作业十三

蔡合森 2022K8009009004

13.1现有一个文件系统，在其使用文件缓存的情况下，某个应用创建了一个文件“/home/OS24/fs03.pdf”，并往该文件中写入了12 KB的数据，请分析该过程需要写几个块?分别写哪几个块? 如果在任意时刻发生宕机，会出现哪些不一致?请详细列出所有不一致的情况。（注：假设home和OS24目录都已存在）
解:

- **目录块**:
  - 更新 `/home/OS24` 目录的内容：1 块。
- **inode 块**:
  - `fs03.pdf` 文件的 inode 信息：1 块。
- **数据块**:
  - 写入 12 KB 数据，占用 3 个块。
- **总计**：5 个块。
- **不一致**:
  1. 数据块成为孤块。
  2. 文件内容部分丢失或文件不可见。
  3. 文件大小或目录信息错误。
  4. 文件逻辑上不可访问，目录项和 inode 不一致。

13.2某个文件系统在磁盘上保存了一个大小为20 KB的文件A，现有一个进程打开文件 A，并调用write函数一次性向文件 A 的文件块0 和文件块1写入新数据。假设该文件系统使用文件缓存，且宕机可能发生在任意时刻。请分析

1. 如果文件系统采用数据日志，宕机恢复后，文件A的内容是什么?请分不同情况讨论(即在什么样的宕机情况下，文件A的内容是什么);
2. 如果文件系统采用元数据日志，并且采用先改数据再改元数据的方式，宕机恢复后，文件 A 的内容是什么?请分不同情况讨论(即在什么样的宕机情况下，文件 A 的内容是什么)。

1. 文件系统采用数据日志

1. **宕机发生在日志记录之前**:
   - 数据日志未写入，文件块 0 和块 1 的更新数据丢失。
   - 恢复后，文件 A 的内容保持为旧数据（块 0 和块 1 块未更新）。

2. **宕机发生在日志记录完成后、数据块未更新时**：
    - 数据日志中有块 0 和块 1 的新数据，但磁盘上的文件块仍为旧数据。
    - 恢复后，系统会从日志中重放操作，将块 0 和块 1 更新为新数据。
    - 文件 A 的内容为新数据（块 0 和块 1 块更新）。
3. **宕机发生在数据块更新部分完成时**：
    - 数据块 0 已完成更新，但块 1 未完成更新。
    - 恢复后，日志中的完整数据会覆盖部分更新的块，确保块 0 和块 1 都恢复为新数据。
    - 文件 A 的内容为新数据（块 0 和块 1 块一致更新）。

2. 文件系统采用元数据日志

1. **宕机发生在数据块写入之前**：
    - 数据块 0 和块 1 的写入尚未开始。
    - 恢复后，元数据未更新，文件 A 的内容保持为旧数据（块 0 和块 1 块未更新）。
2. **宕机发生在数据块部分写入时**：
    - 块 0 写入完成，但块 1 尚未写入或部分写入。
    - 恢复后，元数据未更新，文件仍指向旧的块。
    - 文件 A 的内容保持为旧数据（块 0 和块 1 块未更新）。
3. **宕机发生在数据块写入完成、元数据更新之前**：
    - 块 0 和块 1 写入完成，但元数据日志尚未更新。
    - 恢复后，元数据未记录新的块指针，文件仍指向旧的块。
    - 文件 A 的内容保持为旧数据（块 0 和块 1 块未更新）。
4. **宕机发生在元数据日志记录完成后**：
    - 块 0 和块 1 的数据写入完成，元数据日志也已写入。
    - 恢复后，系统从日志中重放元数据操作，文件指向新数据块。
    - 文件 A 的内容更新为新数据（块 0 和块 1 块均更新）。

13.3LFS 的imap和CR都采用类似数组的结构，下标是ino或imap块号，每一项保存对应i-node或imap块的磁盘地址。例如， imap[k]记录ino为k的i-node的磁盘地址；CR[n]记录第n个imap块的磁盘地址。假设一个LFS的块大小为 4KB，磁盘地址占4B。如果已经分配了200万个i-node，请问：

1. 该LFS的imap有多少个块？请给出计算过程;
2. 该LFS的CR有多少个块？请给出计算过程;
3. 如何查ino=356302的inode的磁盘地址？请给出查找和计算过程。

1. 计算 imap 的块数

- 每个 `imap` 项记录一个 i-node 的磁盘地址，占用 4 字节。
- 一个块的大小为 4 KB，能存储的 `imap` 项数量为：
  $$
  \text{每块记录数} = \frac{\text{块大小}}{\text{每项大小}} = \frac{4096}{4} = 1024 \, (\text{项})
  $$
- 已分配了 200 万个 i-node，`imap` 总共需要的项数为 200 万项。所需的块数为：
  $$
  \text{imap 块数} = \lceil \frac{\text{总项数}}{\text{每块记录数}} \rceil = \lceil \frac{2000000}{1024} \rceil = 1954 \, (\text{块})
  $$

