Subject: seems to be a flaw in cfq
Posted by Vasily Tarasov on Tue, 19 Dec 2006 13:53:04 GMT
View Forum Message <> Reply to Message

ello, Jens.

Seems, that we've found some problem in CFQ.
I used your fio tool of version 1.9 to reproduce it.
2.6.18 vanilla kernel.
Generally situation is the following:
there are several readers, which read from their own files, except
two readers, which read one shared file.
These two readers experienced huge starvation!

I want to note, that there is no such situation on deadline scheduler
and situation is much better for anticipatory scheduler.

Here is a job file:

<snip>
; cfq bug reproduce
; each file is 1Gib size

[global]
timeout=200

[reader1]
filename=file1

[reader2]
filename=file2

[reader3]
filename=file3

; --- this two reads the same file ---
[reader4]
filename=file4

[reader5]
filename=file4
; -----------------------------------

[reader6]
filename=file6

[reader7]
filename=file7

[reader8]
filename=file8

[reader9]
filename=file9

[reader10]
filename=file10

[reader11]
filename=file11
</snip>

Here is an output of fio:
fio job1.file
reader1: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader2: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader3: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader4: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader5: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader6: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader7: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader8: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader9: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader10: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
reader11: (g=0): rw=read, odir=1, bs=4K-4K/4K-4K, rate=0, ioengine=sync,
iodepth=1
Starting 11 threads
Threads running: 11: [RRRRRRRRRRR] [100.00% done] [31298/ 0 kb/s]
[eta 00m:00s]
reader1: (groupid=0): err= 0:
read : io= 662MiB, bw= 3469KiB/s, runt=200058msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1362, avg= 1.06, dev=30.90

bw (KiB/s) : min= 490, max=12582, per=12.03%, avg=3732.10, dev=4034.82
cpu : usr=0.15%, sys=4.43%, ctx=169761
reader2: (groupid=0): err= 0:
read : io= 644MiB, bw= 3375KiB/s, runt=200041msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1454, avg= 1.09, dev=31.69
bw (KiB/s) : min= 485, max=12558, per=11.59%, avg=3595.80, dev=3795.64
cpu : usr=0.15%, sys=4.36%, ctx=165149
reader3: (groupid=0): err= 0:
read : io= 679MiB, bw= 3561KiB/s, runt=200057msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1584, avg= 1.03, dev=30.33
bw (KiB/s) : min= 445, max=12689, per=12.53%, avg=3886.43, dev=4161.05
cpu : usr=0.15%, sys=4.63%, ctx=174219
reader4: (groupid=0): err= 0:
read : io= 1MiB, bw= 9KiB/s, runt=200009msec
<<< ONLY ONE MIB
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1327, avg=411.34, dev=597.62
bw (KiB/s) : min= 3, max= 24, per=0.03%, avg= 9.92, dev=11.60
cpu : usr=0.00%, sys=0.01%, ctx=501
reader5: (groupid=0): err= 0:
read : io= 1MiB, bw= 9KiB/s, runt=200009msec
<<< ONLY ONE MIB
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1327, avg=413.70, dev=599.16
bw (KiB/s) : min= 3, max= 24, per=0.03%, avg= 9.67, dev=11.18
cpu : usr=0.00%, sys=0.01%, ctx=491
reader6: (groupid=0): err= 0:
read : io= 661MiB, bw= 3466KiB/s, runt=200045msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1453, avg= 1.06, dev=30.92
bw (KiB/s) : min= 483, max=12222, per=12.14%, avg=3765.32, dev=4002.38
cpu : usr=0.14%, sys=4.51%, ctx=169566
reader7: (groupid=0): err= 0:
read : io= 655MiB, bw= 3435KiB/s, runt=200056msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1585, avg= 1.07, dev=31.36
bw (KiB/s) : min= 602, max=12115, per=11.78%, avg=3654.40, dev=3858.70
cpu : usr=0.14%, sys=4.48%, ctx=168098
reader8: (groupid=0): err= 0:
read : io= 666MiB, bw= 3496KiB/s, runt=200000msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1358, avg= 1.05, dev=30.95
bw (KiB/s) : min= 7, max=12730, per=12.23%, avg=3791.85, dev=4177.52
cpu : usr=0.15%, sys=4.53%, ctx=170999
reader9: (groupid=0): err= 0:
read : io= 666MiB, bw= 3495KiB/s, runt=200016msec

slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1456, avg= 1.05, dev=30.78
bw (KiB/s) : min= 6, max=12681, per=12.13%, avg=3761.05, dev=4038.50
cpu : usr=0.17%, sys=4.52%, ctx=170982
reader10: (groupid=0): err= 0:
read : io= 648MiB, bw= 3398KiB/s, runt=200026msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1356, avg= 1.09, dev=31.57
bw (KiB/s) : min= 5, max=12533, per=11.79%, avg=3657.35, dev=3930.12
cpu : usr=0.15%, sys=4.35%, ctx=166233
reader11: (groupid=0): err= 0:
read : io= 628MiB, bw= 3296KiB/s, runt=200049msec
slat (msec): min= 0, max= 0, avg= 0.00, dev= 0.00
clat (msec): min= 0, max= 1591, avg= 1.12, dev=32.34
bw (KiB/s) : min= 3, max=13910, per=11.59%, avg=3595.44, dev=4000.10
cpu : usr=0.16%, sys=4.27%, ctx=161300

Run status group 0 (all jobs):
READ: io=5917MiB, aggrb=31013, minb=9, maxb=3561, mint=200000msec,
maxt=200058msec

Disk stats (read/write):
sda: ios=1516020/65, merge=242/28, ticks=1975258/49498,
in_queue=2024681, util=100.00%

Is such behavior expected?

Thanks in advance,
Vasily.

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Tue, 19 Dec 2006 14:37:55 GMT
View Forum Message <> Reply to Message

On Tue, Dec 19 2006, Vasily Tarasov wrote:
> ello, Jens.
>
> Seems, that we've found some problem in CFQ.
> I used your fio tool of version 1.9 to reproduce it.
> 2.6.18 vanilla kernel.

I'll look over this report - in the mean time, can you see if the same
situation exists in 2.6.19 and 2.6.20-rc1? Would help a lot!

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Tue, 19 Dec 2006 14:54:51 GMT
View Forum Message <> Reply to Message

On Tue, Dec 19 2006, Jens Axboe wrote:
> On Tue, Dec 19 2006, Vasily Tarasov wrote:
> > ello, Jens.
> >
> > Seems, that we've found some problem in CFQ.
> > I used your fio tool of version 1.9 to reproduce it.
> > 2.6.18 vanilla kernel.
>
> I'll look over this report - in the mean time, can you see if the same
> situation exists in 2.6.19 and 2.6.20-rc1? Would help a lot!

I just tried to reproduce it in a recent kernel, and it does show
something very close to what you reported. Using 5 threads, 2 of them
sharing the same file, the 3 first threads get 6067 -> 6182KiB/sec each,
and the two threads sharing a file get 975KiB/sec each. That's really
bizarre, I'll take a good look at this! Thanks for reporting it.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Tue, 19 Dec 2006 14:59:37 GMT
View Forum Message <> Reply to Message

On Tue, Dec 19 2006, Jens Axboe wrote:
> On Tue, Dec 19 2006, Jens Axboe wrote:
> > On Tue, Dec 19 2006, Vasily Tarasov wrote:
> > > ello, Jens.
> > >
> > > Seems, that we've found some problem in CFQ.
> > > I used your fio tool of version 1.9 to reproduce it.
> > > 2.6.18 vanilla kernel.
> >
> > I'll look over this report - in the mean time, can you see if the same
> > situation exists in 2.6.19 and 2.6.20-rc1? Would help a lot!
>
> I just tried to reproduce it in a recent kernel, and it does show
> something very close to what you reported. Using 5 threads, 2 of them
> sharing the same file, the 3 first threads get 6067 -> 6182KiB/sec each,
> and the two threads sharing a file get 975KiB/sec each. That's really
> bizarre, I'll take a good look at this! Thanks for reporting it.

