

---

Subject: Re: [RFC] kernel/pid.c pid allocation wierdness

Posted by [xemul](#) on Fri, 16 Mar 2007 11:58:57 GMT

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

---

Eric Dumazet wrote:

> On Friday 16 March 2007 11:57, Pavel Emelianov wrote:

>> Oleg Nesterov wrote:

>>> On 03/14, Eric W. Biederman wrote:

>>>> Pavel Emelianov <xemul@sw.ru> writes:

>>>>> Hi.

>>>>>

>>>>> I'm looking at how alloc\_pid() works and can't understand

>>>>> one (simple/stupid) thing.

>>>>>

>>>>> It first kmem\_cache\_alloc()-s a struct pid, then calls

>>>>> alloc\_pidmap() and at the end it takes a global pidmap\_lock()

>>>>> to add new pid to hash.

>>> We need some global lock. pidmap\_lock is already here, and it is

>>> only used to protect pidmap->page allocation. low, it is almost

>>> unused. So it was very natural to re-use it while implementing

>>> pidrefs.

>>>

>>>>> The question is - why does alloc\_pidmap() use at least

>>>>> two atomic ops and potentially loop to find a zero bit

>>>>> in pidmap? Why not call alloc\_pidmap() under pidmap\_lock

>>>>> and find zero pid in pidmap w/o any loops and atomics?

>>> Currently we search for zero bit lockless, why do you want

>>> to do it under spin\_lock ?

>> Search isn't lockless. Look:

>>

>> while (1) {

>> if (!test\_and\_set\_bit(...)) {

>> atomic\_dec(&nr\_free);

>> return pid;

>> }

>> ...

>> }

>>

>> we use two atomic operations to find and set a bit in a map.

>

> The finding of the zero bit is done without lock. (Search/lookup)

>

> Then , the reservation of the found bit (test\_and\_set\_bit) is done, and

> decrement of nr\_free. It may fail because the search was done lockless.

:\\ I do understand how this algorithm works. What I don't understand is why it is done so, if we take a global lock anyway.

> Finding a zero bit in a 4096 bytes array may consume about 6000 cycles on  
> modern hardware. Much more on SMP/NUMA machines, or on machines where  
> PAGE\_SIZE is 64K instead of 4K :)  
>  
> You don't want to hold pidmad\_lock for so long period.

OK, thanks. That's explanations looks good.

> -  
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in  
> the body of a message to majordomo@vger.kernel.org  
> More majordomo info at <http://vger.kernel.org/majordomo-info.html>  
> Please read the FAQ at <http://www.tux.org/lkml/>  
>

---

Containers mailing list  
Containers@lists.osdl.org  
<https://lists.osdl.org/mailman/listinfo/containers>

---