Subject: [PATCH 4/4] swapcgroup: modify vm_swap_full for cgroup
Posted by Daisuke Nishimura on Thu, 22 May 2008 06:22:24 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 think I can free freeable swap caches more agressively
with one of Rik's splitlru patchset:

  [patch 04/14] free swap space on swap-in/activation

but, I should verify whether this change to vm_swap_full()
is valid.

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

---
 include/linux/memcontrol.h | 3 +++
 include/linux/swap.h | 6 +++++-
 mm/memcontrol.c | 10 ++++++++++
 mm/memory.c | 2 +-
 mm/swapfile.c | 35 +++++++++++++++++++++++++++++++++++-
 5 files changed, 53 insertions(+), 3 deletions(-)

diff --git a/include/linux/memcontrol.h b/include/linux/memcontrol.h
index a7e6621..256b298 100644
--- a/include/linux/memcontrol.h
+++ b/include/linux/memcontrol.h
@@ -179,6 +179,9 @@ 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 int swap_cgroup_vm_swap_full(struct page *page);
+extern u64 swap_cgroup_read_usage(struct mem_cgroup *mem);
+extern u64 swap_cgroup_read_limit(struct mem_cgroup *mem);
 #else /* CONFIG_CGROUP_SWAP_RES_CTLR */
 static inline int swap_cgroup_charge(struct page *page,
    struct swap_info_struct *si,
diff --git a/include/linux/swap.h b/include/linux/swap.h
index 18887f0..ef156c9 100644
--- a/include/linux/swap.h
+++ b/include/linux/swap.h

```
@@ -159,8 +159,12 @@ struct swap_list_t {
  int next; /* swapfile to be used next */
 };

+#ifndef CONFIG_CGROUP_SWAP_RES_CTLR
 /* Swap 50% full? Release swapcache more aggressively.. */
-#define vm_swap_full() (nr_swap_pages*2 < total_swap_pages)
+#define vm_swap_full(page) (nr_swap_pages*2 < total_swap_pages)
+#else
+#define vm_swap_full(page) swap_cgroup_vm_swap_full(page)
+#endif

 /* linux/mm/page_alloc.c */
 extern unsigned long totalram_pages;
diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index 84e803d..58d72ca 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -1265,5 +1265,15 @@ void swap_cgroup_uncharge(struct swap_info_struct *si,
   css_put(&mem->css);
  }
 }
+
+u64 swap_cgroup_read_usage(struct mem_cgroup *mem)
+{
+ return res_counter_read_u64(&mem->swap_res, RES_USAGE);
+}
+
+u64 swap_cgroup_read_limit(struct mem_cgroup *mem)
+{
+ return res_counter_read_u64(&mem->swap_res, RES_LIMIT);
+}
 #endif

diff --git a/mm/memory.c b/mm/memory.c
index df8f0e9..be2ff96 100644
--- a/mm/memory.c
+++ b/mm/memory.c
@@ -2175,7 +2175,7 @@ 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())
+ if (vm_swap_full(page))
   remove_exclusive_swap_page(page);
  unlock_page(page);
```

```
diff --git a/mm/swapfile.c b/mm/swapfile.c
index 682b71e..9256c2d 100644
--- a/mm/swapfile.c
+++ b/mm/swapfile.c
@@ -429,7 +429,7 @@ 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))) {
    delete_from_swap_cache(page);
    SetPageDirty(page);
   }
@@ -1892,3 +1892,36 @@ int valid_swaphandles(swp_entry_t entry, unsigned long *offset)
  *offset = ++toff;
  return nr_pages? ++nr_pages: 0;
 }
+
+#ifdef CONFIG_CGROUP_SWAP_RES_CTLR
+int swap_cgroup_vm_swap_full(struct page *page)
+{
+ int ret;
+ struct swap_info_struct *p;
+ struct mem_cgroup *mem;
+ u64 usage;
+ u64 limit;
+ swp_entry_t entry;
+
+ VM_BUG_ON(!PageLocked(page));
+ VM_BUG_ON(!PageSwapCache(page));
+
+ ret = 0;
+ entry.val = page_private(page);
+ p = swap_info_get(entry);
+ if (!p)
+  goto out;
+
+ mem = p->memcg[swp_offset(entry)];
+ usage = swap_cgroup_read_usage(mem) / PAGE_SIZE;
+ limit = swap_cgroup_read_limit(mem) / PAGE_SIZE;
+ limit = (limit < total_swap_pages) ? limit : total_swap_pages;
+
+ ret = usage * 2 > limit;
+
+ spin_unlock(&swap_lock);
+
+out:
+ return ret;
```

+}
+#endif

_____

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