Oh, thinking about this - it could be an artifact of being too fair. The

default is to use O_DIRECT, so the sharing threads probably end up being
blocked waiting for each other to finish the same blocks of io. They
will both be reading from the start of the file to the end, so if they
run alongside each other inside the file, they'll be blocking each other
waiting for io to finish. The same should happen for AS, though. Perhaps
it's bad alias handling in CFQ.

If I add offset=512m to the last thread so that the two sharing threads
read different parts of the file, the result is completely fair
(4.9MiB/sec -> 5.1MiB/sec for each thread).

I'll keep looking.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Tue, 19 Dec 2006 18:47:48 GMT
View Forum Message <> Reply to Message

On Tue, Dec 19 2006, Jens Axboe wrote:
> On Tue, Dec 19 2006, Jens Axboe wrote:
> > On Tue, Dec 19 2006, Jens Axboe wrote:
> > > On Tue, Dec 19 2006, Vasily Tarasov wrote:
> > > > ello, Jens.
> > > >
> > > > Seems, that we've found some problem in CFQ.
> > > > I used your fio tool of version 1.9 to reproduce it.
> > > > 2.6.18 vanilla kernel.
> > >
> > > I'll look over this report - in the mean time, can you see if the same
> > > situation exists in 2.6.19 and 2.6.20-rc1? Would help a lot!
> >
> > I just tried to reproduce it in a recent kernel, and it does show
> > something very close to what you reported. Using 5 threads, 2 of them
> > sharing the same file, the 3 first threads get 6067 -> 6182KiB/sec each,
> > and the two threads sharing a file get 975KiB/sec each. That's really
> > bizarre, I'll take a good look at this! Thanks for reporting it.
>
> Oh, thinking about this - it could be an artifact of being too fair. The
> default is to use O_DIRECT, so the sharing threads probably end up being
> blocked waiting for each other to finish the same blocks of io. They
> will both be reading from the start of the file to the end, so if they
> run alongside each other inside the file, they'll be blocking each other
> waiting for io to finish. The same should happen for AS, though. Perhaps
> it's bad alias handling in CFQ.
>

> If I add offset=512m to the last thread so that the two sharing threads
> read different parts of the file, the result is completely fair
> (4.9MiB/sec -> 5.1MiB/sec for each thread).
>
> I'll keep looking.

Back after dinner, the fresh energy served it's purpose - I think I know
what the issue is. We allow merging across process queues, which will
effectively serialize some io if they are sync (like this case). I'll
hack up a fix for current git and give it a test spin, to verify that
this is the problem here.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Vasily Tarasov on Wed, 20 Dec 2006 07:09:57 GMT
View Forum Message <> Reply to Message

Hello, Jens.

Jens Axboe wrote:
>
> Back after dinner, the fresh energy served it's purpose - I think I know
> what the issue is. We allow merging across process queues, which will
> effectively serialize some io if they are sync (like this case). I'll
> hack up a fix for current git and give it a test spin, to verify that
> this is the problem here.
>
As far as I understand merging is an often event in this situation only
because of sequential
reading, Below is a job-file with rw=randread and direct=false. However
we got the same bizarre results.

Job-file:

; cfq bug reproduce

[global]
rw=randread
direct=0

timeout=200

[reader1]
filename=file1

```
[reader2]
filename=file2

[reader3]
filename=file3

; --- this two reads the same file ---
[reader4]
filename=file4

[reader5]
filename=file4
; -----------------------------------

[reader6]
filename=file6

[reader7]
filename=file7


[reader8]
filename=file8


[reader9]
filename=file9

[reader10]
filename=file10

[reader11]
filename=file11
```

