
Subject: [PATCH 05/10] slab: allow enable_cpu_cache to use preset values for its tunables

Posted by [Glauber Costa](#) on Wed, 25 Jul 2012 14:38:16 GMT

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

SLAB allows us to tune a particular cache behavior with tunables. When creating a new memcg cache copy, we'd like to preserve any tunables the parent cache already had.

This could be done by an explicit call to `do_tune_cpucache()` after the cache is created. But this is not very convenient now that the caches are created from common code, since this function is SLAB-specific.

Another method of doing that is taking advantage of the fact that `do_tune_cpucache()` is always called from `enable_cpucache()`, which is called at cache initialization. We can just preset the values, and then things work as expected.

Signed-off-by: Glauber Costa <glommer@parallels.com>

CC: Christoph Lameter <cl@linux.com>

CC: Pekka Enberg <penberg@cs.helsinki.fi>

CC: Michal Hocko <mhocko@suse.cz>

CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

CC: Johannes Weiner <hannes@cmpxchg.org>

CC: Suleiman Souhlal <suleiman@google.com>

```
include/linux/slab.h | 3 ++-
mm/memcontrol.c      | 2 +-
mm/slab.c            | 19 ++++++++-----
mm/slab_common.c     | 7 +++++-
4 files changed, 23 insertions(+), 8 deletions(-)
```

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

index 9d3fd56..249a0d3 100644

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

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

```
@@ -128,7 +128,7 @@ struct kmem_cache *kmem_cache_create(const char *, size_t, size_t,
    void (*)(void *));
```

```
struct kmem_cache *
```

```
kmem_cache_create_memcg(struct mem_cgroup *, const char *, size_t, size_t,
```

```
- unsigned long, void (*)(void *));
```

```
+ unsigned long, void (*)(void *), struct kmem_cache *);
```

```
void kmem_cache_destroy(struct kmem_cache *);
```

```
int kmem_cache_shrink(struct kmem_cache *);
```

```
void kmem_cache_free(struct kmem_cache *, void *);
```

```
@@ -184,6 +184,7 @@ unsigned int kmem_cache_size(struct kmem_cache *);
```

```
#ifdef CONFIG_MEMCG_KMEM
```

```
struct mem_cgroup_cache_params {
```

```

    struct mem_cgroup *memcg;
+ struct kmem_cache *parent;
    int id;
};
#endif
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 854f6cc..b933474 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -585,7 +585,7 @@ static struct kmem_cache *kmem_cache_dup(struct mem_cgroup
*memcg,
    return NULL;

    new = kmem_cache_create_memcg(memcg, name, s->object_size, s->align,
-    (s->flags & ~SLAB_PANIC), s->ctor);
+    (s->flags & ~SLAB_PANIC), s->ctor, s);

    kfree(name);
    return new;
diff --git a/mm/slab.c b/mm/slab.c
index 7e7ec59..76bc98f 100644
--- a/mm/slab.c
+++ b/mm/slab.c
@@ -3916,8 +3916,19 @@ static int do_tune_cpucache(struct kmem_cache *cachep, int limit,
static int enable_cpucache(struct kmem_cache *cachep, gfp_t gfp)
{
    int err;
- int limit, shared;
-
+ int limit = 0;
+ int shared = 0;
+ int batchcount = 0;
+
+ #ifdef CONFIG_MEMCG_KMEM
+     if (cachep->memcg_params.parent) {
+         limit = cachep->memcg_params.parent->limit;
+         shared = cachep->memcg_params.parent->shared;
+         batchcount = cachep->memcg_params.parent->batchcount;
+     }
+ #endif
+ if (limit && shared && batchcount)
+     goto skip_setup;
    /*
     * The head array serves three purposes:
     * - create a LIFO ordering, i.e. return objects that are cache-warm
@@ -3959,7 +3959,9 @@ static int enable_cpucache(struct kmem_cache *cachep, gfp_t gfp)
    if (limit > 32)
        limit = 32;

```

```

#endif
- err = do_tune_cpucache(cachep, limit, (limit + 1) / 2, shared, gfp);
+ batchcount = (limit + 1) / 2;
+skip_setup:
+ err = do_tune_cpucache(cachep, limit, batchcount, shared, gfp);
  if (err)
    printk(KERN_ERR "enable_cpucache failed for %s, error %d.\n",
           cachep->name, -err);
diff --git a/mm/slab_common.c b/mm/slab_common.c
index 1080ef2..562146b 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -80,7 +80,8 @@ unsigned long calculate_alignment(unsigned long flags,

struct kmem_cache *
kmem_cache_create_memcg(struct mem_cgroup *memcg, const char *name, size_t size,
- size_t align, unsigned long flags, void (*ctor)(void *))
+ size_t align, unsigned long flags, void (*ctor)(void *),
+ struct kmem_cache *parent_cache)
{
  struct kmem_cache *s = NULL;
  char *n;
@@ -149,9 +150,9 @@ kmem_cache_create_memcg(struct mem_cgroup *memcg, const char
*name, size_t size,
  s->ctor = ctor;
  s->flags = flags;
  s->align = calculate_alignment(flags, align, size);
-
#ifdef CONFIG_MEMCG_KMEM
  s->memcg_params.memcg = memcg;
+ s->memcg_params.parent = parent_cache;
#endif

  r = __kmem_cache_create(s);
@@ -184,7 +185,7 @@ out:
struct kmem_cache *kmem_cache_create(const char *name, size_t size, size_t align,
  unsigned long flags, void (*ctor)(void *))
{
- return kmem_cache_create_memcg(NULL, name, size, align, flags, ctor);
+ return kmem_cache_create_memcg(NULL, name, size, align, flags, ctor, NULL);
}
EXPORT_SYMBOL(kmem_cache_create);

--
1.7.10.4

```
