

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

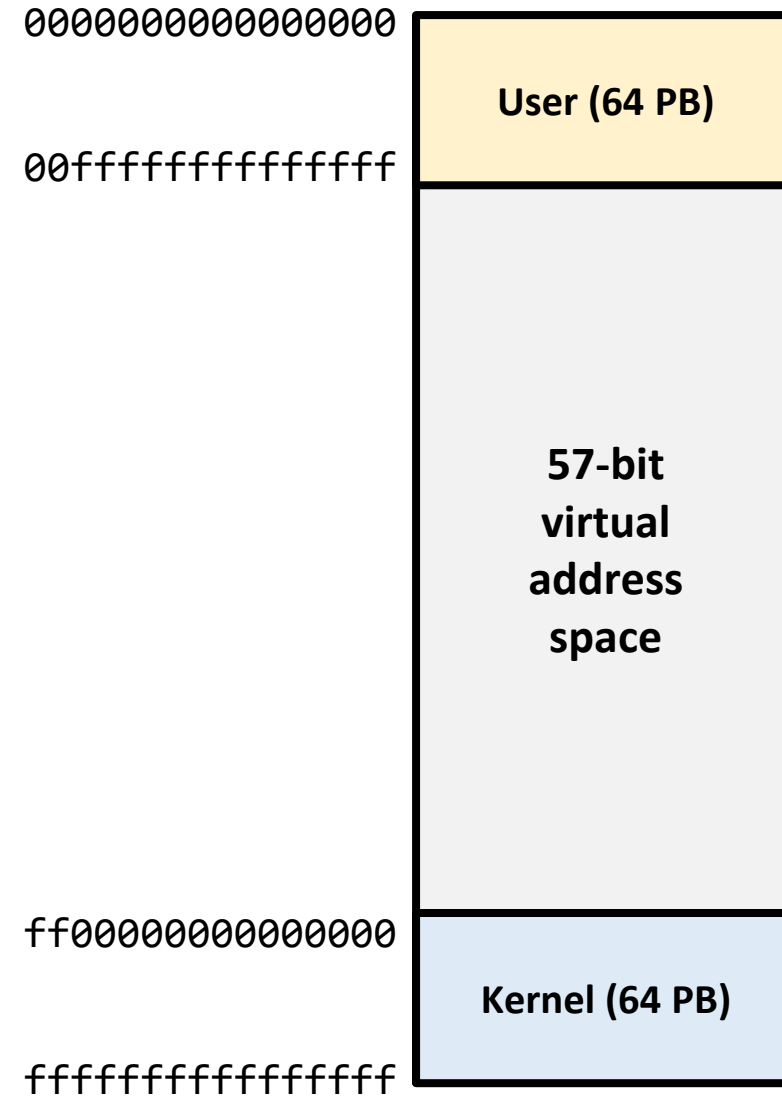
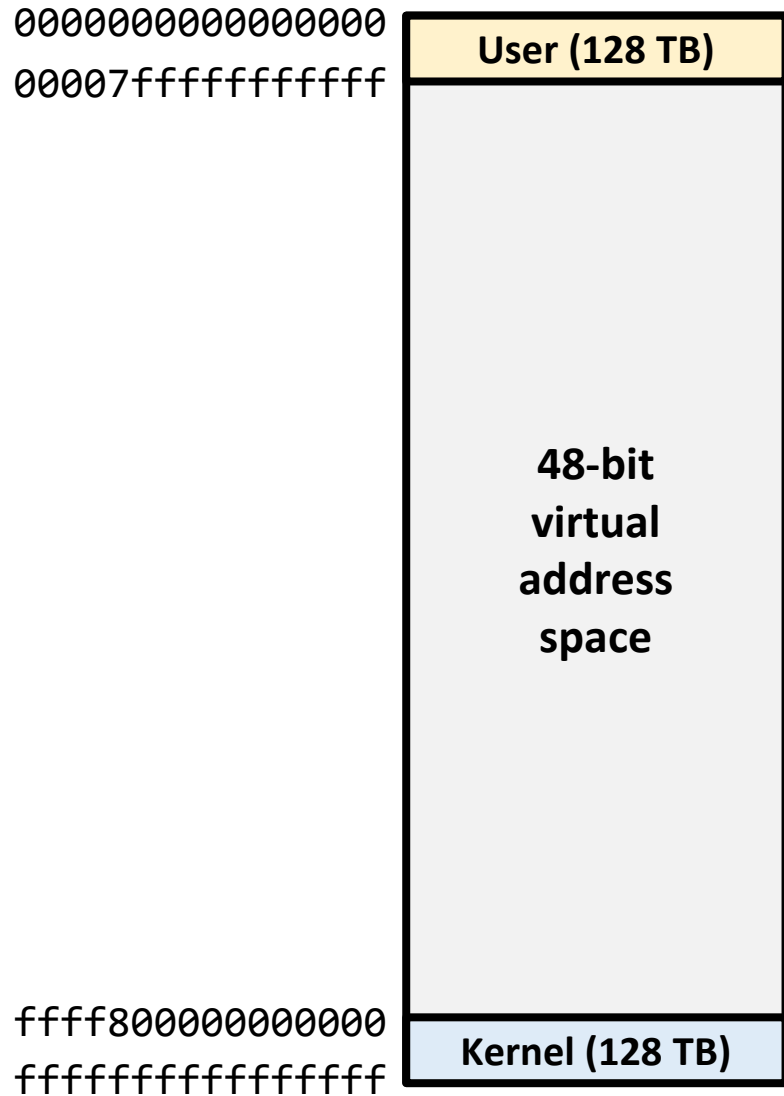
Spring 2026

