Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 22:27:53 GMT

Christoph, responding to Nick:
> > Look at what the patches do. These are not only for hard partitioning
> > of memory per container but also those that share memory (eg. you might
> > want each to share 100MB of memory, up to a max of 80MB for an individual
> > container).
>
> So far I have not been able to find the hooks to the VM. The sharing
> would also work with nodes. Just create a couple of nodes with the sizes you
> want and then put the node with the shared memory into the cpusets for the
> apps sharing them.

Cpusets certainly allows for sharing - in the sense that multiple
tasks can be each be allowed to allocate from the same node (fake
or real.)

However, this is not sharing quite in the sense that Nick describes it.

In cpuset sharing, it is predetermined which pages are allowed to be
allocated by which tasks. Not "how many" pages, but "just which" pages.

Let's say we carve this 100 MB's up into 5 cpusets, of 20 MBs each, and
allow each of our many tasks to allocate from some specified 4 of these
5 cpusets. Then, even if some of those 100 MB's were still free, and
if a task was well below its allowed 80 MB's, the task might still not
be able to use that free memory, if that free memory happened to be in
whatever was the 5th cpuset that it was not allowed to use.

Seth:

    Could your container proposal handle the above example, and let that
    task have some of that memory, up to 80 MB's if available, but not
    more, regardless of what node the free memory was on?

    I presume so.

Another example that highlights this difference - airline overbooking.
If an airline has to preassign every seat, it can't overbook, short of
putting two passengers in the same seat and hoping one is a no show,
which is pretty cut throat. If an airline is willing to bet that
seldom more than 90% of the ticketed passengers will show up, and it
doesn't preassign all seats, it can wait until flight time, see who
shows up, and hand out the seats then. It can preassign some seats,
but it needs some passengers showing up unassigned, free to take what's

left over.

Cpusets preassigns which nodes are allowed a task. If not all the
pages on a node are allocated by one of the tasks it is preassigned to,
those pages "fly empty" -- remain unallocated. This happens regardless
of how overbooked is the memory on other nodes.

If you just want to avoid fisticuffs at the gate between overbooked
passengers, cpusets are enough. If you further want to maximize utilization,
then you need the capacity management of resource groups, or some such.

> > The nodes+cpusets stuff doesn't seem to help with that because you
> > with that because you fundamentally need to track pages on a per
> > container basis otherwise you don't know who's got what.
>
> Hmmm... That gets into issues of knowing how many pages are in use by an
> application and that is fundamentally difficult to do due to pages being
> shared between processes.

Fundamentally difficult or not, it seems to be required for what Nick
describes, and for sure cpusets doesn't do it (track memory usage per
container.)

> > Now if, in practice, it turns out that nobody really needed these
> > features then of course I would prefer the cpuset+nodes approach. My
> > point is that I am not in a position to know who wants what, so I
> > hope people will come out and discuss some of these issues.

I don't know either ;).

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