
Subject: [PATCH 22/23] memcg: Per-memcg memory.kmem.slabinfo file.

Posted by [Glauber Costa](#) on Sun, 22 Apr 2012 23:53:39 GMT

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

From: Suleiman Souhlal <ssouhlal@FreeBSD.org>

This file shows all the kmem_caches used by a memcg.

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

```
include/linux/slab.h |  1 +
mm/memcontrol.c    | 17 ++++++++
mm/slab.c          | 88 ++++++++++++++++++++++++++++++-----
mm/slub.c          |  5 +++
4 files changed, 88 insertions(+), 23 deletions(-)
```

diff --git a/include/linux/slab.h b/include/linux/slab.h

index 0dc49fa..6932205 100644

--- a/include/linux/slab.h

+++ b/include/linux/slab.h

```
@@ -327,6 +327,7 @@ extern void * __kmalloc_track_caller(size_t, gfp_t, unsigned long);
extern struct kmem_cache *kmem_cache_dup(struct mem_cgroup *memcg,
    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);
```

```
#else
```

```
#define MAX_KMEM_CACHE_TYPES 0
```

```
#endif /* CONFIG_CGROUP_MEM_RES_CTLR_KMEM */
```

diff --git a/mm/memcontrol.c b/mm/memcontrol.c

index 547b632..46ebd11 100644

--- a/mm/memcontrol.c

+++ b/mm/memcontrol.c

```
@@ -5092,6 +5092,19 @@ static int mem_control numa_stat_open(struct inode *unused, struct
file *file)
#endif /* CONFIG_NUMA */
```

```
#ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
```

```
+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.limit_in_bytes",
@@ -5116,6 +5129,10 @@ static struct cftype kmem_cgroup_files[] = {
    .trigger = mem_cgroup_reset,
    .read = mem_cgroup_read,
},
+
{
    .name = "kmem.slabinfo",
    .read_seq_string = mem_cgroup_slabinfo_show,
},
{},
};

diff --git a/mm/slab.c b/mm/slab.c
index 86f2275..3e13fef 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -4518,21 +4519,26 @@ 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)
@@ -4544,31 +4550,43 @@ 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;
@@ -4576,12 +4594,12 @@ 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);
#endif STATS
{ /* list3 stats */
  unsigned long high = cachep->high_mark;
@@ -4615,6 +4633,30 @@ static int s_show(struct seq_file *m, void *p)
  return 0;
}

+ifdef CONFIG_CGROUP_MEM_RES_CTLR_KMEM
+int mem_cgroup_slabinfo(struct mem_cgroup *memcg, 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 != memcg)
+     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 */
+
/*
 * slabinfo_op - iterator that generates /proc/slabinfo
 */

diff --git a/mm/slub.c b/mm/slub.c
index 9b22139..1031d4d 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ @ -4147,6 +4147,11 @@ void kmem_cache_drop_ref(struct kmem_cache *s)
    BUG_ON(s->memcg_params.id != -1);
    kmem_cache_destroy(s);
}
+
+int mem_cgroup_slabinfo(struct mem_cgroup *memcg, struct seq_file *m)
+{
+ return 0;
+}
#endif

#ifndef CONFIG_SMP
--
```

1.7.7.6
