Subject: [PATCH v4 04/19] sl[au]b: process slabinfo_show in common code
Posted by Glauber Costa on Fri, 12 Oct 2012 13:40:58 GMT
View Forum Message <> Reply to Message

With all the infrastructure in place, we can now have slabinfo_show
done from slab_common.c. A cache-specific function is called to grab
information about the cache itself, since that is still heavily
dependent on the implementation. But with the values produced by it, all
the printing and handling is done from common code.

[v2: moved objects_per_slab and cache_order to slabinfo]

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: David Rientjes <rientjes@google.com>
---
 mm/slab.c | 26 ++++++++++++++++++----------
 mm/slab.h | 16 ++++++++++++++-
 mm/slab_common.c | 18 ++++++++++++++++++-
 mm/slub.c | 24 ++++++++++++++----------
 4 files changed, 57 insertions(+), 27 deletions(-)

diff --git a/mm/slab.c b/mm/slab.c
index 864a9e9..98b3460 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -4262,9 +4262,8 @@ out:
 }

 #ifdef CONFIG_SLABINFO
-int slabinfo_show(struct seq_file *m, void *p)
+void get_slabinfo(struct kmem_cache *cachep, struct slabinfo *sinfo)
 {
- struct kmem_cache *cachep = list_entry(p, struct kmem_cache, list);
  struct slab *slabp;
  unsigned long active_objs;
  unsigned long num_objs;
@@ -4319,13 +4318,20 @@ int slabinfo_show(struct seq_file *m, void *p)
  if (error)
   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->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);
+ sinfo->active_objs = active_objs;
+ sinfo->num_objs = num_objs;
+ sinfo->active_slabs = active_slabs;
+ sinfo->num_slabs = num_slabs;
+ sinfo->shared_avail = shared_avail;
+ sinfo->limit = cachep->limit;
+ sinfo->batchcount = cachep->batchcount;
+ sinfo->shared = cachep->shared;
+ sinfo->objects_per_slab = cachep->num;
+ sinfo->cache_order = cachep->gfporder;
+}
+
+void slabinfo_show_stats(struct seq_file *m, struct kmem_cache *cachep)
+{
 #if STATS
  {   /* list3 stats */
   unsigned long high = cachep->high_mark;
@@ -4355,8 +4361,6 @@ int slabinfo_show(struct seq_file *m, void *p)
       allochit, allocmiss, freehit, freemiss);
  }
 #endif
- seq_putc(m, '\n');
- return 0;
 }

 #define MAX_SLABINFO_WRITE 128
diff --git a/mm/slab.h b/mm/slab.h
index e9ba23f..66a62d3 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -74,8 +74,22 @@ int __kmem_cache_shutdown(struct kmem_cache *);

 struct seq_file;
 struct file;
-int slabinfo_show(struct seq_file *m, void *p);

+struct slabinfo {
+ unsigned long active_objs;
+ unsigned long num_objs;
+ unsigned long active_slabs;
+ unsigned long num_slabs;
+ unsigned long shared_avail;
+ unsigned int limit;
+ unsigned int batchcount;
+ unsigned int shared;
+ unsigned int objects_per_slab;
+ unsigned int cache_order;
```

```
+};
+
+void get_slabinfo(struct kmem_cache *s, struct slabinfo *sinfo);
+void slabinfo_show_stats(struct seq_file *m, struct kmem_cache *s);
 ssize_t slabinfo_write(struct file *file, const char __user *buffer,
         size_t count, loff_t *ppos);
 #endif
diff --git a/mm/slab_common.c b/mm/slab_common.c
index bb4d751..1ee1d6f 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -246,7 +246,23 @@ static void s_stop(struct seq_file *m, void *p)

 static int s_show(struct seq_file *m, void *p)
 {
- return slabinfo_show(m, p);
+ struct kmem_cache *s = list_entry(p, struct kmem_cache, list);
+ struct slabinfo sinfo;
+
+ memset(&sinfo, 0, sizeof(sinfo));
+ get_slabinfo(s, &sinfo);
+
+ seq_printf(m, "%-17s %6lu %6lu %6u %4u %4d",
+     s->name, sinfo.active_objs, sinfo.num_objs, s->size,
+     sinfo.objects_per_slab, (1 << sinfo.cache_order));
+
+ seq_printf(m, " : tunables %4u %4u %4u",
+     sinfo.limit, sinfo.batchcount, sinfo.shared);
+ seq_printf(m, " : slabdata %6lu %6lu %6lu",
+     sinfo.active_slabs, sinfo.num_slabs, sinfo.shared_avail);
+ slabinfo_show_stats(m, s);
+ seq_putc(m, '\n');
+ return 0;
 }

 /*
diff --git a/mm/slub.c b/mm/slub.c
index 91e1f3b..a34548e 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -5394,18 +5394,14 @@ __initcall(slab_sysfs_init);
  * The /proc/slabinfo ABI
  */
 #ifdef CONFIG_SLABINFO
-int slabinfo_show(struct seq_file *m, void *p)
+void get_slabinfo(struct kmem_cache *s, struct slabinfo *sinfo)
 {
  unsigned long nr_partials = 0;
```

```
  unsigned long nr_slabs = 0;
- unsigned long nr_inuse = 0;
  unsigned long nr_objs = 0;
  unsigned long nr_free = 0;
- struct kmem_cache *s;
  int node;

- s = list_entry(p, struct kmem_cache, list);
-
  for_each_online_node(node) {
   struct kmem_cache_node *n = get_node(s, node);

@@ -5418,16 +5414,16 @@ int slabinfo_show(struct seq_file *m, void *p)
   nr_free += count_partial(n, count_free);
  }

- nr_inuse = nr_objs - nr_free;
+ sinfo->active_objs = nr_objs - nr_free;
+ sinfo->num_objs = nr_objs;
+ sinfo->active_slabs = nr_slabs;
+ sinfo->num_slabs = nr_slabs;
+ sinfo->objects_per_slab = oo_objects(s->oo);
+ sinfo->cache_order = oo_order(s->oo);
+}

- seq_printf(m, "%-17s %6lu %6lu %6u %4u %4d", s->name, nr_inuse,
-     nr_objs, s->size, oo_objects(s->oo),
-     (1 << oo_order(s->oo)));
- seq_printf(m, " : tunables %4u %4u %4u", 0, 0, 0);
- seq_printf(m, " : slabdata %6lu %6lu %6lu", nr_slabs, nr_slabs,
-     0UL);
- seq_putc(m, '\n');
- return 0;
+void slabinfo_show_stats(struct seq_file *m, struct kmem_cache *s)
+{
 }

 ssize_t slabinfo_write(struct file *file, const char __user *buffer,
--
1.7.11.4
```