Subject: [RFC][PATCH] memory cgroup enhancements updated [5/10] per cpu accounting
Posted by KAMEZAWA Hiroyuki on Fri, 19 Oct 2007 09:32:03 GMT
View Forum Message <> Reply to Message

Add statistics account infrastructure for memory controller.

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 |  120 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++----
 1 file changed, 113 insertions(+), 7 deletions(-)

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 +35,64 @@ 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_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).
+ * 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 +121,10 @@ struct mem_cgroup {
```

```
   */
  spinlock_t lru_lock;
  unsigned long control_type; /* control RSS or RSS+Pagecache */
+ /*
+  * statistics.
+  */
+ struct mem_cgroup_stat stat;
 };

 /*
@@ -96,6 +158,33 @@ 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, int charge)
+{
+ int val = (charge)? 1 : -1;
+ struct mem_cgroup_stat *stat = &mem->stat;
+ preempt_disable();
+
+ if (flags & PCGF_PAGECACHE)
+  __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_PAGECACHE, val);
+ else
+  __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_RSS, val);
+
+ if (flags & PCGF_ACTIVE)
+  __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_ACTIVE, val);
+ else
+  __mem_cgroup_stat_add(stat, MEM_CGROUP_STAT_INACTIVE, val);
+
+ preempt_enable();
+}
+
+
+
+
 static struct mem_cgroup init_mem_cgroup;

 static inline
@@ -209,12 +298,27 @@ clear_page_cgroup(struct page *page, str

 static void __mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
 {
```

```
+ int moved = 0;
+ struct mem_cgroup *mem = pc->mem_cgroup;
+
+ 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);
+  list_move(&pc->lru, &mem->active_list);
  } else {
   pc->flags &= ~PCGF_ACTIVE;
-  list_move(&pc->lru, &pc->mem_cgroup->inactive_list);
+  list_move(&pc->lru, &mem->inactive_list);
  }
 }

@@ -233,15 +337,12 @@ int task_in_mem_cgroup(struct task_struc
  */
 void mem_cgroup_move_lists(struct page_cgroup *pc, bool active)
 {
- struct mem_cgroup *mem;
  if (!pc)
   return;

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

 unsigned long mem_cgroup_isolate_pages(unsigned long nr_to_scan,
@@ -440,6 +541,9 @@ 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);
@@ -505,6 +609,7 @@ 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);
   }
  }
@@ -580,6 +685,7 @@ 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