Subject: Re: [PATCH] powerpc pseries eeh: Convert to kthread API
Posted by Benjamin Herrenschmid on Tue, 24 Apr 2007 02:42:24 GMT
View Forum Message <> Reply to Message

On Mon, 2007-04-23 at 20:08 -0600, Eric W. Biederman wrote:
> Benjamin Herrenschmidt <benh@kernel.crashing.org> writes:
>
> >> The only reason for using threads here is to get the error recovery
> >> out of an interrupt context (where errors may be detected), and then,
> >> an hour later, decrement a counter (which is how we limit these to
> >> 6 per hour). Thread reaping is "trivial", the thread just exits
> >> after an hour.
> >
> > In addition, it should be a thread and not done from within keventd
> > because :
> >
> >  - It can take a long time (well, relatively but still too long for a
> > work queue)
> >
> >  - The driver callbacks might need to use keventd or do flush_workqueue
> > to synchronize with their own workqueues when doing an internal
> > recovery.
> >
> >> Since these are events rare, I've no particular concern about
> >> performance or resource consumption. The current code seems
> >> to work just fine. :-)
> >
> > I think moving to kthread's is cleaner (just a wrapper around kernel
> > threads that simplify dealing with reaping them out mostly) and I agree
> > with Christoph that it would be nice to be able to "fire off" kthreads
> > from interrupt context.. in many cases, we abuse work queues for things
> > that should really done from kthreads instead (basically anything that
> > takes more than a couple hundred microsecs or so).
>
> On that note does anyone have a problem is we manage the irq spawning
> safe kthreads the same way that we manage the work queue entries.
>
> i.e. by a structure allocated by the caller?

Not sure... I can see places where I might want to spawn an arbitrary
number of these without having to preallocate structures... and if I
allocate on the fly, then I need a way to free that structure when the
kthread is reaped which I don't think we have currently, do we ? (In
fact, I could use that for other things too now that I'm thinking of
it ... I might have a go at providing optional kthread destructors).

Ben.