Subject: Re: [dm-devel] Re: dm: bounce_pfn limit added Posted by Kiyoshi Ueda on Wed, 31 Oct 2007 22:00:16 GMT

View Forum Message <> Reply to Message

Hi,

```
On Wed, 31 Oct 2007 08:36:01 +0100, Hannes Reinecke <hare@suse.de> wrote:
> Vasily Averin wrote:
> > Alasdair G Kergon wrote:
>>> So currently we treat bounce pfn as a property that does not need to be
>>> propagated through the stack.
> >>
> >> But is that the right approach?
>>> - Is there a blk_queue_bounce() missing either from dm or elsewhere?
>>> (And BTW can the bio_alloc() that lurks within lead to deadlock?)
> >>
>>> Firstly, what's going wrong?
>>> - What is the dm table you are using? (output of 'dmsetup table')
>>> - Which dm targets and with how many underlying devices?
>>> - Which underlying driver?
>>> - Is this direct I/O to the block device from userspace, or via some
> >> filesystem or what?
> >
>> On my testnode I have 6 Gb memory (1Gb normal zone for i386 kernels),
> > i2o hardware and lvm over i2o.
> >
> > [root@ts10 ~]# dmsetup table
> > vzvg-vz: 0 10289152 linear 80:5 384
> > vzvg-vzt: 0 263127040 linear 80:5 10289536
>> [root@ts10 ~]# cat /proc/partitions
> > major minor #blocks name
> >
>> 80
          0 143374336 i2o/hda
>> 80
              514048 i2o/hda1
>> 80
         2 4096575 i2o/hda2
>> 80
          3 2040255 i2o/hda3
>> 80
          4
                 1 i2o/hda4
>> 80
          5 136721151 i2o/hda5
>> 253
          0 5144576 dm-0
>> 253
          1 131563520 dm-1
> >
>> Diotest from LTP test suite with ~1Mb buffer size and files on dm-over-i2o
> > paritions corrupts i2o_iop0_msg_inpool slab.
> >
>> 120 on this node is able to handle only requests with up to 38 segments. Device
> > mapper correctly creates such requests and as you know it uses
>> max_pfn=BLK_BOUNCE_ANY. When this request translates to underlying device, it
> > clones bio and cleans BIO SEG VALID flag.
```

> >

- >> In this way underlying device calls blk recalc rg segments() to recount number
- > > of segments. However blk_recalc_rq_segments uses bounce_pfn=BLK_BOUNCE_HIGH
- > > taken from underlying device. As result number of segments become over than
- > > max_hw_segments limit.

> >

- >> Unfortunately there is not any checks and when i2o driver handles this incorrect
- > > request it fills the memory out of i2o_iop0_msg_inpool slab.

> >

- > We actually had a similar issue with some raid drivers (gdth iirc), and Neil Brown
- > did a similar patch for it. These were his comments on it:

> >

- > > dm handles max_hw_segments by using an 'io_restrictions' structure
- > > that keeps the most restrictive values from all component devices.

> >

> > So it should not allow more than max_hw_segments.

> >

- > > However I just notices that it does not preserve bounce_pfn as a restriction.
- >> So when the request gets down to the driver, it may be split up in to more
- > > segments than was expected up at the dm level.

> >

> So I guess we should take this.

How about the case that other dm device is stacked on the dm device? (e.g. dm-linear over dm-multipath over i2o with bounce_pfn=64GB, and the multipath table is changed to i2o with bounce_pfn=1GB.)

With this example, the patch propagates the restriction of i2o to dm-multipath but not to dm-linear.

So I guess the same problem happens.

Although it may sound like a corner case, such situation could occur with pymove of LVM2, for example.

I think we should take care of it too so that system won't be destroyed.

Rejecting to load such table will at least prevent the problem.

Thanks, Kiyoshi Ueda