Subject: Kernel text size with pid namespace
Posted by Sukadev Bhattiprolu on Thu, 20 Sep 2007 00:16:44 GMT
View Forum Message <> Reply to Message

Matt,

The pid-namespace patcheset (http://lkml.org/lkml/2007/8/10/118)
was added to the -mm tree in 2.6.23-rc3-mm1.

With CONFIG_CC_OPTIMIZE_FOR_SIZE=y this patchset increases the kernel
text size by about 5K (closer to 6K when the config token is set to N).

As a quick test, I uninlined several helper functions and with this
the text size increased by about 4K. But since most of these inline
functions are used in process creation/termination, we would need to
keep them inline, when optimizing for performance.

We also do not have a config token to select pid namespace (its always
enabled).

Is there a cause for concern with the 5K to 6K increase in text size ?
If so, can/should we conditionally inline some functions ? Or move
some pid namespace creation code under CONFIG_TINY or something ?
Are there other techniques besides uninling we could apply ?

For reference, I am including below, some numbers for 2.6.23-rc2-mm2
kernel for an x86_64 config file. In the following filenames:

```
 "clean"  no pid ns patches
 "opt-size" CONFIG_CC_OPTIMIZE_FOR_SIZE=y
 "no-opt" CONFIG_CC_OPTIMIZE_FOR_SIZE=n
 "uninline" uninline several new inline functions.
```

```
$ size vmlinux*

   text    data     bss     dec     hex filename

6016101  906266  772424 7694791  7569c7 vmlinux-clean-no-opt-size
6021869  906330  772424 7700623  75808f vmlinux-pidns-no-opt-size
6020805  906330  772424 7699559  757c67 vmlinux-pidns-no-opt-uninline-task-pid

5299192  906330  772424 6977946  6a799a vmlinux-clean-opt-size
5304588  906394  772424 6983406  6a8eee vmlinux-pidns-opt-size
5303348  906394  772424 6982166  6a8a16 vmlinux-pidns-opt-size-uninline-task-pid
```

Thanks,

Suka

Subject: Re: Kernel text size with pid namespace
Posted by Matt Mackall on Thu, 20 Sep 2007 03:39:45 GMT
View Forum Message <> Reply to Message

On Wed, Sep 19, 2007 at 05:16:44PM -0700, sukadev@us.ibm.com wrote:
> Matt,
>
> The pid-namespace patcheset (http://lkml.org/lkml/2007/8/10/118)
> was added to the -mm tree in 2.6.23-rc3-mm1.
>
> With CONFIG_CC_OPTIMIZE_FOR_SIZE=y this patchset increases the kernel
> text size by about 5K (closer to 6K when the config token is set to N).

That's not too bad.

> As a quick test, I uninlined several helper functions and with this
> the text size increased by about 4K. But since most of these inline
> functions are used in process creation/termination, we would need to
> keep them inline, when optimizing for performance.

You are aware that functions as critical as spinlocks are now
completely out of line, right? Given that a cache miss is
significantly more expensive than a function call, fitting more in
cache by reducing inlining tends to be a substantial win.

Inline functions still tend to make performance sense when the actual
function body is more complex than setting up the call frame, of
course, but in those cases, uninlining will tend to increase code
size.

But I'd be very surprised if uninlining things showed up negatively
even on a microbenchmark like lmbench.

Also, quick question (I haven't really looked at this code in any detail):

```
 static inline pid_t pid_nr(struct pid *pid)
 {
 pid_t nr = 0;
 if (pid)
-    nr = pid->nr;
+       nr = pid->numbers[0].nr;
+       return nr;
+}
```

Is calling this with a null struct pid a sensible thing to do or is it
a bug? If the latter, it'd be preferable to just do:

```
 return pid->numbers[0].nr;
```

