EPIC
Institute of Technology
Powered by <epam>

Operating System Internals

Operating Systems

Alexander Kernozhitsky

Contents

01 **CPU and Scheduling**

02 **Virtual Memory**

03 **Memory Mapping**

01

CPU and Scheduling

CPU

- Execute commands one by one
- React to interrupts (≈ signals)
    - Hardware events
    - Software exceptions (division by zero, page fault, etc.)
    - Double fault, triple fault
- Sleeps until an interrupt is received
- Multi-core, each core executes commands independently

Processes

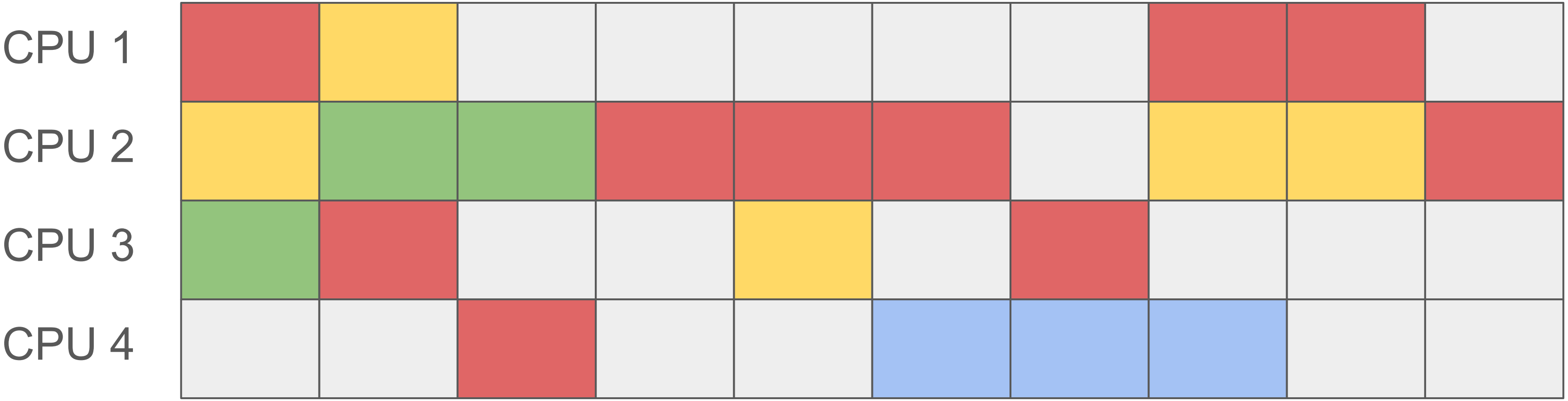
- Up to ~200 CPU cores
- Thousands of processes
- Most processes are almost always idle and only react to some events
- How to schedule processes between cores?
  - Cooperative multitasking (very old Windows)
  - Preemptive multitasking (most modern OSes)
  - No multitasking at all (MS-DOS)

Cooperative multitasking

- Each process decides when to give control back to OS
- What if process hangs?

Preemptive multitasking

- Each process is preempted by timer
- Timer generates interrupt and gives control back to OS



