Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Mon, 30 Oct 2006 14:08:03 GMT
View Forum Message <> Reply to Message

[snip]

>
> Consensus/Debated Points
> ----------
>
> Consensus:
>
> - Provide resource control over a group of tasks
> - Support movement of task from one resource group to another
> - Dont support heirarchy for now
> - Support limit (soft and/or hard depending on the resource
> type) in controllers. Guarantee feature could be indirectly
> met thr limits.
>
> Debated:
> - syscall vs configfs interface

1. One of the major configfs ideas is that lifetime of
the objects is completely driven by userspace.
Resource controller shouldn't live as long as user
want. It "may", but not "must"! As you have seen from
our (beancounters) patches beancounters disapeared
as soon as the last reference was dropped. Removing
configfs entries on beancounter's automatic destruction
is possible, but it breaks the logic of configfs.

2. Having configfs as the only interface doesn't alow
people having resource controll facility w/o configfs.
Resource controller must not depend on any "feature".

3. Configfs may be easily implemented later as an additional
interface. I propose the following solution:
- First we make an interface via any common kernel
facility (syscall, ioctl, etc);
- Later we may extend this with configfs. This will
alow one to have configfs interface build as a module.

> - Interaction of resource controllers, containers and cpusets
> - Should we support, for instance, creation of resource
> groups/containers under a cpuset?
> - Should we have different groupings for different resources?

This breaks the idea of groups isolation.

> - Support movement of all threads of a process from one group
>   to another atomically?

This is not a critical question. This is something that
has difference in

- move_task_to_container(task);
+ do_each_thread_all(g, p) {
+ if (g->mm == task->mm)
+   move_task_to_container(g);
+ } while_each_thread_all(g, p);

or similar. If we have an infrastructure for accounting and
moving one task_struct into group then solution of how many
task to move in one syscall may be taken, but not the other
way round.

I also add devel@openvz.org to Cc. Please keep it on your replies.

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Paul Jackson on Mon, 30 Oct 2006 14:23:32 GMT
View Forum Message <> Reply to Message

Pavel wrote:
> 1. One of the major configfs ideas is that lifetime of
>   the objects is completely driven by userspace.
>   Resource controller shouldn't live as long as user
>   want. It "may", but not "must"!

I had trouble understanding what you are saying here.

What does the phrase "live as long as user want" mean?

> 2. Having configfs as the only interface doesn't alow
>   people having resource controll facility w/o configfs.
>   Resource controller must not depend on any "feature".
>
> 3. Configfs may be easily implemented later as an additional
>   interface. I propose the following solution:
>   - First we make an interface via any common kernel
>   facility (syscall, ioctl, etc);
>   - Later we may extend this with configfs. This will
>   alow one to have configfs interface build as a module.

So you would add bloat to the kernel, with two interfaces
to the same facility, because you don't want the resource
controller to depend on configfs.

I am familiar with what is wrong with kernel bloat.

Can you explain to me what is wrong with having resource
groups depend on configfs? Is there something wrong with
configfs that would be a significant problem for some systems
needing resource groups?

It is better where possible, I would think, to reuse common
infrastructure and minimize redundancy. If there is something
wrong with configfs that makes this a problem, perhaps we
should fix that.

--

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

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Mon, 30 Oct 2006 14:38:33 GMT
View Forum Message <> Reply to Message

Paul Jackson wrote:
> Pavel wrote:
>> 1. One of the major configfs ideas is that lifetime of
>>   the objects is completely driven by userspace.
>>   Resource controller shouldn't live as long as user
>>   want. It "may", but not "must"!
>
> I had trouble understanding what you are saying here.
>
> What does the phrase "live as long as user want" mean?

What if if user creates a controller (configfs directory)
and doesn't remove it at all. Should controller stay in memory
even if nobody uses it?

