Subject: [PATCH 2/4] BIO tracking take2
Posted by Hirokazu Takahashi on Tue, 20 May 2008 12:02:37 GMT
View Forum Message <> Reply to Message

Hi,

This patch is for cleaning up the code of the cgroup memory subsystem
to remove a lot of "#ifdef"s.

Signed-off-by: Hirokazu Takahashi <taka@valinux.co.jp>

```
diff -dupr linux-2.6.26-rc2.cg1/mm/memcontrol.c linux-2.6.26-rc2/mm/memcontrol.c
--- linux-2.6.26-rc2.cg1/mm/memcontrol.c 2008-05-19 12:33:32.000000000 +0900
+++ linux-2.6.26-rc2/mm/memcontrol.c 2008-05-19 12:34:16.000000000 +0900
@@ -228,6 +228,47 @@ struct mem_cgroup *mem_cgroup_from_task(
     struct mem_cgroup, css);
 }

+static inline void get_mem_cgroup(struct mem_cgroup *mem)
+{
+ css_get(&mem->css);
+}
+
+static inline void put_mem_cgroup(struct mem_cgroup *mem)
+{
+ css_put(&mem->css);
+}
+
+static inline void set_mem_cgroup(struct page_cgroup *pc,
+                                  struct mem_cgroup *mem)
+{
+ pc->mem_cgroup = mem;
+}
+
+static inline void clear_mem_cgroup(struct page_cgroup *pc)
+{
+ struct mem_cgroup *mem = pc->mem_cgroup;
+ res_counter_uncharge(&mem->res, PAGE_SIZE);
+ pc->mem_cgroup = NULL;
+ put_mem_cgroup(mem);
+}
+
+static inline struct mem_cgroup *get_mem_page_cgroup(struct page_cgroup *pc)
+{
+ struct mem_cgroup *mem = pc->mem_cgroup;
+ css_get(&mem->css);
```

```
+ return mem;
+}
+
+/* This sould be called in an RCU-protected section. */
+static inline struct mem_cgroup *mm_get_mem_cgroup(struct mm_struct *mm)
+{
+ struct mem_cgroup *mem;
+
+ mem = mem_cgroup_from_task(rcu_dereference(mm->owner));
+ get_mem_cgroup(mem);
+ return mem;
+}
+
 static void __mem_cgroup_remove_list(struct mem_cgroup_per_zone *mz,
    struct page_cgroup *pc)
 {
@@ -278,6 +319,26 @@ static void __mem_cgroup_move_lists(stru
  }
 }

+static inline void mem_cgroup_add_page(struct page_cgroup *pc)
+{
+ struct mem_cgroup_per_zone *mz = page_cgroup_zoneinfo(pc);
+ unsigned long flags;
+
+ spin_lock_irqsave(&mz->lru_lock, flags);
+ __mem_cgroup_add_list(mz, pc);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);
+}
+
+static inline void mem_cgroup_remove_page(struct page_cgroup *pc)
+{
+ struct mem_cgroup_per_zone *mz = page_cgroup_zoneinfo(pc);
+ unsigned long flags;
+
+ spin_lock_irqsave(&mz->lru_lock, flags);
+ __mem_cgroup_remove_list(mz, pc);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);
+}
+
 int task_in_mem_cgroup(struct task_struct *task, const struct mem_cgroup *mem)
 {
  int ret;
@@ -317,6 +378,36 @@ void mem_cgroup_move_lists(struct page *
  unlock_page_cgroup(page);
 }

+static inline int mem_cgroup_try_to_allocate(struct mem_cgroup *mem,
```

```
+      gfp_t gfp_mask)
+{
+ unsigned long nr_retries = MEM_CGROUP_RECLAIM_RETRIES;
+
+ while (res_counter_charge(&mem->res, PAGE_SIZE)) {
+  if (!(gfp_mask & __GFP_WAIT))
+   return -1;
+
+  if (try_to_free_mem_cgroup_pages(mem, gfp_mask))
+   continue;
+
+  /*
+   * try_to_free_mem_cgroup_pages() might not give us a full
+   * picture of reclaim. Some pages are reclaimed and might be
+   * moved to swap cache or just unmapped from the cgroup.
