Subject: [PATCH 1/5] blk-cgroup: move blk-cgroup.h in include/linux/blk-cgroup.h
Posted by Andrea Righi on Tue, 22 Feb 2011 17:12:52 GMT
View Forum Message <> Reply to Message

Move blk-cgroup.h in include/linux for generic usage.

Signed-off-by: Andrea Righi <arighi@develer.com>
---
 block/blk-cgroup.c | 2 +-
 block/blk-cgroup.h | 335 --
 block/blk-throttle.c | 2 +-
 block/cfq.h | 2 +-
 include/linux/blk-cgroup.h | 337 ++
 5 files changed, 340 insertions(+), 338 deletions(-)
 delete mode 100644 block/blk-cgroup.h
 create mode 100644 include/linux/blk-cgroup.h

diff --git a/block/blk-cgroup.c b/block/blk-cgroup.c
index 455768a..bf9d354 100644
--- a/block/blk-cgroup.c
+++ b/block/blk-cgroup.c
@@ -17,7 +17,7 @@
 #include <linux/err.h>
 #include <linux/blkdev.h>
 #include <linux/slab.h>
-#include "blk-cgroup.h"
+#include <linux/blk-cgroup.h>
 #include <linux/genhd.h>

 #define MAX_KEY_LEN 100
diff --git a/block/blk-cgroup.h b/block/blk-cgroup.h
deleted file mode 100644
index ea4861b..0000000
--- a/block/blk-cgroup.h
+++ /dev/null
@@ -1,335 +0,0 @@
-#ifndef _BLK_CGROUP_H
-#define _BLK_CGROUP_H
-/*
- * Common Block IO controller cgroup interface
- *
- * Based on ideas and code from CFQ, CFS and BFQ:
- * Copyright (C) 2003 Jens Axboe <axboe@kernel.dk>
- *
- * Copyright (C) 2008 Fabio Checconi <fabio@gandalf.sssup.it>
- *   Paolo Valente <paolo.valente@unimore.it>
- *
- * Copyright (C) 2009 Vivek Goyal <vgoyal@redhat.com>

```
- *		Nauman Rafique <nauman@google.com>
- */
-
-#include <linux/cgroup.h>
-
-enum blkio_policy_id {
- BLKIO_POLICY_PROP = 0,  /* Proportional Bandwidth division */
- BLKIO_POLICY_THROTL,  /* Throttling */
-};
-
-/* Max limits for throttle policy */
-#define THROTL_IOPS_MAX  UINT_MAX
-
-#if defined(CONFIG_BLK_CGROUP) || defined(CONFIG_BLK_CGROUP_MODULE)
-
-#ifndef CONFIG_BLK_CGROUP
-/* When blk-cgroup is a module, its subsys_id isn't a compile-time constant */
-extern struct cgroup_subsys blkio_subsys;
-#define blkio_subsys_id blkio_subsys.subsys_id
-#endif
-
-enum stat_type {
- /* Total time spent (in ns) between request dispatch to the driver and
-  * request completion for IOs doen by this cgroup. This may not be
-  * accurate when NCQ is turned on. */
- BLKIO_STAT_SERVICE_TIME = 0,
- /* Total bytes transferred */
- BLKIO_STAT_SERVICE_BYTES,
- /* Total IOs serviced, post merge */
- BLKIO_STAT_SERVICED,
- /* Total time spent waiting in scheduler queue in ns */
- BLKIO_STAT_WAIT_TIME,
- /* Number of IOs merged */
- BLKIO_STAT_MERGED,
- /* Number of IOs queued up */
- BLKIO_STAT_QUEUED,
- /* All the single valued stats go below this */
- BLKIO_STAT_TIME,
- BLKIO_STAT_SECTORS,
-#ifdef CONFIG_DEBUG_BLK_CGROUP
- BLKIO_STAT_AVG_QUEUE_SIZE,
- BLKIO_STAT_IDLE_TIME,
- BLKIO_STAT_EMPTY_TIME,
- BLKIO_STAT_GROUP_WAIT_TIME,
- BLKIO_STAT_DEQUEUE
-#endif
-};
-
```

```
-enum stat_sub_type {
- BLKIO_STAT_READ = 0,
- BLKIO_STAT_WRITE,
- BLKIO_STAT_SYNC,
