ORACLE

Faster & Fewer Page Faults

Kernel Recipes

Matthew Wilcox
Technical Advisor
Oracle Linux Development
2023-09-27

Safe harbor statement

Four projects

- Maple Tree
- Per-VMA Locking
- Large Folios
- New PTE manipulation interfaces

https://www.cs.virginia.edu/~robins/YouAndYourResearch.html

Linked Lists are Immoral

- Your CPU is attempting to extract parallelism from your sequential code
- My 2.8GHz laptop CPU is able to issue 6 insn/clock
  30 insn/5-cycle L1 cache hit
  70 insn/14-cycle L2 cache hit
  200 insn/40-cycle (14ns) L3 cache hit
  1680 insn/100ns L3 cache miss
- Linked lists bottleneck on fetching the next entry in the list
- Arrays can be prefetched
- Walking a million-entry array is 12x faster than a million-entry list on my laptop

Anatomy of a page fault

- Look up VMA (Virtual Memory Area) for virtual address
- Walk down the page tables
- If VMA is anonymous, allocate a page
- Otherwise, call VMA fault handler
  - Fault handler may return a page or populate page table directly
- If page provided, insert entry into page table

Four projects

- **Maple Tree**
- Per-VMA Locking
- Large Folios
- New PTE manipulation interfaces

Looking up a VMA

- VMAs were originally stored on a singly-linked list in 0.98 (1992)
- An AVL tree was added in 1.1.83 (1995)
- A Red-Black tree replaced the AVL tree in 2.4.9.11 (2001)
- A Maple Tree replaced the linked list & Red-Black tree in 6.1 (2022)

Maple Tree

- In-memory, RCU-safe B-tree for non-overlapping ranges
- Average branching factor of eight creates shallower trees (faster lookups)
- Modifications allocate memory (slower modifications)
- Applications typically have between 20 VMAs (`cat`) and 1000 (Mozilla)
  - Can be millions in pathological cases (ElectricFence)
- RCU safety guarantees that a VMA which was present before the RCU lock was taken, and is still present after the RCU lock is released will be found.

Four projects

- Maple Tree
- **Per-VMA Locking**
- Large Folios
- New PTE manipulation interfaces

VMA tree locking

- Protected by a semaphore from 2.0.19 (1996)
- Changed to a read-write semaphore from 2.4.2.5 (2001)
- Added per-VMA read-write semaphores in 6.4 (2023)

Per-VMA locking lookup

- Take RCU read lock to prevent Maple tree nodes and VMAs from being freed
- Load VMA from Maple tree
- Read-trylock the per-VMA lock
  - If write-locked, a writer is modifying this VMA.
- If MM seqcount is equal to VMA seqcount, VMA is locked
  - This allows a writer to unlock all locked VMAs just by updating mm seqcount
- Drop RCU read lock; we will not look at the Maple Tree, and the VMA cannot be freed

Support for per-VMA locking

- Anonymous VMAs handled from 6.4 on arm64, powerpc, s390, x86; 6.5 on riscv
- Swap and Userfaultfd support in 6.6
- In-core page cache VMAs support in 6.6
- DAX support in 6.6
- Page cache faults that need reads in 6.7?
- COW faults of page cache VMAs in 6.7?
- More support is possible, both architectures and types of memory
  - Device drivers may rely on mmap_sem synchronisation
  - HugeTLB faults have not yet been converted

Four projects

- Maple Tree
- Per-VMA Locking
- **Large Folios**
- New PTE manipulation interfaces

Large Folios

- XFS files can be buffered in larger chunks than PAGE_SIZE since 5.17 (2022)
  - AFS since 6.0, EROFS since 6.2
- Large folios can be created on `write()` since 6.6
- Support for other filesystems & anonymous memory is in progress

Four projects

- Maple Tree
- Per-VMA Locking
- Large Folios
- **New PTE manipulation interfaces**

New PTE manipulation interfaces

- `set_pte_at()` could only insert a single Page Table Entry
- `set_ptes()` can insert *n* consecutive Page Table Entries pointing to contiguous pages
- `flush_dcache_folio()` flushes the entire folio from the data cache
- `flush_icache_pages()` flushes *n* consecutive pages from the instruction cache
- `update_mmu_cache_range()` acts on *n* consecutive pages
  - Also tells the architecture which page was actually requested

Projects I Don’t Have Time To Talk About

- Large Anonymous Folios
- Removing `→writepage()`
- Removing `→launder_folio()`
- Shrinking `struct page`
- Batched folio freeing
- `bdev_getblk()`
- ext2 directory handling
- `folio_end_read()`
- `mrlock` removal
- Converting buffer_heads to use folios
- Lockless page faults
- Removing `GFP_NOFS`
- `struct ptdesc`
- A better approach to the LRU list
- Block size > PAGE_SIZE
- Removing `arch_make_page_accessible()`
- Why kernel-doc is not my favourite
- Rewriting the swap subsystem
- Removing `__GFP_COMP`
- What does folio mapcount mean anyway?
- Replacing the XArray radix tree with the maple tree
- Converting HugeTLBfs to folios
- Making HugeTLBfs less special
- mshare
- Improving readahead for modern storage
- Support folios larger than PMD size

Thanks

- Andrew Morton
- Darrick Wong
- Dave Chinner
- David Howells
- David Hildenbrand
- David Rientjes
- Davidlohr Bueso
- Greg Marsden
- Jan Kara
- Johannes Weiner
- Jon Corbet
- Kiryl Shutsemau
- Laurent Dufour
- Liam Howlett
- Michal Hocko
- Michel Lespinasse
- Mike Kravetz
- Mike Rapoport
- Paul McKenney
- Ryan Roberts
- Song Liu
- Suren Baghdasaryan
- Vlastimil Babka
- Yin Fengwei

ORACLE