Subject: [PATCH -mm 4/5] swapcgroup (v3): modify vm_swap_full()
Posted by Daisuke Nishimura on Fri, 04 Jul 2008 06:22:44 GMT
View Forum Message <> Reply to Message

This patch modifies vm_swap_full() to calculate
the rate of swap usage per cgroup.

The purpose of this change is to free freeable swap caches
(that is, swap entries) per cgroup, so that swap_cgroup_charge()
fails less frequently.

I tested whether this patch can reduce the swap usage or not,
by running qsbench (8 x 40M) 10 times in mem:128M/swap:256M group
and mem:256M/swap:128M group.
And I confirmed that this patch can keep swap usage to
(half of the limit < usage < the limit),
whereas without this patch, swap usage tends to keep near to the limit.

Change log
v2->v3
- Rebased on 2.6.26-rc5-mm3
- take into account global swap usage.
- change arg of vm_swap_full from page to memcg.
- add check on mem_cgroup_subsys.disabled
v1->v2
- new patch

Signed-off-by: Daisuke Nishimura <nishimura@mxp.nes.nec.co.jp>

---
 include/linux/memcontrol.h | 12 ++++++++++
 include/linux/swap.h | 4 +++-
 mm/memcontrol.c | 18 ++++++++++++++++++
 mm/memory.c | 4 +++-
 mm/swapfile.c | 38 ++++++++++++++++++++++++++++++++++++++-
 mm/vmscan.c | 6 ++++---
 6 files changed, 76 insertions(+), 6 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index f3c84f7..1e6eb0c 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -175,6 +175,8 @@ extern int swap_cgroup_charge(struct page *page,
    unsigned long offset);
 extern void swap_cgroup_uncharge(struct swap_info_struct *si,
     unsigned long offset);

```
+extern struct mem_cgroup *page_to_memcg(struct page *page);
+extern int swap_cgroup_vm_swap_full(struct mem_cgroup *memcg);
 #else
 static inline
 struct mem_cgroup **swap_info_clear_memcg(struct swap_info_struct *p)
@@ -199,6 +201,16 @@ static inline void swap_cgroup_uncharge(struct swap_info_struct *si,
     unsigned long offset)
 {
 }
+
+static inline struct mem_cgroup *page_to_memcg(struct page *page)
+{
+ return NULL;
+}
+
+static inline int swap_cgroup_vm_swap_full(struct mem_cgroup *memcg)
+{
+ return 0;
+}
 #endif

 #endif /* _LINUX_MEMCONTROL_H */
diff --git a/include/linux/swap.h b/include/linux/swap.h
index f80255b..2b95df9 100644
--- a/include/linux/swap.h
+++ b/include/linux/swap.h
@@ -161,7 +161,9 @@ struct swap_list_t {
 };

 /* Swap 50% full? Release swapcache more aggressively.. */
-#define vm_swap_full() (nr_swap_pages*2 < total_swap_pages)
+#define vm_swap_full(memcg) ((nr_swap_pages*2 < total_swap_pages) \
+    || swap_cgroup_vm_swap_full(memcg))
+

 /* linux/mm/page_alloc.c */
 extern unsigned long totalram_pages;
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index d16d0a5..160fca1 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -1296,5 +1296,23 @@ void swap_cgroup_uncharge(struct swap_info_struct *si,
   css_put(&mem->css);
  }
 }
+
+int swap_cgroup_vm_swap_full(struct mem_cgroup *memcg)
+{
```

```
+ u64 usage;
+ u64 limit;
+
+ if (memcg)
+  css_get(&memcg->css);
+ else
+  return 0;
+
+ usage = res_counter_read_u64(&memcg->swap_res, RES_USAGE);
+ limit = res_counter_read_u64(&memcg->swap_res, RES_LIMIT);
+
+ css_put(&memcg->css);
+
+ return usage * 2 > limit;
