
Subject: [RFC][PATCH 3/9] cgroups: block: cfq: I/O bandwidth controlling subsystem for CGroups based on CFQ

Posted by [Vasily Tarasov](#) on Fri, 15 Feb 2008 06:59:45 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Vasily Tarasov <vtaras@openvz.org>

Extends the original CFQ data structures and adds the major cfqio_subsys data structure: cfqio_cgroup_data. Adds several helper functions, which will be called later from CFQ code to form proper data structures interconnection.

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

```
--- linux-2.6.25-rc5-mm1/include/linux/cfqio-cgroup.h.mainstruct 2008-02-15 01:06:40.000000000
-0500
+++ linux-2.6.25-rc5-mm1/include/linux/cfqio-cgroup.h 2008-02-15 01:07:29.000000000 -0500
@@ -13,6 +13,10 @@
#ifdef _LINUX_CFQIO_CGROUP_H
#define _LINUX_CFQIO_CGROUP_H

#include <linux/list.h>
#include <linux/spinlock.h>
#include <linux/cfq-iosched.h>
+
#define CFQIO_SS_IOPRIO_DEF 4
#define CFQIO_SS_IOPRIO_MAX 7
#define CFQIO_SS_IOPRIO_MIN 0
@@ -21,6 +25,31 @@
struct cfqio_ss_css {
    struct cgroup_subsys_state css;
    unsigned int ioprio;
+ struct list_head cfqio_cgrp_head;
+ /* this lock protects the list above */
+ rwlock_t cfqio_cgrp_lock;
+ /* list of all such objects, anchored at cfqio_ss_css_list */
+ struct list_head list;
};

#ifdef CONFIG_CGROUP_CFQIO
extern struct cfqio_cgroup_data *
+cfqio_cgrp_findcreate(struct cfqio_ss_css *, struct cfq_data *, gfp_t gfp_mask);
extern void cfqio_ss_exit_queue(struct cfq_data *);
#else
+static inline struct cfqio_cgroup_data *
+cfqio_cgrp_findcreate(struct cfqio_ss_css *cfqio_ss,
```



```

+ * it belongs to, and later use this information in order to add
+ * the queue to proper lists.
+ *
+ * We can't place this structure to cfqio-cgroup.h because of include
+ * files circular dependency.
+ */
+struct cfqio_cgroup_data {
+ /* for cfqio_ss_css->cfqio_cgrp_head */
+ struct list_head cfqio_cgrp_list;
+ /* for cfqd->act_cfqio_cgrp_head */
+ struct list_head act_cfqio_cgrp_list;
+ struct cfq_data *cfqd;
+ struct cfqio_ss_css *cfqio_css;
+ /* rr list of queues with requests */
+ struct cfq_rb_root service_tree;
+};
+
+/*
+ * Per block device queue structure
+ */
+struct cfq_data {
+ struct request_queue *queue;
+
+#ifndef CONFIG_CGROUP_CFQIO
+ /* use this cgroup if CGROUP_CFQIO is off:
+  look at cfqio_cgrp_findcreate() */
+ struct cfqio_cgroup_data cfqio_cgroup;
+#endif
+ /*
+  * rr list of queues with requests and the count of them
+  */
+@@ -59,6 +109,11 @@ struct cfq_data {
+ unsigned int cfq_slice_idle;
+
+ struct list_head cic_list;
+
+ /* list of cgroups that have requests */
+ struct list_head act_cfqio_cgrp_head;
+ /* cgroup that owns a timeslice at the moment */
+ struct cfqio_cgroup_data *active_cfqio_cgroup;
+};
+
+/*
+@@ -98,6 +153,9 @@ struct cfq_queue {
+
+ /* various state flags, see below */
+ unsigned int flags;
+

```

```

+ /* cgroup/device this queue belongs to */
+ struct cfqio_cgroup_data *cfqio_cgrp;
+};

#ifdef /* _LINUX_CFQ_IOSCHED_H */
--- linux-2.6.25-rc5-mm1/block/cfqio-cgroup.c.mainstruct 2008-02-15 01:06:40.000000000 -0500
+++ linux-2.6.25-rc5-mm1/block/cfqio-cgroup.c 2008-02-15 01:07:29.000000000 -0500
@@ -10,17 +10,127 @@
*
*/

#include <linux/ioprio.h>
#include <linux/cgroup.h>
#include <linux/cfqio-cgroup.h>
#include <linux/err.h>

+LIST_HEAD(cfqio_ss_css_head);
+/* This lock protects the list above.
+ * The global order of locking is the following:
+ * 1) queue_lock
+ * 2) cfqio_ss_css_locka
+ * 3) cfqio_ss_css->cfqio_cgrp_lock
+ */
+DEFINE_SPINLOCK(cfqio_ss_css_lock);
+
+static struct cfqio_cgroup_data *
+__find_cfqio_cgrp(struct cfqio_ss_css *cfqio_css, struct cfq_data *cfqd)
+{
+ struct cfqio_cgroup_data *cfqio_cgrp;
+
+ list_for_each_entry(cfqio_cgrp, &cfqio_css->cfqio_cgrp_head,
+ cfqio_cgrp_list)
+ if (cfqio_cgrp->cfqd == cfqd)
+ return cfqio_cgrp;
+
+ return NULL;
+}
+
+struct cfqio_cgroup_data *cfqio_cgrp_findcreate(struct cfqio_ss_css *cfqio_css,
+ struct cfq_data *cfqd, gfp_t gfp_mask)
+{
+ struct cfqio_cgroup_data *cfqio_cgrp_new;
+ struct cfqio_cgroup_data *cfqio_cgrp;
+
+ read_lock(&cfqio_css->cfqio_cgrp_lock);
+ cfqio_cgrp = __find_cfqio_cgrp(cfqio_css, cfqd);
+ read_unlock(&cfqio_css->cfqio_cgrp_lock);
+