>
>
>> 2. Having configfs as the only interface doesn't alow
>>   people having resource controll facility w/o configfs.
>>   Resource controller must not depend on any "feature".
>>
>> 3. Configfs may be easily implemented later as an additional

>> interface. I propose the following solution:
>> - First we make an interface via any common kernel
>> facility (syscall, ioctl, etc);
>> - Later we may extend this with configfs. This will
>> alow one to have configfs interface build as a module.
>
> So you would add bloat to the kernel, with two interfaces
> to the same facility, because you don't want the resource
> controller to depend on configfs.
>
> I am familiar with what is wrong with kernel bloat.
>
> Can you explain to me what is wrong with having resource
> groups depend on configfs? Is there something wrong with

Resource controller has nothing common with confgifs.
That's the same as if we make netfilter depend on procfs.

> configfs that would be a significant problem for some systems
> needing resource groups?

Why do we need to make some dependency if we can avoid it?

> It is better where possible, I would think, to reuse common
> infrastructure and minimize redundancy. If there is something
> wrong with configfs that makes this a problem, perhaps we
> should fix that.

The same can be said about system calls interface, isn't it?

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Paul Jackson on Mon, 30 Oct 2006 15:18:38 GMT
View Forum Message <> Reply to Message

Pavel wrote:
> >> 3. Configfs may be easily implemented later as an additional
> >> interface. I propose the following solution:
> >> ...
> >
> Resource controller has nothing common with confgifs.
> That's the same as if we make netfilter depend on procfs.

Well ... if you used configfs as an interface to resource
controllers, as you said was easily done, then they would
have something to do with each other, right ;)?

Choose the right data structure for the job, and then reuse

what fits for that choice.

Neither avoid nor encouraging code reuse is the key question.

What's the best fit, long term, for the style of kernel-user
API, for this use? That's the key question.

--

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

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Mon, 30 Oct 2006 15:26:49 GMT
View Forum Message <> Reply to Message

Paul Jackson wrote:
> Pavel wrote:
>>>> 3. Configfs may be easily implemented later as an additional
>>>> interface. I propose the following solution:
>>>> ...
>> Resource controller has nothing common with confgifs.
>> That's the same as if we make netfilter depend on procfs.
>
> Well ... if you used configfs as an interface to resource
> controllers, as you said was easily done, then they would
> have something to do with each other, right ;)?

Right. We'll create a dependency that is not needed.

> Choose the right data structure for the job, and then reuse
> what fits for that choice.
>
> Neither avoid nor encouraging code reuse is the key question.
>
> What's the best fit, long term, for the style of kernel-user
> API, for this use? That's the key question.

I agree, but you've cut some importaint questions away,
so I ask them again:

 > What if if user creates a controller (configfs directory)
 > and doesn't remove it at all. Should controller stay in
 > memory even if nobody uses it?

This is importaint to solve now - wether we want or not to
keep "empty" beancounters in memory. If we do not then configfs

usage is not acceptible.

> The same can be said about system calls interface, isn't it?

I haven't seen any objections against system calls yet.

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Paul Menage on Mon, 30 Oct 2006 18:01:40 GMT
View Forum Message <> Reply to Message

On 10/30/06, Pavel Emelianov <xemul@openvz.org> wrote:
> > Debated:
> > - syscall vs configfs interface
>
> 1. One of the major configfs ideas is that lifetime of
> the objects is completely driven by userspace.
> Resource controller shouldn't live as long as user
> want. It "may", but not "must"! As you have seen from
> our (beancounters) patches beancounters disapeared
> as soon as the last reference was dropped.

Why is this an important feature for beancounters? All the other resource control approaches seem to prefer having userspace handle removing empty/dead groups/containers.

> 2. Having configfs as the only interface doesn't alow
> people having resource controll facility w/o configfs.
> Resource controller must not depend on any "feature".

Why is depending on a feature like configfs worse than depending on a feature of being able to extend the system call interface?

> > - Interaction of resource controllers, containers and cpusets
> > - Should we support, for instance, creation of resource
> > groups/containers under a cpuset?
> > - Should we have different groupings for different resources?
>
> This breaks the idea of groups isolation.

That's fine - some people don't want total isolation. If we're looking for a solution that fits all the different requirements, then we need that flexibility. I agree that the default would probably want to be that the groupings be the same for all resource controllers / subsystems.

