Google

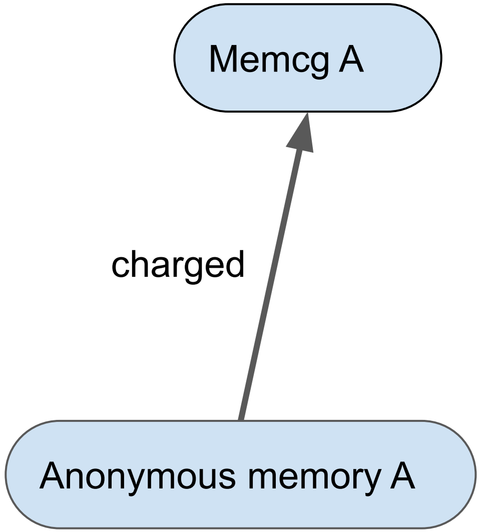
Reducing Zombie Memcgs

LSF/MM/BPF 2023

T.J. Mercier (tjmercier@google.com)
Yosry Ahmed (yosryahmed@google.com)
Chris Li (chriscli@google.com)

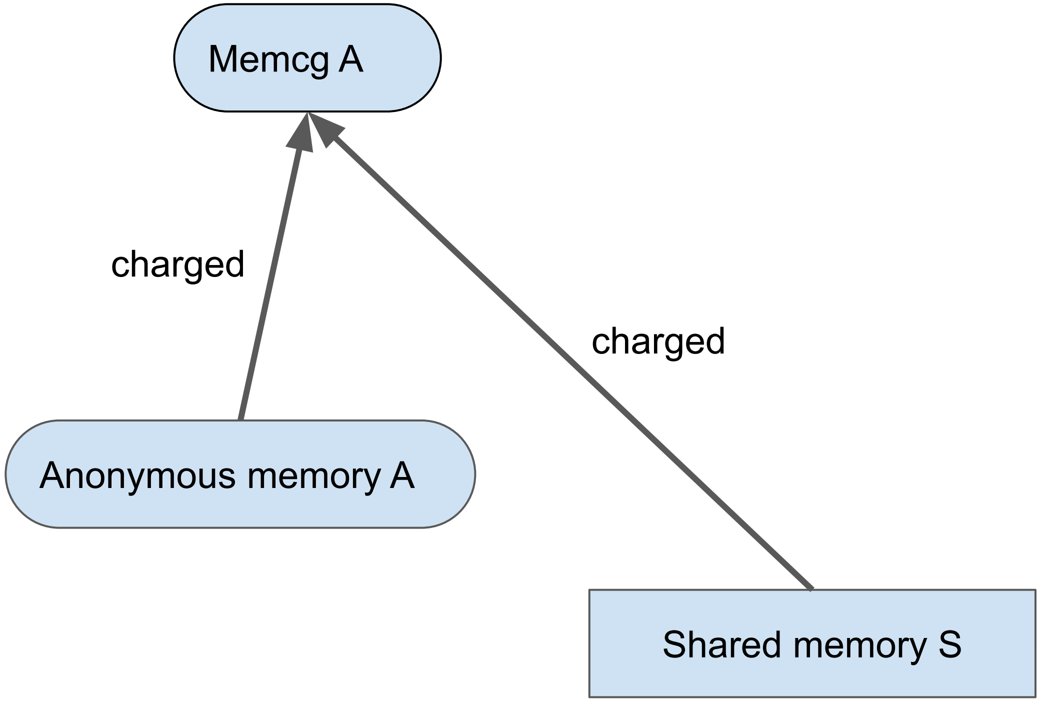
Google

Memcg A with Anonymous Memory



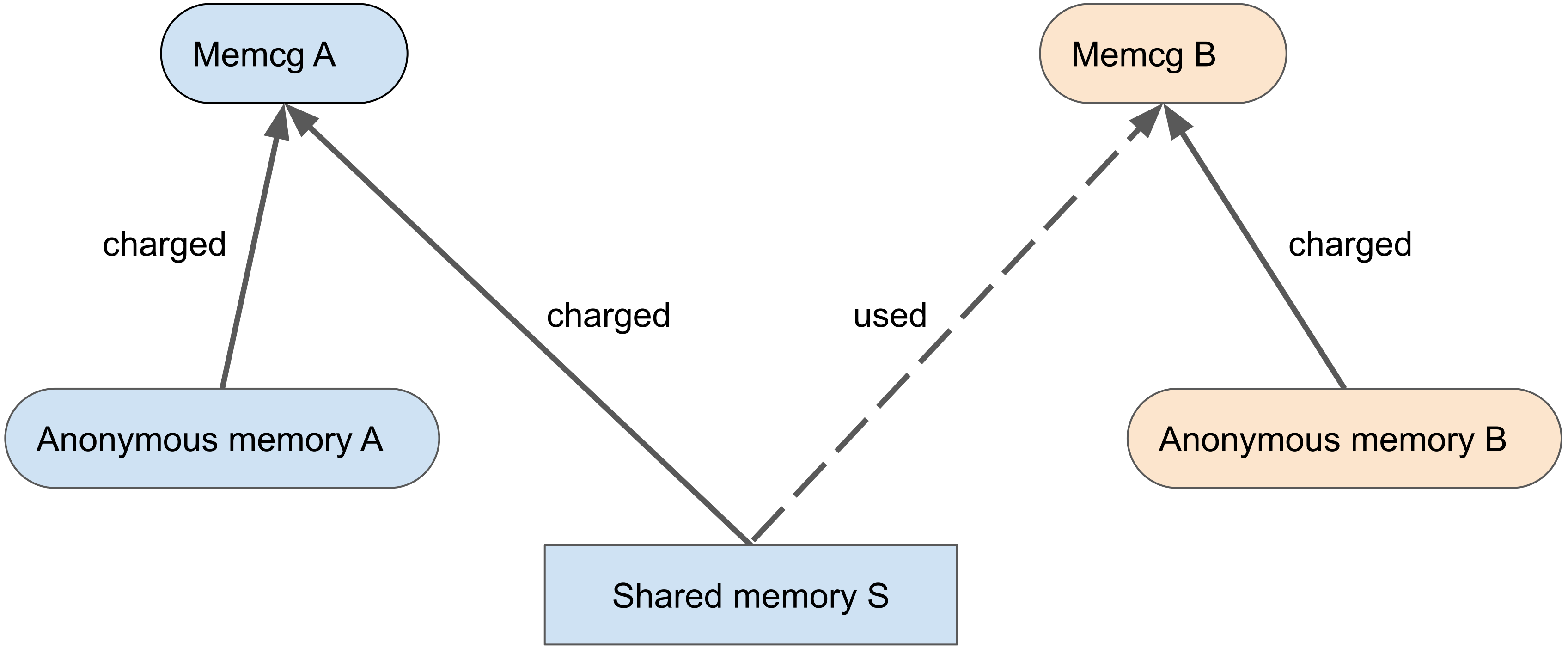
Google

Memcg A Allocates Shared Memory S



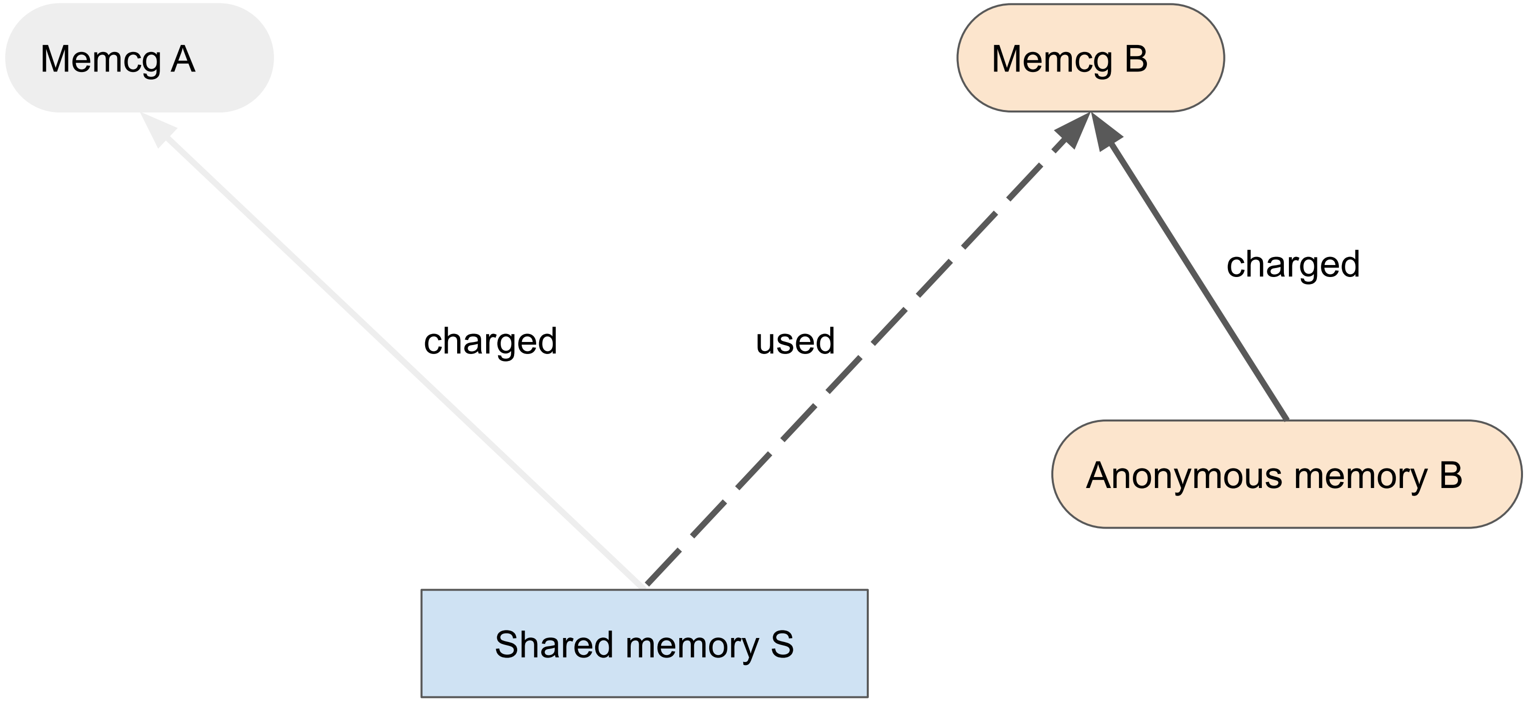
Google

Memcgs A and B Share Memory S



Google

Memcg A Removed (Zombie)



Google

The Problem

- Zombie cgroups can accumulate.
- A lot. (thousands)
- This consumes kernel memory (per-cpu in struct mem_cgroup), and makes kernel operations (e.g. reclaim) less efficient.

Google

Non-Fixes for The Problem

- Manual reclaim (memory.reclaim)
