Subject: [RFC] Default child of a cgroup
Posted by Srivatsa Vaddagiri on Thu, 31 Jan 2008 02:40:49 GMT
View Forum Message <> Reply to Message

Hi,
As we were implementing multiple-hierarchy support for CPU controller, we hit some oddities in its implementation, partly related to current cgroups implementation. Peter and I have been debating on the exact solution and I thought of bringing that discussion to lkml.

Consider the cgroup filesystem structure for managing cpu resource.

```
# mount -t cgroup -ocpu,cpuacct none /cgroup
# mkdir /cgroup/A
# mkdir /cgroup/B
# mkdir /cgroup/A/a1
```

will result in:

```
/cgroup
   |------<tasks>
   |------<cpuacct.usage>
    |------<cpu.shares>
   |
   |----[A]
   |     |----<tasks>
   |     |----<cpuacct.usage>
   |     |----<cpu.shares>
   |     |
   |     |---[a1]
   |           |----<tasks>
   |           |----<cpuacct.usage>
   |           |----<cpu.shares>
   |           |
   |
   |----[B]
   |     |----<tasks>
   |     |----<cpuacct.usage>
   |     |----<cpu.shares>
   |
```

Here are some questions that arise in this picture:

1. What is the relationship of the task-group in A/tasks with the task-group in A/a1/tasks? In otherwords do they form siblings of the same parent A?

2. Somewhat related to the above question, how much resource should the
   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
   in A/tasks)?

3. What should A/cpuacct.usage reflect? CPU usage of A/tasks? Or CPU usage
   of all siblings put together? It can reflect only one, in which case
   user has to manually derive the other component of the statistics.

It seems to me that tasks in A/tasks form what can be called the
"default" child group of A, in which case:

4. Modifications to A/cpu.shares should affect the parent or its default
   child group (A/tasks)?

To avoid these ambiguities, it may be good if cgroup create this
"default child group" automatically whenever a cgroup is created?
Something like below (not the absence of tasks file in some directories
now):

```
 /cgroup
    |
    |------<cpuacct.usage>
     |------<cpu.shares>
    |
      |---[def_child]
    |     |----<tasks>
    |     |----<cpuacct.usage>
    |     |----<cpu.shares>
    |     |
    |
    |----[A]
    |     |
    |     |----<cpuacct.usage>
    |     |----<cpu.shares>
    |     |
    |     |---[def_child]
    |     |     |----<tasks>
    |     |     |----<cpuacct.usage>
    |     |     |----<cpu.shares>
    |     |     |
    |     |
    |     |---[a1]
    |           |
    |           |----<cpuacct.usage>
    |           |----<cpu.shares>
    |           |
```

```
|        |---[def_child]
|        |      |---<tasks>
|        |      |---<cpuacct.usage>
|        |      |---<cpu.shares>
|        |      |
|
|----[B]
|    |
|    |----<cpuacct.usage>
|    |----<cpu.shares>
|    |
|    |---[def_child]
|    |    |----<tasks>
|    |    |----<cpuacct.usage>
|    |    |----<cpu.shares>
|    |    |
```

Note that user cannot create subdirectories under def_child with this scheme! I am also not sure what impact this will have on other resources like cpusets ..

Thoughts?

--
Regards,
vatsa

Subject: Re: [RFC] Default child of a cgroup
Posted by serue on Thu, 31 Jan 2008 17:44:03 GMT
View Forum Message <> Reply to Message

Quoting Srivatsa Vaddagiri (vatsa@linux.vnet.ibm.com):
> Hi,
> As we were implementing multiple-hierarchy support for CPU
> controller, we hit some oddities in its implementation, partly related
> to current cgroups implementation. Peter and I have been debating on the
> exact solution and I thought of bringing that discussion to lkml.
>
> Consider the cgroup filesystem structure for managing cpu resource.
>
> # mount -t cgroup -ocpu,cpuacct none /cgroup
> # mkdir /cgroup/A

> # mkdir /cgroup/B
> # mkdir /cgroup/A/a1
>
> will result in:
>
> /cgroup
> |------<tasks>
> |------<cpuacct.usage>
> |------<cpu.shares>
> |
> |----[A]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> | |---[a1]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> |
> |----[B]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> |
>
>
> Here are some questions that arise in this picture:
>
> 1. What is the relationship of the task-group in A/tasks with the
>   task-group in A/a1/tasks? In otherwords do they form siblings
>   of the same parent A?
>
> 2. Somewhat related to the above question, how much resource should the
>   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
>   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
>   in A/tasks)?
>
> 3. What should A/cpuacct.usage reflect? CPU usage of A/tasks? Or CPU usage
>   of all siblings put together? It can reflect only one, in which case
>   user has to manually derive the other component of the statistics.
>
> It seems to me that tasks in A/tasks form what can be called the
> "default" child group of A, in which case:
>
> 4. Modifications to A/cpu.shares should affect the parent or its default
>   child group (A/tasks)?

