Subject: Re: [ckrm-tech] [RFC][PATCH 5/7] UBC: kernel memory accounting (core)
Posted by dev on Mon, 21 Aug 2006 12:36:24 GMT
View Forum Message <> Reply to Message

Dave Hansen wrote:
> On Fri, 2006-08-18 at 13:31 +0400, Kirill Korotaev wrote:
>
>>they all are troublesome :/
>>user can create lots of vmas, w/o page tables.
>>lots of fdsets, ipcids.
>>These are not reclaimable.
>
>
> In the real world, with the customers to which you've given these
> patches, which of these objects is most likely to be consuming the most
> space? Is there one set of objects that we could work on that would fix
> _most_ of the cases which you have encountered?
the question is not about which one consumes more in "real life".
The question is "which of the resources are allocated on user demand
and should be limited for the environment to be secure and isolated".

>>Also consider the following scenario with reclaimable page tables.
>>e.g. user hit kmemsize limit due to fat page tables.
>>kernel reclaims some of the page tables and frees user kenerl memory.
>>after that user creates some uncreclaimable objects like fdsets or ipcs
>>and then accesses memory with reclaimed page tables.
>>Sooner or later we kill user with SIGSEGV from page fault due to
>>no memory. This is worse then returning ENOMEM from poll() or
>>mmap() where user allocates kernel objects.
>
>
> I think you're claiming that doing reclaim of kernel objects causes much
> more severe and less recoverable errors than does reclaiming of user
> pages.
I also claim that reclaiming some of kernel pages is almost undoable :)
Look, the page can be used by slab objects from 2 different containers.
Why one container should suffer from the second one which needs to be reclaimed?
What to do? separate allocators per container? And if you want to reclaim
say a page with vma you need to replaces tons of pointers all over the objects
in SMP safe manner.

That might generally be true, but I have one example that's
> killing me. (You shouldn't have mentioned mmap ;)
>
> Let's say you have a memory area mapped by one pagetable page, and with
> 1 user page mapped in it. The system is out of memory, and if you
> reclaim either the pagetable page or the user page, you're never going
> to get it back.

>
> So, you pick the user page to reclaim. The user touches it, the memory
> allocation fails, and the process gets killed.
>
> Instead, you reclaim the pagetable page. The user touches their page,
> the memory allocation for the pagetable fails, and the process gets
> killed.
>
> Seems like the same end result to me.
because you suggest the same limit for pagetables and user memory.
Thats why we have separate kmemsize limit for kernel objects and privvmpages for user memory.
privvmpages limit hit will result in -ENOMEM on mmap() system call,
which is memory friendly.

Kirill
