Subject: [RFC][patch 5/12][CFQ-cgroup] Create cfq driver unique data
Posted by Satoshi UCHIDA on Thu, 03 Apr 2008 07:14:06 GMT
View Forum Message <> Reply to Message

This patch exacts driver unique data into new structure(cfq_driver_data)
in order to move top control layer(cfq_meata_data layer
 in next patch).

CFQ_DRV_UNIQ_DATA macro calculates control data in top control layer.
In one lalyer CFQ, macro selects cfq_driver_data in cfq_data.
In two lalyer CFQ, macro selects cfq_driver_data in cfq_meta_data.
(in [7/12] patch)

Signed-off-by: Satoshi UCHIDA <uchida@ap.jp.nec.com>

---
 block/cfq-iosched.c | 138 ++++++++++++++++++++++++++----------------
 include/linux/cfq-iosched.h | 48 ++++++++++++----
 2 files changed, 102 insertions(+), 84 deletions(-)

diff --git a/block/cfq-iosched.c b/block/cfq-iosched.c
index c1f9da9..aaf5d7e 100644
--- a/block/cfq-iosched.c
+++ b/block/cfq-iosched.c
@@ -177,7 +177,7 @@ static inline int cfq_bio_sync(struct bio *bio)
 static inline void cfq_schedule_dispatch(struct cfq_data *cfqd)
 {
  if (cfqd->busy_queues)
- kblockd_schedule_work(&cfqd->unplug_work);
+ kblockd_schedule_work(&CFQ_DRV_UNIQ_DATA(cfqd).unplug_work);
 }

 static int cfq_queue_empty(struct request_queue *q)
@@ -260,7 +260,7 @@ cfq_choose_req(struct cfq_data *cfqd, struct request *rq1, struct request
*rq2)
  s1 = rq1->sector;
  s2 = rq2->sector;

- last = cfqd->last_position;
+ last = CFQ_DRV_UNIQ_DATA(cfqd).last_position;

  /*
   * by definition, 1KiB is 2 sectors
@@ -535,7 +535,7 @@ static void cfq_add_rq_rb(struct request *rq)
   * if that happens, put the alias on the dispatch list
   */
  while ((__alias = elv_rb_add(&cfqq->sort_list, rq)) != NULL)
- cfq_dispatch_insert(cfqd->queue, __alias);

```
+  cfq_dispatch_insert(CFQ_DRV_UNIQ_DATA(cfqd).queue, __alias);

  if (!cfq_cfqq_on_rr(cfqq))
   cfq_add_cfqq_rr(cfqd, cfqq);
@@ -579,7 +579,7 @@ static void cfq_activate_request(struct request_queue *q, struct request
*rq)
 {
  struct cfq_data *cfqd = q->elevator->elevator_data;

- cfqd->rq_in_driver++;
+ CFQ_DRV_UNIQ_DATA(cfqd).rq_in_driver++;

  /*
   * If the depth is larger 1, it really could be queueing. But lets
@@ -587,18 +587,18 @@ static void cfq_activate_request(struct request_queue *q, struct
request *rq)
   * low queueing, and a low queueing number could also just indicate
   * a SCSI mid layer like behaviour where limit+1 is often seen.
