Subject: [PATCH] fix bad behavior in use_hierarchy file
Posted by Glauber Costa on Mon, 25 Jun 2012 09:21:01 GMT
View Forum Message <> Reply to Message

I have an application that does the following:

* copy the state of all controllers attached to a hierarchy
* replicate it as a child of the current level.

I would expect writes to the files to mostly succeed, since they
are inheriting sane values from parents.

But that is not the case for use_hierarchy. If it is set to 0, we
succeed ok. If we're set to 1, the value of the file is automatically
set to 1 in the children, but if userspace tries to write the
very same 1, it will fail. That same situation happens if we
set use_hierarchy, create a child, and then try to write 1 again.

Now, there is no reason whatsoever for failing to write a value
that is already there. It doesn't even match the comments, that
states:

 /* If parent's use_hierarchy is set, we can't make any modifications
  * in the child subtrees...

since we are not changing anything.

The following patch tests the new value against the one we're storing,
and automatically return 0 if we're not proposing a change.

Signed-off-by: Glauber Costa <glommer@parallels.com>
CC: Dhaval Giani <dhaval.giani@gmail.com>
CC: Michal Hocko <mhocko@suse.cz>
CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: Johannes Weiner <hannes@cmpxchg.org>
---
 mm/memcontrol.c | 6 ++++++
 1 file changed, 6 insertions(+)

diff --git a/mm/memcontrol.c b/mm/memcontrol.c
index ac35bcc..cccebbc 100644
--- a/mm/memcontrol.c
+++ b/mm/memcontrol.c
@@ -3779,6 +3779,10 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont, struct
cftype *cft,
   parent_memcg = mem_cgroup_from_cont(parent);

  cgroup_lock();

```
+
+ if (memcg->use_hierarchy == val)
+  goto out;
+
  /*
   * If parent's use_hierarchy is set, we can't make any modifications
   * in the child subtrees. If it is unset, then the change can
@@ -3795,6 +3799,8 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont, struct
cftype *cft,
    retval = -EBUSY;
  } else
   retval = -EINVAL;
+
+out:
  cgroup_unlock();

  return retval;
--
1.7.10.2
```

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by KAMEZAWA Hiroyuki on Mon, 25 Jun 2012 09:54:49 GMT
View Forum Message <> Reply to Message

(2012/06/25 18:21), Glauber Costa wrote:
> I have an application that does the following:
>
> * copy the state of all controllers attached to a hierarchy
> * replicate it as a child of the current level.
>
> I would expect writes to the files to mostly succeed, since they
> are inheriting sane values from parents.
>
> But that is not the case for use_hierarchy. If it is set to 0, we
> succeed ok. If we're set to 1, the value of the file is automatically
> set to 1 in the children, but if userspace tries to write the
> very same 1, it will fail. That same situation happens if we
> set use_hierarchy, create a child, and then try to write 1 again.
>
> Now, there is no reason whatsoever for failing to write a value
> that is already there. It doesn't even match the comments, that
> states:
>
> /* If parent's use_hierarchy is set, we can't make any modifications
>   * in the child subtrees...
>
> since we are not changing anything.

>
> The following patch tests the new value against the one we're storing,
> and automatically return 0 if we're not proposing a change.
>
> Signed-off-by: Glauber Costa <glommer@parallels.com>
> CC: Dhaval Giani <dhaval.giani@gmail.com>
> CC: Michal Hocko <mhocko@suse.cz>
> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> CC: Johannes Weiner <hannes@cmpxchg.org>

Hm.
Acked-by: KAMEZAWA Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Michal Hocko on Mon, 25 Jun 2012 12:08:23 GMT
View Forum Message <> Reply to Message

On Mon 25-06-12 13:21:01, Glauber Costa wrote:
> I have an application that does the following:
>
> * copy the state of all controllers attached to a hierarchy
> * replicate it as a child of the current level.
>
> I would expect writes to the files to mostly succeed, since they
> are inheriting sane values from parents.
>
> But that is not the case for use_hierarchy. If it is set to 0, we
> succeed ok. If we're set to 1, the value of the file is automatically
> set to 1 in the children, but if userspace tries to write the
> very same 1, it will fail. That same situation happens if we
> set use_hierarchy, create a child, and then try to write 1 again.
>
> Now, there is no reason whatsoever for failing to write a value
> that is already there. It doesn't even match the comments, that
> states:
>
> /* If parent's use_hierarchy is set, we can't make any modifications
>   * in the child subtrees...
>
> since we are not changing anything.
>
> The following patch tests the new value against the one we're storing,
> and automatically return 0 if we're not proposing a change.

