Subject: [PATCH 0/3] Make tasks always have non-zero pids
Posted by Pavel Emelianov on Wed, 03 Oct 2007 14:16:47 GMT
View Forum Message <> Reply to Message

Some time ago Sukadev noticed that the vmlinux size has
grown 5Kb due to merged pid namespaces. One of the big
problems with it was fat inline functions. The other thing
was noticed by Matt - the checks for task's pid to be not
NULL take place and make the kernel grow due to inlining,
but these checks are not always needed.

In this series I introduce a static pid (dummy), according
to Matt's proposal, which is assigned to tasks during the
detach_pid and transfer_pid instead of NULL. This pid lives
in the init pid namespace and has the id = 0, so all the
task_xid_xnr() calls will still return 0 on a dead task.

Places that get the struct pid from task either get it from
the current (in this case they will never get this dummy),
or use it to compare with some other value (so they will
work the same for both NULL and dummy pids).

This saves up to 340 bytes for i386 kernel with minimal
config and probably more with more users of pids.

Tested on i386 and x86_64 boxes. Tasks still live and die,
namespaces and proc still work.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Subject: [PATCH 1/3] Introduce the dummy_pid
Posted by Pavel Emelianov on Wed, 03 Oct 2007 14:19:01 GMT
View Forum Message <> Reply to Message

This is a pid which is attached to tasks when they detach
their pids. This is done in detach_pid() and transfer_pid().
The pid_alive() check is changed to reflect this fact.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

---

diff --git a/include/linux/sched.h b/include/linux/sched.h
index 4f21af1..e17b8f8 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1286,9 +1286,11 @@ static inline pid_t task_ppid_nr_ns(stru

```
  * If pid_alive fails, then pointers within the task structure
  * can be stale and must not be dereferenced.
  */
+extern struct pid dummy_pid;
+
 static inline int pid_alive(struct task_struct *p)
 {
- return p->pids[PIDTYPE_PID].pid != NULL;
+ return p->pids[PIDTYPE_PID].pid != &dummy_pid;
 }

 /**
diff --git a/kernel/pid.c b/kernel/pid.c
index d7388d7..b7a11cf 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -81,6 +81,17 @@ struct pid_namespace init_pid_ns = {
 };
 EXPORT_SYMBOL_GPL(init_pid_ns);

+struct pid dummy_pid = {
+ .count   = ATOMIC_INIT(1),
+ .numbers = { {
+  .nr  = 0,  /* this is what pid_nr will return
+        * for tasks with no pids
+        */
+  .ns  = &init_pid_ns,
+ }, }
+};
+EXPORT_SYMBOL(dummy_pid);
+
 int is_cgroup_init(struct task_struct *tsk)
 {
  int ret = 0;
@@ -339,7 +350,7 @@ void fastcall detach_pid(struct task_str
  pid = link->pid;

  hlist_del_rcu(&link->node);
- link->pid = NULL;
+ link->pid = &dummy_pid;

  for (tmp = PIDTYPE_MAX; --tmp >= 0; )
   if (!hlist_empty(&pid->tasks[tmp]))
@@ -354,7 +365,7 @@ void fastcall transfer_pid(struct task_s
 {
  new->pids[type].pid = old->pids[type].pid;
  hlist_replace_rcu(&old->pids[type].node, &new->pids[type].node);
- old->pids[type].pid = NULL;
```

```
+ old->pids[type].pid = &dummy_pid;
 }

 struct task_struct * fastcall pid_task(struct pid *pid, enum pid_type type)
```

Subject: [PATCH 2/3] Prepare pid_nr() etc functions to work with not-NULL pids
Posted by Pavel Emelianov on Wed, 03 Oct 2007 14:20:43 GMT
View Forum Message <> Reply to Message

Just make the __pid_nr() etc functions that expect the argument
to always be not NULL.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/include/linux/pid.h b/include/linux/pid.h
index e29a900..50b6899 100644
--- a/include/linux/pid.h
+++ b/include/linux/pid.h
@@ -135,21 +135,32 @@ extern void zap_pid_ns_processes(struct
  * see also task_xid_nr() etc in include/linux/sched.h
  */

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

+pid_t __pid_nr_ns(struct pid *pid, struct pid_namespace *ns);
 pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns);