CPU 1	🟥	🟨	⬜	⬜	⬜	⬜	⬜	🟥	🟥	⬜
CPU 2	🟨	🟩	🟩	🟥	🟥	🟥	⬜	🟨	🟨	🟥
CPU 3	🟩	🟥	⬜	⬜	🟨	⬜	🟥	⬜	⬜	⬜
CPU 4	⬜	⬜	🟥	⬜	⬜	🟦	🟦	🟦	⬜	⬜

Protection

- Protection rings: 0, 1, 2, 3
- Privileged instructions
- Kernel memory

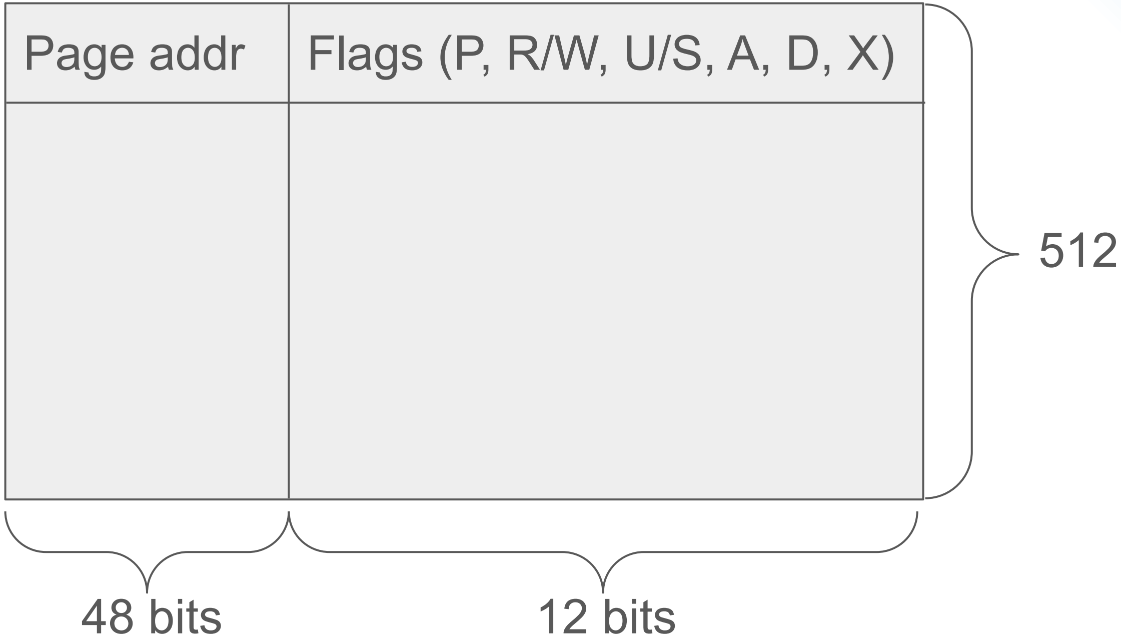
02

Virtual Memory

Memory

- A sequential array of data
- Some parts are reserved for communication with hardware
- How to split between processes?
  - Segments
  - Pages (~4 KB on x86)
- Isolation?

Page Table



Page addr

Flags (P, R/W, U/S, A, D, X)