And if the former, could we arrange to avoid using null struct pids at
all? Perhaps by having a dummy zeropid?

```
> Is there a cause for concern with the 5K to 6K increase in text size ?
> If so, can/should we conditionally inline some functions ? Or move
> some pid namespace creation code under CONFIG_TINY or something ?
> Are there other techniques besides uninling we could apply ?
>
> For reference, I am including below, some numbers for 2.6.23-rc2-mm2
> kernel for an x86_64 config file.  In the following filenames:
>
>  "clean"  no pid ns patches
>  "opt-size" CONFIG_CC_OPTIMIZE_FOR_SIZE=y
>  "no-opt" CONFIG_CC_OPTIMIZE_FOR_SIZE=n
>  "uninline" uninline several new inline functions.
>
> $ size vmlinux*
>
>    text    data     bss     dec     hex filename
>
> 6016101  906266  772424 7694791  7569c7 vmlinux-clean-no-opt-size
> 6021869  906330  772424 7700623  75808f vmlinux-pidns-no-opt-size
> 6020805  906330  772424 7699559  757c67 vmlinux-pidns-no-opt-uninline-task-pid
>
> 5299192  906330  772424 6977946  6a799a vmlinux-clean-opt-size
> 5304588  906394  772424 6983406  6a8eee vmlinux-pidns-opt-size
> 5303348  906394  772424 6982166  6a8a16 vmlinux-pidns-opt-size-uninline-task-pid
```

You might try running scripts/bloat-o-meter against a pair of these.

--
Mathematics is the supreme nostalgia of our time.

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

Subject: Re: Kernel text size with pid namespace
Posted by Pavel Emelianov on Thu, 20 Sep 2007 09:13:22 GMT

sukadev@us.ibm.com wrote:
> Matt,
>
> The pid-namespace patcheset (http://lkml.org/lkml/2007/8/10/118)
> was added to the -mm tree in 2.6.23-rc3-mm1.
>
> With CONFIG_CC_OPTIMIZE_FOR_SIZE=y this patchset increases the kernel
> text size by about 5K (closer to 6K when the config token is set to N).

Well, this is a new functionality, so the vmlinux size has to increase.
And as Matt has noticed 5k is not that bad :)

> As a quick test, I uninlined several helper functions and with this
> the text size increased by about 4K. But since most of these inline
> functions are used in process creation/termination, we would need to
> keep them inline, when optimizing for performance.

I'd keep them inline for performance reasons.

> We also do not have a config token to select pid namespace (its always
> enabled).

This was one of Andrew's requirements. However, I assume that we can
have the cloning code under this option.

> Is there a cause for concern with the 5K to 6K increase in text size ?
> If so, can/should we conditionally inline some functions ? Or move
> some pid namespace creation code under CONFIG_TINY or something ?
> Are there other techniques besides uninling we could apply ?
>
> For reference, I am including below, some numbers for 2.6.23-rc2-mm2
> kernel for an x86_64 config file. In the following filenames:
>
> "clean" no pid ns patches
> "opt-size" CONFIG_CC_OPTIMIZE_FOR_SIZE=y
> "no-opt" CONFIG_CC_OPTIMIZE_FOR_SIZE=n
> "uninline" uninline several new inline functions.
>

```
> $ size vmlinux*
>
>    text    data     bss     dec     hex filename
>
> 6016101  906266  772424 7694791  7569c7 vmlinux-clean-no-opt-size
> 6021869  906330  772424 7700623  75808f vmlinux-pidns-no-opt-size
> 6020805  906330  772424 7699559  757c67 vmlinux-pidns-no-opt-uninline-task-pid
>
> 5299192  906330  772424 6977946  6a799a vmlinux-clean-opt-size
> 5304588  906394  772424 6983406  6a8eee vmlinux-pidns-opt-size
> 5303348  906394  772424 6982166  6a8a16 vmlinux-pidns-opt-size-uninline-task-pid
```
>
> Thanks,
>
> Suka
>

Subject: Re: Kernel text size with pid namespace
Posted by Paul Jackson on Thu, 20 Sep 2007 11:04:06 GMT
View Forum Message <> Reply to Message

> > functions are used in process creation/termination, we would need to
> > keep them inline, when optimizing for performance.
>
> I'd keep them inline for performance reasons.