Paul

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Matt Helsley on Tue, 31 Oct 2006 00:26:17 GMT
View Forum Message <> Reply to Message

On Mon, 2006-10-30 at 18:26 +0300, Pavel Emelianov wrote:
> Paul Jackson wrote:
> > Pavel wrote:
> >>>> 3. Configfs may be easily implemented later as an additional
> >>>> interface. I propose the following solution:
> >>>> ...
> >> Resource controller has nothing common with confgifs.
> >> That's the same as if we make netfilter depend on procfs.
> >
> > Well ... if you used configfs as an interface to resource
> > controllers, as you said was easily done, then they would
> > have something to do with each other, right ;)?
>
> Right. We'll create a dependency that is not needed.
>
> > Choose the right data structure for the job, and then reuse
> > what fits for that choice.
> >
> > Neither avoid nor encouraging code reuse is the key question.
> >
> > What's the best fit, long term, for the style of kernel-user
> > API, for this use? That's the key question.
>
> I agree, but you've cut some importaint questions away,
> so I ask them again:
>
> > What if if user creates a controller (configfs directory)
> > and doesn't remove it at all. Should controller stay in
> > memory even if nobody uses it?

 Yes. The controller should stay in memory until userspace decides that control of the resource is no longer desired. Though not all controllers should be removable since that may impose unreasonable restrictions on what useful/performant controllers can be implemented.

 That doesn't mean that the controller couldn't reclaim memory it uses when it's no longer needed.

<snip>

Cheers,
 -Matt Helsley

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Tue, 31 Oct 2006 08:31:28 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On 10/30/06, Pavel Emelianov <xemul@openvz.org> wrote:
>> > Debated:
>> > - syscall vs configfs interface
>>
>> 1. One of the major configfs ideas is that lifetime of
>> the objects is completely driven by userspace.
>> Resource controller shouldn't live as long as user
>> want. It "may", but not "must"! As you have seen from
>> our (beancounters) patches beancounters disapeared
>> as soon as the last reference was dropped.
>
> Why is this an important feature for beancounters? All the other
> resource control approaches seem to prefer having userspace handle
> removing empty/dead groups/containers.

That's functionality user may want. I agree that some users
may want to create some kind of "persistent" beancounters, but
this must not be the only way to control them. I like the way
TUN devices are done. Each has TUN_PERSIST flag controlling
whether or not to destroy device right on closing. I think that
we may have something similar - a flag BC_PERSISTENT to keep
beancounters with zero refcounter in memory to reuse them.

Objections?

>> 2. Having configfs as the only interface doesn't alow
>> people having resource controll facility w/o configfs.
>> Resource controller must not depend on any "feature".
>
> Why is depending on a feature like configfs worse than depending on a
> feature of being able to extend the system call interface?

Because configfs is a _feature_, while system calls interface is
a mandatory part of a kernel. Since "resource beancounters" is a
core thing it shouldn't depend on "optional" kernel stuff. E.g.
procfs is the way userspace gets information about running tasks,
but disabling procfs doesn't disable such core functionality
as fork-ing and execve-ing.

Moreover, I hope you agree that beancounters can't be made as
module. If so user will have to built-in configfs, and thus
CONFIG_CONFIGFS_FS essentially becomes "bool", not a "tristate".

I have nothing against using configfs as additional, optional

interface, but I do object using it as the only window inside
BC world.

>> > - Interaction of resource controllers, containers and cpusets
>> > - Should we support, for instance, creation of resource
>> > groups/containers under a cpuset?
>> > - Should we have different groupings for different resources?
>>
>> This breaks the idea of groups isolation.
>
> That's fine - some people don't want total isolation. If we're looking
> for a solution that fits all the different requirements, then we need
> that flexibility. I agree that the default would probably want to be
> that the groupings be the same for all resource controllers /
> subsystems.

