Subject: Re: [RFC][PATCH 2/7] RSS controller core
Posted by mel on Wed, 14 Mar 2007 15:38:24 GMT
View Forum Message <> Reply to Message

On (13/03/07 10:05), Dave Hansen didst pronounce:
> On Tue, 2007-03-13 at 03:48 -0800, Andrew Morton wrote:
> > If we use a physical zone-based containment scheme: fake-numa,
> > variable-sized zones, etc then it all becomes moot. You set up a container
> > which has 1.5GB of physial memory then toss processes into it. As that
> > process set increases in size it will toss out stray pages which shouldn't
> > be there, then it will start reclaiming and swapping out its own pages and
> > eventually it'll get an oom-killing.
>
> I was just reading through the (comprehensive) thread about this from
> last week, so forgive me if I missed some of it. The idea is really
> tempting, precisely because I don't think anyone really wants to have to
> screw with the reclaim logic.
>
> I'm just brain-dumping here, hoping that somebody has already thought
> through some of this stuff. It's not a bitch-fest, I promise. :)
>
> How do we determine what is shared, and goes into the shared zones?

Assuming we had a means of creating a zone that was assigned to a container,
a second zone for shared data between a set of containers. For shared data,
the time the pages are being allocated is at page fault time. At that point,
the faulting VMA is known and you also know if it's MAP_SHARED or not.

The caller allocating the page would select (or create) a zonelist that
is appropriate for the container. For shared mappings, it would be one
zone - the shared zone for the set. For private mappings, it would be
one zone - the shared zone for the set.

For overcommit, the allowable zones for overcommit could be included.
Allowing overcommit opens the possibility for containers to interfere with
each other but I'm guessing that if overcommit is enabled, the administrator
is willing to live with that interference.

This has the awkward possibility of having two "shared" zones for two container
sets and one file that needs sharing. Similarly, there is a possibility for
having a container that has no shared zone and faulted in shared data. In
that case, the page ends up in the first faulting container set and it's
too bad it got "charged" for the page use on behalf of other containers. I'm
not sure there is a sane way of accounting this situation fairly.

I think that it's important to note that once data is shared between containers
at all that they have the potential to interfere with each other (by reclaiming
within the shared zone for example).

> Once we've allocated a page, it's too late because we already picked.

We'd choose the appropriate zonelist before faulting. Once allocated,
the page stays there.

> Do we just assume all page cache is shared? Base it on filesystem,
> mount, ...? Mount seems the most logical to me, that a sysadmin would
> have to set up a container's fs, anyway, and will likely be doing
> special things to shared data, anyway (r/o bind mounts :).
>

I have no strong feelings here. To me, it's "who do I assign this fake
zone to?" I guess you would have at least one zone per container mount
for private data.

> There's a conflict between the resize granularity of the zones, and the
> storage space their lookup consumes. We'd want a container to have a
> limited ability to fill up memory with stuff like the dcache, so we'd
> appear to need to put the dentries inside the software zone. But, that
> gets us to our inability to evict arbitrary dentries.

Stuff like shrinking dentry caches is already pretty course-grained.
Last I looked, we couldn't even shrink within a specific node, let alone
a zone or a specific dentry. This is a separate problem.

> After a while,
> would containers tend to pin an otherwise empty zone into place? We
> could resize it, but what is the cost of keeping zones that can be
> resized down to a small enough size that we don't mind keeping it there?
> We could merge those "orphaned" zones back into the shared zone.

Merging "orphaned" zones back into the "main" zone would seem a sensible
choice.

> Were there any requirements about physical contiguity?

For the lookup to software zone to be efficient, it would be easiest to have
them as MAX_ORDER_NR_PAGES contiguous. This would avoid having to break the
existing assumptions in the buddy allocator about MAX_ORDER_NR_PAGES
always being in the same zone.

> What about minimum
> zone sizes?
>

MAX_ORDER_NR_PAGES would be the minimum zone size.

> If we really do bind a set of processes strongly to a set of memory on a
> set of nodes, then those really do become its home NUMA nodes. If the
> CPUs there get overloaded, running it elsewhere will continue to grab
> pages from the home. Would this basically keep us from ever being able
> to move tasks around a NUMA system?
>

Moving the tasks around would not be easy. It would require a new zone
to be created based on the new NUMA node and all the data migrated. hmm

--
Mel Gorman
Part-time Phd Student   Linux Technology Center
University of Limerick   IBM Dublin Software Lab