
Subject: [PATCH 09/10] memcg: Per-memcg memory.kmem.slabinfo file.

Posted by [Suleiman Souhlal](#) on Mon, 27 Feb 2012 22:58:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

This file shows all the kmem_caches used by a memcg.

Signed-off-by: Suleiman Souhlal <suleiman@google.com>

```
---
include/linux/slab_def.h | 7 ++++
mm/memcontrol.c          | 22 ++++++++
mm/slab.c                | 88 +++++++++++++++++++++++++++++++++++++-----
3 files changed, 94 insertions(+), 23 deletions(-)

diff --git a/include/linux/slab_def.h b/include/linux/slab_def.h
index 185e4a2..407eaf5 100644
--- a/include/linux/slab_def.h
+++ b/include/linux/slab_def.h
@@ -242,6 +242,7 @@ struct kmem_cache *kmem_cache_create_memcg(struct kmem_cache
 *cachep,
     char *name);
void kmem_cache_destroy_cpuset(struct kmem_cache *cachep);
void kmem_cache_drop_ref(struct kmem_cache *cachep);
+int mem_cgroup_slabinfo(struct mem_cgroup *mem, struct seq_file *m);

static inline void
kmem_cache_get_ref(struct kmem_cache *cachep)
@@ -301,6 +302,12 @@ static inline void
mem_cgroup_kmem_cache_finish_sleep(struct kmem_cache *cachep)
{
}
+
+static inline int
+memcg_slabinfo(void *unused, struct seq_file *m)
+{
+ return 0;
+}
#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */

#endif /* _LINUX_SLAB_DEF_H */
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index d4cdb8e..d4a6053 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -4637,6 +4637,22 @@ static int mem_cgroup_independent_kmem_limit_write(struct cgroup
 *cgrp,
     return 0;
}
```

```

+ #ifdef CONFIG_SLAB
+ static int
+ mem_cgroup_slabinf_show(struct cgroup *cgroup, struct cftype *ctf,
+   struct seq_file *m)
+ {
+   struct mem_cgroup *mem;
+
+   mem = mem_cgroup_from_cont(cgroup);
+
+   if (mem == root_mem_cgroup)
+     mem = NULL;
+
+   return mem_cgroup_slabinf(mem, m);
+ }
+ #endif
+
+ static struct cftype kmem_cgroup_files[] = {
+   {
+     .name = "kmem.independent_kmem_limit",
@@ -4654,6 +4670,12 @@ static struct cftype kmem_cgroup_files[] = {
+     .private = MEMFILE_PRIVATE(_KMEM, RES_USAGE),
+     .read_u64 = mem_cgroup_read,
+   },
+ #ifdef CONFIG_SLAB
+   {
+     .name = "kmem.slabinf",
+     .read_seq_string = mem_cgroup_slabinf_show,
+   },
+ #endif
+ };

static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
diff --git a/mm/slab.c b/mm/slab.c
index bf38af6..6d9f069 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -4498,21 +4498,26 @@ static void s_stop(struct seq_file *m, void *p)
+   mutex_unlock(&cache_chain_mutex);
+ }

-static int s_show(struct seq_file *m, void *p)
- {
-   struct kmem_cache *cachep = list_entry(p, struct kmem_cache, next);
-   struct slab *slabp;
+ struct slab_counts {
+   unsigned long active_objs;
+   unsigned long active_slabs;
+   unsigned long num_slabs;

