Subject: [PATCH 02/10] consider a memcg parameter in kmem_create_cache
Posted by Glauber Costa on Wed, 25 Jul 2012 14:38:13 GMT
View Forum Message <> Reply to Message

Allow a memcg parameter to be passed during cache creation.
When the slub allocator is being used, it will only merge
caches that belong to the same memcg.

Default function is created as a wrapper, passing NULL
to the memcg version. We only merge caches that belong
to the same memcg.

>From the memcontrol.c side, 3 helper functions are created:

1) memcg_css_id: because slub needs a unique cache name
   for sysfs. Since this is visible, but not the canonical
   location for slab data, the cache name is not used, the
   css_id should suffice.

2) mem_cgroup_register_cache: is responsible for assigning
   a unique index to each cache, and other general purpose
   setup. The index is only assigned for the root caches. All
   others are assigned index == -1.

3) mem_cgroup_release_cache: can be called from the root cache
   destruction, and will release the index for
   other caches.

We can't assign indexes until the basic slab is up and running
this is because the ida subsystem will itself call slab functions
such as kmalloc a couple of times. Because of that, we have
a late_initcall that scan all caches and register them after the
kernel is booted up. Only caches registered after that receive
their index right away.

This index mechanism was developed by Suleiman Souhlal.
Changed to a idr/ida based approach based on suggestion
from Kamezawa.