+static inline pid_t __pid_vnr(struct pid *pid)
+{
+ return pid->numbers[pid->level].nr;
+}
+
 static inline pid_t pid_vnr(struct pid *pid)
```

```
 {
  pid_t nr = 0;
  if (pid)
-  nr = pid->numbers[pid->level].nr;
+  nr = __pid_vnr(pid);
  return nr;
 }

diff --git a/kernel/pid.c b/kernel/pid.c
index d7388d7..b7a11cf 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -430,12 +441,12 @@ struct pid *find_get_pid(pid_t nr)
  return pid;
 }

-pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
+pid_t __pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
 {
  struct upid *upid;
  pid_t nr = 0;

- if (pid && ns->level <= pid->level) {
+ if (ns->level <= pid->level) {
    upid = &pid->numbers[ns->level];
    if (upid->ns == ns)
     nr = upid->nr;
@@ -443,6 +454,14 @@ pid_t pid_nr_ns(struct pid *pid, struct
  return nr;
 }

+pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
+{
+ pid_t nr = 0;
+ if (pid)
+  nr = __pid_nr_ns(pid, ns);
+ return nr;
+}
+
 pid_t task_pid_nr_ns(struct task_struct *tsk, struct pid_namespace *ns)
 {
  return pid_nr_ns(task_pid(tsk), ns);
```

Subject: [PATCH 3/3] Use the __pid_nr() calls where appropriate
Posted by Pavel Emelianov on Wed, 03 Oct 2007 14:23:20 GMT
View Forum Message <> Reply to Message

Since we now have all the tasks have non-zero pids we may
call the __pid_nr() instead of pid_nr() when the pid is
get from task with task_pid() or similar call.

Besides, there are some other places where the check for
pid to be not NULL is already done, so change them too.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/fs/proc/base.c b/fs/proc/base.c
index 6ea12de..d94fda9 100644
--- a/fs/proc/base.c
+++ b/fs/proc/base.c
@@ -2394,7 +2394,7 @@ retry:
  task = NULL;
  pid = find_ge_pid(tgid, ns);
  if (pid) {
-  tgid = pid_nr_ns(pid, ns) + 1;
+  tgid = __pid_nr_ns(pid, ns) + 1;
   task = pid_task(pid, PIDTYPE_PID);
   /* What we to know is if the pid we have find is the
    * pid of a thread_group_leader.  Testing for task
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 4f21af1..e17b8f8 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1229,7 +1229,7 @@ pid_t task_pid_nr_ns(struct task_struct

 static inline pid_t task_pid_vnr(struct task_struct *tsk)
 {
- return pid_vnr(task_pid(tsk));
+ return __pid_vnr(task_pid(tsk));
 }


@@ -1242,7 +1242,7 @@ pid_t task_tgid_nr_ns(struct task_struct

 static inline pid_t task_tgid_vnr(struct task_struct *tsk)
 {
- return pid_vnr(task_tgid(tsk));
+ return __pid_vnr(task_tgid(tsk));
 }


@@ -1255,7 +1255,7 @@ pid_t task_pgrp_nr_ns(struct task_struct
```

```
 static inline pid_t task_pgrp_vnr(struct task_struct *tsk)
 {
- return pid_vnr(task_pgrp(tsk));
+ return __pid_vnr(task_pgrp(tsk));
 }