Fair enough.

>

> Signed-off-by: Glauber Costa <glommer@parallels.com>
> CC: Dhaval Giani <dhaval.giani@gmail.com>
> CC: Michal Hocko <mhocko@suse.cz>
> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
> CC: Johannes Weiner <hannes@cmpxchg.org>

One comment bellow...
Acked-by: Michal Hocko <mhocko@suse.cz>

> ---
> mm/memcontrol.c | 6 ++++++
> 1 file changed, 6 insertions(+)
>
> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> index ac35bcc..cccebbc 100644
> --- a/mm/memcontrol.c
> +++ b/mm/memcontrol.c
> @@ -3779,6 +3779,10 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont, struct
cftype *cft,
>   parent_memcg = mem_cgroup_from_cont(parent);
>
>   cgroup_lock();
> +
> + if (memcg->use_hierarchy == val)
> + goto out;
> +

Why do you need cgroup_lock to check the value? Even if we have 2
CPUs racing (one trying to set to 0 other to 1 with use_hierarchy==0)
then the "set to 0" operation might fail depending on who hits the
cgroup_lock first anyway.

So while this is correct I think there is not much point to take the global
cgroup lock in this case.

>   /*
>   * If parent's use_hierarchy is set, we can't make any modifications
>   * in the child subtrees. If it is unset, then the change can
> @@ -3795,6 +3799,8 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont, struct
cftype *cft,
>   retval = -EBUSY;
>   } else
>   retval = -EINVAL;
> +
> +out:
>   cgroup_unlock();
>
>   return retval;

> --
> 1.7.10.2
>

--
Michal Hocko
SUSE Labs
SUSE LINUX s.r.o.
Lihovarska 1060/12
190 00 Praha 9
Czech Republic

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Glauber Costa on Mon, 25 Jun 2012 12:11:01 GMT
View Forum Message <> Reply to Message

On 06/25/2012 04:08 PM, Michal Hocko wrote:
> On Mon 25-06-12 13:21:01, Glauber Costa wrote:
>> I have an application that does the following:
>>
>> * copy the state of all controllers attached to a hierarchy
>> * replicate it as a child of the current level.
>>
>> I would expect writes to the files to mostly succeed, since they
>> are inheriting sane values from parents.
>>
>> But that is not the case for use_hierarchy. If it is set to 0, we
>> succeed ok. If we're set to 1, the value of the file is automatically
>> set to 1 in the children, but if userspace tries to write the
>> very same 1, it will fail. That same situation happens if we
>> set use_hierarchy, create a child, and then try to write 1 again.
>>
>> Now, there is no reason whatsoever for failing to write a value
>> that is already there. It doesn't even match the comments, that
>> states:
>>
>> /* If parent's use_hierarchy is set, we can't make any modifications
>> * in the child subtrees...
>>
>> since we are not changing anything.
>>
>> The following patch tests the new value against the one we're storing,
>> and automatically return 0 if we're not proposing a change.
>
> Fair enough.
>
>>

>> Signed-off-by: Glauber Costa <glommer@parallels.com>
>> CC: Dhaval Giani <dhaval.giani@gmail.com>
>> CC: Michal Hocko <mhocko@suse.cz>
>> CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
>> CC: Johannes Weiner <hannes@cmpxchg.org>
>
> One comment bellow...
> Acked-by: Michal Hocko <mhocko@suse.cz>
>
>> ---
>> mm/memcontrol.c | 6 ++++++
>> 1 file changed, 6 insertions(+)
>>
>> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
>> index ac35bcc..cccebbc 100644
>> --- a/mm/memcontrol.c
>> +++ b/mm/memcontrol.c
>> @@ -3779,6 +3779,10 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont, struct
cftype *cft,
>> parent_memcg = mem_cgroup_from_cont(parent);
>>
>> cgroup_lock();
>> +
>> + if (memcg->use_hierarchy == val)
>> + goto out;
>> +
>
> Why do you need cgroup_lock to check the value? Even if we have 2
> CPUs racing (one trying to set to 0 other to 1 with use_hierarchy==0)
> then the "set to 0" operation might fail depending on who hits the
> cgroup_lock first anyway.
>
> So while this is correct I think there is not much point to take the global
> cgroup lock in this case.
>
Well, no.