Linux

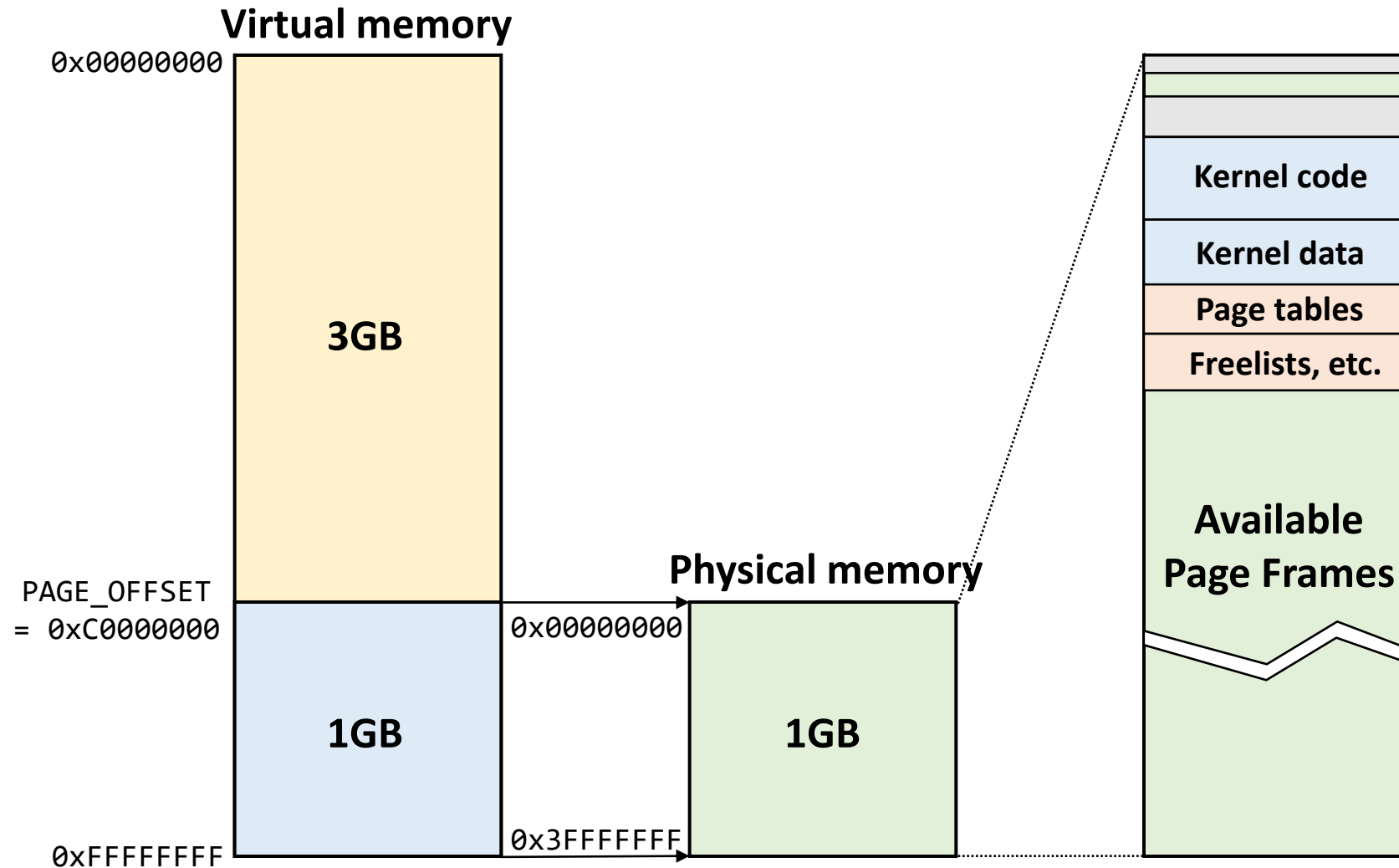
Virtual Memory



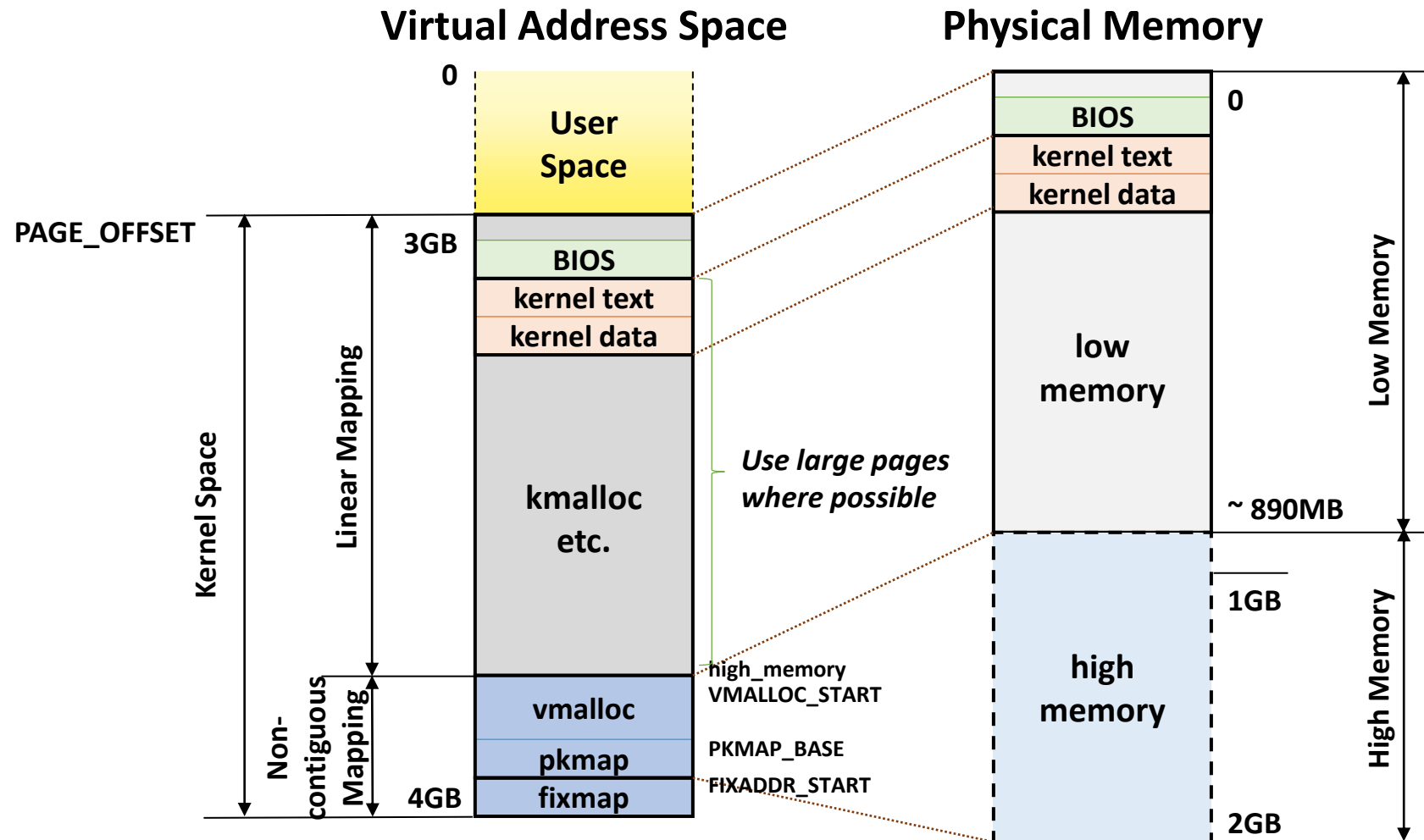
Virtual Address Map



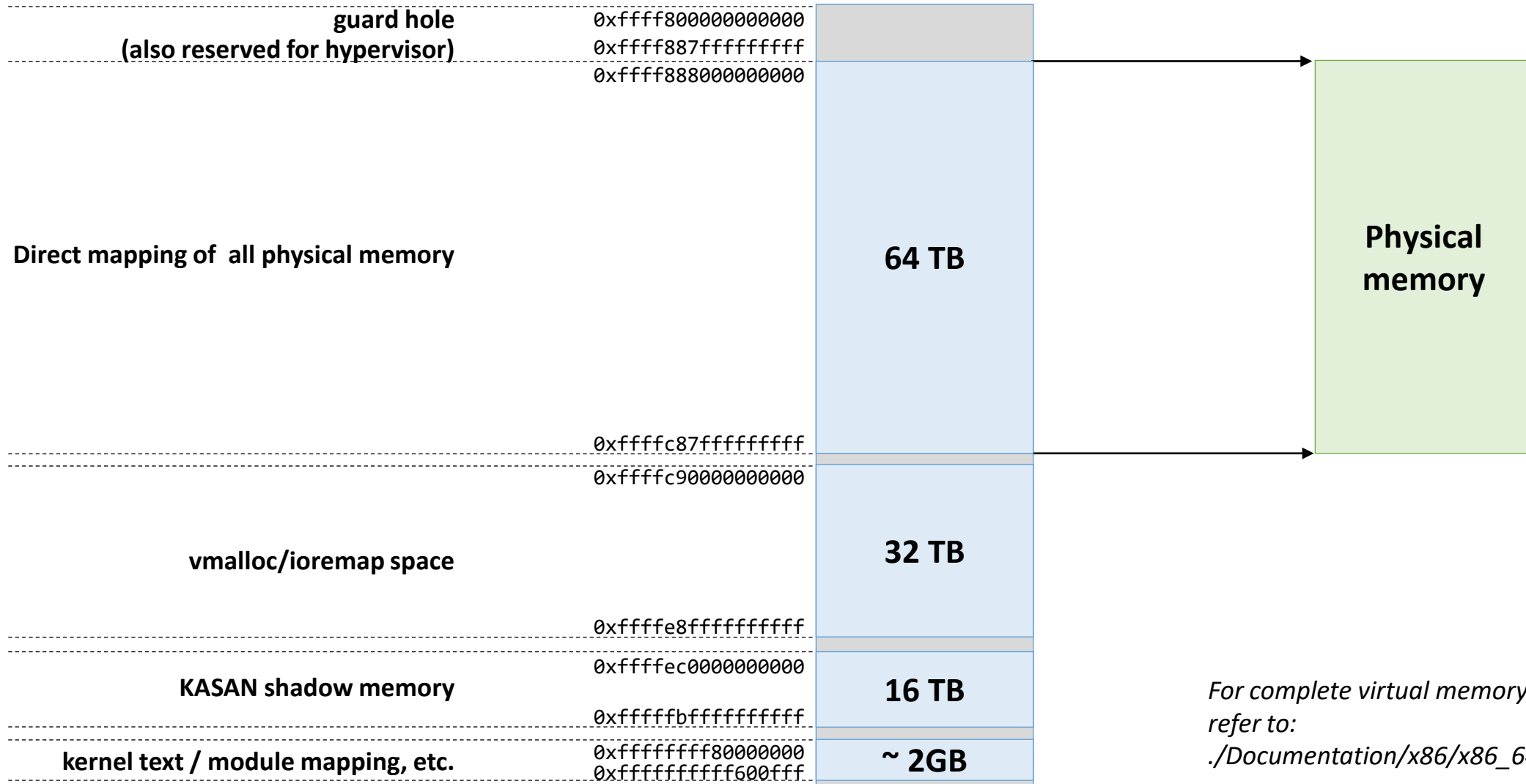
Kernel Address Space (i386, < 1GB)



