Subject: [RFC][patch 9/12][CFQ-cgroup] Control cfq_data per cgroup
Posted by Satoshi UCHIDA on Thu, 03 Apr 2008 07:16:06 GMT
View Forum Message <> Reply to Message

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

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

---
block/cfq-cgroup.c | 121 ++-
include/linux/cfq-iosched.h | 6 ++
include/linux/cgroup.h | 1 +
kernel/cgroup.c | 6 ++
4 files changed, 133 insertions(+), 1 deletions(-)

diff --git a/block/cfq-cgroup.c b/block/cfq-cgroup.c
index ba0f3db..1ad9d33 100644
--- a/block/cfq-cgroup.c
+++ b/block/cfq-cgroup.c
@@ -23,6 +23,9 @@ static const int cfq_cgroup_slice_idle = HZ / 125;
struct cfq_cgroup {
struct cgroup_subsys_state css;
unsigned int ioprio;
+
+ struct rb_root sibling_tree;
+ unsigned int siblings;
};

@@ -35,6 +38,8 @@ static inline struct cfq_cgroup *cgroup_to_cfq_cgroup(struct cgroup *cont)
/*
* Add device or cgroup data functions.
*/
+struct cfq_data *__cfq_cgroup_init_queue(struct request_queue *q, void *data);
+
static struct cfq_meta_data *cfq_cgroup_init_meta_data(struct cfq_data *cfqd, struct request_queue *q)
{
struct cfq_meta_data *cfqmd;
@@ -90,16 +95,75 @@ static void cfq_meta_data_sibling_tree_add(struct cfq_meta_data *cfqmd,
cfqmd->siblings++;
cfqd->cfqmd = cfqmd;
}
-
+
+static void cfq_cgroup_sibling_tree_add(struct cfq_cgroup *cfqc,

```
+     struct cfq_data *cfqd)
+{
+ struct rb_node **p;
+ struct rb_node *parent = NULL;
+
+ BUG_ON(!RB_EMPTY_NODE(&cfqd->group_node));
+
+ p = &cfqc->sibling_tree.rb_node;
+
+ while (*p) {
+  struct cfq_data *__cfqd;
+  struct rb_node **n;
+
+  parent = *p;
+  __cfqd = rb_entry(parent, struct cfq_data, group_node);
+
+  if (cfqd->cfqmd < __cfqd->cfqmd) {
+   n = &(*p)->rb_left;
+  } else {
+   n = &(*p)->rb_right;
+  }
+  p = n;
+ }
+
+ rb_link_node(&cfqd->group_node, parent, p);
+ rb_insert_color(&cfqd->group_node, &cfqc->sibling_tree);
+ cfqc->siblings++;
+ cfqd->cfqc = cfqc;
+}
+
+static void *cfq_cgroup_init_cfq_data(struct cfq_cgroup *cfqc, struct cfq_data *cfqd)
+{
+ struct cgroup *child;
+
+ /* setting cfq_data for cfq_cgroup */
+ if (!cfqc) {
+  cfqc = cgroup_to_cfq_cgroup(get_root_subsys(&cfq_subsys));
+  cfq_cgroup_sibling_tree_add(cfqc, cfqd);
+ } else {
+  struct cfq_data *__cfqd;
+  __cfqd = __cfq_cgroup_init_queue(CFQ_DRV_UNIQ_DATA(cfqd).queue, cfqd->cfqmd);
+  if (!__cfqd)
+   return NULL;
+  cfq_cgroup_sibling_tree_add(cfqc, __cfqd);
+ }
+
+ /* check and create cfq_data for children */
+ if (cfqc->css.cgroup)
```

```
+  list_for_each_entry(child, &cfqc->css.cgroup->children, children){
+   cfq_cgroup_init_cfq_data(cgroup_to_cfq_cgroup(child), cfqd);
+  }
+
+ return cfqc;
+}
+
+
 struct cfq_data *__cfq_cgroup_init_queue(struct request_queue *q, void *data)
 {
  struct cfq_meta_data *cfqmd = (struct cfq_meta_data *)data;
  struct cfq_data *cfqd = __cfq_init_cfq_data(q);
+ int root = 0;

  if (!cfqd)
   return NULL;

  RB_CLEAR_NODE(&cfqd->sib_node);
+   RB_CLEAR_NODE(&cfqd->group_node);

  if (!cfqmd) {
         cfqmd = cfq_cgroup_init_meta_data(cfqd, q);
@@ -107,12 +171,35 @@ struct cfq_data *__cfq_cgroup_init_queue(struct request_queue *q,
void *data)
    kfree(cfqd);
    return NULL;
   }
+  root = 1;
  }
  cfq_meta_data_sibling_tree_add(cfqmd, cfqd);

+ if (root)
+  cfq_cgroup_init_cfq_data(NULL, cfqd);
+
  return cfqd;
 }

+static void *cfq_cgroup_init_cgroup(struct cfq_cgroup *cfqc, struct cgroup *parent)
+{
+ struct rb_node *p;
+ if (parent) {
+  struct cfq_cgroup *cfqc_p = cgroup_to_cfq_cgroup(parent);
+
