Subject: [PATCH v3 06/28] slab: use obj_size field of struct kmem_cache when not debugging
Posted by Glauber Costa on Fri, 25 May 2012 13:03:26 GMT
View Forum Message <> Reply to Message

The kmem controller needs to keep track of the object size of
a cache so it can later on create a per-memcg duplicate. Logic
to keep track of that already exists, but it is only enable while
debugging.

This patch makes it also available when the kmem controller code
is compiled in.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
---
 include/linux/slab_def.h | 4 +++-
 mm/slab.c | 37 ++++++++++++++++++++++++++-----------
 2 files changed, 29 insertions(+), 12 deletions(-)

diff --git a/include/linux/slab_def.h b/include/linux/slab_def.h
index d41effe..cba3139 100644
--- a/include/linux/slab_def.h
+++ b/include/linux/slab_def.h
@@ -78,8 +78,10 @@ struct kmem_cache {
    * variables contain the offset to the user object and its size.
    */
   int obj_offset;
- int obj_size;
 #endif /* CONFIG_DEBUG_SLAB */
+#if defined(CONFIG_DEBUG_SLAB) || defined(CONFIG_CGROUP_MEM_RES_CTLR_KMEM)
+ int obj_size;
+#endif

 /* 6) per-cpu/per-node data, touched during every alloc/free */
   /*
diff --git a/mm/slab.c b/mm/slab.c
index 1057a32..41345f6 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -413,8 +413,28 @@ static void kmem_list3_init(struct kmem_list3 *parent)
 #define STATS_INC_FREEMISS(x) do { } while (0)
 #endif

-#if DEBUG
+#if defined(CONFIG_DEBUG_SLAB) || defined(CONFIG_CGROUP_MEM_RES_CTLR_KMEM)
+static int obj_size(struct kmem_cache *cachep)

```
+{
+ return cachep->obj_size;
+}
+static void set_obj_size(struct kmem_cache *cachep, int size)
+{
+ cachep->obj_size = size;
+}
+
+#else
+static int obj_size(struct kmem_cache *cachep)
+{
+ return cachep->buffer_size;
+}
+
+static void set_obj_size(struct kmem_cache *cachep, int size)
+{
+}
+#endif

+#if DEBUG
 /*
  * memory layout of objects:
  * 0  : objp
@@ -433,11 +453,6 @@ static int obj_offset(struct kmem_cache *cachep)
  return cachep->obj_offset;
 }

-static int obj_size(struct kmem_cache *cachep)
-{
- return cachep->obj_size;
-}
-
 static unsigned long long *dbg_redzone1(struct kmem_cache *cachep, void *objp)
 {
  BUG_ON(!(cachep->flags & SLAB_RED_ZONE));
@@ -465,7 +480,6 @@ static void **dbg_userword(struct kmem_cache *cachep, void *objp)
 #else

 #define obj_offset(x)   0
-#define obj_size(cachep)  (cachep->buffer_size)
 #define dbg_redzone1(cachep, objp) ({BUG(); (unsigned long long *)NULL;})
 #define dbg_redzone2(cachep, objp) ({BUG(); (unsigned long long *)NULL;})
 #define dbg_userword(cachep, objp) ({BUG(); (void **)NULL;})
@@ -1555,9 +1569,9 @@ void __init kmem_cache_init(void)
   */
  cache_cache.buffer_size = offsetof(struct kmem_cache, array[nr_cpu_ids]) +
       nr_node_ids * sizeof(struct kmem_list3 *);
-#if DEBUG
```

```
- cache_cache.obj_size = cache_cache.buffer_size;
-#endif
+
+ set_obj_size(&cache_cache, cache_cache.buffer_size);
+
  cache_cache.buffer_size = ALIGN(cache_cache.buffer_size,
      cache_line_size());
  cache_cache.reciprocal_buffer_size =
@@ -2418,8 +2432,9 @@ kmem_cache_create (const char *name, size_t size, size_t align,
   goto oops;

  cachep->nodelists = (struct kmem_list3 **)&cachep->array[nr_cpu_ids];
+
+ set_obj_size(cachep, size);
 #if DEBUG
- cachep->obj_size = size;

  /*
   * Both debugging options require word-alignment which is calculated
--
1.7.7.6
```