File System and Storage Performance

Operating Systems Lecture

Instructor: Tanaya Bowade & Dr Shabih Fatima

Introduction to File Systems

What is a File System?

A method and data structure that an operating system uses to control how data is stored and retrieved.

Key Roles:

Data Management
Organizes and manages files and directories on storage devices

Abstraction Layer
Provides a logical view of data storage, hiding physical hardware details

Data Integrity
Ensures consistency and reliability through journaling and error checking

Access Control
Implements security measures like permissions and ownership

Importance of I/O Performance

Storage and I/O performance directly impact system responsiveness and user experience.

Why I/O Performance Matters:

- Slow I/O operations cause CPU waiting time, reducing system throughput
- I/O bottlenecks manifest as slow application loading and unresponsive interfaces
- Efficient I/O management is critical for modern computing environments
- Storage technology evolution (HDD to SSD) significantly impacts performance

File System Concepts

Core Components

Files

Basic unit of data storage. A named collection of related information recorded on secondary storage. Can contain text documents, images, or executable programs.

Directories

Special files that contain lists of other files and subdirectories. Provide hierarchical structure for organizing files, making them easier to locate and manage.

Metadata

Data about data. Key information includes:

- Size
- Owner
- Permissions
- Timestamps
- Location
- Type

Attributes & Naming

File Attributes

Read-only
Prevents modification

Hidden
Invisible in listings

System
Critical OS file

Archive
Needs backup

Naming Conventions

- **Length Restrictions:** Maximum characters (e.g., 255 in many systems)
- **Forbidden Characters:** Characters not allowed (e.g., /, \, :, *, ? in Windows)
- **Case Sensitivity:** Some systems are case-sensitive (UNIX/Linux)
- **Extensions:** Suffixes indicating file type (e.g., .txt, .docx)

File System Structure

Layered Architecture

File systems are organized in layers, each providing services to the layer above.

Application Interface
System calls: open(), read(), write(), close()

Logical File System
File organization, directory management, access control

File-Organization Module
Logical to physical block mapping, free space management

Basic File System
I/O scheduling, device-independent I/O

I/O Control
Device drivers, interrupt handling

Devices
Disk drives, SSDs, other storage media

Real-world Implementations

UNIX File System (UFS)

- Uses inodes to store metadata separately from data blocks
- Case-sensitive file names
- Efficient for sequential access workloads

NTFS (Windows)

- Case-insensitive but preserves case
- Advanced security features and permissions
- Better support for large volumes

File Allocation and Access Methods

File Allocation Methods

Contiguous Allocation

File occupies consecutive blocks on disk

- Simple implementation
- Fast sequential access
- External fragmentation

Linked Allocation

File stored as linked or chain list of blocks

- No external fragmentation
- Flexible file size
- Slow direct access

Indexed Allocation

Index block points to data blocks

- No external fragmentation
- Fast direct access
- Index block overhead

File Access Methods

Sequential Access

Data accessed in order, from start to finish

- Similar to reading a tape or log file
- Processed one record at a time
- **Example:** Reading text files line by line

Direct Access (Random Access)

Access data at any arbitrary position

- Jump to specific block or record
- Essential for database applications
- Requires efficient indexing mechanisms

Directory Implementation and Management

Directory Structures

Single-Level Directory

All files contained in a single directory.

Directory
fileA fileB fileC

- Simplicity
- Naming conflicts

Two-Level Directory

Separate directory for each user.

Master Dir
User1_Dir
fileA fileB
User2_Dir
fileA fileC

- Resolves naming conflicts
- No user grouping

Tree-Structured Directory

Hierarchical structure with multiple levels.

Root Dir1
SubDirA
fileX
fileY

Dir2
SubDirB
fileZ

- Hierarchical organization
- No naming conflicts

Storage Devices and Performance Basics

HDD vs. SSD

Feature	HDD	SSD
Mechanism	Mechanical spinning platters	Flash memory (NAND-based)
Seek Time	Milliseconds (ms)	Microseconds (μs)
Transfer Rate	50-200 MB/s	500-7000 MB/s
Random I/O	Poor	Excellent
Durability	Susceptible to physical shock	More robust, no moving parts

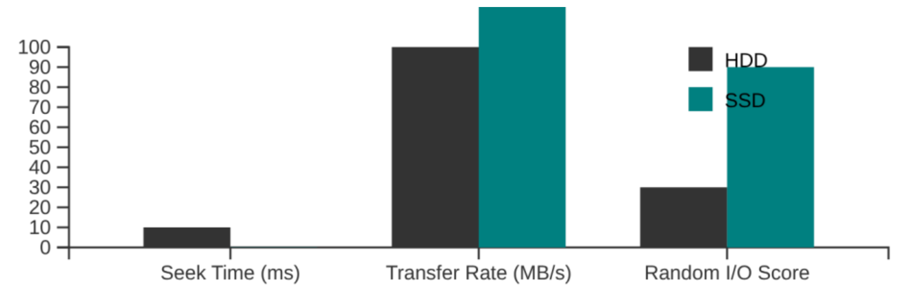
Key Performance Metrics

Latency
Delay between I/O request and data transfer start. Measured in ms (HDD) or μs (SSD).

