Subject: Revert for cgroups CPU accounting subsystem patch
Posted by Paul Menage on Tue, 13 Nov 2007 05:25:32 GMT
View Forum Message <> Reply to Message

Hi Linus,

Please can you revert commit 62d0df64065e7c135d0002f069444fbdfc64768f,
entitled "Task Control Groups: example CPU accounting subsystem" ?

This was originally intended as a simple initial example of how to
create a control groups subsystem; it wasn't intended for mainline,
but I didn't make this clear enough to Andrew.

The CFS cgroup subsystem now has better functionality for the
per-cgroup usage accounting (based directly on CFS stats) than the
"usage" status file in this patch, and the "load" status file is
rather simplistic - although having a per-cgroup load average report
would be a useful feature, I don't believe this patch actually
provides it. If it gets into the final 2.6.24 we'd probably have to
support this interface for ever.

Thanks,

Paul

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

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Srivatsa Vaddagiri on Tue, 13 Nov 2007 05:51:46 GMT
View Forum Message <> Reply to Message

On Mon, Nov 12, 2007 at 09:25:32PM -0800, Paul Menage wrote:
> Hi Linus,
>
> Please can you revert commit 62d0df64065e7c135d0002f069444fbdfc64768f,
> entitled "Task Control Groups: example CPU accounting subsystem" ?

Hi Paul,
 On second thoughts, this may be a usefull controller of its own.
Say I just want to "monitor" usage (for accounting purpose) of a group of
tasks, but don't want to control their cpu consumption, then cpuacct
controller would come in handy.

--
Regards,

vatsa

_____

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

---

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Paul Menage on Tue, 13 Nov 2007 06:05:24 GMT
View Forum Message <> Reply to Message

On Nov 12, 2007 10:00 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> On second thoughts, this may be a usefull controller of its own.
> Say I just want to "monitor" usage (for accounting purpose) of a group of
> tasks, but don't want to control their cpu consumption, then cpuacct
> controller would come in handy.
>

That's plausible, but having two separate ways of tracking and
reporting the CPU usage of a cgroup seems wrong.

How bad would it be in your suggested case if you just give each
cgroup the same weight? So there would be fair scheduling between
cgroups, which seems as reasonable as any other choice in the event
that the CPU is contended.

Paul

_____

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

---

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Balbir Singh on Tue, 13 Nov 2007 07:00:20 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On Nov 12, 2007 10:00 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>> On second thoughts, this may be a usefull controller of its own.
>> Say I just want to "monitor" usage (for accounting purpose) of a group of
>> tasks, but don't want to control their cpu consumption, then cpuacct
>> controller would come in handy.
>>
>
> That's plausible, but having two separate ways of tracking and
> reporting the CPU usage of a cgroup seems wrong.

>
> How bad would it be in your suggested case if you just give each
> cgroup the same weight? So there would be fair scheduling between
> cgroups, which seems as reasonable as any other choice in the event
> that the CPU is contended.
>

Right now, one of the limitations of the CPU controller is that
the moment you create another control group, the bandwidth gets
divided by the default number of shares. We can't create groups
just for monitoring. cpu_acct fills this gap. I think in the
long run, we should move the helper functions into cpu_acct.c
and the interface logic into kernel/sched.c (cpu controller).

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Paul Menage on Tue, 13 Nov 2007 07:10:11 GMT
View Forum Message <> Reply to Message

On Nov 12, 2007 11:00 PM, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> Right now, one of the limitations of the CPU controller is that
> the moment you create another control group, the bandwidth gets
> divided by the default number of shares. We can't create groups
> just for monitoring.

Could we get around this with, say, a flag that always treats a CFS
schedulable entity as having a weight equal to the number of runnable
tasks in it? So CPU bandwidth would be shared between groups in
proportion to the number of runnable tasks, which would distribute the
cycles approximately equivalently to them all being separate
schedulable entities.

> cpu_acct fills this gap.

Agreed, but not in the right way IMO.

Paul

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Balbir Singh on Tue, 13 Nov 2007 07:29:59 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On Nov 12, 2007 11:00 PM, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>> Right now, one of the limitations of the CPU controller is that
>> the moment you create another control group, the bandwidth gets
>> divided by the default number of shares. We can't create groups
>> just for monitoring.
>
> Could we get around this with, say, a flag that always treats a CFS
> schedulable entity as having a weight equal to the number of runnable
> tasks in it? So CPU bandwidth would be shared between groups in
> proportion to the number of runnable tasks, which would distribute the
> cycles approximately equivalently to them all being separate
> schedulable entities.
>

I think it's a good hack, but not sure about the complexity to implement
the code. I worry that if the number of tasks increase (say run into
thousands for one or more groups and a few groups have just a few
tasks), we'll lose out on accuracy.

>> cpu_acct fills this gap.
>
> Agreed, but not in the right way IMO.
>

I think we already have the code, we need to make it more useful and
reusable.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Paul Menage on Tue, 13 Nov 2007 07:34:47 GMT
View Forum Message <> Reply to Message

On Nov 12, 2007 11:29 PM, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> I think it's a good hack, but not sure about the complexity to implement
> the code. I worry that if the number of tasks increase (say run into
> thousands for one or more groups and a few groups have just a few
> tasks), we'll lose out on accuracy.

But for the case we're looking at, you've already said that you don't care much about actually controlling CPU, just monitoring it. So what does it matter if the CPU sharing isn't perfectly accurate? I don't think that you're painting a realistic scenario.

>
> I think we already have the code, we need to make it more useful and
> reusable.

In that case we should take the existing cpu_acct code out of 2.6.24 until the API has been made more useful and reusable, so that we don't have to support it for ever.

Paul

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

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Srivatsa Vaddagiri on Tue, 13 Nov 2007 07:37:40 GMT
View Forum Message <> Reply to Message

On Mon, Nov 12, 2007 at 10:05:24PM -0800, Paul Menage wrote:
> On Nov 12, 2007 10:00 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> > On second thoughts, this may be a usefull controller of its own.
> > Say I just want to "monitor" usage (for accounting purpose) of a group of
> > tasks, but don't want to control their cpu consumption, then cpuacct
> > controller would come in handy.
> >
>
> That's plausible, but having two separate ways of tracking and
> reporting the CPU usage of a cgroup seems wrong.
>
> How bad would it be in your suggested case if you just give each
> cgroup the same weight?

That's still introducing a deviation from the normal behavior we would
have had we allowed all tasks to be part of the same "control" group/runqueue.

For ex: using nice value to vary bandwidth between tasks makes sense if
they are all part of the same group.

Also an application with more tasks will get more cpu power (as intended)
compared to another app with less tasks, provided they are all
part of the same group.

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).

This will let account and control grouping to be independent if desired.

What do you think?

> So there would be fair scheduling between
> cgroups, which seems as reasonable as any other choice in the event
> that the CPU is contended.

--
Regards,
vatsa

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

---

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Srivatsa Vaddagiri on Tue, 13 Nov 2007 07:49:02 GMT
View Forum Message <> Reply to Message

On Tue, Nov 13, 2007 at 12:59:59PM +0530, Balbir Singh wrote:
> Paul Menage wrote:
> > On Nov 12, 2007 11:00 PM, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
> >> Right now, one of the limitations of the CPU controller is that
> >> the moment you create another control group, the bandwidth gets
> >> divided by the default number of shares. We can't create groups
> >> just for monitoring.
> >

> > Could we get around this with, say, a flag that always treats a CFS
> > schedulable entity as having a weight equal to the number of runnable
> > tasks in it? So CPU bandwidth would be shared between groups in
> > proportion to the number of runnable tasks, which would distribute the
> > cycles approximately equivalently to them all being separate
> > schedulable entities.
> >
>
> I think it's a good hack, but not sure about the complexity to implement
> the code.

I agree that it would be adding unnecessary complexity, just to meet the accounting needs.

Thinking of it more, this requirement to "group tasks for only accounting purpose" may be required for other resources (mem, io, network etc) as well? Should we have a generic accounting controller which can provide these various resource usgae stats for a group (cpu, mem etc) by iterating thr' the task list for the group and summing up the corresponding stats already present in task structure?

--
Regards,
vatsa

_____

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

---

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Paul Menage on Tue, 13 Nov 2007 07:57:03 GMT
View Forum Message <> Reply to Message

On Nov 12, 2007 11:48 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> 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

_____

Subject: Re: Revert for cgroups CPU accounting subsystem patch
Posted by Paul Menage on Tue, 13 Nov 2007 07:59:50 GMT
View Forum Message <> Reply to Message

On Nov 12, 2007 11:59 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>
> Thinking of it more, this requirement to "group tasks for only accounting
> purpose" may be required for other resources (mem, io, network etc) as well?
> Should we have a generic accounting controller which can provide these
> various resource usgae stats for a group (cpu, mem etc) by iterating thr' the
> task list for the group and summing up the corresponding stats already present
> in task structure?

In theory it could certainly be useful - but it can only be done if
something in the kernel is already keeping track of resources on a
per-task basis. This works for CPU, but isn't really possible for
memory without doing something lame like just adding up the tasks' RSS
values (since the page accounting is the hard part - limiting is easy
once you have accounting).

Paul

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

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

---

Subject: Re: [PATCH] sched: cpu accounting controller
Posted by Ingo Molnar on Thu, 29 Nov 2007 19:20:58 GMT
View Forum Message <> Reply to Message

* Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:

> 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.

ok, this looks certainly doable for v2.6.24. I've added it to the
scheduler fixes queue and will let it brew there for a few days and send
it to Linus after that if everything goes fine - unless anyone objects.
This is pre-existing, tested code so the risk is small.

 Ingo

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

---

Subject: Re: [PATCH] sched: cpu accounting controller
Posted by Srivatsa Vaddagiri on Thu, 29 Nov 2007 19:28:48 GMT
View Forum Message <> Reply to Message

On Thu, Nov 29, 2007 at 08:20:58PM +0100, Ingo Molnar wrote:
> ok, this looks certainly doable for v2.6.24. I've added it to the
> scheduler fixes queue and will let it brew there for a few days and send
> it to Linus after that if everything goes fine - unless anyone objects.

Thanks.

--
Regards,
vatsa

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

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.

Subject: Re: [PATCH] sched: cpu accounting controller

On Thu, Nov 29, 2007 at 11:30:35AM -0800, Andrew Morton wrote:
> > - Make the accounting scalable on SMP systems (perhaps
> > for 2.6.25)
>
> That sounds like a rather important todo. How bad is it now?

It is indeed an important todo. Right now we take a per-group global
lock on every accounting update (which can be very frequent) and hence
it is pretty bad.

Ingo had expressed the need to reintroduce this patch asap and hence I resent
it w/o attacking the scalability part. I will take a shot at scalability
enhancements tomorrow and send it as a separate patch.

> > +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.

This symbol is needed in kernel/cgroup.c file, where it does this:

static struct cgroup_subsys *subsys[] = {
#include <linux/cgroup_subsys.h>
};

and hence it cant be static. Thanks for the rest of your comments. I
have fixed them in this version below:

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 | 101 ++
kernel/sched.c | 27 ----------
kernel/sched_fair.c | 5 ++
kernel/sched_rt.c | 1
8 files changed, 136 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 *tsk, u64 cputime);
+#else
+static inline void cpuacct_charge(struct task_struct *tsk, 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 +1,101 @@
+/*
+ * 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>
+
+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

---

Subject: [PATCH] sched: cpu accounting controller (V2)
Posted by Srivatsa Vaddagiri on Fri, 30 Nov 2007 12:32:37 GMT
View Forum Message <> Reply to Message

On Fri, Nov 30, 2007 at 01:48:33AM +0530, Srivatsa Vaddagiri wrote:
> It is indeed an important todo. Right now we take a per-group global
> lock on every accounting update (which can be very frequent) and hence
> it is pretty bad.
>
> Ingo had expressed the need to reintroduce this patch asap and hence I resent
> it w/o attacking the scalability part. I will take a shot at scalability
> enhancements tomorrow and send it as a separate patch.

Here's V2 of the cpu acccounting controller patch, which makes
accounting scale better on SMP systems by splitting the usage counter to
be per-cpu.

---->

Commit cfb5285660aad4931b2ebbfa902ea48a37dfffa1 removed a useful feature for
us, which provided a cpu accounting resource controller. This feature would be
useful 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
- Make accounting scalable on SMP systems by splitting the usage
counter to be per-cpu
- Move the code from kernel/cpu_acct.c to kernel/sched.c (since the
code is not big enough to warrant a new file and also this rightly
needs to live inside the scheduler. Also things like accessing
rq->lock while reading cpu usage becomes easier if the code lived in
kernel/sched.c)

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

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

---
include/linux/cgroup_subsys.h | 7 +
init/Kconfig | 7 +
kernel/sched.c | 155 ++++++++++++++++++++++++++++++++++++++--------
kernel/sched_fair.c | 6 +
kernel/sched_rt.c | 1
5 files changed, 150 insertions(+), 26 deletions(-)

Index: current/include/linux/cgroup_subsys.h
===
--- current.orig/include/linux/cgroup_subsys.h
+++ current/include/linux/cgroup_subsys.h
@@ -30,3 +30,10 @@ SUBSYS(cpu_cgroup)
#endif

/* */
+
+#ifdef CONFIG_CGROUP_CPUACCT
+SUBSYS(cpuacct)
+#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/sched.c
===================================================================
--- current.orig/kernel/sched.c
+++ current/kernel/sched.c
@@ -854,6 +854,12 @@ iter_move_one_task(struct rq *this_rq, i
      struct rq_iterator *iterator);
 #endif

+#ifdef CONFIG_CGROUP_CPUACCT
+static void cpuacct_charge(struct task_struct *tsk, u64 cputime);
+#else
+static inline void cpuacct_charge(struct task_struct *tsk, u64 cputime) {}
+#endif
+
 #include "sched_stats.h"
 #include "sched_idletask.c"
 #include "sched_fair.c"
@@ -7221,38 +7227,12 @@ 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)
@@ -7272,3 +7252,126 @@ struct cgroup_subsys cpu_cgroup_subsys =
 };

 #endif /* CONFIG_FAIR_CGROUP_SCHED */
+
+#ifdef CONFIG_CGROUP_CPUACCT
+
+/*
+ * CPU accounting code for task groups.
+ *
+ * Based on the work by Paul Menage (menage@google.com) and Balbir Singh
+ * (balbir@in.ibm.com).
+ */
+
+/* track cpu usage of a group of tasks */
+struct cpuacct {
+ struct cgroup_subsys_state css;
+ /* cpuusage holds pointer to a u64-type object on every cpu */
+ u64 *cpuusage;
+};
+
+struct cgroup_subsys cpuacct_subsys;
+
+/* return cpu accounting group corresponding to this container */
+static inline struct cpuacct *cgroup_ca(struct cgroup *cont)
```

```
+{
+ return container_of(cgroup_subsys_state(cont, cpuacct_subsys_id),
+       struct cpuacct, css);
+}
+
+/* return cpu accounting group to which this task belongs */
+static inline struct cpuacct *task_ca(struct task_struct *tsk)
+{
+ return container_of(task_subsys_state(tsk, cpuacct_subsys_id),
+       struct cpuacct, css);
+}
+
+/* create a new cpu accounting group */
+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);
+
+ ca->cpuusage = alloc_percpu(u64);
+ if (!ca->cpuusage) {
+  kfree(ca);
+  return ERR_PTR(-ENOMEM);
+ }
+
+ return &ca->css;
+}
+
+/* destroy an existing cpu accounting group */
+static void cpuacct_destroy(struct cgroup_subsys *ss,
+       struct cgroup *cont)
+{
+ struct cpuacct *ca = cgroup_ca(cont);
+
+ free_percpu(ca->cpuusage);
+ kfree(ca);
+}
+
+/* return total cpu usage (in nanoseconds) of a group */
+static u64 cpuusage_read(struct cgroup *cont, struct cftype *cft)
+{
+ struct cpuacct *ca = cgroup_ca(cont);
+ u64 totalcpuusage = 0;
+ int i;
+
+ for_each_possible_cpu(i) {
```

```
+  u64 *cpuusage = percpu_ptr(ca->cpuusage, i);
+
+  /*
+   * Take rq->lock to make 64-bit addition safe on 32-bit
+   * platforms.
+   */
+  spin_lock_irq(&cpu_rq(i)->lock);
+  totalcpuusage += *cpuusage;
+  spin_unlock_irq(&cpu_rq(i)->lock);
+ }
+
+ return totalcpuusage;
+}
+
+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));
+}
+
+/*
+ * charge this task's execution time to its accounting group.
+ *
+ * called with rq->lock held.
+ */
+static void cpuacct_charge(struct task_struct *tsk, u64 cputime)
+{
+ struct cpuacct *ca;
+
+ if (!cpuacct_subsys.active)
+  return;
+
+ ca = task_ca(tsk);
+ if (ca) {
+  u64 *cpuusage = percpu_ptr(ca->cpuusage, task_cpu(tsk));
+
+  *cpuusage += cputime;
+ }
+}
+
+struct cgroup_subsys cpuacct_subsys = {
+ .name = "cpuacct",
```

```
+ .create = cpuacct_create,
+ .destroy = cpuacct_destroy,
+ .populate = cpuacct_populate,
+ .subsys_id = cpuacct_subsys_id,
+};
+#endif /* CONFIG_CGROUP_CPUACCT */
Index: current/kernel/sched_fair.c
===================================================================
--- current.orig/kernel/sched_fair.c
+++ current/kernel/sched_fair.c
@@ -351,6 +351,12 @@ 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
```

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Ingo Molnar on Fri, 30 Nov 2007 12:35:13 GMT
View Forum Message <> Reply to Message

* Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:

> Here's V2 of the cpu acccounting controller patch, which makes
> accounting scale better on SMP systems by splitting the usage counter
> to be per-cpu.

thanks, applied. But you dont seem to have incorporated all of the
review feedback from Andrew. (such as making cpuacct_subsys static)

 Ingo

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

---

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Balbir Singh on Fri, 30 Nov 2007 12:45:43 GMT
View Forum Message <> Reply to Message

Srivatsa Vaddagiri wrote:
[snip]

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

--
 Warm Regards,
 Balbir Singh
 Linux Technology Center
 IBM, ISTL

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

---

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Srivatsa Vaddagiri on Fri, 30 Nov 2007 13:09:03 GMT
View Forum Message <> Reply to Message

On Fri, Nov 30, 2007 at 01:35:13PM +0100, Ingo Molnar wrote:
> * Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>
> > Here's V2 of the cpu acccounting controller patch, which makes

> > accounting scale better on SMP systems by splitting the usage counter
> > to be per-cpu.
>
> thanks, applied. But you dont seem to have incorporated all of the
> review feedback from Andrew. (such as making cpuacct_subsys static)

cpuacct_subsys can't be made static, as I have pointed out at
http://marc.info/?l=linux-kernel&m=119636730930886. Other than that,
AFAICS, rest of Andrew's comments have been taken care of. If there are any
remaining that I have over-looked, I had be happy to fix them.

--
Regards,
vatsa

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

---

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Ingo Molnar on Fri, 30 Nov 2007 13:34:55 GMT
View Forum Message <> Reply to Message

* Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:

> On Fri, Nov 30, 2007 at 01:35:13PM +0100, Ingo Molnar wrote:
> > * Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> >
> > > Here's V2 of the cpu acccounting controller patch, which makes
> > > accounting scale better on SMP systems by splitting the usage counter
> > > to be per-cpu.
> >
> > thanks, applied. But you dont seem to have incorporated all of the
> > review feedback from Andrew. (such as making cpuacct_subsys static)
>
> cpuacct_subsys can't be made static, as I have pointed out at
> http://marc.info/?l=linux-kernel&m=119636730930886. [...]

ah, ok - i missed that.

 Ingo

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

---

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Ingo Molnar on Fri, 30 Nov 2007 13:53:31 GMT
View Forum Message <> Reply to Message

* Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> Srivatsa Vaddagiri wrote:
> [snip]
>
> > Signed-off-by: Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com>
> >
> Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>

thx. Have you done targeted testing of it as well? The (v2) patch is in
sched-devel.git:

  git://git.kernel.org/pub/scm/linux/kernel/git/mingo/linux-2.6-sched-devel.git

 Ingo

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

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Balbir Singh on Fri, 30 Nov 2007 14:00:01 GMT
View Forum Message <> Reply to Message

Ingo Molnar wrote:
> * Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
>> Srivatsa Vaddagiri wrote:
>> [snip]
>>
>>> Signed-off-by: Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com>
>>>
>> Signed-off-by: Balbir Singh <balbir@linux.vnet.ibm.com>
>
> thx. Have you done targeted testing of it as well? The (v2) patch is in
> sched-devel.git:
>
>   git://git.kernel.org/pub/scm/linux/kernel/git/mingo/linux-2.6-sched-devel.git
>
> Ingo

Vatsa tested the patches. I am going to independently test them as well.
In the process of getting the test setup up and running and doing some

testing.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

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

---

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Ingo Molnar on Fri, 30 Nov 2007 19:46:47 GMT
View Forum Message <> Reply to Message

* Balbir Singh <balbir@linux.vnet.ibm.com> wrote:

> > thx. Have you done targeted testing of it as well? The (v2) patch is
> > in sched-devel.git:
> >
> > git://git.kernel.org/pub/scm/linux/kernel/git/mingo/linux-2.6-sched-devel.git
> > Ingo
>
> Tested the patches on top of 2.6.24-rc3. The patches work fine. Ran
> some simple tests like cpuspin (spin on the cpu), ran several tasks in
> the same group and timed them. Compared their time stamps with
> cpuacct.usage.
>
> Tested-by: Balbir Singh <balbir@linux.vnet.ibm.com>

thanks!

Ingo

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

---

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Paul Menage on Sat, 01 Dec 2007 07:48:15 GMT
View Forum Message <> Reply to Message

Hi Vatsa,

Thanks, this looks pretty good.

On Nov 30, 2007 4:42 AM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>
> - 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.

The "load" value was never a load average, it was just a count of the
% cpu time used in the previous fixed window of time, updated at the
end of each window.

Maybe we can instead do something based tracking the length of the run
queue for the cgroup?

Paul

Subject: Re: [PATCH] sched: cpu accounting controller (V2)
Posted by Balbir Singh on Sat, 01 Dec 2007 09:51:09 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> Hi Vatsa,
>
> Thanks, this looks pretty good.
>
> On Nov 30, 2007 4:42 AM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>> - 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.
>
> The "load" value was never a load average, it was just a count of the
> % cpu time used in the previous fixed window of time, updated at the
> end of each window.
>
> Maybe we can instead do something based tracking the length of the run
> queue for the cgroup?
>
> Paul

Length of the runqueue gives no idea of the weight of the tasks on the
runqueue. I still prefer to have a top'ish view of the load on the
system. The load average can be extracted using container group statistics.

--
Warm Regards,
Balbir Singh
Linux Technology Center
IBM, ISTL

_____

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