作业 15

15.1 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)。

答：以下考虑，假设采用批量提交，即日志可能会不落盘

（1）、如果文件系统采用数据日志，那么所有的数据写入操作都会先被记录到日志中。在宕机恢复后，文件系统会检查日志并重新执行所有未完成的操作。

将对文件 A 的写入过程分为以下几个阶段：

① 对文件块的写操作写入到日志中（包括 inode 块，可能有 bitmap 块，如果需要分配新数据块）

② 将日志刷新到磁盘中，(包括 Begin Transaction 和 Commit）

③ 对缓存中文件 A 的文件块写入并标记为脏块(日志 checkpoint 后)

④ 将缓存中的文件块 A 的数据刷新到磁盘中

宕机恢复后，文件 A 的内容因此有两种可能：

- 如果宕机发生在数据被写入日志之后，但在数据被写入文件 A 之前，即发生在②和④之间，那么宕机恢复后，会根据日志文件的内容重新执行写入操作，文件 A 将包含新的数据。
- 如果宕机发生在数据被写入日志之前，即发生在过程②中或②之前，那么恢复后，文件 A 的内容将不会改变，因为日志块中没有记录任何未完成的写入操作。

如果宕机恢复后再出现宕机，那么由于操作并未完成，再次恢复后依然可以通过日志文件进行恢复。

（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，那么只有元数据的更改会被记录到日志中。在这种情况下，文件 A 的内容也取决于宕机发生的时间：

同样是将对文件 A 的写入过程分为以下几个阶段：

① 将数据写进到缓存块中，并标志为脏位

② 在日志中记录对 inode 的改动，（包括 inode 块，可能有 bitmap 块，如果需要分配新数据块）

③ 将日志刷新到磁盘中

④ 将缓存中的数据写入到磁盘中

- 如果宕机发生在①之前，但在元数据被写入日志之前，那么恢复后，文件 A 将不包含新的数据，因为数据尚未被写回到磁盘中，但是元数据的更改也未被记录，因此数据与元数据是一致的，不会产生影响。
- 如果宕机发生在①之后④之前，那么恢复后，由于缓存中的数据已经被刷新，文件 A 的内容将不会改变，但是元数据将会通过日志被更改，因此就出现了元数据和数据不一致的情况，这个时候就会出现问题。
- 如果宕机发生在④之后，那么数据将被写入到磁盘中，元数据也会更改，宕机恢复后是正常的情况。

15.2 LFS 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=654321 的 inode 的磁盘地址？请给出查找和计算过程。

答：

（1）、磁盘地址占 4B，已经分配了 200 万个 i-node，那么各 imap 数组总长度之和就将为 200 万，总大小即$2 \times 10^6 \times 4B = 1953.125 \times 4\text{KB}$，也即 1954 个 imap 块。

（2）、CR 记录了每个 imap 块的磁盘地址，因此其数组的大小应为 1954，又每个磁盘地址为 4B，因此总的 CR 数组共需要$1954 \times 4B \approx 1.9082 \times 4KB$，即 CR 共有两个块。

（3）、ino 为 654321，那么首先由每个 CR 块共可存储 1024 个 imap 块，每个 imap 块可存储 1024 个 inode 磁盘块地址，那么每个 CR 块即可找到$1024 \times 1024 = 1048576$个 inode 磁盘地址。

于是由$654321/1048576 = 0$,即对应的 imap 块位于第一个 CR 块中，又$654321/1024 = 638$,因此该 inode 位于第一个 CR 块中的第 638 个 imap 块中，又$654321 - 638 \times 1024 = 1009$，所以实际上的 inode 磁盘地址位于第一个 CR 块中的第 638 个 imap 块中的第 1009 个磁盘地址处。

15.3 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) 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?

4) 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?

答：

（1）、文件块采用多级索引，一个 LFS 的块大小为 4KB，那么 10 个直接指针共可索引 40KB 的文件大小，一级间接指针可索引$1024 \times 4KB = 4MB$的文件大小，二级索引指针可索引$1024 \times 1024 \times 4KB = 4GB$的文件大小，由于文件 foo 的大小为 10MB，因此文件 foo 的文件块索引结构包括 10 个直接指针，1 个一级指针和 1 个二级指针，且 2 级指针所指向的二级索引块只使用了 3 个项（共 1024 个）。

（2）、首先需要读取改文件块的 inode 信息（尚未存在于内存中），即需要读取 CR 块找到 Imap 块再找到 inode 块，并读取二级索引（共两块，为了判断是否已分配数据块），因此这将产生 5 次 I/O 读操作。

按照 LFS 的设计，写该块将在新的位置进行写操作(假设逻辑磁盘块为 Aj)，并且一次需要写入三个块，包括数据块，inode 块和 imap 块，由于存在 segment，此时只产生 1 次 I/O 操作

其磁盘块地址是顺序的，也即:

D[2560]	索引指针更改原本的 2560 块的地址 Aj+2 I[k]	map[k]:Aj+3 Imap

同时还需要更改 CR 中 Imap 块的信息，CR 位于**磁盘首尾部分**，因此还需要 2 次 I/O 操作。总共需要 8 次 I/O。

（3）、FFS 和普通的文件系统一样，但是其可用缩短寻道时间，因此除了读取 inode 块（假设尚未存在于内存中）和两个索引块之外也需要改写对应的数磁盘块即（Ai）,然后改写该 CG 的 inode 块，如果该文件位于多个 CG 中，那么还需要同步其他 CG 中的 inode 块，因此至少需要 5 次 I/O 操作。

（4）、首先需要修改数据块，由于采用元数据日志，日志不批量提交，每次日志写操作进行完后就需将日志中的更新落盘。

则只需要依次记录 inode 块信息，同时在写入 inode 块信息前需要先写入 TxB，写完元数据块后还需再写入 TxE 块。

因此读入 inode，读入二级间址块、一级间址块，需要 3 次 I/O 读操作

写该块，写 TxB，写 inode 块，TxE，共 4 次 I/O 写操作

之后还需要修改 inode 块，1 次 I/O 写操作;删除日志块，由于日志块的内容在磁盘中连续，只需要 1 次 I/O 写操作。总共为 9 次 I/O 操作。