# Linux Storage and Caching — Comprehensive Notes

---

## 🧱1. Difference Between Block and Object Storage

### Definition

- **Block Storage:** Data is stored in fixed-size chunks (blocks), typically 512 bytes or 4 KB. These blocks are addressed and managed by the operating system directly. Common in databases and virtual disks.
- **Object Storage:** Data is stored as *objects*, each containing the data, metadata, and a unique identifier. It's designed for large-scale, unstructured data like media, backups, and logs.

### Comparison Table

| Feature | **Block Storage** | **Object Storage** |
|---|---|---|
| **Storage Unit** | Fixed-size blocks | Objects (data + metadata + ID) |
| **Access Method** | Low-level (iSCSI, Fibre Channel) | RESTful HTTP APIs |
| **Metadata** | Minimal | Extensive, customizable |
| **Structure** | Organized into volumes | Flat namespace in buckets |
| **Performance** | High IOPS, low latency | High throughput |
| **Use Cases** | Databases, VM disks | Backups, archives, media |
| **Examples** | AWS EBS, SAN | AWS S3, Azure Blob, GCS |

**Analogy:** - Block Storage = shelves with numbered pages (precise, fast access) - Object Storage = a digital library with searchable metadata

---

## 🧮2. Caching and Write-Through / Write-Back Policies

### Caching Overview

Caching stores frequently accessed data in faster memory (RAM or SSD) to speed up data access.

**Goal:** Minimize disk access and latency.

### Write Policies

#### A. Write-Through Cache

- Writes go to both cache and main memory immediately.
- Ensures data consistency and safety.

✅Pros: Safe, consistent data.
❌Cons: Slower writes.

#### B. Write-Back Cache

- Writes go only to cache initially, then later to memory.
- Cache marks dirty data until written back.

✅Pros: Very fast writes.
❌Cons: Risk of data loss if power fails.

**Comparison:**

| Property | **Write-Through** | **Write-Back** |
|---|---|---|
| Speed | Slower | Faster |
| Safety | High | Moderate |
| Consistency | Immediate | Delayed |
| Complexity | Simple | Complex |
| Use Case | Databases, logs | CPU cache, graphics |

---

## 🏺 3. Deep Dive: Caching & Storage in Linux

### 🕰️3.1 Linux Storage Layers

| Type | Description | Example |
|---|---|---|
| **Block Storage** | Raw device (e.g., /dev/sda) | HDD, SSD |
| **File Storage** | Managed by file system | ext4, XFS |
| **Object Storage** | Accessed via user-space layers | Ceph, MinIO |

### 🕰️3.2 Linux Page Cache

- Stores recently accessed disk pages in RAM.
- Avoids frequent disk reads.

**Commands:**

```
free -h
cat /proc/meminfo | grep -i cache
```

To clear cache (for testing):

```
sudo sync; echo 3 | sudo tee /proc/sys/vm/drop_caches
```

### 🕰️3.3 Write Caching in Linux

Linux supports both **write-through** and **write-back** caching.

**Write-Through (Sync)**

```
mount -o sync /dev/sdb1 /mnt/data
```

Flushes data immediately to disk.

**Write-Back (Default)**

Buffered writes in page cache, periodically flushed by kernel writeback daemons.

**Monitor writeback:**

```
cat /proc/meminfo | grep Dirty
```

### 🕰️3.4 Buffer Cache vs Page Cache

| Type | Purpose | Managed By |
|---|---|---|
| **Buffer Cache** | Raw block I/O | Block Layer |
| **Page Cache** | File system caching | VFS |
| **Unified Cache** | Combined for efficiency | Kernel |

### 🕰️3.5 Linux Writeback Path

```
App → Page Cache → Dirty Pages → pdflush → Disk
```

### 🕰️3.6 I/O Scheduler Types

| Scheduler | Use Case |
|---|---|
| **CFQ** | Balanced I/O |
| **Deadline** | Predictable latency |
| **NOOP** | Best for SSDs |
| **BFQ** | Fair throughput |

### 🕰️3.7 Disk Caching Commands

Enable/disable hardware write cache:

```
sudo hdparm -W 1 /dev/sda   # Enable
sudo hdparm -W 0 /dev/sda   # Disable
```

