Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Paul Jackson on Sat, 03 Mar 2007 21:22:44 GMT
View Forum Message <> Reply to Message

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

Whether two somewhat similar needs should be met by one shared mechanism, or two distinct mechanisms, cannot really be decided by listing the similarities.

One has to determine if there are any significant differences in needs that are too difficult for a shared mechanism to provide.

A couple of things you wrote in your second message might touch on such possible significant differences:
 - resources must be hierarchically suballocated, and
 - key resource management code hooks can't cause hot cache lines.

In a later message, Herbert wrote:
> 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 still can't claim to have my head around this, but what you write here, Herbert, writes here touches on what I suspect is a key difference between namespaces and resources that would make it impractical to accomplish both with a shared mechanism for aggregating tasks.

It is a natural and desirable capability when managing resources, which are relatively scarce (that's why they're worth all this trouble) commodities of which we have some limited amount, to subdivide allowances of them. Some group of tasks gets the right to use certain memory pages or cpu time slices, and in turn suballocates that allotment to some subgroup of itself. This naturally leads to a hierarchy of allocated resources.

There is no such necessary, hierarchy for name spaces. One name space might be derived from another at setup, by some arbitrary conventions,

but once initialized, this way or that, they are separate name spaces,
or at least naturally can (must?) be separate.

The cpuset hierarchy is an important part of the API that cpusets
presents to user space, where that hierarchy reflects the suballocation
of resources. If B is a child of A in the cpuset hierarchy, then
the CPUs and Memory Nodes allowed to B -must- be a subset of those allowed
to A. That is the key semantic of the cpuset hierarchy. This includes
forcing the removal of a resource from B if for some reason it must
be removed from A, in order to preserve the hierarchical suballocation,
which requirement is causing a fair bit of hard work for the cpu hot
unplug folks.

I am quite willing to believe that name spaces has no need for such a
hierarchy, and further that it probably never will have such ... "too
much overhead" as you say.

> > 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 ...

Do either of these dereferences require locks?

The two critical resources that cpusets manages, memory pages and
time slices on a cpu, cannot afford such dereferences or locking
in the key code paths (allocating a page or scheduling a cpu.) The
existing cpuset code is down to one RCU guarded dereference of
current->cpuset in the page allocation code path (and then only on
systems actively using cpusets), and no such dereferences at all in the
scheduler path.

It took a fair bit of hard work (for someone of my modest abilities)
to get that far; I doubt we can accept much regression on this point.

Most likely the other folks doing resource management will have similar
concerns in many cases - memory pages and cpu slices are not the only
resources we're trying to manage on critical code paths.

In short - the issues seem to be:
 - resources need to be hierarchical, name spaces don't (can't?), and
 - no hot cache lines allowed by the resource hooks in key code paths.

--

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