
Subject: Re: [RFC][PATCH 0/1]a new optional function for task assignment to cgroup

Posted by [Paul Menage](#) on Wed, 05 Mar 2008 05:56:13 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi Kazunaga,

On Tue, Mar 4, 2008 at 9:39 PM, Kazunaga Ikeno <k-ikeno@ak.jp.nec.com> wrote:

> Hi -

>

> This is a patch of a new optional function for task assignment to cgroup, RFC.

>

>

> == Purpose =====

>

> To provide the function that leads a task, corresponding to the conditions specified
> beforehand, to a specific cgroup directory.

>

This is something that's been discussed before, originally as part of CKRM with a complex rule engine in the kernel space.

Basically, the general agreement was that it's a case where a simple API is going to be too simple for the majority of users, and a complex API that satisfies everyone is going to be too messy/heavyweight.

This is something that can be done in a userspace daemon via the process events connector - when you get a PROC_EVENT_UID event, you can move the process into the appropriate cgroup (you may also need to check any recently-forked children). This also gives you more flexibility than you can have in the kernel - you can base your decision on more complex factors than simply the uid of the process.

Dhaval Giani had a prototype implementation of such a daemon.

Paul

>

> == Description =====

>

> This patch provides the function that leads a task, corresponding to the conditions
> specified beforehand, to a specific cgroup directory.

>

> Currently, this patch uses user-id as a condition to lead a task. On its I/F,
> specifies user-id of a task and a cgroup directory.

>

> The task set to specified user-id will automatically lead to the cgroup directory.
> (it is attached to specific cgroup)

```

>
> This function makes possible to attach a task to cgroup automatically when
> specific user logs in, also to attach a task of a service which is set to
> specific effective user-id to specific cgroup mechanically.
>
> This function is just option, all the functions of cgroup are the same.
> Also the migration of a task between cgroup directories can do by rewriting pid
> of a control tasks file, including a task leading by this option.
>
> It is able to enter two or more set of user-id and cgroup directory.
> Specified cgroup directory may be the same or that may not be.
> But it's not able to enter same user-id to plural cgroup directories to lead.
>
>
> == Interface =====
>
> /lead_option - control file of this option
>
> [example for reading a configuration]
>
> # cat /cgroup/lead_option
>
> uid:202    leadto:/cpuset/bar_cg
> uid:201    leadto:/cpuset/foo_cg
>
> * nothing appears before assignment.
>
> [example for adding an entry]
> - To lead a task(uid 201) to /cgroup/foo_cg
>
> # echo uid:201 leadto:/cpuset/foo_cg > /cpuset/lead_option
>
> * set a uid of task and cgroup dirctory to lead.
> * Remake an entry uid to cgroup directory if set uid already exists.
>
> [example for delete an entry]
> - To delete an entry of uid
>
> # echo uid:201 > /cpuset/lead_option
>
> * To delete a registration, omit "leadto:" token.
>
>
> == Operation example (chronological order) =====
>
> The follows is an example of the operation.
>
> # #####

```

```

> # # Various confirmation before testing
> # #####
> # id
> uid=0(root) gid=0(root) groups=0(root)
> # df /cpuset
> Filesystem          1K-blocks    Used Available Use% Mounted on
> none                 0          0          0 - /cpuset
> # more /proc/self/cgroup
> cpuset:/
> # id foo
> uid=201(foo) gid=100(users) groups=100(users)
> # id bar
> uid=202(foo) gid=100(users) groups=100(users)
> # #####
> # # Add an entry of user foo,bar
> # #####
> # echo uid:201 leadto:/cpuset/foo_cg > /cpuset/lead_option
> # echo uid:202 leadto:/cpuset/bar_cg > /cpuset/lead_option
> # more /cpuset/lead_option
> uid:202    leadto:/cpuset/bar_cg
> uid:201    leadto:/cpuset/foo_cg
> # #####
> # # Confirmation of the assignment of user foo,bar - (1)
> # #####
> # su - foo
> $ more /proc/$$/cgroup
> cpuset:/foo_cg
> $ more /proc/self/cgroup
> cpuset:/foo_cg
> $ su bar --command "more /proc/self/cgroup"
> cpuset:/bar_cg
> $ exit
> # #####
> # # Delete an entry of user foo,bar
> # #####
> # echo uid:201 > /cpuset/lead_option
> # echo uid:202 > /cpuset/lead_option
> # more /cpuset/lead_option
> # #####
> # # Confirmation of the assignment of user foo,bar - (2)
> # #####
> # su - foo
> $ more /proc/$$/cgroup
> cpuset:/
> $ su bar --command "more /proc/self/cgroup"
> cpuset:/
> $
>

