

---

Subject: [PATCH 6/6 mm] memcgroup: revert swap\_state mods  
Posted by [Hugh Dickins](#) on Fri, 09 Nov 2007 07:14:22 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

If we're charging rss and we're charging cache, it seems obvious that we should be charging swapcache - as has been done. But in practice that doesn't work out so well: both swapin readahead and swapoff leave the majority of pages charged to the wrong cgroup (the cgroup that happened to read them in, rather than the cgroup to which they belong).

(Which is why unuse\_pte's GFP\_KERNEL while holding pte lock never showed up as a problem: no allocation was ever done there, every page read being already charged to the cgroup which initiated the swapoff.)

It all works rather better if we leave the charging to do\_swap\_page and unuse\_pte, and do nothing for swapcache itself: revert mm/swap\_state.c to what it was before the memory-controller patches. This also speeds up significantly a contained process working at its limit: because it no longer needs to keep waiting for swap writeback to complete.

Is it unfair that swap pages become uncharged once they're unmapped, even though they're still clearly private to particular cgroups? For a short while, yes; but PageReclaim arranges for those pages to go to the end of the inactive list and be reclaimed soon if necessary.

shmem/tmpfs pages are a distinct case: their charging also benefits from this change, but their second life on the lists as swapcache pages may prove more unfair - that I need to check next.

Signed-off-by: Hugh Dickins <[hugh@veritas.com](mailto:hugh@veritas.com)>

---

Insert just after 5/6: the tree builds okay if it goes earlier, just after memory-controller-bug\_on.patch, but 5/6 fixes OOM made more likely by 6/6. Alternatively, hand edit all of the mm/swap\_state.c mods out of all of the memory-controller patches which modify it.

```
mm/swap_state.c | 15 ++-----  
1 file changed, 2 insertions(+), 13 deletions(-)
```

```
--- patch5/mm/swap_state.c 2007-11-08 15:58:50.000000000 +0000  
+++ patch6/mm/swap_state.c 2007-11-08 16:01:11.000000000 +0000  
@@ -17,7 +17,6 @@  
#include <linux/backing-dev.h>  
#include <linux/pagevec.h>  
#include <linux/migrate.h>  
-#include <linux/memcontrol.h>  
  
#include <asm/pgtable.h>
```

```

@@ -79,11 +78,6 @@ static int __add_to_swap_cache(struct pa
    BUG_ON(!PageLocked(page));
    BUG_ON(PageSwapCache(page));
    BUG_ON(PagePrivate(page));
-
- error = mem_cgroup_cache_charge(page, current->mm, gfp_mask);
- if (error)
- goto out;
-
    error = radix_tree_preload(gfp_mask);
    if (!error) {
        write_lock_irq(&swapper_space.tree_lock);
@@ -95,14 +89,10 @@ static int __add_to_swap_cache(struct pa
    set_page_private(page, entry.val);
    total_swapcache_pages++;
    __inc_zone_page_state(page, NR_FILE_PAGES);
- } else
- mem_cgroup_uncharge_page(page);
-
+ }
    write_unlock_irq(&swapper_space.tree_lock);
    radix_tree_preload_end();
- } else
- mem_cgroup_uncharge_page(page);
-out:
+ }
    return error;
}

@@ -143,7 +133,6 @@ void __delete_from_swap_cache(struct pag
    BUG_ON(PageWriteback(page));
    BUG_ON(PagePrivate(page));

- mem_cgroup_uncharge_page(page);
    radix_tree_delete(&swapper_space.page_tree, page_private(page));
    set_page_private(page, 0);
    ClearPageSwapCache(page);

```

---

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

---