As Matt Mackall explained more carefully in his reply, it's no longer
clear that inlining is best for performance in as many situations as
it was the past. Cache footprint size tends to dominate performance on
present day processors.

See also Matt's comments on the NULL struct pid check. Getting rid
of conditional jumps may be the more important performance issue here.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

Subject: Re: Kernel text size with pid namespace
Posted by Pavel Emelianov on Thu, 20 Sep 2007 11:55:01 GMT
View Forum Message <> Reply to Message

Paul Jackson wrote:
>>> functions are used in process creation/termination, we would need to

>>> keep them inline, when optimizing for performance.
>> I'd keep them inline for performance reasons.
>
> As Matt Mackall explained more carefully in his reply, it's no longer
> clear that inlining is best for performance in as many situations as
> it was the past. Cache footprint size tends to dominate performance on
> present day processors.

True, but AFAIR, when I developed the namespaces I tried to move
the task_pid_nr etc calls in kernel/pid.c and the performance on
unixbench spawn and nptlperf tests became worse. That's why I said
that I'd keep them inline.

> See also Matt's comments on the NULL struct pid check. Getting rid
> of conditional jumps may be the more important performance issue here.

That's a valid argument. I will look over it.

Thanks,
Pavel

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

---

Subject: Re: Kernel text size with pid namespace
Posted by Sukadev Bhattiprolu on Fri, 21 Sep 2007 05:03:59 GMT
View Forum Message <> Reply to Message

Matt Mackall [mpm@selenic.com] wrote:
| On Wed, Sep 19, 2007 at 05:16:44PM -0700, sukadev@us.ibm.com wrote:
| > Matt,
| >
| > The pid-namespace patcheset (http://lkml.org/lkml/2007/8/10/118)
| > was added to the -mm tree in 2.6.23-rc3-mm1.
| >
| > With CONFIG_CC_OPTIMIZE_FOR_SIZE=y this patchset increases the kernel
| > text size by about 5K (closer to 6K when the config token is set to N).
|
| That's not too bad.

Ok, thanks, I won't worry about for now :-)

Just curious, is there a magic number like 8K or 32K increase in size (of
unconditional code) that one should watch out for ?

|

| > As a quick test, I uninlined several helper functions and with this
| > the text size increased by about 4K. But since most of these inline
| > functions are used in process creation/termination, we would need to
| > keep them inline, when optimizing for performance.
|
| You are aware that functions as critical as spinlocks are now
| completely out of line, right? Given that a cache miss is
| significantly more expensive than a function call, fitting more in
| cache by reducing inlining tends to be a substantial win.

I am aware now :-)

|
| Inline functions still tend to make performance sense when the actual
| function body is more complex than setting up the call frame, of
| course, but in those cases, uninlining will tend to increase code
| size.
|
| But I'd be very surprised if uninlining things showed up negatively
| even on a microbenchmark like lmbench.
|
| Also, quick question (I haven't really looked at this code in any detail):
|

```
|  static inline pid_t pid_nr(struct pid *pid)
|  {
|  pid_t nr = 0;
|  if (pid)
| -    nr = pid->nr;
| +       nr = pid->numbers[0].nr;
| +       return nr;
| +}
```

|
| Is calling this with a null struct pid a sensible thing to do or is it
| a bug?

Its not a bug. It just depends on whether the process is exiting or not.

| If the latter, it'd be preferable to just do:
|
| return pid->numbers[0].nr;
|
| And if the former, could we arrange to avoid using null struct pids at
| all? Perhaps by having a dummy zeropid?

