Subject: [RFC][PATCH 1/9] cgroups: block: cfq: I/O bandwidth controlling subsystem for CGroups based on CFQ
Posted by Vasily Tarasov on Fri, 15 Feb 2008 06:59:41 GMT
View Forum Message <> Reply to Message

From: Vasily Tarasov <vtaras@openvz.org>

Simply moves CFQ data structure definitions to the header file.

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

```
--- /dev/null 2008-02-14 08:57:49.255923728 -0500
+++ linux-2.6.25-rc5-mm1/include/linux/cfq-iosched.h 2008-02-15 01:03:38.000000000 -0500
@@ -0,0 +1,103 @@
+#ifndef _LINUX_CFQ_IOSCHED_H
+#define _LINUX_CFQ_IOSCHED_H
+
+/*
+ * Most of our rbtree usage is for sorting with min extraction, so
+ * if we cache the leftmost node we don't have to walk down the tree
+ * to find it. Idea borrowed from Ingo Molnars CFS scheduler. We should
+ * move this into the elevator for the rq sorting as well.
+ */
+struct cfq_rb_root {
+ struct rb_root rb;
+ struct rb_node *left;
+};
+#define CFQ_RB_ROOT (struct cfq_rb_root) { RB_ROOT, NULL, }
+
+/*
+ * Per block device queue structure
+ */
+struct cfq_data {
+ struct request_queue *queue;
+
+ /*
+  * rr list of queues with requests and the count of them
+  */
+ struct cfq_rb_root service_tree;
+ unsigned int busy_queues;
+
+ int rq_in_driver;
+ int sync_flight;
+ int hw_tag;
+
+ /*
```

```
+  * idle window management
+  */
+ struct timer_list idle_slice_timer;
+ struct work_struct unplug_work;
+
+ struct cfq_queue *active_queue;
+ struct cfq_io_context *active_cic;
+
+ /*
+  * async queue for each priority case
+  */
+ struct cfq_queue *async_cfqq[2][IOPRIO_BE_NR];
+ struct cfq_queue *async_idle_cfqq;
+
+ sector_t last_position;
+ unsigned long last_end_request;
+
+ /*
+  * tunables, see top of file
+  */
+ unsigned int cfq_quantum;
+ unsigned int cfq_fifo_expire[2];
+ unsigned int cfq_back_penalty;
+ unsigned int cfq_back_max;
+ unsigned int cfq_slice[2];
+ unsigned int cfq_slice_async_rq;
+ unsigned int cfq_slice_idle;
+
+ struct list_head cic_list;
+};
+
+/*
+ * Per process-grouping structure
+ */
+struct cfq_queue {
+ /* reference count */
+ atomic_t ref;
+ /* parent cfq_data */
+ struct cfq_data *cfqd;
+ /* service_tree member */
+ struct rb_node rb_node;
+ /* service_tree key */
+ unsigned long rb_key;
+ /* sorted list of pending requests */
+ struct rb_root sort_list;
+ /* if fifo isn't expired, next request to serve */
+ struct request *next_rq;
+ /* requests queued in sort_list */
```

```
+ int queued[2];
+ /* currently allocated requests */
+ int allocated[2];
+ /* pending metadata requests */
+ int meta_pending;
+ /* fifo list of requests in sort_list */
+ struct list_head fifo;
+
+ unsigned long slice_end;
+ long slice_resid;
+
+ /* number of requests that are on the dispatch list or inside driver */
+ int dispatched;
+
+ /* io prio of this group */
+ unsigned short ioprio, org_ioprio;
+ unsigned short ioprio_class, org_ioprio_class;
+
+ /* various state flags, see below */
+ unsigned int flags;
+};
+
+#endif /* _LINUX_CFQ_IOSCHED_H */
--- linux-2.6.25-rc5-mm1/block/cfq-iosched.c.move 2008-02-15 00:49:11.000000000 -0500
+++ linux-2.6.25-rc5-mm1/block/cfq-iosched.c 2008-02-15 01:03:38.000000000 -0500
@@ -11,6 +11,7 @@
 #include <linux/elevator.h>
 #include <linux/rbtree.h>
 #include <linux/ioprio.h>
+#include <linux/cfq-iosched.h>

 /*
  * tunables
@@ -58,105 +59,6 @@ static struct completion *ioc_gone;

 #define sample_valid(samples) ((samples) > 80)

-/*
- * Most of our rbtree usage is for sorting with min extraction, so
- * if we cache the leftmost node we don't have to walk down the tree
- * to find it. Idea borrowed from Ingo Molnars CFS scheduler. We should
- * move this into the elevator for the rq sorting as well.
- */
-struct cfq_rb_root {
- struct rb_root rb;
- struct rb_node *left;
-};
-#define CFQ_RB_ROOT (struct cfq_rb_root) { RB_ROOT, NULL, }
```

```
-
-/*
- * Per block device queue structure
- */
-struct cfq_data {
- struct request_queue *queue;
-
- /*
-  * rr list of queues with requests and the count of them
-  */
- struct cfq_rb_root service_tree;
- unsigned int busy_queues;
-
- int rq_in_driver;
- int sync_flight;
- int hw_tag;
-
- /*
-  * idle window management
-  */
- struct timer_list idle_slice_timer;
- struct work_struct unplug_work;
-
- struct cfq_queue *active_queue;
- struct cfq_io_context *active_cic;
-
- /*
-  * async queue for each priority case
-  */
- struct cfq_queue *async_cfqq[2][IOPRIO_BE_NR];
- struct cfq_queue *async_idle_cfqq;
-
- sector_t last_position;
- unsigned long last_end_request;
-
- /*
-  * tunables, see top of file
-  */
- unsigned int cfq_quantum;
- unsigned int cfq_fifo_expire[2];
- unsigned int cfq_back_penalty;
- unsigned int cfq_back_max;
- unsigned int cfq_slice[2];
- unsigned int cfq_slice_async_rq;
- unsigned int cfq_slice_idle;
-
- struct list_head cic_list;
-};
```

```
-
-/*
- * Per process-grouping structure
- */
-struct cfq_queue {
- /* reference count */
- atomic_t ref;
- /* parent cfq_data */
- struct cfq_data *cfqd;
- /* service_tree member */
- struct rb_node rb_node;
- /* service_tree key */
- unsigned long rb_key;
- /* sorted list of pending requests */
- struct rb_root sort_list;
- /* if fifo isn't expired, next request to serve */
- struct request *next_rq;
- /* requests queued in sort_list */
- int queued[2];
- /* currently allocated requests */
- int allocated[2];
- /* pending metadata requests */
- int meta_pending;
- /* fifo list of requests in sort_list */
- struct list_head fifo;
-
- unsigned long slice_end;
- long slice_resid;
-
- /* number of requests that are on the dispatch list or inside driver */
- int dispatched;
-
- /* io prio of this group */
- unsigned short ioprio, org_ioprio;
- unsigned short ioprio_class, org_ioprio_class;
-
- /* various state flags, see below */
- unsigned int flags;
-};
-
 enum cfqq_state_flags {
  CFQ_CFQQ_FLAG_on_rr = 0, /* on round-robin busy list */
  CFQ_CFQQ_FLAG_wait_request, /* waiting for a request */
```