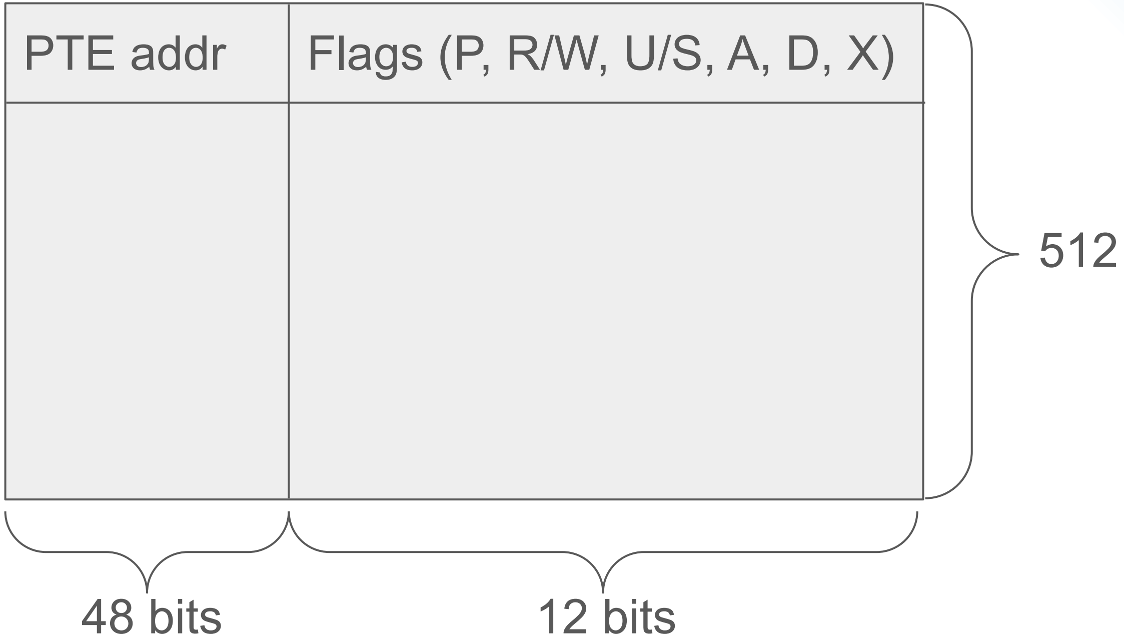
512

48 bits

12 bits

PTE Size = 1 page (4 KB)
Addresses up to 4KB*512 = 2MB

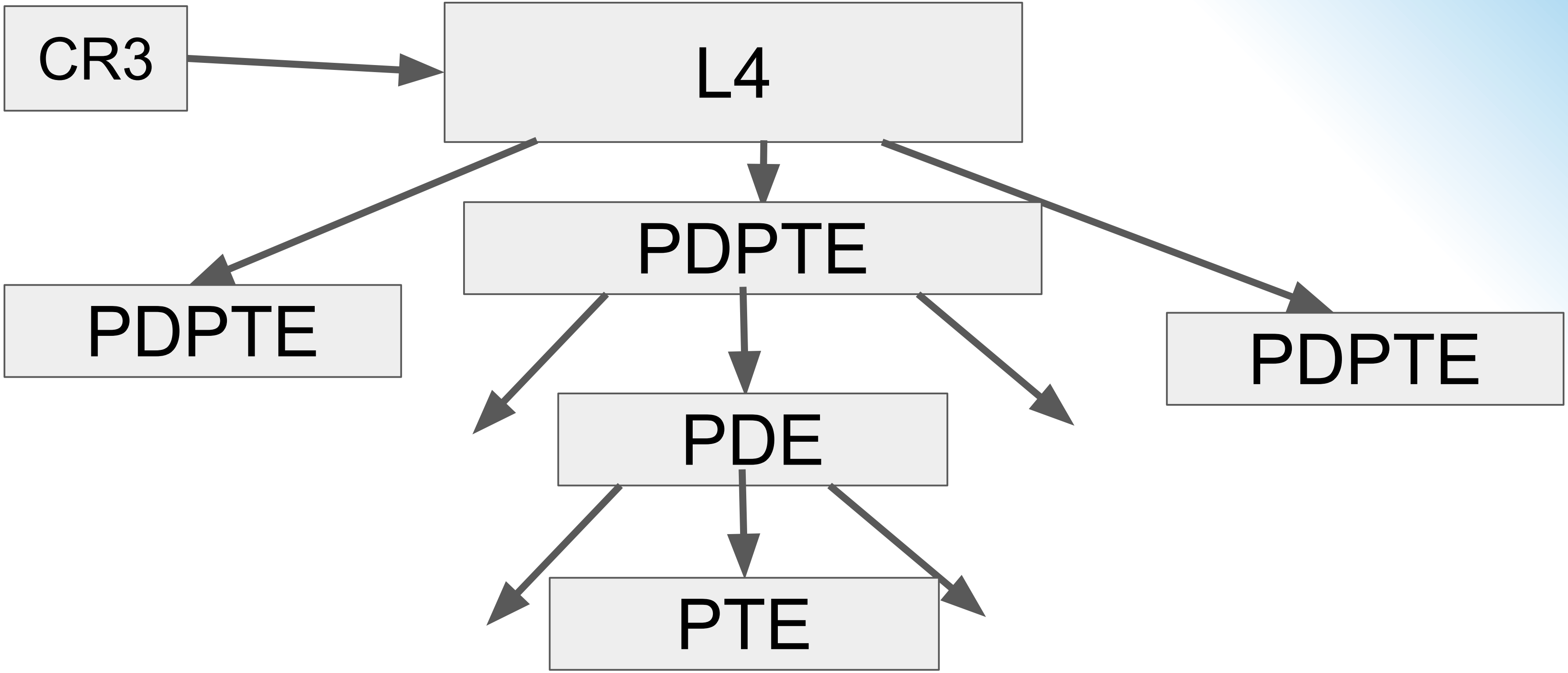
Page Table



PTE addr	Flags (P, R/W, U/S, A, D, X)

PDE Size = 1 page (4 KB)
Addresses up to 1GB

Page Table



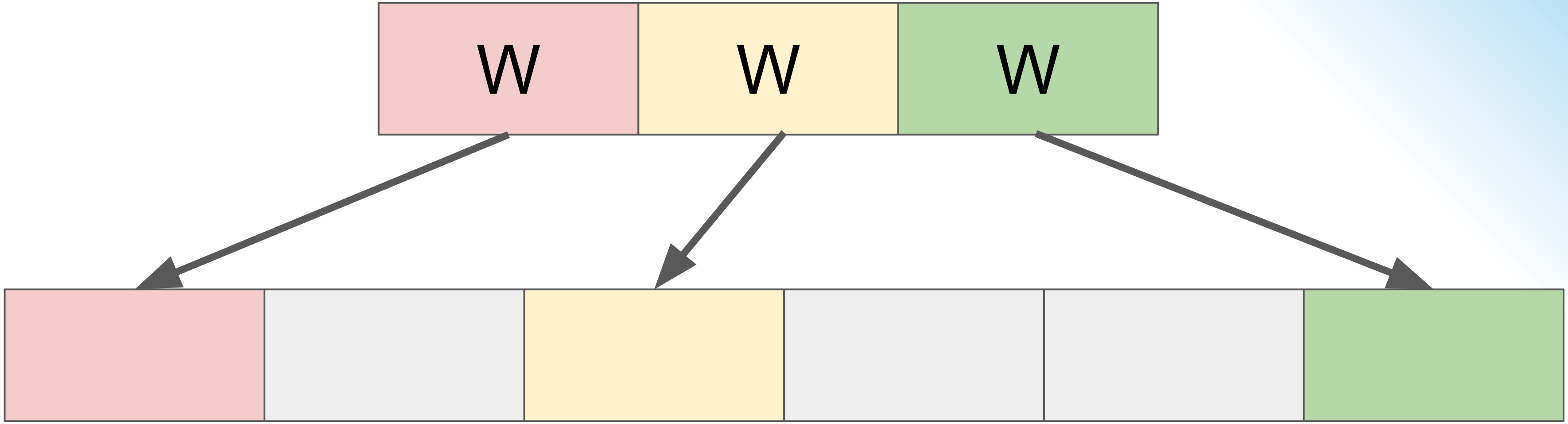
More Paging

- MMU
- Large pages (2 MB)
- Huge pages (1 GB)
- Caching (TLB)
- Total: 48 bit address