Yes that sounds like a good idea, but requires us to carefully all uses
of the struct pid. Will look into it.

|

```
| > Is there a cause for concern with the 5K to 6K increase in text size ?
| > If so, can/should we conditionally inline some functions ? Or move
| > some pid namespace creation code under CONFIG_TINY or something ?
| > Are there other techniques besides uninling we could apply ?
| >
| > For reference, I am including below, some numbers for 2.6.23-rc2-mm2
| > kernel for an x86_64 config file.  In the following filenames:
| >
| >  "clean"  no pid ns patches
| >  "opt-size" CONFIG_CC_OPTIMIZE_FOR_SIZE=y
| >  "no-opt" CONFIG_CC_OPTIMIZE_FOR_SIZE=n
| >  "uninline" uninline several new inline functions.
| >
| > $ size vmlinux*
| >
| >    text    data     bss     dec     hex filename
| >
| > 6016101  906266  772424 7694791  7569c7 vmlinux-clean-no-opt-size
| > 6021869  906330  772424 7700623  75808f vmlinux-pidns-no-opt-size
| > 6020805  906330  772424 7699559  757c67 vmlinux-pidns-no-opt-uninline-task-pid
| >
| > 5299192  906330  772424 6977946  6a799a vmlinux-clean-opt-size
| > 5304588  906394  772424 6983406  6a8eee vmlinux-pidns-opt-size
| > 5303348  906394  772424 6982166  6a8a16 vmlinux-pidns-opt-size-uninline-task-pid
|
| You might try running scripts/bloat-o-meter against a pair of these.
|
| --
| Mathematics is the supreme nostalgia of our time.
```

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

Subject: Re: Kernel text size with pid namespace
Posted by Carl-Daniel Hailfinge on Fri, 21 Sep 2007 11:31:54 GMT

View Forum Message <> Reply to Message

On 20.09.2007 11:13, Pavel Emelyanov wrote:
> sukadev@us.ibm.com wrote:
>>
>> The pid-namespace patcheset (http://lkml.org/lkml/2007/8/10/118)
>> was added to the -mm tree in 2.6.23-rc3-mm1.
>>
>> With CONFIG_CC_OPTIMIZE_FOR_SIZE=y this patchset increases the kernel
>> text size by about 5K (closer to 6K when the config token is set to N).
>

> Well, this is a new functionality, so the vmlinux size has to increase.
> And as Matt has noticed 5k is not that bad :)

Speaking with my LinuxBIOS hat on, even 1kB more can be the critical
code portion which causes the Linux kernel not to fit onto a BIOS ROM
chip any more. We already optimize for size and use LZMA compression to
fit a kernel onto an 1MB chip.

>> We also do not have a config token to select pid namespace (its always
>> enabled).
>
> This was one of Andrew's requirements. However, I assume that we can
> have the cloning code under this option.

If all new code which increases text size can be disabled via config
options, I'm happy. However, I fear that some size increase will be
unavoidable even with the config token disabled. As long as that doesn't
end up being more than 0.5kB or at maximum 1kB, it will be probably OK.

Thanks for taking size issues into consideration!

Regards,
Carl-Daniel

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

---

Subject: Re: Kernel text size with pid namespace
Posted by Matt Mackall on Fri, 21 Sep 2007 15:15:55 GMT
View Forum Message <> Reply to Message

On Thu, Sep 20, 2007 at 10:03:59PM -0700, sukadev@us.ibm.com wrote:
> Matt Mackall [mpm@selenic.com] wrote:
> | On Wed, Sep 19, 2007 at 05:16:44PM -0700, sukadev@us.ibm.com wrote:
> | > Matt,
> | >
> | > The pid-namespace patcheset (http://lkml.org/lkml/2007/8/10/118)
> | > was added to the -mm tree in 2.6.23-rc3-mm1.
> | >
> | > With CONFIG_CC_OPTIMIZE_FOR_SIZE=y this patchset increases the kernel
> | > text size by about 5K (closer to 6K when the config token is set to N).
> |
> | That's not too bad.
>
> Ok, thanks, I won't worry about for now :-)
>

> Just curious, is there a magic number like 8K or 32K increase in size (of
> unconditional code) that one should watch out for ?

All size increase is a step backward for folks who already have a
working kernel. There are today more than a million cellphones running
Linux where the number one priority is reducing footprint. PID
namespaces are quite low on their wishlist. So as long as the feature
is non-optional, you've got a fairly heavy burden to make it as small
as possible. Especially as this is just the first part of several
namespace pieces.

--
Mathematics is the supreme nostalgia of our time.