Operating Systems - Chapter 18 Summary

Paging: Introduction

• Alternative to segmentation: divide memory into fixed-sized units (pages).
• Virtual memory divided into pages, physical memory divided into page frames.
• Each virtual page maps to a physical frame via the page table.
• Goal: eliminate external fragmentation, simplify memory management, support sparse address spaces.

Simple Example

• Example: 64-byte virtual address space with 16-byte pages → 4 virtual pages.
• 128-byte physical memory with 8 frames.
• Virtual pages placed arbitrarily in frames; tracked by page table.
• Address translation: split virtual address into VPN (virtual page number) and offset.
• VPN indexes page table → PFN (physical frame number). Offset unchanged.

Page Tables

• Per-process data structure mapping virtual pages to physical frames.
• Large: e.g., 32-bit address space with 4KB pages → 1M entries (~4MB per table).
• Stored in memory, not hardware; OS tracks with page-table base register (PTBR).

Page Table Entries (PTEs)

• Typical bits include:
• Valid bit: whether translation exists (supporting sparse address spaces).
• Protection bits: read/write/execute permissions.
• Present bit: whether page is in memory (or swapped out).
• Dirty bit: whether page has been modified.
• Accessed/reference bit: tracks usage for replacement decisions.

Performance Challenges

• Each memory access requires extra memory access to fetch PTE.
• Slows programs by ~2x if not optimized.
• Hardware and OS optimizations (e.g., TLB) are needed to make paging practical.

Memory Trace Example

• Simple loop initializing an array causes multiple accesses:
– Instruction fetch → page table + instruction.
– Array access → page table + data.
• Demonstrates overhead of paging without optimizations.

Summary

Paging divides virtual memory into fixed-sized pages mapped to physical frames. It avoids external fragmentation and supports sparse address spaces, but introduces large page tables and performance overhead. Page tables include translation info and control bits (valid, protection, dirty, accessed). Without optimizations like TLBs, paging doubles memory access cost. Future chapters focus on reducing space/time overhead to make paging efficient.