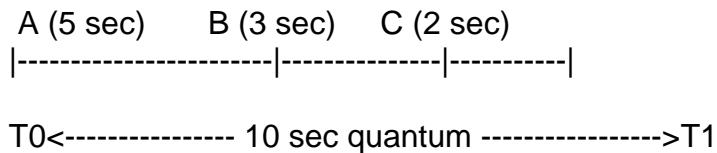


This patch does the actual job of dividing up CPU time among various task-groups as per their quota.

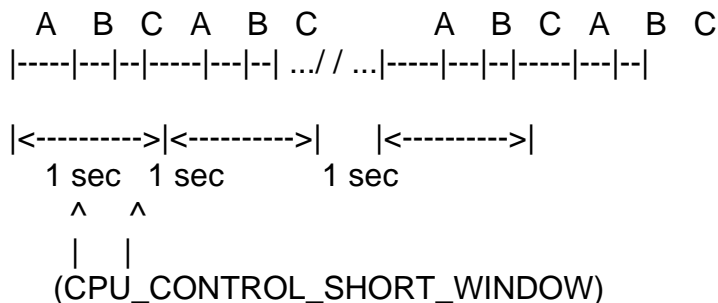
High-level overview:

Lets say that we have three task-groups/classes A, B and C whose quota is 50%, 30% and 20%. The quota specifies how much of execution time each group gets per some quantum. If 10 seconds is chosen as this quantum, then group A's tasks get 5 seconds worth of execution time -every- 10 seconds, grp B gets 3 seconds and so on, as shown below:



In this scheme, grp C's tasks could starve potentially, waiting for their turn.

This patch avoids the above problem by breaking the 10 sec quantum into several smaller quantum. A, B and C get their due share in each smaller quantum and hence each group progress more quickly than in the above scheme.



The latter scheme is implemented by maintaining two counters (tokens) for each task-group (on each CPU).

- o ticks -> which is exhausted over the short quantum
- o long_ticks -> which is exhausted over the longer quantum

To begin with:

```
grp->ticks = grp->quota * CPU_CONTROL_SHORT_WINDOW * HZ
grp->long_ticks = CPU_CONTROL_LONG_WINDOW/CPU_CONTROL_SHORT_WINDOW
```

grp->ticks is decremented every timer ticks. When it hits zero, grp->long_ticks is decremented. If resulting grp->long_ticks is non-zero, grp->ticks wraps back to the initial value and the group is put in an expired bucket. Else if resulting grp->long_ticks is zero, the group is put in a greedy bucket.

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

1 file changed, 115 insertions(+), 9 deletions(-)

```
diff -puN kernel/sched.c~grp_timeslice kernel/sched.c
--- linux-2.6.20/kernel/sched.c~grp_timeslice 2007-04-11 12:00:49.000000000 +0530
+++ linux-2.6.20-vatsa/kernel/sched.c 2007-04-12 11:07:18.000000000 +0530
@@ -158,9 +158,6 @@
 (JIFFIES_TO_NS(MAX_SLEEP_AVG * \
 (MAX_BONUS / 2 + DELTA((p)) + 1) / MAX_BONUS - 1))

-#define TASK_PREEMPTS_CURR(p, rq) \
- ((p)->prio < (rq)->curr->prio)
-
#define SCALE_PRIO(x, prio) \
max(x * (MAX_PRIO - prio) / (MAX_USER_PRIO / 2), MIN_TIMESLICE)

@@ -218,7 +215,9 @@ struct task_grp_rq {
 struct prio_array arrays[2];
 struct prio_array *active, *expired, *rq_array;
 unsigned long expired_timestamp;
- int prio; /* Priority of the task-group */
+ unsigned short ticks, long_ticks;
+ int prio; /* Priority of the task-group */
+ unsigned long last_update;
 struct list_head list;
 struct task_grp *tg;
};
@@ -227,6 +226,7 @@ static DEFINE_PER_CPU(struct task_grp_rq
```

```

/* task-group object - maintains information about each task-group */
struct task_grp {
+ unsigned short ticks, long_ticks; /* bandwidth given to task-group */
  struct task_grp_rq *rq[NR_CPUS]; /* runqueue pointer for every cpu */
};

@@ -270,7 +270,9 @@ struct rq {
  /* these arrays hold task-groups.
   * xxx: Avoid this for !CONFIG_CPUMETER?
   */
- struct prio_array *active, *expired, arrays[2];
+ struct prio_array *active, *expired, *greedy_active, *greedy_expired;
+ struct prio_array arrays[4];
+ unsigned long last_update;
  atomic_t nr_iowait;

#ifdef CONFIG_SMP
@@ -729,6 +731,14 @@ sched_info_switch(struct task_struct *pr
#define sched_info_switch(t, next) do { } while (0)
#endif /* CONFIG_SCHEDSTATS || CONFIG_TASK_DELAY_ACCT */

+#define CPU_CONTROL_SHORT_WINDOW      (HZ)
+#define CPU_CONTROL_LONG_WINDOW      (5 * HZ)
+#define NUM_LONG_TICKS (unsigned short) (CPU_CONTROL_LONG_WINDOW /\
+ CPU_CONTROL_SHORT_WINDOW)
+
+#define greedy_grp(grp) (!grp->long_ticks)
+#define greedy_task(p) (greedy_grp(task_grp_rq(p)))
+
  /* Dequeue task-group object from the main per-cpu runqueue */
  static void dequeue_task_grp(struct task_grp_rq *tgrq)
  {
@@ -772,12 +782,29 @@ static inline void update_task_grp_prio(
  dequeue_task_grp(tgrq);
  tgrq->prio = new_prio;
  if (new_prio != MAX_PRIO) {
- if (!array)
+ if (!greedy_grp(tgrq))
    array = rq->active;
+ else
+ array = rq->greedy_active;
  enqueue_task_grp(tgrq, array, head);
  }
}

#ifdef CONFIG_CPUMETER
+