+  p = rb_first(&cfqc_p->sibling_tree);
+  while (p != NULL) {
+   struct cfq_data *__cfqd;
+   __cfqd = rb_entry(p, struct cfq_data, group_node);
+
```

```
+   cfq_cgroup_init_cfq_data(cfqc, __cfqd);
+
+   p = rb_next(p);
+  }
+ }
+
+ return cfqc;
+}

 static struct cgroup_subsys_state *
 cfq_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
@@ -130,6 +217,11 @@ cfq_cgroup_create(struct cgroup_subsys *ss, struct cgroup *cont)
   return ERR_PTR(-ENOMEM);

  cfqc->ioprio = 3;
+ cfqc->sibling_tree = RB_ROOT;
+ cfqc->siblings = 0;
+
+ if (!cfq_cgroup_init_cgroup(cfqc, cont->parent))
+  return ERR_PTR(-ENOMEM);

  return &cfqc->css;
 }
@@ -144,6 +236,12 @@ static void cfq_cgroup_erase_meta_data_siblings(struct cfq_meta_data
*cfqmd, str
  cfqmd->siblings--;
 }

+static void cfq_cgroup_erase_cgroup_siblings(struct cfq_cgroup *cfqc, struct cfq_data *cfqd)
+{
+ rb_erase(&cfqd->group_node, &cfqc->sibling_tree);
+ cfqc->siblings--;
+}
+
 static void cfq_exit_device_group(struct cfq_meta_data *cfqmd)
 {
  struct rb_node *p, *n;
@@ -156,6 +254,7 @@ static void cfq_exit_device_group(struct cfq_meta_data *cfqmd)
   cfqd = rb_entry(p, struct cfq_data, sib_node);

   cfq_cgroup_erase_meta_data_siblings(cfqmd, cfqd);
+  cfq_cgroup_erase_cgroup_siblings(cfqd->cfqc, cfqd);
   __cfq_exit_data(cfqd);

   p = n;
@@ -170,8 +269,28 @@ static void __cfq_cgroup_exit_data(struct cfq_data *cfqd)
  kfree(cfqmd);
 }
```

```
+static void cfq_exit_cgroup(struct cfq_cgroup *cfqc)
+{
+ struct rb_node *p, *n;
+ struct cfq_data *cfqd;
+
+ p =  rb_first(&cfqc->sibling_tree);
+
+ while (p) {
+  n = rb_next(p);
+  cfqd = rb_entry(p, struct cfq_data, group_node);
+
+  cfq_cgroup_erase_meta_data_siblings(cfqd->cfqmd, cfqd);
+  cfq_cgroup_erase_cgroup_siblings(cfqc, cfqd);
+  __cfq_exit_data(cfqd);
+
+  p = n;
+ }
+}
+
 static void cfq_cgroup_destroy(struct cgroup_subsys *ss, struct cgroup *cont)
 {
+ cfq_exit_cgroup(cgroup_to_cfq_cgroup(cont));
  kfree(cgroup_to_cfq_cgroup(cont));
 }

diff --git a/include/linux/cfq-iosched.h b/include/linux/cfq-iosched.h
index c8879a7..93256ce 100644
--- a/include/linux/cfq-iosched.h
+++ b/include/linux/cfq-iosched.h
@@ -94,10 +94,16 @@ struct cfq_data {
  struct list_head cic_list;

 #ifdef CONFIG_CGROUP_CFQ
+ /* device unique attribute */
  struct cfq_meta_data *cfqmd;
  /* sibling_tree member for cfq_meta_data */
  struct rb_node sib_node;

+ /* cfq_cgroup attribute */
+ struct cfq_cgroup *cfqc;
+ /* group_tree member for cfq_cgroup */
+ struct rb_node group_node;
+
 #else
  struct cfq_driver_data cfq_driv_d;
 #endif
diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
```

```
index 785a01c..6423951 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@@ -356,6 +356,7 @@ struct task_struct *cgroup_iter_next(struct cgroup *cgrp,
 void cgroup_iter_end(struct cgroup *cgrp, struct cgroup_iter *it);
 int cgroup_scan_tasks(struct cgroup_scanner *scan);
 int cgroup_attach_task(struct cgroup *, struct task_struct *);
+struct cgroup* get_root_subsys(struct cgroup_subsys *css);

 #else /* !CONFIG_CGROUPS */

diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index e8e8ec4..1ef1de7 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -1290,6 +1290,12 @@ static int attach_task_by_pid(struct cgroup *cgrp, char *pidbuf)
  return ret;
 }

+
+struct cgroup* get_root_subsys(struct cgroup_subsys *css)
+{
+ return &css->root->top_cgroup;
+}
+
 /* The various types of files and directories in a cgroup file system */
 enum cgroup_filetype {
  FILE_ROOT,
--
1.5.4.1
```