
Subject: Re: [PATCH 13/16] Switch to operating with pid_numbers instead of pids
Posted by [Pavel Emelianov](#) on Thu, 26 Jul 2007 06:42:22 GMT
[View Forum Message](#) <> [Reply to Message](#)

sukadev@us.ibm.com wrote:

> Pavel Emelianov [xemul@openvz.org] wrote:

> | sukadev@us.ibm.com wrote:

> | > Pavel Emelianov [xemul@openvz.org] wrote:

> | > | Make alloc_pid() initialize pid_numbers and hash them

> | > | into the hashtable, not the struct pid itself.

> | > |

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

> | > |

> | > | ---

> | > |

> | > | pid.c | 47 ++++++-----

> | > | 1 files changed, 33 insertions(+), 14 deletions(-)

> | > |

> | > | --- ./kernel/pid.c.ve12 2007-07-05 11:06:41.000000000 +0400

> | > | +++ ./kernel/pid.c 2007-07-05 11:08:23.000000000 +0400

> | > | @@ -28,8 +28,10 @@

> | > | #include <linux/hash.h>

> | > | #include <linux/pid_namespace.h>

> | > | #include <linux/init_task.h>

> | > | +#include <linux/proc_fs.h>

> | > |

> | > | -#define pid_hashfn(nr) hash_long((unsigned long)nr, pidhash_shift)

> | > | +#define pid_hashfn(nr, ns) \

> | > | + hash_long((unsigned long)nr + (unsigned long)ns, pidhash_shift)

> | > | static struct hlist_head *pid_hash;

> | > | static int pidhash_shift;

> | > | struct pid init_struct_pid = INIT_STRUCT_PID;

> | > | @@ -194,7 +198,7 @@ fastcall void put_pid(struct pid *pid)

> | > | if (!pid)

> | > | return;

> | > |

> | > | - ns = pid->numbers[0].ns;

> | > | + ns = pid->numbers[pid->level].ns;

> | > | if ((atomic_read(&pid->count) == 1) ||

> | > | atomic_dec_and_test(&pid->count))

> | > | kmem_cache_free(ns->pid_cache, pid);

> | > | @@ -210,13 +214,17 @@ static void delayed_put_pid(struct rcu_h

> | > | fastcall void free_pid(struct pid *pid)

> | > | {

> | > | /* We can be called with write_lock_irq(&tasklist_lock) held */

> | > | + int i;

> | > | unsigned long flags;

> | > |

```

> |> spin_lock_irqsave(&pidmap_lock, flags);
> |> - hlist_del_rcu(&pid->pid_chain);
> |> + for (i = 0; i <= pid->level; i++)
> |> + hlist_del_rcu(&pid->numbers[i].pid_chain);
> |> spin_unlock_irqrestore(&pidmap_lock, flags);
> |>
> |> - free_pidmap(&init_pid_ns, pid->nr);
> |> + for (i = 0; i <= pid->level; i++)
> |> + free_pidmap(pid->numbers[i].ns, pid->numbers[i].nr);
> |> +
> |> call_rcu(&pid->rcu, delayed_put_pid);
> |> }
> |>
@@ -224,30 +232,43 @@ struct pid *alloc_pid(struct pid_namespace
> |> {
> |> struct pid *pid;
> |> enum pid_type type;
> |> - int nr = -1;
> |> + struct pid_namespace *ns;
> |> + int i, nr;
> |>
> |> - pid = kmem_cache_alloc(init_pid_ns.pid_cachep, GFP_KERNEL);
> |> + pid = kmem_cache_alloc(pid_ns->pid_cachep, GFP_KERNEL);
> |> if (!pid)
> |> goto out;
> |>
> |> - nr = alloc_pidmap(current->nsproxy->pid_ns);
> |> - if (nr < 0)
> |> - goto out_free;
> |> + ns = pid_ns;
> |> + for (i = pid_ns->level; i >= 0; i--) {
> |> + nr = alloc_pidmap(ns);
> |> + if (nr < 0)
> |> + goto out_free;
> |>
> |> If pid_ns->level is say 3 and alloc_pidmap() succeeds when i=0,1
> |
> | It cannot :) If level is 3, then we'll allocate for 3, 2, 1, 0 sequence.
> | The loop is descending, not ascending...
>
> Aah descending - thats right. But I still think there is a problem.
>
> Here is your code that I am referring to:
>
> pid = kmem_cache_alloc(pid_ns->pid_cachep, GFP_KERNEL);
> if (!pid)
> goto out;
>

```

```

>     ns = pid_ns;
>     for (i = pid_ns->level; i >= 0; i--) {
>         nr = alloc_pidmap(ns);
>         if (nr < 0)
>             goto out_free;
>
>         pid->numbers[i].nr = nr;
>         pid->numbers[i].ns = ns;
>         ns = ns->parent;
>     }
>
> pid->level = pid_ns->level;
>
> <snip>
>
> out:
> return pid;
> out_free:
>     for (i++; i <= pid->level; i++)
>         free_pidmap(pid->numbers[i].ns, pid->numbers[i].nr);
>
>     kmem_cache_free(pid_ns->pid_cachep, pid);
>     pid = NULL;
>     goto out;
>
> <end code>
>
> Lets say initially pid_ns->level = 3 and alloc_pidmap() succeeds for
> i=3 and i=2 but fails for i=1 and we execute "goto out_free".
>
> But pid->level is uninitialized at this point right ?
>
> Even if it were set to zero (using kmem_cache_zalloc()), we may not
> free the two pidmap entries we allocated for i=3 and i=2.

```

Yes. I found this after detailed look at the code and (hope) fixed.

> Suka
>

Thanks,
Pavel

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