Kernel Address Space (i386)



Kernel Address Space (x86_64, 48-bit)

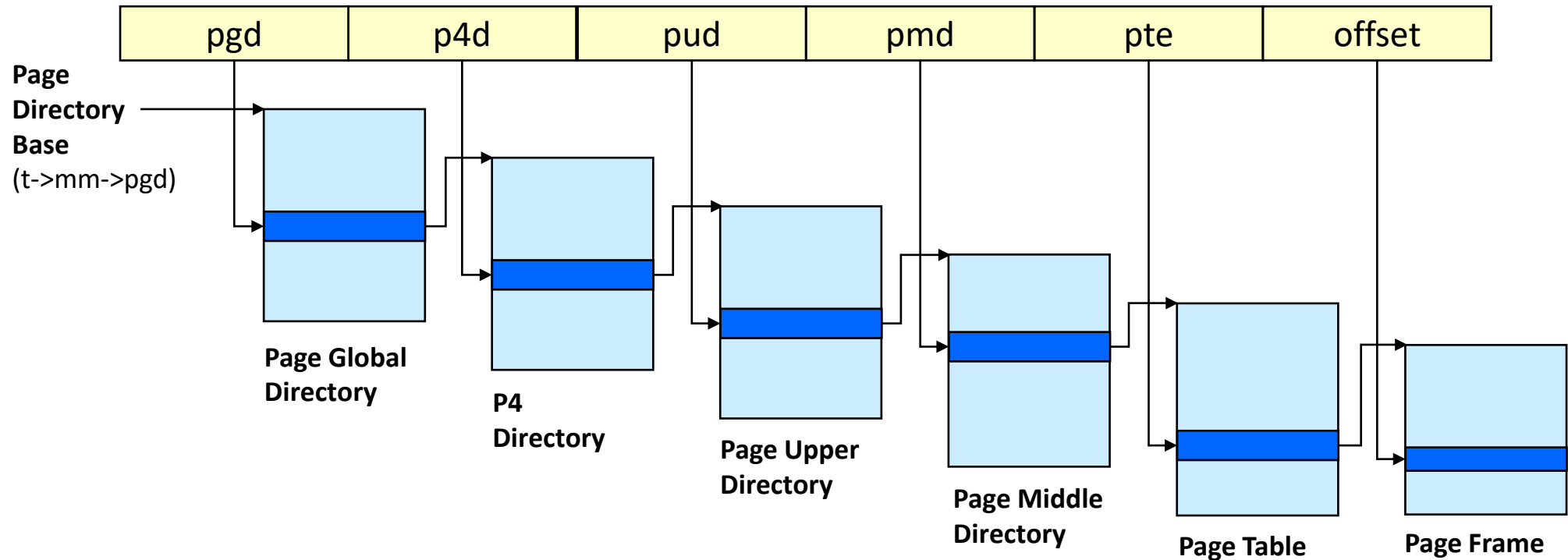


For complete virtual memory map, refer to:
[./Documentation/x86/x86_64/mm.rst](#)