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/memcontrol.h | 14 ++++++++++++++

```
 include/linux/slab.h       |   10 ++++++++++
 mm/memcontrol.c            |   24 ++++++++++++++++++++++
 mm/slab.h                  |   24 ++++++++++++++++++++----
 mm/slab_common.c           |   36 ++++++++++++++++++++++++++++++++++++-----
 mm/slub.c                  |   16 ++++++++++++----
 6 files changed, 111 insertions(+), 13 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index 323d9e5..d9229a3 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -28,6 +28,7 @@ struct mem_cgroup;
 struct page_cgroup;
 struct page;
 struct mm_struct;
+struct kmem_cache;

 /* Stats that can be updated by kernel. */
 enum mem_cgroup_page_stat_item {
@@ -418,7 +419,20 @@ extern struct static_key memcg_kmem_enabled_key;
 bool __memcg_kmem_new_page(gfp_t gfp, void *handle, int order);
 void __memcg_kmem_commit_page(struct page *page, void *handle, int order);
 void __memcg_kmem_free_page(struct page *page, int order);
+int memcg_css_id(struct mem_cgroup *memcg);
+void memcg_register_cache(struct mem_cgroup *memcg,
+          struct kmem_cache *s);
+void memcg_release_cache(struct kmem_cache *cachep);
 #else
+static inline void memcg_register_cache(struct mem_cgroup *memcg,
+          struct kmem_cache *s)
+{
+}
+
+static inline void memcg_release_cache(struct kmem_cache *cachep)
+{
+}
+
 static inline void sock_update_memcg(struct sock *sk)
 {
 }
diff --git a/include/linux/slab.h b/include/linux/slab.h
index 3152bcd..9d3fd56 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -116,6 +116,7 @@ struct kmem_cache {
 };
 #endif
```

```
+struct mem_cgroup;
 /*
  * struct kmem_cache related prototypes
  */
@@ -125,6 +126,9 @@ int slab_is_available(void);
 struct kmem_cache *kmem_cache_create(const char *, size_t, size_t,
    unsigned long,
    void (*)(void *));
+struct kmem_cache *
+kmem_cache_create_memcg(struct mem_cgroup *, const char *, size_t, size_t,
+   unsigned long, void (*)(void *));
 void kmem_cache_destroy(struct kmem_cache *);
 int kmem_cache_shrink(struct kmem_cache *);
 void kmem_cache_free(struct kmem_cache *, void *);
@@ -337,6 +341,12 @@ extern void *__kmalloc_track_caller(size_t, gfp_t, unsigned long);
  __kmalloc(size, flags)
 #endif /* DEBUG_SLAB */

+#ifdef CONFIG_MEMCG_KMEM
+#define MAX_KMEM_CACHE_TYPES 400
+#else
+#define MAX_KMEM_CACHE_TYPES 0
+#endif /* CONFIG_MEMCG_KMEM */
+
 #ifdef CONFIG_NUMA
 /*
  * kmalloc_node_track_caller is a special version of kmalloc_node that
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 1321a02..88bb826 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -371,6 +371,11 @@ static void memcg_kmem_clear_account_parent(struct mem_cgroup
*memcg)
 {
  clear_bit(KMEM_ACCOUNTED_PARENT, &memcg->kmem_accounted);
 }
+
+int memcg_css_id(struct mem_cgroup *memcg)
+{
+ return css_id(&memcg->css);
+}
 #endif /* CONFIG_MEMCG_KMEM */

 /* Stuffs for move charges at task migration. */
@@ -527,6 +532,24 @@ static inline bool memcg_kmem_enabled(struct mem_cgroup *memcg)
   memcg->kmem_accounted;
 }
```

```
+struct ida cache_types;
+
+void memcg_register_cache(struct mem_cgroup *memcg, struct kmem_cache *cachep)
+{
+ int id = -1;
+
+ if (!memcg)
+  id = ida_simple_get(&cache_types, 0, MAX_KMEM_CACHE_TYPES,
+        GFP_KERNEL);
+ cachep->memcg_params.id = id;
+}
+
+void memcg_release_cache(struct kmem_cache *cachep)
+{
+ if (cachep->memcg_params.id != -1)
+  ida_simple_remove(&cache_types, cachep->memcg_params.id);
+}
+
 /*
  * We need to verify if the allocation against current->mm->owner's memcg is
  * possible for the given order. But the page is not allocated yet, so we'll
@@ -5182,6 +5205,7 @@ mem_cgroup_create(struct cgroup *cont)
 #ifdef CONFIG_MEMCG_KMEM
   WARN_ON(cgroup_add_cftypes(&mem_cgroup_subsys,
         kmem_cgroup_files));
+  ida_init(&cache_types);
 #endif

   if (mem_cgroup_soft_limit_tree_init())
diff --git a/mm/slab.h b/mm/slab.h
index 124be70..3c637d2 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -39,12 +39,15 @@ unsigned long calculate_alignment(unsigned long flags,
 int __kmem_cache_create(struct kmem_cache *s);

 int __kmem_cache_initcall(void);
+struct mem_cgroup;
 #ifdef CONFIG_SLUB
-struct kmem_cache *__kmem_cache_alias(const char *name, size_t size,
- size_t align, unsigned long flags, void (*ctor)(void *));
+struct kmem_cache *
+__kmem_cache_alias(struct mem_cgroup *memcg, const char *name, size_t size,
+     size_t align, unsigned long flags, void (*ctor)(void *));
 #else
-static inline struct kmem_cache *__kmem_cache_alias(const char *name, size_t size,
- size_t align, unsigned long flags, void (*ctor)(void *))
+static inline struct kmem_cache *
```

```
+__kmem_cache_alias(struct mem_cgroup *memcg, const char *name, size_t size,