Check cache status:

```
sudo hdparm -W /dev/sda
```

### 🕰️3.8 Write Behavior Summary

| Step | Component | Action |
|---|---|---|
| 1 | App | Writes data |
| 2 | Kernel | Adds to page cache |
| 3 | Kernel | Marks as dirty |
| 4 | Daemon | Flushes to disk |
| 5 | Disk | Commits write |

### 🕰️3.9 Tuning Parameters

| Parameter | Description | Default |
|---|---|---|
| `vm.dirty_ratio` | Max % dirty memory before blocking writes | 20 |
| `vm.dirty_background_ratio` | % when background flushing starts | 10 |
| `vm.swappiness` | Swap preference | 60 |

---

## 💽 4. Deep Dive on `sudo hdparm -W 1 /dev/sda`

### Purpose:

Enables the **disk's internal write cache**.

### Meaning of Components:

- `hdparm`: Utility for controlling drive parameters.
- `-W 1`: Enable write caching (0 disables it).
- `/dev/sda`: Target device.

### Effect:

Data is acknowledged as written before it's actually saved to disk — increasing speed but risking loss on power failure.

### Verification:

```
sudo hdparm -W /dev/sda
```

Output:

```
/dev/sda:
 write-caching =  1 (on)
```

### Persistence:

Add to `/etc/hdparm.conf`:

```
/dev/sda {
    write_cache = on
}
```

### Performance vs Reliability:

| Environment | Recommendation |
|---|---|
| Laptop/Desktop | Enable for performance |
| Database Server (no UPS) | Disable for safety |
| Enterprise Server with UPS | Enable safely |

## 5. Disk Cache vs Memory Cache

| Cache Type | Location | Managed By | Purpose |
|---|---|---|---|
| **CPU Cache (L1–L3)** | Inside CPU | Hardware | Store recent instructions/data |
| **Page Cache (RAM)** | Kernel memory | OS | Cache file pages |
| **Disk Cache** | Disk firmware | Drive | Buffer disk reads/writes |

**Data Path:**

```
CPU → L1/L2/L3 Cache → RAM (Page Cache) → Disk Cache → Disk Media
```

They complement each other at different layers for optimal I/O performance.

---

## ⚙️6. Background Flush Threshold (vm.dirty_background_ratio)

### Definition:

Determines when Linux starts **background flushing** of dirty pages (modified data in page cache) to disk.

### How It Works:

1. Data written → stays dirty in RAM.
2. Kernel monitors % of dirty pages.
3. When dirty pages exceed `vm.dirty_background_ratio`, background daemons (`flush-x:y`) begin writing data slowly to disk.
4. If it exceeds `vm.dirty_ratio`, processes are blocked until enough data is written.

### Example:

For 8 GB RAM: - `vm.dirty_background_ratio = 10` → starts flushing at 0.8 GB dirty data. - `vm.dirty_ratio = 20` → blocks at 1.6 GB dirty data.

### Tuning:

```
sudo sysctl -w vm.dirty_background_ratio=5
sudo sysctl -w vm.dirty_ratio=15
```

**Persistent Setting:** Add to `/etc/sysctl.conf`:

```
vm.dirty_background_ratio = 5
vm.dirty_ratio = 15
```

**Analogy:**

Memory = water tank, Disk = drain. - `dirty_background_ratio` = when to start draining. - `dirty_ratio` = when tank overflows — urgent flush.

---

## 🔗 7. Summary

| Concept | Key Takeaway |
|---|---|
| Block vs Object Storage | Blocks = speed & structure; Objects = scalability & metadata |
| Write-Through vs Write-Back | Safety vs Speed trade-off |
| Linux Page Cache | RAM-level caching for files |
| hdparm -W | Controls hardware disk write cache |
| Disk vs Memory Cache | Hardware vs System layer caching |
| Background Flush | Prevents cache overload by proactive flushing |

---

**In essence:** Linux uses multiple caching layers — from CPU to disk — working in harmony. Each layer (page cache, disk cache, background flush daemon) plays a role in balancing **speed, safety, and stability** of data handling.