Paging in Linux

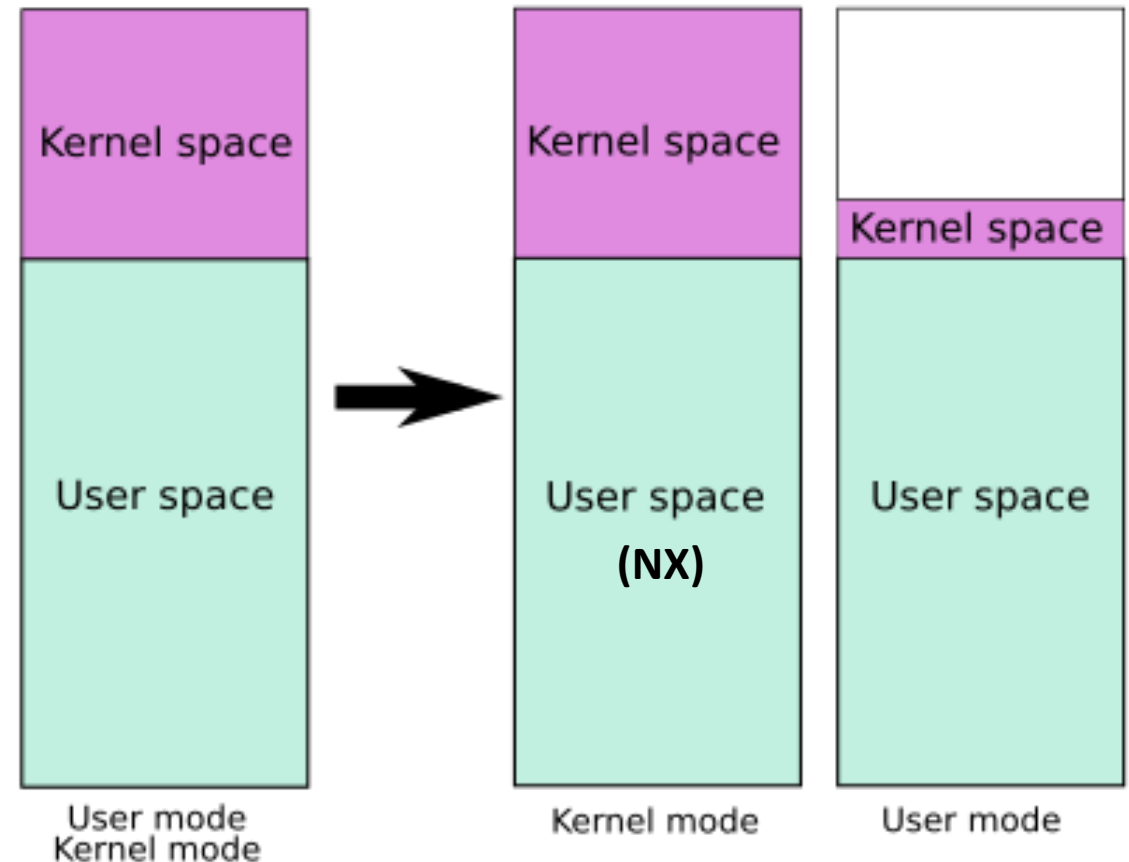
■ Five-level address translation

- 5-level paging for Intel “Ice Lake” processors and beyond (57-bit virtual address)
- For 48-bit virtual address, the size of P4D is set to 1



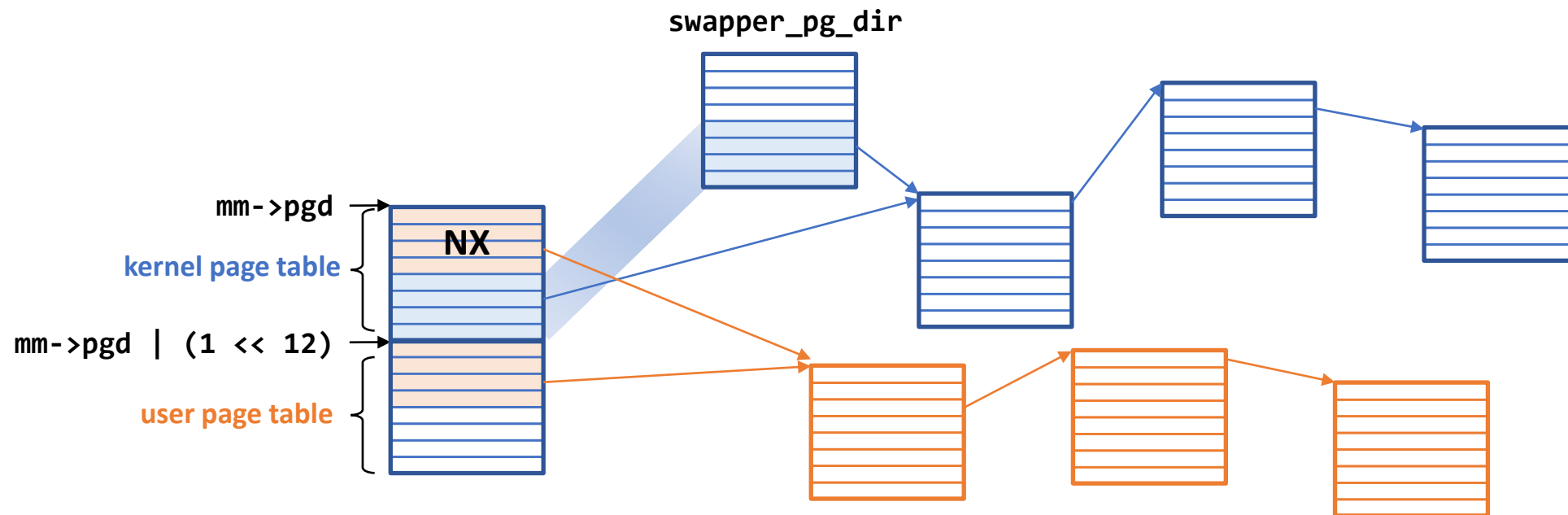
Kernel Page-Table Isolation (KPTI)

- To mitigate Meltdown vulnerability
- Separate page table for kernel
- Minimal kernel space for syscall, page fault & interrupt handling
- Merged in 4.15
- `CONFIG_PAGE_TABLE_ISOLATION=y`
- Disabled by 'nopti' at boot time
- ASID becomes critical to the performance



Page Table Allocation

- When KPTI is enabled, two PGDs are allocated (@ _pgd_alloc())
 - PGD_ALLOCATION_ORDER = 1 (8KB in size and 8KB-aligned)
 - One for kernel address space, the other for user address space
 - User portion of the kernel page table is set with the NX bit



VMA

■ Virtual Memory Area

- A contiguous, page-aligned subset of the virtual address space
- VMAs are linked with a red-black tree for fast lookup of the region corresponding to any virtual address