   */
- if (!cfqd->hw_tag && cfqd->rq_in_driver > 4)
-  cfqd->hw_tag = 1;
+ if (!CFQ_DRV_UNIQ_DATA(cfqd).hw_tag && CFQ_DRV_UNIQ_DATA(cfqd).rq_in_driver > 4)
+  CFQ_DRV_UNIQ_DATA(cfqd).hw_tag = 1;

- cfqd->last_position = rq->hard_sector + rq->hard_nr_sectors;
+ CFQ_DRV_UNIQ_DATA(cfqd).last_position = rq->hard_sector + rq->hard_nr_sectors;
 }

 static void cfq_deactivate_request(struct request_queue *q, struct request *rq)
 {
  struct cfq_data *cfqd = q->elevator->elevator_data;

- WARN_ON(!cfqd->rq_in_driver);
- cfqd->rq_in_driver--;
+ WARN_ON(!CFQ_DRV_UNIQ_DATA(cfqd).rq_in_driver);
+ CFQ_DRV_UNIQ_DATA(cfqd).rq_in_driver--;
 }

 static void cfq_remove_request(struct request *rq)
@@ -706,7 +706,7 @@ __cfq_slice_expired(struct cfq_data *cfqd, struct cfq_queue *cfqq,
       int timed_out)
 {
  if (cfq_cfqq_wait_request(cfqq))
-  del_timer(&cfqd->idle_slice_timer);
+  del_timer(&CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer);

  cfq_clear_cfqq_must_dispatch(cfqq);
  cfq_clear_cfqq_wait_request(cfqq);
```

```
@@ -722,9 +722,9 @@ __cfq_slice_expired(struct cfq_data *cfqd, struct cfq_queue *cfqq,
  if (cfqq == cfqd->active_queue)
   cfqd->active_queue = NULL;

- if (cfqd->active_cic) {
-  put_io_context(cfqd->active_cic->ioc);
-  cfqd->active_cic = NULL;
+ if (CFQ_DRV_UNIQ_DATA(cfqd).active_cic) {
+  put_io_context(CFQ_DRV_UNIQ_DATA(cfqd).active_cic->ioc);
+  CFQ_DRV_UNIQ_DATA(cfqd).active_cic = NULL;
  }
 }

@@ -763,15 +763,15 @@ static struct cfq_queue *cfq_set_active_queue(struct cfq_data *cfqd)
 static inline sector_t cfq_dist_from_last(struct cfq_data *cfqd,
        struct request *rq)
 {
- if (rq->sector >= cfqd->last_position)
-  return rq->sector - cfqd->last_position;
+ if (rq->sector >= CFQ_DRV_UNIQ_DATA(cfqd).last_position)
+  return rq->sector - CFQ_DRV_UNIQ_DATA(cfqd).last_position;
  else
-  return cfqd->last_position - rq->sector;
+  return CFQ_DRV_UNIQ_DATA(cfqd).last_position - rq->sector;
 }

 static inline int cfq_rq_close(struct cfq_data *cfqd, struct request *rq)
 {
- struct cfq_io_context *cic = cfqd->active_cic;
+ struct cfq_io_context *cic = CFQ_DRV_UNIQ_DATA(cfqd).active_cic;

  if (!sample_valid(cic->seek_samples))
   return 0;
@@ -804,13 +804,13 @@ static void cfq_arm_slice_timer(struct cfq_data *cfqd)
  /*
   * idle is disabled, either manually or by past process history
   */
- if (!cfqd->cfq_slice_idle || !cfq_cfqq_idle_window(cfqq))
+ if (!CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle || !cfq_cfqq_idle_window(cfqq))
   return;

  /*
   * task has exited, don't wait
   */
- cic = cfqd->active_cic;
+ cic = CFQ_DRV_UNIQ_DATA(cfqd).active_cic;
  if (!cic || !atomic_read(&cic->ioc->nr_tasks))
   return;
```

```
@@ -829,11 +829,11 @@ static void cfq_arm_slice_timer(struct cfq_data *cfqd)
   * fair distribution of slice time for a process doing back-to-back
   * seeks. so allow a little bit of time for him to submit a new rq
   */
- sl = cfqd->cfq_slice_idle;
+ sl = CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle;
  if (sample_valid(cic->seek_samples) && CIC_SEEKY(cic))
   sl = min(sl, msecs_to_jiffies(CFQ_MIN_TT));

- mod_timer(&cfqd->idle_slice_timer, jiffies + sl);
+ mod_timer(&CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer, jiffies + sl);
 }

 /*
@@ -849,7 +849,7 @@ static void cfq_dispatch_insert(struct request_queue *q, struct request
*rq)
  elv_dispatch_sort(q, rq);

  if (cfq_cfqq_sync(cfqq))
-  cfqd->sync_flight++;
+  CFQ_DRV_UNIQ_DATA(cfqd).sync_flight++;
 }

 /*
@@ -918,7 +918,7 @@ static struct cfq_queue *cfq_select_queue(struct cfq_data *cfqd)
   * flight or is idling for a new request, allow either of these
   * conditions to happen (or time out) before selecting a new queue.
