Chapter 42: Crash Consistency: FSCK and Journaling

1. The Crash Consistency Problem

- File systems must be able to handle unexpected system crashes (e.g., power outages).
- A single logical operation (e.g., appending to a file) may require multiple physical writes to the disk.
- If a crash occurs between these writes, the file system can be left in an **inconsistent state**.
- **Example:** Appending to a file requires updating the inode, the data block, and the data bitmap. A crash could leave the inode pointing to a garbage block.

2. Solution #1: The File System Checker (fsck)

- `fsck` is a tool that scans the entire disk to find and fix inconsistencies.
- It is typically run at boot time.
- **Problem:** `fsck` is very slow, especially on large disks, as it needs to read the entire disk.

3. Solution #2: Journaling (Write-Ahead Logging)

- **Journaling** is a more efficient solution to the crash consistency problem.
- **Key Idea:** Before writing any changes to their final locations on disk, the file system first writes a description of the changes to a **journal** (or log).
- The journal is a sequential, append-only log.
- This allows for fast recovery, as the file system only needs to read the journal to restore consistency.

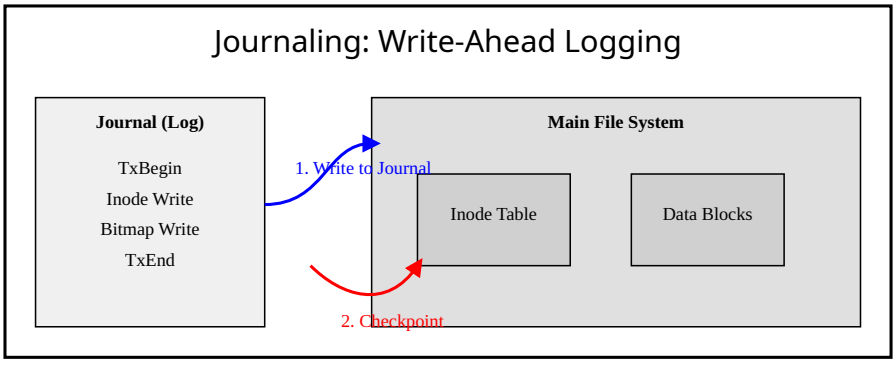

Figure 1: Journaling Overview

Journaling Code Example

```
// Simplified journaling logic
void file_append(file_t *f, char *data) {
    // 1. Write to journal
    journal_begin_tx();
    journal_write(f->inode, new_inode_data);
    journal_write(data_bitmap, new_bitmap_data);
    journal_write(data_block, data);
    journal_end_tx();

    // 2. Checkpoint (write to final locations)
    write_to_disk(f->inode, new_inode_data);
    write_to_disk(data_bitmap, new_bitmap_data);
    write_to_disk(data_block, data);
}
```

4. Data vs. Metadata Journaling

- **Data Journaling:**
  - Both metadata and user data are written to the journal.
  - Provides the strongest consistency guarantee.
  - **Problem:** It doubles the amount of write traffic, which can hurt performance.
- **Metadata Journaling (Ordered Journaling):**
  - Only metadata is written to the journal.
  - User data is written to its final location *before* the corresponding metadata transaction is committed to the journal.
  - This is the most common approach, as it provides a good balance of performance and consistency.

5. Other Approaches to Crash Consistency

- **Soft Updates:** Carefully orders all writes to the file system to ensure that the on-disk structures are never left in an inconsistent state.
- **Copy-on-Write (COW):** Instead of overwriting data in place, COW file systems write new versions of data to unused locations on the disk.
- **Backpointer-based Consistency (BBC):** Adds a backpointer from each block to the inode it belongs to, allowing the system to verify consistency without enforcing strict write ordering.

6. Summary

- Journaling is a widely used technique for ensuring crash consistency in modern file systems.
- It provides a significant performance improvement over `fsck` by avoiding the need to scan the entire disk.

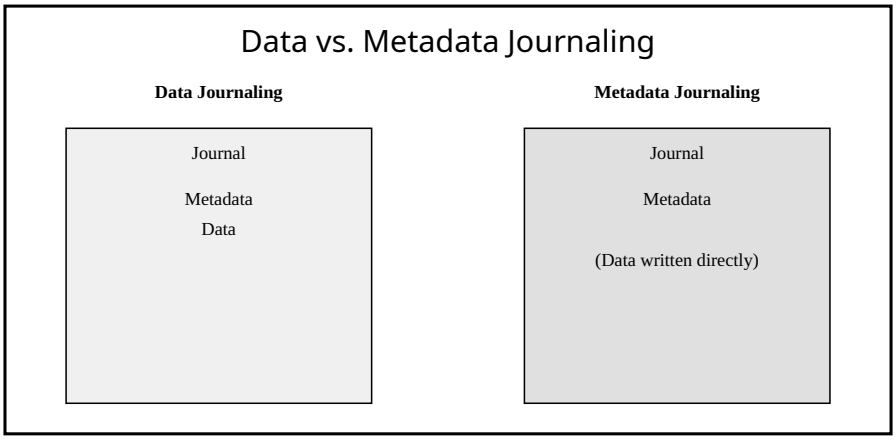

Figure 2: Journaling Types

- Ordered metadata journaling is the most popular implementation, as it offers a good trade-off between performance and consistency.