- BLKIO_STAT_ASYNC,
- BLKIO_STAT_TOTAL
-};
-
-/* blkg state flags */
-enum blkg_state_flags {
- BLKG_waiting = 0,
- BLKG_idling,
- BLKG_empty,
-};
-
-/* cgroup files owned by proportional weight policy */
-enum blkcg_file_name_prop {
- BLKIO_PROP_weight = 1,
- BLKIO_PROP_weight_device,
- BLKIO_PROP_io_service_bytes,
- BLKIO_PROP_io_serviced,
- BLKIO_PROP_time,
- BLKIO_PROP_sectors,
- BLKIO_PROP_io_service_time,
- BLKIO_PROP_io_wait_time,
- BLKIO_PROP_io_merged,
- BLKIO_PROP_io_queued,
- BLKIO_PROP_avg_queue_size,
- BLKIO_PROP_group_wait_time,
- BLKIO_PROP_idle_time,
- BLKIO_PROP_empty_time,
- BLKIO_PROP_dequeue,
-};
-
-/* cgroup files owned by throttle policy */
-enum blkcg_file_name_throtl {
- BLKIO_THROTL_read_bps_device,
- BLKIO_THROTL_write_bps_device,
- BLKIO_THROTL_read_iops_device,
- BLKIO_THROTL_write_iops_device,
- BLKIO_THROTL_io_service_bytes,
- BLKIO_THROTL_io_serviced,
-};
-
-struct blkio_cgroup {
- struct cgroup_subsys_state css;
- unsigned int weight;
- spinlock_t lock;
```

- struct hlist_head blkg_list;
- struct list_head policy_list; /* list of blkio_policy_node */
-};
-
-struct blkio_group_stats {
- /* total disk time and nr sectors dispatched by this group */
- uint64_t time;
- uint64_t sectors;
- uint64_t stat_arr[BLKIO_STAT_QUEUED + 1][BLKIO_STAT_TOTAL];
-#ifdef CONFIG_DEBUG_BLK_CGROUP
- /* Sum of number of IOs queued across all samples */
- uint64_t avg_queue_size_sum;
- /* Count of samples taken for average */
- uint64_t avg_queue_size_samples;
- /* How many times this group has been removed from service tree */
- unsigned long dequeue;
-
- /* Total time spent waiting for it to be assigned a timeslice. */
- uint64_t group_wait_time;
- uint64_t start_group_wait_time;
-
- /* Time spent idling for this blkio_group */
- uint64_t idle_time;
- uint64_t start_idle_time;
- /*
- * Total time when we have requests queued and do not contain the
- * current active queue.
- */
- uint64_t empty_time;
- uint64_t start_empty_time;
- uint16_t flags;
-#endif
-};
-
-struct blkio_group {
- /* An rcu protected unique identifier for the group */
- void *key;
- struct hlist_node blkcg_node;
- unsigned short blkcg_id;
- /* Store cgroup path */
- char path[128];
- /* The device MKDEV(major, minor), this group has been created for */
- dev_t dev;
- /* policy which owns this blk group */
- enum blkio_policy_id plid;
-
- /* Need to serialize the stats in the case of reset/update */
- spinlock_t stats_lock;

```
- struct blkio_group_stats stats;
-};
-
-struct blkio_policy_node {
- struct list_head node;
- dev_t dev;
- /* This node belongs to max bw policy or porportional weight policy */
- enum blkio_policy_id plid;
- /* cgroup file to which this rule belongs to */
- int fileid;
-
- union {
-  unsigned int weight;
-  /*
-   * Rate read/write in terms of byptes per second
-   * Whether this rate represents read or write is determined
-   * by file type "fileid".