   */
- if (timer_pending(&cfqd->idle_slice_timer) ||
+ if (timer_pending(&CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer) ||
      (cfqq->dispatched && cfq_cfqq_idle_window(cfqq))) {
   cfqq = NULL;
   goto keep_queue;
@@ -957,13 +957,13 @@ __cfq_dispatch_requests(struct cfq_data *cfqd, struct cfq_queue
*cfqq,
   /*
    * finally, insert request into driver dispatch list
    */
-  cfq_dispatch_insert(cfqd->queue, rq);
+  cfq_dispatch_insert(CFQ_DRV_UNIQ_DATA(cfqd).queue, rq);

   dispatched++;

-  if (!cfqd->active_cic) {
+  if (!CFQ_DRV_UNIQ_DATA(cfqd).active_cic) {
    atomic_inc(&RQ_CIC(rq)->ioc->refcount);
-   cfqd->active_cic = RQ_CIC(rq);
```

```
+   CFQ_DRV_UNIQ_DATA(cfqd).active_cic = RQ_CIC(rq);
   }

   if (RB_EMPTY_ROOT(&cfqq->sort_list))
@@ -990,7 +990,7 @@ static int __cfq_forced_dispatch_cfqq(struct cfq_queue *cfqq)
  int dispatched = 0;

  while (cfqq->next_rq) {
-  cfq_dispatch_insert(cfqq->cfqd->queue, cfqq->next_rq);
+  cfq_dispatch_insert(CFQ_DRV_UNIQ_DATA(cfqq->cfqd).queue, cfqq->next_rq);
   dispatched++;
  }

@@ -1044,12 +1044,12 @@ static int cfq_dispatch_requests(struct request_queue *q, int force)
     break;
   }

-  if (cfqd->sync_flight && !cfq_cfqq_sync(cfqq))
+  if (CFQ_DRV_UNIQ_DATA(cfqd).sync_flight && !cfq_cfqq_sync(cfqq))
    break;

   cfq_clear_cfqq_must_dispatch(cfqq);
   cfq_clear_cfqq_wait_request(cfqq);
-  del_timer(&cfqd->idle_slice_timer);
+  del_timer(&CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer);

   dispatched += __cfq_dispatch_requests(cfqd, cfqq, max_dispatch);
  }
@@ -1175,7 +1175,7 @@ static void cfq_exit_single_io_context(struct io_context *ioc,
  struct cfq_data *cfqd = cic->key;

  if (cfqd) {
-  struct request_queue *q = cfqd->queue;
+  struct request_queue *q = CFQ_DRV_UNIQ_DATA(cfqd).queue;
   unsigned long flags;

   spin_lock_irqsave(q->queue_lock, flags);
@@ -1200,7 +1200,7 @@ cfq_alloc_io_context(struct cfq_data *cfqd, gfp_t gfp_mask)
  struct cfq_io_context *cic;

  cic = kmem_cache_alloc_node(cfq_ioc_pool, gfp_mask | __GFP_ZERO,
-       cfqd->queue->node);
+     CFQ_DRV_UNIQ_DATA(cfqd).queue->node);
  if (cic) {
   cic->last_end_request = jiffies;
   INIT_LIST_HEAD(&cic->queue_list);
@@ -1265,7 +1265,7 @@ static void changed_ioprio(struct io_context *ioc, struct cfq_io_context
*cic)
```

```
  if (unlikely(!cfqd))
   return;

- spin_lock_irqsave(cfqd->queue->queue_lock, flags);
+ spin_lock_irqsave(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock, flags);

  cfqq = cic->cfqq[ASYNC];
  if (cfqq) {
@@ -1281,7 +1281,7 @@ static void changed_ioprio(struct io_context *ioc, struct cfq_io_context
*cic)
  if (cfqq)
   cfq_mark_cfqq_prio_changed(cfqq);

- spin_unlock_irqrestore(cfqd->queue->queue_lock, flags);
+ spin_unlock_irqrestore(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock, flags);
 }

 static void cfq_ioc_set_ioprio(struct io_context *ioc)
@@ -1313,16 +1313,16 @@ retry:
     * the allocator to do whatever it needs to attempt to
     * free memory.
