作业 14

14.1 现有一个文件系统，它的文件块索引采用多级间址。该文件系统的 inode，包含 10 个直接指针，1 个一级间址指针，1 个二级间址指针和 1 个三级间址指针。假设文件块大小为 4KB，每个文件块对应的磁盘块地址为 4B。取 1K=1024。

1) 请问该索引结构能够索引的最大文件是多大?

2) 请问一个 1GB 的文件需要几级间址?它总共有多少间址块?其中，各级间址块分别是多少? 如何找到第 20,000 块?

1、10 个直接指针可以索引 10 个块，1 个一级间址指针可以索引 1024（4KB/4B）个块，1 个二级间址指针可以索引 1024*1024 个块，1 个三级间址指针可以索引 1024*1024*1024 个块。因此，该索引结构能够索引的最大文件大小为：$4KB \times (10 + 2^{10} + 2^{20} + 2^{30}) = 40KB + 4MB + 4GB + 4TB.$

2、由上题，1 个 1GB 的文件需要二级间址。注意到，inode 还有一个一级间接指针，共可索引 4MB 的内存大小，这里直接索引指针的数据块太小，忽略不计。考虑上这一点的话.

由一个一级间址块可以索引到 1024 个二级间址块，一个二级间址块可以索引到 1024 个数据块，即每个二级间址块共可索引到 1024*4KB=4MB 的空间。对于 1GB 的文件，该文件可以用 1 个一级间址块来索引 1024 个数据块，再用一个一级间址来索引到 255 个二级间址块。只需要256×1024个文件块，因此只需要 255 个二级间址块，2 个一级间址块，其中一个是索引到下一级文件块，共 257 个间址块。

要找到第 20,000 块，除去用于直接索引到数据块的一级间址（共索引到 1024 个数据块）和直接索引的 10 个数据块，位于二级索引的数据块共有$20000 - 10 - 1024 = 18966$，首先$\frac{18966}{1024} = 18$，$18966\%1024 = 534$，则第 20000 块即在第 18 个二级间址块上，第 534 个磁盘地址对应的入口地址上。

14.2 某用户 X 刚挂载了一个文件系统（假设此时该文件系统的所有 inode 已被加载到内存），该文件系统使用的磁盘块大小为 4KB，能用到的大小最大为 512MB。随后，该用户执行如下所示程序 A。请分析（请写出分析过程）

1) 当程序 A 打开 fs02.ppt 文件时，文件系统需要从磁盘读取几个磁盘块?

2) 假设该文件系统采用 write through 的缓存策略，当程序 A 完成对 fs02.ppt 的写入操作后，文件系统写几次磁盘块？分别写哪些磁盘块？如果该文件系统采用的是 write back 缓存策略，那么程序 A 在写完 fs02.ppt 还未关闭文件时，文件系统共写入了几个磁盘块?

3) 程序 A 执行完成后，用户 Y 再次运行该程序，当程序 A 打开 fs02.ppt 时，文件系统需要从磁盘读取几个磁盘块?

4) 用户 Y 将程序 A 中打开的文件修改为/home/os23/fs01.ppt，并编译执行程序 A，那么当程序 A 打开 fs01.ppt 时，文件系统需要从磁盘读取几个磁盘块?

注：假设（1）每个目录下的所有条目都只占用 1 个磁盘块；（2）fs01.ppt 和 fs02.ppt 两个文件已在文件系统中存在，且 fs02.ppt 的文件长度超过 1MB。

程序 A 代码如下

```
#define MAX (1024)
char buf[MAX];
int fd = open("/home/os23/fs02.ppt", O_CREAT | O_RDWR, 0666);
int n=0, i=0;
if (fd < 0 ) {
    perror("open");
    exit(-1);
}
for (i = 0; i < MAX; i++) {
    bzero(buf, sizeof(buf));
    sprintf(buf, "%6d\n\n", i);
    n = write(fd, buf, strlen(buf));
    printf("len=%d\n", strlen(buf));
    if (n != strlen(buf)){
        perror("write");
        printf("length=%d, buf=[%s]", strlen(buf), buf);
}}
close(fd);
```

（1）、打开文件时，文件系统需要获知文件对应的磁盘块。该文件系统的 inode 已被加载到内存，程序 A 只需要读取 fs02.ppt 文件的 inode 信息，即读取根目录、home 目录、os23 目录对应的磁盘块，找到 fs02.ppt 的 inode 号，因此共读取 3 个磁盘块。

（2）、根据该程序，buf 最长为 1024，每次都往 buf 里写入 8B 数据（共计 8 个 ASCII 码），实际每次写操作的大小即为 8B.

