
Subject: Re: [PATCH 10/16] Changes in copy_process() to work with pid namespaces

Posted by [Sukadev Bhattiprolu](#) on Thu, 12 Jul 2007 00:21:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

Pavel Emelianov [xemul@openvz.org] wrote:

| We must pass the namespace pointer to the alloc_pid() to
| show what namespace to allocate the pid from and we should
| call this *after* the namespace is copied.

|
| Essentially, the task->pid etc initialization is done after
| the alloc_pid().

|
| To do so I move the alloc_pid() inside copy_process() and
| introduce an argument to the alloc_pid() function.

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

| ---

| include/linux/pid.h | 2 +-
| kernel/fork.c | 29 ++++++-----
| kernel/pid.c | 2 +-
| 3 files changed, 19 insertions(+), 14 deletions(-)

| --- ./include/linux/pid.h.ve9 2007-07-06 11:03:55.000000000 +0400

| +++ ./include/linux/pid.h 2007-07-06 11:03:55.000000000 +0400

| @@ -116,7 +116,7 @@ extern struct pid *FASTCALL(find_pid_ns(
| extern struct pid *find_get_pid(int nr);
| extern struct pid *find_ge_pid(int nr, struct pid_namespace *);

| -extern struct pid *alloc_pid(void);
| +extern struct pid *alloc_pid(struct pid_namespace *ns);
| extern void FASTCALL(free_pid(struct pid *pid));

| /*

| --- ./kernel/fork.c.ve9 2007-07-06 11:03:55.000000000 +0400

| +++ ./kernel/fork.c 2007-07-06 11:04:07.000000000 +0400

| @@ -50,6 +50,7 @@
| #include <linux/taskstats_kern.h>
| #include <linux/random.h>
| #include <linux/tty.h>
| +#include <linux/pid.h>

| #include <asm/pgtable.h>

| #include <asm/pgalloc.h>

| @@ -1032,7 +1033,6 @@ static struct task_struct *copy_process(
| p->did_exec = 0;

```

| delayacct_tsk_init(p); /* Must remain after dup_task_struct() */
| copy_flags(clone_flags, p);
| - p->pid = pid_nr(pid);
| INIT_LIST_HEAD(&p->children);
| INIT_LIST_HEAD(&p->sibling);
| p->vfork_done = NULL;
| @@ -1107,10 +1107,6 @@ static struct task_struct *copy_process(
| p->blocked_on = NULL; /* not blocked yet */
| #endif
|
| - p->tgid = p->pid;
| - if (clone_flags & CLONE_THREAD)
| - p->tgid = current->tgid;
| -
| if ((retval = security_task_alloc(p)))
| goto bad_fork_cleanup_policy;
| if ((retval = audit_alloc(p)))
| @@ -1132,9 +1128,14 @@ static struct task_struct *copy_process(
| goto bad_fork_cleanup_mm;
| if ((retval = copy_namespaces(clone_flags, p)))
| goto bad_fork_cleanup_keys;
| + if (likely(pid == NULL)) {
| + pid = alloc_pid(p->nsproxy->pid_ns);
| + if (pid == NULL)
| + goto bad_fork_cleanup_namespaces;
| + }
| retval = copy_thread(0, clone_flags, stack_start, stack_size, p, regs);
| if (retval)
| - goto bad_fork_cleanup_namespaces;
| + goto bad_fork_cleanup_pid;
|
| p->set_child_tid = (clone_flags & CLONE_CHILD_SETTID) ? child_tidptr : NULL;
| /*
| @@ -1255,6 +1256,11 @@ static struct task_struct *copy_process(
| }
| }
|
| + p->pid = pid_nr(pid);
| + p->tgid = p->pid;
| + if (clone_flags & CLONE_THREAD)
| + p->tgid = current->tgid;
| +
| if (likely(p->pid)) {
| add_parent(p);
| if (unlikely(p->ptrace & PT_PTRACED))
| @@ -1288,6 +1294,8 @@ static struct task_struct *copy_process(
| proc_fork_connector(p);
| return p;

```

```
|
| +bad_fork_cleanup_pid:
| + free_pid(pid);
```

If copy_process() was called from fork_idle(), pid would refer to &init_struct_pid. Don't you need something like:

```
if (pid == &init_struct_pid)
    return;
```

in free_pid() to not do anything in that case ?

```
| bad_fork_cleanup_namespaces:
|   exit_task_namespaces(p);
| bad_fork_cleanup_keys:
| @@ -1380,19 +1388,16 @@ long do_fork(unsigned long clone_flags,
| {
|   struct task_struct *p;
|   int trace = 0;
| - struct pid *pid = alloc_pid();
|   long nr;
|
| - if (!pid)
| -   return -EAGAIN;
| - nr = pid->nr;
|   if (unlikely(current->ptrace)) {
|     trace = fork_traceflag (clone_flags);
|     if (trace)
|       clone_flags |= CLONE_PTRACE;
|   }
|
| - p = copy_process(clone_flags, stack_start, regs, stack_size, parent_tidptr, child_tidptr, pid);
| + p = copy_process(clone_flags, stack_start, regs, stack_size,
| +   parent_tidptr, child_tidptr, NULL);
|   /*
|    * Do this prior waking up the new thread - the thread pointer
|    * might get invalid after that point, if the thread exits quickly.
|   @@ -1418,6 +1423,7 @@ long do_fork(unsigned long clone_flags,
|   else
|     p->state = TASK_STOPPED;
|
| + nr = pid_vnr(task_pid(p));
|   if (unlikely (trace)) {
|     current->ptrace_message = nr;
|     ptrace_notify ((trace << 8) | SIGTRAP);
|   @@ -1433,7 +1439,6 @@ long do_fork(unsigned long clone_flags,
|   }
```

```

| }
| } else {
| - free_pid(pid);
|   nr = PTR_ERR(p);
| }
| return nr;
| --- ./kernel/pid.c.ve9 2007-07-06 11:03:55.000000000 +0400
| +++ ./kernel/pid.c 2007-07-06 11:03:55.000000000 +0400
| @@ -206,7 +206,7 @@ fastcall void free_pid(struct pid *pid)
|   call_rcu(&pid->rcu, delayed_put_pid);
| }
|
| -struct pid *alloc_pid(void)
| +struct pid *alloc_pid(struct pid_namespace *pid_ns)
| {
|   struct pid *pid;
|   enum pid_type type;

```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
