Subject: [RFC][patch 12/12][CFQ-cgroup] entry/remove active cfq_data
Posted by Satoshi UCHIDA on Thu, 03 Apr 2008 07:18:10 GMT
View Forum Message <> Reply to Message

This patch controls whether cfq_data is active or not.
When cfq_data is not active and active cfq_queue is inserted into cfq_data,
cfq_data is activated.
When cfq_data is active and active cfq_queue is not exist,
cfq_data is deactivated.

The new cfq optional operations:
The "cfq_add_cfqq_opt_fn" defines a function that runs an additional process
when active queue is inserted into cfq_data.

The "cfq_del_cfqq_opt_fn" defines a function that runs an additional process
when active queue is removed in cfq_data.

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

---
block/cfq-cgroup.c | 28 ++++++++++++++++++++++++++++
block/cfq-iosched.c | 6 ++++++
include/linux/cfq-iosched.h | 4 ++++
3 files changed, 38 insertions(+), 0 deletions(-)

```
diff --git a/block/cfq-cgroup.c b/block/cfq-cgroup.c
index 27a9a7a..f868f4f 100644
--- a/block/cfq-cgroup.c
+++ b/block/cfq-cgroup.c
@@ -741,6 +741,32 @@ static int cfq_cgroup_active_data_check(struct cfq_data *cfqd)
  return (cfqd->cfqmd->active_data == cfqd);
 }

+static void cfq_cgroup_add_cfqd_rr(struct cfq_data *cfqd)
+{
+ if (!cfq_cfqd_on_rr(cfqd)) {
+  cfq_mark_cfqd_on_rr(cfqd);
+  cfqd->cfqmd->busy_data++;
+
+  cfq_cgroup_service_tree_add(cfqd, 0);
+ }
+}
+
+
+static void cfq_cgroup_del_cfqd_rr(struct cfq_data *cfqd)
+{
+ if (RB_EMPTY_ROOT(&cfqd->service_tree.rb)) {
+  struct cfq_meta_data *cfqdd = cfqd->cfqmd;
```

```
+  BUG_ON(!cfq_cfqd_on_rr(cfqd));
+  cfq_clear_cfqd_on_rr(cfqd);
+  if (!RB_EMPTY_NODE(&cfqd->rb_node)) {
+   cfq_rb_erase(&cfqd->rb_node,
+         &cfqdd->service_tree);
+  }
+  BUG_ON(!cfqdd->busy_data);
+  cfqdd->busy_data--;
+ }
+}
+
 struct cfq_ops opt = {
  .cfq_init_queue_fn = __cfq_cgroup_init_queue,
  .cfq_exit_queue_fn = __cfq_cgroup_exit_data,
@@ -749,4 +775,6 @@ struct cfq_ops opt = {
  .cfq_completed_request_after_fn = cfq_cgroup_completed_request_after,
  .cfq_empty_fn = cfq_cgroup_queue_empty,
  .cfq_active_check_fn = cfq_cgroup_active_data_check,
+ .cfq_add_cfqq_opt_fn =  cfq_cgroup_add_cfqd_rr,
+ .cfq_del_cfqq_opt_fn =  cfq_cgroup_del_cfqd_rr,
 };
diff --git a/block/cfq-iosched.c b/block/cfq-iosched.c
index 505e425..8f5227f 100644
--- a/block/cfq-iosched.c
+++ b/block/cfq-iosched.c
@@ -492,6 +492,9 @@ static void cfq_add_cfqq_rr(struct cfq_data *cfqd, struct cfq_queue *cfqq)
  cfqd->busy_queues++;

  cfq_resort_rr_list(cfqd, cfqq);
+
+ if (opt.cfq_add_cfqq_opt_fn)
+  opt.cfq_add_cfqq_opt_fn(cfqd);
 }

 /*
@@ -508,6 +511,9 @@ static void cfq_del_cfqq_rr(struct cfq_data *cfqd, struct cfq_queue *cfqq)

  BUG_ON(!cfqd->busy_queues);
  cfqd->busy_queues--;
+
+ if (opt.cfq_del_cfqq_opt_fn)
+  opt.cfq_del_cfqq_opt_fn(cfqd);
 }

 /*
diff --git a/include/linux/cfq-iosched.h b/include/linux/cfq-iosched.h
index 63d2545..ed35050 100644
--- a/include/linux/cfq-iosched.h
```

```
+++ b/include/linux/cfq-iosched.h
@@ -161,6 +161,8 @@ typedef int (cfq_dispatch_requests_fn)(struct cfq_data *, int);
 typedef int (cfq_completed_request_after_fn)(struct cfq_data *);
 typedef int (cfq_active_check_fn)(struct cfq_data *);
 typedef int (cfq_empty_fn)(struct cfq_data *);
+typedef void (cfq_add_cfqq_opt_fn)(struct cfq_data *);
+typedef void (cfq_del_cfqq_opt_fn)(struct cfq_data *);

 struct cfq_ops
 {
@@ -171,6 +173,8 @@ struct cfq_ops
   cfq_completed_request_after_fn *cfq_completed_request_after_fn;
   cfq_active_check_fn *cfq_active_check_fn;
   cfq_empty_fn *cfq_empty_fn;
+ cfq_add_cfqq_opt_fn *cfq_add_cfqq_opt_fn;
+ cfq_del_cfqq_opt_fn *cfq_del_cfqq_opt_fn;
 };

 extern struct cfq_ops opt;
--
1.5.4.1
```

_____

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