Linux
Plumbers
Conference 2022

>> Dublin, Ireland / September 12-14, 2022

Scalability solutions for the mmap_lock

- Maple Trees
- Per-VMA locks
- SPF

Quick recap: the problem

mm_struct contains mmap_lock rw_semaphore which

- Protects the VMA list / rbtree
- Prevents VMA from being freed while in use by other threads
- Protects many other fields in mm_struct

Issue: mmap_lock is a coarse-grained lock that creates contention. Examples:

- Android: multi-threaded application launch
- Google Fibers: threads creating a set of VMAs
- smaps/maps polling

https://lwn.net/Articles/591978/, https://lwn.net/Articles/787629/

Maple Tree Review

Cache Efficiencies

- Reduces the mm_struct (1000 -> 992)*
- Reduces vm_area_struct size (192 -> 144)*
  - *: depends on config options, same config was used in comparison

Data Structure Reduction

- Removes VMA doubly linked list
- Removes VMA augmented rbtree
- Removes vmacache

Supports RCU

Per-VMA locks: The idea

Each VMA gets a rw_semaphore lock.

VMA modifier takes per-VMA write lock for:

- VMA unmapping, remapping, copying, merging, splitting, resizing
- VMA flags or protection changes

Page fault handler finds VMA containing the faulting address under RCU protection and tries to take per-VMA read lock. On failure it falls back to mmap_lock.

VMAs are freed after RCU grace period.

Per-VMA locks: Encountered issues and Results

Multiple VMAs might need to be locked (vma_merge/vma_split) - adds complexity.

- Addressed by marking VMAs as locked and unlocking in bulk

Some paths in fork and exit_mm should take all per-VMA locks instead of one mmap_lock.

- Regressions in exit path are fixed by freeing vm_area_structs in bulk

Results: Improves performance of PFT benchmarks and Android launch times (~75% of the improvement that we saw with SPF).

RFC link: https://lore.kernel.org/all/20220829212531.3184856-1-surenb@google.com

Per-VMA locks: The anatomy

To avoid tracking locked VMAs and to be able to unlock them in bulk two sequence counters are introduced:

vm_area_struct.vm_lock_seq

mm_struct.mm_lock_seq

Main functions:

VMA is write locked => (vm_area_struct.vm_lock_seq == mm_struct.mm_lock_seq)

VMA is write unlocked => (vm_area_struct.vm_lock_seq != mm_struct.mm_lock_seq)

Lock VMA => (vm_area_struct.vm_lock_seq = mm_struct.mm_lock_seq)

Unlock all VMAs => (mm_struct.mm_lock_seq++)