@@ -1268,14 +1268,14 @@ pid_t task_session_nr_ns(struct task_str

 static inline pid_t task_session_vnr(struct task_struct *tsk)
 {
- return pid_vnr(task_session(tsk));
+ return __pid_vnr(task_session(tsk));
 }


 static inline pid_t task_ppid_nr_ns(struct task_struct *tsk,
   struct pid_namespace *ns)
 {
- return pid_nr_ns(task_pid(rcu_dereference(tsk->real_parent)), ns);
+ return __pid_nr_ns(task_pid(rcu_dereference(tsk->real_parent)), ns);
 }

 /**
diff --git a/kernel/fork.c b/kernel/fork.c
index f85731a..b640a10 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1161,7 +1161,7 @@ static struct task_struct *copy_process(
   }
  }

- p->pid = pid_nr(pid);
+ p->pid = __pid_nr(pid);
  p->tgid = p->pid;
  if (clone_flags & CLONE_THREAD)
   p->tgid = current->tgid;
diff --git a/kernel/pid.c b/kernel/pid.c
index d7388d7..b7a11cf 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -88,7 +99,7 @@ int is_cgroup_init(struct task_struct *t

  rcu_read_lock();
  pid = task_pid(tsk);
- if (pid != NULL && pid->numbers[pid->level].nr == 1)
+ if (pid->numbers[pid->level].nr == 1)
   ret = 1;
```

```
  rcu_read_unlock();

@@ -443,25 +454,25 @@ pid_t pid_nr_ns(struct pid *pid, struct

 pid_t task_pid_nr_ns(struct task_struct *tsk, struct pid_namespace *ns)
 {
- return pid_nr_ns(task_pid(tsk), ns);
+ return __pid_nr_ns(task_pid(tsk), ns);
 }
 EXPORT_SYMBOL(task_pid_nr_ns);

 pid_t task_tgid_nr_ns(struct task_struct *tsk, struct pid_namespace *ns)
 {
- return pid_nr_ns(task_tgid(tsk), ns);
+ return __pid_nr_ns(task_tgid(tsk), ns);
 }
 EXPORT_SYMBOL(task_tgid_nr_ns);

 pid_t task_pgrp_nr_ns(struct task_struct *tsk, struct pid_namespace *ns)
 {
- return pid_nr_ns(task_pgrp(tsk), ns);
+ return __pid_nr_ns(task_pgrp(tsk), ns);
 }
 EXPORT_SYMBOL(task_pgrp_nr_ns);

 pid_t task_session_nr_ns(struct task_struct *tsk, struct pid_namespace *ns)
 {
- return pid_nr_ns(task_session(tsk), ns);
+ return __pid_nr_ns(task_session(tsk), ns);
 }
 EXPORT_SYMBOL(task_session_nr_ns);

diff --git a/kernel/sys.c b/kernel/sys.c
index 796299c..fa187d2 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -975,7 +975,7 @@ asmlinkage long sys_setpgid(pid_t pid, p
   detach_pid(p, PIDTYPE_PGID);
   pid = find_vpid(pgid);
   attach_pid(p, PIDTYPE_PGID, pid);
-  set_task_pgrp(p, pid_nr(pid));
+  set_task_pgrp(p, __pid_nr(pid));
  }

  err = 0;
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index b11d22b..9bda3b5 100644
--- a/kernel/sysctl.c
```

```
+++ b/kernel/sysctl.c
@@ -2286,7 +2286,7 @@ static int proc_do_cad_pid(struct ctl_ta
  pid_t tmp;
  int r;

- tmp = pid_nr_ns(cad_pid, current->nsproxy->pid_ns);
+ tmp = __pid_nr_ns(cad_pid, current->nsproxy->pid_ns);

  r = __do_proc_dointvec(&tmp, table, write, filp, buffer,
         lenp, ppos, NULL, NULL);
```

