Subject: [PATCH 9/9] Interface with process container patchset
Posted by Srivatsa Vaddagiri on Thu, 12 Apr 2007 18:01:27 GMT
View Forum Message <> Reply to Message

This patch registers cpu controller as a subsystem in process container
patches.

How to use the controller:

1. Apply these patches first on top of 2.6.20 kernel
   http://lkml.org/lkml/2007/4/6/301
   http://lkml.org/lkml/2007/4/6/300
   http://marc.info/?l=ckrm-tech&m=117590464104300&w=2

2. Apply all patches in this thread (from 1 - 9)

3. Select CONFIG_CPUMETER and build kernel

After bootup:

```
# mkdir /dev/cpuctl
# mount -t container -o cpuctl none /dev/cpuctl
# cd /dev/cpuctl
# mkdir a       # Create group A
# mkdir b       # Create group B

# echo 80 > a/quota     # Give 80% quota to A
# echo 20 > b/quota     # Give 20% quota to B

# echo some_pid > a/tasks       # Move some_pid to Group A

# echo another_pid > b/tasks    # move another task to Group B
```

some_pid and another_pid should share CPU in the ratio of 80:20 now

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

---

linux-2.6.20-vatsa/include/linux/container_subsys.h | 6

```
diff -puN include/linux/container_subsys.h~container_if include/linux/container_subsys.h
--- linux-2.6.20/include/linux/container_subsys.h~container_if 2007-04-12 09:10:40.000000000
+0530
+++ linux-2.6.20-vatsa/include/linux/container_subsys.h 2007-04-12 09:10:40.000000000 +0530
@@ -17,4 +17,10 @@ SUBSYS(cpuacct)

 /* */

+#ifdef CONFIG_CPUMETER
+SUBSYS(cpuctlr)
+#endif
+
+/* */
+
 /* */
diff -puN kernel/sched.c~container_if kernel/sched.c
--- linux-2.6.20/kernel/sched.c~container_if 2007-04-12 09:10:40.000000000 +0530
+++ linux-2.6.20-vatsa/kernel/sched.c 2007-04-12 11:07:10.000000000 +0530
@@ -53,6 +53,8 @@
 #include <linux/kprobes.h>
 #include <linux/delayacct.h>
 #include <linux/cpu_acct.h>
+#include <linux/container.h>
+#include <linux/fs.h>
 #include <asm/tlb.h>

 #include <asm/unistd.h>
@@ -226,6 +228,7 @@ static DEFINE_PER_CPU(struct task_grp_rq

 /* task-group object - maintains information about each task-group */
 struct task_grp {
+ struct container_subsys_state css;
  unsigned short ticks, long_ticks; /* bandwidth given to task-group */
   int left_over_pct;
   int total_dont_care_grps;
@@ -430,10 +433,8 @@ static inline void finish_lock_switch(st
 /* return the task-group to which a task belongs */
 static inline struct task_grp *task_grp(struct task_struct *p)
 {
- /* Simply return the default group for now. A later patch modifies
-  * this function.
-  */
- return &init_task_grp;
+ return container_of(task_subsys_state(p, cpuctlr_subsys_id),
+  struct task_grp, css);
 }

 /*
```

```
@@ -7487,6 +7488,12 @@ void set_curr_task(int cpu, struct task_

 #ifdef CONFIG_CPUMETER

+static struct task_grp *container_tg(struct container *cont)
+{
+ return container_of(container_subsys_state(cont, cpuctlr_subsys_id),
+     struct task_grp, css);
+}
+
 /* Distribute left over bandwidth equally to all "dont care" task groups */
 static void recalc_dontcare(struct task_grp *tg_root)
 {
@@ -7511,16 +7518,26 @@ static void recalc_dontcare(struct task_
 }

 /* Allocate runqueue structures for the new task-group */
-static int sched_create_group(struct task_grp *tg_parent)
+static int sched_create_group(struct container_subsys *ss,
+     struct container *cont)
 {
- struct task_grp *tg;
+ struct task_grp *tg, *tg_parent;
  struct task_grp_rq *tgrq;
  int i;

- if (tg_parent->parent)
+ if (!cont->parent) {
+  /* This is early initialization for the top container */
+  cont->subsys[cpuctlr_subsys_id] = &init_task_grp.css;
+  init_task_grp.css.container = cont;
+  return 0;
+ }
+
+ if (cont->parent->parent)
   /* We don't support hierarchical CPU res mgmt (yet) */
   return -EINVAL;

+ tg_parent = container_tg(cont->parent);
+
  tg = kzalloc(sizeof(*tg), GFP_KERNEL);
  if (!tg)
   return -ENOMEM;
@@ -7549,7 +7566,9 @@ static int sched_create_group(struct tas
   list_add_tail(&tg->list, &tg->dont_care_list);
  }

- /* A later patch will make 'tg' accessible beyond this function */
```

```
+ cont->subsys[cpuctlr_subsys_id] = &tg->css;
+ tg->css.container = cont;
+
  return 0;
 oom:
  while (i--)
@@ -7560,9 +7579,11 @@ oom:
 }

 /* Deallocate runqueue structures */
-static void sched_destroy_group(struct task_grp *tg)
+static void sched_destroy_group(struct container_subsys *ss,
+      struct container *cont)
 {
  int i;
+ struct task_grp *tg = container_tg(cont);
  struct task_grp *tg_root = tg->parent;

  if (!tg_root)
@@ -7581,10 +7602,22 @@ static void sched_destroy_group(struct t
 }

 /* Assign quota to this group */
-static int sched_set_quota(struct task_grp *tg, int quota)
+static ssize_t sched_set_quota(struct container *cont, struct cftype *cft,
+    struct file *file, const char __user *userbuf,
+    size_t nbytes, loff_t *ppos)
 {
+ struct task_grp *tg = container_tg(cont);
  int i, old_quota = 0;
  struct task_grp *tg_root = tg->parent;
+ int quota;
+ char buffer[64];
+
+ if (copy_from_user(buffer, userbuf, sizeof(quota)))
+                return -EFAULT;
+
+ buffer[sizeof(quota)] = 0;
+
+ quota = simple_strtoul(buffer, NULL, 10);

  if (!tg_root)
   tg_root = tg;
@@ -7611,16 +7644,29 @@ static int sched_set_quota(struct task_g
  tg_root->left_over_pct -= (quota - old_quota);
  recalc_dontcare(tg_root);

- return 0;
```

```
+ return nbytes;
 }

 /* Return assigned quota for this group */
-static int sched_get_quota(struct task_grp *tg)
-{
+static ssize_t sched_get_quota(struct container *cont, struct cftype *cft,
+   struct file *file, char __user *buf, size_t nbytes,
+   loff_t *ppos)
+{
+ struct task_grp *tg = container_tg(cont);
+ char quotabuf[64];
+ char *s = quotabuf;
+ int quota;
+
  if (tg->dont_care)
-  return 0;
+  quota = 0;
  else
-  return cpu_quota(tg);
+  quota = cpu_quota(tg);
+
+ s += sprintf(s, "%d\n", quota);
+
+ return simple_read_from_buffer(buf, nbytes, ppos, quotabuf,
+       s - quotabuf);
+
 }

 /*
@@ -7628,23 +7674,35 @@ static int sched_get_quota(struct task_g
  * runqueue, this involves removing the task from its old group's runqueue
  * and adding to its new group's runqueue.
  */
-static void sched_move_task(struct task_grp *tg_new, struct task_grp *tg_old,
-     struct task_struct *tsk)
+static void sched_move_task(struct container_subsys *ss, struct container *cont,
+   struct container *old_cont, struct task_struct *tsk)
 {
  struct rq *rq;
  unsigned long flags;

- if (tg_new == tg_old)
-  return;
-
  rq = task_rq_lock(tsk, &flags);

  if (tsk->array) {
```

```
-  /* Set tsk->group to tg_old here */
+  struct task_grp *tg_old, *tg_new;
+
+  task_lock(tsk);
+
+  tg_old = container_tg(old_cont);
+  tg_new = container_tg(cont);
+
+  /* deactivate_task and activate_task rely on
+   * tsk->containers->subsys[cpuctlr_subsys_id] to know the
+   * appropritate group from which the task has to be dequeued
+   * and queued. Modify that appropriately before invoking them
+   */
+  tsk->containers->subsys[cpuctlr_subsys_id] = &tg_old->css;
   deactivate_task(tsk, rq);
-  /* Set tsk->group to tg_new here */
+
+  tsk->containers->subsys[cpuctlr_subsys_id] = &tg_new->css;
   set_load_weight(tsk);
   __activate_task(tsk, rq);
+
+  task_unlock(tsk);
  }

  task_rq_unlock(rq, &flags);
@@ -7652,4 +7710,49 @@ static void sched_move_task(struct task_
  return;
 }

+static struct cftype cft_quota = {
+ .name = "quota",
+ .read = sched_get_quota,
+ .write = sched_set_quota,
+};
+
+static int sched_populate(struct container_subsys *ss, struct container *cont)
+{
+ int err;
+
+ if ((err = container_add_file(cont, &cft_quota)))
+  return err;
+
+ return err;
+}
+
+static void sched_exit_task(struct container_subsys *ss, struct task_struct *p)
+{
+ struct rq *rq;
```

```
+ unsigned long flags;
+
+ rq = task_rq_lock(p, &flags);
+
+ if (p->array) {
+  struct task_grp_rq *init_tgrq = init_task_grp.rq[task_cpu(p)];
+
+  dequeue_task(p, p->array);
+  enqueue_task(p, init_tgrq->active);
+ }
+
+ task_rq_unlock(rq, &flags);
+}
+
+
+struct container_subsys cpuctlr_subsys = {
+ .name = "cpuctl",
+ .create = sched_create_group,
+ .destroy  = sched_destroy_group,
+ .attach = sched_move_task,
+ .populate = sched_populate,
+ .subsys_id = cpuctlr_subsys_id,
+ .exit = sched_exit_task,
+ .early_init = 1,
+};
+
 #endif /* CONFIG_CPUMETER */
_

--
Regards,
vatsa
```

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