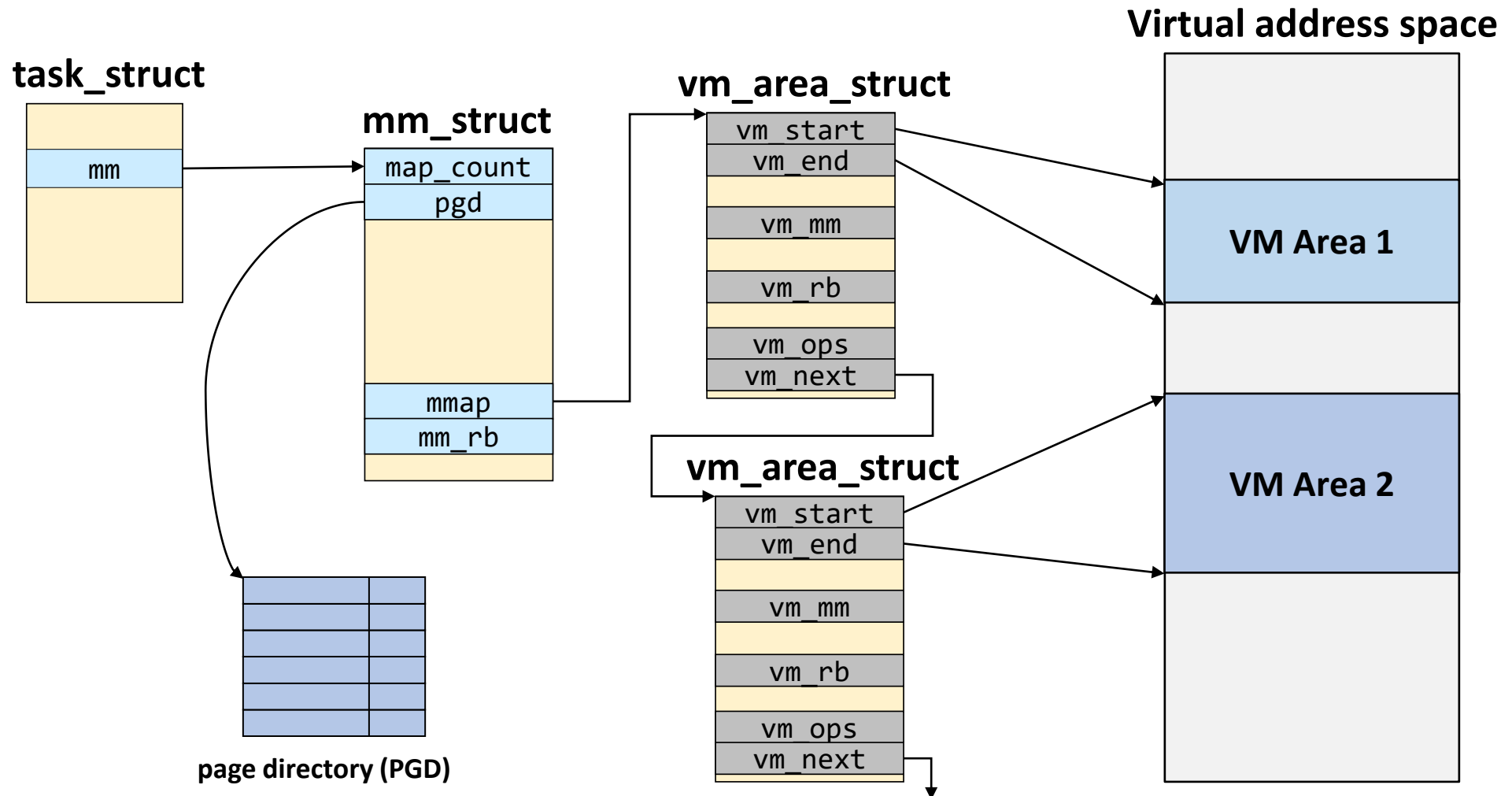
- Described by a `vm_area_struct`
- Either file-backed or anonymous

- `/proc/PID/maps`

```
[sys:~-2028] cat /proc/self/maps
562a517de000-562a517e6000 r-xp 00000000 08:01 2228226 /bin/cat
562a519e5000-562a519e6000 r--p 00007000 08:01 2228226 /bin/cat
562a519e6000-562a519e7000 rw-p 00008000 08:01 2228226 /bin/cat
562a52e6a000-562a52e8b000 rw-p 00000000 00:00 0 [heap]
7f2244bd0000-7f2244db7000 r-xp 00000000 08:01 14286948 /lib/x86_64-linux-gnu/libc-2.27.so
7f2244db7000-7f2244fb7000 ---p 001e7000 08:01 14286948 /lib/x86_64-linux-gnu/libc-2.27.so
7f2244fb7000-7f2244fbb000 r--p 001e7000 08:01 14286948 /lib/x86_64-linux-gnu/libc-2.27.so
7f2244fbb000-7f2244fbd000 rw-p 001eb000 08:01 14286948 /lib/x86_64-linux-gnu/libc-2.27.so
7f2244fbd000-7f2244fc1000 rw-p 00000000 00:00 0
7f2244fc1000-7f2244fe8000 r-xp 00000000 08:01 14286873 /lib/x86_64-linux-gnu/ld-2.27.so
7f224500e000-7f2245030000 rw-p 00000000 00:00 0
7f2245030000-7f22451cb000 r--p 00000000 08:01 262169 /usr/lib/locale/locale-archive
7f22451cb000-7f22451cd000 rw-p 00000000 00:00 0
7f22451e8000-7f22451e9000 r--p 00027000 08:01 14286873 /lib/x86_64-linux-gnu/ld-2.27.so
7f22451e9000-7f22451ea000 rw-p 00028000 08:01 14286873 /lib/x86_64-linux-gnu/ld-2.27.so
7f22451ea000-7f22451eb000 rw-p 00000000 00:00 0
7ffffdbb92000-7ffffdbbb3000 rw-p 00000000 00:00 0 [stack]
7ffffdbbea000-7ffffdbbed000 r--p 00000000 00:00 0 [vvar]
7ffffdbbed000-7ffffdbbef000 r-xp 00000000 00:00 0 [vdso]
ffffffffffff600000-ffffffffffff601000 r-xp 00000000 00:00 0 [vsyscall]
```