>
> To avoid these ambiguities, it may be good if cgroup create this
> "default child group" automatically whenever a cgroup is created?
> Something like below (not the absence of tasks file in some directories
> now):

I didn't think it was actually ambiguous. /A/cpu.shares will specify what all tasks under /A and its children (just /A/a1/tass in this example) get to share, while /A/a1/cpu.share specifies what tasks under /A/a1/tasks get. Tasks which are in /A/tasks get whatever is left over, that is /A/cpu.share - /A/a1/cpu.shares. /A/cpuacct.usage reflects all usage by tasks under /A and its children.

>
>
> /cgroup
> |
> |------<cpuacct.usage>
> |------<cpu.shares>
> |
> |---[def_child]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> |
> |----[A]
> | |
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> | |---[def_child]
> | | |----<tasks>
> | | |----<cpuacct.usage>
> | | |----<cpu.shares>
> | | |
> | |
> | |---[a1]
> | |
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> | |---[def_child]
> | | |---<tasks>
> | | |---<cpuacct.usage>
> | | |---<cpu.shares>
> | | |
> |

> |----[B]
> | |
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> | |---[def_child]
> | | |----<tasks>
> | | |----<cpuacct.usage>
> | | |----<cpu.shares>
> | | |
>
> Note that user cannot create subdirectories under def_child with this
> scheme! I am also not sure what impact this will have on other resources
> like cpusets ..
>
> Thoughts?
>
>
> --
> Regards,
> vatsa
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

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

---

Subject: Re: [RFC] Default child of a cgroup
Posted by Balbir Singh on Thu, 31 Jan 2008 18:09:12 GMT
View Forum Message <> Reply to Message

Srivatsa Vaddagiri wrote:
> Hi,
> As we were implementing multiple-hierarchy support for CPU
> controller, we hit some oddities in its implementation, partly related
> to current cgroups implementation. Peter and I have been debating on the
> exact solution and I thought of bringing that discussion to lkml.
>
> Consider the cgroup filesystem structure for managing cpu resource.
>
> # mount -t cgroup -ocpu,cpuacct none /cgroup
> # mkdir /cgroup/A
> # mkdir /cgroup/B
> # mkdir /cgroup/A/a1

>
> will result in:
>
> /cgroup
> |------<tasks>
> |------<cpuacct.usage>
> |------<cpu.shares>
> |
> |----[A]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> | |---[a1]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> |
> |----[B]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> |
>
>
> Here are some questions that arise in this picture:
>
> 1. What is the relationship of the task-group in A/tasks with the
>   task-group in A/a1/tasks? In otherwords do they form siblings
>   of the same parent A?
>

I consider them to be the same relationship between directories and files.
A/tasks are siblings of A/a1 and A/other children, *but* the entities of
interest are A and A/a1.

> 2. Somewhat related to the above question, how much resource should the
>   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
>   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
>   in A/tasks)?
>

I propose that it gets 1/2 of the bandwidth, here is why

1. Assume that a task in A/tasks forks 1000 children, what happens to the
bandwidth of A/a1's tasks then? We have no control over how many tasks can be
created on A/tasks as a consequence of moving one task to A/tasks. Doing it the

other way would mean, that A/a1/tasks will get 1/1001 of the bandwidth (sounds very unfair and prone to Denial of Service/Fairness)

> 3. What should A/cpuacct.usage reflect? CPU usage of A/tasks? Or CPU usage
> of all siblings put together? It can reflect only one, in which case
> user has to manually derive the other component of the statistics.
>

It should reflect the accumulated usage of A's children and the tasks in A.