Subject: Re: [PATCH 2/3] Prepare pid_nr() etc functions to work with not-NULL pids
Posted by Matt Mackall on Wed, 03 Oct 2007 16:42:58 GMT
View Forum Message <> Reply to Message

On Wed, Oct 03, 2007 at 06:20:43PM +0400, Pavel Emelyanov wrote:
> Just make the __pid_nr() etc functions that expect the argument
> to always be not NULL.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

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

Is there a patch that removes these inlines? Otherwise this looks good
to me.

--
Mathematics is the supreme nostalgia of our time.

Subject: Re: [PATCH 1/3] Introduce the dummy_pid
Posted by Randy Dunlap on Wed, 03 Oct 2007 18:06:10 GMT
View Forum Message <> Reply to Message

On Wed, 03 Oct 2007 18:19:01 +0400 Pavel Emelyanov wrote:

> This is a pid which is attached to tasks when they detach
> their pids. This is done in detach_pid() and transfer_pid().
> The pid_alive() check is changed to reflect this fact.

>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
> ---
>
> diff --git a/include/linux/sched.h b/include/linux/sched.h
> index 4f21af1..e17b8f8 100644
> --- a/include/linux/sched.h
> +++ b/include/linux/sched.h
> @@ -1286,9 +1286,11 @@ static inline pid_t task_ppid_nr_ns(stru
> * If pid_alive fails, then pointers within the task structure
> * can be stale and must not be dereferenced.
> */
> +extern struct pid dummy_pid;
> +
> static inline int pid_alive(struct task_struct *p)
> {
> - return p->pids[PIDTYPE_PID].pid != NULL;
> + return p->pids[PIDTYPE_PID].pid != &dummy_pid;
> }
>
> /**
> diff --git a/kernel/pid.c b/kernel/pid.c
> index d7388d7..b7a11cf 100644
> --- a/kernel/pid.c
> +++ b/kernel/pid.c
> @@ -81,6 +81,17 @@ struct pid_namespace init_pid_ns = {
> };
> EXPORT_SYMBOL_GPL(init_pid_ns);
>
> +struct pid dummy_pid = {
> + .count = ATOMIC_INIT(1),
> + .numbers = { {
> + .nr = 0, /* this is what pid_nr will return
> + * for tasks with no pids
> + */
> + .ns = &init_pid_ns,
> + }, }
> +};
> +EXPORT_SYMBOL(dummy_pid);

Why is the EXPORT_SYMBOL() needed?
IOW, is some loadable module going to use this?

~Randy

Subject: Re: [PATCH 2/3] Prepare pid_nr() etc functions to work with not-NULL pids
Posted by Pavel Emelianov on Thu, 04 Oct 2007 08:54:17 GMT
View Forum Message <> Reply to Message

Matt Mackall wrote:
> On Wed, Oct 03, 2007 at 06:20:43PM +0400, Pavel Emelyanov wrote:
>> Just make the __pid_nr() etc functions that expect the argument
>> to always be not NULL.
>>
>> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>
>> static inline pid_t pid_nr(struct pid *pid)
>> {
>>   pid_t nr = 0;
>>   if (pid)
>> - nr = pid->numbers[0].nr;
>> + nr = __pid_nr(pid);
>>   return nr;
>> }
>
> Is there a patch that removes these inlines? Otherwise this looks good
> to me.

Not yet. Some of are uninlined already, but others are not. I'd like
to make some testing before uninline them.

> --
> Mathematics is the supreme nostalgia of our time.
>

Subject: Re: [PATCH 1/3] Introduce the dummy_pid
Posted by Pavel Emelianov on Thu, 04 Oct 2007 08:56:37 GMT
View Forum Message <> Reply to Message

Randy Dunlap wrote:
> On Wed, 03 Oct 2007 18:19:01 +0400 Pavel Emelyanov wrote:
>
>> This is a pid which is attached to tasks when they detach
>> their pids. This is done in detach_pid() and transfer_pid().
>> The pid_alive() check is changed to reflect this fact.
>>
>> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>>
>> ---
>>
>> diff --git a/include/linux/sched.h b/include/linux/sched.h
>> index 4f21af1..e17b8f8 100644

```
>> --- a/include/linux/sched.h
>> +++ b/include/linux/sched.h
>> @@ -1286,9 +1286,11 @@ static inline pid_t task_ppid_nr_ns(stru
>>   * If pid_alive fails, then pointers within the task structure
>>   * can be stale and must not be dereferenced.
>>   */
>> +extern struct pid dummy_pid;
>> +
>>  static inline int pid_alive(struct task_struct *p)
>>  {
>> - return p->pids[PIDTYPE_PID].pid != NULL;
>> + return p->pids[PIDTYPE_PID].pid != &dummy_pid;
>>  }
>>
>>  /**
>> diff --git a/kernel/pid.c b/kernel/pid.c
>> index d7388d7..b7a11cf 100644
>> --- a/kernel/pid.c
>> +++ b/kernel/pid.c
>> @@ -81,6 +81,17 @@ struct pid_namespace init_pid_ns = {
>>  };
>>  EXPORT_SYMBOL_GPL(init_pid_ns);
>>
>> +struct pid dummy_pid = {
>> + .count   = ATOMIC_INIT(1),
>> + .numbers = { {
>> +  .nr  = 0,  /* this is what pid_nr will return
>> +        * for tasks with no pids
>> +        */
>> +  .ns  = &init_pid_ns,
>> + }, }
>> +};
>> +EXPORT_SYMBOL(dummy_pid);
>
> Why is the EXPORT_SYMBOL() needed?

Because pid_alive() uses it and is declared in sched.h, so
some module can use it.

> IOW, is some loadable module going to use this?

Right now - no modules use it. You're right - I will remove this export.

> ---
> ~Randy
>
```

