Subject: [PATCH][for -mm] Fix and Enhancements for memory cgroup [6/6] add force reclaim interface

Posted by KAMEZAWA Hiroyuki on Tue, 09 Oct 2007 09:55:10 GMT

View Forum Message <> Reply to Message

This patch adds an interface "memory.force_reclaim".
Any write to this file will drop all charges in this cgroup if
there is no task under.

```
%echo 1 > /....../memory.force_reclaim
```

will drop all charges of memory cgroup if cgroup's tasks is empty.

This is useful to invoke rmdir() against memory cgroup successfully.

Tested and worked well on x86_64/fake-NUMA system.

Changelog:
 - added a new interface force_relcaim.
 - changes spin_lock to spin_lock_irqsave().

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

```
 mm/memcontrol.c |   79
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 79 insertions(+)

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
@@ -507,6 +507,55 @@ retry:
 	return;
 }

+static void
+mem_cgroup_force_reclaim_list(struct mem_cgroup *mem, struct list_head *list)
+{
+	struct page_cgroup *pc;
+	struct page *page;
+	int count = SWAP_CLUSTER_MAX;
+	unsigned long flags;
+
+	spin_lock_irqsave(&mem->lru_lock, flags);
+
+	while (!list_empty(list)) {
```

```
+  pc = list_entry(list->prev, struct page_cgroup, lru);
+  page = pc->page;
+  if (clear_page_cgroup(page, pc) == pc) {
+   css_put(&mem->css);
+   res_counter_uncharge(&mem->res, PAGE_SIZE);
+   list_del_init(&pc->lru);
+   kfree(pc);
+  } else
+   count = 1; /* race? ...do relax */
+
+  if (--count == 0) {
+   spin_unlock_irqrestore(&mem->lru_lock, flags);
+   cond_resched();
+   spin_lock_irqsave(&mem->lru_lock, flags);
+   count = SWAP_CLUSTER_MAX;
+  }
+ }
+ spin_unlock_irqrestore(&mem->lru_lock, flags);
+}
+
+int mem_cgroup_force_reclaim(struct mem_cgroup *mem)
+{
+ int ret = -EBUSY;
+ while (!list_empty(&mem->active_list) ||
+        !list_empty(&mem->inactive_list)) {
+  if (atomic_read(&mem->css.cgroup->count) > 0)
+   goto out;
+  mem_cgroup_force_reclaim_list(mem, &mem->active_list);
+  mem_cgroup_force_reclaim_list(mem, &mem->inactive_list);
+ }
+ ret = 0;
+out:
+ css_put(&mem->css);
+ return ret;
+}
+
+
+
 int mem_cgroup_write_strategy(char *buf, unsigned long long *tmp)
 {
  *tmp = memparse(buf, &buf);
@@ -592,6 +641,31 @@ static ssize_t mem_control_type_read(str
    ppos, buf, s - buf);
 }

+
+static ssize_t mem_force_reclaim_write(struct cgroup *cont,
+    struct cftype *cft, struct file *file,
+    const char __user *userbuf,
+    size_t nbytes, loff_t *ppos)
+{
+ struct mem_cgroup *mem = mem_cgroup_from_cont(cont);
+ int ret;
+ ret = mem_cgroup_force_reclaim(mem);
+ if (!ret)
+  ret = nbytes;
+ return ret;
+}
+
+static ssize_t mem_force_reclaim_read(struct cgroup *cont,
+    struct cftype *cft,
+    struct file *file, char __user *userbuf,
+    size_t nbytes, loff_t *ppos)
+{
+ char buf[2] = "0";
+ return simple_read_from_buffer((void __user *)userbuf, nbytes,
+   ppos, buf, strlen(buf));
+}
+
+
 static struct cftype mem_cgroup_files[] = {
  {
   .name = "usage_in_bytes",
@@ -614,6 +688,11 @@ static struct cftype mem_cgroup_files[]
   .write = mem_control_type_write,
   .read = mem_control_type_read,
  },
+ {
+  .name = "force_reclaim",
+  .write = mem_force_reclaim_write,
+  .read = mem_force_reclaim_read,
+ },
 };

 static struct mem_cgroup init_mem_cgroup;
```