+   * Check the limit again to see if the reclaim reduced the
+   * current usage of the cgroup before giving up
+   */
+  if (res_counter_check_under_limit(&mem->res))
+   continue;
+
+  if (!nr_retries--) {
+   mem_cgroup_out_of_memory(mem, gfp_mask);
+   return -1;
+  }
+ }
+ return 0;
+}
+
 /*
  * Calculate mapped_ratio under memory controller. This will be used in
  * vmscan.c for deteremining we have to reclaim mapped pages.
@@ -517,7 +608,7 @@ static int mem_cgroup_force_empty(struct
  if (mem_cgroup_disabled())
   return 0;

- css_get(&mem->css);
+ get_mem_cgroup(mem);
  /*
   * page reclaim code (kswapd etc..) will move pages between
   * active_list <-> inactive_list while we don't take a lock.
@@ -538,7 +629,7 @@ static int mem_cgroup_force_empty(struct
  }
  ret = 0;
 out:
- css_put(&mem->css);
+ put_mem_cgroup(mem);
  return ret;
```

```
 }

@@ -825,10 +916,37 @@ struct cgroup_subsys mem_cgroup_subsys =

 #else /* CONFIG_CGROUP_MEM_RES_CTLR */

+struct mem_cgroup;
+
 static inline int mem_cgroup_disabled(void)
 {
  return 1;
 }
+
+static inline void mem_cgroup_add_page(struct page_cgroup *pc) {}
+static inline void mem_cgroup_remove_page(struct page_cgroup *pc) {}
+static inline void get_mem_cgroup(struct mem_cgroup *mem) {}
+static inline void put_mem_cgroup(struct mem_cgroup *mem) {}
+static inline void set_mem_cgroup(struct page_cgroup *pc,
+      struct mem_cgroup *mem) {}
+static inline void clear_mem_cgroup(struct page_cgroup *pc) {}
+
+static inline struct mem_cgroup *get_mem_page_cgroup(struct page_cgroup *pc)
+{
+ return NULL;
+}
+
+static inline struct mem_cgroup *mm_get_mem_cgroup(struct mm_struct *mm)
+{
+ return NULL;
+}
+
+static inline int mem_cgroup_try_to_allocate(struct mem_cgroup *mem,
+      gfp_t gfp_mask)
+{
+ return 0;
+}
+
 #endif /* CONFIG_CGROUP_MEM_RES_CTLR */

 static inline int page_cgroup_locked(struct page *page)
@@ -863,12 +981,7 @@ static int mem_cgroup_charge_common(stru
     struct mem_cgroup *memcg)
 {
  struct page_cgroup *pc;
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
  struct mem_cgroup *mem;
- unsigned long flags;
- unsigned long nr_retries = MEM_CGROUP_RECLAIM_RETRIES;
```

```
- struct mem_cgroup_per_zone *mz;
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */

  if (mem_cgroup_disabled())
   return 0;
@@ -912,50 +1025,18 @@ retry:
    mm = &init_mm;

   rcu_read_lock();
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
-  mem = mem_cgroup_from_task(rcu_dereference(mm->owner));
-  /*
-   * For every charge from the cgroup, increment reference count
-   */
-  css_get(&mem->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  mem = mm_get_mem_cgroup(mm);
   rcu_read_unlock();
  } else {
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
   mem = memcg;
-  css_get(&memcg->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  get_mem_cgroup(mem);
  }

-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- while (res_counter_charge(&mem->res, PAGE_SIZE)) {
-  if (!(gfp_mask & __GFP_WAIT))
-   goto out;
-
-  if (try_to_free_mem_cgroup_pages(mem, gfp_mask))
-   continue;
-
-  /*
-   * try_to_free_mem_cgroup_pages() might not give us a full
-   * picture of reclaim. Some pages are reclaimed and might be
-   * moved to swap cache or just unmapped from the cgroup.
