
Subject: [PATCH 12/29] task containersv11 example cpu accounting subsystem
Posted by [Paul Menage](#) on Tue, 11 Sep 2007 19:52:51 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Paul Menage <menage@google.com>

This example demonstrates how to use the generic cgroup subsystem for a simple resource tracker that counts, for the processes in a cgroup, the total CPU time used and the %CPU used in the last complete 10 second interval.

Portions contributed by Balbir Singh <balbir@in.ibm.com>

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

Cc: Serge E. Hallyn <serue@us.ibm.com>

Cc: "Eric W. Biederman" <ebiederm@xmission.com>

Cc: Dave Hansen <haveblue@us.ibm.com>

Cc: Balbir Singh <balbir@in.ibm.com>

Cc: Paul Jackson <pj@sgi.com>

Cc: Kirill Korotaev <dev@openvz.org>

Cc: Herbert Poetzl <herbert@13thfloor.at>

Cc: Srivatsa Vaddagiri <vatsa@in.ibm.com>

Cc: Cedric Le Goater <cld@fr.ibm.com>

Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

```
include/linux/cgroup_subsys.h |  6
include/linux/cpu_acct.h     | 14 ++
init/Kconfig                 |  7 +
kernel/Makefile               |   1
kernel/cpu_acct.c            | 186 ++++++++++++++++++++++++++++++
kernel/sched.c                | 14 ++
6 files changed, 225 insertions(+), 3 deletions(-)
```

```
diff -puN include/linux/cgroup_subsys.h~task-cgroups v11-example-cpu-accounting-subsystem
include/linux/cgroup_subsys.h
--- a/include/linux/cgroup_subsys.h~task-cgroups v11-example-cpu-accounting-subsystem
+++ a/include/linux/cgroup_subsys.h
@@ -13,4 +13,10 @@ SUBSYS(cpuset)
```

/* */

```
+#ifdef CONFIG_CGROUP_CPUACCT
+SUBSYS(cpuacct)
+#endif
+
+/* */
+
/* */
```

```

diff -puN /dev/null include/linux/cpu_acct.h
--- /dev/null
+++ a/include/linux/cpu_acct.h
@@ @ -0,0 +1,14 @@
+
+ifndef _LINUX_CPU_ACCT_H
#define _LINUX_CPU_ACCT_H
+
+#include <linux/cgroup.h>
+#include <asm/cputime.h>
+
+ifdef CONFIG_CGROUP_CPUACCT
+extern void cpacct_charge(struct task_struct *, cputime_t cputime);
+else
+static void inline cpacct_charge(struct task_struct *p, cputime_t cputime) {}
+endif
+
+endif
diff -puN init/Kconfig~task-cgroups11-example-cpu-accounting-subsystem init/Kconfig
--- a/init/Kconfig~task-cgroups11-example-cpu-accounting-subsystem
+++ a/init/Kconfig
@@ @ -318,6 +318,13 @@ config PROC_PID_CPUSET
depends on CPUSETS
default y

+config CGROUP_CPUACCT
+ bool "Simple CPU accounting cgroup subsystem"
+ depends on CGROUPS
+ help
+ Provides a simple Resource Controller for monitoring the
+ total CPU consumed by the tasks in a cgroup
+
config RELAY
bool "Kernel->user space relay support (formerly relayfs)"
help
diff -puN kernel/Makefile~task-cgroups11-example-cpu-accounting-subsystem kernel/Makefile
--- a/kernel/Makefile~task-cgroups11-example-cpu-accounting-subsystem
+++ a/kernel/Makefile
@@ @ -40,6 +40,7 @@ obj-$(CONFIG_KEXEC) += kexec.o
obj-$(CONFIG_COMPAT) += compat.o
obj-$(CONFIG_CGROUPS) += cgroup.o
obj-$(CONFIG_CPUSETS) += cpuset.o
+obj-$(CONFIG_CGROUP_CPUACCT) += cpu_acct.o
obj-$(CONFIG_IKCONFIG) += configs.o
obj-$(CONFIG_STOP_MACHINE) += stop_machine.o
obj-$(CONFIG_AUDIT) += audit.o auditfilter.o
diff -puN /dev/null kernel/cpu_acct.c
--- /dev/null