-   */
-  u64 bps;
-  unsigned int iops;
- } val;
-};
-
-extern unsigned int blkcg_get_weight(struct blkio_cgroup *blkcg,
-         dev_t dev);
-extern uint64_t blkcg_get_read_bps(struct blkio_cgroup *blkcg,
-         dev_t dev);
-extern uint64_t blkcg_get_write_bps(struct blkio_cgroup *blkcg,
-         dev_t dev);
-extern unsigned int blkcg_get_read_iops(struct blkio_cgroup *blkcg,
-         dev_t dev);
-extern unsigned int blkcg_get_write_iops(struct blkio_cgroup *blkcg,
-         dev_t dev);
-
-typedef void (blkio_unlink_group_fn) (void *key, struct blkio_group *blkg);
-
-typedef void (blkio_update_group_weight_fn) (void *key,
-   struct blkio_group *blkg, unsigned int weight);
-typedef void (blkio_update_group_read_bps_fn) (void * key,
-   struct blkio_group *blkg, u64 read_bps);
-typedef void (blkio_update_group_write_bps_fn) (void *key,
-   struct blkio_group *blkg, u64 write_bps);
-typedef void (blkio_update_group_read_iops_fn) (void *key,
-   struct blkio_group *blkg, unsigned int read_iops);
-typedef void (blkio_update_group_write_iops_fn) (void *key,
-   struct blkio_group *blkg, unsigned int write_iops);
-
-struct blkio_policy_ops {
```

```
- blkio_unlink_group_fn *blkio_unlink_group_fn;
- blkio_update_group_weight_fn *blkio_update_group_weight_fn;
- blkio_update_group_read_bps_fn *blkio_update_group_read_bps_fn;
- blkio_update_group_write_bps_fn *blkio_update_group_write_bps_fn;
- blkio_update_group_read_iops_fn *blkio_update_group_read_iops_fn;
- blkio_update_group_write_iops_fn *blkio_update_group_write_iops_fn;
-};
-
-struct blkio_policy_type {
- struct list_head list;
- struct blkio_policy_ops ops;
- enum blkio_policy_id plid;
-};
-
-/* Blkio controller policy registration */
-extern void blkio_policy_register(struct blkio_policy_type *);
-extern void blkio_policy_unregister(struct blkio_policy_type *);
-
-static inline char *blkg_path(struct blkio_group *blkg)
-{
- return blkg->path;
-}
-
-#else
-
-struct blkio_group {
-};
-
-struct blkio_policy_type {
-};
-
-static inline void blkio_policy_register(struct blkio_policy_type *blkiop) { }
-static inline void blkio_policy_unregister(struct blkio_policy_type *blkiop) { }
-
-static inline char *blkg_path(struct blkio_group *blkg) { return NULL; }
-
-#endif
-
-#define BLKIO_WEIGHT_MIN 100
-#define BLKIO_WEIGHT_MAX 1000
-#define BLKIO_WEIGHT_DEFAULT 500
-
-#ifdef CONFIG_DEBUG_BLK_CGROUP
-void blkiocg_update_avg_queue_size_stats(struct blkio_group *blkg);
-void blkiocg_update_dequeue_stats(struct blkio_group *blkg,
-    unsigned long dequeue);
-void blkiocg_update_set_idle_time_stats(struct blkio_group *blkg);
-void blkiocg_update_idle_time_stats(struct blkio_group *blkg);
```

```
-void blkiocg_set_start_empty_time(struct blkio_group *blkg);
-
-#define BLKG_FLAG_FNS(name)      \
-static inline void blkio_mark_blkg_##name(    \
-  struct blkio_group_stats *stats)   \
-{        \
- stats->flags |= (1 << BLKG_##name);    \
-}        \
-static inline void blkio_clear_blkg_##name(    \