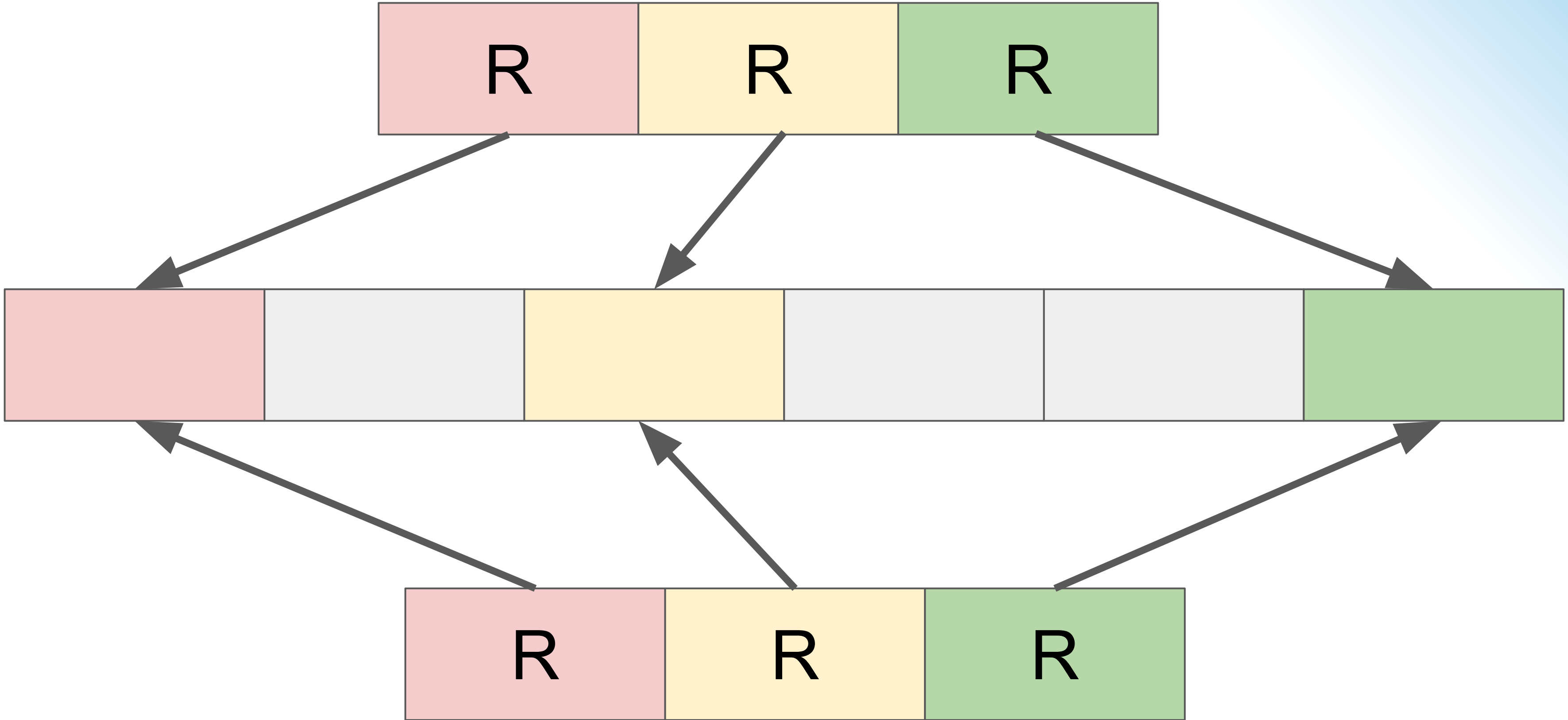
Isolation

- Each process gets its own page table
- Doesn’t see the memory of other processes
- Threads of one process get the same page table
- Shared memory: same physical page, different virtual

