Subject: Re: [RFC][PATCH 2/7] RSS controller core
Posted by mel on Tue, 20 Mar 2007 18:57:39 GMT
View Forum Message <> Reply to Message

On (14/03/07 13:42), Dave Hansen didst pronounce:
> On Wed, 2007-03-14 at 15:38 +0000, Mel Gorman wrote:
> > On (13/03/07 10:05), Dave Hansen didst pronounce:
> > > 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.
>
> Well, but MAP_SHARED does not necessarily mean shared outside of the
> container, right?

Well, the data could also be shared outside of the container. I would see
that happening for library text sections for example.

> Somebody wishing to get around resource limits could
> just MAP_SHARED any data they wished to use, and get it into the shared
> area before their initial use, right?
>

They would only be able to impact other containers in a limited sense.
Specifically, if 5 containers have one shared area, then any process in
those 5 containers could exceed their container limits at the expense of
the shared area.

> How do normal read/write()s fit into this?
>

A normal read/write if it's the first reader of a file would get charged to the
container, not to the shared area. It is less likely that a file that is read()
is expected to be shared where as mapping MAP_SHARED is relatively explicit.

> > > 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.
>
> I shouldn't have used dentries as an example. I'm just saying that if

> we end up (or can end up with) with a whole ton of these software zones,
> we might have troubles storing them. I would imagine the issue would
> come immediately from lack of page->flags to address lots of them.
>

That is an immediate problem. There needs to be a way of mapping an arbitrary
page to a software zone. page_zone() as it is could only resolve the "main"
zone. If additional bits were used in page->flags, there would be very hard
limits on the number of containers that can exist.

If zones were physically contiguous to MAX_ORDER, pageblock flags from the
anti-fragmentation could be used to record that a block of pages was in a
container and what the ID is. If non-contiguous software zones were required,
page->zone could be reintroduced for software zones to be used when a page
belongs to a container. It's not ideal the proper way of mapping pages to
software zones might be more obvious then when we'd see where page->zone
was used.

With either approach, the important thing that occured to me is be to be
sure that pages only came from the same hardware zone. For example, do
not mix HIGHMEM pages with DMA pages because it'll fail miserably. For RSS
accounting, this is not much of a restriction but it does have an impact on
keeping kernel allocations within a container on systems with HighMemory.

> > > 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.
>
> OK, but merging wouldn't be possible if they're not physically
> contiguous. I guess this could be worked around by just calling it a
> shared zone, no matter where it is physically.
>

More than likely, yes.

> > > 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.
>
> I was mostly wondering about zones spanning other zones. We _do_

> support this today

In practice, overlapping zones never happen today so a few new bugs
based on assumptions about MAX_ORDER_NR_PAGES being aligned in a zone
may crop up.

>, and it might make quite a bit more merging possible.
>
> > > 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
>
> I know we _try_ to avoid this these days, but I'm not sure how taking it
> away as an option will affect anything.
>

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