Subject: [PATCH 3/4] Block I/O tracking
Posted by Hirokazu Takahashi on Tue, 18 Mar 2008 09:30:33 GMT
View Forum Message <> Reply to Message

Hi,

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

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

```
--- linux-2.6.25-rc5.pagecgroup3/mm/memcontrol.c 2008-03-18 13:01:57.000000000 +0900
+++ linux-2.6.25-rc5-mm1/mm/memcontrol.c 2008-03-18 14:00:23.000000000 +0900
@@ -216,6 +216,51 @@ static struct mem_cgroup *mem_cgroup_fro
     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);
+}
+
+/* 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 = rcu_dereference(mm->mem_cgroup);
+ get_mem_cgroup(mem);
```

```
+ return mem;
+}
+
+static inline void mm_init_mem_cgroup(struct mm_struct *mm,
+     struct task_struct *p)
+{
+ struct mem_cgroup *mem = mem_cgroup_from_task(p);
+ get_mem_cgroup(mem);
+ mm->mem_cgroup = mem;
+}
+
+static inline void mm_free_mem_cgroup(struct mm_struct *mm)
+{
+ put_mem_cgroup(mm->mem_cgroup);
+}
+
 static void __mem_cgroup_remove_list(struct page_cgroup *pc)
 {
  int from = pc->flags & PAGE_CGROUP_FLAG_ACTIVE;
@@ -266,6 +311,26 @@ 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(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(pc);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);
+}
+
 int task_in_mem_cgroup(struct task_struct *task, const struct mem_cgroup *mem)
 {
  int ret;
@@ -305,6 +370,37 @@ 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;
+  }
+  congestion_wait(WRITE, HZ/10);
+ }
+ return 0;
+}
+
 /*
  * Calculate mapped_ratio under memory controller. This will be used in
  * vmscan.c for deteremining we have to reclaim mapped pages.
@@ -497,7 +593,7 @@ 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.
@@ -518,7 +614,7 @@ static int mem_cgroup_force_empty(struct
  }
  ret = 0;
```

```
 out:
- css_put(&mem->css);
+ put_mem_cgroup(mem);
  return ret;
 }

@@ -750,9 +846,9 @@ static void mem_cgroup_move_task(struct
  if (p->tgid != p->pid)
   goto out;

- css_get(&mem->css);
+ get_mem_cgroup(mem);
  rcu_assign_pointer(mm->mem_cgroup, mem);
- css_put(&old_mem->css);
+ put_mem_cgroup(old_mem);

 out:
  mmput(mm);
@@ -770,29 +866,47 @@ struct cgroup_subsys mem_cgroup_subsys =
 };
 #else /* CONFIG_CGROUP_MEM_RES_CTLR */

+struct mem_cgroup;
+
 static inline int mem_cgroup_disabled(void)
 {
  return 1;
 }
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */

-void mm_init_cgroup(struct mm_struct *mm, struct task_struct *p)
+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 *mm_get_mem_cgroup(struct mm_struct *mm)
 {
- struct mem_cgroup *mem;
+ return NULL;
+}

-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- mem = mem_cgroup_from_task(p);
- css_get(&mem->css);
```

```
- mm->mem_cgroup = mem;
+static inline void mm_init_mem_cgroup(struct mm_struct *mm,
+     struct task_struct *p) {}
+static inline void mm_free_mem_cgroup(struct mm_struct *mm) {}
+
+static inline int mem_cgroup_try_to_allocate(struct mem_cgroup *mem,
+     gfp_t gfp_mask)
+{
+ return 0;
+}
+
 #endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+
+void mm_init_cgroup(struct mm_struct *mm, struct task_struct *p)
+{
+ mm_init_mem_cgroup(mm, p);
  mm_init_bio_cgroup(mm, p);
 }

 void mm_free_cgroup(struct mm_struct *mm)
 {
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- css_put(&mm->mem_cgroup->css);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ mm_free_mem_cgroup(mm);
  mm_free_bio_cgroup(mm);
 }

@@ -827,12 +941,7 @@ static int mem_cgroup_charge_common(stru
     gfp_t gfp_mask, enum charge_type ctype)
 {
  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 */
  struct bio_cgroup *biog;

  if (mem_cgroup_disabled() && bio_cgroup_disabled())
@@ -876,46 +985,18 @@ retry:
   mm = &init_mm;

  rcu_read_lock();
-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- mem = rcu_dereference(mm->mem_cgroup);
  /*
```

```
   * 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);
  biog = mm_get_bio_cgroup(mm);
  rcu_read_unlock();

-#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;
-  }
-  congestion_wait(WRITE, HZ/10);
- }
-#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);
  set_bio_cgroup(pc, biog);
  pc->page = page;
  pc->flags = PAGE_CGROUP_FLAG_ACTIVE;
@@ -930,31 +1011,21 @@ 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);
   clear_bio_cgroup(pc);
   kfree(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(pc);
- spin_unlock_irqrestore(&mz->lru_lock, flags);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ mem_cgroup_add_page(pc);
  bio_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);
  put_bio_cgroup(biog);
  kfree(pc);
 err:
@@ -983,9 +1054,6 @@ int mem_cgroup_cache_charge(struct page
 void mem_cgroup_uncharge_page(struct page *page)
 {
  struct page_cgroup *pc;
- struct mem_cgroup *mem;
- struct mem_cgroup_per_zone *mz;
- unsigned long flags;

  if (mem_cgroup_disabled() && bio_cgroup_disabled())
   return;
@@ -1002,22 +1070,13 @@ 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(pc);
-  spin_unlock_irqrestore(&mz->lru_lock, flags);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+  mem_cgroup_remove_page(pc);
   bio_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);
   clear_bio_cgroup(pc);

   kfree(pc);
@@ -1060,8 +1119,6 @@ void mem_cgroup_end_migration(struct pag
 void mem_cgroup_page_migration(struct page *page, struct page *newpage)
 {
  struct page_cgroup *pc;
- struct mem_cgroup_per_zone *mz;
- unsigned long flags;

  lock_page_cgroup(page);
  pc = page_get_page_cgroup(page);
@@ -1070,12 +1127,7 @@ void mem_cgroup_page_migration(struct pa
   return;
  }

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

  page_assign_page_cgroup(page, NULL);
@@ -1085,12 +1137,7 @@ void mem_cgroup_page_migration(struct pa
  lock_page_cgroup(newpage);
  page_assign_page_cgroup(newpage, pc);

-#ifdef CONFIG_CGROUP_MEM_RES_CTLR
- mz = page_cgroup_zoneinfo(pc);
- spin_lock_irqsave(&mz->lru_lock, flags);
```

```
- __mem_cgroup_add_list(pc);
- spin_unlock_irqrestore(&mz->lru_lock, flags);
-#endif /* CONFIG_CGROUP_MEM_RES_CTLR */
+ mem_cgroup_add_page(pc);
  bio_cgroup_add_page(pc);

  unlock_page_cgroup(newpage);
```

_____

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