> It seems to me that tasks in A/tasks form what can be called the
> "default" child group of A, in which case:
>
> 4. Modifications to A/cpu.shares should affect the parent or its default
> child group (A/tasks)?
>
> To avoid these ambiguities, it may be good if cgroup create this
> "default child group" automatically whenever a cgroup is created?
> Something like below (not the absence of tasks file in some directories
> now):
>

I think the concept makes sense, but creating a default child is going to be confusing, as it is not really a child of A.

>
> /cgroup
> |
> |------<cpuacct.usage>
> |------<cpu.shares>
> |
> |---[def_child]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> |
> |----[A]
> | |
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> | |---[def_child]
> | | |----<tasks>
> | | |----<cpuacct.usage>
> | | |----<cpu.shares>
> | | |

```
>    |    |
>    |    |---[a1]
>    |         |
>    |         |----<cpuacct.usage>
>    |         |----<cpu.shares>
>    |         |
>    |        |---[def_child]
>    |        |      |---<tasks>
>    |        |      |---<cpuacct.usage>
>    |        |      |---<cpu.shares>
>    |        |      |
>    |
>    |----[B]
>    |    |
>    |    |----<cpuacct.usage>
>    |    |----<cpu.shares>
>    |    |
>    |    |---[def_child]
>    |    |    |----<tasks>
>    |    |    |----<cpuacct.usage>
>    |    |    |----<cpu.shares>
>    |    |    |
>
> Note that user cannot create subdirectories under def_child with this
> scheme! I am also not sure what impact this will have on other resources
> like cpusets ..
>
```

Which means we'll need special logic in the cgroup filesystem to handle
def_child. Not a very good idea.

> Thoughts?
>
>

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

Subject: Re: [RFC] Default child of a cgroup
Posted by Peter Zijlstra on Thu, 31 Jan 2008 20:37:42 GMT
View Forum Message <> Reply to Message

On Thu, 2008-01-31 at 23:39 +0530, Balbir Singh wrote:
> Srivatsa Vaddagiri wrote:
> > Hi,
> > As we were implementing multiple-hierarchy support for CPU
> > controller, we hit some oddities in its implementation, partly related
> > to current cgroups implementation. Peter and I have been debating on the
> > exact solution and I thought of bringing that discussion to lkml.
> >
> > Consider the cgroup filesystem structure for managing cpu resource.
> >
> > # mount -t cgroup -ocpu,cpuacct none /cgroup
> > # mkdir /cgroup/A
> > # mkdir /cgroup/B
> > # mkdir /cgroup/A/a1
> >
> > will result in:
> >
> > /cgroup
> > |------<tasks>
> > |------<cpuacct.usage>
> > |------<cpu.shares>
> > |
> > |----[A]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[a1]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > |
> > |----[B]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > |
> >
> >
> > Here are some questions that arise in this picture:
> >
> > 1. What is the relationship of the task-group in A/tasks with the
> > task-group in A/a1/tasks? In otherwords do they form siblings
> > of the same parent A?

> >
>
> I consider them to be the same relationship between directories and files.
> A/tasks are siblings of A/a1 and A/other children, *but* the entities of
> interest are A and A/a1.
>
> > 2. Somewhat related to the above question, how much resource should the
> > task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
> > A's share or 1/(1 + N) of parent A's share (where N = number of tasks
> > in A/tasks)?
> >
>
> I propose that it gets 1/2 of the bandwidth, here is why
>
> 1. Assume that a task in A/tasks forks 1000 children, what happens to the
> bandwidth of A/a1's tasks then? We have no control over how many tasks can be
> created on A/tasks as a consequence of moving one task to A/tasks. Doing it the
> other way would mean, that A/a1/tasks will get 1/1001 of the bandwidth (sounds
> very unfair and prone to Denial of Service/Fairness)

And I oppose this, it means not all siblings are treated equal. Also, I
miss the story of the 'hidden' group here. The biggest objection is this
hidden group with no direct controls.

My proposal is to make it a hard constraint, either a group has task
children or a group has group children, but not mixed. That keeps the
interface explicit and doesn't hide the tricks we play.

> > 3. What should A/cpuacct.usage reflect? CPU usage of A/tasks? Or CPU usage
> > of all siblings put together? It can reflect only one, in which case
> > user has to manually derive the other component of the statistics.
> >
>
> It should reflect the accumulated usage of A's children and the tasks in A.

A's children includes tasks in this context. See where the confusion is?

