Subject: Re: [patch00/05]: Containers(V2)- Introduction
Posted by Paul Jackson on Wed, 20 Sep 2006 19:48:13 GMT
View Forum Message <> Reply to Message

Peter wrote:
> > Which comes naturally with cpusets.
>
> How are shared mappings dealt with, are pages charged to the set that
> first faults them in?

Cpusets does not attempt to manage how much memory a task can allocate,
but where it can allocate it. If a task can find an existing page to
share, and avoid the allocation, then it entirely avoids dealing with
cpusets in that case.

Cpusets pays no attention to how often a page is shared. It controls
which tasks can allocate a given free page, based on the node on which
that page resides. If that node is allowed in a tasks 'nodemask_t
mems_allowed' (a task struct field), then the task can allocate
that page, so far as cpusets is concerned.

Cpusets does not care who links to a page, once it is allocated.

Every page is assigned to one specific node, and may only be allocated
by tasks allowed to allocate from that node.

These cpusets can overlap - which so far as memory goes, roughly means
that the various mems_allowed nodemask_t's of different tasks can overlap.

Here's an oddball example configuration that might make this easier to
think about.

> Let's say we have a modest sized NUMA system with an extra bank
> of memory added, in addition to the per-node memory. Let's say
> the extra bank is a huge pile of cheaper (slower) memory, off a
> slower bus.
>
> Normal sized tasks running on one or more of the NUMA nodes just
> get to fight for the CPUs and memory on those nodes allowed them.
>
> Let's say an occassional big memory job is to be allowed to use
> some of the extra cheap memory, and we use the idea of Andrew
> and others to split that memory into fake nodes to manage the
> portion of memory available to specified tasks.
>
> Then one of these big jobs could be in a cpuset that let it use
> one or more of the CPUs and memory on the node it ran on, plus
> some number of the fake nodes on the extra cheap memory.

> Other jobs could be allowed, using cpusets, to use any combination
> of the same or overlapping CPUs or nodes, and/or other disjoint
> CPUs or nodes, fake or real.

Another example, restating some of the above.

> If say some application happened to fault in a libc.so page,
> it would be required to place that page on one of the nodes
> allowed to it. If an other application comes along later and
> ends up wanting shared references to that same page, it could
> certainly do so, regardless of its cpuset settings. It would
> not be allocating a new page for this, so would not encounter
> the cpuset constraints on where it could allocate such a page.

--

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