VMA *permission offset device i-node* *mapped file name*

VMA Structure



Lazy TLB Flushing

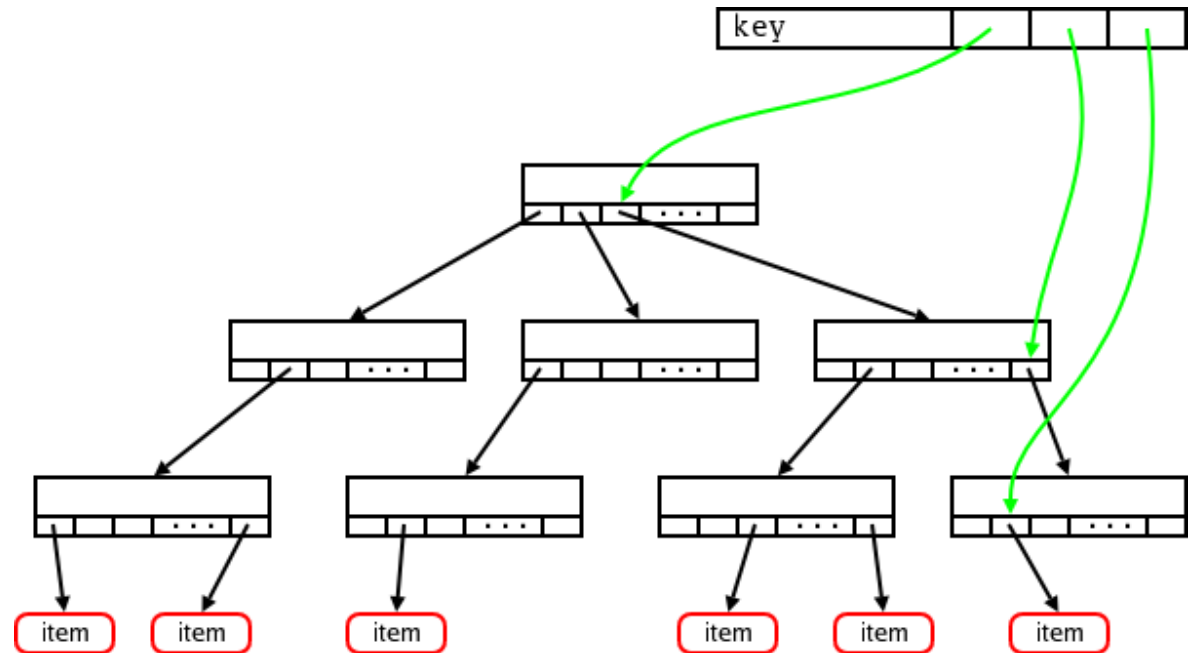
- TLB flush is expensive
- No need to flush TLB for a task without user virtual address space (i.e., kernel threads)
- Implementation
 - `(task_struct *) t->mm`: virtual address space of a process
 - `(task_struct *) t->active_mm`: the effective mm
 - Normally, `t->active_mm == t->mm`
 - On context switch, `t->active_mm->pgd` is stored into CR3
 - If the next task's mm is NULL, use previous task's `active_mm` to avoid TLB flush

Page Cache

- A cache of pages in RAM
 - From reads and writes of regular filesystem files, block device files, mmap'ed files, ...
 - Group cached pages belonging to the same inode

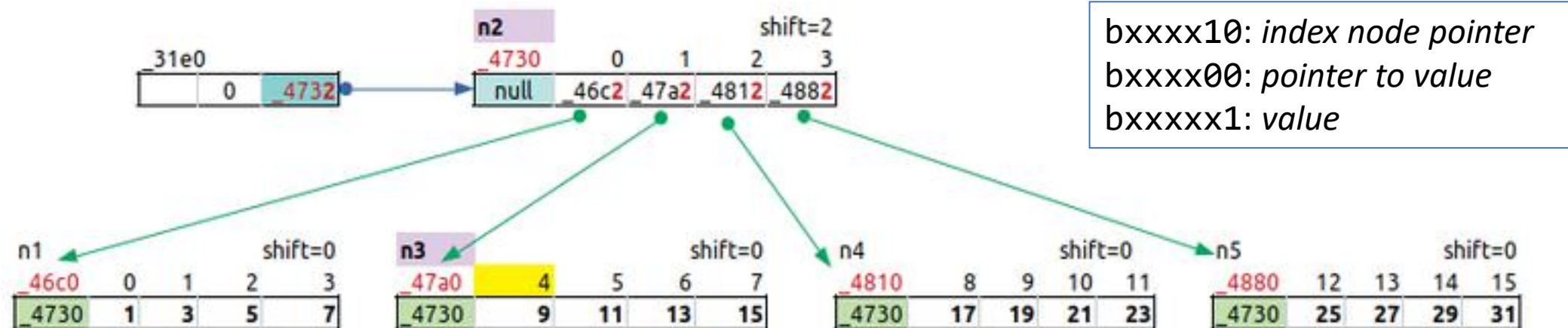
- Page cache lookup

- Each inode has a unique radix tree
- Key: <inode, page offset>
- The radix tree points to the cached page
- Fanout: 64
(16 for small system)



Xarray (eXtensible Array)

- An abstract data type which behaves like a very large array of pointers
 - Radix tree is replaced with XArray since 4.19 (by Matthew Wilcox)
 - Can go to the next or previous entry in a cache-efficient manner compared to a hash
 - No need to copy data or change MMU mappings compared to a resizable array
 - More memory-efficient, parallelizable and cache-friendly than a doubly-linked list
 - Perform lookups without locking using RCU