     */
-   spin_unlock_irq(cfqd->queue->queue_lock);
+   spin_unlock_irq(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock);
    new_cfqq = kmem_cache_alloc_node(cfq_pool,
      gfp_mask | __GFP_NOFAIL | __GFP_ZERO,
-     cfqd->queue->node);
-   spin_lock_irq(cfqd->queue->queue_lock);
+     CFQ_DRV_UNIQ_DATA(cfqd).queue->node);
+   spin_lock_irq(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock);
    goto retry;
   } else {
    cfqq = kmem_cache_alloc_node(cfq_pool,
      gfp_mask | __GFP_ZERO,
-     cfqd->queue->node);
+     CFQ_DRV_UNIQ_DATA(cfqd).queue->node);
    if (!cfqq)
     goto out;
   }
@@ -1494,9 +1494,9 @@ static int cfq_cic_link(struct cfq_data *cfqd, struct io_context *ioc,
   radix_tree_preload_end();

   if (!ret) {
-   spin_lock_irqsave(cfqd->queue->queue_lock, flags);
+   spin_lock_irqsave(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock, flags);
    list_add(&cic->queue_list, &cfqd->cic_list);
-   spin_unlock_irqrestore(cfqd->queue->queue_lock, flags);
+   spin_unlock_irqrestore(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock, flags);
```

```
  }
 }

@@ -1519,7 +1519,7 @@ cfq_get_io_context(struct cfq_data *cfqd, gfp_t gfp_mask)

  might_sleep_if(gfp_mask & __GFP_WAIT);

- ioc = get_io_context(gfp_mask, cfqd->queue->node);
+ ioc = get_io_context(gfp_mask, CFQ_DRV_UNIQ_DATA(cfqd).queue->node);
  if (!ioc)
   return NULL;

@@ -1551,7 +1551,7 @@ static void
 cfq_update_io_thinktime(struct cfq_data *cfqd, struct cfq_io_context *cic)
 {
  unsigned long elapsed = jiffies - cic->last_end_request;
- unsigned long ttime = min(elapsed, 2UL * cfqd->cfq_slice_idle);
+ unsigned long ttime = min(elapsed, 2UL * CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle);

  cic->ttime_samples = (7*cic->ttime_samples + 256) / 8;
  cic->ttime_total = (7*cic->ttime_total + 256*ttime) / 8;
@@ -1604,11 +1604,11 @@ cfq_update_idle_window(struct cfq_data *cfqd, struct cfq_queue
*cfqq,

  enable_idle = cfq_cfqq_idle_window(cfqq);

- if (!atomic_read(&cic->ioc->nr_tasks) || !cfqd->cfq_slice_idle ||
-     (cfqd->hw_tag && CIC_SEEKY(cic)))
+ if (!atomic_read(&cic->ioc->nr_tasks) || !CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle ||
+     (CFQ_DRV_UNIQ_DATA(cfqd).hw_tag && CIC_SEEKY(cic)))
   enable_idle = 0;
  else if (sample_valid(cic->ttime_samples)) {
-  if (cic->ttime_mean > cfqd->cfq_slice_idle)
+  if (cic->ttime_mean > CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle)
    enable_idle = 0;
   else
    enable_idle = 1;
@@ -1657,7 +1657,7 @@ cfq_should_preempt(struct cfq_data *cfqd, struct cfq_queue
*new_cfqq,
  if (rq_is_meta(rq) && !cfqq->meta_pending)
   return 1;