  - Doesn’t work for unreclaimable memory (any shared/pinned memory that’s still in use).
  - Can result in swapped pages, which *still* keeps the cgroup around.
  - Can only attempt once before rmdir, and that can fail to reclaim everything.

- Reparent during offlining (move the charge up to the parent)
  - The parent *also* has no actual ownership of the memory.
  - Hides/mixes zombie memory with parent’s.
  - Affects pgscan/pgsteal for the new parent.
  - Can happen multiple times until the root cgroup gets stuck with it.
  - Non-deterministic memory use (same job, different memory use).

Shared Resources (The Fundamental Problem)

- Pages have a single owning memcg (stored directly or indirectly in memcg_data).

- If a page is shared between cgroups for any reason, the charge can outlive the owning cgroup (Keeping the zombie ~~alive~~ undead).

- For pages that are not shared, reclaim should eventually clean up zombies (but it would be nice to accelerate this).

Google

Maybe-Fixes for The Problem

- Short Term
  - Recharge during offlining (move the charge to some other cgroup).
    - Which other cgroup?

- Long Term
  - Add first class support for tracking shared memory resources.

Short Term? Memory Recharging

- When a memcg is offlined, recharge pages charged to it to other memcgs.
- What types of pages do we have?
  - Kernel pages → already being reparented.
  - **Mapped LRU pages.**
  - Unmapped LRU pages.
    - **Page cache pages.**
    - Anonymous pages in the swap cache (ignore for now [1]).
  - Anything else?