```

>
> Thanks,
> - Kazunaga Ikeno.
>
>
>
> --
> To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at <http://vger.kernel.org/majordomo-info.html>
> Please read the FAQ at <http://www.tux.org/lkml/>
>

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

Subject: RE: [RFC][PATCH 0/1]a new optional function for task assignment to cgroup
Posted by [Kazunaga Ikeno](#) on Wed, 05 Mar 2008 07:02:17 GMT
[View Forum Message](#) <> [Reply to Message](#)

Paul Menage wrote:

> This is something that's been discussed before, originally as part of
> CKRM with a complex rule engine in the kernel space.
>
> Basically, the general agreement was that it's a case where a simple
> API is going to be too simple for the majority of users, and a complex
> API that satisfies everyone is going to be too messy/heavyweight.
>
> This is something that can be done in a userspace daemon via the
> process events connector - when you get a PROC_EVENT_UID event, you
> can move the process into the appropriate cgroup (you may also need to
> check any recently-forked children). This also gives you more
> flexibility than you can have in the kernel - you can base your
> decision on more complex factors than simply the uid of the process.
>
> Dhaval Giani had a prototype implementation of such a daemon.

Paul -

Thank you for your comment.
Because it was the almost same timing, I did not notice about Dhaval Giani's plan.
I will investigate it.

- Kazunaga Ikeno.

```

>
> Paul
>
>>
>> == Description =====
>>
>> This patch provides the function that leads a task, corresponding to the conditions
>> specified beforehand, to a specific cgroup directory.
>>
>> Currently, this patch uses user-id as a condition to lead a task. On its I/F,
>> specifies user-id of a task and a cgroup directory.
>>
>> The task set to specified user-id will automatically lead to the cgroup directory.
>> (it is attached to specific cgroup)
>>
>> This function makes possible to attach a task to cgroup automatically when
>> specific user logs in, also to attach a task of a service which is set to
>> specific effective user-id to specific cgroup mechanically.
>>
>> This function is just option, all the functions of cgroup are the same.
>> Also the migration of a task between cgroup directories can do by rewriting pid
>> of a control tasks file, including a task leading by this option.
>>
>> It is able to enter two or more set of user-id and cgroup directory.
>> Specified cgroup directory may be the same or that may not be.
>> But it's not able to enter same user-id to plural cgroup directories to lead.
>>
>>
>> == Interface =====
>>
>> /lead_option - control file of this option
>>
>> [example for reading a configuration]
>>
>> # cat /cgroup/lead_option
>>
>> uid:202    leadto:/cpuset/bar_cg
>> uid:201    leadto:/cpuset/foo_cg
>>
>> * nothing appears before assignment.
>>
>> [example for adding an entry]
>> - To lead a task(uid 201) to /cgroup/foo_cg
>>
>> # echo uid:201 leadto:/cpuset/foo_cg > /cpuset/lead_option
>>
>> * set a uid of task and cgroup directory to lead.
>> * Remake an entry uid to cgroup directory if set uid already exists.

