

Re: [RFC] killing the NR_IRQS arrays.

Re: [RFC] killing the NR_IRQS arrays.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg06126.html>

- *From:* ebiederm@xxxxxxxxxxxxx (Eric W. Biederman)
 - *Date:* Sat, 17 Feb 2007 01:51:33 -0700
-

Benjamin Herrenschmidt <benh@xxxxxxxxxxxxxxxxxxxxx> writes:

On Fri, 2007-02-16 at 05:10 -0700, Eric W. Biederman wrote:

Getting the drivers changed actually looks to be pretty straight forward it will just be a very large mechanical change. We change the type where of variables where appropriate and every once in a while introduce an `irq_nr(irq)` to get the actual irq number for the places that care (ISA or print statements).

Dunno about that `irq_nr` thingy. If we go that way, I'd be tempted to remove the number completely from the "public" side of `irq_desc...` or not.

When dealing with users and userspace for `/proc/interrupts` `/proc/irq` and the like we need a way to talk about irqs. Currently we use the interrupt number for that and we are likely to break the user/kernel interface if we don't preserve that. Debugging would tend to suck if we couldn't print out the irq number of the irq a driver has been assigned and trace it through various data structures.

For hardware that is not hotplug or auto discoverable I think we will need the irq number to talk about the ISA number as well.

So I don't see a way that we can get rid of a number completely but it should be of much less significance.

On powerpc, we have this remapped thingy because we completely separate the linux "virtual" interrupt domain from the physical numbering domains of each PIC. Your change would turn the linux virtual domain into pointers, removing the need for an array and associated limitations, which is nice.

So to a given `irq_desc / irq` "virtual" number today, I match a pair HW

Re: [RFC] killing the NR_IRQS arrays.

number (which is a special typedef which is currently defined as an unsigned long) and a pointer to the irq "host" (which is the entity that define a HW number domain).

That means that you can have multiple hosts and a given HW number can exist multiple times, once per host.

Do you think the irq_hwnumber_t thingy I have should then be generalized and put into the irq_desc ? I would need an additional void * pointer to the irq host as well (it's not a 1:1 relationship to an irq chip and need to be accessed by generic code).

Having taken a little bit of time to digest roughly what the concept is I think I can finally answer this one.

No. I don't think we should make your irq_hwnumber_t thingy general because it is not general. I don't understand why you need it to be an unsigned long, that still puzzles me. But for the rest it actually appears that ppc has a simpler model to deal with.

I don't think I actually can describe x86 hardware in you hwnumber_t world. Although I can approximate.

In non-legacy mode at the top of the tree I have a network cooperating irq controllers. For each cpu there is an lapic next to each cpu that catches interrupt packets and below that I have interrupt controllers that throw interrupt packets. In the network of cooperating interrupt controllers a interrupt packet has a destination address that looks like (cpu#, vector#) where cpu# is currently at 8 bits and slowly growing and the vector# is a fixed 8 bits.

The interrupt controllers that throw those packets have a fixed number of irq slots usually 24 or so. Each slot (referred to in the code as a pin) can be programmed which (cpu#, vector#) packet it throws when an interrupt occurs. Including an option to vary the cpu# between a set of cpus.

So to be frank to handle this model properly I need to deal with this properly I need.

```
#define NR_IRQS (NR_CPUS*256)
```

There is enough flexibility in this model that hardware vectors have not found a need to cascade interrupt controllers.

Having the HW number be clearly specific to a "domain controller" makes also a lot of sense in the embedded field with lots of cascaded interrupt controllers. It avoids having to play all sorts of tricks to assign ranges of numbers to various controllers in the system. Only the

Re: [RFC] killing the NR_IRQS arrays.

Re: [RFC] killing the NR_IRQS arrays.

local number on a given controller matters, the rest is dynamically assigned.

Ben I have no problem with a number that is specific to an irq controller for dealing with the internal irq controller implementations, heck I think everyone has that to some degree

The linux irq number will remain an arbitrary software number for use by the linux system for talking about the source of the interrupt.

Why in a sparse address space you would find it hard to allocate a range of numbers to an irq controller that only has a fixed number of irqs it can deal with is something I don't understand and I think it does a disservice to your users. But that is all it is a quality of implementation issue. ia64 does the same foolish thing.

The only time it really makes sense to me to let the irq number vary arbitrary are when things are truly dynamic, like with MSI, a hypervisor, or hot plug interrupt controllers.

Another option would be to have the irq_desc be created by the arch and "embedded" in a larger data structure, in which case the HW number would be part of the private part of that data structure. Though I suppose that could be a problem with ISA...

This definitely what I intend to have the gneirq code start allowing. For all intents and purposes we already do this today.

I suspect that for backward compatibility, we will need to keep something (optionally maybe via CONFIG_*) for ISA/legacy interrupts. That is a 16 entries irq_desc* array, so we can go from a legacy IRQ number to an irq_desc on platform that have legacy/ISA crap floating around.

Yes.

On powerpc, what I do is that I always reserve entries 0...15 of my remapping array in such a way that linux virtual irq 0 is always reserved, and 1...15 are only ever assigned to legacy interrupts if they exist in the system, or left unassigned if they don't.

Yep. Once we are done you can remove the reserve on 0. And leave

Re: [RFC] killing the NR_IRQS arrays.

Re: [RFC] killing the NR_IRQS arrays.

0..15 only ever assigned to ISA style interrupts if they are in the system.

I really don't like the term legacy or old/new, when referring to things. Because today's current hip/new is tomorrow legacy and we have lots of generations of hardware.

If we want to throw the legacy term around I hereby designate all non MSI-X interrupt controllers legacy.

I think we can make this change fairly smoothly if before the code is merged into Linus's tree we have a patchset prepared with a all of the core infrastructure changes and a best effort at all of the driver changes. Then early some merge window we merge the patchset, and fixup the drivers that were missed.

As long as we do things properly and not with a big "DESIGNED FOR x86" hack in the middle that makes it hard for everybody else, I agree.

Sure, and I have the same issue with a big "DESIGNED FOR ppc" in the middle, or "DESIGNED FOR arch/x". However the unfortunate truth is that the x86 has enough volume that frequently other architectures use some x86 hardware and thus get some of x86's warts. So anything that doesn't cope with the x86's warts is frequently doomed to failure.

Eric

-

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

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

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