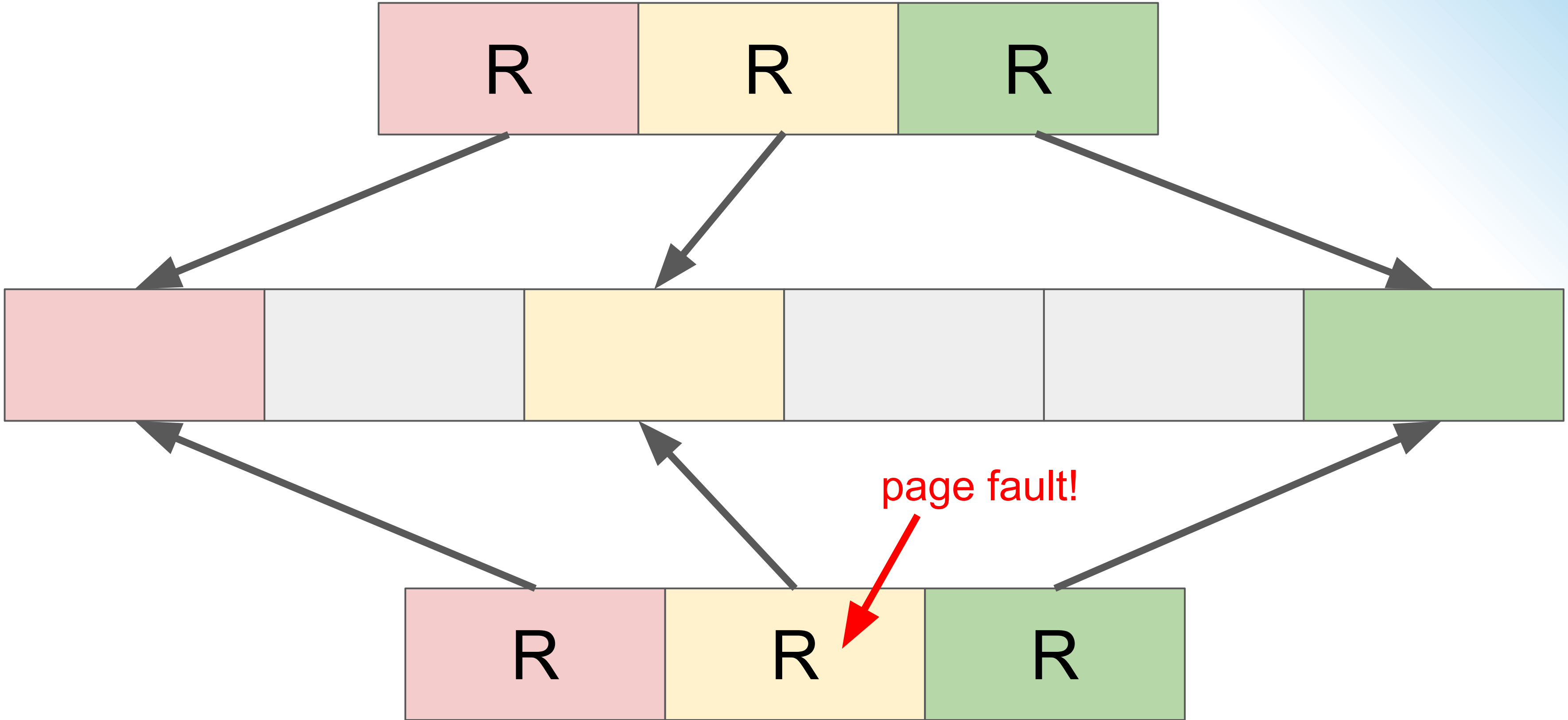
Fork, CoW



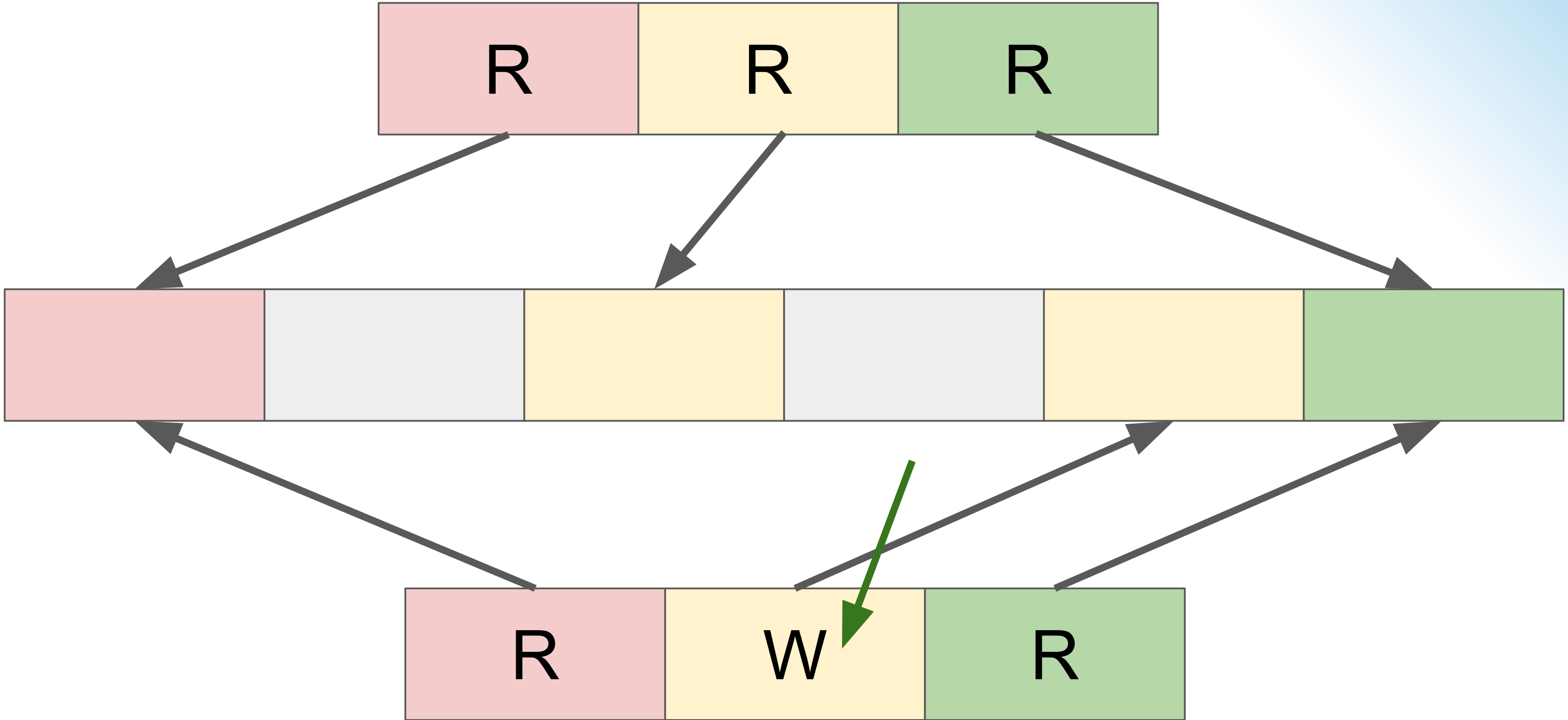
Fork, CoW



Fork, CoW



Fork, CoW



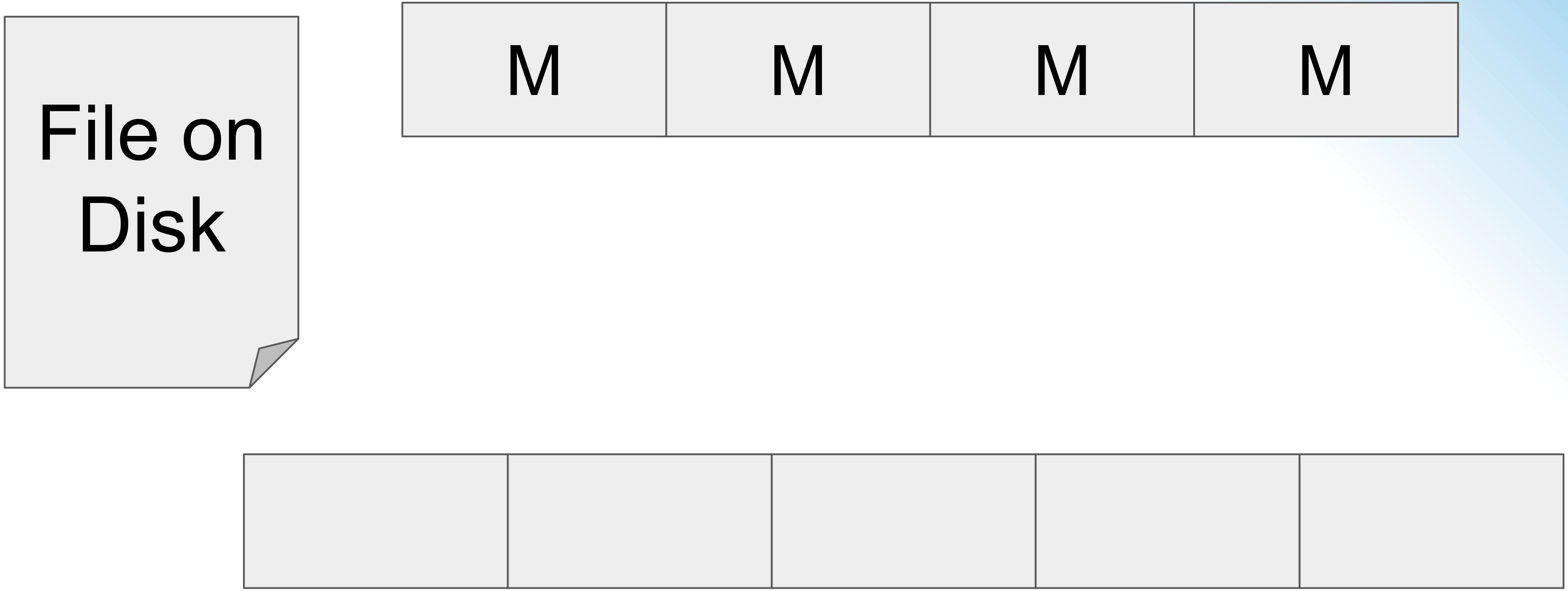
