Subject: [PATCH] block: blk_max_pfn is somtimes wrong
Posted by Vasily Tarasov on Thu, 08 Feb 2007 12:39:18 GMT
View Forum Message <> Reply to Message

There is a small problem in handling page bounce.

At the moment blk_max_pfn equals max_pfn, which is in fact
not maximum possible _number_ of a page frame, but the _amount_
of page frames. For example for the 32bit x86 node with 4Gb RAM,
max_pfn = 0x100000, but not 0xFFFF.

request_queue structure has a member q->bounce_pfn and queue needs
bounce pages for the pages _above_ this limit. This routine is handled
by blk_queue_bounce(), where the following check is produced:

```
 if (q->bounce_pfn >= blk_max_pfn)
  return;
```

Assume, that a driver has set q->bounce_pfn to 0xFFFF, but
blk_max_pfn equals 0x10000. In such situation the check above
fails and for each bio we always fall down for iterating over
pages tied to the bio.

I want to notice, that for quite a big range of device drivers
(ide, md, ...) such problem doesn't happen because they use
BLK_BOUNCE_ANY for bounce_pfn. BLK_BOUNCE_ANY is defined as
blk_max_pfn << PAGE_SHIFT, and then the check above doesn't fail.
But for other drivers, which obtain reuired value from drivers,
it fails. For example sata_nv uses ATA_DMA_MASK or dev->dma_mask.

I propose to use (max_pfn - 1) for blk_max_pfn. And the
same for blk_max_low_pfn. The patch also cleanses some checks
related with bounce_pfn.

Signed-off-by: Vasily Tarasov <vtaras@openvz.org>

```
--- ./block/ll_rw_blk.c.max_pfn 2007-01-10 03:35:11.000000000 +0300
+++ ./block/ll_rw_blk.c 2007-02-08 14:42:48.000000000 +0300
@@ -1221,7 +1221,7 @@ void blk_recount_segments(request_queue_
    * considered part of another segment, since that might
    * change with the bounce page.
    */
-  high = page_to_pfn(bv->bv_page) >= q->bounce_pfn;
+  high = page_to_pfn(bv->bv_page) > q->bounce_pfn;
   if (high || highprv)
    goto new_hw_segment;
   if (cluster) {
@@ -3658,8 +3658,8 @@ int __init blk_dev_init(void)
  open_softirq(BLOCK_SOFTIRQ, blk_done_softirq, NULL);
  register_hotcpu_notifier(&blk_cpu_notifier);

- blk_max_low_pfn = max_low_pfn;
- blk_max_pfn = max_pfn;
+ blk_max_low_pfn = max_low_pfn - 1;
+ blk_max_pfn = max_pfn - 1;

  return 0;
 }
--- ./mm/bounce.c.max_pfn 2006-11-30 00:57:37.000000000 +0300
+++ ./mm/bounce.c 2007-02-08 14:49:35.000000000 +0300
@@ -204,7 +204,7 @@ static void __blk_queue_bounce(request_q
   /*
    * is destination page below bounce pfn?
    */
-  if (page_to_pfn(page) < q->bounce_pfn)
+  if (page_to_pfn(page) <= q->bounce_pfn)
    continue;

   /*
```

Subject: Re: [PATCH] block: blk_max_pfn is somtimes wrong
Posted by Jens Axboe on Fri, 09 Feb 2007 17:28:05 GMT
View Forum Message <> Reply to Message

On Thu, Feb 08 2007, Vasily Tarasov wrote:
> There is a small problem in handling page bounce.
>
> At the moment blk_max_pfn equals max_pfn, which is in fact
> not maximum possible _number_ of a page frame, but the _amount_
> of page frames. For example for the 32bit x86 node with 4Gb RAM,
> max_pfn = 0x100000, but not 0xFFFF.
>
> request_queue structure has a member q->bounce_pfn and queue needs
> bounce pages for the pages _above_ this limit. This routine is handled
> by blk_queue_bounce(), where the following check is produced:
>
> if (q->bounce_pfn >= blk_max_pfn)
>   return;
>
> Assume, that a driver has set q->bounce_pfn to 0xFFFF, but
> blk_max_pfn equals 0x10000. In such situation the check above
> fails and for each bio we always fall down for iterating over
> pages tied to the bio.

>
> I want to notice, that for quite a big range of device drivers
> (ide, md, ...) such problem doesn't happen because they use
> BLK_BOUNCE_ANY for bounce_pfn. BLK_BOUNCE_ANY is defined as
> blk_max_pfn << PAGE_SHIFT, and then the check above doesn't fail.
> But for other drivers, which obtain reuired value from drivers,
> it fails. For example sata_nv uses ATA_DMA_MASK or dev->dma_mask.
>
> I propose to use (max_pfn - 1) for blk_max_pfn. And the
> same for blk_max_low_pfn. The patch also cleanses some checks
> related with bounce_pfn.
>
> Signed-off-by: Vasily Tarasov <vtaras@openvz.org>

I will add that this is also a performance optimization, as it was
initially discovered by Vasily because he saw bounces issues with
blktrace. The blktrace notify was done early when we iterated the bio
segments to check for bounces, no bounces were actually issued (this
problem was fixed for 2.6.20 by moving the notify in mm/bounce.c to when
we actually bounce). So it does cause needless bio segment iterations on
some setups, because of this one-off.

Acked-by: Jens Axboe <jens.axboe@oracle.com>

--
Jens Axboe