Subject: [RFC PATCH 1/2] cgroup: add per cgroup timer_slack_ns
Posted by jacob.jun.pan on Wed, 01 Dec 2010 19:00:11 GMT
View Forum Message <> Reply to Message

From: Jacob Pan <jacob.jun.pan@linux.intel.com>

Per task timer_slack_ns was introduced a while ago to allow tuning
timer rounding behavior such that tasks can be made mode power friendly.
This patch introduces per cgroup timer slack value which will override
the default timer slack value once a task is attached to a cgroup. By
default, the root cgroup timer slack value is set to the same 50us as
in all the tasks.
At runtime, user can choose to change cgroup timer slack value by
echo xxx > cgroup.timer_slack_ns

The usage of such feature can be found in mobile devices where certain
background apps are attached to a cgroup and minimum wakeups are desired.

Signed-off-by: Jacob Pan <jacob.jun.pan@linux.intel.com>
---
 include/linux/cgroup.h | 1 +
 include/linux/init_task.h | 3 ++-
 kernel/cgroup.c | 43 +++
 3 files changed, 46 insertions(+), 1 deletions(-)

diff --git a/include/linux/cgroup.h b/include/linux/cgroup.h
index ed4ba11..afac9bb 100644
--- a/include/linux/cgroup.h
+++ b/include/linux/cgroup.h
@@ -194,6 +194,7 @@ struct cgroup_pidlist {

 struct cgroup {
  unsigned long flags; /* "unsigned long" so bitops work */
+ unsigned long timer_slack_ns;

  /*
   * count users of this cgroup. >0 means busy, but doesn't
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index 1f8c06c..5860af6 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -110,6 +110,7 @@ extern struct cred init_cred;
 # define INIT_PERF_EVENTS(tsk)
 #endif

+#define TIMER_SLACK_NS_DEFAULT (50000) /* 50 usec default slack */
 /*
  * INIT_TASK is used to set up the first task table, touch at

```
  * your own risk!. Base=0, limit=0x1fffff (=2MB)
@@ -163,7 +164,7 @@ extern struct cred init_cred;
  .cpu_timers = INIT_CPU_TIMERS(tsk.cpu_timers),  \
  .fs_excl = ATOMIC_INIT(0),    \
  .pi_lock = __RAW_SPIN_LOCK_UNLOCKED(tsk.pi_lock), \
- .timer_slack_ns = 50000, /* 50 usec default slack */  \
+ .timer_slack_ns = TIMER_SLACK_NS_DEFAULT,         \
  .pids = {       \
   [PIDTYPE_PID]  = INIT_PID_LINK(PIDTYPE_PID),  \
   [PIDTYPE_PGID] = INIT_PID_LINK(PIDTYPE_PGID),  \
diff --git a/kernel/cgroup.c b/kernel/cgroup.c
index 66a416b..0e0a254 100644
--- a/kernel/cgroup.c
+++ b/kernel/cgroup.c
@@ -57,6 +57,7 @@
 #include <linux/vmalloc.h> /* TODO: replace with more sophisticated array */
 #include <linux/eventfd.h>
 #include <linux/poll.h>
+#include <linux/init_task.h>

 #include <asm/atomic.h>

@@ -1324,6 +1325,7 @@ static void init_cgroup_root(struct cgroupfs_root *root)
  root->number_of_cgroups = 1;
  cgrp->root = root;
  cgrp->top_cgroup = cgrp;
+ cgrp->timer_slack_ns = TIMER_SLACK_NS_DEFAULT;
  init_cgroup_housekeeping(cgrp);
 }

@@ -1787,6 +1789,7 @@ int cgroup_attach_task(struct cgroup *cgrp, struct task_struct *tsk)
   goto out;
  }
  rcu_assign_pointer(tsk->cgroups, newcg);
+ tsk->timer_slack_ns = cgrp->timer_slack_ns;
  task_unlock(tsk);

  /* Update the css_set linked lists if we're using them */
@@ -3049,6 +3052,38 @@ static int cgroup_write_notify_on_release(struct cgroup *cgrp,
  return 0;
 }

+static u64 cgroup_read_timer_slack_ns(struct cgroup *cgrp,
+    struct cftype *cft)
+{
+ return cgrp->timer_slack_ns;
+}
+
```

```
+static int cgroup_write_timer_slack_ns(struct cgroup *cgrp,
+    struct cftype *cft,
+    u64 val)
+{
+ struct cgroup_iter it;
+ struct task_struct *task;
+
+ /* TODO: upper range checking for max slack */
+ if (val)
+  cgrp->timer_slack_ns = val;
+ else {
+  printk(KERN_ERR "cgroup %s: invalid timer slack value %llu\n",
+   cgrp->dentry->d_name.name, val);
+  return -EINVAL;
+ }
+
+ /* change timer slack value for all tasks in the cgroup */
+ cgroup_iter_start(cgrp, &it);
+ while ((task = cgroup_iter_next(cgrp, &it)))
+  task->timer_slack_ns = val;
+
+ cgroup_iter_end(cgrp, &it);
+
+ return 0;
+}
+
 /*
  * Unregister event and free resources.
  *
@@ -3268,6 +3303,13 @@ static struct cftype files[] = {
   .read_u64 = cgroup_clone_children_read,
   .write_u64 = cgroup_clone_children_write,
  },
+ {
+  .name = CGROUP_FILE_GENERIC_PREFIX "timer_slack_ns",
+  .read_u64 = cgroup_read_timer_slack_ns,
+  .write_u64 = cgroup_write_timer_slack_ns,
+  .mode = S_IRUGO | S_IWUSR,
+ },
+
 };

 static struct cftype cft_release_agent = {
@@ -3393,6 +3435,7 @@ static long cgroup_create(struct cgroup *parent, struct dentry *dentry,
  cgrp->parent = parent;
  cgrp->root = parent->root;
  cgrp->top_cgroup = parent->top_cgroup;
+ cgrp->timer_slack_ns = TIMER_SLACK_NS_DEFAULT;
```

```
  if (notify_on_release(parent))
   set_bit(CGRP_NOTIFY_ON_RELEASE, &cgrp->flags);
--
1.7.0.4
```