```

```

+ unsigned long free_objects;
+ unsigned long shared_avail;
+ unsigned long num_objs;
- unsigned long active_slabs = 0;
- unsigned long num_slabs, free_objects = 0, shared_avail = 0;
- const char *name;
- char *error = NULL;
- int node;
+};
+
+static char *
+get_slab_counts(struct kmem_cache *cachep, struct slab_counts *c)
+{
+    struct kmem_list3 *l3;
+    struct slab *slabp;
+    char *error;
+    int node;
+
+    error = NULL;
+    memset(c, 0, sizeof(struct slab_counts));

- active_objs = 0;
- num_slabs = 0;
    for_each_online_node(node) {
        l3 = cachep->nodelists[node];
        if (!l3)
@@ -4524,31 +4529,43 @@ static int s_show(struct seq_file *m, void *p)
        list_for_each_entry(slabp, &l3->slabs_full, list) {
            if (slabp->inuse != cachep->num && !error)
                error = "slabs_full accounting error";
- active_objs += cachep->num;
- active_slabs++;
+ c->active_objs += cachep->num;
+ c->active_slabs++;
        }
        list_for_each_entry(slabp, &l3->slabs_partial, list) {
            if (slabp->inuse == cachep->num && !error)
                error = "slabs_partial inuse accounting error";
            if (!slabp->inuse && !error)
                error = "slabs_partial/inuse accounting error";
- active_objs += slabp->inuse;
- active_slabs++;
+ c->active_objs += slabp->inuse;
+ c->active_slabs++;
        }
        list_for_each_entry(slabp, &l3->slabs_free, list) {
            if (slabp->inuse && !error)
                error = "slabs_free/inuse accounting error";

```

```

- num_slabs++;
+ c->num_slabs++;
}
- free_objects += l3->free_objects;
+ c->free_objects += l3->free_objects;
  if (l3->shared)
- shared_avail += l3->shared->avail;
+ c->shared_avail += l3->shared->avail;

  spin_unlock_irq(&l3->list_lock);
}
- num_slabs += active_slabs;
- num_objs = num_slabs * cachep->num;
- if (num_objs - active_objs != free_objects && !error)
+ c->num_slabs += c->active_slabs;
+ c->num_objs = c->num_slabs * cachep->num;
+
+ return error;
+}
+
+static int s_show(struct seq_file *m, void *p)
+{
+ struct kmem_cache *cachep = list_entry(p, struct kmem_cache, next);
+ struct slab_counts c;
+ const char *name;
+ char *error;
+
+ error = get_slab_counts(cachep, &c);
+ if (c.num_objs - c.active_objs != c.free_objects && !error)
+   error = "free_objects accounting error";

  name = cachep->name;
@@ -4556,12 +4573,12 @@ static int s_show(struct seq_file *m, void *p)
  printk(KERN_ERR "slab: cache %s error: %s\n", name, error);

  seq_printf(m, "%-17s %6lu %6lu %6u %4u %4d",
-   name, active_objs, num_objs, cachep->buffer_size,
+   name, c.active_objs, c.num_objs, cachep->buffer_size,
    cachep->num, (1 << cachep->gfporder));
  seq_printf(m, " : tunables %4u %4u %4u",
    cachep->limit, cachep->batchcount, cachep->shared);
  seq_printf(m, " : slabdata %6lu %6lu %6lu",
-   active_slabs, num_slabs, shared_avail);
+   c.active_slabs, c.num_slabs, c.shared_avail);
#ifdef STATS
  { /* list3 stats */
    unsigned long high = cachep->high_mark;
@@ -4692,6 +4709,31 @@ static const struct file_operations proc_slabinfo_operations = {

```

```

.release = seq_release,
};

#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+int
+mem_cgroup_slabinfo(struct mem_cgroup *mem, struct seq_file *m)
+{
+ struct kmem_cache *cachep;
+ struct slab_counts c;
+
+ seq_printf(m, "# name          <active_objs> <num_objs> <objsize>\n");
+
+ mutex_lock(&cache_chain_mutex);
+ list_for_each_entry(cachep, &cache_chain, next) {
+ if (cachep->memcg != mem)
+ continue;
+
+ get_slab_counts(cachep, &c);
+
+ seq_printf(m, "%-17s %6lu %6lu %6u\n", cachep->name,
+ c.active_objs, c.num_objs, cachep->buffer_size);
+ }
+ mutex_unlock(&cache_chain_mutex);
+
+ return 0;
+}
#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
+
#ifdef CONFIG_DEBUG_SLAB_LEAK

static void *leaks_start(struct seq_file *m, loff_t *pos)
--

```

1.7.7.3