03

Memory Mapping

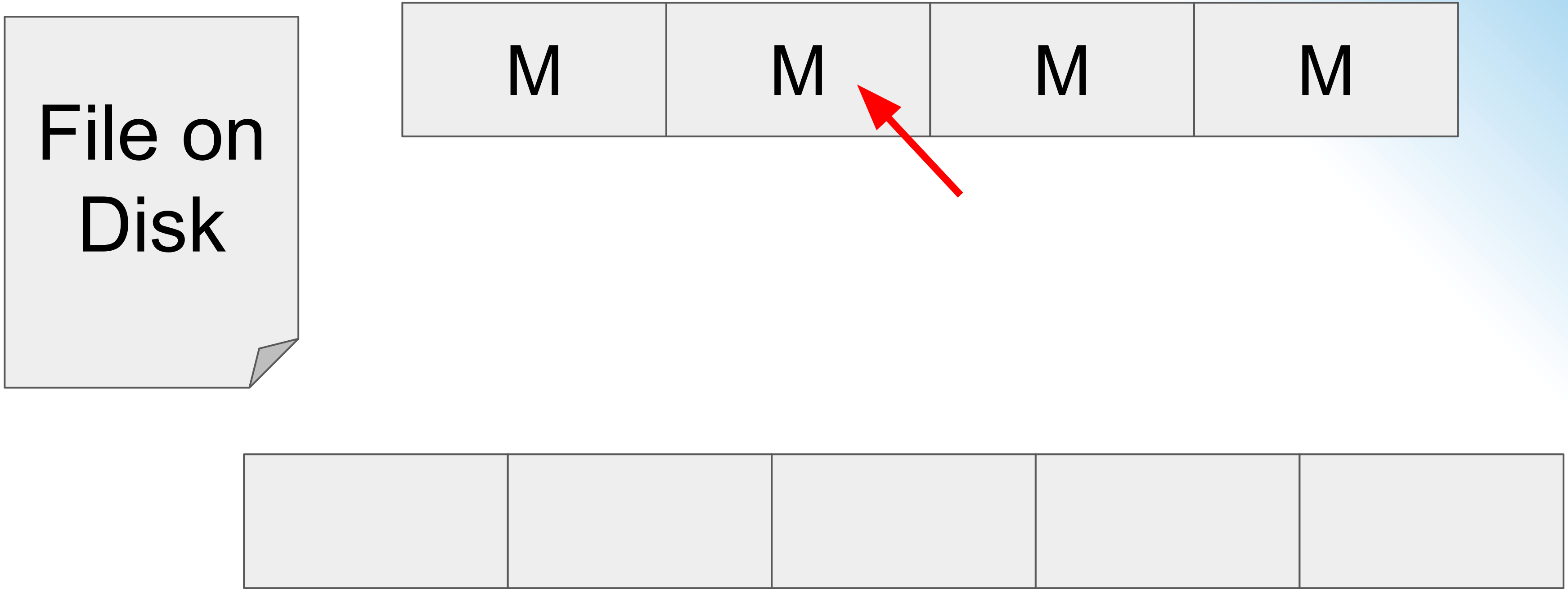
Page cache

- Stores file contents in memory
- Speeds up access to data
- If the page is dirty → write back to disk