```

```

+ if (cfqio_cgrp)
+ return cfqio_cgrp;
+
+ cfqio_cgrp_new = kzalloc(sizeof(*cfqio_cgrp_new), gfp_mask);
+ if (!cfqio_cgrp_new)
+ return NULL;
+
+ cfqio_init_cfqio_cgroup(cfqio_cgrp_new);
+ cfqio_cgrp_new->cfqd = cfqd;
+ cfqio_cgrp_new->cfqio_css = cfqio_css;
+
+ write_lock(&cfqio_css->cfqio_cgrp_lock);
+ cfqio_cgrp = __find_cfqio_cgrp(cfqio_css, cfqd);
+ if (cfqio_cgrp)
+ kfree(cfqio_cgrp_new);
+ else {
+ list_add_tail(&cfqio_cgrp_new->cfqio_cgrp_list,
+ &cfqio_css->cfqio_cgrp_head);
+ cfqio_cgrp = cfqio_cgrp_new;
+ }
+ write_unlock(&cfqio_css->cfqio_cgrp_lock);
+
+ return cfqio_cgrp;
+}
+
+static void release_cfqio_cgrp(struct cfqio_cgroup_data *cfqio_cgrp)
+{
+ list_del(&cfqio_cgrp->cfqio_cgrp_list);
+ kfree(cfqio_cgrp);
+}
+
+/* called on device queue exit */
+void cfqio_ss_exit_queue(struct cfq_data *cfqd)
+{
+ struct cfqio_ss_css *cfqio_css;
+ struct cfqio_cgroup_data *cfqio_cgrp;
+
+ spin_lock(&cfqio_ss_css_lock);
+ list_for_each_entry(cfqio_css, &cfqio_ss_css_head, list) {
+ write_lock(&cfqio_css->cfqio_cgrp_lock);
+ cfqio_cgrp = __find_cfqio_cgrp(cfqio_css, cfqd);
+ if (!cfqio_cgrp) {
+ write_unlock(&cfqio_css->cfqio_cgrp_lock);
+ continue;
+ }
+ release_cfqio_cgrp(cfqio_cgrp);
+ write_unlock(&cfqio_css->cfqio_cgrp_lock);
+ }
+}

```

```

+ spin_unlock(&cfqio_ss_css_lock);
+}
+
+static void cfqio_ss_css_list_del(struct cfqio_ss_css *cfqio_css)
+{
+ spin_lock(&cfqio_ss_css_lock);
+ list_del(&cfqio_css->list);
+ spin_unlock(&cfqio_ss_css_lock);
+}
+
+static void cfqio_ss_css_list_add(struct cfqio_ss_css *cfqio_css)
+{
+ spin_lock(&cfqio_ss_css_lock);
+ list_add(&cfqio_css->list, &cfqio_ss_css_head);
+ spin_unlock(&cfqio_ss_css_lock);
+}
+
+static void cfqio_ss_fini(struct cfqio_ss_css *cfqio_css)
+{
+ struct cfqio_cgroup_data *cfqio_cgrp;
+ struct cfqio_cgroup_data *cfqio_cgrp_tmp;
+
+ cfqio_ss_css_list_del(cfqio_css);
+
+ /* no lock since cgroup is already dead */
+ list_for_each_entry_safe(cfqio_cgrp, cfqio_cgrp_tmp,
+ &cfqio_css->cfqio_cgrp_head, cfqio_cgrp_list)
+ release_cfqio_cgrp(cfqio_cgrp);
+}

static void cfqio_ss_init(struct cfqio_ss_css *cfqio_css)
{
cfqio_css->ioprio = CFQIO_SS_IOPRIO_DEF;
+ INIT_LIST_HEAD(&cfqio_css->cfqio_cgrp_head);
+ rwlock_init(&cfqio_css->cfqio_cgrp_lock);
+ cfqio_ss_css_list_add(cfqio_css);
}

static struct cgroup_subsys_state *

```
