
Subject: [RFC][patch 8/12][CFQ-cgroup] Control cfq_data per driver
Posted by [Satoshi UCHIDA](#) on Thu, 03 Apr 2008 07:15:42 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch expands cfq_meta_data to handling multi cfq_data.
This control is used rb_tree and the key is used a pointer of cfq_data.

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

```
---
block/cfq-cgroup.c      | 62 ++++++
include/linux/cfq-iosched.h | 7 +++++
2 files changed, 68 insertions(+), 1 deletions(-)

diff --git a/block/cfq-cgroup.c b/block/cfq-cgroup.c
index 68336b1..ba0f3db 100644
--- a/block/cfq-cgroup.c
+++ b/block/cfq-cgroup.c
@@ -54,9 +54,42 @@ static struct cfq_meta_data *cfq_cgroup_init_meta_data(struct cfq_data
*cfqd, st
    cfqmd->cfq_driv_d.idle_slice_timer.data = (unsigned long) cfqd;
    cfqmd->cfq_driv_d.cfq_slice_idle = cfq_cgroup_slice_idle;

+ cfqmd->sibling_tree = RB_ROOT;
+ cfqmd->siblings = 0;
+
    return cfqmd;
}

+static void cfq_meta_data_sibling_tree_add(struct cfq_meta_data *cfqmd,
+      struct cfq_data *cfqd)
+{
+    struct rb_node **p;
+    struct rb_node *parent = NULL;
+
+    BUG_ON(!RB_EMPTY_NODE(&cfqd->sib_node));
+
+    p = &cfqmd->sibling_tree.rb_node;
+
+    while (*p) {
+        struct cfq_data *__cfqd;
+        struct rb_node **n;
+
+        parent = *p;
+        __cfqd = rb_entry(parent, struct cfq_data, sib_node);
+
+        if (cfqd < __cfqd) {
+            n = &(*p)->rb_left;

```

```

+ } else {
+   n = &(*p)->rb_right;
+ }
+ p = n;
+ }
+
+ rb_link_node(&cfqd->sib_node, parent, p);
+ rb_insert_color(&cfqd->sib_node, &cfqmd->sibling_tree);
+ cfqmd->siblings++;
+ cfqd->cfqmd = cfqmd;
+}

struct cfq_data *__cfq_cgroup_init_queue(struct request_queue *q, void *data)
{
@@ -66,6 +99,8 @@ struct cfq_data *__cfq_cgroup_init_queue(struct request_queue *q, void
*data)
    if (!cfqd)
        return NULL;

+ RB_CLEAR_NODE(&cfqd->sib_node);
+
    if (!cfqmd) {
        cfqmd = cfq_cgroup_init_meta_data(cfqd, q);
        if (!cfqmd) {
@@ -73,6 +108,7 @@ struct cfq_data *__cfq_cgroup_init_queue(struct request_queue *q, void
*data)
            return NULL;
        }
    }
+ cfq_meta_data_sibling_tree_add(cfqmd, cfqd);

    return cfqd;
}
@@ -102,11 +138,35 @@ cfq_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
/*
 * Remove device or cgroup data functions.
 */
+static void cfq_cgroup_erase_meta_data_siblings(struct cfq_meta_data *cfqmd, struct cfq_data
*cfqd)
+{
+ rb_erase(&cfqd->sib_node, &cfqmd->sibling_tree);
+ cfqmd->siblings--;
+}
+
+static void cfq_exit_device_group(struct cfq_meta_data *cfqmd)
+{
+ struct rb_node *p, *n;
+ struct cfq_data *cfqd;

```

```

+
+ p = rb_first(&cfqmd->sibling_tree);
+
+ while (p) {
+   n = rb_next(p);
+   cfqd = rb_entry(p, struct cfq_data, sib_node);
+
+   cfq_cgroup_erase_meta_data_siblings(cfqmd, cfqd);
+   __cfq_exit_data(cfqd);
+
+   p = n;
+ }
+}
+
static void __cfq_cgroup_exit_data(struct cfq_data *cfqd)
{
    struct cfq_meta_data *cfqmd = cfqd->cfqmd;

-   __cfq_exit_data(cfqd);
+   cfq_exit_device_group(cfqmd);
    kfree(cfqmd);
}

diff --git a/include/linux/cfq-iosched.h b/include/linux/cfq-iosched.h
index e5b41da..c8879a7 100644
--- a/include/linux/cfq-iosched.h
+++ b/include/linux/cfq-iosched.h
@@ -54,6 +54,10 @@ struct cfq_driver_data {
    */
    struct cfq_meta_data {
        struct cfq_data *elv_data;
+
+   /* device siblings */
+   struct rb_root sibling_tree;
+   unsigned int siblings;

    struct cfq_driver_data cfq_driv_d;
};
@@ -91,6 +95,9 @@ struct cfq_data {

#ifdef CONFIG_CGROUP_CFQ
    struct cfq_meta_data *cfqmd;
+   /* sibling_tree member for cfq_meta_data */
+   struct rb_node sib_node;
+
#else
    struct cfq_driver_data cfq_driv_d;
#endif

```

--

1.5.4.1

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
