
Subject: [PATCH][just for review] memory controller enhancements [4/5] statistics for memory cgroup

Posted by [KAMEZAWA Hiroyuki](#) on Thu, 11 Oct 2007 09:54:22 GMT

[View Forum Message](#) <> [Reply to Message](#)

Add statistics account infrastructure for memory controller.

Changes from original:

- divided into 2 patch (this one and show info)
- changed from u64 to s64
- added mem_cgroup_stat_add() and batched statistics modification logic.
- removed stat init code because mem_cgroup is allocated by kzalloc().

Problem:

- charge/uncharge count can be overflow. But they are unnecessary ?

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

Signed-off-by: YAMAMOTO Takashi <yamamoto@valinux.co.jp>

mm/memcontrol.c | 120

+++++

1 file changed, 116 insertions(+), 4 deletions(-)

Index: devel-2.6.23-rc8-mm2/mm/memcontrol.c

=====

--- devel-2.6.23-rc8-mm2.orig/mm/memcontrol.c

+++ devel-2.6.23-rc8-mm2/mm/memcontrol.c

@@ -35,6 +35,63 @@ struct cgroup_subsys mem_cgroup_subsys;

static const int MEM_CGROUP_RECLAIM_RETRIES = 5;

/*

+ * Statistics for memory cgroup.

+ */

+enum mem_cgroup_stat_index {

+ /*

+ * For MEM_CONTAINER_TYPE_ALL, usage = pagecache + rss.

+ */

+ MEM_CGROUP_STAT_PAGECACHE, /* # of pages charged as cache */

+ MEM_CGROUP_STAT_RSS, /* # of pages charged as rss */

+

+ /*

+ * usage = charge - uncharge.

+ */

+ MEM_CGROUP_STAT_CHARGE, /* # of pages charged */

+ MEM_CGROUP_STAT_UNCHARGE, /* # of pages uncharged */

+

+ MEM_CGROUP_STAT_ACTIVE, /* # of pages in active list */

+ MEM_CGROUP_STAT_INACTIVE, /* # of pages on inactive list */

```

+
+ MEM_CGROUP_STAT_NSTATS,
+};
+
+struct mem_cgroup_stat_cpu {
+ s64 count[MEM_CGROUP_STAT_NSTATS];
+} ____cacheline_aligned_in_smp;
+
+struct mem_cgroup_stat {
+ struct mem_cgroup_stat_cpu cpustat[NR_CPUS];
+};
+
+/*
+ * For batching....mem_cgroup_charge_statistics()(see below).
+ */
+static inline void mem_cgroup_stat_add(struct mem_cgroup_stat *stat,
+      enum mem_cgroup_stat_index idx, int val)
+{
+ int cpu = smp_processor_id();
+ stat->cpustat[cpu].count[idx] += val;
+}
+
+static inline void mem_cgroup_stat_inc(struct mem_cgroup_stat *stat,
+      enum mem_cgroup_stat_index idx)
+{
+ preempt_disable();
+ mem_cgroup_stat_add(stat, idx, 1);
+ preempt_enable();
+}
+
+static inline void mem_cgroup_stat_dec(struct mem_cgroup_stat *stat,
+      enum mem_cgroup_stat_index idx)
+{
+ preempt_disable();
+ mem_cgroup_stat_add(stat, idx, -1);
+ preempt_enable();
+}
+
+/*
+ * 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,
+ @@ -63,6 +120,10 @@ struct mem_cgroup {
+ */
+ spinlock_t lru_lock;
+ unsigned long control_type; /* control RSS or RSS+Pagecache */
+ /*

```

```

+ * statistics.
+ */
+ struct mem_cgroup_stat stat;
+ };

+/*
@@ -96,6 +157,37 @@ enum {
    MEM_CGROUP_TYPE_MAX,
+ };

+/*
+ * Batched statistics modification.
+ * We have to modify several values at charge/uncharge..
+ */
+static inline void
+mem_cgroup_charge_statistics(struct mem_cgroup *mem, int flags, bool charge)
+{
+ int val = (charge)? 1 : -1;
+ preempt_disable();
+
+ if (flags & PCGF_PAGECACHE)
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_PAGECACHE, val);
+ else
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_RSS, val);
+
+ if (flags & PCGF_ACTIVE)
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_ACTIVE, val);
+ else
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_INACTIVE, val);
+
+ if(charge)
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_CHARGE, 1);
+ else
+ mem_cgroup_stat_add(&mem->stat, MEM_CGROUP_STAT_UNCHARGE, 1);
+
+ preempt_enable();
+}
+
+
+
+
+static struct mem_cgroup init_mem_cgroup;

static inline
@@ -207,8 +299,23 @@ clear_page_cgroup(struct page *page, str
+ }

```

```

-static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
+static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active,
+    struct mem_cgroup *mem)
{
+ int moved = 0;
+
+ if (active && (pc->flags & PCGF_ACTIVE) == 0)
+ moved = 1; /* Move from inactive to active */
+ else if (!active && (pc->flags & PCGF_ACTIVE))
+ moved = -1; /* Move from active to inactive */
+
+ if (moved) {
+ struct mem_cgroup_stat *stat = &mem->stat;
+ preempt_disable();
+ mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_ACTIVE, moved);
+ mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_INACTIVE, -moved);
+ preempt_enable();
+ }
+ if (active) {
+ pc->flags |= PCGF_ACTIVE;
+ list_move(&pc->lru, &pc->mem_cgroup->active_list);
@@ -230,7 +337,7 @@ void mem_cgroup_move_lists(struct page_c
+ mem = pc->mem_cgroup;

+ spin_lock(&mem->lru_lock);
- __mem_cgroup_move_lists(pc, active);
+ __mem_cgroup_move_lists(pc, active, mem);
+ spin_unlock(&mem->lru_lock);
+ }

@@ -267,12 +374,12 @@ unsigned long mem_cgroup_isolate_pages(u
+ }

+ if (PageActive(page) && !active) {
- __mem_cgroup_move_lists(pc, true);
+ __mem_cgroup_move_lists(pc, true, pc->mem_cgroup);
+ scan--;
+ continue;
+ }
+ if (!PageActive(page) && active) {
- __mem_cgroup_move_lists(pc, false);
+ __mem_cgroup_move_lists(pc, false, pc->mem_cgroup);
+ scan--;
+ continue;
+ }
@@ -428,6 +535,9 @@ noreclaim:
+ goto retry;
+ }

```

```

+ /* Update statistics vector */
+ mem_cgroup_charge_statistics(mem, pc->flags, true);
+
+ spin_lock_irqsave(&mem->lru_lock, flags);
+ list_add(&pc->lru, &mem->active_list);
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
@@ -494,6 +604,7 @@ void mem_cgroup_uncharge(struct page_cgr
+ spin_lock_irqsave(&mem->lru_lock, flags);
+ list_del_init(&pc->lru);
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
+ mem_cgroup_charge_statistics(mem, pc->flags, false);
+ kfree(pc);
+ }
+ }
@@ -566,6 +677,7 @@ mem_cgroup_force_empty_list(struct mem_c
+ css_put(&mem->css);
+ res_counter_uncharge(&mem->res, PAGE_SIZE);
+ list_del_init(&pc->lru);
+ mem_cgroup_charge_statistics(mem, pc->flags, false);
+ kfree(pc);
+ } else
+ count = 1; /* being uncharged ? ...do relax */

```

Containers mailing list

Containers@lists.linux-foundation.org

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