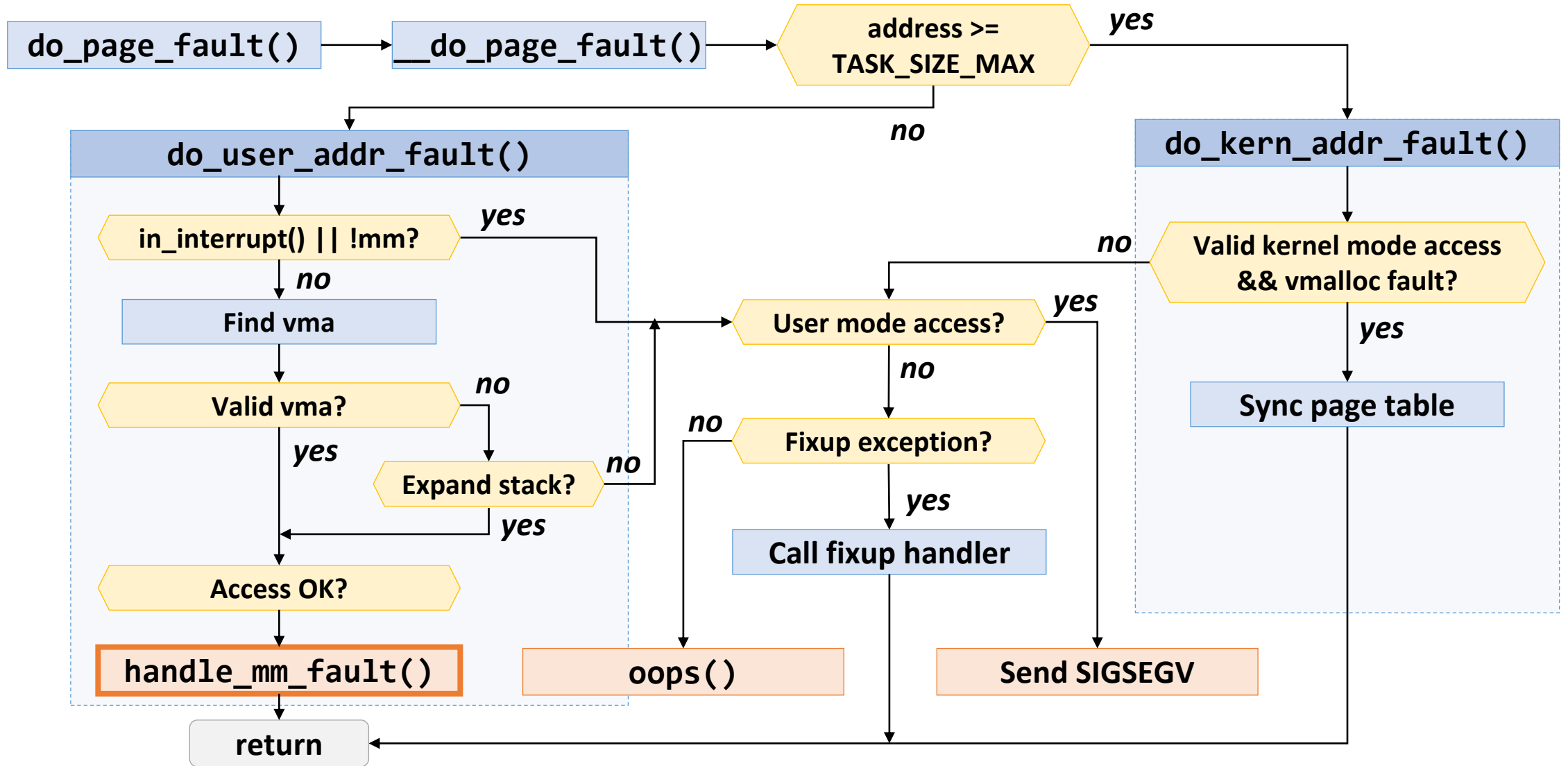


Source: <https://www.kernel.bz/boardPost/118679/19>

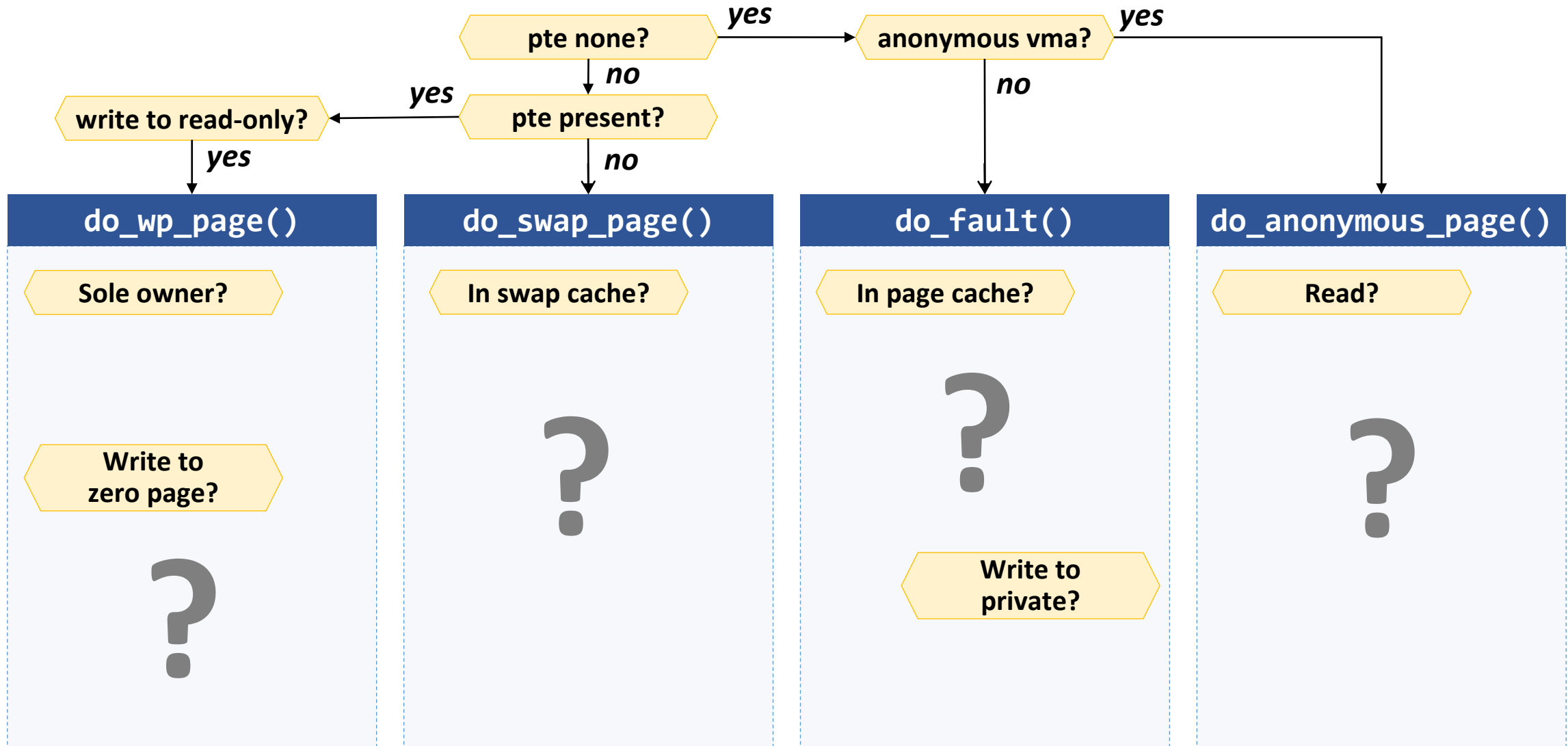
Classifying Page Faults

	User address	Kernel address
User mode	Invalid user address: SIGSEGV Valid user address: Normal page faults	Invalid kernel address: SIGSEGV
Kernel mode	Fixup address: Jump to the fixup handler Kernel bug: OOPS	Inconsistent PGD: Synchronize PGD Fixup address: Jump to the fixup handler Kernel bug: OOPS

