Subject: [RFC][for -mm] memory controller enhancements for NUMA [2/10] account active inactive
Posted by KAMEZAWA Hiroyuki on Wed, 14 Nov 2007 08:42:33 GMT
View Forum Message <> Reply to Message

Counting active/inactive per-zone in memory controller.

This patch adds per-zone status in memory cgroup.
These values are often read (as per-zone value) by page reclaiming.

In current design, per-zone stat is just a unsigned long value and
not an atomic value because they are modified only under lru_lock.

If we need to modify value withou taking lock, we'll have to use
atomic64_t.

This patch adds TOTAL and INACTIVE values.

TOTAL .... the number of pages of zone in memory cgroup.
INACTIVE .... the number of inactive pages of zone in memory cgroup.

the number of acitve pages is TOTAL - INACTIVE.

This patch turns memory controler's early_init to be 0 for calling
kmalloc().

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

mm/memcontrol.c | 131
++-
1 file changed, 130 insertions(+), 1 deletion(-)

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
@@ -78,6 +78,30 @@ static s64 mem_cgroup_read_stat(struct m
}

/*
+ * some of parameters used for page reclaiming should be counted per zone.
+ * Its array size will be MAX_NUMNODES * MAX_NR_ZONES * # of params.
+ * This seems a bit big for per_cpu....
+ *
+ * Values are modified under mem_cgroup->lru_lock. (now)
+ */

```
+
+enum mem_cgroup_zstat_index {
+ MEM_CGROUP_ZSTAT_TOTAL,
+ MEM_CGROUP_ZSTAT_INACTIVE, /* Active = Total - Inactive */
+
+ NR_MEM_CGROUP_ZSTAT,
+};
+
+struct mem_cgroup_stat_zone {
+ unsigned long count[MAX_NR_ZONES][NR_MEM_CGROUP_ZSTAT];
+};
+
+struct mem_cgroup_zonestat {
+ struct mem_cgroup_stat_zone *nodestat[MAX_NUMNODES];
+};
+
+
+/*
  * The memory controller data structure. The memory controller controls both
  * page cache and RSS per cgroup. We would eventually like to provide
  * statistics based on the statistics developed by Rik Van Riel for clock-pro,
@@ -110,6 +134,10 @@ struct mem_cgroup {
   * statistics.
   */
  struct mem_cgroup_stat stat;
+ /*
+  * Per zone statistics (used for memory reclaim)
+  */
+ struct mem_cgroup_zonestat zstat;
 };

 /*
@@ -168,6 +196,52 @@ static void mem_cgroup_charge_statistics

 }

+/*
+ * Always used with page_cgroup pointer. (now)
+ */
+static void mem_cgroup_add_zonestat(struct page_cgroup *pc,
+   enum mem_cgroup_zstat_index idx, int val)
+{
+ struct mem_cgroup *mem = pc->mem_cgroup;
+ struct mem_cgroup_stat_zone *zs = mem->zstat.nodestat[pc->nid];
+
+ BUG_ON(!zs);
+
+ zs->count[pc->zid][idx] += val;
```

```
+}
+
+static void mem_cgroup_sub_zonestat(struct page_cgroup *pc,
+   enum mem_cgroup_zstat_index idx, int val)
+{
+ struct mem_cgroup *mem = pc->mem_cgroup;
+ struct mem_cgroup_stat_zone *zs = mem->zstat.nodestat[pc->nid];
+
+ BUG_ON(!zs);
+
+ zs->count[pc->zid][idx] -= val;
+}
+
+static u64 mem_cgroup_get_zonestat(struct mem_cgroup *mem,
+     int nid, int zid,
+     enum mem_cgroup_zstat_index idx)
+{
+ struct mem_cgroup_stat_zone *zs = mem->zstat.nodestat[nid];
+ if (!zs)
+  return 0;
+ return zs->count[zid][idx];
+}
+
+static unsigned long mem_cgroup_get_all_zonestat(struct mem_cgroup *mem,
+     enum mem_cgroup_zstat_index idx)
+{
+ int nid, zid;
+ u64 total = 0;
+ for_each_online_node(nid)
+  for (zid = 0; zid < MAX_NR_ZONES; zid++)
+   total += mem_cgroup_get_zonestat(mem, nid, zid, idx);
+ return total;
+}
+
 static struct mem_cgroup init_mem_cgroup;

 static inline
@@ -284,6 +358,13 @@ static struct page_cgroup *clear_page_cg

 static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
 {
+ int from = pc->flags & PAGE_CGROUP_FLAG_ACTIVE;
+
+ if (from && !active) /* active -> inactive */
+  mem_cgroup_add_zonestat(pc, MEM_CGROUP_ZSTAT_INACTIVE, 1);
+ else if (!from & active) /* inactive to active */
+  mem_cgroup_sub_zonestat(pc, MEM_CGROUP_ZSTAT_INACTIVE, 1);
+
```

```
  if (active) {
   pc->flags |= PAGE_CGROUP_FLAG_ACTIVE;
