Subject: Re: [RFC][PATCH 0/3] Kernel memory accounting container (v2)
Posted by Pavel Emelianov on Thu, 13 Sep 2007 11:28:49 GMT
View Forum Message <> Reply to Message

Balbir Singh wrote:
> Pavel Emelyanov wrote:
>> Long time ago we decided to start memory control with the
>> user memory container. Now this container in -mm tree and
>> I think we can start with (at least discussion of) the
>> kmem one.
>>
>> Changes from v.1:
>> * fixed Paul's comment about subsystem registration
>> * return ERR_PTR from ->create callback, not NULL
>> * make container-to-object assignment in rcu-safe section
>> * make turning accounting on and off with "1" and "0"
>>
>> ============================================================
>>
>> First of all - why do we need this kind of control. The major
>> "pros" is that kernel memory control protects the system
>> from DoS attacks by processes that live in container. As our
>> experience shows many exploits simply do not work in the
>> container with limited kernel memory.
>>
>> I can split the kernel memory container into 4 parts:
>>
>> 1. kmalloc-ed objects control
>> 2. vmalloc-ed objects control
>> 3. buddy allocated pages control
>> 4. kmem_cache_alloc-ed objects control
>>
>> the control of first tree types of objects has one peculiarity:
>> one need to explicitly point out which allocations he wants to
>> account and this becomes not-configurable and is to be discussed.
>>
>> On the other hands such objects as anon_vma-s, file-s, sighangds,
>> vfsmounts, etc are created by user request always and should
>> always be accounted. Fortunately they are allocated from their
>> own caches and thus the whole kmem cache can be accountable.
>>
>> This is exactly what this patchset does - it adds the ability
>> to account for the total size of kmem-cache-allocated objects
>> from specified kmem caches.
>>
>> This is based on the SLUB allocator, Paul's containers and the
>> resource counters I made for RSS controller and which are in
>> -mm tree already.

>>
>
> Does this mean that the kernel memory container will have a dependency
> on SLUB and it will be disabled for SLAB and SLOB allocators?
> SLAB is going to go away soon anyway and I guess not too many
> people use SLOB.

Right now it is, but I can port it on booth - slab and slob
when slub is accepted.

>> To play with it, one need to mount the container file system
>> with -o kmem and then mark some caches as accountable via
>> /sys/slab/<cache_name>/cache_account.
>>
>> As I have already told kmalloc caches cannot be accounted easily
>> so turning the accounting on for them will fail with -EINVAL.
>> Turning the accounting off is possible only if the cache has
>> no objects. This is done so because turning accounting off
>> implies unaccounting of all the objects in the cache, but due
>> to full-pages in slub are not stored in any lists (usually)
>> this is impossible to do so, however I'm open for discussion
>> of how to make this work.
>>
>
> I remember discussing with you, but I can't remember the rational,
> could you please explain it again.

The pages that are full of objects are not linked in any list
in kmem_cache so we just cannot find them.

>> I know it's maybe too late, since some of you may be preparing
>> for the Summit or LinixConf, but I think that we can go on
>> discussing these on LinuxConf.
>>
>
> The LinuxConf and kernel summit is done now :-)

Oops :) Copy-paste :(

>> The patches are applicable to the latest Morton's tree (that
>> without the RSS controll) with the resource counters patch
>> Andrew committed recently.
>>
>
> This is a bit confusing, it is applicable to 2.6.23-rc4-mm1?

Yup. Copy-paste again... sorry :(

>> I've made some minimal testing for that and the similar code
>> (without the containers interface but with the kmalloc
>> accounting) is already in our 2.6.22 OpenVZ tree, so testing
>> is going on.
>>
>> Thanks,
>> Pavel
>
>

---