```

```

+++ a/kernel/cpu_acct.c
@@ -0,0 +1,186 @@
+/*
+ * kernel/cpu_acct.c - CPU accounting cgroup subsystem
+ *
+ * Copyright (C) Google Inc, 2006
+ *
+ * Developed by Paul Menage (menage@google.com) and Balbir Singh
+ * (balbir@in.ibm.com)
+ *
+ */
+
+/*
+ * Example cgroup subsystem for reporting total CPU usage of tasks in a
+ * cgroup, along with percentage load over a time interval
+ */
+
+#
+#include <linux/module.h>
+#include <linux/cgroup.h>
+#include <linux/fs.h>
+#include <linux/rcupdate.h>
+
+#
+#include <asm/div64.h>
+
+struct cpuacct {
+    struct cgroup_subsys_state css;
+    spinlock_t lock;
+    /* total time used by this class */
+    cputime64_t time;
+
+    /* time when next load calculation occurs */
+    u64 next_interval_check;
+
+    /* time used in current period */
+    cputime64_t current_interval_time;
+
+    /* time used in last period */
+    cputime64_t last_interval_time;
+};
+
+struct cgroup_subsys cpuacct_subsys;
+
+static inline struct cpuacct *cgroup_ca(struct cgroup *cont)
+{
+    return container_of(cgroup_subsys_state(cont, cpuacct_subsys_id),
+        struct cpuacct, css);
+}
+

```

```

+static inline struct cpacct *task_ca(struct task_struct *task)
+{
+    return container_of(task_subsys_state(task, cpacct_subsys_id),
+        struct cpacct, css);
+}
+
+">#define INTERVAL (HZ * 10)
+
+static inline u64 next_interval_boundary(u64 now)
+{
+ /* calculate the next interval boundary beyond the
+ * current time */
+ do_div(now, INTERVAL);
+ return (now + 1) * INTERVAL;
+}
+
+static struct cgroup_subsys_state *cpacct_create(
+    struct cgroup_subsys *ss, struct cgroup *cont)
+{
+    struct cpacct *ca = kzalloc(sizeof(*ca), GFP_KERNEL);
+
+    if (!ca)
+        return ERR_PTR(-ENOMEM);
+    spin_lock_init(&ca->lock);
+    ca->next_interval_check = next_interval_boundary(get_jiffies_64());
+    return &ca->css;
+}
+
+static void cpacct_destroy(struct cgroup_subsys *ss,
+    struct cgroup *cont)
+{
+    kfree(cgroup_ca(cont));
+}
+
+/* Lazily update the load calculation if necessary. Called with ca locked */
+static void cpusage_update(struct cpacct *ca)
+{
+    u64 now = get_jiffies_64();
+
+    /* If we're not due for an update, return */
+    if (ca->next_interval_check > now)
+        return;
+
+    if (ca->next_interval_check <= (now - INTERVAL)) {
+        /* If it's been more than an interval since the last
+         * check, then catch up - the last interval must have
+         * been zero load */
+        ca->last_interval_time = 0;

```

```

+ ca->next_interval_check = next_interval_boundary(now);
+ } else {
+ /* If a steal takes the last interval time negative,
+ * then we just ignore it */
+ if ((s64)ca->current_interval_time > 0)
+ ca->last_interval_time = ca->current_interval_time;
+ else
+ ca->last_interval_time = 0;
+ ca->next_interval_check += INTERVAL;
+ }
+ ca->current_interval_time = 0;
+}
+
+static u64 cpuusage_read(struct cgroup *cont, struct cftype *cft)
+{
+ struct cpuacct *ca = cgroup_ca(cont);
+ u64 time;
+
+ spin_lock_irq(&ca->lock);
+ cpuusage_update(ca);
+ time = cputime64_to_jiffies64(ca->time);
+ spin_unlock_irq(&ca->lock);
+
+ /* Convert 64-bit jiffies to seconds */
+ time *= 1000;
+ do_div(time, HZ);
+ return time;
+}
+
+static u64 load_read(struct cgroup *cont, struct cftype *cft)
+{
+ struct cpuacct *ca = cgroup_ca(cont);
+ u64 time;
+
+ /* Find the time used in the previous interval */
+ spin_lock_irq(&ca->lock);
+ cpuusage_update(ca);
+ time = cputime64_to_jiffies64(ca->last_interval_time);
+ spin_unlock_irq(&ca->lock);
+
+ /* Convert time to a percentage, to give the load in the
+ * previous period */
+ time *= 100;
+ do_div(time, INTERVAL);
+
+ return time;
+}
+

