Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Alan Cox on Thu, 17 Aug 2006 14:41:45 GMT
View Forum Message <> Reply to Message

Ar Iau, 2006-08-17 am 07:26 -0700, ysgrifennodd Dave Hansen:
> My main thought is that _everybody_ is going to have to live with the
> entry in the 'struct page'. Distros ship one kernel for everybody, and
> the cost will be paid by those not even using any kind of resource
> control or containers.
>
> That said, it sure is simpler to implement, so I'm all for it!

I don't see any good way around that. For the page struct it is a
material issue, for the others its not a big deal providing we avoid
accounting dumb stuff like dentries.

At the VM summit Linus suggested one option for user page allocation
tracking would be to track not per page but by block of pages (say the
2MB chunks) and hand those out per container. That would really need the
defrag work though.

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Andi Kleen on Thu, 17 Aug 2006 15:34:07 GMT
View Forum Message <> Reply to Message

> I don't see any good way around that. For the page struct it is a
> material issue, for the others its not a big deal providing we avoid
> accounting dumb stuff like dentries.
>
> At the VM summit Linus suggested one option for user page allocation
> tracking would be to track not per page but by block of pages (say the
> 2MB chunks) and hand those out per container. That would really need the
> defrag work though.

One could always use a second set of arrays, mirroring mem_map

-Andi

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by Dave Hansen on Thu, 17 Aug 2006 16:37:39 GMT
View Forum Message <> Reply to Message

On Thu, 2006-08-17 at 16:01 +0100, Alan Cox wrote:
> Ar Iau, 2006-08-17 am 07:26 -0700, ysgrifennodd Dave Hansen:
> > My main thought is that _everybody_ is going to have to live with the

> > entry in the 'struct page'. Distros ship one kernel for everybody, and
> > the cost will be paid by those not even using any kind of resource
> > control or containers.
> >
> > That said, it sure is simpler to implement, so I'm all for it!
>
> I don't see any good way around that. For the page struct it is a
> material issue, for the others its not a big deal providing we avoid
> accounting dumb stuff like dentries.

The only way I see around it is using other mechanisms to more loosely attribute ownership of a page to particular containers. I know that my suggestion of traversing the rmap chains at page reclaim time is going appears to be slow compared to the regular reclaim path, but I'm not sure it really matters.

Let's say you have 20 containers sharing 20 pages which are on the LRU, and those pages are evenly distributed so that each container owns one page. Only one of those 20 containers is over its limit. With something that strictly assigns container ownership to one and only one container, you're going to have to walk half of the LRU to find the page.

With a "loose ownership" scheme, you'd hit on the first page, and you'd pay the cost by having to walk halfway through the 20 rmap entries. Admittedly, the rmap walk is much slower than an LRU walk.

-- Dave

Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by dev on Fri, 18 Aug 2006 10:52:28 GMT
View Forum Message <> Reply to Message

Andi Kleen wrote:
>>I don't see any good way around that. For the page struct it is a
>>material issue, for the others its not a big deal providing we avoid
>>accounting dumb stuff like dentries.
>>
>>At the VM summit Linus suggested one option for user page allocation
>>tracking would be to track not per page but by block of pages (say the
>>2MB chunks) and hand those out per container. That would really need the
>>defrag work though.
>
>
> One could always use a second set of arrays, mirroring mem_map
which one do you prefer:
- having a pointer on the struct page?

kernels without resource accounting won't have this.

- having a mirroring mem_map?
  on i686 it is easy, but not sure about sparse mem
  or numa configurations.
  advantage: run-time configurable on boot time.
  disadvantage: much lower performance with accounting.

- address_space/anon_vma can be replaced with some kind of proxy object
  with 2 pointers - address_space and ub. however I don't see how it is
  better then a single ptr on page.

Thanks,
Kirill
