Subject: dm: bounce_pfn limit added
Posted by vaverin on Mon, 29 Oct 2007 06:31:39 GMT
View Forum Message <> Reply to Message

Device mapper uses its own bounce_pfn that may differ from one on underlying
device. In that way dm can build incorrect requests that contain sg elements
greater than underlying device is able to handle.

This is the cause of slab corruption in i2o layer, occurred on i386 arch when
very long direct IO requests are addressed to dm-over-i2o device.

Signed-off-by: Vasily Averin <vvs@sw.ru>

```
--- a/drivers/md/dm-table.c
+++ b/drivers/md/dm-table.c
@@ -102,6 +102,8 @@ static void combine_restrictions_low(struct io_restrictions
  lhs->seg_boundary_mask =
   min_not_zero(lhs->seg_boundary_mask, rhs->seg_boundary_mask);

+ lhs->bounce_pfn = min_not_zero(lhs->bounce_pfn, rhs->bounce_pfn);
+
  lhs->no_cluster |= rhs->no_cluster;
 }

@@ -566,6 +568,8 @@ void dm_set_device_limits(struct dm_target *ti, struct
   min_not_zero(rs->seg_boundary_mask,
        q->seg_boundary_mask);

+ rs->bounce_pfn = min_not_zero(rs->bounce_pfn, q->bounce_pfn);
+
  rs->no_cluster |= !test_bit(QUEUE_FLAG_CLUSTER, &q->queue_flags);
 }
 EXPORT_SYMBOL_GPL(dm_set_device_limits);
@@ -707,6 +711,8 @@ static void check_for_valid_limits(struct io_restrictions
   rs->max_segment_size = MAX_SEGMENT_SIZE;
  if (!rs->seg_boundary_mask)
   rs->seg_boundary_mask = -1;
+ if (!rs->bounce_pfn)
+  rs->bounce_pfn = -1;
 }

 int dm_table_add_target(struct dm_table *t, const char *type,
@@ -891,6 +897,7 @@ void dm_table_set_restrictions(struct dm_table *t, struct
  q->hardsect_size = t->limits.hardsect_size;
  q->max_segment_size = t->limits.max_segment_size;
  q->seg_boundary_mask = t->limits.seg_boundary_mask;
+ q->bounce_pfn = t->limits.bounce_pfn;
  if (t->limits.no_cluster)
```

```
   q->queue_flags &= ~(1 << QUEUE_FLAG_CLUSTER);
  else
--- a/include/linux/device-mapper.h
+++ b/include/linux/device-mapper.h
@@ -111,6 +111,7 @@ struct target_type {

 struct io_restrictions {
  unsigned long  seg_boundary_mask;
+ unsigned long  bounce_pfn;
  unsigned int  max_sectors;
  unsigned int  max_segment_size;
  unsigned short  max_phys_segments;
```

Subject: Re: dm: bounce_pfn limit added
Posted by akpm on Tue, 30 Oct 2007 20:11:38 GMT
View Forum Message <> Reply to Message

On Mon, 29 Oct 2007 09:31:39 +0300
Vasily Averin <vvs@sw.ru> wrote:

> Device mapper uses its own bounce_pfn that may differ from one on underlying
> device. In that way dm can build incorrect requests that contain sg elements
> greater than underlying device is able to handle.
>
> This is the cause of slab corruption in i2o layer, occurred on i386 arch when
> very long direct IO requests are addressed to dm-over-i2o device.
>
> Signed-off-by: Vasily Averin <vvs@sw.ru>
>
> --- a/drivers/md/dm-table.c
> +++ b/drivers/md/dm-table.c
> @@ -102,6 +102,8 @@ static void combine_restrictions_low(struct io_restrictions
>   lhs->seg_boundary_mask =
>   min_not_zero(lhs->seg_boundary_mask, rhs->seg_boundary_mask);
>
> + lhs->bounce_pfn = min_not_zero(lhs->bounce_pfn, rhs->bounce_pfn);
> +
>   lhs->no_cluster |= rhs->no_cluster;
> }
>
> @@ -566,6 +568,8 @@ void dm_set_device_limits(struct dm_target *ti, struct
>   min_not_zero(rs->seg_boundary_mask,
>   q->seg_boundary_mask);
>
> + rs->bounce_pfn = min_not_zero(rs->bounce_pfn, q->bounce_pfn);
> +
>   rs->no_cluster |= !test_bit(QUEUE_FLAG_CLUSTER, &q->queue_flags);

```
>  }
>  EXPORT_SYMBOL_GPL(dm_set_device_limits);
> @@ -707,6 +711,8 @@ static void check_for_valid_limits(struct io_restrictions
>    rs->max_segment_size = MAX_SEGMENT_SIZE;
>   if (!rs->seg_boundary_mask)
>    rs->seg_boundary_mask = -1;
> + if (!rs->bounce_pfn)
> +  rs->bounce_pfn = -1;
>  }
>
>  int dm_table_add_target(struct dm_table *t, const char *type,
> @@ -891,6 +897,7 @@ void dm_table_set_restrictions(struct dm_table *t, struct
>   q->hardsect_size = t->limits.hardsect_size;
>   q->max_segment_size = t->limits.max_segment_size;
>   q->seg_boundary_mask = t->limits.seg_boundary_mask;
> + q->bounce_pfn = t->limits.bounce_pfn;
>   if (t->limits.no_cluster)
>    q->queue_flags &= ~(1 << QUEUE_FLAG_CLUSTER);
>   else
> --- a/include/linux/device-mapper.h
> +++ b/include/linux/device-mapper.h
> @@ -111,6 +111,7 @@ struct target_type {
>
>  struct io_restrictions {
>   unsigned long  seg_boundary_mask;
> + unsigned long  bounce_pfn;
>   unsigned int  max_sectors;
>   unsigned int  max_segment_size;
>   unsigned short  max_phys_segments;
```