```

```

+static struct cftype files[] = {
+{
+ .name = "usage",
+ .read_uint = cpususage_read,
+},
+{
+ .name = "load",
+ .read_uint = load_read,
+}
+};
+
+static int cpacct_populate(struct cgroup_subsys *ss, struct cgroup *cont)
+{
+ return cgroup_add_files(cont, ss, files, ARRAY_SIZE(files));
+}
+
+void cpacct_charge(struct task_struct *task, cputime_t cputime)
+{
+
+ struct cpacct *ca;
+ unsigned long flags;
+
+ if (!cpacct_subsys.active)
+ return;
+ rCU_read_lock();
+ ca = task_ca(task);
+ if (ca) {
+ spin_lock_irqsave(&ca->lock, flags);
+ cpususage_update(ca);
+ ca->time = cputime64_add(ca->time, cputime);
+ ca->current_interval_time =
+ cputime64_add(ca->current_interval_time, cputime);
+ spin_unlock_irqrestore(&ca->lock, flags);
+ }
+ rCU_read_unlock();
+}
+
+struct cgroup_subsys cpacct_subsys = {
+ .name = "cpacct",
+ .create = cpacct_create,
+ .destroy = cpacct_destroy,
+ .populate = cpacct_populate,
+ .subsys_id = cpacct_subsys_id,
+};
diff -puN kernel/sched.c~task-cgroups v11-example-cpu-accounting-subsystem kernel/sched.c
--- a/kernel/sched.c~task-cgroups v11-example-cpu-accounting-subsystem
+++ a/kernel/sched.c
@@ -51,6 +51,7 @@
```

```

#include <linux/cpu.h>
#include <linux/cpuset.h>
#include <linux/percpu.h>
+ #include <linux/cpu_acct.h>
#include <linux/kthread.h>
#include <linux/seq_file.h>
#include <linux/sysctl.h>
@@ -3268,9 +3269,13 @@ void account_user_time(struct task_struct
{
    struct cpu_usage_stat *cpustat = &kstat_this_cpu.cpustat;
    cputime64_t tmp;
+   struct rq *rq = this_rq();
    p->utime = cputime_add(p->utime, cputime);

+   if (p != rq->idle)
+       cpuacct_charge(p, cputime);
+
/* Add user time to cpustat. */
tmp = cputime_to_cputime64(cputime);
if (TASK_NICE(p) > 0)
@@ -3300,9 +3305,10 @@ void account_system_time(struct task_struct
    cpustat->irq = cputime64_add(cpustat->irq, tmp);
else if (softirq_count())
    cpustat->softirq = cputime64_add(cpustat->softirq, tmp);
- else if (p != rq->idle)
+ else if (p != rq->idle) {
    cpustat->system = cputime64_add(cpustat->system, tmp);
- else if (atomic_read(&rq->nr_iowait) > 0)
+   cpuacct_charge(p, cputime);
+ } else if (atomic_read(&rq->nr_iowait) > 0)
    cpustat->iowait = cputime64_add(cpustat->iowait, tmp);
else
    cpustat->idle = cputime64_add(cpustat->idle, tmp);
@@ -3327,8 +3333,10 @@ void account_stal_time(struct task_struct
    cpustat->iowait = cputime64_add(cpustat->iowait, tmp);
else
    cpustat->idle = cputime64_add(cpustat->idle, tmp);
- } else
+ } else {
    cpustat->steal = cputime64_add(cpustat->steal, tmp);
+   cpuacct_charge(p, -tmp);
+ }
}

/*
-

```

--

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