+     size_t align, unsigned long flags, void (*ctor)(void *))
 { return NULL; }
 #endif

@@ -56,9 +59,22 @@ static inline bool slab_is_parent(struct kmem_cache *s, struct
kmem_cache *p)
 {
  return p == s->memcg_params.parent;
 }
+
+static inline bool cache_match_memcg(struct kmem_cache *cachep,
+         struct mem_cgroup *memcg)
+{
+        return cachep->memcg_params.memcg == memcg;
+}
+
 #else
 static inline bool slab_is_parent(struct kmem_cache *s, struct kmem_cache *p)
 {
  return false;
 }
+
+static inline bool cache_match_memcg(struct kmem_cache *cachep,
+         struct mem_cgroup *memcg)
+{
+        return true;
+}
 #endif
diff --git a/mm/slab_common.c b/mm/slab_common.c
index 66df8d5..1080ef2 100644
--- a/mm/slab_common.c
+++ b/mm/slab_common.c
@@ -16,6 +16,7 @@
 #include <asm/cacheflush.h>
 #include <asm/tlbflush.h>
 #include <asm/page.h>
+#include <linux/memcontrol.h>

 #include "slab.h"

@@ -77,8 +78,9 @@ unsigned long calculate_alignment(unsigned long flags,
  * as davem.
  */

-struct kmem_cache *kmem_cache_create(const char *name, size_t size, size_t align,
-  unsigned long flags, void (*ctor)(void *))
+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 *))
 {
  struct kmem_cache *s = NULL;
  char *n;
@@ -114,7 +116,7 @@ struct kmem_cache *kmem_cache_create(const char *name, size_t size,
size_t align
    continue;
   }

-  if (!strcmp(s->name, name)) {
+  if (cache_match_memcg(s, memcg) && !strcmp(s->name, name)) {
    printk(KERN_ERR "kmem_cache_create(%s): Cache name"
     " already exists.\n",
     name);
@@ -127,7 +129,7 @@ struct kmem_cache *kmem_cache_create(const char *name, size_t size,
size_t align
  WARN_ON(strchr(name, ' ')); /* It confuses parsers */
 #endif

- s = __kmem_cache_alias(name, size, align, flags, ctor);
+ s = __kmem_cache_alias(memcg, name, size, align, flags, ctor);
  if (s)
   goto oops;

@@ -148,11 +150,17 @@ struct kmem_cache *kmem_cache_create(const char *name, size_t
size, size_t align
  s->flags = flags;
  s->align = calculate_alignment(flags, align, size);

+#ifdef CONFIG_MEMCG_KMEM
+ s->memcg_params.memcg = memcg;
+#endif
+
  r = __kmem_cache_create(s);

  if (!r) {
   s->refcount = 1;
   list_add(&s->list, &slab_caches);
+  if (slab_state >= FULL)
+   memcg_register_cache(memcg, s);
  }
  else {
   kmem_cache_free(kmem_cache, s);
@@ -172,6 +180,12 @@ out:

  return s;
 }
```

```
+
+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);
+}
 EXPORT_SYMBOL(kmem_cache_create);

 void kmem_cache_destroy(struct kmem_cache *s)
@@ -184,6 +198,7 @@ void kmem_cache_destroy(struct kmem_cache *s)
   if (s->flags & SLAB_DESTROY_BY_RCU)
    rcu_barrier();

+  memcg_release_cache(s);
   kfree(s->name);
   kmem_cache_free(kmem_cache, s);
  } else {
@@ -204,6 +219,17 @@ int slab_is_available(void)

 static int __init kmem_cache_initcall(void)
 {
- return __kmem_cache_initcall();
+ int r = __kmem_cache_initcall();
+#ifdef CONFIG_MEMCG_KMEM
+ struct kmem_cache *s;
+
+ if (r)
+  return r;
+ mutex_lock(&slab_mutex);
+ list_for_each_entry(s, &slab_caches, list)
+  memcg_register_cache(NULL, s);
+ mutex_unlock(&slab_mutex);
+#endif
+ return r;
 }
 __initcall(kmem_cache_initcall);
diff --git a/mm/slub.c b/mm/slub.c
index 824ea2e..417a806 100644
--- a/mm/slub.c
+++ b/mm/slub.c
@@ -31,6 +31,7 @@
 #include <linux/fault-inject.h>
 #include <linux/stacktrace.h>
 #include <linux/prefetch.h>
+#include <linux/memcontrol.h>

 #include <trace/events/kmem.h>
```

```
@@ -3819,7 +3820,7 @@ static int slab_unmergeable(struct kmem_cache *s)
  return 0;
 }

-static struct kmem_cache *find_mergeable(size_t size,
+static struct kmem_cache *find_mergeable(struct mem_cgroup *memcg, size_t size,
    size_t align, unsigned long flags, const char *name,
    void (*ctor)(void *))
 {
@@ -3855,17 +3856,20 @@ static struct kmem_cache *find_mergeable(size_t size,
    if (s->size - size >= sizeof(void *))
     continue;

+  if (!cache_match_memcg(s, memcg))
+   continue;
    return s;
   }
  return NULL;
 }

-struct kmem_cache *__kmem_cache_alias(const char *name, size_t size,
-  size_t align, unsigned long flags, void (*ctor)(void *))
+struct kmem_cache *
+__kmem_cache_alias(struct mem_cgroup *memcg, const char *name, size_t size,
+     size_t align, unsigned long flags, void (*ctor)(void *))
 {
  struct kmem_cache *s;

- s = find_mergeable(size, align, flags, name, ctor);
+ s = find_mergeable(memcg, size, align, flags, name, ctor);
  if (s) {
   s->refcount++;
   /*
@@ -5195,6 +5199,10 @@ static char *create_unique_id(struct kmem_cache *s)
  if (p != name + 1)
   *p++ = '-';
  p += sprintf(p, "%07d", s->size);
+#ifdef CONFIG_MEMCG_KMEM
+ if (s->memcg_params.memcg)
+  p += sprintf(p, "-%08d", memcg_css_id(s->memcg_params.memcg));
+#endif
  BUG_ON(p > name + ID_STR_LENGTH - 1);
  return name;
 }
--
1.7.10.4
```