All operations will succeed, unless the cgroup breeds new children.
That's the operation we're racing against.

So we need to guarantee a snapshot of what is the status of the file in
the moment we said we'd create a new children.

Besides, I believe taking the lock is conceptually the right thing to
do, even if by an ordering artifact we would happen to be safe.

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Michal Hocko on Mon, 25 Jun 2012 12:49:05 GMT
View Forum Message <> Reply to Message

On Mon 25-06-12 16:11:01, Glauber Costa wrote:
> On 06/25/2012 04:08 PM, Michal Hocko wrote:
> >On Mon 25-06-12 13:21:01, Glauber Costa wrote:
[...]
> >>diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> >>index ac35bcc..cccebbc 100644
> >>--- a/mm/memcontrol.c
> >>+++ b/mm/memcontrol.c
> >>@@ -3779,6 +3779,10 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont,
struct cftype *cft,
> >>   parent_memcg = mem_cgroup_from_cont(parent);
> >>
> >>   cgroup_lock();
> >>+
> >>+ if (memcg->use_hierarchy == val)
> >>+ goto out;
> >>+
> >
> >Why do you need cgroup_lock to check the value? Even if we have 2
> >CPUs racing (one trying to set to 0 other to 1 with use_hierarchy==0)
> >then the "set to 0" operation might fail depending on who hits the
> >cgroup_lock first anyway.
> >
> >So while this is correct I think there is not much point to take the global
> >cgroup lock in this case.
> >
> Well, no.
>
> All operations will succeed, unless the cgroup breeds new children.
> That's the operation we're racing against.

I am not sure I understand. The changelog says that you want to handle
a situation where you are copying a hierarchy along with their
attributes and you don't want to fail when setting sane values.

If we race with a new child creation then the success always depends on
the lock ordering but once the value is set then it is final so the test
will work even outside of the lock. Or am I still missing something?

Just to make it clear the lock is necessary in the function I just do
not see why it should be held while we are trying to handle no-change
case.

>
> So we need to guarantee a snapshot of what is the status of the file

> in the moment we said we'd create a new children.
>
> Besides, I believe taking the lock is conceptually the right thing
> to do, even if by an ordering artifact we would happen to be safe.
>
> --
> To unsubscribe from this list: send the line "unsubscribe cgroups" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html

--
Michal Hocko
SUSE Labs
SUSE LINUX s.r.o.
Lihovarska 1060/12
190 00 Praha 9
Czech Republic

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Glauber Costa on Mon, 25 Jun 2012 12:55:31 GMT
View Forum Message <> Reply to Message

On 06/25/2012 04:49 PM, Michal Hocko wrote:
> On Mon 25-06-12 16:11:01, Glauber Costa wrote:
>> On 06/25/2012 04:08 PM, Michal Hocko wrote:
>>> On Mon 25-06-12 13:21:01, Glauber Costa wrote:
> [...]
>>>> diff --git a/mm/memcontrol.c b/mm/memcontrol.c
>>>> index ac35bcc..cccebbc 100644
>>>> --- a/mm/memcontrol.c
>>>> +++ b/mm/memcontrol.c
>>>> @@ -3779,6 +3779,10 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont,
struct cftype *cft,
>>>>   parent_memcg = mem_cgroup_from_cont(parent);
>>>>
>>>>   cgroup_lock();
>>>> +
>>>> + if (memcg->use_hierarchy == val)
>>>> + goto out;
>>>> +
>>>
>>> Why do you need cgroup_lock to check the value? Even if we have 2
>>> CPUs racing (one trying to set to 0 other to 1 with use_hierarchy==0)
>>> then the "set to 0" operation might fail depending on who hits the
>>> cgroup_lock first anyway.
>>>
>>> So while this is correct I think there is not much point to take the global