Google

Short Term? Memory Recharging

- What toolkit do we have?
  - Evict pages
    - Simple, but aggressive. [Un|Re]charges for file-backed pages. Reparents swap-backed pages. Doesn’t help for pinned pages.
  - Direct recharge to a mapper
    - Memory recharged to the rightful owner – but can be disruptive (nondeterministic charges, potential OOM kills).
  - Deferred recharge (Two-step recharge)
    - Recharge to the parent, then to the rightful owner on next access or mapping.
    - Complicated, additional work in data access path.

Google

Short Term? Memory Recharging

- Proposed Workflow → async LRU walk for offlined memcg:
  - If the page is unmapped
    - If the page is file-backed
      - Evict
    - If the page is swap-backed
      - Deferred recharge to the next accessor (?)
  - If the page is mapped
      - Recharge to a mapper (direct or deferred*)
- What about already swapped memory?
  - Idea: periodically walk swap_cgroups and reparent those charged to offline memcgs*

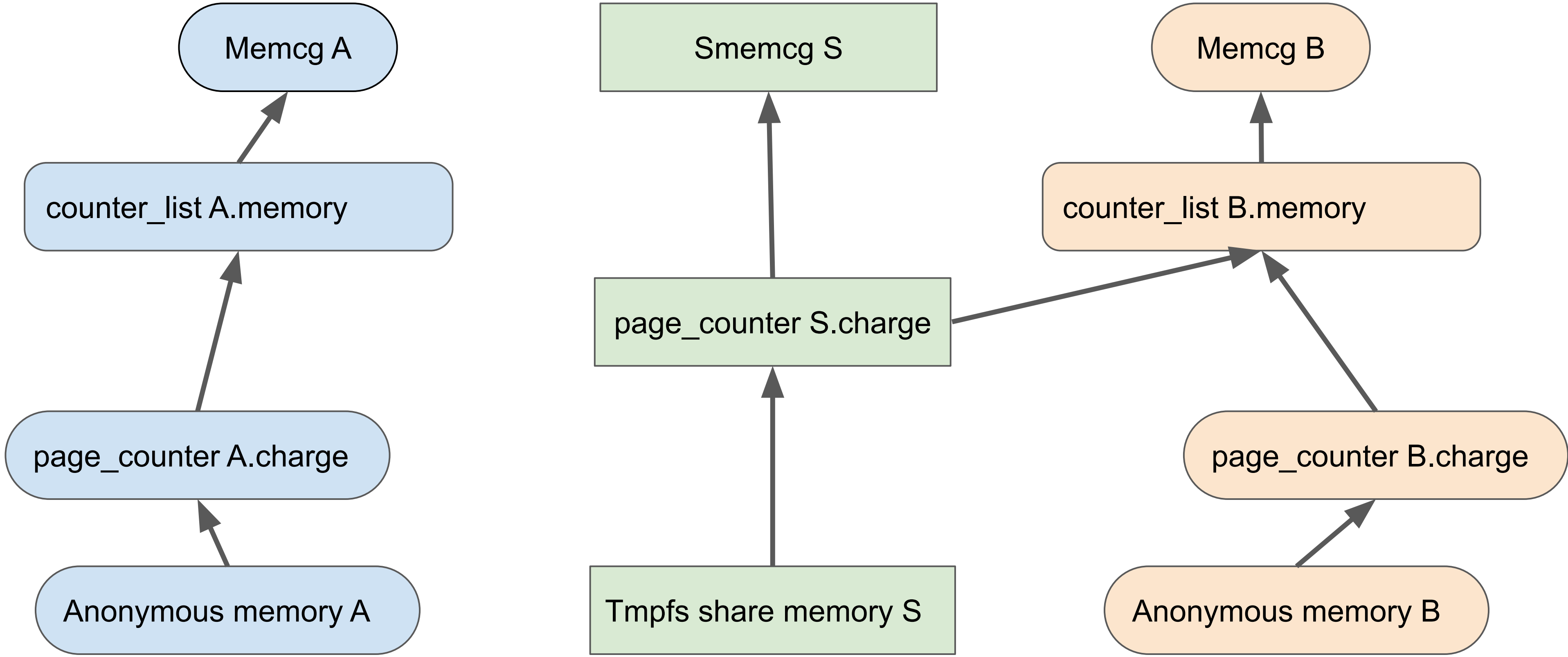
Google

Long Term: Properly Track the Shared Relationship

- Shared Memory Controller (Chris Li)
  - Shared memory owned by the smemcg
  - The shared resource lifecycle is not tied to the lifecycle of any one memory cgroup
  - Track the shared resource usage separately with borrow counter.
  - No charge movement
  - No zombie memcg if all shared resource account in smemcg

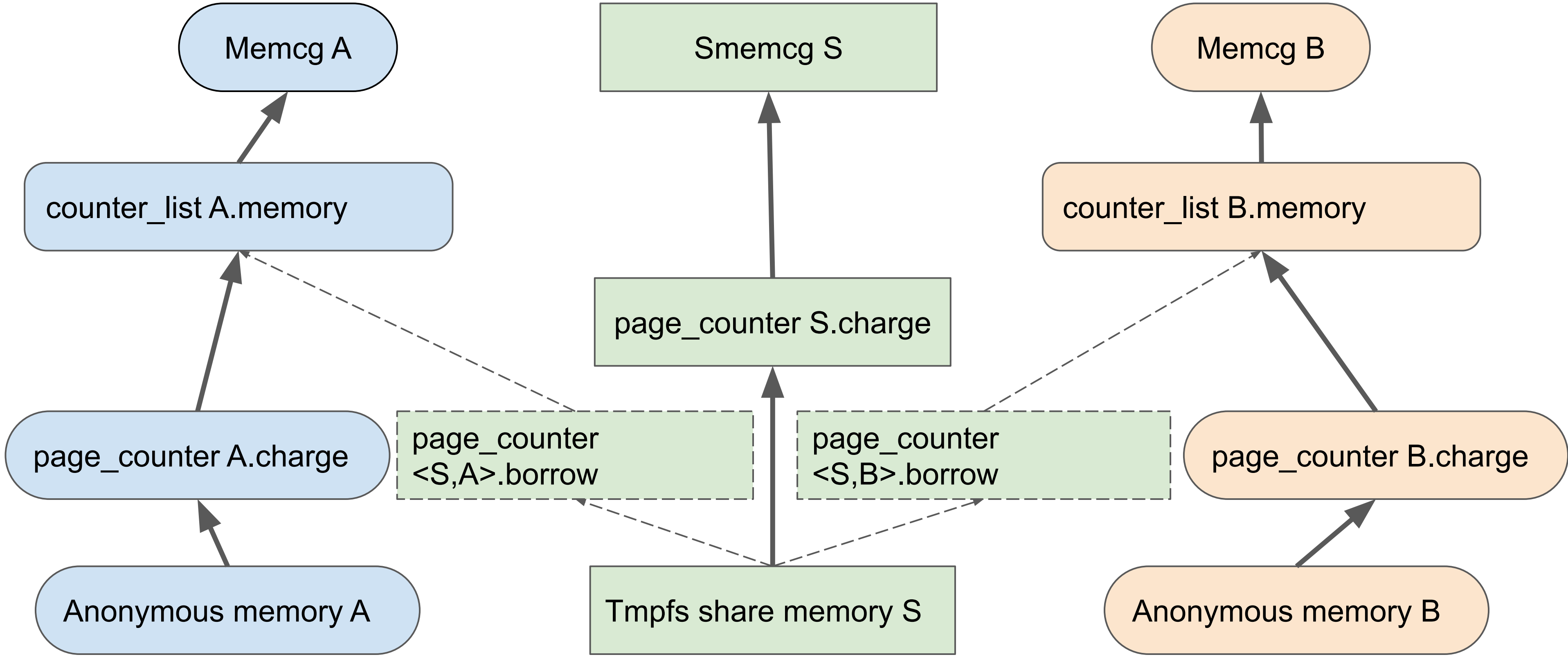
Google

Memcgs A and B Share Memory S (set membership)



Google

Memcgs A and B Share Memory S (charge tracking)



Discussion