Throughput
Amount of data transferred per second (MB/s or GB/s). Critical for sequential operations.

IOPS
Input/output operations per second. Important for workloads with many small random I/Os.

Performance Comparison



Storage Performance Monitoring

Key Metrics

Latency
Delay between I/O request and data transfer, measured in ms (HDD) or μs (SSD)

Throughput
Data transferred per unit time, measured in MB/s or GB/s

IOPS
Input/output operations per second, critical for workloads with many small I/O requests

Utilization
Percentage of time storage device is busy processing I/O requests

Queue Depth
Number of I/O requests waiting to be processed

Monitoring Tools

Linux Tools

iostat - Reports I/O statistics

```
Linux 5.15.0-generic (hostname) 11/03/2025 _x86_64_ (8 CPU)
avg-cpu: %user %nice %system %iowait %steal %idle
0.50 0.00 0.50 0.00 0.00 99.00
Device tps kB_read/s kB_wrtn/s
sda 0.00 0.00 0.00
sdb 0.00 0.00 0.00
```

tps: Transactions per second

vmstat - Reports system statistics

Includes block I/O statistics (bi, bo) that indicate disk activity

Windows Tools

Performance Monitor (perfmon)

Graphical tool for monitoring storage performance counters

Can track metrics like Disk Queue Length, % Disk Time, and Avg. Disk Response Time

I/O Performance Analysis

Analysis Techniques

Bottleneck Identification

Pinpointing the component in the I/O path that limits overall performance.

Workload Characterization

- Read-heavy vs. Write-heavy: Proportion of read vs. write operations
- Random vs. Sequential: Access patterns (impact on HDD performance)
- I/O Size: Average size of data blocks being read/written
- Concurrency: Number of simultaneous I/O requests

Optimization Techniques

Faster Storage Media
Upgrading from HDD to SSD

RAID Configurations
Combining multiple disks for performance

Load Balancing
Distributing I/O across multiple devices

Application Improvements
Optimizing code to reduce I/O

Caching & Buffering
Implementing effective caching

File System Tuning
Adjusting parameters for workloads

Caching and Buffering

Role in Performance

Caching
Stores copies of frequently accessed data in faster, temporary storage (e.g., RAM). Reduces latency for subsequent accesses.

Buffering
Temporarily holds data during transfer between devices or between a device and an application. Smoothes out differences in data transfer rates.

Cache Operation:

Cache Miss ↓

Cache Hit ↓

Cache Hit ↓

Cache improves read performance by storing frequently accessed data in faster memory

Modern OS Examples

Implementation in Modern OS:

Linux
Implementing sophisticated caching through the unified buffer cache/page cache, which dynamically manages memory for both file data and metadata.

Windows
Uses memory-mapped files and I/O buffering to optimize file system performance and reduce physical disk accesses.

Key Benefit: Modern operating systems dynamically allocate and deallocate memory for caching based on system load and application demands, ensuring optimal performance.

Resource Management Context

File system performance is deeply intertwined with the broader concept of operating system resource management. The OS orchestrates the efficient use of CPU, memory, and I/O devices, which are interconnected and affect each other's performance.

File System

CPU Management

- High CPU utilization by other processes delays file system request processing
- Insufficient CPU cycles for file system operations slows metadata updates

Memory Management

- Excessive page swapping increases I/O operations and slows system performance
- Limited memory for caches reduces hit rate, forcing more disk accesses

I/O Management

- Device drivers and storage controllers impact I/O request processing
- Queue depth and scheduling algorithms affect storage performance

Key Insight: Effective resource management requires a holistic approach, as performance issues in one area can impact others.

Review Questions

Key questions for test preparation

1. Define and explain the main components of a file system.

2. What are the key differences between HDD and SSD performance?

3. How can caching improve I/O performance?

4. Describe one method used to monitor storage performance.

5. How does resource management influence file system efficiency?

These questions cover the key topics. Review your notes and slides for detailed answers.