Subject: Re: [PATCH 2/3] Prepare pid_nr() etc functions to work with not-NULL pids
Posted by Matt Mackall on Thu, 04 Oct 2007 17:13:11 GMT
View Forum Message <> Reply to Message

On Thu, Oct 04, 2007 at 12:54:17PM +0400, Pavel Emelyanov wrote:
> Matt Mackall wrote:
> > On Wed, Oct 03, 2007 at 06:20:43PM +0400, Pavel Emelyanov wrote:
> >> Just make the __pid_nr() etc functions that expect the argument
> >> to always be not NULL.
> >>
> >> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
> >
> > static inline pid_t pid_nr(struct pid *pid)
> > {
> >   pid_t nr = 0;
> >   if (pid)
> > - nr = pid->numbers[0].nr;
> > + nr = __pid_nr(pid);
> >   return nr;
> > }
> >
> > Is there a patch that removes these inlines? Otherwise this looks good
> > to me.
>
> Not yet. Some of are uninlined already, but others are not. I'd like
> to make some testing before uninline them.

I was asking about the whole function, actually, not the keyword. Is
this function not equivalent to __pid_nr now?

--
Mathematics is the supreme nostalgia of our time.

Subject: Re: [PATCH 0/3] Make tasks always have non-zero pids
Posted by Sukadev Bhattiprolu on Fri, 05 Oct 2007 05:53:17 GMT
View Forum Message <> Reply to Message

Pavel Emelianov [xemul@openvz.org] wrote:
| Some time ago Sukadev noticed that the vmlinux size has

Cedric pointed it out to me first :-)

| grown 5Kb due to merged pid namespaces. One of the big
| problems with it was fat inline functions. The other thing
| was noticed by Matt - the checks for task's pid to be not
| NULL take place and make the kernel grow due to inlining,
| but these checks are not always needed.

|
| In this series I introduce a static pid (dummy), according
| to Matt's proposal, which is assigned to tasks during the
| detach_pid and transfer_pid instead of NULL. This pid lives
| in the init pid namespace and has the id = 0, so all the
| task_xid_xnr() calls will still return 0 on a dead task.
|
| Places that get the struct pid from task either get it from
| the current (in this case they will never get this dummy),
| or use it to compare with some other value (so they will
| work the same for both NULL and dummy pids).
|
| This saves up to 340 bytes for i386 kernel with minimal
| config and probably more with more users of pids.
|
| Tested on i386 and x86_64 boxes. Tasks still live and die,
| namespaces and proc still work.
|
| Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Acked-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

Subject: Re: [PATCH 2/3] Prepare pid_nr() etc functions to work with not-NULL pids
Posted by Pavel Emelianov on Fri, 05 Oct 2007 13:16:19 GMT
View Forum Message <> Reply to Message

Matt Mackall wrote:
> On Thu, Oct 04, 2007 at 12:54:17PM +0400, Pavel Emelyanov wrote:
>> Matt Mackall wrote:
>>> On Wed, Oct 03, 2007 at 06:20:43PM +0400, Pavel Emelyanov wrote:
>>>> Just make the __pid_nr() etc functions that expect the argument
>>>> to always be not NULL.
>>>>
>>>> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
>>>> static inline pid_t pid_nr(struct pid *pid)
>>>> {
>>>>   pid_t nr = 0;
>>>>   if (pid)
>>>> - nr = pid->numbers[0].nr;
>>>> + nr = __pid_nr(pid);
>>>>   return nr;
>>>> }
>>> Is there a patch that removes these inlines? Otherwise this looks good
>>> to me.
>> Not yet. Some of are uninlined already, but others are not. I'd like
>> to make some testing before uninline them.
>

> I was asking about the whole function, actually, not the keyword. Is
> this function not equivalent to __pid_nr now?

Oh, I see. I haven't managed to check the whole kernel yet that all
the users of pid_xnr() calls pass not-null pointer there. This is
in TODO list.

Thanks,
Pavel