Results:

```
# fio job1.file
reader1: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader2: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader3: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader4: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader5: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader6: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
```

```
ioengine=sync, iodepth=1
reader7: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader8: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader9: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader10: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
reader11: (g=0): rw=randread, odir=0, bs=4K-4K/4K-4K, rate=0,
ioengine=sync, iodepth=1
Starting 11 threads
Threads running: 11: [rrrrrrrrrrr] [100.00% done] [   528/     0 kb/s]
[eta 00m:00s]
reader1: (groupid=0): err= 0:
  read : io=    10MiB, bw=    53KiB/s, runt=200006msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max=  988, avg=75.54, dev=244.04
    bw (KiB/s) : min=   25, max=   70, per=10.87%, avg=53.18, dev=54.17
  cpu          : usr=0.00%, sys=0.07%, ctx=2888
reader2: (groupid=0): err= 0:
  read : io=    10MiB, bw=    54KiB/s, runt=200067msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max= 1034, avg=74.57, dev=241.93
    bw (KiB/s) : min=   29, max=   90, per=11.04%, avg=54.01, dev=55.09
  cpu          : usr=0.00%, sys=0.07%, ctx=2925
reader3: (groupid=0): err= 0:
  read : io=    10MiB, bw=    53KiB/s, runt=200039msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max=  965, avg=75.64, dev=243.64
    bw (KiB/s) : min=   29, max=   86, per=10.91%, avg=53.34, dev=54.48
  cpu          : usr=0.00%, sys=0.07%, ctx=2885
reader4: (groupid=0): err= 0:
  read : io=     1MiB, bw=     5KiB/s, runt=200049msec
                 <<<<<
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    3, max= 1705, avg=716.52, dev=894.16
    bw (KiB/s) : min=    2, max=   35, per=1.14%, avg= 5.59, dev= 7.47
  cpu          : usr=0.00%, sys=0.00%, ctx=369
reader5: (groupid=0): err= 0:
  read : io=     1MiB, bw=     5KiB/s, runt=200049msec
               <<<<<<
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    3, max= 1705, avg=716.52, dev=894.17
    bw (KiB/s) : min=    2, max=   35, per=1.14%, avg= 5.59, dev= 7.47
  cpu          : usr=0.00%, sys=0.00%, ctx=366
reader6: (groupid=0): err= 0:
  read : io=    10MiB, bw=    52KiB/s, runt=200055msec
```

```
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    1, max=  969, avg=77.21, dev=246.03
    bw (KiB/s) : min=   22, max=   97, per=10.74%, avg=52.50, dev=53.67
  cpu          : usr=0.00%, sys=0.08%, ctx=2832
reader7: (groupid=0): err= 0:
  read : io=    10MiB, bw=    53KiB/s, runt=200011msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max= 1047, avg=76.68, dev=245.52
    bw (KiB/s) : min=   18, max=   77, per=10.76%, avg=52.60, dev=53.61
  cpu          : usr=0.00%, sys=0.09%, ctx=2852
reader8: (groupid=0): err= 0:
  read : io=    10MiB, bw=    53KiB/s, runt=200072msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max= 1033, avg=76.68, dev=245.69
    bw (KiB/s) : min=    7, max=   71, per=10.71%, avg=52.37, dev=53.40
  cpu          : usr=0.00%, sys=0.08%, ctx=2851
reader9: (groupid=0): err= 0:
  read : io=    10MiB, bw=    52KiB/s, runt=200022msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max=  973, avg=76.90, dev=245.66
    bw (KiB/s) : min=    5, max=  188, per=10.76%, avg=52.63, dev=54.71
  cpu          : usr=0.00%, sys=0.06%, ctx=2841
reader10: (groupid=0): err= 0:
  read : io=     9MiB, bw=    52KiB/s, runt=200028msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max=  972, avg=78.06, dev=247.65
    bw (KiB/s) : min=    5, max=  106, per=10.58%, avg=51.72, dev=53.05
  cpu          : usr=0.00%, sys=0.09%, ctx=2807
reader11: (groupid=0): err= 0:
  read : io=    10MiB, bw=    52KiB/s, runt=200084msec
    slat (msec): min=    0, max=    0, avg= 0.00, dev= 0.00
    clat (msec): min=    0, max=  979, avg=77.57, dev=247.05
    bw (KiB/s) : min=    4, max=   73, per=10.60%, avg=51.82, dev=52.90
  cpu          : usr=0.00%, sys=0.07%, ctx=2822

Run status group 0 (all jobs):
   READ: io=93MiB, aggrb=489, minb=5, maxb=54, mint=200006msec,
maxt=200084msec

Disk stats (read/write):
  sda: ios=26139/56, merge=0/20, ticks=2000184/30587, in_queue=2030770,
util=100.00%
```

Thank you,
Vasily.

