

Re: [ANNOUNCE] high-res-timers patches for 2.6.6

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-06/3742.html>

From: George Anzinger (george_at_mvista.com)

Date: 06/17/04

Date: Wed, 16 Jun 2004 15:33:25 -0700
To: Mark Gross <mgross@linux.jf.intel.com>

Mark Gross wrote:

> *On Monday 14 June 2004 17:21, George Anzinger wrote:*

>

>> *Mark Gross wrote:*

>>

>>> *On Monday 14 June 2004 13:48, George Anzinger wrote:*

>>>

>>>> *Mark Gross wrote:*

>>>>

>>>>> *On Friday 11 June 2004 15:33, George Anzinger wrote:*

>>>>>

>>>>>> *I have been thinking of a major rewrite which would leave this code
>>>>>> alone, but would introduce an additional list and, of course, overhead
>>>>>> for high-res timers. This will take some time and be sub optimal, so I
>>>>>> wonder if it is needed.*

>>>>>

>>>>>> *What would your goal for the major rewrite be?*

>>>>>> *Redesign the implementation?*

>>>>>> *Clean up / re-factor the current design?*

>>>>>> *Add features?*

>>>>>

>>>>>> *Mostly I would like to make it "clean" enough to get the community to
>>>>>> accept it. As I look at the current implemtation, the biggest intrusion
>>>>>> into the "normal" kernel is in the timer list area. Thus, my thinking is
>>>>>> to introduce a second or slave list which would only be used by HR
>>>>>> timers. This list would be "checked" by putting a "normal" i.e.
>>>>>> add_timer, timer in place to mark the jiffie that a HR timer was to
>>>>>> expire in. The "check" code would then set up the HR interrupt to
>>>>>> expire the timer.*

>>>>>

>>>>>> *I am also considering removing a lot of the ifdefs one way or another.
>>>>>> AND, I think I can make the whole thing configureable at boot time just
>>>>>> as the pm/TSC/etc. timers are.*

>>>>>

>>>>>> *Sounds good to me. The higher level code can use this type of clean up.*

Linux-Kernel: Re: [ANNOUNCE] high-res-timers patches for 2.6.6

>>>
>>>
>>>>>I've been wondering lately if a significant restructuring of the
>>>>>implementation could be done. Something bottom's up that enabled
>>>>>changing / using different time bases without rebooting and coexisted
>>>>>nicey with HPET.
>>>>>
>>>>>Something along the lines of;
>>>>>* abstracting the time base's, calibration and computation of the next
>>>>>interrupt time into a polymorphic interface along with the
>>>>>implementation of a few of your time bases (ACPI, TSC) as a stand
>>>>>allown patch.
>>>>>
>>>>>Uh, is this something like the current TSC/ pmtimer/ HPET/ PIT selection
>>>>>code in the x86? Or do you have something else in mind here. Given the
>>>>>goal of integration with and inclusion in the kernel.org kernel, I don't
>>>>>>want to wander too far from what they are doing now.
>>>>>
>>>>>Sort of but implemented with a dynamic binding as opposed to the current
>>>>>compile time binding via ifdefs.
>>>>>
>>>>>The current HRT code implements a kind of static / compile time
>>>>>polymorphism that is hard for me to read and keep straight. It
>>>>>implements N time bases, with M interrupt sources for K architectures.
>>>>>Implementing the binding logic between all these at compile time leads to
>>>>>a lot of ifdefs and hard to grok code.
>>>>>
>>>>>What the 2.6 x86 timer code does is to "try" different clocks until they
>>>>>find one that "accepts" the job. A boot time option can override this to
>>>>>force a given clock, but all are compile in. (By the way, the first
>>>>>machine I wrote code for had a total of 8K bytes, so I really don't like to
>>>>>waste memory :))
>>>>>
>>>>>I started to do this with the latest patch but stopped when I realized that
>>>>>I would have to redo the conversion code. Still, this isn't too hard and I
>>>>>may finish this conversion. This would eliminate the pm/ TSC configure
>>>>>option AND allow the user to completely eliminate the HR (buy choosing a
>>>>>non-HR option at boot time), which, by the way, he can do now.
>>>>>
>>>>>
>>>>>Eliminating the PM/TSC configure option idea sounds good to me. Are you
>>>>>talking about the code your private codebase or the patch from SF for 2.6.5?

