

Re: Request for feedback on common data backstore in the kernel

Re: Request for feedback on common data backstore in the kernel

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2007-09/msg00066.html>

- *From:* Hans Petter Selasky <hselasky@xxxxxxx>
 - *Date:* Fri, 28 Sep 2007 17:53:29 +0200
-

On Thursday 27 September 2007, Scott Long wrote:

Hans Petter Selasky wrote:

Hi Scott,

The discussion has been moved to "freebsd-arch@xxxxxxxxxxxx". Please only reply there next time.

On Wednesday 26 September 2007, Scott Long wrote:

Hans Petter Selasky wrote:

Hi,

Please keep me CC'ed, hence I'm not on all these lists.

In the kernel we currently have two different data backstores:

struct mbuf

and

struct buf

These two backstores serve two different device types. "mbufs" are for network devices and "buf" is for disk devices.

Problem:

The current backstores are loaded into DMA by using the BUS-DMA framework. This appears not to be too fast

Re: Request for feedback on common data backstore in the kernel

according to Kip Macy. See:

<http://perforce.freebsd.org/chv.cgi?CH=126455>

Busdma isn't fast enough for 1Gb and 10Gb network drivers that are trying to max out their packet rates. It's still fine for storage drivers and other slow/medium speed device drivers, like USB and Firewire. I am working on techniques to make the API easier to use and the implementation fast enough for the aforementioned devices.

That's great!

Well, the point is that I'm not sure why you're so worried about performance issues with USB and busdma. Do you have any test data that shows that it's a problem?

It is a problem on embedded devices. Typically not for an ordinary PC user.

Some ideas I have:

When a buffer is out of range for a hardware device and a data-copy is needed I want to simply copy that data in smaller parts to/from a pre-allocated bounce buffer. I want to avoid allocating this buffer when "bus_dmamap_load()" is called.

I think you've missed the point of having architecture portable drivers.
John-Mark describes this further.

I use the bus_dma framework to allocate and sync all DMA memory, and I assume that is correct.

That is one of the uses of busdma, yes. The other is to handle buffers that have been allocated elsewhere in the system.

Re: Request for feedback on common data backstore in the kernel

Re: Request for feedback on common data backstore in the kernel

Ok.

It also makes little sense to push the responsibility for handling bounce buffers out of a central subsystem and back into every driver.

I'm thinking about putting some wrappers around the "bus_dmamap_load()" function like:

```
void usbd_rx_buf_load(struct usbd_xfer *xfer, void *buf,  
uint32_t framebuffer_offset, uint32_t framebuffer_index, uint32_t len);
```

```
void usbd_tx_buf_load(struct usbd_xfer *xfer, void *buf,  
uint32_t framebuffer_offset, uint32_t framebuffer_index, uint32_t len);
```

But I haven't figured out all the details yet. The "usbd_XXX_load()" functions should automagically figure out what is best to do and it won't be more than a few lines of code.

Can you describe what these two wrappers are supposed to do? Are you expecting bus_dmamap_load() to operate synchronously?

I will come back to this question after the weekend, if you don't mind. Right now I'm in a hurry.

--HPS

freebsd-arch@xxxxxxxxxxxxx mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@xxxxxxxxxxxxx"