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Wed, 20 Dec 2006 09:33:45 GMT
View Forum Message <> Reply to Message

On Wed, Dec 20 2006, Vasily Tarasov wrote:
> Hello, Jens.
>
> Jens Axboe wrote:
> >
> > Back after dinner, the fresh energy served it's purpose - I think I know
> > what the issue is. We allow merging across process queues, which will
> > effectively serialize some io if they are sync (like this case). I'll
> > hack up a fix for current git and give it a test spin, to verify that
> > this is the problem here.
> >
> As far as I understand merging is an often event in this situation only
> because of sequential
> reading, Below is a job-file with rw=randread and direct=false. However
> we got the same bizarre results.

Can you repeat with the below patch applied? It's against 2.6.20-rc1
(ish), so you'll need to update to that kernel first. Best would be to
upgrade to current git for testing purposes. Then verify that you are
still seeing the unfair results, apply the patch, then try and
reproduce.

The patch fixed the scenario for me. I'll retry with the random buffered
read load you describe here.

```
diff --git a/block/cfq-iosched.c b/block/cfq-iosched.c
index 533a293..9fc5eaf 100644
--- a/block/cfq-iosched.c
+++ b/block/cfq-iosched.c
@@ -568,6 +568,38 @@ cfq_merged_requests(request_queue_t *q, struct request *rq,
  cfq_remove_request(next);
 }

+static int cfq_allow_merge(request_queue_t *q, struct request *rq,
+       struct bio *bio)
+{
+ struct cfq_data *cfqd = q->elevator->elevator_data;
+ const int rw = bio_data_dir(bio);
+ struct cfq_queue *cfqq;
+ pid_t key;
+
+ /*
+  * If bio is async or a write, always allow merge
+  */
+ if (!bio_sync(bio) || rw == WRITE)
```

```
+  return 1;
+
+ /*
+  * bio is sync. if request is not, disallow.
+  */
+ if (!rq_is_sync(rq))
+  return 0;
+
+ /*
+  * Ok, both bio and request are sync. Allow merge if they are
+  * from the same queue.
+  */
+ key = cfq_queue_pid(current, rw, 1);
+ cfqq = cfq_find_cfq_hash(cfqd, key, current->ioprio);
+ if (cfqq != RQ_CFQQ(rq))
+  return 0;
+
+ return 1;
+}
+
 static inline void
 __cfq_set_active_queue(struct cfq_data *cfqd, struct cfq_queue *cfqq)
 {
@@ -2125,6 +2157,7 @@ static struct elevator_type iosched_cfq = {
   .elevator_merge_fn =   cfq_merge,
   .elevator_merged_fn =  cfq_merged_request,
   .elevator_merge_req_fn = cfq_merged_requests,
+  .elevator_allow_merge_fn = cfq_allow_merge,
   .elevator_dispatch_fn =  cfq_dispatch_requests,
   .elevator_add_req_fn =  cfq_insert_request,
   .elevator_activate_req_fn = cfq_activate_request,
diff --git a/block/elevator.c b/block/elevator.c
index c0063f3..62c7a30 100644
--- a/block/elevator.c
+++ b/block/elevator.c
@@ -51,6 +51,21 @@ static const int elv_hash_shift = 6;
 #define ELV_ON_HASH(rq)  (!hlist_unhashed(&(rq)->hash))

 /*
+ * Query io scheduler to see if the current process issuing bio may be
+ * merged with rq.
+ */
+static int elv_iosched_allow_merge(struct request *rq, struct bio *bio)
+{
+ request_queue_t *q = rq->q;
+ elevator_t *e = q->elevator;
+
+ if (e->ops->elevator_allow_merge_fn)
```

```
+  return e->ops->elevator_allow_merge_fn(q, rq, bio);
+
+ return 1;
+}
+
+/*
  * can we safely merge with this request?
  */
 inline int elv_rq_merge_ok(struct request *rq, struct bio *bio)
@@ -65,12 +80,15 @@ inline int elv_rq_merge_ok(struct request *rq, struct bio *bio)
   return 0;

  /*
-  * same device and no special stuff set, merge is ok
+  * must be same device and not a special request
   */
- if (rq->rq_disk == bio->bi_bdev->bd_disk && !rq->special)
-  return 1;
+ if (rq->rq_disk != bio->bi_bdev->bd_disk || !rq->special)
+  return 0;

- return 0;
+ if (!elv_iosched_allow_merge(rq, bio))
+  return 0;
+
+ return 1;
 }
 EXPORT_SYMBOL(elv_rq_merge_ok);

diff --git a/include/linux/elevator.h b/include/linux/elevator.h
index a24931d..e88fcbc 100644
--- a/include/linux/elevator.h
+++ b/include/linux/elevator.h
@@ -12,6 +12,8 @@ typedef void (elevator_merge_req_fn) (request_queue_t *, struct request *,
struc

 typedef void (elevator_merged_fn) (request_queue_t *, struct request *, int);

+typedef int (elevator_allow_merge_fn) (request_queue_t *, struct request *, struct bio *);
+
 typedef int (elevator_dispatch_fn) (request_queue_t *, int);

 typedef void (elevator_add_req_fn) (request_queue_t *, struct request *);
@@ -33,6 +35,7 @@ struct elevator_ops
  elevator_merge_fn *elevator_merge_fn;
  elevator_merged_fn *elevator_merged_fn;
  elevator_merge_req_fn *elevator_merge_req_fn;
+ elevator_allow_merge_fn *elevator_allow_merge_fn;
```

