Subject: [PATCH][for -mm] per-zone and reclaim enhancements for memory controller take 3 [10/10] per-zone-loc
Posted by KAMEZAWA Hiroyuki on Tue, 27 Nov 2007 03:10:49 GMT
View Forum Message <> Reply to Message

Now, lru is per-zone.

Then, lru_lock can be (should be) per-zone, too.
This patch implementes per-zone lru lock.

lru_lock is placed into mem_cgroup_per_zone struct.

lock can be accessed by
```
  mz = mem_cgroup_zoneinfo(mem_cgroup, node, zone);
  &mz->lru_lock

  or
  mz = page_cgroup_zoneinfo(page_cgroup);
  &mz->lru_lock
```

Signed-off-by: KAMEZAWA hiroyuki <kmaezawa.hiroyu@jp.fujitsu.com>

```
 mm/memcontrol.c |   71 ++++++++++++++++++++++++++++++++++++----------------------
 1 file changed, 44 insertions(+), 27 deletions(-)

Index: linux-2.6.24-rc3-mm1/mm/memcontrol.c
===================================================================
--- linux-2.6.24-rc3-mm1.orig/mm/memcontrol.c 2007-11-27 11:24:16.000000000 +0900
+++ linux-2.6.24-rc3-mm1/mm/memcontrol.c 2007-11-27 11:24:22.000000000 +0900
@@ -89,6 +89,10 @@
 };

 struct mem_cgroup_per_zone {
+ /*
+  * spin_lock to protect the per cgroup LRU
+  */
+ spinlock_t  lru_lock;
  struct list_head active_list;
  struct list_head inactive_list;
  unsigned long count[NR_MEM_CGROUP_ZSTAT];
@@ -126,10 +130,7 @@
   * per zone LRU lists.
   */
  struct mem_cgroup_lru_info info;
- /*
-  * spin_lock to protect the per cgroup LRU
-  */
```

```
- spinlock_t lru_lock;
+
  unsigned long control_type; /* control RSS or RSS+Pagecache */
  int prev_priority; /* for recording reclaim priority */
  /*
@@ -410,15 +411,16 @@
  */
 void mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
 {
- struct mem_cgroup *mem;
+ struct mem_cgroup_per_zone *mz;
+ unsigned long flags;
+
  if (!pc)
   return;

- mem = pc->mem_cgroup;
-
- spin_lock(&mem->lru_lock);
+ mz = page_cgroup_zoneinfo(pc);
+ spin_lock_irqsave(&mz->lru_lock, flags);
  __mem_cgroup_move_lists(pc, active);
- spin_unlock(&mem->lru_lock);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);
 }

 /*
@@ -528,7 +530,7 @@
   src = &mz->inactive_list;


- spin_lock(&mem_cont->lru_lock);
+ spin_lock(&mz->lru_lock);
  scan = 0;
  list_for_each_entry_safe_reverse(pc, tmp, src, lru) {
   if (scan >= nr_to_scan)
@@ -558,7 +560,7 @@
  }

  list_splice(&pc_list, src);
- spin_unlock(&mem_cont->lru_lock);
+ spin_unlock(&mz->lru_lock);

  *scanned = scan;
  return nr_taken;
@@ -577,6 +579,7 @@
  struct page_cgroup *pc;
  unsigned long flags;
```

```
  unsigned long nr_retries = MEM_CGROUP_RECLAIM_RETRIES;
+ struct mem_cgroup_per_zone *mz;

  /*
   * Should page_cgroup's go to their own slab?
@@ -688,10 +691,11 @@
   goto retry;
  }

- spin_lock_irqsave(&mem->lru_lock, flags);
+ mz = page_cgroup_zoneinfo(pc);
+ spin_lock_irqsave(&mz->lru_lock, flags);
  /* Update statistics vector */
  __mem_cgroup_add_list(pc);
- spin_unlock_irqrestore(&mem->lru_lock, flags);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);

 done:
  return 0;
@@ -733,6 +737,7 @@
 void mem_cgroup_uncharge(struct page_cgroup *pc)
 {
  struct mem_cgroup *mem;
+ struct mem_cgroup_per_zone *mz;
  struct page *page;
  unsigned long flags;

@@ -745,6 +750,7 @@

  if (atomic_dec_and_test(&pc->ref_cnt)) {
   page = pc->page;