Hm... OK, I don't mind although don't see any reasonable use of it.
Thus we add one more point to our "agreement" list
http://wiki.openvz.org/Containers/UBC_discussion

- all resource groups are independent

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Tue, 31 Oct 2006 08:34:59 GMT
View Forum Message <> Reply to Message

[snip]

> Yes. The controller should stay in memory until userspace decides that
> control of the resource is no longer desired. Though not all controllers
> should be removable since that may impose unreasonable restrictions on
> what useful/performant controllers can be implemented.
>
> That doesn't mean that the controller couldn't reclaim memory it uses
> when it's no longer needed.
>

I've already answered Paul Menage about this. Shortly:

... I agree that some users may want to create some
kind of "persistent" beancounters, but this must not be
the only way to control them...
... I think that we may have something [like this] - a flag
BC_PERSISTENT to keep beancounters with zero refcounter in
memory to reuse them...
... I have nothing against using configfs as additional,

optional interface, but I do object using it as the only
window inside BC world...

Please, refer to my full reply for comments.

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Chris Friesen on Tue, 31 Oct 2006 16:33:14 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> Paul Jackson wrote:

> I agree, but you've cut some importaint questions away,
> so I ask them again:
>
> > What if if user creates a controller (configfs directory)
> > and doesn't remove it at all. Should controller stay in
> > memory even if nobody uses it?
>
> This is importaint to solve now - wether we want or not to
> keep "empty" beancounters in memory. If we do not then configfs
> usage is not acceptible.

I can certainly see scenarios where we would want to keep "empty"
beancounters around.

For instance, I move all the tasks out of a group but still want to be
able to obtain stats on how much cpu time the group has used.

Maybe we can do that without persisting the actual beancounters...I'm
not familiar enough with the code to say.

Chris

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Srivatsa Vaddagiri on Tue, 31 Oct 2006 16:34:18 GMT
View Forum Message <> Reply to Message

On Mon, Oct 30, 2006 at 05:08:03PM +0300, Pavel Emelianov wrote:
> 1. One of the major configfs ideas is that lifetime of
> the objects is completely driven by userspace.
> Resource controller shouldn't live as long as user
> want. It "may", but not "must"! As you have seen from
> our (beancounters) patches beancounters disapeared
> as soon as the last reference was dropped. Removing

> configfs entries on beancounter's automatic destruction
> is possible, but it breaks the logic of configfs.

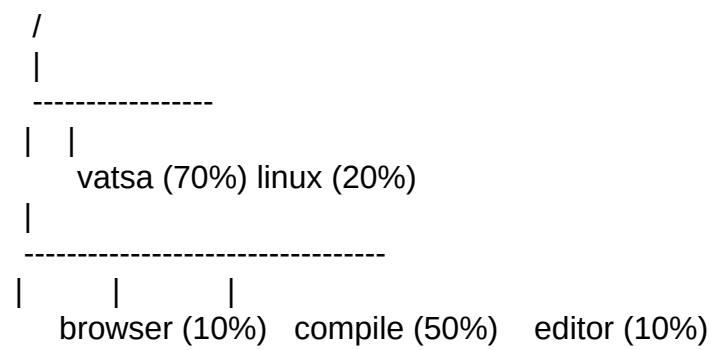
cpusets has a neat flag called notify_on_release. If set, some userspace agent is invoked when the last task exists from a cpuset.

Can't we use a similar flag as a configfs file and (if set) invoke a userspace agent (to cleanup) upon last reference drop? How would this violate logic of configfs?

> 2. Having configfs as the only interface doesn't alow
> people having resource controll facility w/o configfs.
> Resource controller must not depend on any "feature".

One flexibility configfs (and any fs-based interface) offers is, as Matt had pointed out sometime back, the ability to delage management of a sub-tree to a particular user (without requiring root permission).

For ex:



In this, group 'vatsa' has been alloted 70% share of cpu. Also user 'vatsa' has been given permissions to manage this share as he wants. If the cpu controller supports hierarchy, user 'vatsa' can create further sub-groups (browser, compile ..etc) -without- requiring root access.

Also it is convenient to manipulate resource hierarchy/parameters thr a shell-script if it is fs-based.

> 3. Configfs may be easily implemented later as an additional
> interface. I propose the following solution:

Ideally we should have one interface - either syscall or configfs - and not both.