elevator_dispatch_fn *elevator_dispatch_fn;
elevator_add_req_fn *elevator_add_req_fn;

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Wed, 20 Dec 2006 09:35:39 GMT
View Forum Message <> Reply to Message

On Wed, Dec 20 2006, Jens Axboe wrote:
> On Wed, Dec 20 2006, Vasily Tarasov wrote:
> > Hello, Jens.
> >
> > Jens Axboe wrote:
> > >
> > > Back after dinner, the fresh energy served it's purpose - I think I know
> > > what the issue is. We allow merging across process queues, which will
> > > effectively serialize some io if they are sync (like this case). I'll
> > > hack up a fix for current git and give it a test spin, to verify that
> > > this is the problem here.
> > >
> > As far as I understand merging is an often event in this situation only
> > because of sequential
> > reading, Below is a job-file with rw=randread and direct=false. However
> > we got the same bizarre results.
>
> Can you repeat with the below patch applied? It's against 2.6.20-rc1
> (ish), so you'll need to update to that kernel first. Best would be to
> upgrade to current git for testing purposes. Then verify that you are
> still seeing the unfair results, apply the patch, then try and
> reproduce.
>
> The patch fixed the scenario for me. I'll retry with the random buffered
> read load you describe here.

I'm still seeing odd results with buffered io, while the direct case is
definitely fixed. So we are either looking at two bugs, or one bug that
the merge fixup hides.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq

Posted by Jens Axboe on Wed, 20 Dec 2006 09:38:55 GMT
View Forum Message <> Reply to Message

On Wed, Dec 20 2006, Jens Axboe wrote:
> I'm still seeing odd results with buffered io, while the direct case is
> definitely fixed. So we are either looking at two bugs, or one bug that
> the merge fixup hides.

Ah hang on, buffered reads will be serialized by the page cache. If I
rerun the buffered randread case with anticipatory, the results are
skewed in the other direction but still unfair. So I'm not sure there's
a real bug there in the io scheduler.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Vasily Tarasov on Thu, 21 Dec 2006 09:03:20 GMT
View Forum Message <> Reply to Message

Hello, Jens,

Sorry for late answer.
The situation is the following:

direct seqread
(2.6.18) - reproducable
(2.6.20-rc1) - reproducable
(2.6.20-rc1 + your fix) - not reproducable

buffered randread
(2.6.18) - reproducable
(2.6.20-rc1) - not reproducable
(2.6.20-rc1 + your fix) - not reproducable

So the conclusion: problem with "direct seqread" is fixed by your patch
(thanks!) and
the problem with "buffered randread" was fixed in 2.6.19/2.6.20.