+  mz = page_cgroup_zoneinfo(pc);
   /*
    * get page->cgroup and clear it under lock.
    * force_empty can drop page->cgroup without checking refcnt.
@@ -753,9 +759,9 @@
    mem = pc->mem_cgroup;
    css_put(&mem->css);
    res_counter_uncharge(&mem->res, PAGE_SIZE);
-   spin_lock_irqsave(&mem->lru_lock, flags);
+   spin_lock_irqsave(&mz->lru_lock, flags);
    __mem_cgroup_remove_list(pc);
-   spin_unlock_irqrestore(&mem->lru_lock, flags);
+   spin_unlock_irqrestore(&mz->lru_lock, flags);
    kfree(pc);
   }
  }
@@ -794,24 +800,29 @@
```

```
  struct page_cgroup *pc;
  struct mem_cgroup *mem;
  unsigned long flags;
+ struct mem_cgroup_per_zone *mz;
 retry:
  pc = page_get_page_cgroup(page);
  if (!pc)
   return;
  mem = pc->mem_cgroup;
+ mz = page_cgroup_zoneinfo(pc);
  if (clear_page_cgroup(page, pc) != pc)
   goto retry;
-
- spin_lock_irqsave(&mem->lru_lock, flags);
+ spin_lock_irqsave(&mz->lru_lock, flags);

  __mem_cgroup_remove_list(pc);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);
+
  pc->page = newpage;
  lock_page_cgroup(newpage);
  page_assign_page_cgroup(newpage, pc);
  unlock_page_cgroup(newpage);
- __mem_cgroup_add_list(pc);

- spin_unlock_irqrestore(&mem->lru_lock, flags);
+ mz = page_cgroup_zoneinfo(pc);
+ spin_lock_irqsave(&mz->lru_lock, flags);
+ __mem_cgroup_add_list(pc);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);
  return;
 }

@@ -822,18 +833,26 @@
  */
 #define FORCE_UNCHARGE_BATCH (128)
 static void
-mem_cgroup_force_empty_list(struct mem_cgroup *mem, struct list_head *list)
+mem_cgroup_force_empty_list(struct mem_cgroup *mem,
+       struct mem_cgroup_per_zone *mz,
+       int active)
 {
  struct page_cgroup *pc;
  struct page *page;
  int count;
  unsigned long flags;
+ struct list_head *list;
+
```

```
+ if (active)
+  list = &mz->active_list;
+ else
+  list = &mz->inactive_list;

  if (list_empty(list))
   return;
 retry:
  count = FORCE_UNCHARGE_BATCH;
- spin_lock_irqsave(&mem->lru_lock, flags);
+ spin_lock_irqsave(&mz->lru_lock, flags);

  while (--count && !list_empty(list)) {
   pc = list_entry(list->prev, struct page_cgroup, lru);
@@ -850,7 +869,7 @@
   } else  /* being uncharged ? ...do relax */
    break;
  }
- spin_unlock_irqrestore(&mem->lru_lock, flags);
+ spin_unlock_irqrestore(&mz->lru_lock, flags);
  if (!list_empty(list)) {
   cond_resched();
   goto retry;
@@ -881,11 +900,9 @@
     struct mem_cgroup_per_zone *mz;
     mz = mem_cgroup_zoneinfo(mem, node, zid);
     /* drop all page_cgroup in active_list */
-    mem_cgroup_force_empty_list(mem,
-       &mz->active_list);
+    mem_cgroup_force_empty_list(mem, mz, 1);
     /* drop all page_cgroup in inactive_list */
-    mem_cgroup_force_empty_list(mem,
-       &mz->inactive_list);
+    mem_cgroup_force_empty_list(mem, mz, 0);
    }
  }
  ret = 0;
@@ -1112,6 +1129,7 @@
   mz = &pn->zoneinfo[zone];
   INIT_LIST_HEAD(&mz->active_list);
   INIT_LIST_HEAD(&mz->inactive_list);
+  spin_lock_init(&mz->lru_lock);
  }
  return 0;
 }
@@ -1136,7 +1154,6 @@

  res_counter_init(&mem->res);
```

- spin_lock_init(&mem->lru_lock);
  mem->control_type = MEM_CGROUP_TYPE_ALL;
  memset(&mem->info, 0, sizeof(mem->info));

_____

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