

Re: [RFC][PATCH] Experimental enhanced NTP error accounting patch

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-03/msg01239.html>

- *From:* Roman Zippel <zippel@xxxxxxxxxxxxxxxxx>
 - *Date:* Sat, 4 Mar 2006 14:55:08 +0100 (CET)
-

Hi,

On Fri, 3 Mar 2006, john stultz wrote:

Hey Roman,
I'm sure you've got a number of things going on, but since I didn't hear anything back from you last time I posted this, I figured I'd try again.

Sorry about that, I was working on other things. Below are some notes which hopefully help to understand the basic parts.

Here is my first pass implementation of your suggested enhanced NTP error accounting for the generic timekeeping code.

I don't think it's the right way to go. You do a lot of extra work and adding pieces of my code on top of it is going to be error-prone. To help understanding the basics of my patch let me outline the core math of it. The ntp code provides the time update per tick:

```
ntp_update = 1sec + adjustments
```

That's all the clock code needs from the ntp code (the details of its calculation is not really important here) and ntp updates are done in tick intervals. For simplicity let's assume HZ=1 so that the clock is updated every tick by:

```
xtime += freq * mult
```

Due to resolution differences this produces an error with every update, which we accumulate as:

```
error += ntp_update - freq * mult
```

Re: [RFC][PATCH] Experimental enhanced NTP error accounting patch

As soon as the error is large we can calculate a new multiplier:

$$\text{freq} * \text{mult_new} = \text{freq} * \text{mult} + \text{error}$$

which becomes:

$$\text{mult_adj} = \text{mult_new} - \text{mult} = \text{error} / \text{freq}$$

Implementation note: The adjustment calculation is a bit more complicated than this for various reasons, but that's really just an implementation detail.

To keep timeofday the same after changing the multiplier we also have to change the base xtime:

$$\text{timeofday} = \text{xtime} + \text{off} * \text{mult} = \text{xtime_new} + \text{off} * \text{mult_new}$$

so simplified xtime is changed as:

$$\text{xtime} -= \text{off} * \text{mult_adj}$$

After changing the multiplier this will also reduce the error:

$$\text{error} -= \text{freq} * \text{mult} - (\text{off} * \text{mult} + (\text{freq} - \text{off}) * \text{mult_new})$$

This is the difference between the old update and the new update (where the new multiplier is only applied for the rest of the tick), simplified this becomes:

$$\text{error} -= (\text{freq} - \text{off}) * \text{mult_adj}$$

Finally setting time and changing clock can be done with the same mechanism:

$$\text{xtime} = \text{timeofday} - \text{off} * \text{mult}$$

timeofday is simply the new time or the current time of the old clock, this means that a clock change can be done outside the periodic hook.

This is basically the core part of my prototype to implement a continuous timeofday, the rest is just an implementation detail (e.g. how mult_adj becomes a shift). It's maybe not trivial, but I don't think it's that hard to understand (at least using an example) and results in very simple code.

Maybe the main question I have about your code now is why you want to calculate nsec intervals during the update, but as you can see above, it's not really necessary, so what do you want to use it for?

Another problem is timeofday_periodic_hook(), while I can understand that you want to make it independent of the timer tick, some kind of timer tick

Re: [RFC][PATCH] Experimental enhanced NTP error accounting patch

is still needed at which ntp time and clock time is synchronized. You go to great lengths to hide it, but it's needed nevertheless, which makes your code harder to read and more complex than needed.

It would be a lot simpler to keep the updates at first at HZ frequency. Later we can still change the update frequency, but ntp time has to be updated in fixed intervals if we want precise time keeping. Note that error adjustments can still be done asynchronously as shown above, but dynamic interval ntp updates are just too complex and error-prone.

I could give it a try to remove all that extra complexity to get it into something mergable, but that would mean dropping quite some bits and I'd prefered if we could agree on something. I simply don't have the time right now to work on patches, which are rejected again in the end.

bye, Roman

—

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

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

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