Thanks for cooperation,
Vasily.

Jens Axboe wrote:
> On Wed, Dec 20 2006, Jens Axboe wrote:
>
>> I'm still seeing odd results with buffered io, while the direct case is
>> definitely fixed. So we are either looking at two bugs, or one bug that

>> the merge fixup hides.
>>
>
> Ah hang on, buffered reads will be serialized by the page cache. If I
> rerun the buffered randread case with anticipatory, the results are
> skewed in the other direction but still unfair. So I'm not sure there's
> a real bug there in the io scheduler.

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Thu, 21 Dec 2006 09:14:28 GMT
View Forum Message <> Reply to Message

On Thu, Dec 21 2006, Vasily Tarasov wrote:
> Hello, Jens,
>
> Sorry for late answer.
> The situation is the following:
>
> direct seqread
> (2.6.18) - reproducable
> (2.6.20-rc1) - reproducable
> (2.6.20-rc1 + your fix) - not reproducable
>
> buffered randread
> (2.6.18) - reproducable
> (2.6.20-rc1) - not reproducable
> (2.6.20-rc1 + your fix) - not reproducable
>
> So the conclusion: problem with "direct seqread" is fixed by your patch
> (thanks!) and
> the problem with "buffered randread" was fixed in 2.6.19/2.6.20.

Perfect, explains why I didn't see much badness with buffered io on
2.6.20-rc1'ish. Thanks for retesting and the initial report, the fix I
sent you is going upstream (sitting in the 'for-linus' branch awaiting a
pull) for 2.6.20-rc2.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Thu, 21 Dec 2006 09:15:09 GMT
View Forum Message <> Reply to Message

On Thu, Dec 21 2006, Jens Axboe wrote:

> On Thu, Dec 21 2006, Vasily Tarasov wrote:
> > Hello, Jens,
> >
> > Sorry for late answer.
> > The situation is the following:
> >
> > direct seqread
> > (2.6.18) - reproducable
> > (2.6.20-rc1) - reproducable
> > (2.6.20-rc1 + your fix) - not reproducable
> >
> > buffered randread
> > (2.6.18) - reproducable
> > (2.6.20-rc1) - not reproducable
> > (2.6.20-rc1 + your fix) - not reproducable
> >
> > So the conclusion: problem with "direct seqread" is fixed by your patch
> > (thanks!) and
> > the problem with "buffered randread" was fixed in 2.6.19/2.6.20.
>
> Perfect, explains why I didn't see much badness with buffered io on
> 2.6.20-rc1'ish. Thanks for retesting and the initial report, the fix I
> sent you is going upstream (sitting in the 'for-linus' branch awaiting a
> pull) for 2.6.20-rc2.

Oh, Linus just pulled a few minutes ago, the fix is already upstream
now.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Vasily Tarasov on Thu, 21 Dec 2006 12:16:50 GMT
View Forum Message <> Reply to Message

Hello,

On 2.6.19 "buffered randread" bizarre behavior persists. So it was fixed
somewhere in 2.6.20... Can you give me a guess, what changes in 2.6.20
cause this fix? The point is that we want to port it back to 2.6.18
(rhel-5), if possible.

Thanks,
Vasily.

