Subject: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Tue, 23 Oct 2007 00:49:39 GMT
View Forum Message <> Reply to Message

Report CPU usage in CFS Cgroup directories

Adds a cpu.usage file to the CFS cgroup that reports CPU usage in
milliseconds for that cgroup's tasks

This replaces the "example CPU Accounting CGroup subsystem" that
was merged into mainline last week.

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

---
 kernel/sched.c | 32 ++++++++++++++++++++++++++++-----
 1 file changed, 27 insertions(+), 5 deletions(-)

Index: container-2.6.23-mm1/kernel/sched.c
===
--- container-2.6.23-mm1.orig/kernel/sched.c
+++ container-2.6.23-mm1/kernel/sched.c
@@ -7005,15 +7005,37 @@ static u64 cpu_shares_read_uint(struct c
  return (u64) tg->shares;
 }

-static struct cftype cpu_shares = {
- .name = "shares",
- .read_uint = cpu_shares_read_uint,
- .write_uint = cpu_shares_write_uint,
+static u64 cpu_usage_read(struct cgroup *cgrp, struct cftype *cft)
+{
+ struct task_group *tg = cgroup_tg(cgrp);
+ int i;
+ u64 res = 0;
+ for_each_possible_cpu(i) {
+ unsigned long flags;
+ spin_lock_irqsave(&tg->cfs_rq[i]->rq->lock, flags);
+ res += tg->se[i]->sum_exec_runtime;
+ spin_unlock_irqrestore(&tg->cfs_rq[i]->rq->lock, flags);
+ }
+ /* Convert from ns to ms */
+ do_div(res, 1000000);
+ 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)
 {
- return cgroup_add_file(cont, ss, &cpu_shares);
+ return cgroup_add_files(cont, ss, cpu_files, ARRAY_SIZE(cpu_files));
 }

 struct cgroup_subsys cpu_cgroup_subsys = {
```

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Srivatsa Vaddagiri on Tue, 23 Oct 2007 02:30:41 GMT
View Forum Message <> Reply to Message

On Mon, Oct 22, 2007 at 05:49:39PM -0700, Paul Menage wrote:
> +static u64 cpu_usage_read(struct cgroup *cgrp, struct cftype *cft)
> +{
> + struct task_group *tg = cgroup_tg(cgrp);
> + int i;
> + u64 res = 0;
> + for_each_possible_cpu(i) {
> + unsigned long flags;
> + spin_lock_irqsave(&tg->cfs_rq[i]->rq->lock, flags);

Is the lock absolutely required here?

Hmm .. I hope the cgroup code prevents a task group from being destroyed while
we are still reading a task group's cpu usage. Is that so?

> + res += tg->se[i]->sum_exec_runtime;
> + spin_unlock_irqrestore(&tg->cfs_rq[i]->rq->lock, flags);
> + }
> + /* Convert from ns to ms */
> + do_div(res, 1000000);
> + return res;

> +}

--
Regards,
vatsa

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

---

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Balbir Singh on Tue, 23 Oct 2007 03:17:09 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> Report CPU usage in CFS Cgroup directories
>
> Adds a cpu.usage file to the CFS cgroup that reports CPU usage in
> milliseconds for that cgroup's tasks
>
> This replaces the "example CPU Accounting CGroup subsystem" that
> was merged into mainline last week.
>
> Signed-off-by: Paul Menage <menage@google.com>
>
> ---
> kernel/sched.c | 32 ++++++++++++++++++++++++++++-----
> 1 file changed, 27 insertions(+), 5 deletions(-)
>
> Index: container-2.6.23-mm1/kernel/sched.c
> ===
> --- container-2.6.23-mm1.orig/kernel/sched.c
> +++ container-2.6.23-mm1/kernel/sched.c
> @@ -7005,15 +7005,37 @@ static u64 cpu_shares_read_uint(struct c
> return (u64) tg->shares;
> }
>
> -static struct cftype cpu_shares = {
> - .name = "shares",
> - .read_uint = cpu_shares_read_uint,
> - .write_uint = cpu_shares_write_uint,
> +static u64 cpu_usage_read(struct cgroup *cgrp, struct cftype *cft)
> +{
> + struct task_group *tg = cgroup_tg(cgrp);
> + int i;
> + u64 res = 0;
> + for_each_possible_cpu(i) {

```
> +       unsigned long flags;
> +       spin_lock_irqsave(&tg->cfs_rq[i]->rq->lock, flags);
> +       res += tg->se[i]->sum_exec_runtime;
> +       spin_unlock_irqrestore(&tg->cfs_rq[i]->rq->lock, flags);
> +   }
> +   /* Convert from ns to ms */
> +   do_div(res, 1000000);
> +   return res;
> +}
> +
```

I think we also need the notion of load, like we have in cpu_acct.c
Don't we need to do a css_get() on the cgrp to ensure that the cgroup
does not go away if it's empty and someone does an rmdir on it?

```
> +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)
> {
> -   return cgroup_add_file(cont, ss, &cpu_shares);
> +    return cgroup_add_files(cont, ss, cpu_files, ARRAY_SIZE(cpu_files));
> }
>
> struct cgroup_subsys cpu_cgroup_subsys = {
```

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

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Tue, 23 Oct 2007 06:06:54 GMT
View Forum Message <> Reply to Message

On 10/22/07, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> On Mon, Oct 22, 2007 at 05:49:39PM -0700, Paul Menage wrote:
> > +static u64 cpu_usage_read(struct cgroup *cgrp, struct cftype *cft)
> > +{
> > + struct task_group *tg = cgroup_tg(cgrp);
> > + int i;
> > + u64 res = 0;
> > + for_each_possible_cpu(i) {
> > + unsigned long flags;
> > + spin_lock_irqsave(&tg->cfs_rq[i]->rq->lock, flags);
>
> Is the lock absolutely required here?

I'm not sure, I was hoping you or Ingo could comment on this. But some kind of locking seems to required at least on 32-bit platforms, since sum_exec_runtime is a 64-bit number.

>
> Hmm .. I hope the cgroup code prevents a task group from being destroyed while
> we are still reading a task group's cpu usage. Is that so?

Good point - cgroups certainly prevents a cgroup itself from being freed while a control file is being read in an RCU section, and prevents a task group from being destroyed when that task group has been read via a task's cgroups pointer and the reader is still in an RCU section, but we need a generic protection for subsystem state objects being accessed via control files too.

Using cgroup_mutex is certainly possible for now, although more heavy-weight than I'd like long term. Using css_get isn't the right approach, I think - we shouldn't be able to cause an rmdir to fail due to a concurrent read.

Paul

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

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Tue, 23 Oct 2007 06:09:29 GMT
View Forum Message <> Reply to Message

On 10/22/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> I think we also need the notion of load, like we have in cpu_acct.c

Yes, a notion of load would be good - but the "load" calculated by
cpu_acct.c isn't all that useful yet - it's just a total of the CPU
cycles used in the 10 second time interval prior to the current
interval. Something that's more analagous to the real machine load
would be nice, and I hope the numbers could be made available via CFS.

> Don't we need to do a css_get() on the cgrp to ensure that the cgroup
> does not go away if it's empty and someone does an rmdir on it?

See my other email. Yes, I think we need more here, but probably
something more generic.

Paul

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Tue, 23 Oct 2007 07:21:59 GMT
View Forum Message <> Reply to Message

On 10/22/07, Paul Menage <menage@google.com> wrote:
>
> Using cgroup_mutex is certainly possible for now, although more
> heavy-weight than I'd like long term. Using css_get isn't the right
> approach, I think - we shouldn't be able to cause an rmdir to fail due
> to a concurrent read.
>

OK, the obvious solution is to use the same approach for subsystem
state objects as we do for the struct cgroup itself - move the calls
to the subsystem destroy methods to cgroup_diput. A control file
dentry will keep alive the parent dir's dentry, which will keep alive
the cgroup and (with this change) the subsystem state objects too.

The only potential drawback that I can see is that an open fd on a
cgroup directory or a control file will keep more memory alive than it
would have done previously.

Paul

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Balbir Singh on Tue, 23 Oct 2007 07:49:35 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On 10/22/07, Paul Menage <menage@google.com> wrote:
>> Using cgroup_mutex is certainly possible for now, although more
>> heavy-weight than I'd like long term. Using css_get isn't the right
>> approach, I think - we shouldn't be able to cause an rmdir to fail due
>> to a concurrent read.
>>

Well, most people who care about deletion will use the notify_on_release
callback and retry. I think we have a good mechanism in place for
deletion, so I would not worry about read delaying deletion.

--
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 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Tue, 23 Oct 2007 07:53:16 GMT
View Forum Message <> Reply to Message

On 10/23/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> Well, most people who care about deletion will use the notify_on_release
> callback and retry.

I'm not convinced this is true. Certainly the userspace approaches
we're developing at Google don't (currently) use notify_on_release.

Paul

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Balbir Singh on Tue, 23 Oct 2007 07:57:40 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On 10/23/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>> Well, most people who care about deletion will use the notify_on_release
>> callback and retry.
>
> I'm not convinced this is true. Certainly the userspace approaches
> we're developing at Google don't (currently) use notify_on_release.
>
> Paul

Well, without notify_on_release you can never be sure if a new task
got added to the control group or if someone acquired a reference
to it. I can't think of a safe way of removing control groups/cpusets
without using notify_on_release.

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

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Tue, 23 Oct 2007 08:08:07 GMT
View Forum Message <> Reply to Message

On 10/23/07, Balbir Singh <balbir@linux.vnet.ibm.com> wrote:
>
> Well, without notify_on_release you can never be sure if a new task
> got added to the control group or if someone acquired a reference
> to it. I can't think of a safe way of removing control groups/cpusets
> without using notify_on_release.

>

Using notify_on_release doesn't make any guarantees that the
notification, or the action based on that notification, will occur
before the cgroup becomes busy again.

An rmdir() on a busy cgroup is perfectly safe, it just fails with
-EBUSY. I'd just like to avoid making active control file fds trigger
than failure mode.

Paul

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Srivatsa Vaddagiri on Tue, 23 Oct 2007 16:31:35 GMT
View Forum Message <> Reply to Message

On Mon, Oct 22, 2007 at 11:06:54PM -0700, Paul Menage wrote:
> > > + for_each_possible_cpu(i) {
> > > + unsigned long flags;
> > > + spin_lock_irqsave(&tg->cfs_rq[i]->rq->lock, flags);
> >
> > Is the lock absolutely required here?
>
> I'm not sure, I was hoping you or Ingo could comment on this. But some
> kind of locking seems to required at least on 32-bit platforms, since
> sum_exec_runtime is a 64-bit number.

I tend to agree abt 32-bit platforms requiring a lock to read the 64-bit
sum_exec_runtime field.

Ingo/Dmitry, what do you think? fs/proc/array.c:task_utime() is also
buggy in that case.

--
Regards,
vatsa

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Srivatsa Vaddagiri on Tue, 23 Oct 2007 16:32:54 GMT
View Forum Message <> Reply to Message

On Mon, Oct 22, 2007 at 05:49:39PM -0700, Paul Menage wrote:
> + for_each_possible_cpu(i) {
> + unsigned long flags;
> + spin_lock_irqsave(&tg->cfs_rq[i]->rq->lock, flags);

A simpler form of this would be just:

  spin_lock_irqsave(&cpu_rq(i)->lock, flags);

--
Regards,
vatsa

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

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Srivatsa Vaddagiri on Tue, 23 Oct 2007 16:36:38 GMT
View Forum Message <> Reply to Message

On Mon, Oct 22, 2007 at 05:49:39PM -0700, Paul Menage wrote:
> Report CPU usage in CFS Cgroup directories
>
> Adds a cpu.usage file to the CFS cgroup that reports CPU usage in
> milliseconds for that cgroup's tasks

It would be nice to split this into user and sys time at some point.
We have also received request to provide idle time for a
container/cgroup. I presume these will need to be provided as additional files
(in addition to cpu.usage that is)?

--
Regards,
vatsa

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

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage

Posted by Paul Menage on Tue, 23 Oct 2007 16:41:49 GMT
View Forum Message <> Reply to Message

On 10/23/07, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> > Adds a cpu.usage file to the CFS cgroup that reports CPU usage in
> > milliseconds for that cgroup's tasks
>
> It would be nice to split this into user and sys time at some point.

Sounds reasonable - but does CFS track this?

> We have also received request to provide idle time for a
> container/cgroup.

The semantics of "idle time" for a cgroup on a shared system seem a bit fuzzy. How would you define it?

Suppose you have two cgroups that would each want to use, say, 55% of a CPU - technically they should each be regarded as having 45% idle time, but if they run on a the same CPU the chances are that they will both always have some processes on their runqueue due to contention with the other group. So how would you measure the difference between this and a cgroup that really is trying to use 100%?

Paul

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Srivatsa Vaddagiri on Tue, 23 Oct 2007 17:30:15 GMT
View Forum Message <> Reply to Message

On Tue, Oct 23, 2007 at 09:41:49AM -0700, Paul Menage wrote:
> > > Adds a cpu.usage file to the CFS cgroup that reports CPU usage in
> > > milliseconds for that cgroup's tasks
> >
> > It would be nice to split this into user and sys time at some point.
>
> Sounds reasonable - but does CFS track this?

No, not for a group. We could extend account_user_time() and account_systime_time() in this regard.

> > We have also received request to provide idle time for a
> > container/cgroup.

>
> The semantics of "idle time" for a cgroup on a shared system seem a
> bit fuzzy. How would you define it?

I think the percentage of time when it didn't have any runnable task in
its runqueues.

> Suppose you have two cgroups that would each want to use, say, 55% of
> a CPU - technically they should each be regarded as having 45% idle
> time, but if they run on a the same CPU the chances are that they will
> both always have some processes on their runqueue due to contention
> with the other group. So how would you measure the difference between
> this and a cgroup that really is trying to use 100%?

Good point. I think we need to subtract out the time it was waiting on runqueue
when calculating idle time.

```
|------- . . . . . ---------zzzzzzzzzzzz.......-----------|
t0     t1      t2       t3   t4     t5         t6
```

---- -> Running time
.... -> Waiting time (to get on the cpu)
zzzz -> Sleeping time (when it didnt want to run because of
    lack of tasks)

So, in this case,

idle time = (t4 - t3) / [(t6 - t1) - (t2-t1) - (t5-t4)

?

This idle time will be a per-cpu stat for every cgroup and needs to be
consolidated across cpus into a single idle-stat number, just like how
top does it.

--
Regards,
vatsa

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

---

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Wed, 24 Oct 2007 02:28:22 GMT

On 10/23/07, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> > Suppose you have two cgroups that would each want to use, say, 55% of
> > a CPU - technically they should each be regarded as having 45% idle
> > time, but if they run on a the same CPU the chances are that they will
> > both always have some processes on their runqueue due to contention
> > with the other group. So how would you measure the difference between
> > this and a cgroup that really is trying to use 100%?
>
> Good point. I think we need to subtract out the time it was waiting on runqueue
> when calculating idle time.
>
> |------- . . . . . ---------zzzzzzzzzzzz.....----------|
> t0 t1 t2 t3 t4 t5 t6
>
>
> ---- -> Running time
> .... -> Waiting time (to get on the cpu)
> zzzz -> Sleeping time (when it didnt want to run because of
> lack of tasks)
>
> So, in this case,
>
> idle time = (t4 - t3) / [(t6 - t1) - (t2-t1) - (t5-t4)
>

Do you mean (t6 - t0) where you have (t6 - t1)?

> ?
>
> This idle time will be a per-cpu stat for every cgroup and needs to be
> consolidated across cpus into a single idle-stat number, just like how
> top does it.

This would be an idle fraction, not an idle time. (seconds divided by seconds)

It doesn't seem quite right to me that a cgroup's idle time metric be
affected by the activity of other cgroups on the machine, but it's
hard to come up with a way of measuring it that doesn't have this
behaviour - which is why, in the absence of hard CPU partitioning,
it's not clear to me how much use this would be.

What would people be planning to use it for?

Paul

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Srivatsa Vaddagiri on Wed, 24 Oct 2007 04:29:37 GMT
View Forum Message <> Reply to Message

On Tue, Oct 23, 2007 at 07:28:22PM -0700, Paul Menage wrote:
> > Good point. I think we need to subtract out the time it was waiting on runqueue
> > when calculating idle time.
> >
> > |------- . . . . . ---------zzzzzzzzzzzz.....----------|
> > t0 t1 t2 t3 t4 t5 t6
> >
> >
> > ---- -> Running time
> > .... -> Waiting time (to get on the cpu)
> > zzzz -> Sleeping time (when it didnt want to run because of
> > lack of tasks)
> >
> > So, in this case,
> >
> > idle time = (t4 - t3) / [(t6 - t1) - (t2-t1) - (t5-t4)
> >
>
> Do you mean (t6 - t0) where you have (t6 - t1)?

Ah ..yes.

> > ?
> >
> > This idle time will be a per-cpu stat for every cgroup and needs to be
> > consolidated across cpus into a single idle-stat number, just like how
> > top does it.
>
> This would be an idle fraction, not an idle time. (seconds divided by seconds)

agreed, we need to be reporting idle time in (milli)seconds, although
the requirement we had was to report it back in percentage. I guess the
percentage figure can be derived from the raw idle time number.

How about:

 idle time = t4-t3 (effectively sleep time)

in the above example?

> It doesn't seem quite right to me that a cgroup's idle time metric be

> affected by the activity of other cgroups on the machine,

I don't see how the idle time metric defined above (t4-t3) can be
affected by other cgroup activity, unless the execution pattern of one
cgroup is dependent on the others.

However the minute you tranlsate this idle time into a percentage wrt
wall-clock time, then yes a cgroup's idle percentage can be affected by
others. For idle percentage to be meaningfull, I would imagine that
user-space tools will need to calculate it after discarding a group's
wait time.

> but it's
> hard to come up with a way of measuring it that doesn't have this
> behaviour - which is why, in the absence of hard CPU partitioning,
> it's not clear to me how much use this would be.
>
> What would people be planning to use it for?

I think primarily for systems management tools to report back various
statistics about a OpenVZ/VServer-like container (just like top reports stats
for a OS currently). Let me check if there are other uses envisoned for
it.

--
Regards,
vatsa

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

---

Subject: Re: [PATCH 2/2] CFS CGroup: Report usage
Posted by Paul Menage on Fri, 26 Oct 2007 01:24:55 GMT

View Forum Message <> Reply to Message

On 10/23/07, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>
> agreed, we need to be reporting idle time in (milli)seconds, although
> the requirement we had was to report it back in percentage. I guess the
> percentage figure can be derived from the raw idle time number.
>
> How about:
>
> idle time = t4-t3 (effectively sleep time)
>
> in the above example?

>
> > It doesn't seem quite right to me that a cgroup's idle time metric be
> > affected by the activity of other cgroups on the machine,
>
> I don't see how the idle time metric defined above (t4-t3) can be
> affected by other cgroup activity, unless the execution pattern of one
> cgroup is dependent on the others.

If the other cgroups are busier, and t1-t2 is longer, then the cgroup will get to the point where it's ready to sleep later in wallclock time, and t4-t3 will be shorter in absolute terms. If there were no other cgroups running, then presumably the sleep time would actually be the sum of those three periods.

Even so, I guess you're right that t4-t3 is the most appropriate thing to report, as long as people realise that it's a bit of a fuzzy value.

> I think primarily for systems management tools to report back various
> statistics about a OpenVZ/VServer-like container (just like top reports stats
> for a OS currently). Let me check if there are other uses envisoned for
> it.

Sorry, I didn't mean "how will you report it to users?", I meant "what kinds of useful information will the users be able to get from it?"

Paul