> > It seems to me that tasks in A/tasks form what can be called the
> > "default" child group of A, in which case:
> >
> > 4. Modifications to A/cpu.shares should affect the parent or its default
> > child group (A/tasks)?
> >
> > To avoid these ambiguities, it may be good if cgroup create this
> > "default child group" automatically whenever a cgroup is created?
> > Something like below (not the absence of tasks file in some directories
> > now):
> >

>
> I think the concept makes sense, but creating a default child is going to be
> confusing, as it is not really a child of A.

Quite so. I really hate this hidden group.

> >
> > /cgroup
> > |
> > |------<cpuacct.usage>
> > |------<cpu.shares>
> > |
> > |---[def_child]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > |
> > |----[A]
> > | |
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |----<tasks>
> > | | |----<cpuacct.usage>
> > | | |----<cpu.shares>
> > | | |
> > | |
> > | |---[a1]
> > | |
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |---<tasks>
> > | | |---<cpuacct.usage>
> > | | |---<cpu.shares>
> > | | |
> > |
> > |----[B]
> > | |
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |----<tasks>
> > | | |----<cpuacct.usage>

> > | | |----<cpu.shares>
> > | | |
> >
> > Note that user cannot create subdirectories under def_child with this
> > scheme! I am also not sure what impact this will have on other resources
> > like cpusets ..
> >
>
> Which means we'll need special logic in the cgroup filesystem to handle
> def_child. Not a very good idea.

agreed.

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

---

Subject: Re: [RFC] Default child of a cgroup
Posted by Vivek Goyal on Thu, 31 Jan 2008 21:13:29 GMT
View Forum Message <> Reply to Message

On Thu, Jan 31, 2008 at 08:10:49AM +0530, Srivatsa Vaddagiri wrote:
> Hi,
> As we were implementing multiple-hierarchy support for CPU
> controller, we hit some oddities in its implementation, partly related
> to current cgroups implementation. Peter and I have been debating on the
> exact solution and I thought of bringing that discussion to lkml.
>
> Consider the cgroup filesystem structure for managing cpu resource.
>
> # mount -t cgroup -ocpu,cpuacct none /cgroup
> # mkdir /cgroup/A
> # mkdir /cgroup/B
> # mkdir /cgroup/A/a1
>
> will result in:
>
> /cgroup
> |------<tasks>
> |------<cpuacct.usage>
> |------<cpu.shares>
> |
> |----[A]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>

> | |
> | |---[a1]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> | |
> |
> |----[B]
> | |----<tasks>
> | |----<cpuacct.usage>
> | |----<cpu.shares>
> |
>
>
> Here are some questions that arise in this picture:
>
> 1. What is the relationship of the task-group in A/tasks with the
> task-group in A/a1/tasks? In otherwords do they form siblings
> of the same parent A?
>

Vatsa,

I don't know much about cgroups but got a query. How do we handle this if we just go one level up? How do we define relationship between /cgroup/tasks and /cgroup/A/tasks, or /cgroup/tasks and /cgroup/B/tasks?

To me lower levels should be handeled in the same way.

Thanks
Vivek

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

---

Subject: Re: [RFC] Default child of a cgroup
Posted by Paul Menage on Fri, 01 Feb 2008 02:39:56 GMT
View Forum Message <> Reply to Message

On Jan 30, 2008 6:40 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
>
> Here are some questions that arise in this picture:
>
> 1. What is the relationship of the task-group in A/tasks with the
> task-group in A/a1/tasks? In otherwords do they form siblings
> of the same parent A?

I'd argue the same as Balbir - tasks in A/tasks are are children of A
and are siblings of a1, a2, etc.

>
> 2. Somewhat related to the above question, how much resource should the
>   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
>   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
>   in A/tasks)?

Each process in A should have a scheduler weight that's derived from
its static_prio field. Similarly each subgroup of A will have a
scheduler weight that's determined by its cpu.shares value. So the cpu
share of any child (be it a task or a subgroup) would be equal to its
own weight divided by the sum of weights of all children.

So yes, if a task in A forks lots of children, those children could
end up getting a disproportionate amount of the CPU compared to tasks
in A/a1 - but that's the same as the situation without cgroups. If you
want to control cpu usage between different sets of processes in A,
they should be in sibling cgroups, not directly in A.

Is there a restriction in CFS that stops a given group from
simultaneously holding tasks and sub-groups? If so, couldn't we change
CFS to make it possible rather than enforcing awkward restructions on
cgroups?