```

```

> >
> > [example for delete an entry]
> > - To delete an entry of uid
> >
> > # echo uid:201 > /cpuset/lead_option
> >
> > * To delete a registration, omit "leadto:" token.
> >
> >
> > == Operation example (chronological order) =====
> >
> > The follows is an example of the operation.
> >
> > # #####
> > # # Various confirmation before testing
> > # #####
> > # id
> > uid=0(root) gid=0(root) groups=0(root)
> > # df /cpuset
> > Filesystem      1K-blocks    Used Available Use% Mounted on
> > none            0          0          0 - /cpuset
> > # more /proc/self/cgroup
> > cpuset:/
> > # id foo
> > uid=201(foo) gid=100(users) groups=100(users)
> > # id bar
> > uid=202(foo) gid=100(users) groups=100(users)
> > # #####
> > # # Add an entry of user foo,bar
> > # #####
> > # echo uid:201 leadto:/cpuset/foo_cg > /cpuset/lead_option
> > # echo uid:202 leadto:/cpuset/bar_cg > /cpuset/lead_option
> > # more /cpuset/lead_option
> > uid:202      leadto:/cpuset/bar_cg
> > uid:201      leadto:/cpuset/foo_cg
> > # #####
> > # # Confirmation of the assignment of user foo,bar - (1)
> > # #####
> > # su - foo
> > $ more /proc/$$/cgroup
> > cpuset:/foo_cg
> > $ more /proc/self/cgroup
> > cpuset:/foo_cg
> > $ su bar --command "more /proc/self/cgroup"
> > cpuset:/bar_cg
> > $ exit
> > # #####
> > # # Delete an entry of user foo,bar

```

```
> > # #####
> > # echo uid:201 > /cpuset/lead_option
> > # echo uid:202 > /cpuset/lead_option
> > # more /cpuset/lead_option
> > # #####
> > # # Confirmation of the assignment of user foo,bar - (2)
> > # #####
> > # su - foo
> > $ more /proc/$$/cgroup
> > cpuset:/
> > $ su bar --command "more /proc/self/cgroup"
> > cpuset:/
> > $
> >
> >
> > Thanks,
> > - Kazunaga Ikeno.
> >
> >
> >
> > --
> > To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
> > the body of a message to majordomo@vger.kernel.org
> > More majordomo info at http://vger.kernel.org/majordomo-info.html
> > Please read the FAQ at http://www.tux.org/lkml/
> >
```

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

Subject: Re: [RFC][PATCH 0/1]a new optional function for task assignment to cgroup
Posted by [Dhaval Giani](#) on Wed, 05 Mar 2008 07:13:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, Mar 04, 2008 at 09:56:13PM -0800, Paul Menage wrote:
> Hi Kazunaga,
>
> On Tue, Mar 4, 2008 at 9:39 PM, Kazunaga Ikeno <k-ikeno@ak.jp.nec.com> wrote:
>> Hi -
>>
>> This is a patch of a new optional function for task assignment to cgroup, RFC.
>>
>>
>> == Purpose =====

> >
> > To provide the function that leads a task, corresponding to the conditions specified
> > beforehand, to a specific cgroup directory.
> >
>
> This is something that's been discussed before, originally as part of
> CKRM with a complex rule engine in the kernel space.
>
> Basically, the general agreement was that it's a case where a simple
> API is going to be too simple for the majority of users, and a complex
> API that satisfies everyone is going to be too messy/heavyweight.
>
> This is something that can be done in a userspace daemon via the
> process events connector - when you get a PROC_EVENT_UID event, you
> can move the process into the appropriate cgroup (you may also need to
> check any recently-forked children). This also gives you more
> flexibility than you can have in the kernel - you can base your
> decision on more complex factors than simply the uid of the process.
>
> Dhaval Giani had a prototype implementation of such a daemon.
>

The daemon was posted at
<http://article.gmane.org/gmane.linux.kernel/553267> . At that point
control groups were called containers. These corrections will have to
made for it to run.

If I can get the time, I will clean it up and try to put it up
somewhere.

Thanks,

--
regards,
Dhaval

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