active_objs, sinfo.num_objs, s->size,
      sinfo.objects_per_slab, (1 << sinfo.cache_order));

  seq_printf(m, " : tunables %4u %4u %4u",
@@ -312,6 +339,13 @@ static int s_show(struct seq_file *m, void *p)
  return 0;
 }

+static int s_show(struct seq_file *m, void *p)
+{
+ struct kmem_cache *s = list_entry(p, struct kmem_cache, list);
+
+ return cache_show(s, m);
+}
+
 /*
  * slabinfo_op - iterator that generates /proc/slabinfo
  *
--
```

1.7.11.7