>>> cgroup lock in this case.
>>>
>> Well, no.
>>
>> All operations will succeed, unless the cgroup breeds new children.
>> That's the operation we're racing against.
>
> I am not sure I understand. The changelog says that you want to handle
> a situation where you are copying a hierarchy along with their
> attributes and you don't want to fail when setting sane values.
>
> If we race with a new child creation then the success always depends on
> the lock ordering but once the value is set then it is final so the test
> will work even outside of the lock. Or am I still missing something?
>
> Just to make it clear the lock is necessary in the function I just do
> not see why it should be held while we are trying to handle no-change
> case.
>

I think you are right in this specific case. But do you think it is
necessary to submit a version of it that tests outside the lock?

We don't gain too much with that anyway.

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Michal Hocko on Mon, 25 Jun 2012 13:22:05 GMT
View Forum Message <> Reply to Message

On Mon 25-06-12 16:55:31, Glauber Costa wrote:
> On 06/25/2012 04:49 PM, Michal Hocko wrote:
> >On Mon 25-06-12 16:11:01, Glauber Costa wrote:
> >>On 06/25/2012 04:08 PM, Michal Hocko wrote:
> >>>On Mon 25-06-12 13:21:01, Glauber Costa wrote:
> >[...]
> >>>>diff --git a/mm/memcontrol.c b/mm/memcontrol.c
> >>>>index ac35bcc..cccebbc 100644
> >>>>--- a/mm/memcontrol.c
> >>>>+++ b/mm/memcontrol.c
> >>>>@@ -3779,6 +3779,10 @@ static int mem_cgroup_hierarchy_write(struct cgroup *cont,
struct cftype *cft,
> >>>>   parent_memcg = mem_cgroup_from_cont(parent);
> >>>>
> >>>>   cgroup_lock();
> >>>>+
> >>>>+ if (memcg->use_hierarchy == val)
> >>>>+ goto out;

> >>>>+
> >>>
> >>>Why do you need cgroup_lock to check the value? Even if we have 2
> >>>CPUs racing (one trying to set to 0 other to 1 with use_hierarchy==0)
> >>>then the "set to 0" operation might fail depending on who hits the
> >>>cgroup_lock first anyway.
> >>>
> >>>So while this is correct I think there is not much point to take the global
> >>>cgroup lock in this case.
> >>>
> >>Well, no.
> >>
> >>All operations will succeed, unless the cgroup breeds new children.
> >>That's the operation we're racing against.
> >
> >I am not sure I understand. The changelog says that you want to handle
> >a situation where you are copying a hierarchy along with their
> >attributes and you don't want to fail when setting sane values.
> >
> >If we race with a new child creation then the success always depends on
> >the lock ordering but once the value is set then it is final so the test
> >will work even outside of the lock. Or am I still missing something?
> >
> >Just to make it clear the lock is necessary in the function I just do
> >not see why it should be held while we are trying to handle no-change
> >case.
> >
>
> I think you are right in this specific case. But do you think it is
> necessary to submit a version of it that tests outside the lock?
>
> We don't gain too much with that anyway.

Well, it was just a concern that the lock is global and the test doesn't
seem to need it. But maybe you are right and it is not worth it.

--
Michal Hocko
SUSE Labs
SUSE LINUX s.r.o.
Lihovarska 1060/12
190 00 Praha 9
Czech Republic

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Tejun Heo on Mon, 25 Jun 2012 20:49:08 GMT

View Forum Message <> Reply to Message

On Mon, Jun 25, 2012 at 01:21:01PM +0400, Glauber Costa wrote:
> I have an application that does the following:
>
> * copy the state of all controllers attached to a hierarchy
> * replicate it as a child of the current level.
>
> I would expect writes to the files to mostly succeed, since they
> are inheriting sane values from parents.
>
> But that is not the case for use_hierarchy. If it is set to 0, we
> succeed ok. If we're set to 1, the value of the file is automatically
> set to 1 in the children, but if userspace tries to write the
> very same 1, it will fail. That same situation happens if we
> set use_hierarchy, create a child, and then try to write 1 again.
>
> Now, there is no reason whatsoever for failing to write a value
> that is already there. It doesn't even match the comments, that
> states:
>
> /* If parent's use_hierarchy is set, we can't make any modifications
>   * in the child subtrees...
>
> since we are not changing anything.
>
> The following patch tests the new value against the one we're storing,
> and automatically return 0 if we're not proposing a change.

