Subject: Re: Re: Linux-VServer example results for sharing vs. separate mappings ...
Posted by dev on Mon, 26 Mar 2007 09:14:02 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:

[...skip....]
> The problem is memory reclaim. A number of schemes which have been
> proposed require a per-container page reclaim mechanism - basically a
> separate scanner.
>
> This is a huge, huge, huge problem. The present scanner has been under
> development for over a decade and has had tremendous amounts of work and
> testing put into it. And it still has problems. But those problems will
> be gradually addressed.
>
> A per-container recaim scheme really really really wants to reuse all that
> stuff rather than creating a separate, parallel, new scanner which has the
> same robustness requirements, only has a decade less test and development
> done on it. And which permanently doubles our maintenance costs.

So if we merge the global/container scanner code,
"virtualizing" it and using abstract lists, it will be ok for you?

> So how do we reuse our existing scanner? With physical containers. One
> can envisage several schemes:
>
> a) slice the machine into 128 fake NUMA nodes, use each node as the
> basic block of memory allocation, manage the binding between these
> memory hunks and process groups with cpusets.
>
> This is what google are testing, and it works.
>
> b) Create a new memory abstraction, call it the "software zone", which
> is mostly decoupled from the present "hardware zones". Most of the MM
> is reworked to use "software zones". The "software zones" are
> runtime-resizeable, and obtain their pages via some means from the
> hardware zones. A container uses a software zone.
>
> c) Something else, similar to the above. Various schemes can be
> envisaged, it isn't terribly important for this discussion.
>
>
> Let me repeat: this all has a huge upside in that it reuses the existing
> page reclaimation logic. And cpusets. Yes, we do discover glitches, but
> those glitches (such as Christoph's recent discovery of suboptimal
> interaction between cpusets and the global dirty ratio) get addressed, and

> we tend to strengthen the overall MM system as we address them.
>
>
> So what are the downsides? I think mainly the sharing issue:

Honestly, I think there is another huge problem:
Effeciency. Look, when you have a single hardware zone,
kernel is able to do LRU shrinking efficiently when there is a global
memory shortage. People tend to overcommit memory,
so efficient behaviour in situations when none of the containers
are over their limit, but we run out of memory - is important for us.

I imagine how good it will work when we have 200 containers on the node
and each should be scanned and shrinked one by one.
Imho with zones approach it is a fundamental limitation
which can not be overcome in efficient and
fair (regarding to containers) manner.

>>>The issue with pagecache (afaik) is that if we use
>>>containers based on physical pages (an approach which
>>>is much preferred by myself) then we can get in a
>>>situation where a pagecache page is physically in
>>>container A, is not actually used by any process in
>>>container A, but is being releatedly referenced by
>>>processes which are in other containers and hence
>>>unjustly consumes resources in container A.
>>
>>>How significant a problem this is likely to be I do
>>>not know.
>>
>>well, with a little imagination, you can extrapolate
>>that from the data you removed from this email, as one
>>example case would be to start two unified guests one
>>after the other, then shutdown almost everything in
>>the first one, you will end up with the first one being
>>accounted all the 'shared' data used by the second one
>>while the second one will have roughly the resources
>>accounted the first one actually uses ...
>
>
> Right - that sort of thing.
>
> But how much of a problem will it be *in practice*? Probably a lot of
> people just won't notice or care. There will be a few situations where it
> may be a problem, but perhaps we can address those? Forced migration of
> pages from one zone into another is possible. Or change the reclaim code
> so that a page which hasn't been referenced from a process within its
> hardware container is considered unreferenced (so it gets reclaimed). Or a

> manual nuke-all-the-pages knob which system administration tools can use.
> All doable, if we indeed have a demonstrable problem which needs to be
> addressed.
>
> And I do think it's worth trying to address these things, because the
> thought of implementing a brand new memory reclaim mechanism scares the
> pants off me.

I think code is mergeable. It requires some efforts, but imho
it is better way to go. What do you think?

Thanks,
Kirill
