Subject: Re: [PATCH 0/2] resource control file system - aka containers on top of nsproxy!
Posted by Paul Jackson on Fri, 09 Mar 2007 04:27:13 GMT
View Forum Message <> Reply to Message

> But "namespace" has well-established historical semantics too - a way
> of changing the mappings of local * to global objects. This
> accurately describes things liek resource controllers, cpusets, resource
> monitoring, etc.

No!

Cpusets don't rename or change the mapping of objects.

I suspect you seriously misunderstand cpusets and are trying to cram them into a 'namespace' remapping role into which they don't fit.

So far as cpusets are concerned, CPU #17 is CPU #17, for all tasks, regardless of what cpuset they are in. They just might not happen to be allowed to execute on CPU #17 at the moment, because that CPU is not allowed by the cpuset they are in.

But they still call it CPU #17.

Similary the namespace of cpusets and of tasks (pid's) are single system-wide namespaces, so far as cpusets are concerned.

Cpusets are not about alternative or multiple or variant name spaces.
They are about (considering just CPUs for the moment):
1) creating a set of maps M0, M1, ... from the set of CPUs to a Boolean,
2) creating a mapping Q from the set of tasks to these M0, ... maps, and
3) imposing constraints on where tasks can run, as follows:
For any task t, that task is allowed to run on CPU x iff Q(t)(x)
is True. Here, Q(t) will be one of the maps M0, ... aka a cpuset.

So far as cpusets are concerned, there is only one each of:
A] a namespace numbering CPUs,
B] a namespace numbering tasks (the process id),
C] a namespace naming cpusets (the hierarchical name space normally
mounted at /dev/cpuset, and corresponding to the Mn maps above) and
D] a mapping of tasks to cpusets, system wide (just a map, not a namespace.)

All tasks (of sufficient authority) can see each of these, using a single system wide name space for each of [A], [B], and [C].

Unless, that is, you call any mapping a "way of changing mappings".
To do so would be a senseless abuse of the phrase, in my view.

More generally, these resource managers all tend to divide some external limited physical resource into multiple separately allocatable units.

If the resource is amorphous (one atom or cycle of it is interchangeable with another) then we usually do something like divide it into 100 equal units and speak of percentages. If the resource is naturally subdivided into sufficiently small units (sufficient for the granularity of resource management we require) then we take those units as is. Occassionally, as in the 'fake numa node' patch by David Rientjes <rientjes@cs.washington.edu>, who worked at Google over the last summer, if the natural units are not of sufficient granularity, we fake up a somewhat finer division.

Then, in any case, and somewhat separately, we divide the tasks running on the system into subsets. More precisely, we partition the tasks, where a partition of a set is a set of subsets of that set, pairwise disjoint, whose union equals that set.

Then, finally, we map the task subsets (partition element) to the resource units, and add hooks in the kernel where this particular resource is allocated or scheduled to constrain the tasks to only using the units to which their task partition element is mapped.

These hooks are usually the 'interesting' part of a resource management patch; one needs to minimize impact on both the kernel source code and on the runtime performance, and for these hooks, that can be a challenge. In particular, what are naturally system wide resource management stuctures cannot be allowed to impose system wide locks on critical resource allocation code paths (and it's usually the most critical resources, such as memory, cpu and network, that we most need to manage in the first place.)

==> This has nothing to do with remapping namespaces as I might use that
    phrase though I cannot claim to be qualified enough to speak on behalf
    of the Generally Established Principles of Computer Science.

I am as qualified as anyone to speak on behalf of cpusets, and I suspect you are not accurately understanding them if you think of them as remapping namespaces.

--

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