
Subject: [RFC][for -mm] memory cgroup enhancements take3 [5/9] per-cpu status accounting

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 30 Oct 2007 11:18:33 GMT

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

Add statistics account infrastructure for memory controller.

Changelov v2 -> v3

- adjusted to rename of #define PAGE_CGROUP_FLAG....

- dropped ACTIVE/INACTIVE counter.

They should be accounted against per zone. Then, using pcp counter, we need array of NR_CPU * MAX_NUMNODES * NR_ZONES against all stats.

This is too big for statistics *per-memory-cgroup*.

ACTIVE/INACTIVE counter is added as per-zone statistics later.

Changelog v1 -> v2

- Removed Charge/Uncharge counter

- reflected comments.

- changes __move_lists() args.

- changes __mem_cgroup_stat_add() name, comment and added VM_BUGON

Changes from original:

- divided into 2 patch (account 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().

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

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

mm/memcontrol.c | 83

+++++

1 file changed, 83 insertions(+)

Index: devel-2.6.23-mm1/mm/memcontrol.c

=====

--- devel-2.6.23-mm1.orig/mm/memcontrol.c

+++ devel-2.6.23-mm1/mm/memcontrol.c

@@ -35,6 +35,58 @@ 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 */
+
+ 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).
+ * MUST be called under preempt_disable().
+ */
+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();
+#ifdef CONFIG_PREEMPT
+ VM_BUG_ON(preempt_count() == 0);
+#endif
+ 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 +115,10 @@ struct mem_cgroup {
    */
    spinlock_t lru_lock;
    unsigned long control_type; /* control RSS or RSS+Pagecache */
+ /*
+ * statistics.
+ */
+ struct mem_cgroup_stat stat;
};

/*
@@ -101,6 +157,28 @@ enum charge_type {
    MEM_CGROUP_CHARGE_TYPE_MAPPED = 0,
};

+/*
+ * 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, int charge)
+{
+ int val = (charge)? 1 : -1;
+ struct mem_cgroup_stat *stat = &mem->stat;
+ preempt_disable();
+
+ if (flags & PAGE_CGROUP_FLAG_CACHE)
+ __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_PAGECACHE, val);
+ else
+ __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_RSS, val);
+
+ preempt_enable();
+}
+
+
+
+
+ static struct mem_cgroup init_mem_cgroup;

static inline
@@ -444,6 +522,9 @@ 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);
@@ -511,6 +592,7 @@ 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);
}
}
@@ -586,6 +668,7 @@ retry:
  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 /* being uncharged ? ...do relax */
  break;

```

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