Assuming your requirement of auto-deleting objects in configfs can be met thr' something similar to cpuset's notify_on_release, what other killer problem do you think configfs will pose?

> > - Should we have different groupings for different resources?
>
> This breaks the idea of groups isolation.

Sorry dont get you here. Are you saying we should support different
grouping for different controllers?

> > - Support movement of all threads of a process from one group
> >   to another atomically?
>
> This is not a critical question. This is something that
> has difference in

It can be a significant pain for some workloads. I have heard that
workload management products often encounter processes with anywhere
between 200-700 threads in a process. Moving all those threads one by
one from user-space can suck.

--
Regards,
vatsa

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Paul Menage on Tue, 31 Oct 2006 16:34:52 GMT
View Forum Message <> Reply to Message

On 10/31/06, Pavel Emelianov <xemul@openvz.org> wrote:
>
> That's functionality user may want. I agree that some users
> may want to create some kind of "persistent" beancounters, but
> this must not be the only way to control them. I like the way
> TUN devices are done. Each has TUN_PERSIST flag controlling
> whether or not to destroy device right on closing. I think that
> we may have something similar - a flag BC_PERSISTENT to keep
> beancounters with zero refcounter in memory to reuse them.

How about the cpusets approach, where once a cpuset has no children
and no processes, a usermode helper can be executed - this could
immediately remove the container/bean-counter if that's what the user
wants. My generic containers patch copies this from cpusets.

>
> Moreover, I hope you agree that beancounters can't be made as
> module. If so user will have to built-in configfs, and thus
> CONFIG_CONFIGFS_FS essentially becomes "bool", not a "tristate".

How about a small custom filesystem as part of the containers support, then? I'm not wedded to using configfs itself, but I do think that a filesystem interface is much more debuggable and extensible than a system call interface, and the simple filesystem is only a couple of hundred lines.

Paul

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Srivatsa Vaddagiri on Tue, 31 Oct 2006 16:57:48 GMT
View Forum Message <> Reply to Message

On Tue, Oct 31, 2006 at 08:34:52AM -0800, Paul Menage wrote:
> How about the cpusets approach, where once a cpuset has no children
> and no processes, a usermode helper can be executed - this could
> immediately remove the container/bean-counter if that's what the user
> wants. My generic containers patch copies this from cpusets.

Bingo. We crossed mails!

Kirill/Pavel,
As I mentioned in the begining of this thread, one of the objective of this RFC is to seek consensus on what could be a good compromise for the infrastructure in going forward. Paul Menage's patches, being rework of existing code, is attactive to maintainers like Andew.

>From that perspective, how well do you think the container infrastructure patches meet your needs?

--
Regards,
vatsa

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Wed, 01 Nov 2006 07:58:42 GMT
View Forum Message <> Reply to Message

Paul Menage wrote:
> On 10/31/06, Pavel Emelianov <xemul@openvz.org> wrote:
>>
>> That's functionality user may want. I agree that some users
>> may want to create some kind of "persistent" beancounters, but
>> this must not be the only way to control them. I like the way

>> TUN devices are done. Each has TUN_PERSIST flag controlling
>> whether or not to destroy device right on closing. I think that
>> we may have something similar - a flag BC_PERSISTENT to keep
>> beancounters with zero refcounter in memory to reuse them.
>
> How about the cpusets approach, where once a cpuset has no children
> and no processes, a usermode helper can be executed - this could

Hmm... Sounds good. I'll think over this.

> immediately remove the container/bean-counter if that's what the user
> wants. My generic containers patch copies this from cpusets.
>
>>
>> Moreover, I hope you agree that beancounters can't be made as
>> module. If so user will have to built-in configfs, and thus
>> CONFIG_CONFIGFS_FS essentially becomes "bool", not a "tristate".
>
> How about a small custom filesystem as part of the containers support,
> then? I'm not wedded to using configfs itself, but I do think that a
> filesystem interface is much more debuggable and extensible than a
> system call interface, and the simple filesystem is only a couple of
> hundred lines.

This sounds more reasonable than using configfs for me.

