
Subject: [BUGFIX][RFC][PATCH][only -mm] FIX memory leak in memory cgroup vs. page migration [0/1]

Posted by [KAMEZAWA Hiroyuki](#) on Tue, 02 Oct 2007 09:30:31 GMT

[View Forum Message](#) <> [Reply to Message](#)

Current implementation of memory cgroup controller does following in migration.

1. uncharge when unmapped.
2. charge again when remapped.

Consider migrate a page from OLD to NEW.

In following case, memory (for page_cgroup) will leak.

1. charge OLD page as page-cache. (charge = 1)
2. A process mmap OLD page. (charge + 1 = 2)
3. A process migrates it.
 - try_to_unmap(OLD) (charge - 1 = 1)
 - replace OLD with NEW
 - remove_migration_pte(NEW) (New is newly charged.)
 - discard OLD page. (page_cgroup for OLD page is not reclaimed.)

patch is in the next mail.

Test Log on 2.6.18-rc8-mm2.

```
==
# mount cgroup and create group_A group_B
[root@drpq kamezawa]# mount -t cgroup none /opt/mem_control/ -o memory
[root@drpq kamezawa]# mkdir /opt/mem_control/group_A/
[root@drpq kamezawa]# mkdir /opt/mem_control/group_B/
[root@drpq kamezawa]# bash
[root@drpq kamezawa]# echo $$ > /opt/mem_control/group_A/tasks
[root@drpq kamezawa]# cat /opt/mem_control/group_A/memory.usage_in_bytes
475136
[root@drpq kamezawa]# grep size-64 /proc/slabinfo
size-64(DMA)      0  0  64 240  1 : tunables 120 60  8 : slabdata  0  0  0
size-64          30425 30960  64 240  1 : tunables 120 60  8 : slabdata 129 129 12

# charge file cache 512Mfile to groupA
[root@drpq kamezawa]# cat 512Mfile > /dev/null
[root@drpq kamezawa]# cat /opt/mem_control/group_A/memory.usage_in_bytes
539525120

# for test, try drop_caches. drop_cache works well and chage decreased.
[root@drpq kamezawa]# echo 3 > /proc/sys/vm/drop_caches
[root@drpq kamezawa]# cat /opt/mem_control/group_A/memory.usage_in_bytes
983040
```

```

# chage file cache 512Mfile again.
[root@drpq kamezawa]# taskset 01 cat 512Mfile > /dev/null
[root@drpq kamezawa]# exit
exit
[root@drpq kamezawa]# cat /opt/mem_control/group_?/memory.usage_in_bytes
539738112
0
[root@drpq kamezawa]# bash
#enter group B
[root@drpq kamezawa]# echo $$ > /opt/mem_control/group_B/tasks
[root@drpq kamezawa]# cat /opt/mem_control/group_?/memory.usage_in_bytes
539738112
557056
[root@drpq kamezawa]# grep size-64 /proc/slabinfo
size-64(DMA)      0  0  64 240  1 : tunables 120 60 8 : slabdata  0  0  0
size-64          48263 59760  64 240  1 : tunables 120 60 8 : slabdata 249 249 12
# migrate_test mmaps 512Mfile and call system call move_pages(). and sleep.
[root@drpq kamezawa]# ./migrate_test 512Mfile 1 &
[1] 4108
#At the end of migration,
[root@drpq kamezawa]# cat /opt/mem_control/group_?/memory.usage_in_bytes
539738112
537706496

#Wow, charge is twice ;)
[root@drpq kamezawa]# grep size-64 /proc/slabinfo
size-64(DMA)      0  0  64 240  1 : tunables 120 60 8 : slabdata  0  0  0
size-64          81180 92400  64 240  1 : tunables 120 60 8 : slabdata 385 385 12

#Kill migrate_test, because 512Mfile is unmapped, charge in group_B is dropped.
[root@drpq kamezawa]# kill %1
[root@drpq kamezawa]# cat /opt/mem_control/group_?/memory.usage_in_bytes
536936448
1458176
[1]+  Terminated          ./migrate_test 512Mfile 1

#Try drop caches again
[root@drpq kamezawa]# echo 3 > /proc/sys/vm/drop_caches
[root@drpq kamezawa]# cat /opt/mem_control/group_?/memory.usage_in_bytes
536920064
1097728
#no change because charge in group_A is leaked.....

[root@drpq kamezawa]# grep size-64 /proc/slabinfo
size-64(DMA)      0  0  64 240  1 : tunables 120 60 8 : slabdata  0  0  0
size-64          48137 60720  64 240  1 : tunables 120 60 8 : slabdata 253 253 210
[root@drpq kamezawa]#

```

==

-Kame

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