Subject: [RFC][PATCH 5/9] cgroups: block: cfq: I/O bandwidth controlling subsystem for CGroups based on CFQ
Posted by Vasily Tarasov on Fri, 15 Feb 2008 06:59:49 GMT
View Forum Message <> Reply to Message

From: Vasily Tarasov <vtaras@openvz.org>

Switches the CFQ scheduler to use per-cgroup queue tree,
instead one queue tree per device.

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

```
--- linux-2.6.25-rc5-mm1/include/linux/cfq-iosched.h.switch 2008-02-15 01:07:29.000000000
-0500
+++ linux-2.6.25-rc5-mm1/include/linux/cfq-iosched.h 2008-02-15 01:09:09.000000000 -0500
@@ -70,9 +70,8 @@ struct cfq_data {
  struct cfqio_cgroup_data cfqio_cgroup;
 #endif
  /*
-  * rr list of queues with requests and the count of them
+  * count of queues
   */
- struct cfq_rb_root service_tree;
  unsigned int busy_queues;

  int rq_in_driver;
--- linux-2.6.25-rc5-mm1/block/cfq-iosched.c.switch 2008-02-15 01:08:25.000000000 -0500
+++ linux-2.6.25-rc5-mm1/block/cfq-iosched.c 2008-02-15 01:09:09.000000000 -0500
@@ -354,6 +354,7 @@ static unsigned long cfq_slice_offset(st
 static void cfq_service_tree_add(struct cfq_data *cfqd,
         struct cfq_queue *cfqq, int add_front)
 {
+ struct cfqio_cgroup_data *cfqio_cgrp = cfqq->cfqio_cgrp;
  struct rb_node **p, *parent;
  struct cfq_queue *__cfqq;
  unsigned long rb_key;
@@ -361,7 +362,7 @@ static void cfq_service_tree_add(struct

  if (cfq_class_idle(cfqq)) {
   rb_key = CFQ_IDLE_DELAY;
-  parent = rb_last(&cfqd->service_tree.rb);
+  parent = rb_last(&cfqio_cgrp->service_tree.rb);
   if (parent && parent != &cfqq->rb_node) {
    __cfqq = rb_entry(parent, struct cfq_queue, rb_node);
    rb_key += __cfqq->rb_key;
@@ -381,12 +382,12 @@ static void cfq_service_tree_add(struct
```

```
   if (rb_key == cfqq->rb_key)
    return;

-  cfq_rb_erase(&cfqq->rb_node, &cfqd->service_tree);
+  cfq_rb_erase(&cfqq->rb_node, &cfqio_cgrp->service_tree);
  }

  left = 1;
  parent = NULL;
- p = &cfqd->service_tree.rb.rb_node;
+ p = &cfqio_cgrp->service_tree.rb.rb_node;
  while (*p) {
   struct rb_node **n;

@@ -418,11 +419,11 @@ static void cfq_service_tree_add(struct
  }

  if (left)
-  cfqd->service_tree.left = &cfqq->rb_node;
+  cfqio_cgrp->service_tree.left = &cfqq->rb_node;

  cfqq->rb_key = rb_key;
  rb_link_node(&cfqq->rb_node, parent, p);
- rb_insert_color(&cfqq->rb_node, &cfqd->service_tree.rb);
+ rb_insert_color(&cfqq->rb_node, &cfqio_cgrp->service_tree.rb);
 }

 /*
@@ -456,11 +457,12 @@ static void cfq_add_cfqq_rr(struct cfq_d
  */
 static void cfq_del_cfqq_rr(struct cfq_data *cfqd, struct cfq_queue *cfqq)
 {
+ struct cfqio_cgroup_data *cfqio_cgrp = cfqq->cfqio_cgrp;
  BUG_ON(!cfq_cfqq_on_rr(cfqq));
  cfq_clear_cfqq_on_rr(cfqq);

  if (!RB_EMPTY_NODE(&cfqq->rb_node))
-  cfq_rb_erase(&cfqq->rb_node, &cfqd->service_tree);
+  cfq_rb_erase(&cfqq->rb_node, &cfqio_cgrp->service_tree);

  BUG_ON(!cfqd->busy_queues);
  cfqd->busy_queues--;
@@ -704,10 +706,16 @@ static inline void cfq_slice_expired(str
  */
 static struct cfq_queue *cfq_get_next_queue(struct cfq_data *cfqd)
 {
- if (RB_EMPTY_ROOT(&cfqd->service_tree.rb))
+ struct cfqio_cgroup_data *cfqio_cgrp;
```

```
+
+ cfqio_cgrp = cfqd->active_cfqio_cgroup;
+ if (!cfqio_cgrp)
+  return NULL;
+
+ if (RB_EMPTY_ROOT(&cfqio_cgrp->service_tree.rb))
   return NULL;

- return cfq_rb_first(&cfqd->service_tree);
+ return cfq_rb_first(&cfqio_cgrp->service_tree);
 }

 /*
@@ -964,7 +972,7 @@ static int __cfq_forced_dispatch_cfqq(st
  * Drain our current requests. Used for barriers and when switching
  * io schedulers on-the-fly.
  */
-static int cfq_forced_dispatch(struct cfq_data *cfqd)
+static int __cfq_forced_dispatch(struct cfqio_cgroup_data *cfqd)
 {
  struct cfq_queue *cfqq;
  int dispatched = 0;
@@ -972,6 +980,25 @@ static int cfq_forced_dispatch(struct cf
  while ((cfqq = cfq_rb_first(&cfqd->service_tree)) != NULL)
   dispatched += __cfq_forced_dispatch_cfqq(cfqq);

+ return dispatched;
+}
+
+static int cfq_forced_dispatch(struct cfq_data *cfqd)
+{
+ struct cfqio_cgroup_data *cfqio_cgrp;
+ struct cfqio_cgroup_data *cfqio_cgrp_tmp;
+ int dispatched;
+
+ dispatched = 0;
+
+ /*
+  * We use _safe iterating here because  __cfq_forced_dispatch()
+  * produces list_del() implicitly
+  */
+ list_for_each_entry_safe(cfqio_cgrp, cfqio_cgrp_tmp,
+    &cfqd->act_cfqio_cgrp_head, act_cfqio_cgrp_list)
+  dispatched += __cfq_forced_dispatch(cfqio_cgrp);
+
  cfq_slice_expired(cfqd, 0);

  BUG_ON(cfqd->busy_queues);
```

```
@@ -2025,7 +2052,6 @@ static void *cfq_init_queue(struct reque
  if (!cfqd)
   return NULL;

- cfqd->service_tree = CFQ_RB_ROOT;
 #ifndef CONFIG_CGROUP_CFQIO
  cfqio_init_cfqio_cgroup(&cfqd->cfqio_cgroup);
 #endif
```
