

---

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

Posted by [KAMEZAWA Hiroyuki](#) on Wed, 03 Oct 2007 00:39:32 GMT

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

---

On Tue, 02 Oct 2007 19:04:15 +0530

Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> KAMEZAWA Hiroyuki wrote:

> > 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.)

> >

>

> Interesting test scenario, I'll try and reproduce the problem here.

> Why does discard OLD page not reclaim page\_cgroup?

Just because OLD page is not page-cache at discarding. (it is replaced with NEW page)

>

> > [root@drpq kamezawa]# ./migrate\_test 512Mfile 1 &

> > [1] 4108

> > #At the end of migration,

>

> Where can I find migrate\_test?

>

here, (just I wrote for this test. works on my RHEL5/2.6.18-rc8-mm2/ia64 NUMA box)

This program doesn't check 'where is file cache ?' before migration. So please check it by yourself before run.

==

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <syscall.h>
```

```
#include <sys/mman.h>
```

```

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <numaif.h>
#include <unistd.h>

#define PAGE_SIZE (16384)

static inline int move_pages(pid_t pid, unsigned long nr_pages,
    const void **address,
    const int *nodes, int *status, int flags)
{
    return syscall(SYS_move_pages, pid, nr_pages, address,
        nodes, status, flags);
}
/*
 * migrate_test.c -- mmap file and migrate it to specified node.
 * %migrate_task file nodeid
 */

int main(int argc, char *argv[])
{
    int ret, fd, val, node[1], status[1];
    char *addr, *c;
    unsigned long size, nr_pages, pos;
    struct stat statbuf;
    void *address[1];
    int target;

    if (argc != 3) {
        fprintf(stderr, "usage: migrate_test file node\n");
        exit(0);
    }
    target = atoi(argv[2]);

    fd = open(argv[1], O_RDONLY);
    if (fd < 0) {
        perror("open");
        exit(1);
    }

    ret = fstat(fd, &statbuf);
    if (ret < 0) {
        perror("fstat");
        exit(1);
    }

```

```

size = statbuf.st_size;
nr_pages = size/PAGESIZE;
size = nr_pages * PAGESIZE;

addr = mmap(NULL, size, PROT_READ, MAP_SHARED, fd, 0);

if (addr == MAP_FAILED) {
    perror("mmap");
    exit(1);
}
/* Touch all */
for (pos = 0; pos < nr_pages; pos++) {
    c = addr + pos * PAGESIZE;
    val += *c;
}
/* Move Pages */
for (pos = 0; pos < nr_pages; pos++) {
    node[0] = target;
    status[0] = 0;
    address[0] = addr + pos * PAGESIZE;
    ret = move_pages(0, 1, address, node, status,
        MPOL_MF_MOVE_ALL);
    if (ret) {
        perror("move_pages");
    }
}
#if 1
    printf("pos %d %p %d %d\n",pos, address[0], node[0], status[0]);
#endif
}
/* Touch all again....maybe unnecessary.*/
for (pos = 0; pos < nr_pages; pos++) {
    c = addr + pos * PAGESIZE;
    val += *c;
}
while (1) {
    /* mmap until killed */
    pause();
}
printf("val %d\n",val);
return 0;
}

```

---

Containers mailing list  
Containers@lists.linux-foundation.org