I started to to this in the 2.6.5 patch. Some of it is still there, in that you can, at boot time, switch to one of the standard clocks and thus disable HRT (or I think you can as I have not tested this).

>
> The code I'm looking at can't really do this, for the various time bases. It
> chooses between including the conversion code from hrtime-Macpi.h ^
> htrime-M586.h at compile time. There can be no "try" different clocks with

Linux-Kernel: Re: [ANNOUNCE] high-res-timers patches for 2.6.6

> *this linkage to the conversion code.*

Yes, I was looking at merging these two when I realized that the big thing they provide is different conversion routines. Lately, however, I have been thinking that we can use the same routines and just changed the conversion_bits.

>

> *get_jiffies_time macro, is statically linked to get_arch_cycles who's implementation id dependent on which hrtime-M*.h is included in at build time. Further the interpretation of "sub_left" is a function of the same build time configuration setting.*

I would like to handle this by using something like the timer get nanosecond stuff in the standard timers.

By the way, I have been talking to John Stultz about changing the code for gettimeofday to get nanoseconds and add that to xtime and THEN do the truncate to microseconds. This would change the timers call interface as well.

>

> *Grepping on USE_APIC_TIMERS shows more of the compile time ifdef code. It cannot use APIC clocks unless its compiled into the kernel there can be no "try" different timers WRT the APIC timer.*

I had not considered not using the APIC timer if it was available. Its availability is a compile time thing. It is used for both PM and TSC clocks and, I would assume, HPET when ever I get such a machine.

>

>

>>>>> ** implement yet another polymorphic interface for the interrupt source used by the patch, along with a few interrupt sources (PIT, APIC, HPET <-- new) * Implement a simple RTC-like charactor driver using the above for testing and integration.*

>>>>

>>>> *I am not sure what wants to be done here. I have to keep in mind that x86 is only one of many archs. I would like to keep it as simple as possible in this area. See the include/linux/hrttime.h file for the arch interface we are now using.*

>>>

>>> *yes but the code in the include/linux/hrttime.h file exports zero abstractions to the architecture independent kernel.*

>>>

>>> *Its mostly a documentation header file, that includes the architecture dependent exports, that then need to be used by the architecture independent code. Its all wrapped up in macros and what not to make it work across a handful of architectures but its still a significant CTAGS work out to follow the logic.*

>>>

>>> *I think that re-working the lower level HRT code to be more object based (like pci and net devices for example) with a layered design would significantly simplify the code and improve the extensibility across architectures and platform hardware time based interrupt sources.*

>>

Linux-Kernel: Re: [ANNOUNCE] high-res-timers patches for 2.6.6

>>I haven't looked at pci or net stuff lately, but my attempt to export the
>>conversion_bits structure was dissed by the arch folks, so I went for just
>>what was needed. Some of them don't export a conversion to micro seconds
>>conversion, for example. I welcome more details...
>>
>
>
>
> Details... That's a hard thing to come by when in a high level design
> discussion.
>
> It's too bad the conversion_bits export got shot down. Perhaps it was because
> you were exporting a data structure that made implicit assumptions rather
> than a more object based interface, with function pointers to conversion
> functions, and private data?

The functions, of course, were also exported. But this is exported from the arch side of things and not the base. They need to provide the conversion functions, the bits just being something that is needed if they export inline code, where as, if they export the functions, they don't need the bits (i.e. they are private). I rather like to export inline code as it is about twice as fast (I would guess).

>
> Regardless of doing an object based implementation of your design or not, if
> we could lose the #ifdefs and implicit ifdefs (i.e. IF_HIGH_RES) from the
> code (especially posix-timers.c) that would be really a good thing.
>
> I do still like the object based design concept ;)

I am afraid I am too old :(I rather think I understand object based code while not finding it very "warm". I have never written anything large that way and find myself objecting in the name of performance, but then, as I said, I may be too old.

George Anzinger 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

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

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