-  struct blkio_group_stats *stats)   \
-{        \
- stats->flags &= ~(1 << BLKG_##name);    \
-}        \
-static inline int blkio_blkg_##name(struct blkio_group_stats *stats) \
-{        \
- return (stats->flags & (1 << BLKG_##name)) != 0;  \
-}        \
-
-BLKG_FLAG_FNS(waiting)
-BLKG_FLAG_FNS(idling)
-BLKG_FLAG_FNS(empty)
-#undef BLKG_FLAG_FNS
-#else
-static inline void blkiocg_update_avg_queue_size_stats(
-      struct blkio_group *blkg) {}
-static inline void blkiocg_update_dequeue_stats(struct blkio_group *blkg,
-      unsigned long dequeue) {}
-static inline void blkiocg_update_set_idle_time_stats(struct blkio_group *blkg)
-{}
-static inline void blkiocg_update_idle_time_stats(struct blkio_group *blkg) {}
-static inline void blkiocg_set_start_empty_time(struct blkio_group *blkg) {}
-#endif
-
-#if defined(CONFIG_BLK_CGROUP) || defined(CONFIG_BLK_CGROUP_MODULE)
-extern struct blkio_cgroup blkio_root_cgroup;
-extern struct blkio_cgroup *cgroup_to_blkio_cgroup(struct cgroup *cgroup);
-extern void blkiocg_add_blkio_group(struct blkio_cgroup *blkcg,
- struct blkio_group *blkg, void *key, dev_t dev,
- enum blkio_policy_id plid);
-extern int blkiocg_del_blkio_group(struct blkio_group *blkg);
-extern struct blkio_group *blkiocg_lookup_group(struct blkio_cgroup *blkcg,
-      void *key);
-void blkiocg_update_timeslice_used(struct blkio_group *blkg,
-     unsigned long time);
-void blkiocg_update_dispatch_stats(struct blkio_group *blkg, uint64_t bytes,
-      bool direction, bool sync);
-void blkiocg_update_completion_stats(struct blkio_group *blkg,
- uint64_t start_time, uint64_t io_start_time, bool direction, bool sync);
```

```
-void blkiocg_update_io_merged_stats(struct blkio_group *blkg, bool direction,
-     bool sync);
-void blkiocg_update_io_add_stats(struct blkio_group *blkg,
-  struct blkio_group *curr_blkg, bool direction, bool sync);
-void blkiocg_update_io_remove_stats(struct blkio_group *blkg,
-     bool direction, bool sync);
-#else
-struct cgroup;
-static inline struct blkio_cgroup *
-cgroup_to_blkio_cgroup(struct cgroup *cgroup) { return NULL; }
-
-static inline void blkiocg_add_blkio_group(struct blkio_cgroup *blkcg,
-  struct blkio_group *blkg, void *key, dev_t dev,
-  enum blkio_policy_id plid) {}
-
-static inline int
-blkiocg_del_blkio_group(struct blkio_group *blkg) { return 0; }
-
-static inline struct blkio_group *
-blkiocg_lookup_group(struct blkio_cgroup *blkcg, void *key) { return NULL; }
-static inline void blkiocg_update_timeslice_used(struct blkio_group *blkg,
-      unsigned long time) {}
-static inline void blkiocg_update_dispatch_stats(struct blkio_group *blkg,
-    uint64_t bytes, bool direction, bool sync) {}
-static inline void blkiocg_update_completion_stats(struct blkio_group *blkg,
-  uint64_t start_time, uint64_t io_start_time, bool direction,
-  bool sync) {}
-static inline void blkiocg_update_io_merged_stats(struct blkio_group *blkg,
-      bool direction, bool sync) {}
