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