Memory Mapping



Memory Mapping



File on Disk

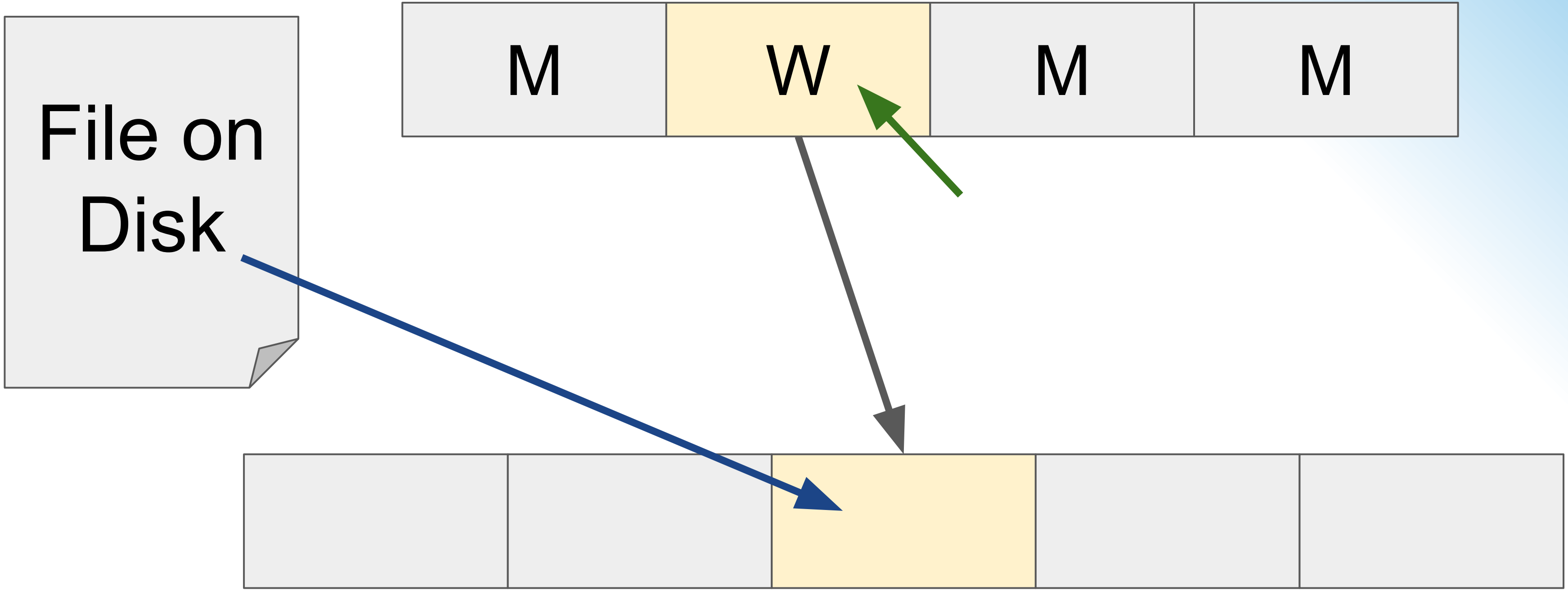
M

M

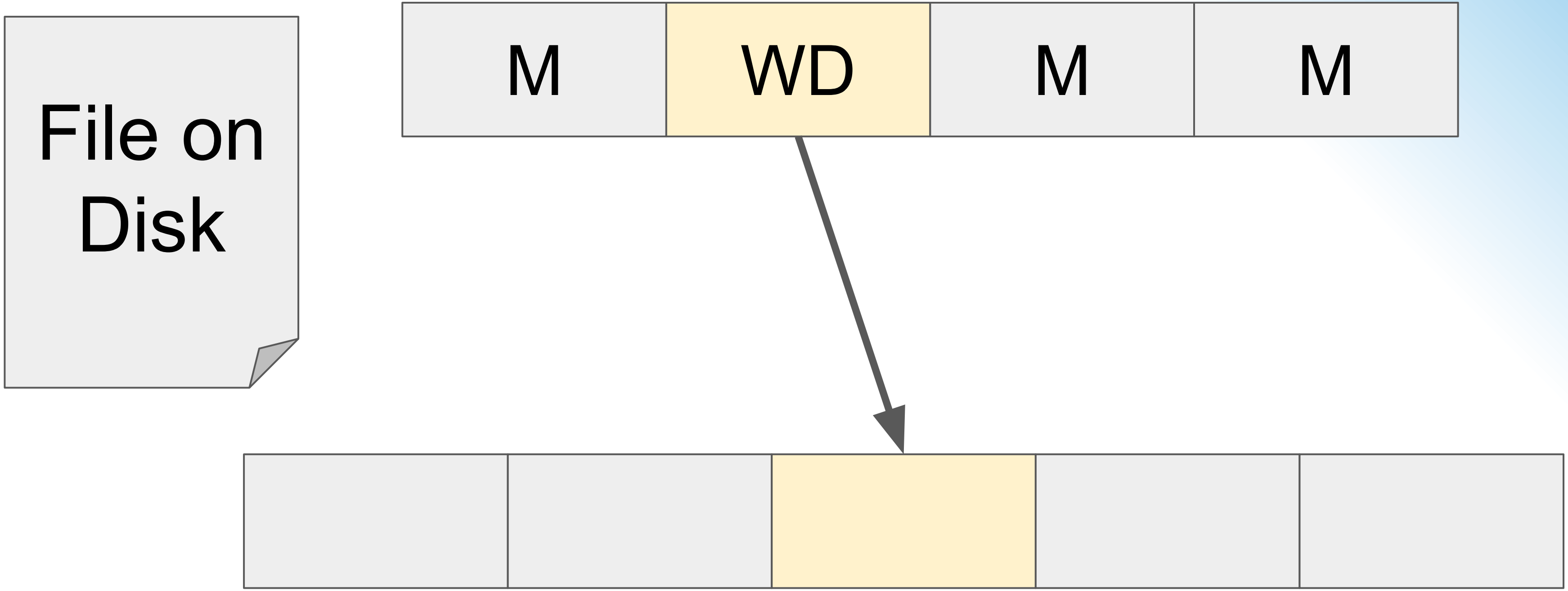