-static inline void blkiocg_update_io_add_stats(struct blkio_group *blkg,
-  struct blkio_group *curr_blkg, bool direction, bool sync) {}
-static inline void blkiocg_update_io_remove_stats(struct blkio_group *blkg,
-      bool direction, bool sync) {}
-#endif
-#endif /* _BLK_CGROUP_H */
diff --git a/block/blk-throttle.c b/block/blk-throttle.c
index a89043a..9ad3d1e 100644
--- a/block/blk-throttle.c
+++ b/block/blk-throttle.c
@@ -9,7 +9,7 @@
 #include <linux/blkdev.h>
 #include <linux/bio.h>
 #include <linux/blktrace_api.h>
-#include "blk-cgroup.h"
+#+#include <linux/blk-cgroup.h>

 /* Max dispatch from a group in 1 round */
 static int throtl_grp_quantum = 8;
```

```
diff --git a/block/cfq.h b/block/cfq.h
index 54a6d90..1213f9b 100644
--- a/block/cfq.h
+++ b/block/cfq.h
@@ -1,6 +1,6 @@
 #ifndef _CFQ_H
 #define _CFQ_H
-#include "blk-cgroup.h"
+#include <linux/blk-cgroup.h>

 #ifdef CONFIG_CFQ_GROUP_IOSCHED
 static inline void cfq_blkiocg_update_io_add_stats(struct blkio_group *blkg,
diff --git a/include/linux/blk-cgroup.h b/include/linux/blk-cgroup.h
new file mode 100644
index 0000000..5e48204
--- /dev/null
+++ b/include/linux/blk-cgroup.h
@@ -0,0 +1,337 @@
+#ifndef _BLK_CGROUP_H
+#define _BLK_CGROUP_H
+/*
+ * Common Block IO controller cgroup interface
+ *
+ * Based on ideas and code from CFQ, CFS and BFQ:
+ * Copyright (C) 2003 Jens Axboe <axboe@kernel.dk>
+ *
+ * Copyright (C) 2008 Fabio Checconi <fabio@gandalf.sssup.it>
+ *                    Paolo Valente <paolo.valente@unimore.it>
+ *
+ * Copyright (C) 2009 Vivek Goyal <vgoyal@redhat.com>
+ *                    Nauman Rafique <nauman@google.com>
+ */
+
+#include <linux/cgroup.h>
+
+enum blkio_policy_id {
+ BLKIO_POLICY_PROP = 0,  /* Proportional Bandwidth division */
+ BLKIO_POLICY_THROTL,  /* Throttling */
+};
+
+/* Max limits for throttle policy */
+#define THROTL_IOPS_MAX  UINT_MAX
+
+#if defined(CONFIG_BLK_CGROUP) || defined(CONFIG_BLK_CGROUP_MODULE)
+
+#ifndef CONFIG_BLK_CGROUP
+/* When blk-cgroup is a module, its subsys_id isn't a compile-time constant */
+extern struct cgroup_subsys blkio_subsys;
```

```
+#define blkio_subsys_id blkio_subsys.subsys_id
+#endif
+
+enum stat_type {
+ /* Total time spent (in ns) between request dispatch to the driver and
+  * request completion for IOs doen by this cgroup. This may not be
+  * accurate when NCQ is turned on. */
+ BLKIO_STAT_SERVICE_TIME = 0,
+ /* Total bytes transferred */
+ BLKIO_STAT_SERVICE_BYTES,
+ /* Total IOs serviced, post merge */
+ BLKIO_STAT_SERVICED,
+ /* Total time spent waiting in scheduler queue in ns */
+ BLKIO_STAT_WAIT_TIME,
+ /* Number of IOs merged */
+ BLKIO_STAT_MERGED,
+ /* Number of IOs queued up */
+ BLKIO_STAT_QUEUED,
+ /* All the single valued stats go below this */
+ BLKIO_STAT_TIME,
+ BLKIO_STAT_SECTORS,
+#ifdef CONFIG_DEBUG_BLK_CGROUP
+ BLKIO_STAT_AVG_QUEUE_SIZE,
+ BLKIO_STAT_IDLE_TIME,
+ BLKIO_STAT_EMPTY_TIME,
+ BLKIO_STAT_GROUP_WAIT_TIME,
+ BLKIO_STAT_DEQUEUE
+#endif
+};
+
+enum stat_sub_type {
