

Re: [ANNOUNCE][RFC] PlugSched-6.3.1 for 2.6.16-rc5

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-04/msg01165.html>

- *From:* Peter Williams <pwil3058@xxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 09 Apr 2006 12:58:29 +1000
-

Al Boldi wrote:

Peter Williams wrote:

Al Boldi wrote:

Can you try the attached mem-eater passing it the number of kb to be eaten.

i.e. '# while ;; do ./eatm 9999 ; done'

This will print the number of bytes eaten and the timing in ms.

Adjust the number of kb to be eaten such that the timing will be less than timeslice (120ms by default for spa). Switch to another vt and start pressing enter. A console lockup should follow within seconds for all spas except ebs.

This doesn't seem to present a problem (other than the eatme loop being hard to kill with control-C) on my system using spa_ws with standard settings. I tried both UP and SMP. I may be doing something wrong or perhaps don't understand what you mean by a console lock up.

Switching from one vt to another receives hardly any response.

Aah. Virtual terminals. I was using Gnome terminals under X.

This is especially visible in spa_no_frills, and spa_ws recovers from this lockup somewhat and starts exhibiting this problem as a choking behavior.

Running '# top d.1 (then shift T)' on another vt shows this choking behavior as the proc gets boosted.

When you say "less than the timeslice" how much smaller do you mean?

This depends on your machine's performance. On my 400MhzP2 UP 128MB, w/ spa_no_frills default settings, looping eatm 9999 takes 63ms per eat and causes the rest of the system to be starved. Raising kb to 19999 takes 126ms which is greater than the default 120ms timeslice and causes no system starvation.

What numbers do you get?

For 9999 I get 20ms. I have 1GB of memory and no swapping is taking place but with only 128MB it's possible that your system is swapping and that could make the effect more pronounced.

But anyway, based on the evidence, I think the problem is caused by the fact that the eatm tasks are running to completion in less than one time slice without sleeping and this means that they never have their priorities reassessed. The reason that spa_ebs doesn't demonstrate the problem is that it uses a smaller time slice for the first time slice that a task gets. The reason that it does this is that it gives newly forked processes a fairly high priority and if they're left to run for a full 120 msec at that high priority they can hose the system. Having a shorter first time slice gives the scheduler a chance to reassess the task's priority before it does much damage.

The reason that the other schedulers don't have this strategy is that I didn't think that it was necessary. Obviously I was wrong and should extend it to the other schedulers. It's doubtful whether this will help a great deal with spa_no_frills as it is pure round robin and doesn't reassess priorities except when nice changes of the task changes policies. This is one good reason not to use spa_no_frills on production systems. Perhaps you should consider creating a child scheduler on top of it that meets your needs?

Anyway, an alternative (and safer) way to reduce the effects of this problem (while your waiting for me to do the above change) is to reduce the size of the time slice. The only bad effects of doing this is that you'll do slightly worse (less than 1%) on kernbench.

Peter

--

Peter Williams pwil3058@xxxxxxxxxxxxxxxx

"Learning, n. The kind of ignorance distinguishing the studious."

-- Ambrose Bierce

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>