OS_Assignment16&17

- 周鹏宇
- 2019K8009929039

1. 一个进程在运行过程中读写文件，它每次只读写 1 个块(块大小为 4KB)，并且在它运 行过程中一共发出 10,000 次读请求和 3,000 次写请求，其中 60%的请求命中文件缓存。

1)如果文件缓存采用 write through，请问最终发到磁盘上的 I/O 请求是多少次?其中读请求和写请求分别是多少?

2)如果文件缓存采用 write back，请问最终发到磁盘上的 I/O 请求是多少次?其中读请求和写请求分别是多少?

对于读而言，二者都是在不命中时才发IO请求，因此发到磁盘上的读请求都是

$10000 \times (1 - 0.6) = 4000$次

a. 对于写穿透策略，只要是写操作，必然要把数据更新到磁盘上，故IO写请求为3000次，与读请求相加。IO请求为7000次。
b. 对于写返回策略，在写一个块后，会将这个块tag中的dirty位置高，而在之后若再次出现对此块的读写操作，才需要将块中的内容更新到磁盘中。因此会发生0-3000次IO写，与读请求相加，IO请求为4000-7000次

2. 在有文件缓存的情况下，挂载一个 FFS 文件系统后，
1) 用户 A 打开文件“/home/OS21/fs01.pdf”，请问需要从磁盘上读几个块?

2) 用 户 A 再次打开“/home/OS21/fs01.pdf”，请问需要从磁盘上读几个块?

3) 用户 B 打开“/home/OS21/fs01.pdf”，请问需要从磁盘上读几个块?

4) 用户 C 打开“/home/OS21/fs03.pdf”，请问需要从磁盘上读几个块? 假设所有目录和文件只一个块。

a. 假设A为初次打开文件，则需要读根目录的inode块和目录块，home的inode块和目录块，OS21的inode块和目录块以及fs01的inode块，共7个

