Subject: Re: Summary of resource management discussion
Posted by ebiederm on Fri, 16 Mar 2007 20:03:03 GMT
View Forum Message <> Reply to Message

Srivatsa Vaddagiri <vatsa@in.ibm.com> writes:

> On Thu, Mar 15, 2007 at 12:12:50PM -0700, Paul Menage wrote:
>> >Yes me too. But maybe to keep in simple in initial versions, we should
>> >avoid that optimisation and at the same time get statistics on duplicates?.
>>
>> That's an implementation detail - we have more important points to
>> agree on right now ...
>
> yes :)
>
> Eric, did you have any opinion on this thread?

A little. Sorry for the delay this is my low priority discussion to follow. Here comes a brain dump...

I do know of one case outside the context of containers/vservers where I would have liked a nice multi-process memory limit. HPC jobs doing rdma have a bad habit of wanting to mlock all of memory and thus send the memory allocator into a tail spin. So the separate use would be nice.

I do suspect that we are likely to look at having a hierarchy for most of the limits. For limits if we can stand the cost having a hierarchy makes sense. However I don't see the advantage of sharing the hierarchy between different classes of resource groups.

I do know that implementing hierarchical counters are inefficient for tracking resource consumption, and if we support deep hierarchies I expect we will want to optimize that, in non-trivial ways. At which point the hierarchy will be composed of more that just a pointer to it's parent. We could easily end up with children taking a lease from their parents saying you can have this many more before you look upwards in the hierarchy again. So with non-trivial hierarchy information adding a pointer at a generic level doesn't seem to be much of a help.

Further my gut feel is that in general we will want to limit all resources on a per container basis. At the same time I expect we will want to limit other resources to select process groups or users even farther inside of a container. So a single hierarchy for multiple resources seems a little questionable.

Which suggests that we want what I think was called multi-hierarchy support.

I guess also with hierarchies and entering we probably need a limit such that if you enter another resource group you can't do anything except stay at the same level or descend into the hierarchy. But you can never remove your parent of that resource type.

With the whole shared subtree concept the mount namespace stores things you can enter into in the mount namespace. This overcomes the difficulty of having to find a process who has the resources you want when you want to enter someplace. I think that concept is a benefit of a filesystem interface. Using mount and unmount to pin magic subsystem state seems a little more natural to me then having to do other filesystem manipulations. Especially since the mount namespace is reference counted so you can be certain everything you have pinned will either remain visible to someone or automatically disappear. I don't like interfaces like sysvipc that require manual destruction. I do think there is some sense in layout out a palette into the mount namespace we can enter into.

There is another issue I'm not quite certain how to address. To some extent it makes sense to be able to compile out the resource controllers ensuring their overhead goes away completely. However for the most part I think it makes sense to assume that when they are compiled in we have an initial set of resource controllers that either doesn't limit us at all or has very liberal limits that have the same effect. Dealing with the case of possibly limiting things when we have the support compiled in does not seem to make a lot of sense to me.

I hope that helps a little. I'm still coming to terms with the issues brought on by resource groups and controlling filesystems.

Eric