Subject: Re: [PATCH] sched: cpu accounting controller
Posted by akpm on Thu, 29 Nov 2007 19:30:35 GMT
View Forum Message <> Reply to Message

On Fri, 30 Nov 2007 00:47:37 +0530
Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:

> 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.
>
> ...
>
>  - Make the accounting scalable on SMP systems (perhaps
>     for 2.6.25)

That sounds like a rather important todo.  How bad is it now?

> 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);

                                        ^ no "p"

> +#else
> +static inline void cpuacct_charge(struct task_struct *p, u64 cputime) {}

                                            ^ "p"

> +#endif

Personally I think "p" is a dopey name - we tend to standardise on "tsk"
for this.

> --- /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>

I don't think this inclusion is needed?

> +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;

This can be made static.