Subject: Re: [PATCH] cfq: get rid of cfqq hash
Posted by Jens Axboe on Wed, 25 Apr 2007 07:27:02 GMT
View Forum Message <> Reply to Message

On Wed, Apr 25 2007, Jens Axboe wrote:
> On Tue, Apr 24 2007, Vasily Tarasov wrote:
> > @@ -1806,7 +1765,11 @@ static int cfq_may_queue(request_queue_t
> > * so just lookup a possibly existing queue, or return 'may queue'
> > * if that fails
> > */
> > - cfqq = cfq_find_cfq_hash(cfqd, key, tsk->ioprio);
> > + cic = cfq_get_io_context_noalloc(cfqd, tsk);
> > + if (!cic)
> > + return ELV_MQUEUE_MAY;
> > +
> > + cfqq = cic->cfqq[rw & REQ_RW_SYNC];
> > if (cfqq) {
> > cfq_init_prio_data(cfqq);
> > cfq_prio_boost(cfqq);
>
> Ahem, how well did you test this patch?

Incremental update to get things working below.

```
diff --git a/block/cfq-iosched.c b/block/cfq-iosched.c
index ed1ce43..17e4660 100644
--- a/block/cfq-iosched.c
+++ b/block/cfq-iosched.c
@@ -202,6 +202,18 @@ static struct cfq_queue *cfq_get_queue(struct cfq_data *, int,
 static struct cfq_io_context *cfq_cic_rb_lookup(struct cfq_data *,
       struct io_context *);

+static inline struct cfq_queue *cic_to_cfqq(struct cfq_io_context *cic,
+         int is_sync)
+{
+ return cic->cfqq[!!is_sync];
+}
+
+static inline void cic_set_cfqq(struct cfq_io_context *cic,
+    struct cfq_queue *cfqq, int is_sync)
+{
+ cic->cfqq[!!is_sync] = cfqq;
+}
+
 /*
  * scheduler run of queue, if there are requests pending and no one in the
  * driver that will restart queueing
@@ -582,7 +594,7 @@ cfq_find_rq_fmerge(struct cfq_data *cfqd, struct bio *bio)
```

```
  if (!cic)
   return NULL;

- cfqq =  cic->cfqq[bio_sync(bio)];
+ cfqq =  cic_to_cfqq(cic, bio_sync(bio));
  if (cfqq) {
   sector_t sector = bio->bi_sector + bio_sectors(bio);

@@ -693,7 +705,7 @@ static int cfq_allow_merge(request_queue_t *q, struct request *rq,
  if (!cic)
   return 0;

- cfqq = cic->cfqq[bio_sync(bio)];
+ cfqq = cic_to_cfqq(cic, bio_sync(bio));
  if (cfqq == RQ_CFQQ(rq))
   return 1;

@@ -1330,7 +1342,7 @@ cfq_get_queue(struct cfq_data *cfqd, int is_sync, struct task_struct
*tsk,
 retry:
  cic = cfq_cic_rb_lookup(cfqd, tsk->io_context);
  /* cic always exists here */
- cfqq = cic->cfqq[is_sync];
+ cfqq = cic_to_cfqq(cic, is_sync);

  if (!cfqq) {
   if (new_cfqq) {
@@ -1821,7 +1833,7 @@ static int cfq_may_queue(request_queue_t *q, int rw)
  if (!cic)
   return ELV_MQUEUE_MAY;

- cfqq = cic->cfqq[rw & REQ_RW_SYNC];
+ cfqq = cic_to_cfqq(cic, rw & REQ_RW_SYNC);
  if (cfqq) {
   cfq_init_prio_data(cfqq);
   cfq_prio_boost(cfqq);
@@ -1877,15 +1889,15 @@ cfq_set_request(request_queue_t *q, struct request *rq, gfp_t
gfp_mask)
  if (!cic)
   goto queue_fail;

- if (!cic->cfqq[is_sync]) {
+ cfqq = cic_to_cfqq(cic, is_sync);
+ if (!cfqq) {
   cfqq = cfq_get_queue(cfqd, is_sync, tsk, gfp_mask);

   if (!cfqq)
    goto queue_fail;
```

```
-  cic->cfqq[is_sync] = cfqq;
- } else
-  cfqq = cic->cfqq[is_sync];
+  cic_set_cfqq(cic, cfqq, is_sync);
+ }

  cfqq->allocated[rw]++;
  cfq_clear_cfqq_must_alloc(cfqq);

--
Jens Axboe
```