Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Paul Jackson on Thu, 01 Mar 2007 19:39:00 GMT
View Forum Message <> Reply to Message

vatsa wrote:
> I suspect we can make cpusets also work
> on top of this very easily.

I'm skeptical, and kinda worried.

... can you show me the code that does this?

Namespaces are not the same thing as actual resources
(memory, cpu cycles, ...). Namespaces are fluid mappings;
Resources are scarce commodities.

I'm wagering you'll break either the semantics, and/or the
performance, of cpusets doing this.

--
I won't rest till it's the best ...
Programmer, Linux Scalability
Paul Jackson <pj@sgi.com> 1.925.600.0401

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Kirill Korotaev on Fri, 02 Mar 2007 15:45:06 GMT
View Forum Message <> Reply to Message

Paul,

>>I suspect we can make cpusets also work
>>on top of this very easily.
>
>
> I'm skeptical, and kinda worried.
>
> ... can you show me the code that does this?
don't worry. we are not planning to commit any code breaking cpusets...
I will be the first one against it.

> Namespaces are not the same thing as actual resources

> (memory, cpu cycles, ...).  Namespaces are fluid mappings;
> Resources are scarce commodities.
hm... interesing comparison.
as for me, I can't see much difference between virtualization namespaces
and resource namespaces.

Both have some impact on what the task in the namespace can do and what can't.
The only difference is that virtualization namespaces usually also
make one user to be invisible to another. That's the only difference imho.

Also if you take a look at IPC namespace you'll note that IPC
can also limit IPC resources in question.
So it is kinda of virtualization + resource namespace.

> I'm wagering you'll break either the semantics, and/or the
> performance, of cpusets doing this.
I like Paul's containers patch. It looks good and pretty well.
After some of the context issues are resolved it's fine.
Maybe it is even the best way of doing things.

Thanks,
Kirill

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Srivatsa Vaddagiri on Sat, 03 Mar 2007 09:36:55 GMT
View Forum Message <> Reply to Message

On Thu, Mar 01, 2007 at 11:39:00AM -0800, Paul Jackson wrote:
> vatsa wrote:
> > I suspect we can make cpusets also work
> > on top of this very easily.
>
> I'm skeptical, and kinda worried.
>
> ... can you show me the code that does this?

In essense, the rcfs patch is same as the original containers
patch. Instead of using task->containers->container[cpuset->hierarchy]
to get to the cpuset structure for a task, it uses
task->nsproxy->ctlr_data[cpuset->subsys_id].

So if the original containers patches could implement cpusets on
containers abstraction, I don't see why it is not possible to implement
on top of nsproxy (which is essentialy same as container_group in Paul
Menage's patches). Any way code speaks best and I will try to post
something soon!

> Namespaces are not the same thing as actual resources
> (memory, cpu cycles, ...).  Namespaces are fluid mappings;
> Resources are scarce commodities.

Yes, perhaps this overloads nsproxy more than what it was intended for.
But, then if we have to to support resource management of each
container/vserver (or whatever group is represented by nsproxy), then nsproxy
seems the best place to store this resource control information for a container.

> I'm wagering you'll break either the semantics, and/or the
> performance, of cpusets doing this.

It should have the same perf overhead as the original container patches
(basically a double dereference - task->containers/nsproxy->cpuset -
required to get to the cpuset from a task).

Regarding semantics, can you be more specific?

In fact I think it will facilitate containers to use cpusets more
easily. You can for example divide the system into two (exclusive)
cpusets A and B, and have container C1 work inside A while C2 uses C2.
So c1's nsproxy->cpuset will point to A will c2's nsproxy->cpuset will
point to B. If you dont want to split the cpus into cpusets like that,
then all nsproxy's->cpuset will point to the top_cpuset.

Basically the rcfs patches demonstrate that is possible to keep track of
hierarchial relationship in resource objects using corresponding file system
objects itself (like dentries). Also if we are hooked to nsproxy, lot of
hard work to mainain life-time of nsproxy's (ref count ) is already in place -
we just reuse that work. These should help us avoid the container
structure abstraction in Paul Menage's patches (which was the main
point of objection from last time).

--
Regards,
vatsa

_____

Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Herbert Poetzl on Sat, 03 Mar 2007 17:32:44 GMT
View Forum Message <> Reply to Message

On Sat, Mar 03, 2007 at 03:06:55PM +0530, Srivatsa Vaddagiri wrote:
> On Thu, Mar 01, 2007 at 11:39:00AM -0800, Paul Jackson wrote:
> > vatsa wrote:
> > > I suspect we can make cpusets also work
> > > on top of this very easily.
> >
> > I'm skeptical, and kinda worried.
> >
> > ... can you show me the code that does this?
>
> In essense, the rcfs patch is same as the original containers
> patch. Instead of using task->containers->container[cpuset->hierarchy]
> to get to the cpuset structure for a task, it uses
> task->nsproxy->ctlr_data[cpuset->subsys_id].
>
> So if the original containers patches could implement cpusets on
> containers abstraction, I don't see why it is not possible to implement
> on top of nsproxy (which is essentialy same as container_group in Paul
> Menage's patches). Any way code speaks best and I will try to post
> something soon!
>
> > Namespaces are not the same thing as actual resources
> > (memory, cpu cycles, ...).  Namespaces are fluid mappings;
> > Resources are scarce commodities.
>
> Yes, perhaps this overloads nsproxy more than what it was intended for.
> But, then if we have to to support resource management of each
> container/vserver (or whatever group is represented by nsproxy),
> then nsproxy seems the best place to store this resource control
> information for a container.