b. 由于有文件缓存，所以无需从磁盘读

c. 在同一设备上不同用户使用的文件缓存是一样的，因此还是无需从磁盘读

d. 需要多读fs03的inode块，故读一个

3. 假设文件系统采用 FFS，块大小为 4KB。执行下面代码:

```
#define MAX (1000)
char buf[MAX];
int fd = open("tmpfile", O_CREAT | O_TRUNC | 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, "%3d\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) 请问这段代码执行成功后，tmpfile 的大小是多少字节?

2) 在没有文件缓存的情况下，总共写几次磁盘，每次多少字节?

3) 在有文件缓 存的情况下，总共写几次磁盘，每次写多少字节?

a. 每次写入3字节的ASCLL和1字节的\n，写1000次，故大小为4KB

b. 没有文件缓存，则有

   a. 更新inode map，写一次磁盘

   b. 新建data map，写一次磁盘

   c. 写data block，每次4字节，写1000次

   d. 写inode块，写一次磁盘

c. 引入文件缓存后，数据可暂时缓存在文件缓存中，在关闭文件时一次性写入4000字节，写一次。

4. 在有文件缓存的情况下，在一个 FFS 文件系统中创建一个文件“/home/OS21/ fs03.pdf”， 需要写几个块?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。

需要写 4 个块：

- inode 位图（新建 `fs03.pdf` 的 inode 块）
- `fs03.pdf` 的 inode 块
- 目录 `OS21` 的数据块
- 目录 `OS21` 的 inode 块

由于引入缓存，这四个块的写入顺序是不确定的，因此宕机时写入未完成的情况需要一一列举：

inode 位图	fs03.pdf 的 inode 块	目录 OS21 的数据块	目录 OS21 的 inode 块	Aa 不一致情况
T	T	T	F	目录 OS21 的 last modified time 等信息与实际信息不一致
T	T	F	T	目录 OS21 的元数据更新，但数据未更新
T	F	T	T	inode 位图更新，但 fs03.pdf 的 inode 块无效

inode 位图	fs03.pdf 的 inode 块	目录 OS21 的数据块	目录 OS21 的 inode 块	Aa 不一致情况
F	T	T	T	fs03.pdf 的 inode 块仍处于空闲状态，与实际被占用的情况不一致；后续可能被错误使用
T	T	F	F	目录 OS21 表项数据、元数据与实际情况不一致。
T	F	F	T	inode 位图更新，但 fs03.pdf 的 inode 块无效；目录 OS21 的元数据更新，但数据未更新
F	F	T	T	目录更新，但 fs03.pdf 的 inode 块和数据未更新
T	F	T	F	inode 位图更新，但 fs03.pdf 的 inode 块无效；目录项更新，但元数据未更新
F	T	F	T	inode 位图与 inode 块不一致；目录 OS21 的元数据更新，但数据未更新
F	T	T	F	inode 位图与 inode 块不一致；目录项更新，但元数据未更新
T	F	F	F	仅 inode 位图更新，莫名奇妙地出现一块被占用；系统其余块都没修改
F	F	F	T	仅目录 OS21 的元数据更新，比如 last modified time 变新了，但除此之外没有任何变化的样纸
F	T	F	F	仅 fs03.pdf 的 inode 块更新，但用它也索引不到文件
F	F	T	F	仅目录 OS21 的数据更新，表项多了一个并不存在的文件
F	F	F	F	即没有开始操作

1. 磁盘上有一个长度为 20KB 的文件 A，如果一个进程打开文件 A，并调用 write 一次性向文件 A 的块 0 和块 1 写入新数据。假设有文件缓存，宕机可能发生在任意时刻。