A bit of delta but is there any chance we can either deprecate
.use_hierarhcy or at least make it global toggle instead of subtree
thing? This seems needlessly complicated. :(

--
tejun

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Glauber Costa on Mon, 25 Jun 2012 22:26:48 GMT
View Forum Message <> Reply to Message

On 06/26/2012 12:49 AM, Tejun Heo wrote:
> On Mon, Jun 25, 2012 at 01:21:01PM +0400, Glauber Costa wrote:
>> I have an application that does the following:
>>
>> * copy the state of all controllers attached to a hierarchy
>> * replicate it as a child of the current level.
>>

>> I would expect writes to the files to mostly succeed, since they
>> are inheriting sane values from parents.
>>
>> But that is not the case for use_hierarchy. If it is set to 0, we
>> succeed ok. If we're set to 1, the value of the file is automatically
>> set to 1 in the children, but if userspace tries to write the
>> very same 1, it will fail. That same situation happens if we
>> set use_hierarchy, create a child, and then try to write 1 again.
>>
>> Now, there is no reason whatsoever for failing to write a value
>> that is already there. It doesn't even match the comments, that
>> states:
>>
>> /* If parent's use_hierarchy is set, we can't make any modifications
>>   * in the child subtrees...
>>
>> since we are not changing anything.
>>
>> The following patch tests the new value against the one we're storing,
>> and automatically return 0 if we're not proposing a change.
>
> A bit of delta but is there any chance we can either deprecate
> .use_hierarhcy or at least make it global toggle instead of subtree
> thing? This seems needlessly complicated. :(
>

I am for deprecating. If this is a long term goal, a two-phase process
making it per-tree seems unnecessary and even more confusing.

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Michal Hocko on Tue, 26 Jun 2012 07:56:53 GMT
View Forum Message <> Reply to Message

[Adding Ying to CC - they are using hierarchies AFAIU in their workloads]

On Mon 25-06-12 13:49:08, Tejun Heo wrote:
[...]
> A bit of delta but is there any chance we can either deprecate
> .use_hierarhcy or at least make it global toggle instead of subtree
> thing?

So what you are proposing is to have all subtrees of the root either
hierarchical or not, right?

> This seems needlessly complicated. :(

Toggle wouldn't help much I am afraid. We would still have to

distinguish (non)hierarchical cases. And I am not sure we can make
everything hierarchical easily.
Most users (from my experience) ignored use_hierarchy for some reasons
and the end results might be really unexpected for them if they used
deeper subtrees (which might be needed due to combination with other
controller(s)).
--
Michal Hocko
SUSE Labs
SUSE LINUX s.r.o.
Lihovarska 1060/12
190 00 Praha 9
Czech Republic

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Glauber Costa on Tue, 26 Jun 2012 10:31:51 GMT
View Forum Message <> Reply to Message

On 06/26/2012 11:56 AM, Michal Hocko wrote:
> [Adding Ying to CC - they are using hierarchies AFAIU in their workloads]
>
> On Mon 25-06-12 13:49:08, Tejun Heo wrote:
> [...]
>> A bit of delta but is there any chance we can either deprecate
>> .use_hierarhcy or at least make it global toggle instead of subtree
>> thing?
>
> So what you are proposing is to have all subtrees of the root either
> hierarchical or not, right?
>
>> This seems needlessly complicated. :(
>
> Toggle wouldn't help much I am afraid. We would still have to
> distinguish (non)hierarchical cases. And I am not sure we can make
> everything hierarchical easily.
> Most users (from my experience) ignored use_hierarchy for some reasons
> and the end results might be really unexpected for them if they used
> deeper subtrees (which might be needed due to combination with other
> controller(s)).
>
Do we have any idea about who those users are, and how is their setup
commonly done?