well, the thing is, as nsproxy is working now, you
will get a new one (with a changed subset of entries)
every time a task does a clone() with one of the
space flags set, which means, that you will end up
with quite a lot of them, but resource limits have
to address a group of them, not a single nsproxy
(or act in a deeply hierarchical way which is not
there atm, and probably will never be, as it simply
adds too much overhead)

> > I'm wagering you'll break either the semantics, and/or the
> > performance, of cpusets doing this.
>

> It should have the same perf overhead as the original
> container patches (basically a double dereference -
> task->containers/nsproxy->cpuset - required to get to the
> cpuset from a task).

on every limit accounting or check? I think that
is quite a lot of overhead ...

best,
Herbert

> Regarding semantics, can you be more specific?
>
> In fact I think it will facilitate containers to use cpusets more
> easily. You can for example divide the system into two (exclusive)
> cpusets A and B, and have container C1 work inside A while C2 uses C2.
> So c1's nsproxy->cpuset will point to A will c2's nsproxy->cpuset will
> point to B. If you dont want to split the cpus into cpusets like that,
> then all nsproxy's->cpuset will point to the top_cpuset.
>
> Basically the rcfs patches demonstrate that is possible to keep track
> of hierarchial relationship in resource objects using corresponding
> file system objects itself (like dentries). Also if we are hooked to
> nsproxy, lot of hard work to mainain life-time of nsproxy's (ref count
> ) is already in place -

> we just reuse that work. These should help us avoid the container
> structure abstraction in Paul Menage's patches (which was the main
> point of objection from last time).
>
> --
> Regards,
> vatsa
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Herbert Poetzl on Sat, 03 Mar 2007 17:45:53 GMT
View Forum Message <> Reply to Message

On Fri, Mar 02, 2007 at 06:45:06PM +0300, Kirill Korotaev wrote:
> Paul,
>
> >>I suspect we can make cpusets also work
> >>on top of this very easily.
> >
> >
> > I'm skeptical, and kinda worried.
> >
> > ... can you show me the code that does this?
> don't worry. we are not planning to commit any code breaking
> cpusets.. I will be the first one against it
>
> > Namespaces are not the same thing as actual resources
> > (memory, cpu cycles, ...).  Namespaces are fluid mappings;
> > Resources are scarce commodities.
> hm... interesing comparison.

> as for me, I can't see much difference between virtualization
> namespaces nd resource namespaces.

I agree here, there is not much difference for the
following aspects:

 - resource accounting, limits and namespaces apply
   to a group of processes
 - they isolate those processes in some way from
   other groups of processes
 - they apply a virtual view and/or limitation to
   those processes

> Both have some impact on what the task in the namespace can do and
> what can't. The only difference is that virtualization namespaces
> usually also make one user to be invisible to another.

IMHO invisibility only applies to the pid space :)

but as I said, the processes are isolated in some
way, might it be pids, networking, ipc, uts or
filesystem, similar can be said for resource limits
and resource accounting, where you are only focusing
on a certain group of processes, applying an artifial
limit and ideally virtualizing all kernel interfaces
in such a way, that it looks like the artifical limit
is a real physical limitation

> That's the only difference imho.
>

> Also if you take a look at IPC namespace you'll note that IPC
> can also limit IPC resources in question.

yes, but they do it in a way a normal Linux system
would do, so no 'new' limits there, unless you
disallow changing those limits from inside a space

best,
Herbert

> So it is kinda of virtualization + resource namespace.
>
> > I'm wagering you'll break either the semantics, and/or the
> > performance, of cpusets doing this.
> I like Paul's containers patch. It looks good and pretty well.
> After some of the context issues are resolved it's fine.
> Maybe it is even the best way of doing things.
>
> Thanks,
> Kirill
>
> _____
> Containers mailing list
> Containers@lists.osdl.org
> https://lists.osdl.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers

---

Subject: Re: [ckrm-tech] [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Srivatsa Vaddagiri on Mon, 05 Mar 2007 17:34:01 GMT
View Forum Message <> Reply to Message

On Sat, Mar 03, 2007 at 06:32:44PM +0100, Herbert Poetzl wrote:
> > Yes, perhaps this overloads nsproxy more than what it was intended for.
> > But, then if we have to to support resource management of each
> > container/vserver (or whatever group is represented by nsproxy),
> > then nsproxy seems the best place to store this resource control
> > information for a container.
>
> well, the thing is, as nsproxy is working now, you
> will get a new one (with a changed subset of entries)
> every time a task does a clone() with one of the
> space flags set, which means, that you will end up
> with quite a lot of them, but resource limits have

> to address a group of them, not a single nsproxy
> (or act in a deeply hierarchical way which is not
> there atm, and probably will never be, as it simply
> adds too much overhead)

Thats why nsproxy has pointers to resource control objects, rather than
embedding resource control information in nsproxy itself.

>From the patches:

struct nsproxy {

+#ifdef CONFIG_RCFS
+ struct list_head list;
+ void *ctlr_data[CONFIG_MAX_RC_SUBSYS];
+#endif

}

This will let different nsproxy structures share the same resource
control objects (ctlr_data) and thus be governed by the same parameters.

Where else do you think the resource control information for a container
should be stored?

> > It should have the same perf overhead as the original
> > container patches (basically a double dereference -
> > task->containers/nsproxy->cpuset - required to get to the
> > cpuset from a task).
>
> on every limit accounting or check? I think that
> is quite a lot of overhead ...

tsk->nsproxy->ctlr_data[cpu_ctlr->id]->limit (4 dereferences) is what we
need to get to the cpu b/w limit for a task.

If cpu_ctlr->id is compile time decided, then that would reduce it to 3.

But I think if CPU scheduler schedules tasks from same container one
after another (to the extent possible that is), then other derefences
(->ctlr_data[] and ->limit) should be fast, as they should be in the cache?

--
Regards,
vatsa

_____
Containers mailing list