Subject: Re: [RFC][PATCH 07/16] Move alloc_pid call to copy_process
Posted by Pavel Emelianov on Fri, 25 May 2007 06:41:58 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Subject: Move alloc_pid call to copy_process
>
> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
>
> Move alloc_pid() into copy_process(). This will keep all pid and pid
> namespace code together and simplify error handling when we support
> multiple pid namespaces.

I haven't found this in patches, so I ask it here:

We clone a new task with CLONE_NEWPIDS flag. This task
allocates its PIDTYPE_PID pid and this pid happens in
booth parent and child namespace. This is OK. Then new
task attaches PIDTYPE_SID and PIDTYPE_PGID pids from parent
task. But these ones are in parent namespace only.

Right? Is that good?

> Changelog:
> - [Eric Biederman] Move the check of copy_process_type to alloc_pid()/
> free_pid() and to avoid clutter in copy_process().
>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
> ---
> include/linux/pid.h | 7 ++++++-
> kernel/fork.c | 21 +++++++++++++++--------
> kernel/pid.c | 10 +++++++++-
> 3 files changed, 28 insertions(+), 10 deletions(-)
>
> Index: lx26-21-mm2/include/linux/pid.h
> ==========
> --- lx26-21-mm2.orig/include/linux/pid.h 2007-05-22 16:59:40.000000000 -0700
> +++ lx26-21-mm2/include/linux/pid.h 2007-05-22 17:06:48.000000000 -0700
> @@ -3,6 +3,11 @@
>
> #include <linux/rcupdate.h>
>
> +enum copy_process_type {
> + COPY_NON_IDLE_PROCESS,
> + COPY_IDLE_PROCESS,
> +};
> +
> enum pid_type

```
>  {
>   PIDTYPE_PID,
> @@ -95,7 +100,7 @@ extern struct pid *FASTCALL(find_pid(int
>  extern struct pid *find_get_pid(int nr);
>  extern struct pid *find_ge_pid(int nr);
>
> -extern struct pid *alloc_pid(void);
> +extern struct pid *alloc_pid(enum copy_process_type);
>  extern void FASTCALL(free_pid(struct pid *pid));
>
>  static inline pid_t pid_to_nr(struct pid *pid)
> Index: lx26-21-mm2/kernel/fork.c
> ===================================================================
> --- lx26-21-mm2.orig/kernel/fork.c 2007-05-22 16:59:41.000000000 -0700
> +++ lx26-21-mm2/kernel/fork.c 2007-05-22 17:06:48.000000000 -0700
> @@ -961,10 +961,11 @@ static struct task_struct *copy_process(
>       unsigned long stack_size,
>       int __user *parent_tidptr,
>       int __user *child_tidptr,
> -     struct pid *pid)
> +     enum copy_process_type copy_src)
>  {
>   int retval;
>   struct task_struct *p = NULL;
> + struct pid *pid;
>
>   if ((clone_flags & (CLONE_NEWNS|CLONE_FS)) == (CLONE_NEWNS|CLONE_FS))
>    return ERR_PTR(-EINVAL);
> @@ -1025,6 +1026,10 @@ static struct task_struct *copy_process(
>   if (p->binfmt && !try_module_get(p->binfmt->module))
>    goto bad_fork_cleanup_put_domain;
>
> + pid = alloc_pid(copy_src);
> + if (!pid)
> +  goto bad_fork_put_binfmt_module;
> +
>   p->did_exec = 0;
>   delayacct_tsk_init(p); /* Must remain after dup_task_struct() */
>   copy_flags(clone_flags, p);
> @@ -1305,6 +1310,8 @@ bad_fork_cleanup_cpuset:
>  #endif
>   cpuset_exit(p);
>   delayacct_tsk_free(p);
> + free_pid(pid);
> +bad_fork_put_binfmt_module:
>   if (p->binfmt)
>    module_put(p->binfmt->module);
>  bad_fork_cleanup_put_domain:
```

```
> @@ -1331,7 +1338,7 @@ struct task_struct * __cpuinit fork_idle
>   struct pt_regs regs;
>
>   task = copy_process(CLONE_VM, 0, idle_regs(&regs), 0, NULL, NULL,
> -    &init_struct_pid);
> +    COPY_IDLE_PROCESS);
>   if (!IS_ERR(task))
>    init_idle(task, cpu);
>
> @@ -1369,19 +1376,16 @@ 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, COPY_NON_IDLE_PROCESS);
>   /*
>    * Do this prior waking up the new thread - the thread pointer
>    * might get invalid after that point, if the thread exits quickly.
> @@ -1389,6 +1393,8 @@ long do_fork(unsigned long clone_flags,
>   if (!IS_ERR(p)) {
>    struct completion vfork;
>
> +  nr = pid_to_nr(task_pid(p));
> +
>    if (clone_flags & CLONE_VFORK) {
>     p->vfork_done = &vfork;
>     init_completion(&vfork);
> @@ -1422,7 +1428,6 @@ long do_fork(unsigned long clone_flags,
>     }
>    }
>   } else {
> -  free_pid(pid);
>    nr = PTR_ERR(p);
>   }
>   return nr;
```

> Index: lx26-21-mm2/kernel/pid.c
> ===
> --- lx26-21-mm2.orig/kernel/pid.c 2007-05-22 16:59:46.000000000 -0700
> +++ lx26-21-mm2/kernel/pid.c 2007-05-22 17:06:48.000000000 -0700
> @@ -216,6 +216,10 @@ fastcall void free_pid(struct pid *pid)
> /* We can be called with write_lock_irq(&tasklist_lock) held */
> unsigned long flags;
>
> + /* check this here to keep copy_process() cleaner */
> + if (unlikely(pid == &init_struct_pid))
> + return;
> +
> spin_lock_irqsave(&pidmap_lock, flags);
> hlist_del_rcu(&pid->pid_chain);
> spin_unlock_irqrestore(&pidmap_lock, flags);
> @@ -224,12 +228,16 @@ fastcall void free_pid(struct pid *pid)
> call_rcu(&pid->rcu, delayed_put_pid);
> }
>
> -struct pid *alloc_pid(void)
> +struct pid *alloc_pid(enum copy_process_type copy_src)
> {
> struct pid *pid;
> enum pid_type type;
> int nr = -1;
>
> + /* check this here to keep copy_process() cleaner */
> + if (unlikely(copy_src == COPY_IDLE_PROCESS))
> + return &init_struct_pid;
> +
> pid = kmem_cache_alloc(pid_cachep, GFP_KERNEL);
> if (!pid)
> goto out;
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>

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