Subject: [PATCH 2/9] task group priority handling
Posted by Srivatsa Vaddagiri on Thu, 12 Apr 2007 17:53:58 GMT
View Forum Message <> Reply to Message

To retain good interactivity, priority of a task-group is defined to be the
same as the priority of the highest-priority runnable task it has in its
runqueue.

This means as tasks come and go, priority of a task-group can change.

This patch deals with such changes to task-group priority.

P.S : Is this a good idea? Don't know. Other option is to to have a static
task-group priority which is decided by its quota.

Signed-off-by : Srivatsa Vaddagiri <vatsa@in.ibm.com>

1 file changed, 78 insertions(+), 2 deletions(-)

```
diff -puN kernel/sched.c~rfc kernel/sched.c
--- linux-2.6.20/kernel/sched.c~rfc 2007-04-11 12:00:48.000000000 +0530
+++ linux-2.6.20-vatsa/kernel/sched.c 2007-04-12 11:07:18.000000000 +0530
@@ -192,7 +192,7 @@ static inline unsigned int task_timeslic

 struct prio_array {
  unsigned int nr_active;
- int best_static_prio;
+ int best_static_prio, best_dyn_prio;
  DECLARE_BITMAP(bitmap, MAX_PRIO+1); /* include 1 bit for delimiter */
  struct list_head queue[MAX_PRIO];
 };
@@ -729,6 +729,55 @@ sched_info_switch(struct task_struct *pr
 #define sched_info_switch(t, next) do { } while (0)
 #endif /* CONFIG_SCHEDSTATS || CONFIG_TASK_DELAY_ACCT */

+/* Dequeue task-group object from the main per-cpu runqueue */
+static void dequeue_task_grp(struct task_grp_rq *tgrq)
+{
+ struct prio_array *array = tgrq->rq_array;
+
+ array->nr_active--;
+ list_del(&tgrq->list);
+ if (list_empty(array->queue + tgrq->prio))
```

```
+  __clear_bit(tgrq->prio, array->bitmap);
+ tgrq->rq_array = NULL;
+}
+
+/* Enqueue task-group object on the main per-cpu runqueue */
+static void enqueue_task_grp(struct task_grp_rq *tgrq, struct prio_array *array,
+    int head)
+{
+ if (!head)
+  list_add_tail(&tgrq->list, array->queue + tgrq->prio);
+ else
+         list_add(&tgrq->list, array->queue + tgrq->prio);
+         __set_bit(tgrq->prio, array->bitmap);
+        array->nr_active++;
+        tgrq->rq_array = array;
+}
+
+/* Priority of a task-group is defined to be the priority of the highest
+ * priority runnable task it has. As tasks belonging to a task-group come in
+ * and go, the task-group priority can change on a particular CPU.
+ */
+static inline void update_task_grp_prio(struct task_grp_rq *tgrq, struct rq *rq,
+         int head)
+{
+ int new_prio;
+ struct prio_array *array = tgrq->rq_array;
+
+ new_prio = tgrq->active->best_dyn_prio;
+ if (new_prio == MAX_PRIO)
+  new_prio = tgrq->expired->best_dyn_prio;
+
+ if (array)
+  dequeue_task_grp(tgrq);
+ tgrq->prio = new_prio;
+ if (new_prio != MAX_PRIO) {
+  if (!array)
+   array = rq->active;
+  enqueue_task_grp(tgrq, array, head);
+ }
+}
+
 /*
  * Adding/removing a task to/from a priority array:
  */
@@ -736,8 +785,17 @@ static void dequeue_task(struct task_str
 {
  array->nr_active--;
  list_del(&p->run_list);
```

```
- if (list_empty(array->queue + p->prio))
+ if (list_empty(array->queue + p->prio)) {
   __clear_bit(p->prio, array->bitmap);
+  if (p->prio == array->best_dyn_prio) {
+   struct task_grp_rq *tgrq = task_grp_rq(p);
+
+   array->best_dyn_prio =
+     sched_find_first_bit(array->bitmap);
+   if (array == tgrq->active || !tgrq->active->nr_active)
+    update_task_grp_prio(tgrq, task_rq(p), 0);
+  }
+ }
 }

 static void enqueue_task(struct task_struct *p, struct prio_array *array)
@@ -747,6 +805,14 @@ static void enqueue_task(struct task_str
  __set_bit(p->prio, array->bitmap);
  array->nr_active++;
  p->array = array;
+
+ if (p->prio < array->best_dyn_prio) {
+  struct task_grp_rq *tgrq = task_grp_rq(p);
+
+  array->best_dyn_prio = p->prio;
+  if (array == tgrq->active || !tgrq->active->nr_active)
+   update_task_grp_prio(tgrq, task_rq(p), 0);
+ }
 }

 /*
@@ -765,6 +831,14 @@ enqueue_task_head(struct task_struct *p,
  __set_bit(p->prio, array->bitmap);
  array->nr_active++;
  p->array = array;
+
+ if (p->prio < array->best_dyn_prio) {
+  struct task_grp_rq *tgrq = task_grp_rq(p);
+
+  array->best_dyn_prio = p->prio;
+  if (array == tgrq->active || !tgrq->active->nr_active)
+   update_task_grp_prio(tgrq, task_rq(p), 1);
+ }
 }

 /*
@@ -7010,7 +7084,9 @@ static void task_grp_rq_init(struct task
  tgrq->expired = tgrq->arrays + 1;
  tgrq->rq_array = NULL;
```

```
  tgrq->expired->best_static_prio = MAX_PRIO;
+ tgrq->expired->best_dyn_prio = MAX_PRIO;
  tgrq->active->best_static_prio = MAX_PRIO;
+ tgrq->active->best_dyn_prio = MAX_PRIO;
  tgrq->prio = MAX_PRIO;
  tgrq->tg = tg;
  INIT_LIST_HEAD(&tgrq->list);
```

_
--
Regards,
vatsa

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

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