- if (!cfqd->active_cic || !cfq_cfqq_wait_request(cfqq))
+ if (!CFQ_DRV_UNIQ_DATA(cfqd).active_cic || !cfq_cfqq_wait_request(cfqq))
   return 0;

  /*
@@ -1717,8 +1717,8 @@ cfq_rq_enqueued(struct cfq_data *cfqd, struct cfq_queue *cfqq,
```

```
     */
    if (cfq_cfqq_wait_request(cfqq)) {
     cfq_mark_cfqq_must_dispatch(cfqq);
-    del_timer(&cfqd->idle_slice_timer);
-    blk_start_queueing(cfqd->queue);
+    del_timer(&CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer);
+    blk_start_queueing(CFQ_DRV_UNIQ_DATA(cfqd).queue);
    }
   } else if (cfq_should_preempt(cfqd, cfqq, rq)) {
    /*
@@ -1728,7 +1728,7 @@ cfq_rq_enqueued(struct cfq_data *cfqd, struct cfq_queue *cfqq,
     */
    cfq_preempt_queue(cfqd, cfqq);
    cfq_mark_cfqq_must_dispatch(cfqq);
-   blk_start_queueing(cfqd->queue);
+   blk_start_queueing(CFQ_DRV_UNIQ_DATA(cfqd).queue);
   }
  }

@@ -1755,16 +1755,16 @@ static void cfq_completed_request(struct request_queue *q, struct
request *rq)

   now = jiffies;

- WARN_ON(!cfqd->rq_in_driver);
+ WARN_ON(!CFQ_DRV_UNIQ_DATA(cfqd).rq_in_driver);
   WARN_ON(!cfqq->dispatched);
- cfqd->rq_in_driver--;
+ CFQ_DRV_UNIQ_DATA(cfqd).rq_in_driver--;
   cfqq->dispatched--;

   if (cfq_cfqq_sync(cfqq))
-   cfqd->sync_flight--;
+   CFQ_DRV_UNIQ_DATA(cfqd).sync_flight--;

   if (!cfq_class_idle(cfqq))
-   cfqd->last_end_request = now;
+   CFQ_DRV_UNIQ_DATA(cfqd).last_end_request = now;

   if (sync)
    RQ_CIC(rq)->last_end_request = now;
@@ -1784,7 +1784,7 @@ static void cfq_completed_request(struct request_queue *q, struct
request *rq)
     cfq_arm_slice_timer(cfqd);
   }

- if (!cfqd->rq_in_driver)
+ if (!CFQ_DRV_UNIQ_DATA(cfqd).rq_in_driver)
```

```
   cfq_schedule_dispatch(cfqd);
 }

@@ -1928,9 +1928,7 @@ queue_fail:

 static void cfq_kick_queue(struct work_struct *work)
 {
- struct cfq_data *cfqd =
-  container_of(work, struct cfq_data, unplug_work);
- struct request_queue *q = cfqd->queue;
+ struct request_queue *q =  __cfq_container_of_queue(work);
   unsigned long flags;

   spin_lock_irqsave(q->queue_lock, flags);
@@ -1948,7 +1946,7 @@ static void cfq_idle_slice_timer(unsigned long data)
   unsigned long flags;
   int timed_out = 1;

- spin_lock_irqsave(cfqd->queue->queue_lock, flags);
+ spin_lock_irqsave(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock, flags);

   cfqq = cfqd->active_queue;
   if (cfqq) {
@@ -1980,13 +1978,13 @@ expire:
 out_kick:
   cfq_schedule_dispatch(cfqd);
 out_cont:
- spin_unlock_irqrestore(cfqd->queue->queue_lock, flags);
+ spin_unlock_irqrestore(CFQ_DRV_UNIQ_DATA(cfqd).queue->queue_lock, flags);
 }

 static void cfq_shutdown_timer_wq(struct cfq_data *cfqd)
 {