We can propose work arounds here, but not without first knowing work
arounds to what =p

One thing that would really influence this, for instance, is whether or

not they limit at all levels in the tree, etc.

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Michal Hocko on Tue, 26 Jun 2012 11:10:25 GMT
View Forum Message <> Reply to Message

On Tue 26-06-12 14:31:51, Glauber Costa wrote:
> On 06/26/2012 11:56 AM, Michal Hocko wrote:
> >[Adding Ying to CC - they are using hierarchies AFAIU in their workloads]
> >
> >On Mon 25-06-12 13:49:08, Tejun Heo wrote:
> >[...]
> >>A bit of delta but is there any chance we can either deprecate
> >>.use_hierarhcy or at least make it global toggle instead of subtree
> >>thing?
> >
> >So what you are proposing is to have all subtrees of the root either
> >hierarchical or not, right?
> >
> >>This seems needlessly complicated. :(
> >
> >Toggle wouldn't help much I am afraid. We would still have to
> >distinguish (non)hierarchical cases. And I am not sure we can make
> >everything hierarchical easily.
> >Most users (from my experience) ignored use_hierarchy for some reasons
> >and the end results might be really unexpected for them if they used
> >deeper subtrees (which might be needed due to combination with other
> >controller(s)).
> >
> Do we have any idea about who those users are, and how is their
> setup commonly done?

Well, most of them use memory controller with combination of other
controller - usually cpuset or cpu - and memcg is used to cap the amount
of memory for each respective group. As I said most of those users
were not aware of use_hierarchy at all.

> We can propose work arounds here, but not without first knowing work
> arounds to what =p

No, please no workarounds. It will be even bigger mess.
Maybe a global switch is the first step in the right direction (on by
default). If somebody encounters any issue we can say it can be turned
off (something like one time switch) or advise on how to fix their
layout to fit hierarchy better. We can put WARN_ON_ONCE when the knob is
set to 0 in the second stage and finally remove the whole knob.

> One thing that would really influence this, for instance, is whether
> or not they limit at all levels in the tree, etc.

Yes and independently. But it's true that I haven't seen many of
them to be honest, people usually use a flat structures.
--
Michal Hocko
SUSE Labs
SUSE LINUX s.r.o.
Lihovarska 1060/12
190 00 Praha 9
Czech Republic

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Glauber Costa on Tue, 26 Jun 2012 11:12:07 GMT
View Forum Message <> Reply to Message

On 06/26/2012 03:10 PM, Michal Hocko wrote:
> On Tue 26-06-12 14:31:51, Glauber Costa wrote:
>> On 06/26/2012 11:56 AM, Michal Hocko wrote:
>>> [Adding Ying to CC - they are using hierarchies AFAIU in their workloads]
>>>
>>> On Mon 25-06-12 13:49:08, Tejun Heo wrote:
>>> [...]
>>>> A bit of delta but is there any chance we can either deprecate
>>>> .use_hierarhcy or at least make it global toggle instead of subtree
>>>> thing?
>>>
>>> So what you are proposing is to have all subtrees of the root either
>>> hierarchical or not, right?
>>>
>>>> This seems needlessly complicated. :(
>>>
>>> Toggle wouldn't help much I am afraid. We would still have to
>>> distinguish (non)hierarchical cases. And I am not sure we can make
>>> everything hierarchical easily.
>>> Most users (from my experience) ignored use_hierarchy for some reasons
>>> and the end results might be really unexpected for them if they used
>>> deeper subtrees (which might be needed due to combination with other
>>> controller(s)).
>>>
>> Do we have any idea about who those users are, and how is their
>> setup commonly done?
>
> Well, most of them use memory controller with combination of other
> controller - usually cpuset or cpu - and memcg is used to cap the amount
> of memory for each respective group. As I said most of those users

> were not aware of use_hierarchy at all.
>
>> We can propose work arounds here, but not without first knowing work
>> arounds to what =p
>
> No, please no workarounds. It will be even bigger mess.
> Maybe a global switch is the first step in the right direction (on by
> default). If somebody encounters any issue we can say it can be turned
> off (something like one time switch) or advise on how to fix their
> layout to fit hierarchy better. We can put WARN_ON_ONCE when the knob is
> set to 0 in the second stage and finally remove the whole knob.
>

Sorry for the wording. I didn't mean work around in the sense of a
kludge. I meant it as actually proposing solutions to the problem that
would disrupt people as little as we can.