If we really can't change CFS in that way, then an alternative would
be similar to Peter's suggestion - make cpu_cgroup_can_attach() fail
if the cgroup has children, and make cpu_cgroup_create() fail if the
cgroup has any tasks - that way you limit the restriction to just the
hierarchy that has CFS attached to it, rather than generically for all
cgroups

BTW, I noticed this code in cpu_cgroup_create():

```
        /* we support only 1-level deep hierarchical scheduler atm */
        if (cgrp->parent->parent)
                return ERR_PTR(-EINVAL);
```

Is anyone working on allowing more levels?

Paul

Subject: Re: [RFC] Default child of a cgroup
Posted by Balbir Singh on Fri, 01 Feb 2008 03:32:33 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:

[snip]
> BTW, I noticed this code in cpu_cgroup_create():
>
> /* we support only 1-level deep hierarchical scheduler atm */
> if (cgrp->parent->parent)
> return ERR_PTR(-EINVAL);
>
> Is anyone working on allowing more levels?
>

Yes, Dhaval nad Vatsa are looking at removing that limitation.
This discussion is a side-effect of the multi-hierarchy discussion/implementation.

> Paul

--
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: [RFC] Default child of a cgroup
Posted by Dhaval Giani on Fri, 01 Feb 2008 03:40:26 GMT
View Forum Message <> Reply to Message

On Thu, Jan 31, 2008 at 06:39:56PM -0800, Paul Menage wrote:
> On Jan 30, 2008 6:40 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> >
> > Here are some questions that arise in this picture:
> >
> > 1. What is the relationship of the task-group in A/tasks with the
> > task-group in A/a1/tasks? In otherwords do they form siblings
> > of the same parent A?
>
> I'd argue the same as Balbir - tasks in A/tasks are are children of A
> and are siblings of a1, a2, etc.

>
> >
> > 2. Somewhat related to the above question, how much resource should the
> > task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
> > A's share or 1/(1 + N) of parent A's share (where N = number of tasks
> > in A/tasks)?
>
> Each process in A should have a scheduler weight that's derived from
> its static_prio field. Similarly each subgroup of A will have a
> scheduler weight that's determined by its cpu.shares value. So the cpu
> share of any child (be it a task or a subgroup) would be equal to its
> own weight divided by the sum of weights of all children.
>
> So yes, if a task in A forks lots of children, those children could
> end up getting a disproportionate amount of the CPU compared to tasks
> in A/a1 - but that's the same as the situation without cgroups. If you
> want to control cpu usage between different sets of processes in A,
> they should be in sibling cgroups, not directly in A.
>
> Is there a restriction in CFS that stops a given group from
> simultaneously holding tasks and sub-groups? If so, couldn't we change
> CFS to make it possible rather than enforcing awkward restructions on
> cgroups?
>
> If we really can't change CFS in that way, then an alternative would
> be similar to Peter's suggestion - make cpu_cgroup_can_attach() fail
> if the cgroup has children, and make cpu_cgroup_create() fail if the
> cgroup has any tasks - that way you limit the restriction to just the
> hierarchy that has CFS attached to it, rather than generically for all
> cgroups
>
> BTW, I noticed this code in cpu_cgroup_create():
>
>   /* we support only 1-level deep hierarchical scheduler atm */
>   if (cgrp->parent->parent)
>   return ERR_PTR(-EINVAL);
>
> Is anyone working on allowing more levels?
>

Yes, I am looking at it.

> Paul

--
regards,
Dhaval

Subject: Re: [RFC] Default child of a cgroup
Posted by Dhaval Giani on Fri, 01 Feb 2008 03:53:25 GMT
View Forum Message <> Reply to Message

On Thu, Jan 31, 2008 at 11:39:12PM +0530, Balbir Singh wrote:
> Srivatsa Vaddagiri wrote:
> > Hi,
> > As we were implementing multiple-hierarchy support for CPU
> > controller, we hit some oddities in its implementation, partly related
> > to current cgroups implementation. Peter and I have been debating on the
> > exact solution and I thought of bringing that discussion to lkml.
> >
> > Consider the cgroup filesystem structure for managing cpu resource.
> >
> > # mount -t cgroup -ocpu,cpuacct none /cgroup
> > # mkdir /cgroup/A
> > # mkdir /cgroup/B
> > # mkdir /cgroup/A/a1
> >
> > will result in:
> >
> > /cgroup
> > |------<tasks>
> > |------<cpuacct.usage>
> > |------<cpu.shares>
> > |
> > |----[A]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[a1]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > |
> > |----[B]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > |
> >