Well that's a rather grave sounding bug. Two days and nobody from DM land has commented? Hello?

I'll tag this as needed in 2.6.23.x as well.

I'll duck the "dm: struct io_restriction reordered" patch. People have been changing things around in there and I had to fix a reject in "dm: bounce_pfn limit added" to make it apply - let's not complicate life.

However it is a good change and hopefully the DM people will pick it up.

Subject: Re: [dm-devel] Re: dm: bounce_pfn limit added
Posted by Alasdair G Kergon on Tue, 30 Oct 2007 23:26:17 GMT
View Forum Message <> Reply to Message

On Tue, Oct 30, 2007 at 01:11:38PM -0700, Andrew Morton wrote:
> On Mon, 29 Oct 2007 09:31:39 +0300

> Vasily Averin <vvs@sw.ru> wrote:
> > Device mapper uses its own bounce_pfn that may differ from one on underlying
> > device. In that way dm can build incorrect requests that contain sg elements
> > greater than underlying device is able to handle.
> > This is the cause of slab corruption in i2o layer, occurred on i386 arch when
> > very long direct IO requests are addressed to dm-over-i2o device.

Interesting. When this patch was originally proposed a couple of years ago, we
did this instead:

http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=daef265f1590cf3e6de9
89d074041a280c82d58b;hp=00d6da9b4d6707b808481372537adb0fb38f99b3

) DM doesn't need to bounce bio's on its own, but the block layer defaults
) to that in blk_queue_make_request(). The lower level drivers should
) bounce ios themselves, that is what they need to do if not layered below
) dm anyways.

Alasdair
--
agk@redhat.com

Subject: Re: [dm-devel] Re: dm: bounce_pfn limit added
Posted by Alasdair G Kergon on Wed, 31 Oct 2007 02:01:33 GMT
View Forum Message <> Reply to Message

On Tue, Oct 30, 2007 at 11:26:17PM +0000, Alasdair G Kergon wrote:
>) DM doesn't need to bounce bio's on its own, but the block layer defaults
>) to that in blk_queue_make_request(). The lower level drivers should
>) bounce ios themselves, that is what they need to do if not layered below
>) dm anyways.

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?

Presumably you've shown the problem goes away when the dm device is

removed from the stack.
What if you swap in alternative dm targets, e.g. if it's linear,
try multipath (round-robin, one path)?

Alasdair
--
agk@redhat.com

Subject: Re: [dm-devel] Re: dm: bounce_pfn limit added
Posted by Alasdair G Kergon on Wed, 31 Oct 2007 02:11:23 GMT
View Forum Message <> Reply to Message

On Wed, Oct 31, 2007 at 02:01:33AM +0000, Alasdair G Kergon wrote:
> What if you swap in alternative dm targets, e.g. if it's linear,
> try multipath (round-robin, one path)?

And try using md instead of dm - does that also show the problem?
(md takes a similar stance to dm on this I believe.)

Alasdair
--
agk@redhat.com

Subject: Re: [dm-devel] Re: dm: bounce_pfn limit added
Posted by vaverin on Wed, 31 Oct 2007 07:13:33 GMT
View Forum Message <> Reply to Message

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   1   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.

I2o 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_rq_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.

Thank you,
 Vasily Averin

Subject: Re: [dm-devel] Re: dm: bounce_pfn limit added
Posted by Hannes Reinecke on Wed, 31 Oct 2007 07:36:01 GMT
View Forum Message <> Reply to Message

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   1   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.
>
> I2o 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_rq_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.

Cheers,

Hannes
--
Dr. Hannes Reinecke zSeries & Storage
hare@suse.de +49 911 74053 688

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   1   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.
> >
> > I2o 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_rq_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 pvmove 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
