
Subject: Re: [PATCH 0/5] Kernel memory accounting container (v5)

Posted by [Paul Menage](#) on Mon, 01 Oct 2007 16:32:29 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi Pavel,

One question about the general design of this - have you tested an approach where rather than tagging each object within the cache with the cgroup that allocated it, you instead have (inside the cache code) a separate cache structure for each cgroup? So the space overheads would go from having a per-object overhead (one pointer per object?) to having a "wastage" overhead (on average half a slab per cgroup). And the time overhead would be the time required to lookup the relevant cache for a cgroup at the start of the allocation operation, and the relevant cache for an object (from its struct page) at deallocation, rather than the time required to update the per-object housekeeping pointer.

Each cache would need to be assigned a unique ID, used as an index into a per-cgroup lookup table of localized caches. (This could almost be regarded as a form of kmem_cache namespace).

It seems to me that this alternative approach would be a lower memory overhead for people who have the kernel memory controller compiled in but aren't using it, or are only using a few groups.

Paul

On 9/25/07, Pavel Emelyanov <xemul@openvz.org> wrote:

> Changes since v.4:

- > * make SLAB_NOTIFY caches mark pages as SlabDebug. That
- > makes the interesting paths simpler (thanks to Christoph);
- > * the change above caused appropriate changes in "turn
- > notifications on" path - all available pages must become
- > SlabDebug and page's freelists must be flushed;
- > * added two more events - "on" and "off" to make kmalloc
- > caches disabling more gracefully;
- > * turning notifications "off" is marked as "TODO". Right
- > now it's hard w/o massive rework of slub.c in respect to
- > full slabs handling.

>

> Changes since v.3:

- > * moved alloc/free notification into slow path and make
- > "notify-able" caches walk this path always;
- > * introduced some optimization for the case, when there's
- > only one listener for SLUB events (saves more that 10%
- > of performance);
- > * ported on 2.6.23-rc6-mm1 tree.

>

> Changes since v.2:

- > * introduced generic notifiers for slub. right now there
- > are only events, needed by accounting, but this set can
- > be extended in the future;
- > * moved the controller core into separate file, so that
- > its extension and/or porting on sLAB will look more
- > logical;
- > * fixed this message :).

>

> Changes since 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".

>

> =====

>

> 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 the kmem one.

>

> 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 control groups and the
> resource counters I made for RSS controller and which are in
> -mm tree already.
>
> 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_notify.
>
> As I have already told kmalloccaches 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 marking of all the slabs in the cache as not-debug, but
> due to full-pages in slub are not stored in any lists (usually)
> this is impossible to do so, however this is in todo list.
>
> Thanks,
> Pavel
>
