Subject: [RFC][PATCH] memory controller per zone patches take 2 [9/10]
per-zone-lru for memory cgroup
Posted by KAMEZAWA Hiroyuki on Fri, 16 Nov 2007 10:25:51 GMT
View Forum Message <> Reply to Message

This patch implements per-zone lru for memory cgroup.
This patch makes use of mem_cgroup_per_zone struct for per zone lru.

LRU can be accessed by

  mz = mem_cgroup_zoneinfo(mem_cgroup, node, zone);
  &mz->active_list
  &mz->inactive_list

  or
  mz = page_cgroup_zoneinfo(page_cgroup, node, zone);
  &mz->active_list
  &mz->inactive_list

Changelog v1->v2
 - merged to mem_cgroup_per_zone struct.
 - handle page migraiton.

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

 mm/memcontrol.c | 63 ++++++++++++++++++++++++++++++++++++++----------------
 1 file changed, 39 insertions(+), 24 deletions(-)

Index: linux-2.6.24-rc2-mm1/mm/memcontrol.c
===
--- linux-2.6.24-rc2-mm1.orig/mm/memcontrol.c
+++ linux-2.6.24-rc2-mm1/mm/memcontrol.c
@@ -89,6 +89,8 @@ enum mem_cgroup_zstat_index {
 };

 struct mem_cgroup_per_zone {
+ struct list_head active_list;
+ struct list_head inactive_list;
  unsigned long count[NR_MEM_CGROUP_ZSTAT];
 };
 /* Macro for accessing counter */
@@ -123,10 +125,7 @@ struct mem_cgroup {
  /*
   * Per cgroup active and inactive list, similar to the
   * per zone LRU lists.
- * TODO: Consider making these lists per zone
- */
- struct list_head active_list;
- struct list_head inactive_list;
+ * TODO: Consider making these lists per zone

```
   */
- struct list_head active_list;
- struct list_head inactive_list;
  struct mem_cgroup_lru_info info;
  /*
   * spin_lock to protect the per cgroup LRU
@@ -369,10 +368,10 @@ static void __mem_cgroup_add_list(struct

  if (!to) {
   MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_INACTIVE) += 1;
-  list_add(&pc->lru, &pc->mem_cgroup->inactive_list);
+  list_add(&pc->lru, &mz->inactive_list);
  } else {
   MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_ACTIVE) += 1;
-  list_add(&pc->lru, &pc->mem_cgroup->active_list);
+  list_add(&pc->lru, &mz->active_list);
  }
  mem_cgroup_charge_statistics(pc->mem_cgroup, pc->flags, true);
 }
@@ -390,11 +389,11 @@ static void __mem_cgroup_move_lists(stru
  if (active) {
   MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_ACTIVE) += 1;
   pc->flags |= PAGE_CGROUP_FLAG_ACTIVE;
-  list_move(&pc->lru, &pc->mem_cgroup->active_list);
+  list_move(&pc->lru, &mz->active_list);
  } else {
   MEM_CGROUP_ZSTAT(mz, MEM_CGROUP_ZSTAT_INACTIVE) += 1;
   pc->flags &= ~PAGE_CGROUP_FLAG_ACTIVE;
-  list_move(&pc->lru, &pc->mem_cgroup->inactive_list);
+  list_move(&pc->lru, &mz->inactive_list);
  }
 }

@@ -518,11 +517,16 @@ unsigned long mem_cgroup_isolate_pages(u
  LIST_HEAD(pc_list);
  struct list_head *src;
  struct page_cgroup *pc, *tmp;
+ int nid = z->zone_pgdat->node_id;
+ int zid = zone_idx(z);
+ struct mem_cgroup_per_zone *mz;

+ mz = mem_cgroup_zoneinfo(mem_cont, nid, zid);
  if (active)
-  src = &mem_cont->active_list;
+  src = &mz->active_list;
  else
-  src = &mem_cont->inactive_list;
+  src = &mz->inactive_list;
```

```
+

  spin_lock(&mem_cont->lru_lock);
  scan = 0;
@@ -544,13 +548,6 @@ unsigned long mem_cgroup_isolate_pages(u
    continue;
   }

-  /*
-   * Reclaim, per zone
-   * TODO: make the active/inactive lists per zone
-   */
-  if (page_zone(page) != z)
-   continue;
-
   scan++;
   list_move(&pc->lru, &pc_list);

@@ -832,6 +829,8 @@ mem_cgroup_force_empty_list(struct mem_c
  int count;
  unsigned long flags;

+ if (list_empty(list))
+  return;
 retry:
  count = FORCE_UNCHARGE_BATCH;
  spin_lock_irqsave(&mem->lru_lock, flags);
@@ -867,20 +866,27 @@ retry:
 int mem_cgroup_force_empty(struct mem_cgroup *mem)
 {
  int ret = -EBUSY;
+ int node, zid;
  css_get(&mem->css);
  /*
   * page reclaim code (kswapd etc..) will move pages between
`  * active_list <-> inactive_list while we don't take a lock.
   * So, we have to do loop here until all lists are empty.
   */
- while (!(list_empty(&mem->active_list) &&
-   list_empty(&mem->inactive_list))) {
+ while (mem->res.usage > 0) {
   if (atomic_read(&mem->css.cgroup->count) > 0)
    goto out;
-  /* drop all page_cgroup in active_list */
-  mem_cgroup_force_empty_list(mem, &mem->active_list);
-  /* drop all page_cgroup in inactive_list */
-  mem_cgroup_force_empty_list(mem, &mem->inactive_list);
+  for_each_node_state(node, N_POSSIBLE)
```

```
+   for (zid = 0; zid < MAX_NR_ZONES; zid++) {
+    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);
+    /* drop all page_cgroup in inactive_list */
+    mem_cgroup_force_empty_list(mem,
+       &mz->inactive_list);
+   }
  }
  ret = 0;
 out:
@@ -1092,15 +1098,25 @@ static struct cftype mem_cgroup_files[]
 static int alloc_mem_cgroup_per_zone_info(struct mem_cgroup *mem, int node)
 {
  struct mem_cgroup_per_node *pn;
+ struct mem_cgroup_per_zone *mz;
+ int zone;

  pn = kmalloc_node(sizeof(*pn), GFP_KERNEL, node);
  if (!pn)
   return 1;
+
  mem->info.nodeinfo[node] = pn;
  memset(pn, 0, sizeof(*pn));
+
+ for (zone = 0; zone < MAX_NR_ZONES; zone++) {
+  mz = &pn->zoneinfo[zone];
+  INIT_LIST_HEAD(&mz->active_list);
+  INIT_LIST_HEAD(&mz->inactive_list);
+ }
  return 0;
 }

+
 static struct mem_cgroup init_mem_cgroup;

 static struct cgroup_subsys_state *
@@ -1119,8 +1135,7 @@ mem_cgroup_create(struct cgroup_subsys *
   return NULL;

  res_counter_init(&mem->res);
- INIT_LIST_HEAD(&mem->active_list);
- INIT_LIST_HEAD(&mem->inactive_list);
+
  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