> >
> > Here are some questions that arise in this picture:
> >
> > 1. What is the relationship of the task-group in A/tasks with the
> >   task-group in A/a1/tasks? In otherwords do they form siblings
> >   of the same parent A?
> >
>
> I consider them to be the same relationship between directories and files.
> A/tasks are siblings of A/a1 and A/other children, *but* the entities of
> interest are A and A/a1.
>
> > 2. Somewhat related to the above question, how much resource should the
> >   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
> >   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
> >   in A/tasks)?
> >
>
> I propose that it gets 1/2 of the bandwidth, here is why
>
> 1. Assume that a task in A/tasks forks 1000 children, what happens to the
> bandwidth of A/a1's tasks then? We have no control over how many tasks can be
> created on A/tasks as a consequence of moving one task to A/tasks. Doing it the
> other way would mean, that A/a1/tasks will get 1/1001 of the bandwidth (sounds
> very unfair and prone to Denial of Service/Fairness)
>
>
> > 3. What should A/cpuacct.usage reflect? CPU usage of A/tasks? Or CPU usage
> >   of all siblings put together? It can reflect only one, in which case
> >   user has to manually derive the other component of the statistics.
> >
>
> It should reflect the accumulated usage of A's children and the tasks in A.
>

I've been taking the root group as an example, and extending it. The
root group does not reflect the usage of all the tasks in it. (IIRC,
can't seem to find the stats file there now, please correct me if I am
wrong)

> > It seems to me that tasks in A/tasks form what can be called the
> > "default" child group of A, in which case:
> >
> > 4. Modifications to A/cpu.shares should affect the parent or its default
> >   child group (A/tasks)?
> >
> > To avoid these ambiguities, it may be good if cgroup create this
> > "default child group" automatically whenever a cgroup is created?

> > Something like below (not the absence of tasks file in some directories
> > now):
> >
>
> I think the concept makes sense, but creating a default child is going to be
> confusing, as it is not really a child of A.
>

For all practical purposes, it is the same as the init_task_group which
is at the parent level.

> >
> > /cgroup
> > |
> > |------<cpuacct.usage>
> > |------<cpu.shares>
> > |
> > |---[def_child]
> > | |----<tasks>
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > |
> > |----[A]
> > | |
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |----<tasks>
> > | | |----<cpuacct.usage>
> > | | |----<cpu.shares>
> > | | |
> > | |
> > | |---[a1]
> > | |
> > | |----<cpuacct.usage>
> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |---<tasks>
> > | | |---<cpuacct.usage>
> > | | |---<cpu.shares>
> > | | |
> > |
> > |----[B]
> > | |
> > | |----<cpuacct.usage>

> > | |----<cpu.shares>
> > | |
> > | |---[def_child]
> > | | |----<tasks>
> > | | |----<cpuacct.usage>
> > | | |----<cpu.shares>
> > | | |
> >
> > Note that user cannot create subdirectories under def_child with this
> > scheme! I am also not sure what impact this will have on other resources
> > like cpusets ..
> >
>
> Which means we'll need special logic in the cgroup filesystem to handle
> def_child. Not a very good idea.
>

Not really. That issue would come into play if every task group was
assigned a control group. The task group is not exposed to the outside
world. (That's why its a hidden task group)

Thanks,
--
regards,
Dhaval

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

---

Subject: Re: [RFC] Default child of a cgroup
Posted by Dhaval Giani on Fri, 01 Feb 2008 04:16:55 GMT
View Forum Message <> Reply to Message

On Thu, Jan 31, 2008 at 09:37:42PM +0100, Peter Zijlstra wrote:
>
> On Thu, 2008-01-31 at 23:39 +0530, Balbir Singh wrote:
> > Srivatsa Vaddagiri wrote:
> > > Hi,
> > > As we were implementing multiple-hierarchy support for CPU
> > > controller, we hit some oddities in its implementation, partly related
> > > to current cgroups implementation. Peter and I have been debating on the
> > > exact solution and I thought of bringing that discussion to lkml.
> > >
> > > Consider the cgroup filesystem structure for managing cpu resource.
> > >
> > > # mount -t cgroup -ocpu,cpuacct none /cgroup