- del_timer_sync(&cfqd->idle_slice_timer);
- kblockd_flush_work(&cfqd->unplug_work);
+ del_timer_sync(&CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer);
+ kblockd_flush_work(&CFQ_DRV_UNIQ_DATA(cfqd).unplug_work);
 }

 static void cfq_put_async_queues(struct cfq_data *cfqd)
@@ -2007,7 +2005,7 @@ static void cfq_put_async_queues(struct cfq_data *cfqd)
 static void cfq_exit_queue(elevator_t *e)
 {
   struct cfq_data *cfqd = e->elevator_data;
- struct request_queue *q = cfqd->queue;
+ struct request_queue *q = CFQ_DRV_UNIQ_DATA(cfqd).queue;

   cfq_shutdown_timer_wq(cfqd);
```

```
@@ -2044,15 +2042,15 @@ static void *cfq_init_queue(struct request_queue *q)
  cfqd->service_tree = CFQ_RB_ROOT;
  INIT_LIST_HEAD(&cfqd->cic_list);

- cfqd->queue = q;
+ CFQ_DRV_UNIQ_DATA(cfqd).queue = q;

- init_timer(&cfqd->idle_slice_timer);
- cfqd->idle_slice_timer.function = cfq_idle_slice_timer;
- cfqd->idle_slice_timer.data = (unsigned long) cfqd;
+ init_timer(&CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer);
+ CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer.function = cfq_idle_slice_timer;
+ CFQ_DRV_UNIQ_DATA(cfqd).idle_slice_timer.data = (unsigned long) cfqd;

- INIT_WORK(&cfqd->unplug_work, cfq_kick_queue);
+ INIT_WORK(&CFQ_DRV_UNIQ_DATA(cfqd).unplug_work, cfq_kick_queue);

- cfqd->last_end_request = jiffies;
+ CFQ_DRV_UNIQ_DATA(cfqd).last_end_request = jiffies;
  cfqd->cfq_quantum = cfq_quantum;
  cfqd->cfq_fifo_expire[0] = cfq_fifo_expire[0];
  cfqd->cfq_fifo_expire[1] = cfq_fifo_expire[1];
@@ -2061,7 +2059,7 @@ static void *cfq_init_queue(struct request_queue *q)
  cfqd->cfq_slice[0] = cfq_slice_async;
  cfqd->cfq_slice[1] = cfq_slice_sync;
  cfqd->cfq_slice_async_rq = cfq_slice_async_rq;
- cfqd->cfq_slice_idle = cfq_slice_idle;
+ CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle = cfq_slice_idle;

  return cfqd;
 }
@@ -2122,7 +2120,7 @@ SHOW_FUNCTION(cfq_fifo_expire_sync_show,
cfqd->cfq_fifo_expire[1], 1);
 SHOW_FUNCTION(cfq_fifo_expire_async_show, cfqd->cfq_fifo_expire[0], 1);
 SHOW_FUNCTION(cfq_back_seek_max_show, cfqd->cfq_back_max, 0);
 SHOW_FUNCTION(cfq_back_seek_penalty_show, cfqd->cfq_back_penalty, 0);
-SHOW_FUNCTION(cfq_slice_idle_show, cfqd->cfq_slice_idle, 1);
+SHOW_FUNCTION(cfq_slice_idle_show, CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle, 1);
 SHOW_FUNCTION(cfq_slice_sync_show, cfqd->cfq_slice[1], 1);
 SHOW_FUNCTION(cfq_slice_async_show, cfqd->cfq_slice[0], 1);
 SHOW_FUNCTION(cfq_slice_async_rq_show, cfqd->cfq_slice_async_rq, 0);
@@ -2152,7 +2150,7 @@ STORE_FUNCTION(cfq_fifo_expire_async_store,
&cfqd->cfq_fifo_expire[0], 1,
 STORE_FUNCTION(cfq_back_seek_max_store, &cfqd->cfq_back_max, 0, UINT_MAX, 0);