+ BLKIO_STAT_READ = 0,
+ BLKIO_STAT_WRITE,
+ BLKIO_STAT_SYNC,
+ BLKIO_STAT_ASYNC,
+ BLKIO_STAT_TOTAL
+};
+
+/* blkg state flags */
+enum blkg_state_flags {
+ BLKG_waiting = 0,
+ BLKG_idling,
+ BLKG_empty,
+};
+
+/* cgroup files owned by proportional weight policy */
+enum blkcg_file_name_prop {
+ BLKIO_PROP_weight = 1,
```

```
+ BLKIO_PROP_weight_device,
+ BLKIO_PROP_io_service_bytes,
+ BLKIO_PROP_io_serviced,
+ BLKIO_PROP_time,
+ BLKIO_PROP_sectors,
+ BLKIO_PROP_io_service_time,
+ BLKIO_PROP_io_wait_time,
+ BLKIO_PROP_io_merged,
+ BLKIO_PROP_io_queued,
+ BLKIO_PROP_avg_queue_size,
+ BLKIO_PROP_group_wait_time,
+ BLKIO_PROP_idle_time,
+ BLKIO_PROP_empty_time,
+ BLKIO_PROP_dequeue,
+};
+
+/* cgroup files owned by throttle policy */
+enum blkcg_file_name_throtl {
+ BLKIO_THROTL_read_bps_device,
+ BLKIO_THROTL_write_bps_device,
+ BLKIO_THROTL_read_iops_device,
+ BLKIO_THROTL_write_iops_device,
+ BLKIO_THROTL_io_service_bytes,
+ BLKIO_THROTL_io_serviced,
+};
+
+struct blkio_cgroup {
+ struct cgroup_subsys_state css;
+ unsigned int weight;
+ spinlock_t lock;
+ struct hlist_head blkg_list;
+ struct list_head policy_list; /* list of blkio_policy_node */
+};
+
+struct blkio_group_stats {
+ /* total disk time and nr sectors dispatched by this group */
+ uint64_t time;
+ uint64_t sectors;
+ uint64_t stat_arr[BLKIO_STAT_QUEUED + 1][BLKIO_STAT_TOTAL];
+#ifdef CONFIG_DEBUG_BLK_CGROUP
+ /* Sum of number of IOs queued across all samples */
+ uint64_t avg_queue_size_sum;
+ /* Count of samples taken for average */
+ uint64_t avg_queue_size_samples;
+ /* How many times this group has been removed from service tree */
+ unsigned long dequeue;
+
+ /* Total time spent waiting for it to be assigned a timeslice. */
```

```
+ uint64_t group_wait_time;
+ uint64_t start_group_wait_time;
+
+ /* Time spent idling for this blkio_group */
+ uint64_t idle_time;
+ uint64_t start_idle_time;
+ /*
+  * Total time when we have requests queued and do not contain the
+  * current active queue.
+  */
+ uint64_t empty_time;
+ uint64_t start_empty_time;
+ uint16_t flags;
+#endif
+};
+
+struct blkio_group {
+ /* An rcu protected unique identifier for the group */
+ void *key;
+ struct hlist_node blkcg_node;
+ unsigned short blkcg_id;
+ /* Store cgroup path */
+ char path[128];
+ /* The device MKDEV(major, minor), this group has been created for */
+ dev_t dev;
+ /* policy which owns this blk group */
+ enum blkio_policy_id plid;
+
+ /* Need to serialize the stats in the case of reset/update */
+ spinlock_t stats_lock;
+ struct blkio_group_stats stats;
+};
+
+struct blkio_policy_node {
+ struct list_head node;
+ dev_t dev;
+ /* This node belongs to max bw policy or porportional weight policy */
+ enum blkio_policy_id plid;
+ /* cgroup file to which this rule belongs to */
+ int fileid;
+
+ union {
+  unsigned int weight;
+  /*
+   * Rate read/write in terms of byptes per second
+   * Whether this rate represents read or write is determined
+   * by file type "fileid".
