
Subject: [RFC][patch 2/11][CFQ-cgroup] Move header file
Posted by [Satoshi UCHIDA](#) on Tue, 01 Apr 2008 09:30:17 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch moves some data structure into header file
(include/linux/cfq-iosched.h).

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

```
diff --git a/block/cfq-iosched.c b/block/cfq-iosched.c
index 0f962ec..c1f9da9 100644
--- a/block/cfq-iosched.c
+++ b/block/cfq-iosched.c
@@ -11,6 +11,7 @@
#include <linux/elevator.h>
#include <linux/rbtree.h>
#include <linux/ioprio.h>
+#include <linux/cfq-iosched.h>

/*
 * tunables
@@ -58,65 +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
diff --git a/include/linux/cfq-iosched.h b/include/linux/cfq-iosched.h
new file mode 100644
index 0000000..cce3993
--- /dev/null
+++ b/include/linux/cfq-iosched.h
@@ -0,0 +1,70 @@
+#ifndef _LINUX_CFQ_IOSCHED_H
+#define _LINUX_CFQ_IOSCHED_H
+

```

```

+ #include <linux/rbtree.h>
+ #include <linux/list.h>
+
+ struct request_queue;
+ struct cfq_io_context;
+
+ /*
+  * 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;
+};
+
+#endif /* _LINUX_CFQ_IOSCHED_H */
```

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