Subject: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Li Zefan on Thu, 17 Apr 2008 03:37:15 GMT View Forum Message <> Reply to Message

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.

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(q, p);
```

```
-- 1.5.4.rc3
```

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

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.

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Paul Menage on Thu, 17 Apr 2008 04:17:34 GMT View Forum Message <> Reply to Message

On Wed, Apr 16, 2008 at 9:11 PM, Andrew Morton <akpm@linux-foundation.org> wrote:

- >
- > I don't fully understand the race. Both paths hold css_set_lock.
- >
- > Can you describe it in more detail please?

Task A starts exiting, passes the check for unlinking current->cg_list.

Before it completely exits task B does the very first cgroup_iter_begin() call (via reading a cgroups tasks file) which links all tasks in to their css_set objects via tsk->cg_list.

Then task A finishes exiting and is freed, but doesn't unlink from the cg_list.

>

> 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).

The important fact there is that the task sets PF_EXITING *before* it checks whether it needs to unlink from current->cg_list.

Paul

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Paul Menage on Thu, 17 Apr 2008 04:18:41 GMT View Forum Message <> Reply to Message

On Wed, Apr 16, 2008 at 8:37 PM, 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.
- >
- > 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

Yes, that looks like it could be the same one - great. But this corruption can only be triggered the first time you cat a tasks file after a reboot, right? That would partly explain why it was hard to reproduce (at least, I had trouble).

My only thought about the downside of this is that an exiting task that gets stuck somewhere between setting PF_EXITING and calling cgroup_exit() won't show up in its cgroup's tasks file, since we'll enable cgroup links but skip it. I guess that's not a big deal.

Maybe it would be better to not do a cgroup_exit() until we're unhashed, so that cgroup_enable_task_cg_list() can't find the exiting task?

Paul

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Paul Menage on Thu, 17 Apr 2008 04:28:33 GMT View Forum Message <> Reply to Message

On Wed, Apr 16, 2008 at 9:18 PM, Paul Menage <menage@google.com> wrote: >

- > My only thought about the downside of this is that an exiting task
- > that gets stuck somewhere between setting PF_EXITING and calling
- > cgroup_exit() won't show up in its cgroup's tasks file, since we'll
- > enable cgroup links but skip it. I guess that's not a big deal.

>

How about this as an alternative approach? We can take advantage of the indirection in tsk->cgroups to create an additional distinguished css_set that indicates the task has passed the point of checking tsk->cg_list:

- create a new css_set, called exit_css_set; it has the same cgroup pointer set as init_css_set.

 in cgroup_exit(), set current->cgroups to &exit_css_set rather than &init_css_set

- in cgroup_enable_task_cg_list(), ignore any task where p->cgroups == &exit_css_set

That way we're synchronizing directly with the task_lock()-protected section in cgroup_exit(), rather than with the setting of PF_EXITING at the beginning of do_exit().

Paul

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by akpm on Thu, 17 Apr 2008 04:59:07 GMT View Forum Message <> Reply to Message

On Wed, 16 Apr 2008 21:17:34 -0700 "Paul Menage" <menage@google.com> wrote:

> On Wed, Apr 16, 2008 at 9:11 PM, Andrew Morton

> <akpm@linux-foundation.org> wrote:

> >

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

> >

>> Can you describe it in more detail please?

>

> Task A starts exiting, passes the check for unlinking current->cg_list.

So cgroup_exit() sees !list_empty(tsk->cg_list)