-   * Check the limit again to see if the reclaim reduced the
-   * current usage of the cgroup before giving up
-   */
-  if (res_counter_check_under_limit(&mem->res))
-   continue;
-
-  if (!nr_retries--) {
-   mem_cgroup_out_of_memory(mem, gfp_mask);
-   goto out;
-  }
```

```
- }
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ if (mem_cgroup_try_to_allocate(mem, gfp_mask) < 0)
+  goto out;

  pc->ref_cnt = 1;
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- pc->mem_cgroup = mem;
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ set_mem_cgroup(pc, mem);
  pc->page = page;
  /*
   * If a page is accounted as a page cache, insert to inactive list.
@@ -974,29 +1055,19 @@ retry:
    * We take lock_page_cgroup(page) again and read
    * page->cgroup, increment refcnt.... just retry is OK.
    */
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
-  res_counter_uncharge(&mem->res, PAGE_SIZE);
-  css_put(&mem->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  clear_mem_cgroup(pc);
   kmem_cache_free(page_cgroup_cache, pc);
   goto retry;
  }
  page_assign_page_cgroup(page, pc);

-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- mz = page_cgroup_zoneinfo(pc);
- spin_lock_irqsave(&mz->lru_lock, flags);
- __mem_cgroup_add_list(mz, pc);
- spin_unlock_irqrestore(&mz->lru_lock, flags);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ mem_cgroup_add_page(pc);

  unlock_page_cgroup(page);
 done:
  return 0;
 out:
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- css_put(&mem->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ put_mem_cgroup(mem);
  kmem_cache_free(page_cgroup_cache, pc);
 err:
  return -ENOMEM;
@@ -1039,11 +1110,6 @@ int mem_cgroup_getref(struct page *page)
 void mem_cgroup_uncharge_page(struct page *page)
```

```
 {
  struct page_cgroup *pc;
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- struct mem_cgroup *mem;
- struct mem_cgroup_per_zone *mz;
- unsigned long flags;
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */

  if (mem_cgroup_disabled())
   return;
@@ -1060,21 +1126,12 @@ void mem_cgroup_uncharge_page(struct pag
  VM_BUG_ON(pc->ref_cnt <= 0);

  if (--(pc->ref_cnt) == 0) {
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
-  mz = page_cgroup_zoneinfo(pc);
-  spin_lock_irqsave(&mz->lru_lock, flags);
-  __mem_cgroup_remove_list(mz, pc);
-  spin_unlock_irqrestore(&mz->lru_lock, flags);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  mem_cgroup_remove_page(pc);

   page_assign_page_cgroup(page, NULL);
   unlock_page_cgroup(page);

-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
-  mem = pc->mem_cgroup;
-  res_counter_uncharge(&mem->res, PAGE_SIZE);
-  css_put(&mem->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  clear_mem_cgroup(pc);

   kmem_cache_free(page_cgroup_cache, pc);
   return;
@@ -1100,10 +1157,7 @@ int mem_cgroup_prepare_migration(struct
  lock_page_cgroup(page);
  pc = page_get_page_cgroup(page);
  if (pc) {
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
-  mem = pc->mem_cgroup;
-  css_get(&mem->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  mem = get_mem_page_cgroup(pc);
   if (pc->flags & PAGE_CGROUP_FLAG_CACHE)
    ctype = MEM_CGROUP_CHARGE_TYPE_CACHE;
  }
@@ -1111,9 +1165,7 @@ int mem_cgroup_prepare_migration(struct
  if (mem) {
```

```
   ret = mem_cgroup_charge_common(newpage, NULL, GFP_KERNEL,
    ctype, mem);
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
-  css_put(&mem->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  put_mem_cgroup(mem);
  }
  return ret;
 }
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers