Subject: [PATCH] Check for error returned by kthread_create on creating journal thread
Posted by xemul on Mon, 16 Apr 2007 07:40:28 GMT
View Forum Message <> Reply to Message

If the thread failed to create the subsequent wait_event
will hang forever.

This is likely to happen if kernel hits max_threads limit.

Will be critical for virtualization systems that limit the
number of tasks and kernel memory usage within the container.

```
--- ./fs/jbd/journal.c.jbdthreads 2007-04-16 11:17:36.000000000 +0400
+++ ./fs/jbd/journal.c 2007-04-16 11:30:09.000000000 +0400
@@ -211,10 +211,16 @@ end_loop:
  return 0;
 }

-static void journal_start_thread(journal_t *journal)
+static int journal_start_thread(journal_t *journal)
 {
- kthread_run(kjournald, journal, "kjournald");
+ struct task_struct *t;
+
+ t = kthread_run(kjournald, journal, "kjournald");
+ if (IS_ERR(t))
+  return PTR_ERR(t);
+
  wait_event(journal->j_wait_done_commit, journal->j_task != 0);
+ return 0;
 }

 static void journal_kill_thread(journal_t *journal)
@@ -840,8 +846,7 @@ static int journal_reset(journal_t *jour

  /* Add the dynamic fields and write it to disk. */
  journal_update_superblock(journal, 1);
- journal_start_thread(journal);
- return 0;
+ return journal_start_thread(journal);
 }

 /**
--- ./fs/jbd2/journal.c.jbdthreads 2007-04-16 11:17:36.000000000 +0400
+++ ./fs/jbd2/journal.c 2007-04-16 11:27:45.000000000 +0400
@@ -211,10 +211,16 @@ end_loop:
  return 0;
```

```
 }

-static void jbd2_journal_start_thread(journal_t *journal)
+static int jbd2_journal_start_thread(journal_t *journal)
 {
- kthread_run(kjournald2, journal, "kjournald2");
+ struct task_struct *t;
+
+ t = kthread_run(kjournald2, journal, "kjournald2");
+ if (IS_ERR(t))
+  return PTR_ERR(t);
+
  wait_event(journal->j_wait_done_commit, journal->j_task != 0);
+ return 0;
 }

 static void journal_kill_thread(journal_t *journal)
@@ -840,8 +846,7 @@ static int journal_reset(journal_t *jour

  /* Add the dynamic fields and write it to disk. */
  jbd2_journal_update_superblock(journal, 1);
- jbd2_journal_start_thread(journal);
- return 0;
+ return jbd2_journal_start_thread(journal);
 }

 /**
```

Subject: Re: [PATCH] Check for error returned by kthread_create on creating journal thread

Posted by Christoph Hellwig on Mon, 16 Apr 2007 10:21:07 GMT

View Forum Message <> Reply to Message

On Mon, Apr 16, 2007 at 11:41:14AM +0400, Pavel Emelianov wrote:
> If the thread failed to create the subsequent wait_event
> will hang forever.
>
> This is likely to happen if kernel hits max_threads limit.
>
> Will be critical for virtualization systems that limit the
> number of tasks and kernel memory usage within the container.

```
> --- ./fs/jbd/journal.c.jbdthreads 2007-04-16 11:17:36.000000000 +0400
> +++ ./fs/jbd/journal.c 2007-04-16 11:30:09.000000000 +0400
> @@ -211,10 +211,16 @@ end_loop:
>   return 0;
>  }
```

>
> -static void journal_start_thread(journal_t *journal)
> +static int journal_start_thread(journal_t *journal)
> {
> - kthread_run(kjournald, journal, "kjournald");
> + struct task_struct *t;
> +
> + t = kthread_run(kjournald, journal, "kjournald");
> + if (IS_ERR(t))
> + return PTR_ERR(t);
> +
>   wait_event(journal->j_wait_done_commit, journal->j_task != 0);

Note that this wait_event should exist at all, and the return
value of kthread_run should be assigned to journal->j_task. Also
the code doesn't use the kthread primitives in other places leading
to crufty code.

Subject: Re: [PATCH] Check for error returned by kthread_create on creating journal thread
Posted by xemul on Mon, 16 Apr 2007 11:07:47 GMT
View Forum Message <> Reply to Message

Christoph Hellwig wrote:
> On Mon, Apr 16, 2007 at 11:41:14AM +0400, Pavel Emelianov wrote:
>> If the thread failed to create the subsequent wait_event
>> will hang forever.
>>
>> This is likely to happen if kernel hits max_threads limit.
>>
>> Will be critical for virtualization systems that limit the
>> number of tasks and kernel memory usage within the container.
>
>> --- ./fs/jbd/journal.c.jbdthreads 2007-04-16 11:17:36.000000000 +0400
>> +++ ./fs/jbd/journal.c 2007-04-16 11:30:09.000000000 +0400
>> @@ -211,10 +211,16 @@ end_loop:
>>   return 0;
>> }
>>
>> -static void journal_start_thread(journal_t *journal)
>> +static int journal_start_thread(journal_t *journal)
>> {
>> - kthread_run(kjournald, journal, "kjournald");
>> + struct task_struct *t;
>> +
>> + t = kthread_run(kjournald, journal, "kjournald");
>> + if (IS_ERR(t))

>> + return PTR_ERR(t);
>> +
>>   wait_event(journal->j_wait_done_commit, journal->j_task != 0);
>
> Note that this wait_event should exist at all, and the return

Should NOT you mean?