> > > # mkdir /cgroup/A
> > > # mkdir /cgroup/B
> > > # mkdir /cgroup/A/a1
> > >
> > > will result in:
> > >
> > > /cgroup
> > > |------<tasks>
> > > |------<cpuacct.usage>
> > > |------<cpu.shares>
> > > |
> > > |----[A]
> > > | |----<tasks>
> > > | |----<cpuacct.usage>
> > > | |----<cpu.shares>
> > > | |
> > > | |---[a1]
> > > | |----<tasks>
> > > | |----<cpuacct.usage>
> > > | |----<cpu.shares>
> > > | |
> > > |
> > > |----[B]
> > > | |----<tasks>
> > > | |----<cpuacct.usage>
> > > | |----<cpu.shares>
> > > |
> > >
> > >
> > > Here are some questions that arise in this picture:
> > >
> > > 1. What is the relationship of the task-group in A/tasks with the
> > >   task-group in A/a1/tasks? In otherwords do they form siblings
> > >   of the same parent A?
> > >
> >
> > I consider them to be the same relationship between directories and files.
> > A/tasks are siblings of A/a1 and A/other children, *but* the entities of
> > interest are A and A/a1.
> >
> > > 2. Somewhat related to the above question, how much resource should the
> > >   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
> > >   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
> > >   in A/tasks)?
> > >
> >
> > I propose that it gets 1/2 of the bandwidth, here is why
> >

> > 1. Assume that a task in A/tasks forks 1000 children, what happens to the
> > bandwidth of A/a1's tasks then? We have no control over how many tasks can be
> > created on A/tasks as a consequence of moving one task to A/tasks. Doing it the
> > other way would mean, that A/a1/tasks will get 1/1001 of the bandwidth (sounds
> > very unfair and prone to Denial of Service/Fairness)
>
> And I oppose this, it means not all siblings are treated equal. Also, I
> miss the story of the 'hidden' group here. The biggest objection is this
> hidden group with no direct controls.
>
> My proposal is to make it a hard constraint, either a group has task
> children or a group has group children, but not mixed. That keeps the
> interface explicit and doesn't hide the tricks we play.
>

That is one solution. Otherwise you provide the controls for the hidden
group. (Namely the shares and the rt_ratio). I've been experimenting
with this approach recently.

<snip>

> > > Note that user cannot create subdirectories under def_child with this
> > > scheme! I am also not sure what impact this will have on other resources
> > > like cpusets ..
> > >

I'm not sure why it would affect other resources? The def_child is not
exposed to the cgroup filesystem. Could someone please explain it to me?

> >
> > Which means we'll need special logic in the cgroup filesystem to handle
> > def_child. Not a very good idea.
>
> agreed.

--
regards,
Dhaval

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

---

Subject: Re: [RFC] Default child of a cgroup
Posted by Peter Zijlstra on Fri, 01 Feb 2008 07:58:32 GMT
View Forum Message <> Reply to Message

On Thu, 2008-01-31 at 18:39 -0800, Paul Menage wrote:
> On Jan 30, 2008 6:40 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> >
> > Here are some questions that arise in this picture:
> >
> > 1. What is the relationship of the task-group in A/tasks with the
> >   task-group in A/a1/tasks? In otherwords do they form siblings
> >   of the same parent A?
>
> I'd argue the same as Balbir - tasks in A/tasks are are children of A
> and are siblings of a1, a2, etc.
>
> >
> > 2. Somewhat related to the above question, how much resource should the
> >   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
> >   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
> >   in A/tasks)?
>
> Each process in A should have a scheduler weight that's derived from
> its static_prio field. Similarly each subgroup of A will have a
> scheduler weight that's determined by its cpu.shares value. So the cpu
> share of any child (be it a task or a subgroup) would be equal to its
> own weight divided by the sum of weights of all children.
>
> So yes, if a task in A forks lots of children, those children could
> end up getting a disproportionate amount of the CPU compared to tasks
> in A/a1 - but that's the same as the situation without cgroups. If you
> want to control cpu usage between different sets of processes in A,
> they should be in sibling cgroups, not directly in A.
>
> Is there a restriction in CFS that stops a given group from
> simultaneously holding tasks and sub-groups? If so, couldn't we change
> CFS to make it possible rather than enforcing awkward restructions on
> cgroups?