2. 计算 CR 的块数

- `CR` 记录了所有 `imap` 块的磁盘地址，每个地址占用 4 字节。
- `imap` 有 1954 块，因此 `CR` 需要记录 1954 个磁盘地址。
- 一个块能记录的磁盘地址数量为：
  $$
  \text{每块记录数} = \frac{\text{块大小}}{\text{每项大小}} = \frac{4096}{4} = 1024 \, (\text{地址})
  $$
- 所需的 `CR` 块数为：
  $$
  \text{CR 块数} = \lceil \frac{\text{总地址数}}{\text{每块记录数}} \rceil = \lceil \frac{1954}{1024} \rceil = 2 \, (\text{块})
  $$

3. 查找 ino=356302 的 i-node 磁盘地址

- 一个 `imap` 块记录 1024 项，因此 `imap[k]` 所在的块号为：
  $$
  \text{imap 块号} = \lfloor \frac{\text{ino}}{\text{每块记录数}} \rfloor = \lfloor \frac{356302}{1024} \rfloor = 348
  $$

- `imap[k]` 在块内的偏移为：
  $$
  \text{块内偏移} = \text{ino} \mod \text{每块记录数} = 356302 \mod 1024 = 270
  $$
- 因此，`imap[356302]` 位于第 348 个 `imap` 块的第 270 项。
- `CR[n]` 记录第 `n` 个 `imap` 块的磁盘地址。
- 所以，通过 `CR[348]` 可以定位到第 348 个 `imap` 块的磁盘地址。
- 根据块内偏移为 270，可以在第 348 个 `imap` 块的第 270 项中找到 `ino=356302` 的磁盘地址。

13.4一个LFS的块大小为4KB，segment大小是4MB。文件块采用多级索引，即包含10个直接指针，以及一、二、三级间接指针各1个。每个指向数据块的指针占4字节。该 LFS 中已经有一个10MB的文件foo，请分析：

1. 给出文件 foo 的文件块索引结构，即文件foo使用了哪些指针？
2. 在该LFS中写文件 foo 的第 2560 块(假设它在磁盘块 Ai 中，Ai 为磁盘逻辑块号)，需要写哪些块？需要几次I/O？请给出它们写在磁盘上的顺序；
   3）如果是 Fast FS (其块大小也为 4KB)，写文件 foo 的第 2560 块，需要写哪些块? 需要几次I/O?
3. 如果是日志文件系统，只记录元数据日志，且日志不采用批量提交，则写文件 foo 的第 2560 块，需要写哪些块？需要几次 I/O?

1. 文件 `foo` 的文件块索引结构

- 每个直接指针指向一个文件块，共有 10 个直接指针，可以索引 **10 块**。
- 一级间接指针指向一个块表（每个表项指向一个文件块），1 个间接块可索引：
  $$
  \text{块表项数} = \frac{\text{块大小}}{\text{指针大小}} = \frac{4096}{4} = 1024 \, (\text{块})
  $$
- 二级间接指针指向一级间接块表，每个二级间接块可索引：
  $$
  \text{二级索引范围} = 1024 \times 1024 = 1048576 \, (\text{块})
  $$
- 三级间接指针类似，可索引：
  $$
  \text{三级索引范围} = 1024 \times 1024 \times 1024
  $$
  文件 `foo` 的块分布：
- 文件大小为 10 MB，占用 2560 个块。

- 块分布如下：
  - **直接指针**：前 10 块通过直接指针索引。
  - **一级间接指针**：接下来的 (2560 - 10 = 2550) 块由一级间接指针索引，需要：
    $$
    \lceil \frac{2550}{1024} \rceil = 3 \, (\text{一级间接块})
    $$
- 文件 `foo` 使用了 10 个直接指针和 3 个一级间接块。

2. LFS 写 `foo` 的第 2560 块

写操作次序：

1. 数据块 (A_i)。
2. 第 3 个一级间接块。
3. i-node。
4. imap 块。
5. CR 块。

- 共需要 **5 次 I/O**。

3. Fast FS 写 `foo` 的第 2560 块

写操作次序：

1. 数据块 (A_i)。
2. 第 3 个一级间接块。
3. i-node。

- 共需要 **3 次 I/O**。

4. 日志文件系统（只记录元数据日志）

写操作次序：

1. 数据块 (A_i)。
2. 元数据更新日志（包含一级间接块和 i-node 的修改）。

写操作数：

- 共需要 **2 次 I/O**