> value of kthread_run should be assigned to journal->j_task. Also
> the code doesn't use the kthread primitives in other places leading
> to crufty code.

Well, this could be done with a separate patch, I think.

> -
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
> Please read the FAQ at http://www.tux.org/lkml/
>

Subject: Re: [PATCH] Check for error returned by kthread_create on creating journal thread
Posted by Christoph Hellwig on Mon, 16 Apr 2007 11:10:26 GMT
View Forum Message <> Reply to Message

On Mon, Apr 16, 2007 at 03:10:42PM +0400, Pavel Emelianov wrote:
> Christoph Hellwig wrote:
> > On Mon, Apr 16, 2007 at 11:41:14AM +0400, Pavel Emelianov wrote:
> >> If the thread failed to create the subsequent wait_event
> >> will hang forever.
> >>
> >> This is likely to happen if kernel hits max_threads limit.
> >>
> >> Will be critical for virtualization systems that limit the
> >> number of tasks and kernel memory usage within the container.
> >
> >> --- ./fs/jbd/journal.c.jbdthreads 2007-04-16 11:17:36.000000000 +0400
> >> +++ ./fs/jbd/journal.c 2007-04-16 11:30:09.000000000 +0400
> >> @@ -211,10 +211,16 @@ end_loop:
> >>   return 0;
> >> }
> >>
> >> -static void journal_start_thread(journal_t *journal)
> >> +static int journal_start_thread(journal_t *journal)
> >> {
> >> - kthread_run(kjournald, journal, "kjournald");

> >> + struct task_struct *t;
> >> +
> >> + t = kthread_run(kjournald, journal, "kjournald");
> >> + if (IS_ERR(t))
> >> + return PTR_ERR(t);
> >> +
> >>   wait_event(journal->j_wait_done_commit, journal->j_task != 0);
> >
> > Note that this wait_event should exist at all, and the return
>
> Should NOT you mean?

Umm, yes - of course :)

> > value of kthread_run should be assigned to journal->j_task. Also
> > the code doesn't use the kthread primitives in other places leading
> > to crufty code.
>
> Well, this could be done with a separate patch, I think.

Subject: Re: [PATCH] Check for error returned by kthread_create on creating journal thread
Posted by Andrew Morton on Fri, 20 Apr 2007 22:38:09 GMT
View Forum Message <> Reply to Message

On Mon, 16 Apr 2007 11:41:14 +0400
Pavel Emelianov <xemul@sw.ru> wrote:

> If the thread failed to create the subsequent wait_event
> will hang forever.
>
> This is likely to happen if kernel hits max_threads limit.
>
> Will be critical for virtualization systems that limit the
> number of tasks and kernel memory usage within the container.
>
>
> [diff-jbd-check-start-journal-thread-return-value text/plain (1.7KB)]
> --- ./fs/jbd/journal.c.jbdthreads 2007-04-16 11:17:36.000000000 +0400
> +++ ./fs/jbd/journal.c 2007-04-16 11:30:09.000000000 +0400
> @@ -211,10 +211,16 @@ end_loop:
>   return 0;
> }
>
> -static void journal_start_thread(journal_t *journal)
> +static int journal_start_thread(journal_t *journal)
> {

> - kthread_run(kjournald, journal, "kjournald");
> + struct task_struct *t;
> +
> + t = kthread_run(kjournald, journal, "kjournald");
> + if (IS_ERR(t))
> + return PTR_ERR(t);
> +
>   wait_event(journal->j_wait_done_commit, journal->j_task != 0);
> + return 0;
> }

Thanks. Please don't forget those Signed-off-by:s

I assume that you runtime tested this and that the mount failed in
an appropriate fashion?

Subject: Re: [PATCH] Check for error returned by kthread_create on creating journal thread
Posted by xemul on Mon, 23 Apr 2007 06:13:45 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Mon, 16 Apr 2007 11:41:14 +0400
> Pavel Emelianov <xemul@sw.ru> wrote:
>
>> If the thread failed to create the subsequent wait_event
>> will hang forever.
>>
>> This is likely to happen if kernel hits max_threads limit.
>>
>> Will be critical for virtualization systems that limit the
>> number of tasks and kernel memory usage within the container.
>>
>>
>> [diff-jbd-check-start-journal-thread-return-value text/plain (1.7KB)]
>> --- ./fs/jbd/journal.c.jbdthreads 2007-04-16 11:17:36.000000000 +0400
>> +++ ./fs/jbd/journal.c 2007-04-16 11:30:09.000000000 +0400
>> @@ -211,10 +211,16 @@ end_loop:
>>   return 0;
>> }
>>
>> -static void journal_start_thread(journal_t *journal)
>> +static int journal_start_thread(journal_t *journal)
>> {
>> - kthread_run(kjournald, journal, "kjournald");
>> + struct task_struct *t;
>> +

>> + t = kthread_run(kjournald, journal, "kjournald");
>> + if (IS_ERR(t))
>> + return PTR_ERR(t);
>> +
>>   wait_event(journal->j_wait_done_commit, journal->j_task != 0);
>> + return 0;
>> }
>
> Thanks.   Please don't forget those Signed-off-by:s

I will :)

> I assume that you runtime tested this and that the mount failed in
> an appropriate fashion?
>

Yes. This was easy to test as we can force kthread_run() to fail
at the moment we want with the help of the beancounters ;)

Moreover this patch lives in our tree for a year or so in both
stable and development branches.