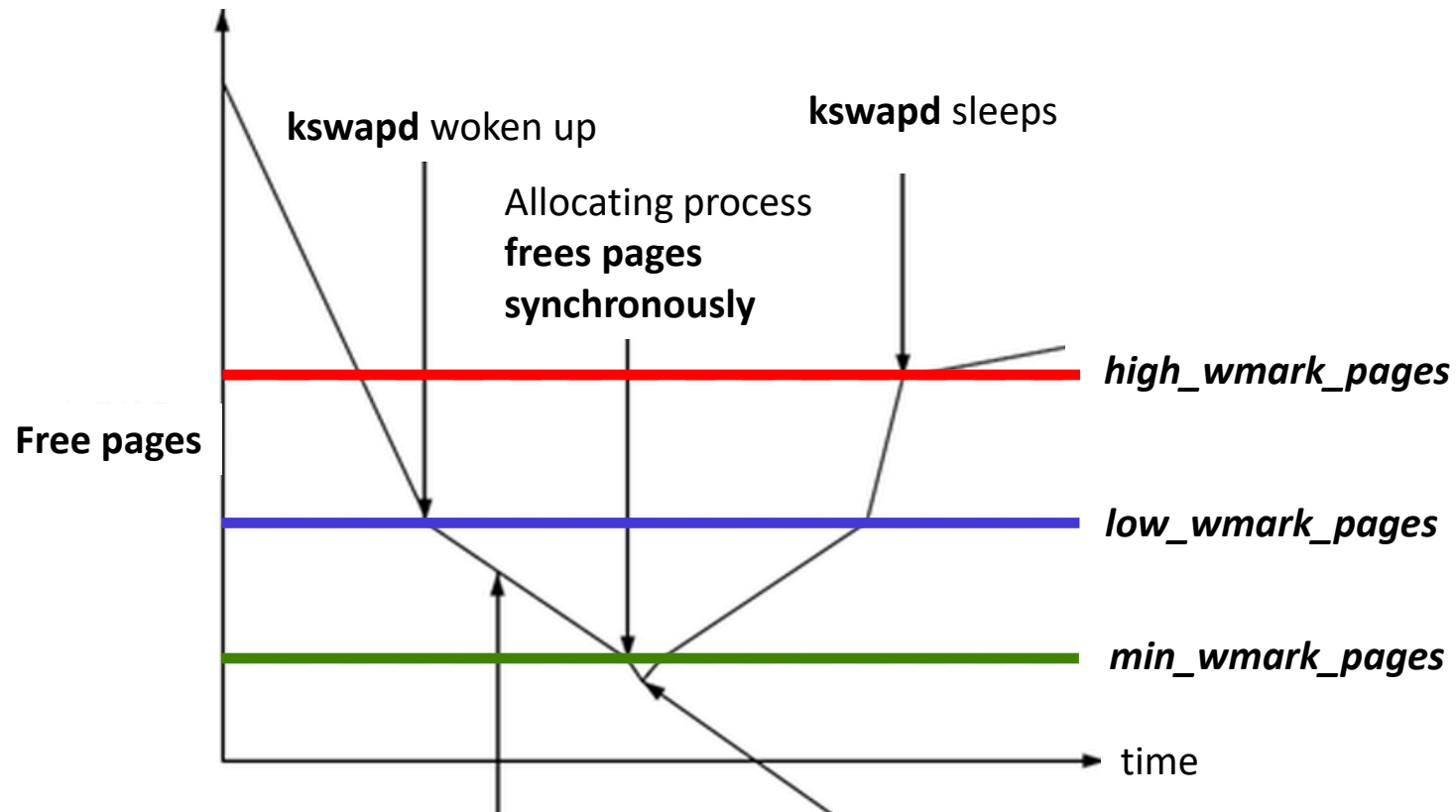
Page Fault Handling



Handling Normal Page Faults



Swapping in Linux



Rate of page consumption is slowed by kswapd but still allocating too fast

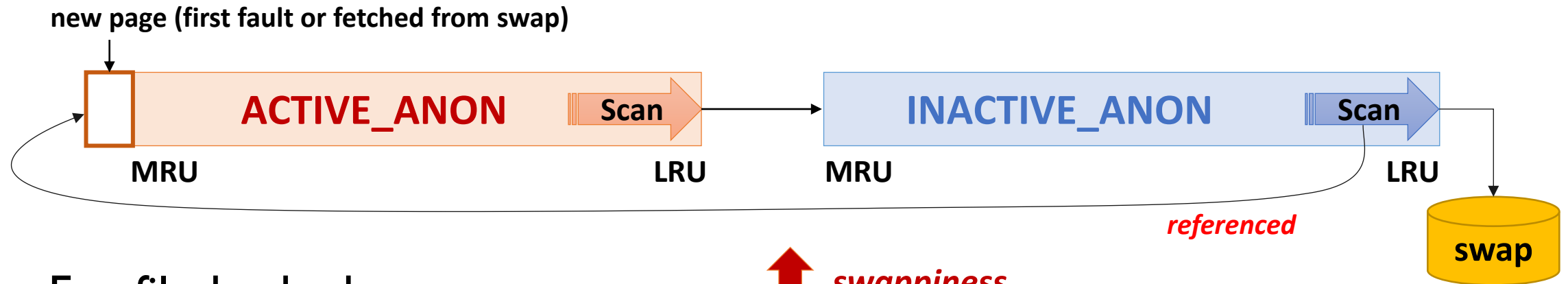
GFP_ATOMIC allocation can go below *min_wmark_pages*

LRU Lists

- `LRU_INACTIVE_ANON`: for inactive anonymous pages
 - `LRU_ACTIVE_ANON`: for active anonymous pages
 - `LRU_INACTIVE_FILE`: for inactive file-backed pages
 - `LRU_ACTIVE_FILE`: for active file-backed pages
 - `LRU_UNEVICTABLE`: for unevictable pages (ramfs, locked pages, etc.)
-
- **Why separate lists for ANON and FILE pages?**
 - Page cache pages may be hidden behind lots of anonymous pages on the LRU
 - The kernel scans over pages that should not be evicted (e.g., scanning anonymous pages when there is no swap)

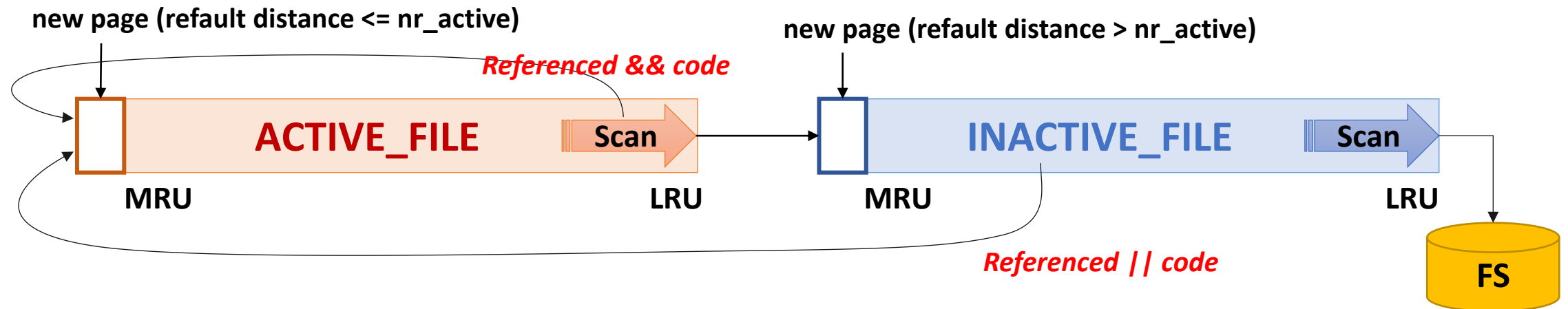
Page Reclaim

- For anonymous pages



- For file-backed pages

↕ swappiness



Multi-Gen LRU

■ MGLRU

- Add more LRU lists (or “generations”) to cover a range of page ages
- PTEs (Page Table Entries) are scanned directly for better locality
- Designed by Yu Zhao at Google, merged in Linux 6.1

