CIS3110 Lecture 8 - Summary Notes

Memory Management Techniques (Virtual Memory, 10.1-10.6)

Memory Allocation Strategies

- **Best fit**: Works well for data structures with regular-sized nodes (graphs, trees)
- **First fit**: More efficient for strings and variable-sized objects
- **Practical considerations**: The overhead of searching for optimal placement may exceed memory savings

Buddy System (Virtual Memory, 10.4)

- Binary search algorithm applied to memory management
- Divides memory holes in half recursively until reaching appropriate size
- Creates natural "buddies" for re-merging memory later
- Mathematically superior to other approaches but complex to implement
- Used extensively in Linux kernel

Memory Caching (Virtual Memory, 10.6)

- **Cache**: Temporary storage for frequently accessed data
- **Purpose**: Reduces access time by keeping copies closer to the CPU
- **RAM**: Functions as a cache for the paging disk
- **Hardware implementation**: Located between CPU registers and main memory

Cache Architecture

- **Cache lines**: Typically 128 bytes (smaller than page size)
- **Tags**: Memory address identifiers used for lookup
- **Lookup method**: Parallel hardware comparison (associative memory)
- **Cache levels**:
  - L1: Inside CPU, smallest, fastest, most expensive
  - L2: Larger, slower, cheaper than L1
  - Main memory: Much larger but significantly slower

Cache Operations

- **Cache hit**: Requested data found in cache
- **Cache miss**: Data not in cache; must fetch from slower memory
- **During context switch**: Caches must be cleared or tagged with process ID
- **Cache coherence**: Ensuring consistency between cache and memory

Hierarchical Page Tables (Virtual Memory, 10.5)

Problem Addressed

- Flat page tables for 64-bit address spaces would waste enormous amounts of memory
- Most entries would be empty (program only uses small portions of address space)

Implementation

- **Two-level lookup**:
  1. Page directory table: Uses most significant bits of virtual address
  2. Page table chunks: Located using page directory entry
- **Benefits**: Only allocates page table chunks that are actually used
- **Addressing**: Divides virtual address into:
  - Offset bits (same as traditional page tables)
  - Page directory bits (most significant)
  - Page table bits (middle portion)

Memory Efficiency

- Page table chunks typically page-sized
- Segmentation violations occur when accessing addresses not mapped to any page table chunk

Threads vs. Processes (Processes, 3.1, 3.4; Threads, 4.1-4.3)

Thread Advantages

- **Resource efficiency**: Share page tables, code, and data segments
- **Creation speed**: Much faster to create than processes
- **Memory usage**: Lower overhead, only need new stack

Thread Disadvantages

- **Complexity**: Requires critical section management
- **Fragility**: One thread crash affects entire process

- **Coordination overhead**: Requires semaphores, mutexes to prevent race conditions

Real-world Examples

- **Microsoft web server**: Thread-based (higher efficiency but less fault tolerance)
- **Apache web server**: Process-based (better isolation, recovery from individual failures)

I/O Systems (I/O Systems, 12.2-12.5)

Computer Architecture for I/O

- **CPU-Memory**: Connected via data bus
- **CPU-Controllers**: Connected via control bus
- **DMA**: Allows controllers direct access to memory

Data Transfer Methods

- **Serial mode**: One word at a time with CPU interrupts
- **DMA mode**: Bulk transfer directly to memory with single completion interrupt
- **Usage patterns**:
  - Block devices (disks): Use DMA for efficiency
  - Character devices (keyboards, mice): Use serial transfer

Mass Storage Structure (Mass-Storage Structure, 11.1-11.3)

Disk Drive Components

- **Platters**: Circular storage media with magnetic surfaces
- **Read/write heads**: Mounted on arms that move across platters
- **Actuator**: Moves heads to different positions

Disk Coordinates

- **Cylinder**: All tracks at same distance from center across all platters
- **Head/Surface**: Which platter surface
- **Sector**: Angular segment of a track

Disk Performance Factors

- **Rotational latency**: Time for disk to spin to correct position (3ms avg. at 10,000 RPM)
- **Seek time**: Time to move heads to correct cylinder (typically 5-10ms)

- **Transfer rate**: Determined by sector density and rotation speed

Modern Storage Technologies

- **HDD**: Traditional spinning disks, higher capacity, lower cost
- **SSD**: No moving parts, faster access, more expensive
- **Interface compatibility**: Both use cylinder/head/sector addressing for driver compatibility

Key Concepts

1. Memory management involves trade-offs between algorithmic complexity and actual efficiency gains.
2. Caching creates a hierarchy of storage with different speed/capacity/cost characteristics.
3. Hierarchical page tables solve the problem of storing page tables for massive address spaces.
4. Both threads and processes have specific advantages and disadvantages depending on application needs.
5. I/O systems use various strategies to optimize data transfer between peripheral devices and main memory.
6. Disk storage performance depends on both physical limitations and logical organization.