
Subject: [RFC 1/4] memcg: drop pages at rmdir (v1)
Posted by [KAMEZAWA Hiroyuki](#) on Tue, 27 May 2008 05:04:30 GMT
[View Forum Message](#) <> [Reply to Message](#)

Now, when we remove memcg, we call force_empty().
This call drops all page_cgroup accounting in this mem_cgroup but doesn't drop pages. So, some page caches can be remained as "not accounted" memory while they are alive. (because it's accounted only when add_to_page_cache())
If they are not used by other memcg, global LRU will drop them.

This patch tries to drop pages at removing memcg. Other memcg will reload and re-account page caches. (but this will increase page-in after rmdir().)

Consideration: should we recharge all pages to the parent at last ?
But it's not precise logic.

Changelog v1->v2
- renamed res_counter_empty().

Signed-off-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

include/linux/res_counter.h | 11 ++++++++
mm/memcontrol.c | 19 ++++++++
2 files changed, 30 insertions(+)

Index: mm-2.6.26-rc2-mm1/mm/memcontrol.c

```
=====
--- mm-2.6.26-rc2-mm1.orig/mm/memcontrol.c
+++ mm-2.6.26-rc2-mm1/mm/memcontrol.c
@@ -791,6 +791,20 @@ int mem_cgroup_shrink_usage(struct mm_st
    return 0;
}

+
+static void mem_cgroup_drop_all_pages(struct mem_cgroup *mem)
+{
+ int progress;
+ while (!res_counter_empty(&mem->res)) {
+ progress = try_to_free_mem_cgroup_pages(mem,
+ GFP_HIGHUSER_MOVABLE);
+ if (!progress) /* we did as much as possible */
+ break;
+ cond_resched();
+ }
+ return;
+}
```

```

+
+ /*
+  * This routine traverse page_cgroup in given list and drop them all.
+  * *And* this routine doesn't reclaim page itself, just removes page_cgroup.
@@ -848,7 +862,12 @@ static int mem_cgroup_force_empty(struct
+ if (mem_cgroup_subsys.disabled)
+ return 0;

```

```

+ if (atomic_read(&mem->css.cgroup->count) > 0)

```

```

+ goto out;

```

```

+

```

```

+ css_get(&mem->css);

```

```

+ /* drop pages as much as possible */

```

```

+ mem_cgroup_drop_all_pages(mem);

```

```

+ /*

```

```

+  * page reclaim code (kswapd etc..) will move pages between

```

```

+  * active_list <-> inactive_list while we don't take a lock.

```

```

Index: mm-2.6.26-rc2-mm1/include/linux/res_counter.h

```

```

=====

```

```

--- mm-2.6.26-rc2-mm1.orig/include/linux/res_counter.h

```

```

+++ mm-2.6.26-rc2-mm1/include/linux/res_counter.h

```

```

@@ -153,4 +153,15 @@ static inline void res_counter_reset_fai

```

```

+ cnt->failcnt = 0;

```

```

+ spin_unlock_irqrestore(&cnt->lock, flags);

```

```

+ }

```

```

+/* returns 0 if usage is 0. */

```

```

+static inline int res_counter_empty(struct res_counter *cnt)

```

```

+{

```

```

+ unsigned long flags;

```

```

+ int ret;

```

```

+

```

```

+ spin_lock_irqsave(&cnt->lock, flags);

```

```

+ ret = (cnt->usage == 0) ? 0 : 1;

```

```

+ spin_unlock_irqrestore(&cnt->lock, flags);

```

```

+ return ret;

```

```

+}

```

```

+ #endif

```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