> Paul
>

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Wed, 01 Nov 2006 08:01:31 GMT
View Forum Message <> Reply to Message

[snip]

>> 2. Having configfs as the only interface doesn't alow
>>   people having resource controll facility w/o configfs.
>>   Resource controller must not depend on any "feature".
>
> One flexibility configfs (and any fs-based interface) offers is, as Matt
> had pointed out sometime back, the ability to delage management of a
> sub-tree to a particular user (without requiring root permission).
>
> For ex:
>
>   /
> |

> ----------
> | |
> vatsa (70%) linux (20%)
> |
> ----------
> | | |
> browser (10%) compile (50%) editor (10%)
>
> In this, group 'vatsa' has been alloted 70% share of cpu. Also user
> 'vatsa' has been given permissions to manage this share as he wants. If
> the cpu controller supports hierarchy, user 'vatsa' can create further
> sub-groups (browser, compile ..etc) -without- requiring root access.

I can do the same using bcctl tool and sudo :)

> Also it is convenient to manipulate resource hierarchy/parameters thr a
> shell-script if it is fs-based.
>
>> 3. Configfs may be easily implemented later as an additional
>> interface. I propose the following solution:
>
> Ideally we should have one interface - either syscall or configfs - and
> not both.

Agree.

> Assuming your requirement of auto-deleting objects in configfs can be
> met thr' something similar to cpuset's notify_on_release, what other
> killer problem do you think configfs will pose?
>
>
>>> - Should we have different groupings for different resources?
>> This breaks the idea of groups isolation.
>
> Sorry dont get you here. Are you saying we should support different
> grouping for different controllers?

Not me, but other people in this thread.

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Srivatsa Vaddagiri on Wed, 01 Nov 2006 17:45:16 GMT
View Forum Message <> Reply to Message

On Wed, Nov 01, 2006 at 11:01:31AM +0300, Pavel Emelianov wrote:
> > Sorry dont get you here. Are you saying we should support different
> > grouping for different controllers?
>

> Not me, but other people in this thread.

Hmm ..I thought OpenVz folks were interested in having different
groupings for different resources i.e grouping for CPU should be
independent of the grouping for memory.

 http://lkml.org/lkml/2006/8/18/98

Isnt that true?

--
Regards,
vatsa

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices
Posted by Pavel Emelianov on Thu, 02 Nov 2006 08:42:41 GMT
View Forum Message <> Reply to Message

Srivatsa Vaddagiri wrote:
> On Wed, Nov 01, 2006 at 11:01:31AM +0300, Pavel Emelianov wrote:
>>> Sorry dont get you here. Are you saying we should support different
>>> grouping for different controllers?
>> Not me, but other people in this thread.
>
> Hmm ..I thought OpenVz folks were interested in having different
> groupings for different resources i.e grouping for CPU should be
> independent of the grouping for memory.
>
> http://lkml.org/lkml/2006/8/18/98
>
> Isnt that true?

That's true. We don't mind having different groupings for
different resources. But what I was sying in this thread is
"I didn't *propose* this thing, I just *agreed* that this
might be usefull for someone."

So if we're going to have different groupings for different
resources what's the use of "container" grouping all "controllers"
together? I see this situation like each task_struct carries
pointers to kmemsize controller, pivate pages controller,
physical pages controller, CPU time controller, disk bandwidth
controller, etc. Right? Or did I miss something?

Subject: Re: [ckrm-tech] [RFC] Resource Management - Infrastructure choices

Posted by David Rientjes on Fri, 03 Nov 2006 01:29:32 GMT
View Forum Message <> Reply to Message

On Thu, 2 Nov 2006, Pavel Emelianov wrote:

> So if we're going to have different groupings for different
> resources what's the use of "container" grouping all "controllers"
> together? I see this situation like each task_struct carries
> pointers to kmemsize controller, pivate pages controller,
> physical pages controller, CPU time controller, disk bandwidth
> controller, etc. Right? Or did I miss something?

My understanding is that the only addition to the task_struct is a pointer to the struct container it belongs to. Then, the various controllers can register the control files through the fs-based container interface and all the manipulation can be done at that level. Having each task_struct containing pointers to individual resource nodes was never proposed.

David