
Subject: [PATCH v2 12/13] memcg: Per-memcg memory.kmem.slabinfo file.
Posted by [Suleiman Souhlal](#) on Fri, 09 Mar 2012 20:39:15 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.h      |  6 +++
include/linux/slab_def.h  |  1 +
mm/memcontrol.c           | 18 ++++++++
mm/slab.c                 | 88 ++++++++++++++++++++++++++++++++++++++-----
4 files changed, 90 insertions(+), 23 deletions(-)
```

```
diff --git a/include/linux/slab.h b/include/linux/slab.h
index fd1d2ba..0ff5ee2 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -401,6 +401,12 @@ static inline void *kmalloc_no_account(size_t size, gfp_t flags)
     return kmalloc(size, flags);
 }
```

```
+static inline int
+mem_cgroup_slabinfo(struct mem_cgroup *mem, struct seq_file *m)
+{
+ return 0;
+}
+
#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM && CONFIG_SLAB */
```

```
#endif /* _LINUX_SLAB_H */
diff --git a/include/linux/slab_def.h b/include/linux/slab_def.h
index 248b8a9..fa6b272 100644
--- a/include/linux/slab_def.h
+++ b/include/linux/slab_def.h
@@ -227,6 +227,7 @@ found:
#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
```

```
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)
```

```
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 9f5e9d8..4a4fa48 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -4672,6 +4672,20 @@ static int mem_cgroup_independent_kmem_limit_write(struct cgroup
```

```

*cgrp,
    return 0;
}

+static int
+mem_cgroup_slabinfo_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_slabinfo(mem, m);
+}
+
+static struct cftype kmem_cgroup_files[] = {
+ {
+ .name = "kmem.independent_kmem_limit",
@@ -4689,6 +4703,10 @@ static struct cftype kmem_cgroup_files[] = {
+ .private = MEMFILE_PRIVATE(_KMEM, RES_USAGE),
+ .read_u64 = mem_cgroup_read,
+ },
+ {
+ .name = "kmem.slabinfo",
+ .read_seq_string = mem_cgroup_slabinfo_show,
+ },
+ };

static int register_kmem_files(struct cgroup *cont, struct cgroup_subsys *ss)
diff --git a/mm/slab.c b/mm/slab.c
index 02239ed..1b35799 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -4528,21 +4528,26 @@ 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)
@@ -4554,31 +4559,43 @@ 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;
@@ -4586,12 +4603,12 @@ 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;
@@ -4726,6 +4743,31 @@ static const struct file_operations proc_slabinf

```

```

.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_params.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
