

Hello All,

This is my new take for the memcg kmem accounting. This should merge all of the previous comments from you guys, specially concerning the big churn inside the allocators themselves.

My focus in this new round was to keep the changes in the cache internals to a minimum. To do that, I relied upon two main pillars:

- \* Cristoph's unification series, that allowed me to put most of the changes in a common file. Even then, the changes are not too many, since the overall level of invasiveness was decreased.
- \* Accounting is done directly from the page allocator. This means some pages can fail to be accounted, but that can only happen when the task calling `kmem_cache_alloc` or `kmallocc` is not the same task allocating a new page. This never happens in steady state operation if the tasks are kept in the same memcg. Naturally, if the page ends up being accounted to a memcg that is not limited (such as root memcg), that particular page will simply not be accounted.

The dispatcher code stays (`mem_cgroup_get_kmem_cache`), being the mechanism who guarantees that, during steady state operation, all objects allocated in a page will belong to the same memcg. I consider this a good compromise point between strict and loose accounting here.

I should point out again that most, if not all, of the code in the caches are wrapped in `static_key` areas, meaning they will be completely patched out until the first limit is set. Enabling and disabling of `static_keys` incorporate the last fixes for sock memcg, and should be pretty robust.

[ v4 ]

- \* I decided to leave the file `memory.kmem.slabinfo` out of this series. It can be done a lot better if the caches have a common interface for that, which seems to be work in progress.
- \* Accounting is done directly from the page allocator.
- \* more code moved out of the cache internals.

[ v3 ]

- \* fixed lockdep bugs in slab (ordering of `get_online_cpus()` vs `slab_mutex`)
- \* improved style in slab and slub with less `#ifdefs` in-code
- \* tested and fixed hierarchical accounting (memcg: propagate kmem limiting...)
- \* some more small bug fixes
- \* No longer using `res_counter_charge_nofail` for `GFP_NOFAIL` submissions. Those go to the root memcg directly.
- \* reordered tests in `mem_cgroup_get_kmem_cache` so we exit even earlier for

tasks in root memcg

- \* no more memcg state for slub initialization
- \* do\_tune\_cpucache will always (only after FULL) propagate to children when they exist.
- \* slab itself will destroy the kmem\_cache string for chained caches, so we don't need to bother with consistency between them.
- \* other minor issues

[ v2 ]

- \* memcgs can be properly removed.
- \* We are not charging based on current->mm->owner instead of current
- \* kmem\_large allocations for slub got some fixes, specially for the free case
- \* A cache that is registered can be properly removed (common module case) even if it spans memcg children. Slab had some code for that, now it works well with both
- \* A new mechanism for skipping allocations is proposed (patch posted separately already). Now instead of having kmallocc\_no\_account, we mark a region as non-accountable for memcg.

Glauber Costa (23):

slab: rename gfpflags to allocflags

provide a common place for initcall processing in kmem\_cache

slab: move FULL state transition to an initcall

Wipe out CFLGS\_OFF\_SLAB from flags during initial slab creation

memcg: Always free struct memcg through schedule\_work()

memcg: change defines to an enum

kmem slab accounting basic infrastructure

slab/slub: struct memcg\_params

consider a memcg parameter in kmem\_create\_cache

sl[au]b: always get the cache from its page in kfree

Add a \_\_GFP\_SLABMEMCG flag

memcg: kmem controller dispatch infrastructure

allow enable\_cpu\_cache to use preset values for its tunables

don't do \_\_ClearPageSlab before freeing slab page.

skip memcg kmem allocations in specified code regions

mm: Allocate kernel pages to the right memcg

memcg: disable kmem code when not in use.

memcg: destroy memcg caches

Track all the memcg children of a kmem\_cache.

slab: slab-specific propagation changes.

memcg: propagate kmem limiting information to children

memcg/slub: shrink dead caches

Documentation: add documentation for slab tracker for memcg

Suleiman Souhlal (2):

memcg: Make it possible to use the stock for more than one page.

memcg: Reclaim when more than one page needed.

```

Documentation/cgroups/memory.txt | 33 ++
include/linux/gfp.h                | 8 +-
include/linux/memcontrol.h         | 100 ++++++
include/linux/page-flags.h         | 2 +-
include/linux/sched.h              | 1 +
include/linux/slab.h               | 25 ++
include/linux/slab_def.h           | 6 +-
include/linux/slub_def.h           | 29 +-
init/Kconfig                       | 2 +-
mm/memcontrol.c                   | 874 ++++++-----
mm/page_alloc.c                   | 16 +-
mm/slab.c                         | 94 +++-
mm/slab.h                         | 57 +-
mm/slab_common.c                  | 72 +++-
mm/slob.c                         | 6 +-
mm/slub.c                         | 32 +-
16 files changed, 1272 insertions(+), 85 deletions(-)

```

--  
1.7.10.2

---