Subject: [PATCH 4/4] make CFLGS_OFF_SLAB visible for all slabs
Posted by Glauber Costa on Thu, 14 Jun 2012 12:17:24 GMT
View Forum Message <> Reply to Message

Since we're now moving towards a unified slab allocator interface,
make CFLGS_OFF_SLAB visible to all allocators, even though SLAB keeps
being its only users. Also, make the name consistent with the other
flags, that start with SLAB_xx.

This will allow us to mask out this flag from common code, which will
of course have no effect in allocators not using it. It will also
avoid other allocators using this bit in the future by mistake.

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>
---
 include/linux/slab.h | 2 ++
 mm/slab.c | 17 ++++++++---------
 2 files changed, 10 insertions(+), 9 deletions(-)

diff --git a/include/linux/slab.h b/include/linux/slab.h
index 3c2181a..62deb32 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -79,6 +79,8 @@
 /* The following flags affect the page allocator grouping pages by mobility */
 #define SLAB_RECLAIM_ACCOUNT 0x00020000UL /* Objects are reclaimable */
 #define SLAB_TEMPORARY SLAB_RECLAIM_ACCOUNT /* Objects are short-lived */
+
+#define SLAB_OFF_SLAB 0x80000000UL
 /*
  * ZERO_SIZE_PTR will be returned for zero sized kmalloc requests.
  *
diff --git a/mm/slab.c b/mm/slab.c
index 2d5fe28..c0cf297 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -358,8 +358,7 @@ static void kmem_list3_init(struct kmem_list3 *parent)
  MAKE_LIST((cachep), (&(ptr)->slabs_free), slabs_free, nodeid); \
  } while (0)

-#define CFLGS_OFF_SLAB (0x80000000UL)
-#define OFF_SLAB(x) ((x)->flags & CFLGS_OFF_SLAB)
+#define OFF_SLAB(x) ((x)->flags & SLAB_OFF_SLAB)

 #define BATCHREFILL_LIMIT 16

```
 /*
@@ -744,7 +743,7 @@ static void cache_estimate(unsigned long gfporder, size_t buffer_size,
   * the slabs are all pages aligned, the objects will be at the
   * correct alignment when allocated.
   */
- if (flags & CFLGS_OFF_SLAB) {
+ if (flags & SLAB_OFF_SLAB) {
   mgmt_size = 0;
   nr_objs = slab_size / buffer_size;

@@ -2076,7 +2075,7 @@ static size_t calculate_slab_order(struct kmem_cache *cachep,
   if (!num)
    continue;

-  if (flags & CFLGS_OFF_SLAB) {
+  if (flags & SLAB_OFF_SLAB) {
    /*
     * Max number of objs-per-slab for caches which
     * use off-slab slabs. Needed to avoid a possible
@@ -2294,7 +2293,7 @@ int __kmem_cache_create(struct kmem_cache *cachep)
    * Size is large, assume best to place the slab management obj
    * off-slab (should allow better packing of objs).
    */
-  flags |= CFLGS_OFF_SLAB;
+  flags |= SLAB_OFF_SLAB;

  size = ALIGN(size, align);

@@ -2310,12 +2309,12 @@ int __kmem_cache_create(struct kmem_cache *cachep)
   * If the slab has been placed off-slab, and we have enough space then
   * move it on-slab. This is at the expense of any extra colouring.
   */
- if (flags & CFLGS_OFF_SLAB && left_over >= slab_size) {
-  flags &= ~CFLGS_OFF_SLAB;
+ if (flags & SLAB_OFF_SLAB && left_over >= slab_size) {
+  flags &= ~SLAB_OFF_SLAB;
   left_over -= slab_size;
  }

- if (flags & CFLGS_OFF_SLAB) {
+ if (flags & SLAB_OFF_SLAB) {
   /* really off slab. No need for manual alignment */
   slab_size =
       cachep->num * sizeof(kmem_bufctl_t) + sizeof(struct slab);
@@ -2343,7 +2342,7 @@ int __kmem_cache_create(struct kmem_cache *cachep)
  cachep->size = size;
  cachep->reciprocal_buffer_size = reciprocal_value(size);
```

```
- if (flags & CFLGS_OFF_SLAB) {
+ if (flags & SLAB_OFF_SLAB) {
   cachep->slabp_cache = kmem_find_general_cachep(slab_size, 0u);
   /*
    * This is a possibility for one of the malloc_sizes caches.
--
1.7.10.2
```