Subject: [PATCH 0/2] Show per-cpu data in cpuacct stats
Posted by Glauber Costa on Wed, 20 Jun 2012 11:38:14 GMT
View Forum Message <> Reply to Message

Hi,

Peter, Paul, and others:

Since Peter is currently reworking put_prev_task/pick_next_task, I split
my cpu stats series in two: the part that touches cpuacct is not dependent
on that at all. So please let me know if you are comfortable with merging this.

Is basically the same code as my last submission, which gets rid of the
key:value mapping to expose the information, resorting to the seq_string
interface.

Glauber Costa (2):
  account guest time per-cgroup as well.
  expose fine-grained per-cpu data for cpuacct stats

 kernel/sched/core.c | 47 ++------
 1 file changed, 41 insertions(+), 6 deletions(-)

--
1.7.10.2

Subject: [PATCH 1/2] account guest time per-cgroup as well.
Posted by Glauber Costa on Wed, 20 Jun 2012 11:38:15 GMT
View Forum Message <> Reply to Message

We already track multiple tick statistics per-cgroup, using
the task_group_account_field facility. This patch accounts
guest_time in that manner as well.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Peter Zijlstra <a.p.zijlstra@chello.nl>
CC: Paul Turner <pjt@google.com>
---
 kernel/sched/core.c | 10 ++++++------
 1 file changed, 4 insertions(+), 6 deletions(-)

diff --git a/kernel/sched/core.c b/kernel/sched/core.c
index f442060..4ce68f6 100644
--- a/kernel/sched/core.c
+++ b/kernel/sched/core.c
@@ -2689,8 +2689,6 @@ void account_user_time(struct task_struct *p, cputime_t cputime,
 static void account_guest_time(struct task_struct *p, cputime_t cputime,

```
          cputime_t cputime_scaled)
 {
- u64 *cpustat = kcpustat_this_cpu->cpustat;
-
   /* Add guest time to process. */
   p->utime += cputime;
   p->utimescaled += cputime_scaled;
@@ -2699,11 +2697,11 @@ static void account_guest_time(struct task_struct *p, cputime_t
cputime,

   /* Add guest time to cpustat. */
   if (TASK_NICE(p) > 0) {
-  cpustat[CPUTIME_NICE] += (__force u64) cputime;
-  cpustat[CPUTIME_GUEST_NICE] += (__force u64) cputime;
+  task_group_account_field(p, CPUTIME_NICE, (__force u64) cputime);
+  task_group_account_field(p, CPUTIME_GUEST, (__force u64) cputime);
   } else {
-  cpustat[CPUTIME_USER] += (__force u64) cputime;
-  cpustat[CPUTIME_GUEST] += (__force u64) cputime;
+  task_group_account_field(p, CPUTIME_USER, (__force u64) cputime);
+  task_group_account_field(p, CPUTIME_GUEST, (__force u64) cputime);
   }
 }

--
1.7.10.2
```

Subject: [PATCH 2/2] expose fine-grained per-cpu data for cpuacct stats
Posted by Glauber Costa on Wed, 20 Jun 2012 11:38:16 GMT
View Forum Message <> Reply to Message

The cpuacct cgroup already exposes user and system numbers in a per-cgroup fashion. But they are a summation along the whole group, not a per-cpu figure. Also, they are coarse-grained version of the stats usually shown at places like /proc/stat.

I want to have enough cgroup data to emulate the /proc/stat interface. To achieve that, I am creating a new file "stat_percpu" that displays the fine-grained per-cpu data. The original data is left alone.

The format of this file is as follows:

1line header
cpux val1 val2 ... valn.

It is the same format used for the cpu part /proc/stat, except for the header, that may allow us to add fields in the future if they prove themselves needed.

```
Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Peter Zijlstra <a.p.zijlstra@chello.nl>
CC: Paul Turner <pjt@google.com>
---
 kernel/sched/core.c |   37 ++++++++++++++++++++++++++++++++++++
 1 file changed, 37 insertions(+)

diff --git a/kernel/sched/core.c b/kernel/sched/core.c
index 4ce68f6..5c19754 100644
--- a/kernel/sched/core.c
+++ b/kernel/sched/core.c
@@ -8155,6 +8155,39 @@ static int cpuacct_stats_show(struct cgroup *cgrp, struct cftype *cft,
  return 0;
 }

+static inline void do_fill_seq(struct seq_file *m, struct cpuacct *ca,
+          int cpu, int index)
+{
+ struct kernel_cpustat *kcpustat = per_cpu_ptr(ca->cpustat, cpu);
+ u64 val;
+
+ val = cputime64_to_clock_t(kcpustat->cpustat[index]);
+ seq_put_decimal_ull(m, ' ', val);
+}
+
+static int cpuacct_stats_percpu_show(struct cgroup *cgrp, struct cftype *cft,
+         struct seq_file *m)
+{
+ struct cpuacct *ca = cgroup_ca(cgrp);
+ int cpu;
+
+ seq_printf(m, "user nice system irq softirq guest guest_nice\n");
+
+ for_each_online_cpu(cpu) {
+  seq_printf(m, "cpu%d", cpu);
+  do_fill_seq(m, ca, cpu, CPUTIME_USER);
+  do_fill_seq(m, ca, cpu, CPUTIME_NICE);
+  do_fill_seq(m, ca, cpu, CPUTIME_SYSTEM);
+  do_fill_seq(m, ca, cpu, CPUTIME_IRQ);
+  do_fill_seq(m, ca, cpu, CPUTIME_SOFTIRQ);
+  do_fill_seq(m, ca, cpu, CPUTIME_GUEST);
+  do_fill_seq(m, ca, cpu, CPUTIME_GUEST_NICE);
+  seq_putc(m, '\n');
+ }
+
+ return 0;
+}
```

```
+
 static struct cftype files[] = {
  {
   .name = "usage",
@@ -8169,6 +8202,10 @@ static struct cftype files[] = {
   .name = "stat",
   .read_map = cpuacct_stats_show,
  },
+ {
+  .name = "stat_percpu",
+  .read_seq_string = cpuacct_stats_percpu_show,
+ },
  { } /* terminate */
 };

--
1.7.10.2
```