Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list
Posted by akpm on Thu, 17 Apr 2008 04:11:44 GMT
View Forum Message <> Reply to Message

On Thu, 17 Apr 2008 11:37:15 +0800 Li Zefan <lizf@cn.fujitsu.com> wrote:

> When I ran a test program to fork mass processes and at the same time
> 'cat /cgroup/tasks', I got the following oops:
>
> ------------[cut here]------------
> kernel BUG at lib/list_debug.c:72!
> invalid opcode: 0000 [#1] SMP
> Pid: 4178, comm: a.out Not tainted (2.6.25-rc9 #72)
> ...
> Call Trace:
> [<c044a5f9>] ? cgroup_exit+0x55/0x94
> [<c0427acf>] ? do_exit+0x217/0x5ba
> [<c0427ed7>] ? do_group_exit+0.65/0x7c
> [<c0427efd>] ? sys_exit_group+0xf/0x11
> [<c0404842>] ? syscall_call+0x7/0xb
> [<c05e0000>] ? init_cyrix+0x2fa/0x479
> ...
> EIP: [<c04df671>] list_del+0x35/0x53 SS:ESP 0068:ebc7df4
> ---[end trace caffb7332252612b]---
> Fixing recursive fault but reboot is needed!
>
> After digging into the code and debugging, I finlly found out a race
> situation:
>   do_exit()
>   ->cgroup_exit()
>   ->if (!list_empty(&tsk->cg_list))
>   list_del(&tsk->cg_list);
>
> cgroup_iter_start()
>   ->cgroup_enable_task_cg_list()
>   ->list_add(&tsk->cg_list, ..);
>
> In this case the list won't be deleted though the process has exited.

I don't fully understand the race. Both paths hold css_set_lock.

Can you describe it in more detail please?

> We got two bug reports in the past, which seem to be the same bug as
> this one:
> http://lkml.org/lkml/2008/3/5/332
> http://lkml.org/lkml/2007/10/17/224
>

> Actually sometimes I got oops on list_del, sometimes oops on list_add.
> And I can change my test program a bit to trigger other oops.
>
> The patch has been tested both on x86_32 and x86_64.
>
> Signed-off-by: Li Zefan <lizf@cn.fujitsu.com>
> ---
> kernel/cgroup.c | 7 ++++++-
> 1 files changed, 6 insertions(+), 1 deletions(-)
>
> diff --git a/kernel/cgroup.c b/kernel/cgroup.c
> index 2727f92..6d8de05 100644
> --- a/kernel/cgroup.c
> +++ b/kernel/cgroup.c
> @@ -1722,7 +1722,12 @@ void cgroup_enable_task_cg_lists(void)
>   use_task_css_set_links = 1;
>   do_each_thread(g, p) {
>   task_lock(p);
> - if (list_empty(&p->cg_list))
> + /*
> +   * We should check if the process is exiting, otherwise
> +   * it will race with cgroup_exit() in that the list
> +   * entry won't be deleted though the process has exited.
> +   */
> + if (!(p->flags & PF_EXITING) && list_empty(&p->cg_list))
>   list_add(&p->cg_list, &p->cgroups->tasks);
>   task_unlock(p);
>   } while_each_thread(g, p);

Don't think I understand the fix either :(

afacit the task at *p could set PF_EXITING immediately after this code has
tested PF_EXITING and then the task at *p could proceed until we hit the
same race (whatever that is).

Perhaps taking p->sighand->siglock would fix that up, but that's just a
guess at this stage.