Subject: [PATCH] sched: cpu accounting controller
Posted by Srivatsa Vaddagiri on Thu, 29 Nov 2007 19:11:48 GMT
View Forum Message <> Reply to Message

On Mon, Nov 12, 2007 at 11:57:03PM -0800, Paul Menage wrote:
> > Regarding your concern about tracking cpu usage in different ways, it
> > could be mitigated if we have cpuacct controller track usage as per
> > information present in a task's sched entity structure
> > (tsk->se.sum_exec_runtime) i.e call cpuacct_charge() from
> > __update_curr() which would accumulate the execution time of the
> > group in a SMP friendly manner (i.e dump it in a per-cpu per-group counter
> > first and then aggregate to a global per-group counter).
>
> That seems more reasonable than the current approach in cpu_acct.c

Paul,
Sorry about the delay in getting back to this thread. I realized
very recently that cpuacct controller has been removed from Linus's tree
and have attempted to rework it as per our discussions.

Linus/Ingo,
Commit cfb5285660aad4931b2ebbfa902ea48a37dfffa1 removed a usefull
feature for us, which provided a cpu accounting resource controller. This
feature would be usefull if someone wants to group tasks only for accounting
purpose and doesnt really want to exercise any control over their cpu
consumption.

The patch below reintroduces the feature. It is based on Paul Menage's
original patch (Commit 62d0df64065e7c135d0002f069444fbdfc64768f), with
these differences:

- Removed load average information. I felt it needs
  more thought (esp to deal with SMP and virtualized platforms)
  and can be added for 2.6.25 after more discussions.
- Convert group cpu usage to be nanosecond accurate (as rest
  of the cfs stats are) and invoke cpuacct_charge() from
  the respective scheduler classes

The patch also modifies the cpu controller not to provide the same
accounting information.

Todo:
- Make the accounting scalable on SMP systems (perhaps
  for 2.6.25)

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

```
---
 include/linux/cgroup_subsys.h |    6 ++
 include/linux/cpu_acct.h      |   14 +++++
 init/Kconfig                  |    7 ++
 kernel/Makefile               |    1
 kernel/cpu_acct.c             |  103 ++++++++++++++++++++++++++++++++++++++++++
 kernel/sched.c                |   27 ----------
 kernel/sched_fair.c           |    5 ++
 kernel/sched_rt.c             |    1
 8 files changed, 138 insertions(+), 26 deletions(-)

Index: current/include/linux/cgroup_subsys.h
===================================================================
--- current.orig/include/linux/cgroup_subsys.h
+++ current/include/linux/cgroup_subsys.h
@@ -11,6 +11,12 @@
 SUBSYS(cpuset)
 #endif

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

 #ifdef CONFIG_CGROUP_DEBUG
Index: current/include/linux/cpu_acct.h
===================================================================
--- /dev/null
+++ current/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 cpuacct_charge(struct task_struct *, u64 cputime);
+#else
+static inline void cpuacct_charge(struct task_struct *p, u64 cputime) {}
+#endif
+
+#endif
Index: current/init/Kconfig
```

```
===================================================================
--- current.orig/init/Kconfig
+++ current/init/Kconfig
@@ -354,6 +354,13 @@ config FAIR_CGROUP_SCHED

 endchoice

+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 SYSFS_DEPRECATED
  bool "Create deprecated sysfs files"
  default y
Index: current/kernel/Makefile
===================================================================
--- current.orig/kernel/Makefile
+++ current/kernel/Makefile
@@ -40,6 +40,7 @@ obj-$(CONFIG_COMPAT) += compat.o
 obj-$(CONFIG_CGROUPS) += cgroup.o
 obj-$(CONFIG_CGROUP_DEBUG) += cgroup_debug.o
 obj-$(CONFIG_CPUSETS) += cpuset.o
+obj-$(CONFIG_CGROUP_CPUACCT) += cpu_acct.o
 obj-$(CONFIG_CGROUP_NS) += ns_cgroup.o
 obj-$(CONFIG_IKCONFIG) += configs.o
 obj-$(CONFIG_STOP_MACHINE) += stop_machine.o
Index: current/kernel/cpu_acct.c
===================================================================
--- /dev/null
+++ current/kernel/cpu_acct.c
@@ -0,0 +1,103 @@
+/*
+ * 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.
+ */
```

```
+
+#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 (in nanoseconds) */
+ u64 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 cpuacct *task_ca(struct task_struct *task)
+{
+ return container_of(task_subsys_state(task, cpuacct_subsys_id),
+       struct cpuacct, css);
+}
+
+static struct cgroup_subsys_state *cpuacct_create(
+ struct cgroup_subsys *ss, struct cgroup *cont)
+{
+ struct cpuacct *ca = kzalloc(sizeof(*ca), GFP_KERNEL);
+
+ if (!ca)
+  return ERR_PTR(-ENOMEM);
+ spin_lock_init(&ca->lock);
+ return &ca->css;
+}
+
+static void cpuacct_destroy(struct cgroup_subsys *ss,
+       struct cgroup *cont)
+{
+ kfree(cgroup_ca(cont));
+}
+
+static u64 cpuusage_read(struct cgroup *cont, struct cftype *cft)
+{
```

```
+ struct cpuacct *ca = cgroup_ca(cont);
+
+ return ca->time;
+}
+
+static struct cftype files[] = {
+ {
+  .name = "usage",
+  .read_uint = cpuusage_read,
+ },
+};
+
+static int cpuacct_populate(struct cgroup_subsys *ss, struct cgroup *cont)
+{
+ return cgroup_add_files(cont, ss, files, ARRAY_SIZE(files));
+}
+
+void cpuacct_charge(struct task_struct *task, u64 cputime)
+{
+ struct cpuacct *ca;
+ unsigned long flags;
+
+ if (!cpuacct_subsys.active)
+  return;
+ rcu_read_lock();
+ ca = task_ca(task);
+ if (ca) {
+  spin_lock_irqsave(&ca->lock, flags);
+  ca->time += cputime;
+  spin_unlock_irqrestore(&ca->lock, flags);
+ }
+ rcu_read_unlock();
+}
+
+struct cgroup_subsys cpuacct_subsys = {
+ .name = "cpuacct",
+ .create = cpuacct_create,
+ .destroy = cpuacct_destroy,
+ .populate = cpuacct_populate,
+ .subsys_id = cpuacct_subsys_id,
+};
Index: current/kernel/sched.c
===================================================================
--- current.orig/kernel/sched.c
+++ current/kernel/sched.c
@@ -52,6 +52,7 @@
 #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>
@@ -7221,38 +7222,12 @@ static u64 cpu_shares_read_uint(struct c
  return (u64) tg->shares;
 }

-static u64 cpu_usage_read(struct cgroup *cgrp, struct cftype *cft)
-{
- struct task_group *tg = cgroup_tg(cgrp);
- unsigned long flags;
- u64 res = 0;
- int i;
-
- for_each_possible_cpu(i) {
-  /*
-   * Lock to prevent races with updating 64-bit counters
-   * on 32-bit arches.
-   */
-  spin_lock_irqsave(&cpu_rq(i)->lock, flags);
-  res += tg->se[i]->sum_exec_runtime;
-  spin_unlock_irqrestore(&cpu_rq(i)->lock, flags);
- }
- /* Convert from ns to ms */
- do_div(res, NSEC_PER_MSEC);
-
- return res;
-}
-
 static struct cftype cpu_files[] = {
  {
   .name = "shares",
   .read_uint = cpu_shares_read_uint,
   .write_uint = cpu_shares_write_uint,
  },
- {
-  .name = "usage",
-  .read_uint = cpu_usage_read,
- },
 };

 static int cpu_cgroup_populate(struct cgroup_subsys *ss, struct cgroup *cont)
Index: current/kernel/sched_fair.c
===================================================================
--- current.orig/kernel/sched_fair.c
+++ current/kernel/sched_fair.c
```

```
@@ -351,6 +351,11 @@ static void update_curr(struct cfs_rq *c

  __update_curr(cfs_rq, curr, delta_exec);
  curr->exec_start = now;
+ if (entity_is_task(curr)) {
+  struct task_struct *curtask = task_of(curr);
+
+  cpuacct_charge(curtask, delta_exec);
+ }
 }

 static inline void
Index: current/kernel/sched_rt.c
===================================================================
--- current.orig/kernel/sched_rt.c
+++ current/kernel/sched_rt.c
@@ -23,6 +23,7 @@ static void update_curr_rt(struct rq *rq

  curr->se.sum_exec_runtime += delta_exec;
  curr->se.exec_start = rq->clock;
+ cpuacct_charge(curr, delta_exec);
 }

 static void enqueue_task_rt(struct rq *rq, struct task_struct *p, int wakeup)
```

--
Regards,
vatsa

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