+   */
```

```
+  u64 bps;
+  unsigned int iops;
+ } val;
+};
+
+extern unsigned int blkcg_get_weight(struct blkio_cgroup *blkcg,
+        dev_t dev);
+extern uint64_t blkcg_get_read_bps(struct blkio_cgroup *blkcg,
+        dev_t dev);
+extern uint64_t blkcg_get_write_bps(struct blkio_cgroup *blkcg,
+        dev_t dev);
+extern unsigned int blkcg_get_read_iops(struct blkio_cgroup *blkcg,
+        dev_t dev);
+extern unsigned int blkcg_get_write_iops(struct blkio_cgroup *blkcg,
+        dev_t dev);
+
+typedef void (blkio_unlink_group_fn) (void *key, struct blkio_group *blkg);
+
+typedef void (blkio_update_group_weight_fn) (void *key,
+   struct blkio_group *blkg, unsigned int weight);
+typedef void (blkio_update_group_read_bps_fn) (void * key,
+   struct blkio_group *blkg, u64 read_bps);
+typedef void (blkio_update_group_write_bps_fn) (void *key,
+   struct blkio_group *blkg, u64 write_bps);
+typedef void (blkio_update_group_read_iops_fn) (void *key,
+   struct blkio_group *blkg, unsigned int read_iops);
+typedef void (blkio_update_group_write_iops_fn) (void *key,
+   struct blkio_group *blkg, unsigned int write_iops);
+
+struct blkio_policy_ops {
+ blkio_unlink_group_fn *blkio_unlink_group_fn;
+ blkio_update_group_weight_fn *blkio_update_group_weight_fn;
+ blkio_update_group_read_bps_fn *blkio_update_group_read_bps_fn;
+ blkio_update_group_write_bps_fn *blkio_update_group_write_bps_fn;
+ blkio_update_group_read_iops_fn *blkio_update_group_read_iops_fn;
+ blkio_update_group_write_iops_fn *blkio_update_group_write_iops_fn;
+};
+
+struct blkio_policy_type {
+ struct list_head list;
+ struct blkio_policy_ops ops;
+ enum blkio_policy_id plid;
+};
+
+/* Blkio controller policy registration */
+extern void blkio_policy_register(struct blkio_policy_type *);
+extern void blkio_policy_unregister(struct blkio_policy_type *);
+
```

```
+static inline char *blkg_path(struct blkio_group *blkg)
+{
+ return blkg->path;
+}
+
+#else
+
+struct blkio_group {
+};
+
+struct blkio_policy_type {
+};
+
+static inline void blkio_policy_register(struct blkio_policy_type *blkiop) { }
+static inline void blkio_policy_unregister(struct blkio_policy_type *blkiop) { }
+
+static inline char *blkg_path(struct blkio_group *blkg) { return NULL; }
+
+#endif
+
+#define BLKIO_WEIGHT_MIN 100
+#define BLKIO_WEIGHT_MAX 1000
+#define BLKIO_WEIGHT_DEFAULT 500
+
+#ifdef CONFIG_DEBUG_BLK_CGROUP
+void blkiocg_update_avg_queue_size_stats(struct blkio_group *blkg);
+void blkiocg_update_dequeue_stats(struct blkio_group *blkg,
+    unsigned long dequeue);
+void blkiocg_update_set_idle_time_stats(struct blkio_group *blkg);
+void blkiocg_update_idle_time_stats(struct blkio_group *blkg);
+void blkiocg_set_start_empty_time(struct blkio_group *blkg);
+
+#define BLKG_FLAG_FNS(name)      \
+static inline void blkio_mark_blkg_##name(    \
+  struct blkio_group_stats *stats)   \
+{        \
+ stats->flags |= (1 << BLKG_##name);    \
+}        \
+static inline void blkio_clear_blkg_##name(    \
+  struct blkio_group_stats *stats)   \
+{        \
+ stats->flags &= ~(1 << BLKG_##name);    \
+}        \
+static inline int blkio_blkg_##name(struct blkio_group_stats *stats) \
+{        \
+ return (stats->flags & (1 << BLKG_##name)) != 0;  \