```

```

+#define TASK_PREEMPTS_CURR(p, rq) \
+ (idle_cpu(task_cpu(p)) || \
+ (greedy_task((rq)->curr) && !greedy_task(p)) || \
+ ((task_grp_rq(p)->rq_array == task_grp_rq((rq)->curr)->rq_array) && \
+      (((p)->prio < (rq)->curr->prio))))
+
+#else
+
+#define TASK_PREEMPTS_CURR(p, rq) \
+      ((p)->prio < (rq)->curr->prio)
+
+#endif
+
/*
 * Adding/removing a task to/from a priority array:
 */
@@ -3273,6 +3300,29 @@ void account_steal_time(struct task_stru
}
}

+#ifdef CONFIG_CPUMETER
+static inline void move_task_grp_list(struct prio_array *src,
+ struct prio_array *dst)
+{
+ unsigned int i;
+
+ if (!src->nr_active)
+ return;
+
+ for (i = 0; i < MAX_PRIO; i++) {
+ struct list_head *list = src->queue + i;
+
+ while (!list_empty(list)) {
+ struct task_grp_rq *tgrq;
+
+ tgrq = list_entry(list->next, struct task_grp_rq, list);
+ dequeue_task_grp(tgrq);
+ enqueue_task_grp(tgrq, dst, 0);
+ }
+ }
+}
+#endif
+
+static void task_running_tick(struct rq *rq, struct task_struct *p)
+{
+ struct task_grp_rq *tgrq = task_grp_rq(p);
@@ -3347,6 +3397,39 @@ static void task_running_tick(struct rq
}

```

```

}
out_unlock:
#ifdef CONFIG_CPUMETER
+ if (!--tgrq->ticks) {
+     struct prio_array *target;
+
+     if (tgrq->long_ticks) {
+         --tgrq->long_ticks;
+         if (tgrq->long_ticks)
+             target = rq->expired;
+         else
+             target = rq->greedy_active;
+     } else /* should be in rq->greedy_active */
+         target = rq->greedy_expired;
+
+     /* Move the task group to expired list */
+     dequeue_task_grp(tgrq);
+     tgrq->ticks = task_grp(p)->ticks;
+     enqueue_task_grp(tgrq, target, 0);
+     set_tsk_need_resched(p);
+ }
+
+ if (unlikely(jiffies - rq->last_update > CPU_CONTROL_LONG_WINDOW)) {
+     if (rq->active->nr_active || rq->expired->nr_active) {
+         move_task_grp_list(rq->greedy_active, rq->active);
+         move_task_grp_list(rq->greedy_expired, rq->active);
+     } else {
+         switch_array(rq->active, rq->greedy_active);
+         switch_array(rq->expired, rq->greedy_expired);
+     }
+     rq->last_update = jiffies;
+     set_tsk_need_resched(p);
+ }
#endif
+
+ spin_unlock(&rq->lock);
}

```

@@ -3643,12 +3726,27 @@ need_resched_nonpreemptible:

```

#ifdef CONFIG_CPUMETER
array = rq->active;
if (unlikely(!array->nr_active)) {
- switch_array(rq->active, rq->expired);
- array = rq->active;
+ if (rq->expired->nr_active) {
+     switch_array(rq->active, rq->expired);
+     array = rq->active;
+ } else {

```

```

+ array = rq->greedy_active;
+ if (!array->nr_active) {
+   switch_array(rq->greedy_active,
+   rq->greedy_expired);
+   array = rq->greedy_active;
+ }
+ }
}
idx = sched_find_first_bit(array->bitmap);
queue = array->queue + idx;
next_grp = list_entry(queue->next, struct task_grp_rq, list);
+
+ if (unlikely(next_grp->last_update != rq->last_update)) {
+ next_grp->ticks = next_grp->tg->ticks;
+ next_grp->long_ticks = next_grp->tg->long_ticks;
+ next_grp->last_update = rq->last_update;
+ }
#else
next_grp = init_task_grp_rq[cpu];
#endif
@@ -7088,6 +7186,8 @@ static void task_grp_rq_init(struct task
tgrq->active->best_static_prio = MAX_PRIO;
tgrq->active->best_dyn_prio = MAX_PRIO;
tgrq->prio = MAX_PRIO;
+ tgrq->ticks = tg->ticks;
+ tgrq->long_ticks = tg->long_ticks;
tgrq->tg = tg;
INIT_LIST_HEAD(&tgrq->list);

@@ -7108,6 +7208,9 @@ void __init sched_init(void)
{
int i, j;

+ init_task_grp.ticks = CPU_CONTROL_SHORT_WINDOW; /* 100% bandwidth */
+ init_task_grp.long_ticks = NUM_LONG_TICKS;
+
for_each_possible_cpu(i) {
struct rq *rq;
struct task_grp_rq *tgrq;
@@ -7120,6 +7223,9 @@ void __init sched_init(void)
rq->nr_running = 0;
rq->active = rq->arrays;
rq->expired = rq->arrays + 1;
+ rq->greedy_active = rq->arrays + 2;
+ rq->greedy_expired = rq->arrays + 3;
+ rq->last_update = tgrq->last_update = jiffies;

#ifdef CONFIG_SMP

```

```
rq->sd = NULL;
@@ -7133,7 +7239,7 @@ void __init sched_init(void)
#endif
    atomic_set(&rq->nr_iowait, 0);

- for (j = 0; j < 2; j++) {
+ for (j = 0; j < 4; j++) {
    struct prio_array *array;
    int k;
```

--
Regards,
vatsa

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