 STORE_FUNCTION(cfq_back_seek_penalty_store, &cfqd->cfq_back_penalty, 1,
   UINT_MAX, 0);
-STORE_FUNCTION(cfq_slice_idle_store, &cfqd->cfq_slice_idle, 0, UINT_MAX, 1);
```

```
+STORE_FUNCTION(cfq_slice_idle_store, &CFQ_DRV_UNIQ_DATA(cfqd).cfq_slice_idle, 0,
UINT_MAX, 1);
 STORE_FUNCTION(cfq_slice_sync_store, &cfqd->cfq_slice[1], 1, UINT_MAX, 1);
 STORE_FUNCTION(cfq_slice_async_store, &cfqd->cfq_slice[0], 1, UINT_MAX, 1);
 STORE_FUNCTION(cfq_slice_async_rq_store, &cfqd->cfq_slice_async_rq, 1,
diff --git a/include/linux/cfq-iosched.h b/include/linux/cfq-iosched.h
index cce3993..035bfc4 100644
--- a/include/linux/cfq-iosched.h
+++ b/include/linux/cfq-iosched.h
@@ -19,30 +19,43 @@ struct cfq_rb_root {
 };
 #define CFQ_RB_ROOT (struct cfq_rb_root) { RB_ROOT, NULL, }

+
+#define CFQ_DRV_UNIQ_DATA(cfqd) ((cfqd)->cfq_driv_d)
+
 /*
- * Per block device queue structure
+ * Driver unique data
  */
-struct cfq_data {
+struct cfq_driver_data {
  struct request_queue *queue;

- /*
-  * rr list of queues with requests and the count of them
-  */
- struct cfq_rb_root service_tree;
- unsigned int busy_queues;
-
  int rq_in_driver;
  int sync_flight;
  int hw_tag;

+ struct cfq_io_context *active_cic;
+ struct work_struct unplug_work;
+
+ sector_t last_position;
+ unsigned long last_end_request;
+
  /*
   * idle window management
   */
  struct timer_list idle_slice_timer;
- struct work_struct unplug_work;
+ unsigned int cfq_slice_idle;
+};
+
```

```
+/*
+ * Per block device queue structure
+ */
+struct cfq_data {
+ /*
+  * rr list of queues with requests and the count of them
+  */
+ struct cfq_rb_root service_tree;
+ unsigned int busy_queues;

  struct cfq_queue *active_queue;
- struct cfq_io_context *active_cic;

  /*
   * async queue for each priority case
@@ -50,9 +63,6 @@ struct cfq_data {
  struct cfq_queue *async_cfqq[2][IOPRIO_BE_NR];
  struct cfq_queue *async_idle_cfqq;

- sector_t last_position;
- unsigned long last_end_request;
-
  /*
   * tunables, see top of file
   */
@@ -62,9 +72,19 @@ struct cfq_data {
  unsigned int cfq_back_max;
  unsigned int cfq_slice[2];
  unsigned int cfq_slice_async_rq;
- unsigned int cfq_slice_idle;

  struct list_head cic_list;
+
+ struct cfq_driver_data cfq_driv_d;
+};
+
+static inline struct request_queue * __cfq_container_of_queue(struct work_struct *work) {
+ struct cfq_driver_data *cfqdd =
+  container_of(work, struct cfq_driver_data, unplug_work);
+ struct cfq_data *cfqd =
+  container_of(cfqdd, struct cfq_data, cfq_driv_d);
+
+ return cfqd->cfq_driv_d.queue;
 };

 #endif  /* _LINUX_CFQ_IOSCHED_H */
--
1.5.4.1
```

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers