Subject: [PATCH 4/6] containers: Simple CPU accounting container subsystem
Posted by Paul Menage on Fri, 22 Dec 2006 14:14:46 GMT
View Forum Message <> Reply to Message

This demonstrates how to use the generic container subsystem for a
simple resource tracker that counts the total CPU time used by all
processes in a container, during the time that they're members of the
container.

Signed-off-by: Paul Menage <menage@google.com>

---
 include/linux/cpu_acct.h | 14 +++++
 init/Kconfig | 7 ++
 kernel/Makefile | 1
 kernel/cpu_acct.c | 117 +++
 kernel/sched.c | 6 ++
 5 files changed, 145 insertions(+)

Index: container-2.6.20-rc1/include/linux/cpu_acct.h
 === =======
--- /dev/null
+++ container-2.6.20-rc1/include/linux/cpu_acct.h
@@ -0,0 +1,14 @@
+
+#ifndef _LINUX_CPU_ACCT_H
+#define _LINUX_CPU_ACCT_H
+
+#include <linux/container.h>
+#include <asm/cputime.h>
+
+#ifdef CONFIG_CONTAINER_CPUACCT
+extern void cpuacct_charge(struct task_struct *, cputime_t cputime);
+#else
+static void inline cpuacct_charge(struct task_struct *p, cputime_t cputime) {}
+#endif
+
+#endif
Index: container-2.6.20-rc1/init/Kconfig
 === =======
--- container-2.6.20-rc1.orig/init/Kconfig
+++ container-2.6.20-rc1/init/Kconfig
@@ -290,6 +290,13 @@ config PROC_PID_CPUSET
  depends on CPUSETS
  default y

+config CONTAINER_CPUACCT
+ bool "Simple CPU accounting container subsystem"

```
+ select CONTAINERS
+ help
+   Provides a simple Resource Controller for monitoring the
+   total CPU consumed by the tasks in a container
+
 config RELAY
  bool "Kernel->user space relay support (formerly relayfs)"
  help
Index: container-2.6.20-rc1/kernel/cpu_acct.c
 ===================================================================
--- /dev/null
+++ container-2.6.20-rc1/kernel/cpu_acct.c
@@ -0,0 +1,117 @@
+/*
+ * kernel/cpu_acct.c - CPU accounting container subsystem
+ *
+ * Copyright (C) Google Inc, 2006
+ *
+ */
+
+/*
+ * Container subsystem for reporting total CPU usage of tasks in a
+ * container.
+ */
+
+#include <linux/module.h>
+#include <linux/container.h>
+#include <linux/fs.h>
+#include <asm/div64.h>
+
+struct cpuacct {
+ struct container_subsys_state css;
+ spinlock_t lock;
+ cputime64_t time; // total time used by this class
+};
+
+static struct container_subsys cpuacct_subsys;
+
+static inline struct cpuacct *container_ca(struct container *cont)
+{
+ return container_of(container_subsys_state(cont, &cpuacct_subsys),
+       struct cpuacct, css);
+}
+
+static inline struct cpuacct *task_ca(struct task_struct *task)
+{
+ return container_ca(task_container(task, &cpuacct_subsys));
+}
```

```
+
+static int cpuacct_create(struct container_subsys *ss, struct container *cont)
+{
+ struct cpuacct *ca = kzalloc(sizeof(*ca), GFP_KERNEL);
+ if (!ca) return -ENOMEM;
+ spin_lock_init(&ca->lock);
+ cont->subsys[cpuacct_subsys.subsys_id] = &ca->css;
+ return 0;
+}
+
+static void cpuacct_destroy(struct container_subsys *ss,
+       struct container *cont)
+{
+ kfree(container_ca(cont));
+}
+
+static ssize_t cpuusage_read(struct container *cont,
+        struct cftype *cft,
+        struct file *file,
+        char __user *buf,
+        size_t nbytes, loff_t *ppos)
+{
+ struct cpuacct *ca = container_ca(cont);
+ cputime64_t time;
+ char usagebuf[64];
+ char *s = usagebuf;
+
+ spin_lock_irq(&ca->lock);
+ time = ca->time;
+ spin_unlock_irq(&ca->lock);
+
+ time *= 1000;
+ do_div(time, HZ);
+ s += sprintf(s, "%llu", (unsigned long long) time);
+
+ return simple_read_from_buffer(buf, nbytes, ppos, usagebuf, s - usagebuf);
+}
+
+static struct cftype cft_usage = {
+ .name = "cpuacct.usage",
+ .read = cpuusage_read,
+};
+
+static int cpuacct_populate(struct container_subsys *ss,
+       struct container *cont)
+{
+ return container_add_file(cont, &cft_usage);
+}
```

```
+
+
+void cpuacct_charge(struct task_struct *task, cputime_t cputime) {
+
+ struct cpuacct *ca;
+ unsigned long flags;
+
+ if (cpuacct_subsys.subsys_id < 0) return;
+ rcu_read_lock();
+ ca = task_ca(task);
+ if (ca) {
+  spin_lock_irqsave(&ca->lock, flags);
+  ca->time = cputime64_add(ca->time, cputime);
+  spin_unlock_irqrestore(&ca->lock, flags);
+ }
+ rcu_read_unlock();
+}
+
+static struct container_subsys cpuacct_subsys = {
+ .name = "cpuacct",
+ .create = cpuacct_create,
+ .destroy = cpuacct_destroy,
+ .populate = cpuacct_populate,
+ .subsys_id = -1,
+};
+
+
+int __init init_cpuacct(void)
+{
+ int id = container_register_subsys(&cpuacct_subsys);
+ return id < 0 ? id : 0;
+}
+
+module_init(init_cpuacct)
Index: container-2.6.20-rc1/kernel/Makefile
 ===================================================================
--- container-2.6.20-rc1.orig/kernel/Makefile
+++ container-2.6.20-rc1/kernel/Makefile
@@ -38,6 +38,7 @@ obj-$(CONFIG_KEXEC) += kexec.o
 obj-$(CONFIG_COMPAT) += compat.o
 obj-$(CONFIG_CONTAINERS) += container.o
 obj-$(CONFIG_CPUSETS) += cpuset.o
+obj-$(CONFIG_CONTAINER_CPUACCT) += cpu_acct.o
 obj-$(CONFIG_IKCONFIG) += configs.o
 obj-$(CONFIG_STOP_MACHINE) += stop_machine.o
 obj-$(CONFIG_AUDIT) += audit.o auditfilter.o
Index: container-2.6.20-rc1/kernel/sched.c
 ===================================================================
```

```
--- container-2.6.20-rc1.orig/kernel/sched.c
+++ container-2.6.20-rc1/kernel/sched.c
@@ -52,6 +52,7 @@
 #include <linux/tsacct_kern.h>
 #include <linux/kprobes.h>
 #include <linux/delayacct.h>
+#include <linux/cpu_acct.h>
 #include <asm/tlb.h>

 #include <asm/unistd.h>
@@ -3068,6 +3069,8 @@ void account_user_time(struct task_struc

  p->utime = cputime_add(p->utime, cputime);

+ cpuacct_charge(p, cputime);
+
  /* Add user time to cpustat. */
  tmp = cputime_to_cputime64(cputime);
  if (TASK_NICE(p) > 0)
@@ -3091,6 +3094,9 @@ void account_system_time(struct task_str

  p->stime = cputime_add(p->stime, cputime);

+ if (p != rq->idle)
+  cpuacct_charge(p, cputime);
+
  /* Add system time to cpustat. */
  tmp = cputime_to_cputime64(cputime);
  if (hardirq_count() - hardirq_offset)

--
```