+}
 #endif

diff --git a/mm/memory.c b/mm/memory.c
index 536d748..afcf737 100644
--- a/mm/memory.c
+++ b/mm/memory.c
@@ -2230,7 +2230,9 @@ static int do_swap_page(struct mm_struct *mm, struct vm_area_struct
*vma,
  page_add_anon_rmap(page, vma, address);

  swap_free(entry);
- if (vm_swap_full() || (vma->vm_flags & VM_LOCKED) || PageMlocked(page))
+ if (vm_swap_full(page_to_memcg(page))
+  || (vma->vm_flags & VM_LOCKED)
+  || PageMlocked(page))
   remove_exclusive_swap_page(page);
  unlock_page(page);

diff --git a/mm/swapfile.c b/mm/swapfile.c
index 57798c5..6a863bc 100644
--- a/mm/swapfile.c
+++ b/mm/swapfile.c
@@ -28,6 +28,7 @@
 #include <linux/capability.h>
 #include <linux/syscalls.h>
 #include <linux/memcontrol.h>
+#include <linux/cgroup.h>

 #include <asm/pgtable.h>
 #include <asm/tlbflush.h>
@@ -447,7 +448,8 @@ void free_swap_and_cache(swp_entry_t entry)
   /* Only cache user (+us), or swap space full? Free it! */
   /* Also recheck PageSwapCache after page is locked (above) */
```

```
   if (PageSwapCache(page) && !PageWriteback(page) &&
-      (one_user || vm_swap_full())) {
+      (one_user
+      || vm_swap_full(page_to_memcg(page)))) {
     delete_from_swap_cache(page);
     SetPageDirty(page);
   }
@@ -1889,3 +1891,37 @@ int valid_swaphandles(swp_entry_t entry, unsigned long *offset)
  *offset = ++toff;
  return nr_pages? ++nr_pages: 0;
 }
+
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+/*
+ * returns mem_cgroup to which the swap cache page is charged as swap.
+ * should be called with the page locked.
+ */
+struct mem_cgroup *page_to_memcg(struct page *page)
+{
+ struct swap_info_struct *p;
+ struct mem_cgroup *mem = NULL;
+ swp_entry_t entry;
+
+ BUG_ON(!PageLocked(page));
+
+ if (mem_cgroup_subsys.disabled)
+  goto out;
+
+ if (!PageSwapCache(page))
+  goto out;
+
+ entry.val = page_private(page);
+ p = swap_info_get(entry);
+ if (!p)
+  goto out;
+
+ mem = p->memcg[swp_offset(entry)];
+
+ spin_unlock(&swap_lock);
+
+out:
+ return mem;
+}
+#endif
+
diff --git a/mm/vmscan.c b/mm/vmscan.c
index 9a5e423..ac45993 100644
--- a/mm/vmscan.c
```

```
+++ b/mm/vmscan.c
@@ -752,7 +752,7 @@ cull_mlocked:

 activate_locked:
   /* Not a candidate for swapping, so reclaim swap space. */
-  if (PageSwapCache(page) && vm_swap_full())
+  if (PageSwapCache(page) && vm_swap_full(page_to_memcg(page)))
    remove_exclusive_swap_page_ref(page);
   VM_BUG_ON(PageActive(page));
   SetPageActive(page);
@@ -1317,7 +1317,7 @@ static void shrink_active_list(unsigned long nr_pages, struct zone
*zone,
    __mod_zone_page_state(zone, NR_LRU_BASE + lru, pgmoved);
    pgmoved = 0;
    spin_unlock_irq(&zone->lru_lock);
-   if (vm_swap_full())
+   if (vm_swap_full(sc->mem_cgroup))
     pagevec_swap_free(&pvec);
    __pagevec_release(&pvec);
    spin_lock_irq(&zone->lru_lock);
@@ -1328,7 +1328,7 @@ static void shrink_active_list(unsigned long nr_pages, struct zone
*zone,
  __count_zone_vm_events(PGREFILL, zone, pgscanned);
  __count_vm_events(PGDEACTIVATE, pgdeactivate);
  spin_unlock_irq(&zone->lru_lock);
- if (vm_swap_full())
+ if (vm_swap_full(sc->mem_cgroup))
   pagevec_swap_free(&pvec);

  pagevec_release(&pvec);
```