   list_move(&pc->lru, &pc->mem_cgroup->active_list);
@@ -493,6 +574,7 @@ noreclaim:
  pc->flags = PAGE_CGROUP_FLAG_ACTIVE;
  if (ctype == MEM_CGROUP_CHARGE_TYPE_CACHE)
   pc->flags |= PAGE_CGROUP_FLAG_CACHE;
+
  if (page_cgroup_assign_new_page_cgroup(page, pc)) {
   /*
    * an another charge is added to this page already.
@@ -506,6 +588,7 @@ noreclaim:
  }

  spin_lock_irqsave(&mem->lru_lock, flags);
+ mem_cgroup_add_zonestat(pc, MEM_CGROUP_ZSTAT_TOTAL, 1);
  /* Update statistics vector */
  mem_cgroup_charge_statistics(mem, pc->flags, true);
  list_add(&pc->lru, &mem->active_list);
@@ -574,11 +657,16 @@ void mem_cgroup_uncharge(struct page_cgr
    spin_lock_irqsave(&mem->lru_lock, flags);
    list_del_init(&pc->lru);
    mem_cgroup_charge_statistics(mem, pc->flags, false);
+   if (!(pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
+    mem_cgroup_sub_zonestat(pc,
+      MEM_CGROUP_ZSTAT_INACTIVE, 1);
+   mem_cgroup_sub_zonestat(pc, MEM_CGROUP_ZSTAT_TOTAL, 1);
    spin_unlock_irqrestore(&mem->lru_lock, flags);
    kfree(pc);
   }
  }
 }
+
 /*
  * Returns non-zero if a page (under migration) has valid page_cgroup member.
  * Refcnt of page_cgroup is incremented.
@@ -647,10 +735,16 @@ retry:
   /* Avoid race with charge */
   atomic_set(&pc->ref_cnt, 0);
   if (clear_page_cgroup(page, pc) == pc) {
+   int active;
    css_put(&mem->css);
+   active = pc->flags & PAGE_CGROUP_FLAG_ACTIVE;
    res_counter_uncharge(&mem->res, PAGE_SIZE);
    list_del_init(&pc->lru);
    mem_cgroup_charge_statistics(mem, pc->flags, false);
+   if (!(pc->flags & PAGE_CGROUP_FLAG_ACTIVE))
+    mem_cgroup_sub_zonestat(pc,
```

```
+      MEM_CGROUP_ZSTAT_INACTIVE, 1);
+   mem_cgroup_sub_zonestat(pc, MEM_CGROUP_ZSTAT_TOTAL, 1);
    kfree(pc);
   } else  /* being uncharged ? ...do relax */
    break;
@@ -829,6 +923,17 @@ static int mem_control_stat_show(struct
   seq_printf(m, "%s %lld\n", mem_cgroup_stat_desc[i].msg,
     (long long)val);
  }
+ /* showing # of active pages */
+ {
+  unsigned long total, inactive;
+
+  inactive = mem_cgroup_get_all_zonestat(mem_cont,
+      MEM_CGROUP_ZSTAT_INACTIVE);
+  total = mem_cgroup_get_all_zonestat(mem_cont,
+      MEM_CGROUP_ZSTAT_TOTAL);
+  seq_printf(m, "active %ld\n", (total - inactive) * PAGE_SIZE);
+  seq_printf(m, "inactive %ld\n", (inactive) * PAGE_SIZE);
+ }
  return 0;
 }

@@ -888,6 +993,7 @@ static struct cgroup_subsys_state *
 mem_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
 {
  struct mem_cgroup *mem;
+ int node;

  if (unlikely((cont->parent) == NULL)) {
   mem = &init_mem_cgroup;
@@ -903,7 +1009,24 @@ mem_cgroup_create(struct cgroup_subsys *
  INIT_LIST_HEAD(&mem->inactive_list);
  spin_lock_init(&mem->lru_lock);
  mem->control_type = MEM_CGROUP_TYPE_ALL;
+ memset(&mem->zstat, 0, sizeof(mem->zstat));
+
+ for_each_node_state(node, N_POSSIBLE) {
+  int size = sizeof(struct mem_cgroup_stat_zone);
+
+  mem->zstat.nodestat[node] =
+    kmalloc_node(size, GFP_KERNEL, node);
+  if (mem->zstat.nodestat[node] == NULL)
+   goto free_out;
+  memset(mem->zstat.nodestat[node], 0, size);
+ }
  return &mem->css;
+free_out:
```

```
+ for_each_node_state(node, N_POSSIBLE)
+  kfree(mem->zstat.nodestat[node]);
+ if (cont->parent != NULL)
+  kfree(mem);
+ return NULL;
 }

 static void mem_cgroup_pre_destroy(struct cgroup_subsys *ss,
@@ -916,6 +1039,12 @@ static void mem_cgroup_pre_destroy(struc
 static void mem_cgroup_destroy(struct cgroup_subsys *ss,
     struct cgroup *cont)
 {
+ int node;
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+
+ for_each_node_state(node, N_POSSIBLE)
+  kfree(mem->zstat.nodestat[node]);
+
  kfree(mem_cgroup_from_cont(cont));
 }

@@ -968,5 +1097,5 @@ struct cgroup_subsys mem_cgroup_subsys =
  .destroy = mem_cgroup_destroy,
  .populate = mem_cgroup_populate,
  .attach = mem_cgroup_move_task,
- .early_init = 1,
+ .early_init = 0,
 };
```

_____

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