

## Re: Scheduler degradation since 2.5.66

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-12/3533.html>

---

**From:** George Anzinger ([george\\_at\\_mvista.com](mailto:george_at_mvista.com))

**Date:** 12/16/03

Date: Mon, 15 Dec 2003 16:39:37 -0800  
To: Nick Piggin <[piggin@cyberone.com.au](mailto:piggin@cyberone.com.au)>

Nick Piggin wrote:

>

>

> *Nick Piggin wrote:*

>

>>

>>

>> *Guillaume Foliard wrote:*

>>

>>> *Hello,*

>>>

>>> *I have been playing with kernel 2.5/2.6 for around 6 months now. I was quite pleased with 2.5.65 to see that the soft real-time behaviour was much better than 2.4.x. Since then I tried most of the 2.5/2.6 versions. But recently someone warned me about some degradations with 2.6.0-test6. To show the degradation since 2.5.66 I have run a simple test program on most of the versions. This simple program is measuring the time it takes to a process to be woken up after a call to nanosleep.*

>>> *As the results are plots, please visit this small website for more information : <http://perso.wanadoo.fr/kayakgabon/linux> I'm ready to perform more tests or provide more information if necessary.*

>>>

>>

>> *This isn't a problem with the scheduler, its a problem with sys\_nanosleep.*

>> *jiffies\_to\_timespec( {1000000us} ) returns 2 jiffies, and nanosleep adds an extra one and asks to sleep for that long (ie. 3ms).*

>

>

>

> *I think you should actually sleep for 2 jiffies here. You have asked to sleep for \_at least\_ 1 real millisecond and you really don't care about the number of jiffies that is. Depending on when the last timer interrupt had fired, the next jiffy might be in another microsecond.*

>

## Linux-Kernel: Re: Scheduler degradation since 2.5.66

- > *So I think you really must sleep for that extra jiffy (but 3 is too*
- > *many I think). Notice your first graphs are actually bad, because*
- > *some sleeps are much less than 1000us.*
- >
- > *I don't know much about the timer code though, perhaps you do need to*
- > *sleep for 3 jiffies...*

We get the request at some time  $t$  between tick  $tt$  and  $tt+1$  to sleep for  $N$  ticks. We round this up to the next higher tick count convert to jiffies dropping any fraction and then add 1. So that should be 2 right? This is added to NOW which, in the test code, is pretty well pinned to the last tick plus processing time. So why do you see 3?

What is missing here is that the request was for 1.000000 ms and a tick is really 0.999849 ms. So the request is for a bit more than a tick which we are obligated to round up to 2 ticks. Then adding the 1 tick guard we get the 3 you are seeing. Now if you actually look at that elapsed time you should see it at about 2.999547 ms and ranging down to 1.999698 ms.

Try running the test with a requested sleep time of something less than 0.999849 ms. All this is for the x86 which is using this time to do the best it can with the PIT which can only get this close to 1 ms ticks. You can even vary this number to see exactly where the round up actually happens. Ah, life in the nano world :)

--

George Anzinger [george@mvista.com](mailto:george@mvista.com)

High-res-timers: <http://sourceforge.net/projects/high-res-timers/>

Preemption patch: <http://www.kernel.org/pub/linux/kernel/people/rml>

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to [majordomo@vger.kernel.org](mailto: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/>