And the list_del() sets tsk->cg_list to LIST_POISON[12], which still means !list_empty(). Or we remove that debugging code and avoid writing to tsk->cg_list, and it _still_ is !list_empty().

> Before it completely exits task B does the very first

> cgroup_iter_begin() call (via reading a cgroups tasks file) which

> links all tasks in to their css_set objects via tsk->cg_list.

But it won't link this task, because it's !list_empty().

> Then task A finishes exiting and is freed, but doesn't unlink from the cg_list.

> > > >> 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).

>

- > The important fact there is that the task sets PF_EXITING *before* it
- > checks whether it needs to unlink from current->cg_list.
- >

> Paul

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Li Zefan on Thu, 17 Apr 2008 05:04:47 GMT View Forum Message <> Reply to Message

Paul Menage wrote:

> On Wed Apr 16, 2008 at 8:37 PM Li Zefan <lizf@cn fuiitsu.com=""> wrote:</lizf@cn>
>> 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</c044a5f9>
>> [<c0427acf>] ? do exit+0x217/0x5ba</c0427acf>
>> [<c0427ed7>] ? do_group_exit+0.65/0x7c</c0427ed7>
>> [<c0427efd>] ? sys_exit_group+0xf/0x11</c0427efd>
>> [<c0404842>] ? syscall_call+0x7/0xb</c0404842>
>> [<c05e0000>] ? init_cyrix+0x2fa/0x479</c05e0000>
>>
>> EIP: [<c04df671>] list_del+0x35/0x53 SS:ESP 0068:ebc7df4</c04df671>
>>[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. >> >> 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 >> > > Yes, that looks like it could be the same one - great. But this > corruption can only be triggered the first time you cat a tasks file > after a reboot, right? That would partly explain why it was hard to > reproduce (at least, I had trouble). >

Right. I was lucky to trigger this and thus knew how to reproduce.

> My only thought about the downside of this is that an exiting task
 > that gets stuck somewhere between setting PF_EXITING and calling
 > cgroup_exit() won't show up in its cgroup's tasks file, since we'll
 > enable cgroup links but skip it. I guess that's not a big deal.

Agree. I think it won't be a problem.

> Maybe it would be better to not do a cgroup_exit() until we're > unhashed, so that cgroup_enable_task_cg_list() can't find the exiting > task?

> > Paul

> 1 a

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Li Zefan on Thu, 17 Apr 2008 05:10:33 GMT View Forum Message <> Reply to Message

Andrew Morton wrote:

> On Wed, 16 Apr 2008 21:17:34 -0700 "Paul Menage" <menage@google.com> wrote:

>> On Wed, Apr 16, 2008 at 9:11 PM, Andrew Morton >> <akpm@linux-foundation.org> wrote: >>> I don't fully understand the race. Both paths hold css_set_lock. >>> >>> Can you describe it in more detail please? >> Task A starts exiting, passes the check for unlinking current->cg list. > > So cgroup_exit() sees !list_empty(tsk->cg_list) > cgroup exit() sees list empty(tsk->cg list), then cgroup enable task cg list() links the task to the list, and then the task exited, so the list entry won't get deleted. > And the list_del() sets tsk->cg_list to LIST_POISON[12], which still means > !list_empty(). Or we remove that debugging code and avoid writing to > tsk->cg_list, and it _still_ is !list_empty(). > >> Before it completely exits task B does the very first >> cgroup_iter_begin() call (via reading a cgroups tasks file) which >> links all tasks in to their css set objects via tsk->cg list. > > But it won't link this task, because it's !list empty(). > >> Then task A finishes exiting and is freed, but doesn't unlink from the cg_list. >> >>> 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). >> The important fact there is that the task sets PF EXITING *before* it >> checks whether it needs to unlink from current->cg list. >> >> Paul > >

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by akpm on Thu, 17 Apr 2008 05:16:02 GMT View Forum Message <> Reply to Message

On Thu, 17 Apr 2008 13:10:33 +0800 Li Zefan <lizf@cn.fujitsu.com> wrote:

> Andrew Morton wrote:

> > On Wed, 16 Apr 2008 21:17:34 -0700 "Paul Menage" <menage@google.com> wrote:

>> >> On Wed, Apr 16, 2008 at 9:11 PM, Andrew Morton
>>> <akpm@linux-foundation.org> wrote:
>>>> I don't fully understand the race. Both paths hold css_set_lock.
>>>>
>>> Can you describe it in more detail please?
>>> Task A starts exiting, passes the check for unlinking current->cg_list.
>>
>> So cgroup_exit() sees !list_empty(tsk->cg_list)
>>
>>
>> cgroup_exit() sees list_empty(tsk->cg_list), then cgroup_enable_task_cg_list()
> links the task to the list, and then the task exited, so the list entry won't
> get deleted.
OK.

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by akpm on Thu, 17 Apr 2008 05:16:55 GMT View Forum Message <> Reply to Message

On Thu, 17 Apr 2008 13:04:47 +0800 Li Zefan <lizf@cn.fujitsu.com> wrote:

>>

> > Yes, that looks like it could be the same one - great. But this > > corruption can only be triggered the first time you cat a tasks file > > after a reboot, right? That would partly explain why it was hard to > > reproduce (at least, I had trouble). > > > > Right. I was lucky to trigger this and thus knew how to reproduce. > >> My only thought about the downside of this is that an exiting task > > that gets stuck somewhere between setting PF_EXITING and calling > > cgroup_exit() won't show up in its cgroup's tasks file, since we'll > > enable cgroup links but skip it. I guess that's not a big deal. > > > > Agree. I think it won't be a problem. > > > Maybe it would be better to not do a cgroup_exit() until we're > > unhashed, so that cgroup_enable_task_cg_list() can't find the exiting > > task?

So we won't be doing what Paul suggested?

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Paul Menage on Thu, 17 Apr 2008 05:20:44 GMT View Forum Message <> Reply to Message

On Wed, Apr 16, 2008 at 9:59 PM, Andrew Morton <akpm@linux-foundation.org> wrote:

> >

> > Task A starts exiting, passes the check for unlinking current->cg_list.

- >
- > So cgroup_exit() sees !list_empty(tsk->cg_list)

We don't actually set up the links running through tsk->cg_list to the css_set objects until the first time someone calls cgroup_iter_begin() - so anyone who never actually uses cgroups doesn't pay the list management overhead. So in this case, the list is empty.

Paul

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Paul Menage on Thu, 17 Apr 2008 05:24:09 GMT View Forum Message <> Reply to Message

On Wed, Apr 16, 2008 at 10:16 PM, Andrew Morton <akpm@linux-foundation.org> wrote:

- > >> Maybe it would be better to not do a cgroup_exit() until we're
- > >> unhashed, so that cgroup_enable_task_cg_list() can't find the exiting
- > > > task?
- >
- > So we won't be doing what Paul suggested?

>

It's not as high a priority as Li's bug fix (which may be a good candidate for 2.6.25.1) but for the future I think I'll implement this distinguished css_set pointer for tasks that have finished

cgroup_exit(), since I think it will make the similar synchronization in attach_task() cleaner, as well as cgroup_enable_task_cg_list().

Paul

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

Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list Posted by Li Zefan on Thu, 17 Apr 2008 05:27:07 GMT View Forum Message <> Reply to Message

Paul Menage wrote:

> On Wed, Apr 16, 2008 at 10:16 PM, Andrew Morton

> <akpm@linux-foundation.org> wrote:

>> >> Maybe it would be better to not do a cgroup_exit() until we're

```
>> >> unhashed, so that cgroup_enable_task_cg_list() can't find the exiting
```

>> > > task?

>>

```
>> So we won't be doing what Paul suggested?
```

>> >

> It's not as high a priority as Li's bug fix (which may be a good

> candidate for 2.6.25.1) but for the future I think I'll implement this

> distinguished css_set pointer for tasks that have finished

> cgroup_exit(), since I think it will make the similar synchronization

> in attach_task() cleaner, as well as cgroup_enable_task_cg_list().

>

Yes, this approach sounds good to me. :)

> Paul

>

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