Subject: Re: [PATCH 8/8] Per-container pages reclamation
Posted by xemul on Mon, 21 May 2007 15:15:32 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Pavel Emelianov wrote:
>> Implement try_to_free_pages_in_container() to free the
>> pages in container that has run out of memory.
>>
>> The scan_control->isolate_pages() function isolates the
>> container pages only.

Sorry for the late answer, but I have just managed to get
to the patches. One comment is below.

>>
>
> Hi, Pavel/Andrew,
>
> I've started running some basic tests like lmbench and LTP vm stress
> on the RSS controller.
>
> With the controller rss_limit set to 256 MB, I saw the following panic
> on a machine
>
> Unable to handle kernel NULL pointer dereference at 000000000000001c RIP:
> [<ffffffff80328581>] _raw_spin_lock+0xd/0xf6
> PGD 3c841067 PUD 5d5d067 PMD 0
> Oops: 0000 [1] SMP
> CPU 2
> Modules linked in: ipv6 hidp rfcomm l2cap bluetooth sunrpc video button battery asus_acpi
backlight ac lp parport_pc parport nvram pcspkr amd_rng rng_core i2c_amd756 i2c_core
> Pid: 13581, comm: mtest01 Not tainted 2.6.20-autokern1 #1
> RIP: 0010:[<ffffffff80328581>] [<ffffffff80328581>] _raw_spin_lock+0xd/0xf6
> RSP: 0000:ffff81003e6c9ce8 EFLAGS: 00010096
> RAX: ffffffff8087f720 RBX: 0000000000000018 RCX: ffff81003f36f9d0
> RDX: ffff8100807bb040 RSI: 0000000000000001 RDI: 0000000000000018
> RBP: 0000000000000000 R08: ffff81003e6c8000 R09: 0000000000000002
> R10: ffff810001021da8 R11: ffffffff8044658f R12: ffff81000c861e01
> R13: 0000000000000018 R14: ffff81000c861eb8 R15: ffff810032d34138
> FS: 00002abf7a1961e0(0000) GS:ffff81003edb94c0(0000) knlGS:0000000000000000
> CS: 0010 DS: 0000 ES: 0000 CR0: 000000008005003b
> CR2: 000000000000001c CR3: 000000002ba6e000 CR4: 00000000000006e0
> Process mtest01 (pid: 13581, threadinfo ffff81003e6c8000, task ffff81003d8ec040)
> Stack: ffff810001003638 ffff810014a8c2c0 0000000000000000 ffff81000c861e01
> 0000000000000018 ffffffff80287166 ffff81000c861eb8 ffff81000000bac0
> ffff81003f36f9a0 ffff81000c861e40 ffff81001d4b6a20 ffffffff8026a92e
> Call Trace:

> [<ffffffff80287166>] container_rss_move_lists+0x3b/0xaf
> [<ffffffff8026a92e>] activate_page+0xc1/0xd0
> [<ffffffff80245f15>] wake_bit_function+0x0/0x23
> [<ffffffff8026ab34>] mark_page_accessed+0x1b/0x2f
> [<ffffffff80265d25>] filemap_nopage+0x180/0x338
> [<ffffffff80270474>] __handle_mm_fault+0x1f2/0xa81
> [<ffffffff804c58ef>] do_page_fault+0x42b/0x7b3
> [<ffffffff802484c4>] hrtimer_cancel+0xc/0x16
> [<ffffffff804c2a89>] do_nanosleep+0x47/0x70
> [<ffffffff802485f4>] hrtimer_nanosleep+0x58/0x119
> [<ffffffff8023bc1f>] sys_sysinfo+0x15b/0x173
> [<ffffffff804c3d3d>] error_exit+0x0/0x84
>
> On analyzing the code, I found that the page is mapped (we have a page_mapped() check in
> container_rss_move_lists()), but the page_container is invalid. Please review the fix
> attached (we reset the page's container pointer to NULL when a page is completely unmapped)
>
>
>
> ---------- ----------
>
> Index: linux-2.6.20/mm/rss_container.c
> ========== =======
> --- linux-2.6.20.orig/mm/rss_container.c 2007-05-15 05:13:46.000000000 -0700
> +++ linux-2.6.20/mm/rss_container.c 2007-05-16 20:45:45.000000000 -0700
> @@ -212,6 +212,7 @@ void container_rss_del(struct page_conta
>
>   css_put(&rss->css);
>   kfree(pc);
> + init_page_container(page);

This hunk is bad.
See, when the page drops its mapcount to 0 it may be reused right
after this if it belongs to a file map - another CPU can touch it.
Thus you're risking to reset the wrong container.

The main idea if the accounting is that you cannot trust the
page_container(page) value after the page's mapcount became 0.

> }
>
> static void rss_move_task(struct container_subsys *ss,
> Index: linux-2.6.20/mm/page_alloc.c
> ========== =======
> --- linux-2.6.20.orig/mm/page_alloc.c 2007-05-16 10:30:10.000000000 -0700
> +++ linux-2.6.20/mm/page_alloc.c 2007-05-16 20:45:24.000000000 -0700
> @@ -41,6 +41,7 @@
> #include <linux/pfn.h>

> #include <linux/backing-dev.h>
> #include <linux/fault-inject.h>
> +#include <linux/rss_container.h>
>
> #include <asm/tlbflush.h>
> #include <asm/div64.h>
> @@ -1977,6 +1978,7 @@ void __meminit memmap_init_zone(unsigned
>   set_page_links(page, zone, nid, pfn);
>   init_page_count(page);
>   reset_page_mapcount(page);
> + init_page_container(page);
>   SetPageReserved(page);
>   INIT_LIST_HEAD(&page->lru);
> #ifdef WANT_PAGE_VIRTUAL
> Index: linux-2.6.20/include/linux/rss_container.h
> ===
> --- linux-2.6.20.orig/include/linux/rss_container.h 2007-05-16 10:31:04.000000000 -0700
> +++ linux-2.6.20/include/linux/rss_container.h 2007-05-16 10:32:14.000000000 -0700
> @@ -28,6 +28,11 @@ void container_rss_move_lists(struct pag
> unsigned long isolate_pages_in_container(unsigned long nr_to_scan,
>   struct list_head *dst, unsigned long *scanned,
>   struct zone *zone, struct rss_container *, int active);
> +static inline void init_page_container(struct page *page)
> +{
> + page_container(page) = NULL;
> +}
> +
> #else
> static inline int container_rss_prepare(struct page *pg,
>   struct vm_area_struct *vma, struct page_container **pc)
> @@ -56,6 +61,10 @@ static inline void mm_free_container(str
> {
> }
>
> +static inline void init_page_container(struct page *page)
> +{
> +}
> +
> #define isolate_container_pages(nr, dst, scanned, rss, act, zone) ({ BUG(); 0;})
> #define container_rss_move_lists(pg, active) do { } while (0)
> #endif