Well, instead of a global switch, a much easier thing would be to set it
to 1 by default. It would actually work as a global switch, because we
always inherit the parent's value.

You can set the root to 0 before you add other groups, but that
generates a warning, as you suggested.

But after it was first set to 0, he would be free to keep using mixed
configurations if needed - this way we're likely to find out if there
are actually users of that around.

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Tejun Heo on Tue, 26 Jun 2012 17:55:13 GMT
View Forum Message <> Reply to Message

Hello,

On Tue, Jun 26, 2012 at 09:56:53AM +0200, Michal Hocko wrote:
> [Adding Ying to CC - they are using hierarchies AFAIU in their workloads]

Ooh, I'm they. :) Asking around.... okay, so google does use
.use_hierarchy but it's a tree-wide thing and would be perfectly happy
with a global switch.

> On Mon 25-06-12 13:49:08, Tejun Heo wrote:
> [...]
> > A bit of delta but is there any chance we can either deprecate
> > .use_hierarhcy or at least make it global toggle instead of subtree
> > thing?
>

> So what you are proposing is to have all subtrees of the root either
> hierarchical or not, right?

Yeap. Just make it a global switch. Probably determined on mount
time.

> > This seems needlessly complicated. :(
>
> Toggle wouldn't help much I am afraid. We would still have to
> distinguish (non)hierarchical cases. And I am not sure we can make
> everything hierarchical easily.

I'm kinda confused by this paragraph. What do you mean by "wouldn't
help much"? Do you mean in terms of complexity?

> Most users (from my experience) ignored use_hierarchy for some reasons
> and the end results might be really unexpected for them if they used
> deeper subtrees (which might be needed due to combination with other
> controller(s)).

Oh yeah, we can't change the default behavior like that. The
transition should be a lot more gradual. Even if making
.use_hierarchy doesn't help much in terms of reducing complexity right
now, it would at least allow us to weed out and prevent wacky woo-hoo
mom-look-at-what-I-can-do configurations which will be a lot more
difficult to deal with for both us and such users (if we end up
forcing hierarchy).

Thanks.

--
tejun

Subject: Re: [PATCH] fix bad behavior in use_hierarchy file
Posted by Ying Han on Mon, 23 Jul 2012 17:22:03 GMT
View Forum Message <> Reply to Message

On Tue, Jun 26, 2012 at 12:56 AM, Michal Hocko <mhocko@suse.cz> wrote:
> [Adding Ying to CC - they are using hierarchies AFAIU in their workloads]

Sorry for late (a month late) to the thread.

Our current production today doesn't support multi-hierarchy setup for
memcg, and all the cgroups are flat under root at least on the memory
resource perspective. However, we do have use_hierarchy set to 1 to
root cgroup upfront.

On the other hand, we started exploring nested cgroup since the flat configuration doesn't fullfill all our usecases. In that case, we will have configurations like: root-> A -> B -> C (not sure about C but at least level to B). Of course, we will have use_hierarchy set to 1 on each level, and the mixed setting won't happen AFAIK.

--Ying
>
> On Mon 25-06-12 13:49:08, Tejun Heo wrote:
> [...]
>> A bit of delta but is there any chance we can either deprecate
>> .use_hierarhcy or at least make it global toggle instead of subtree
>> thing?
>
> So what you are proposing is to have all subtrees of the root either
> hierarchical or not, right?
>
>> This seems needlessly complicated. :(
>
> Toggle wouldn't help much I am afraid. We would still have to
> distinguish (non)hierarchical cases. And I am not sure we can make
> everything hierarchical easily.
> Most users (from my experience) ignored use_hierarchy for some reasons
> and the end results might be really unexpected for them if they used
> deeper subtrees (which might be needed due to combination with other
> controller(s)).
> --
> Michal Hocko
> SUSE Labs
> SUSE LINUX s.r.o.
> Lihovarska 1060/12
> 190 00 Praha 9
> Czech Republic