+}        \
+
```

```
+BLKG_FLAG_FNS(waiting)
+BLKG_FLAG_FNS(idling)
+BLKG_FLAG_FNS(empty)
+#undef BLKG_FLAG_FNS
+#else
+static inline void blkiocg_update_avg_queue_size_stats(
+       struct blkio_group *blkg) {}
+static inline void blkiocg_update_dequeue_stats(struct blkio_group *blkg,
+       unsigned long dequeue) {}
+static inline void blkiocg_update_set_idle_time_stats(struct blkio_group *blkg)
+{}
+static inline void blkiocg_update_idle_time_stats(struct blkio_group *blkg) {}
+static inline void blkiocg_set_start_empty_time(struct blkio_group *blkg) {}
+#endif
+
+#if defined(CONFIG_BLK_CGROUP) || defined(CONFIG_BLK_CGROUP_MODULE)
+extern struct blkio_cgroup blkio_root_cgroup;
+extern bool blkio_cgroup_disabled(void);
+extern struct blkio_cgroup *cgroup_to_blkio_cgroup(struct cgroup *cgroup);
+extern void blkiocg_add_blkio_group(struct blkio_cgroup *blkcg,
+ struct blkio_group *blkg, void *key, dev_t dev,
+ enum blkio_policy_id plid);
+extern int blkiocg_del_blkio_group(struct blkio_group *blkg);
+extern struct blkio_group *blkiocg_lookup_group(struct blkio_cgroup *blkcg,
+       void *key);
+void blkiocg_update_timeslice_used(struct blkio_group *blkg,
+      unsigned long time);
+void blkiocg_update_dispatch_stats(struct blkio_group *blkg, uint64_t bytes,
+       bool direction, bool sync);
+void blkiocg_update_completion_stats(struct blkio_group *blkg,
+ uint64_t start_time, uint64_t io_start_time, bool direction, bool sync);
+void blkiocg_update_io_merged_stats(struct blkio_group *blkg, bool direction,
+      bool sync);
+void blkiocg_update_io_add_stats(struct blkio_group *blkg,
+   struct blkio_group *curr_blkg, bool direction, bool sync);
+void blkiocg_update_io_remove_stats(struct blkio_group *blkg,
+       bool direction, bool sync);
+#else
+struct cgroup;
+static inline bool blkio_cgroup_disabled(void) { return true; }
+static inline struct blkio_cgroup *
+cgroup_to_blkio_cgroup(struct cgroup *cgroup) { return NULL; }
+
+static inline void blkiocg_add_blkio_group(struct blkio_cgroup *blkcg,
+   struct blkio_group *blkg, void *key, dev_t dev,
+   enum blkio_policy_id plid) {}
+
+static inline int
```

```
+blkiocg_del_blkio_group(struct blkio_group *blkg) { return 0; }
+
+static inline struct blkio_group *
+blkiocg_lookup_group(struct blkio_cgroup *blkcg, void *key) { return NULL; }
+static inline void blkiocg_update_timeslice_used(struct blkio_group *blkg,
+       unsigned long time) {}
+static inline void blkiocg_update_dispatch_stats(struct blkio_group *blkg,
+     uint64_t bytes, bool direction, bool sync) {}
+static inline void blkiocg_update_completion_stats(struct blkio_group *blkg,
+  uint64_t start_time, uint64_t io_start_time, bool direction,
+  bool sync) {}
+static inline void blkiocg_update_io_merged_stats(struct blkio_group *blkg,
+       bool direction, bool sync) {}
+static inline void blkiocg_update_io_add_stats(struct blkio_group *blkg,
+  struct blkio_group *curr_blkg, bool direction, bool sync) {}
+static inline void blkiocg_update_io_remove_stats(struct blkio_group *blkg,
+       bool direction, bool sync) {}
+#endif
+#endif /* _BLK_CGROUP_H */
--
1.7.1
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containe rs