

Hi,

This is my current attempt at getting the kmem controller into a mergeable state. IMHO, all the important bits are there, and it shouldn't change *that* much from now on. I am, however, expecting at least a couple more interactions before we sort all the edges out.

This series works for both the slub and the slab. One of my main goals was to make sure that the interfaces we are creating actually makes sense for both allocators.

I did some adaptations to the slab-specific patches, but the bulk of it comes from Suleiman's patches. I did the best to use his patches as-is where possible so to keep authorship information. When not possible, I tried to be fair and quote it in the commit message.

In this series, all existing caches are created per-memcg after its first hit. The main reason is, during discussions in the memory summit we came into agreement that the fragmentation problems that could arise from creating all of them are mitigated by the typically small quantity of caches in the system (order of a few megabytes total for sparsely used caches). The lazy creation from Suleiman is kept, although a bit modified. For instance, I now use a locked scheme instead of `cmpxchg` to make sure cache creation won't fail due to duplicates, which simplifies things by quite a bit.

The slub is a bit more complex than what I came up with in my slub-only series. The reason is we did not need to use the cache-selection logic in the allocator itself - it was done by the cache users. But since now we are lazy creating all caches, this is simply no longer doable.

I am leaving destruction of caches out of the series, although most of the infrastructure for that is here, since we did it in earlier series. This is basically because right now Kame is reworking it for user memcg, and I like the new proposed behavior a lot more. We all seemed to have agreed that reclaim is an interesting problem by itself, and is not included in this already too complicated series. Please note that this is still marked as experimental, so we have so room. A proper shrinker implementation is a hard requirement to take the kmem controller out of the experimental state.

I am also not including documentation, but it should only be a matter of merging what we already wrote in earlier series plus some additions.

Glauber Costa (19):

- slub: don't create a copy of the name string in kmem_cache_create
- slub: always get the cache from its page in kfree
- slab: rename gfpflags to allocflags
- slab: use obj_size field of struct kmem_cache when not debugging
- change defines to an enum
- don't force return value checking in res_counter_charge_nofail
- kmem slab accounting basic infrastructure
- slab/slub: struct memcg_params
- slub: consider a memcg parameter in kmem_create_cache
- slab: pass memcg parameter to kmem_cache_create
- slub: create duplicate cache
- slub: provide kmalloc_no_account
- slab: create duplicate cache
- slab: provide kmalloc_no_account
- kmem controller charge/uncharge infrastructure
- slub: charge allocation to a memcg
- slab: per-memcg accounting of slab caches
- memcg: disable kmem code when not in use.
- slub: create slabinfo file for memcg

Suleiman Souhlal (4):

- memcg: Make it possible to use the stock for more than one page.
- memcg: Reclaim when more than one page needed.
- memcg: Track all the memcg children of a kmem_cache.
- memcg: Per-memcg memory.kmem.slabinfo file.

```
include/linux/memcontrol.h | 87 ++++++
include/linux/res_counter.h | 2 +-
include/linux/slab.h       | 26 ++
include/linux/slab_def.h   | 77 ++++++-
include/linux/slub_def.h   | 36 +++-
init/Kconfig               | 2 +-
mm/memcontrol.c            | 607 ++++++
mm/slab.c                  | 390 ++++++
mm/slub.c                  | 255 ++++++
9 files changed, 1364 insertions(+), 118 deletions(-)
```

--

1.7.7.6