Contribution ID: **403** Type: **not specified**

Taming Zombie Cgroups: Should Reparenting Be Simple or Smart?

Thursday 11 December 2025 18:00 (15 minutes)

The “zombie memory cgroup” problem is a long-standing issue in the Linux Kernel. It occurs when a memory cgroup is destroyed by users, but kernel metadata cannot be freed because its Least Recently Used (LRU) pages, particularly shared file pages, remain charged to it. These pages can outlive the cgroup that originally owned them, acting as a permanent pin. In environments where cgroups are frequently created and destroyed, this leads to an accumulation of zombie cgroups that unnecessarily consume kernel memory over the long term.

This session presents the community-driven solution currently being upstreamed to resolve this problem. We will provide an overview of the approach, which transfers ownership of LRU pages from the memory cgroup to the object cgroup, which is an infrastructure designed to hold charges without pinning the original cgroup. By using the object cgroup API, LRU pages and their charges can be reparented to the parent cgroup when a memory cgroup is offlined. We will focus on the correctness of the core mechanism, which guarantees binding stability between a folio and its memory cgroup while critical locks (like the lruvec lock) are held, ensuring safe and race-free reparenting.

A central challenge we will discuss is the trade-off between simplicity and sophistication in the reparenting process. Merging the LRU lists of a child cgroup into its parent is not trivial, as it can temporarily disrupt the generational ordering crucial for effective page reclaim. We will analyze the open question: should reparenting be made more intelligent to preserve LRU ordering and thus reclaim efficiency, or is a simpler, more predictable merge operation preferable for correctness and maintainability? This discussion is especially critical for supporting Multi-Gen LRU (MGLRU), where generations must be adjusted during reparenting.

This session will outline the solution’s architecture, validate its correctness, and engage the community in optimizing the final reparenting step for real-world performance.

Primary author: YOO, Harry (Oracle)

Co-author: BABULAL, Kamalesh (Oracle)

Presenters: YOO, Harry (Oracle); BABULAL, Kamalesh (Oracle)

Session Classification: Kernel Memory Management MC

Track Classification: Kernel Memory Management MC