在 write through 策略中，每次写操作都会直接写入磁盘。程序 A 在循环中进行了 1024 次写操作，一个磁盘块有 4KB。写入的数据块，首先是文件的元数据块，其次是修改 inode 块中的信息，例如说，修改时间和访问时间等，因此还需要加上 1024 次。所以总共需要 2048 次。（考虑文件指针在打开文件时放置在文件开头，否则一次写可能横跨两个磁盘）

在 write back 策略中，写操作首先写入缓存，当缓存满了或者在某些特定的条件下，缓存的数据才会被写入磁盘。page cache 最大为 512MB，因此 1024 次写操作的大小最多为 8KB，所以如果文件未关闭，不会有写入磁盘块。

（3）、如果在用户 Y 运行程序 A 时，fs02.ppt 的元数据和部分数据仍然在缓存中，那么文件系统可能不需要从磁盘读取任何磁盘块。

否则，文件系统需要从磁盘读取 fs02.ppt 的数据块，这个时候，如果未执行写操作前文件的大小较写入的数据块大，假设为 N 磁盘块，那么文件需要读取 N 个磁盘块（打开文件后，文件指针位于文件开头）；否则，如果写入的数据块更大，那么文件需要读取最多 2 个磁盘块。

（4）、打开的文件修改为/home/os23/fs01.ppt，则程序打开 fs01.ppt 时，根目录、home 目录,os23 目录都已经在缓存中，那么此时也不需要读取磁盘块。

14.3 现有一个文件系统，在其使用文件缓存的情况下，某个应用首先创建了一个文件“/home/os23/fs03.pdf”，再向该文件中写入了 4 KB 的数据，请分析该过程需要写几次磁盘块?分别写哪些块? 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 OS23 目录都已存在）

（1）、

创建文件时需要在父目录（"/home/os23/"）的目录项中添加一个新的条目。

那么文件系统需要为新文件分配一个 inode。同时对 inode_map 进行写入，以及 os23 的目录块、os23 的 inode。均可能分别写入一个磁盘块（inode 不在同一块）。

文件系统需要为新文件分配一个数据块来存储写入的 4KB 数据，这需要写入 bmap 磁盘块，将 4KB 的数据写入分配的数据块，这还需要写入一个磁盘块。

所以，总共需要写入 6 个磁盘块。往其中写入时，还需要再修改 inode 的信息，那么总共写了 7 次磁盘块。

（2） 、

- 创建文件的过程中发生宕机，父目录的状态和实际文件系统状态不一致的情况。

  os 的目录块未写回或 os23 的 inode 未写回，但 fs03.pdf 的 inode 已写回，均会出现无法访问到该文件的情况。

  os 的目录块写回且 os23 的 inode 写回，但 fs03.pdf 的 inode 未写回，可以访问到文件，但是无法读取文件。

  如果均已写回，则是正常情况。

  如果均未写回，则该文件未被创建。

- 分配 inode 或数据块的过程中发生宕机，文件系统的元数据（如空闲 inode 列表或空闲数据块列表）和实际文件系统状态不一致。

  空闲 inode 列表可能已经将新文件的 inode 标记为已使用，但是新文件的 inode 可能还没有被写入。这个时候，失去了一个 inode 位置，但数据未写入

  如果 inode 未写回，但是数据已被写入，那么数据不可以被读出。

如果 inode 已写回，但是数据未写入，则数据未被写入，读出的数据无效。

- 写入数据的过程中发生宕机，文件的数据和实际文件系统状态不一致的情况。

  文件的数据可能已被部分写入，但是文件的元数据（如文件大小或修改时间）可能还没有被更新。无法访问到新写入的数据。