Subject: Re: [PATCH 1/2] rcfs core patch
Posted by Herbert Poetzl on Fri, 09 Mar 2007 13:21:37 GMT
View Forum Message <> Reply to Message

On Fri, Mar 09, 2007 at 12:23:55PM +0300, Kirill Korotaev wrote:
>>> There have been various projects attempting to provide
>>> resource management support in Linux, including
>>> CKRM/Resource Groups and UBC.
>>
>> let me note here, once again, that you forgot Linux-VServer
>> which does quite non-intrusive resource management ...

> Herbert, do you care to send patches except for ask
> others to do something that works for you?

sorry, I'm not in the lucky position that I get payed
for sending patches to LKML, so I have to think twice
before I invest time in coding up extra patches ...

i.e. you will have to live with my comments for now

> Looks like your main argument is non-intrusive...
> "working", "secure", "flexible" are not required to
> people any more? :/

well, Linux-VServer is "working", "secure", "flexible"
and non-intrusive ... it is quite natural that less
won't work for me ... and regarding patches, there
will be a 2.2 release soon, with all the patches ...

>>> Each had its own task-grouping mechanism.

>> the basic 'context' (pid space) is the grouping mechanism
>> we use for resource management too

>>> Paul Menage observed [1] that cpusets in the kernel already has a
>>> grouping mechanism which was working well for cpusets. He went ahead
>>> and generalized the grouping code in cpusets so that it could be
>>> used for overall resource management purpose.

>>> With his patches, it is possible to even create multiple hierarchies
>>> of groups (see [2] on why multiple hierarchies) as follows:

>> do we need or even want that? IMHO the hierarchical
>> concept CKRM was designed with, was also the reason
>> for it being slow, unuseable and complicated

> 1. cpusets are hierarchical already. So hierarchy is required.

> 2. As it was discussed on the call controllers which are flat
>   can just prohibit creation of hierarchy on the filesystem.
>   i.e. allow only 1 depth and continue being fast.
>
>>> mount -t container -o cpuset none /dev/cpuset <- cpuset hierarchy
>>> mount -t container -o mem,cpu none /dev/mem <- memory/cpu hierarchy
>>> mount -t container -o disk none /dev/disk <- disk hierarchy
>>>
>>> In each hierarchy, you can create task groups and manipulate the
>>> resource parameters of each group. You can also move tasks between
>>> groups at run-time (see [3] on why this is required).

>>> Each hierarchy is also manipulated independent of the other.

>>> Paul's patches also introduced a 'struct container' in the kernel,
>>> which serves these key purposes:
>>>
>>> - Task-grouping
>>>   'struct container' represents a task-group created in each hierarchy.
>>>   So every directory created under /dev/cpuset or /dev/mem above will
>>>   have a corresponding 'struct container' inside the kernel. All tasks
>>>   pointing to the same 'struct container' are considered to be part of
>>>   a group
>>>
>>>   The 'struct container' in turn has pointers to resource objects which
>>>   store actual resource parameters for that group. In above example,
>>>   'struct container' created under /dev/cpuset will have a pointer to
>>>   'struct cpuset' while 'struct container' created under /dev/disk will
>>>   have pointer to 'struct disk_quota_or_whatever'.
>>>
>>> - Maintain hierarchical information
>>>   The 'struct container' also keeps track of hierarchical relationship
>>>   between groups.
>>>
>>> The filesystem interface in the patches essentially serves these
>>> purposes:
>>>
>>> - Provide an interface to manipulate task-groups. This includes
>>>   creating/deleting groups, listing tasks present in a group and
>>>   moving tasks across groups
>>>
>>> - Provdes an interface to manipulate the resource objects
>>>   (limits etc) pointed to by 'struct container'.
>>>
>>> As you know, the introduction of 'struct container' was objected
>>> to and was felt redundant as a means to group tasks. Thats where I
>>> took a shot at converting over Paul Menage's patch to avoid 'struct
>>> container' abstraction and insead work with 'struct nsproxy'.

>> which IMHO isn't a step in the right direction, as
>> you will need to handle different nsproxies within
>> the same 'resource container' (see previous email)

> tend to agree.
> Looks like Paul's original patch was in the right way.

> [...]

>>> A separate filesystem would give us more flexibility like the
>>> implementing multi-hierarchy support described above.

>> why is the filesystem approach so favored for this
>> kind of manipulations?

>> IMHO it is one of the worst interfaces I can imagine
>> (to move tasks between spaces and/or assign resources)
>> but yes, I'm aware that filesystems are 'in' nowadays

> I also hate filesystems approach being used nowdays everywhere.
> But, looks like there are reasons still:
> 1. cpusets already use fs interface.
> 2. each controller can have a bit of specific
>   information/controls exported easily.

yes, but there are certain drawbacks too, like:

 - performance of filesystem interfaces is quite bad
 - you need to do a lot to make the fs consistant for
   e.g. find and friends (regarding links and filesize)
 - you have a quite hard time to do atomic operations
   (except for the ioctl interface, which nobody likes)
 - vfs/mnt namespaces complicate the access to this
   new filesystem once you start moving around (between
   the spaces)

> Can you suggest any other extensible/flexible interface for these?

well, as you know, all current solutions use a syscall
interface to do most of the work, in the OpenVZ/Virtuozzo
case several, unassigned syscalls are used, while
FreeVPS and Linux-VServer use a registered and versioned
(multiplexed) system call, which works quite fine for
all known purposes ...

I'm quite happy with the extensibility and flexibility
the versioned syscall interface has, the only thing I'd

change if I would redesign that interface is, that I
would add another pointer argument to eliminate 32/64bit
issues completely (i.e. use 4 args instead of the 3)

best,
Herbert

> Thanks,
> Kirill
>
> -
> 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.osdl.org
https://lists.osdl.org/mailman/listinfo/containers