I think it is possible, just way more work than the proposed hack.

> If we really can't change CFS in that way, then an alternative would
> be similar to Peter's suggestion - make cpu_cgroup_can_attach() fail
> if the cgroup has children, and make cpu_cgroup_create() fail if the
> cgroup has any tasks - that way you limit the restriction to just the
> hierarchy that has CFS attached to it, rather than generically for all
> cgroups

Agreed.

_____
Containers mailing list

Subject: Re: [RFC] Default child of a cgroup
Posted by Srivatsa Vaddagiri on Fri, 01 Feb 2008 08:17:18 GMT
View Forum Message <> Reply to Message

On Thu, Jan 31, 2008 at 06:39:56PM -0800, Paul Menage wrote:
> On Jan 30, 2008 6:40 PM, Srivatsa Vaddagiri <vatsa@linux.vnet.ibm.com> wrote:
> >
> > Here are some questions that arise in this picture:
> >
> > 1. What is the relationship of the task-group in A/tasks with the
> >   task-group in A/a1/tasks? In otherwords do they form siblings
> >   of the same parent A?
>
> I'd argue the same as Balbir - tasks in A/tasks are are children of A
> and are siblings of a1, a2, etc.

> > 2. Somewhat related to the above question, how much resource should the
> >   task-group A/a1/tasks get in relation to A/tasks? Is it 1/2 of parent
> >   A's share or 1/(1 + N) of parent A's share (where N = number of tasks
> >   in A/tasks)?
>
> Each process in A should have a scheduler weight that's derived from
> its static_prio field. Similarly each subgroup of A will have a
> scheduler weight that's determined by its cpu.shares value. So the cpu
> share of any child (be it a task or a subgroup) would be equal to its
> own weight divided by the sum of weights of all children.

Assuming all tasks are of same prio, then what you are saying is that
A/a1/tasks should cumulatively recv 1/(1 + N) of parent's share.

After some thought, that seems like a reasonable expectation. The only issue
I have for that is it breaks current behavior in mainline. Assume this
structure:

```
/
|------<tasks>
|------<cpuacct.usage>
 |------<cpu.shares>
|
|----[A]
|     |----<tasks>
|     |----<cpuacct.usage>
|     |----<cpu.shares>
```

then, going by above argument, /A/tasks should recv 1/(1+M)% of system resources (M -> number of tasks in /tasks), whereas it receives 1/2 of system resources currently (assuming /cpu.shares and /A/cpu.shares are same).

Balbir, is this behaviour same for memory controller as well?

So pick any option, we are talking of deviating from current behavior, which perhaps is a non-issue if we want to DTRT.

> So yes, if a task in A forks lots of children, those children could
> end up getting a disproportionate amount of the CPU compared to tasks
> in A/a1 - but that's the same as the situation without cgroups. If you
> want to control cpu usage between different sets of processes in A,
> they should be in sibling cgroups, not directly in A.
>
> Is there a restriction in CFS that stops a given group from
> simultaneously holding tasks and sub-groups? If so, couldn't we change
> CFS to make it possible rather than enforcing awkward restructions on
> cgroups?

Should be possible, need to look closely at what will need to change (load_balance routines for sure).

--
Regards,
vatsa

Subject: Re: [RFC] Default child of a cgroup
Posted by Paul Menage on Fri, 01 Feb 2008 15:35:46 GMT
View Forum Message <> Reply to Message

On Jan 31, 2008 11:58 PM, Peter Zijlstra <a.p.zijlstra@chello.nl> wrote:
> > Is there a restriction in CFS that stops a given group from
> > simultaneously holding tasks and sub-groups? If so, couldn't we change
> > CFS to make it possible rather than enforcing awkward restrictions on
> > cgroups?
>
> I think it is possible, just way more work than the proposed hack.

Seems to me like the right thing to do though.

>
> > If we really can't change CFS in that way, then an alternative would
> > be similar to Peter's suggestion - make cpu_cgroup_can_attach() fail
> > if the cgroup has children, and make cpu_cgroup_create() fail if the
> > cgroup has any tasks - that way you limit the restriction to just the
> > hierarchy that has CFS attached to it, rather than generically for all
> > cgroups
>
> Agreed.
>

Actually, I realised later that this is impossible - since the root
cgroup will have tasks initially, there'd be no way to create the
first child cgroup in the CFS hierarchy.

Paul