

ORACLE

# Memory folios in the Linux Kernel

---

**Matthew Wilcox**

Technical Advisor

Linux Kernel Development

2022-06-23

## Safe harbor statement

---

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

# Memory is managed in pages

---

- Oracle Server X8-8 has 6TiB of memory
  - 1.5 billion 4KiB pages
- Anonymous & Page Cache memory tracked on a per-NUMA-node LRU list
  - 192 million pages per node
- A long LRU list is inefficient
  - LRU lock contention
  - Lots of cache misses with a spinlock held

# Potential solutions

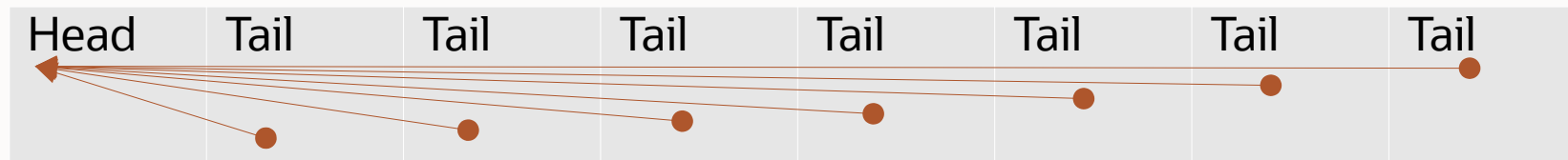
---

- **Manage all memory in large pages**
  - For example 64KiB “superpages”
  - Wastes some memory
  - Hard to maintain user ABI which assumes it can `mmap()` on 4KiB boundary
- **Manage just Page Cache memory in large pages**
  - The `PAGE_CACHE_SIZE` macros were removed in 2016 because they were confusing
  - Similar problems to previous solution
- **Adaptive use of large pages**
  - How to decide when large pages would be beneficial?

# Compound Pages

- Linux can allocate pages in  $2^n$  sizes
  - $n$  is the *order* of the page
- First page is referred to as the head page
- All other pages are tail pages
- Operations on tail pages usually redirect to the head page (eg `lock_page()`)

Order-3 compound page:



# Interfaces

- If you pass a page to a function, does it ...
  - Operate on PAGE\_SIZE bytes
  - Operate on the entire allocation
  - Operate on the entire allocation if it's a head page, otherwise on PAGE\_SIZE bytes
  - BUG\_ON() if it's a tail page
  - Do something else? ...

“Ambiguity is tactically useful, strategically disastrous” – Garry Kasparov

- Some functions use the parameter name to identify what kind of page they expect
  - `int head_compound_mapcount(struct page *head)`
- Cannot refer to `page->mapping` of a tail page
- Can call `lock_page()` on a tail page

# Memory Folios

---

- New type:

```
struct folio { ... contents of struct page here ... };
```

- If you need to refer to a precise page, use struct page

```
struct vm_fault { .page = ... };
```

- If you need to refer to the memory allocation, use struct folio

```
folio_start_writeback(folio)
```

# What's upstream?

---

- v5.14
  - Preparatory work only
- v5.16
  - Initial folio APIs
  - netfs
- v5.17
  - Block
  - Page cache
  - iomap
  - XFS
  - slab
- v5.18
  - Large folios
  - Truncation
  - aops->dirty\_folio()
  - get\_user\_page()
  - rmap\_walk()
- v5.19
  - Most address\_space operations
  - Parts of vmscan, swap, shmem and migrate
- v5.20
  - AFS?
  - More address\_space operations
  - Split movable pages from aops
  - More of vmscan, swap, shmem and migrate

# When to allocate large folios

---

- Hints from userspace?
  - Unreliable in the past
- Filesystem could decide?
  - Hints may be useful, but filesystem doesn't have enough information
- Page Cache readahead already decides how many pages to read ahead
  - Now it also decides how large a folio to allocate
  - Current algorithm very aggressive for testing
- Writing to an uncached part of a file still uses order-0 pages
- Page fault will allocate PMD-order pages if MADV\_HUGEPAGE is set

# Performance results

- 6% reduction in kernel time running kernbench
- Throughput improvements with vm-scalability (Intel Kernel Test Robot):
  - 241% for mmap-pread-seq
  - 65% for mmap-pread-seq-mt
  - 24% for migrate
  - 45% for lru-file-mmap-read

<https://lore.kernel.org/all/20220418144234.GD25584@xsang-OptiPlex-9020/>

- Benefits:
  - Shorter LRU lists, shorter lock hold times, fewer cache misses
  - Memory fragmentation is reduced
  - Larger I/Os
  - Opportunity to get rid of 16KiB / 64KiB PAGE\_SIZE configurations
  - Removal of calls to `compound_head()`

# Future work

---

- Split other struct page users out
  - Network DMA pools
  - zsmalloc
  - Page tables
  - Zone Device
- Dynamically allocate memory descriptors
  - Tricky for slab, but not impossible
- Continue converting mm to folios
- Create large folios for anon memory
- Support other folio sizes in shmem/tmpfs
  - Needs swap to support other folio sizes
- Convert filesystems from buffer heads to iomap
- Continue converting filesystems to folios
  - May or may not include support for large folios
- Improve O\_SYNC writes
  - Write-through instead of writeback
- Count folio mappings less accurately
  - Do not account individual pages
- Map an entire folio on page fault
- Allocate large folios for anonymous memory
  - How should we determine the correct size to allocate?
- Convert to get\_user\_phyr()

# Contributors

---

- Vlastimil Babka (SuSE)
- Dave Chinner (Red Hat)
- Hugh Dickins (Google)
- Brian Foster (Red Hat)
- Christoph Hellwig (independent)
- David Howells (Red Hat)
- John Hubbard (Nvidia)
- Mike Kravetz (Oracle)
- Bill Kucharski (Oracle)
- Song Liu (Facebook)
- Andrew Morton (Google)
- Kent Overstreet (independent)
- Kirill Shutemov (Intel)
- Darrick Wong (Oracle)
- Zi Yan (Nvidia)



ORACLE