Jens Axboe wrote:
> On Thu, Dec 21 2006, Jens Axboe wrote:

>
>> On Thu, Dec 21 2006, Vasily Tarasov wrote:
>>
>>> Hello, Jens,
>>>
>>> Sorry for late answer.
>>> The situation is the following:
>>>
>>> direct seqread
>>> (2.6.18) - reproducable
>>> (2.6.20-rc1) - reproducable
>>> (2.6.20-rc1 + your fix) - not reproducable
>>>
>>> buffered randread
>>> (2.6.18) - reproducable
>>> (2.6.20-rc1) - not reproducable
>>> (2.6.20-rc1 + your fix) - not reproducable
>>>
>>> So the conclusion: problem with "direct seqread" is fixed by your patch
>>> (thanks!) and
>>> the problem with "buffered randread" was fixed in 2.6.19/2.6.20.
>>>
>> Perfect, explains why I didn't see much badness with buffered io on
>> 2.6.20-rc1'ish. Thanks for retesting and the initial report, the fix I
>> sent you is going upstream (sitting in the 'for-linus' branch awaiting a
>> pull) for 2.6.20-rc2.
>>
>
> Oh, Linus just pulled a few minutes ago, the fix is already upstream
> now.

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Thu, 21 Dec 2006 13:41:43 GMT
View Forum Message <> Reply to Message

On Thu, Dec 21 2006, Vasily Tarasov wrote:
> Hello,
>
> On 2.6.19 "buffered randread" bizarre behavior persists. So it was fixed
> somewhere in 2.6.20... Can you give me a guess, what changes in 2.6.20
> cause this fix? The point is that we want to port it back to 2.6.18
> (rhel-5), if possible.

Is the 2.6.19 buffered randread case a CFQ only problem?

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Vasily Tarasov on Thu, 21 Dec 2006 15:49:38 GMT
View Forum Message <> Reply to Message

Jens Axboe wrote:
> On Thu, Dec 21 2006, Vasily Tarasov wrote:
>
>> Hello,
>>
>> On 2.6.19 "buffered randread" bizarre behavior persists. So it was fixed
>> somewhere in 2.6.20... Can you give me a guess, what changes in 2.6.20
>> cause this fix? The point is that we want to port it back to 2.6.18
>> (rhel-5), if possible.
>>
>
> Is the 2.6.19 buffered randread case a CFQ only problem?
>
Yes, deadline and anticipatory don't have no such problem at all.

Thanks.

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Thu, 21 Dec 2006 15:58:05 GMT
View Forum Message <> Reply to Message

On Thu, Dec 21 2006, Vasily Tarasov wrote:
> Jens Axboe wrote:
> > On Thu, Dec 21 2006, Vasily Tarasov wrote:
> >
> >> Hello,
> >>
> >> On 2.6.19 "buffered randread" bizarre behavior persists. So it was fixed
> >> somewhere in 2.6.20... Can you give me a guess, what changes in 2.6.20
> >> cause this fix? The point is that we want to port it back to 2.6.18
> >> (rhel-5), if possible.
> >>
> >
> > Is the 2.6.19 buffered randread case a CFQ only problem?
> >
> Yes, deadline and anticipatory don't have no such problem at all.

My best uneducated guess would be this one:

http://git.kernel.dk/?p=linux-2.6-block.git;a=commit;h=5fccbf61be2a7f32d2002b04afca4c5009612a58

Try and apply that to 2.6.18 and see if that fixes it.

--
Jens Axboe

Subject: Re: seems to be a flaw in cfq
Posted by Vasily Tarasov on Fri, 22 Dec 2006 06:27:14 GMT
View Forum Message <> Reply to Message

Unfortunately this fix exists already in 2.6.19...

Thank you anyway,
Vasily.

Jens Axboe wrote:
> My best uneducated guess would be this one:
>
>
http://git.kernel.dk/?p=linux-2.6-block.git;a=commit;h=5fccbf61be2a7f32d2002b04afca4c5009612a58
>
> Try and apply that to 2.6.18 and see if that fixes it.
>
>

Subject: Re: seems to be a flaw in cfq
Posted by Jens Axboe on Fri, 22 Dec 2006 09:37:28 GMT
View Forum Message <> Reply to Message

On Fri, Dec 22 2006, Vasily Tarasov wrote:
> Unfortunately this fix exists already in 2.6.19...

So it is, hmmm. I don't have time to look into this right now (I'll be gone between tonight and new years), but with a bit of git skills, you should be able to narrow it down.

--
Jens Axboe