M

M

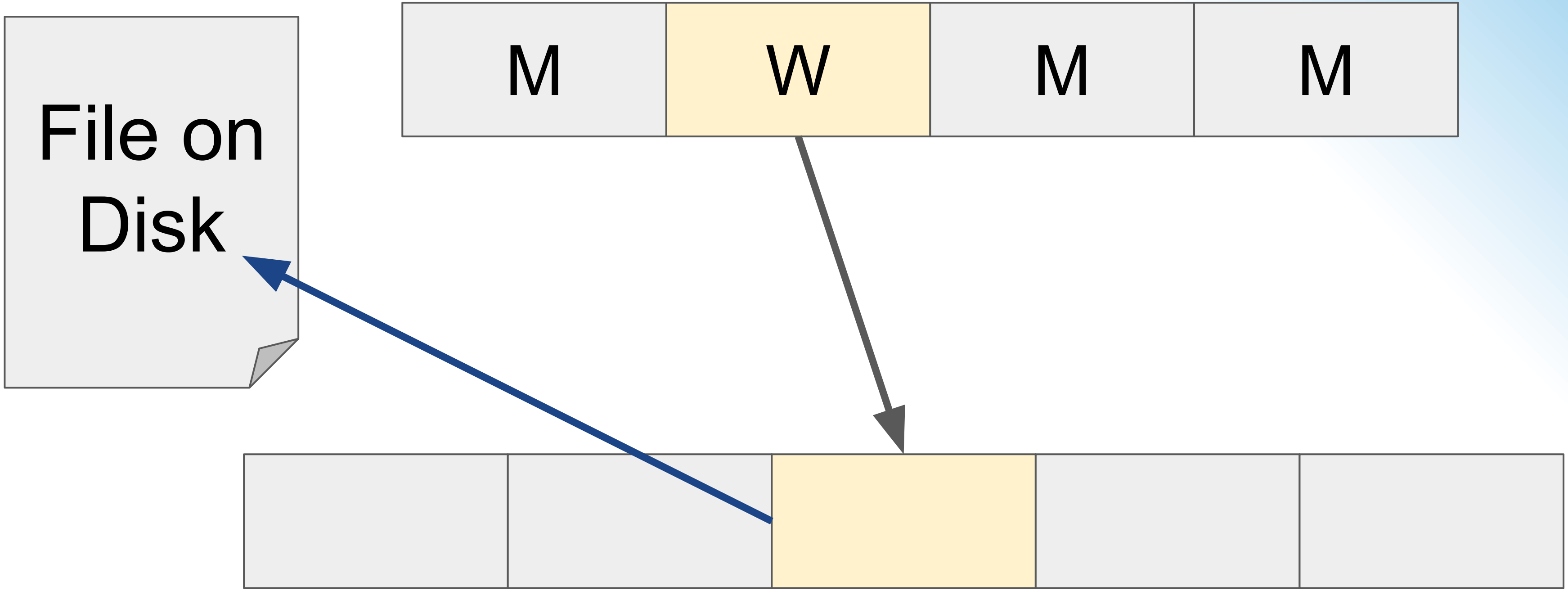
Memory Mapping



Memory Mapping



Memory Mapping


M
W
M
M
File on Disk