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?分不同情 况讨论(在什么情况下，文件 A 的内容是什么);

2) 如果文件系统采用元数据日志，并且采用一致性修改，宕机恢复后，文件 A 的内容是什么?分不同情况讨论(在什么情况下，文件 A 的内容是什么)。

a. 若在提交日志前宕机，为write之前的数据；若在提交日志后宕机，则为write后的数据

b. 采用一致性修改并无法解决多个块的问题

   a. 若在写0块以前宕机，则无事发生

   b. 若在写1块以前宕机，会发生data块变而inode不变的问题

   c. 若在日志提交以前宕机，问题同上

   d. 若在日志提交以后宕机，则都变了，没问题

2. LFS 的 imap 采用类似数组的结构，下标是 ino，每项保存 i-node 的磁盘地址，例如， imap[k]记录 ino 为 k 的 i-node 的磁盘地址。假设一个 LFS 的块大小为 4KB，磁盘地址占 4 字节。如果已经分配了 500 万个 i-node，请问

1) 它的 imap 有多少个块?给出计算过程;

2) 它的 CR 有多少个块?给出计算过程;

3) 如何查 ino=654321 的 inode 的磁盘地址?给出查找和计算过程。

1. $N_{imap_b} = \frac{4 \times N_{inode}}{S_b} = \frac{4 \times 5000000}{4000} = 5000$
2. $N_{cr_b} = \frac{4 \times N_{imp_b}}{S_b} = \frac{2 \times 4 \times 5000}{4000} = 10$
3.

   a. `imap[ino]` 位于第 **⌊654321 / 1024⌋ = 638** 个 imap 块上，是该块的 **654321 mod 1024 = 1009** 项

   b. 指向此 imap 块的指针位于第 0 个 CR 块上

   c. 因此在第 0 块 CR 中找到第 637 项，找到第 638 个 imap 块，在该块里找到第 1009 项，得到 inode 磁盘地址

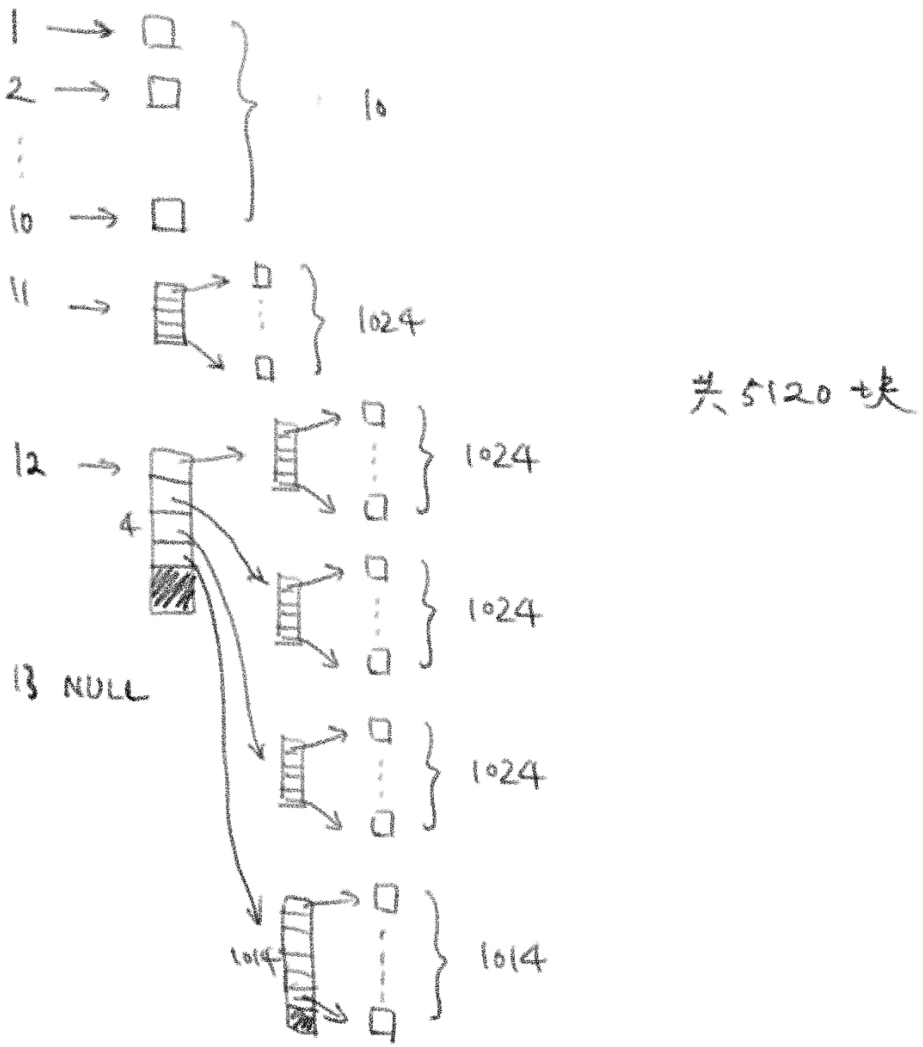
3. 一个 LFS 的块大小为 4KB，segment 大小是 4MB。文件块采用与 FFS 一样的多级索引，每个指向数据块的指针占 4 字节。该 LFS 中已经有一个 20MB 的文 件 foo，

1) 给出文件 foo 的文件块索引结构;

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?给出它们写在磁盘上的顺序。

`foo` 的文件块索引结构

采用多级索引，共需 5K 块，则索引结构如下：



写 `foo` （LFS）

需要写 2560 的新块和 inode 块

全过程为：

1. 读 CR、imap、inode
2. 根据块数读一级、二级间址块，找到第 2560 块
3. 写入该块的新版本、更新 inode 块（一次写入）

故需 6 次 I/O。

写 `foo` （FFS）

题设应为 `foo` 在根目录下。

全过程为：

1. 读根目录块，找到 `foo` 表项，读 `foo` 的 inode 块。
2. 根据块数读一级、二级间址块，找到第 2560 块
3. 写入该块的新版本、更新 inode 块（两次写入）

故需 6 次 I/O。

写 `foo` （元数据日志）

全过程为：

1. 读根目录块，找到 `foo` 表项，读 `foo` 的 inode 块。
2. 根据块数读一级、二级间址块，找到第 2560 块
3. 写入该块的新版本、写 TxB、写 inode 日志、写 TxE、写更新 inode 块、清除日志

共需 10 次 I/O。

1. WAFL 的块大小都为 4KB，指针都为 4 字节，i-node 中有 16 个指针用于文件块索引。 请问

1) WAFL 最大能支持多大的文件?

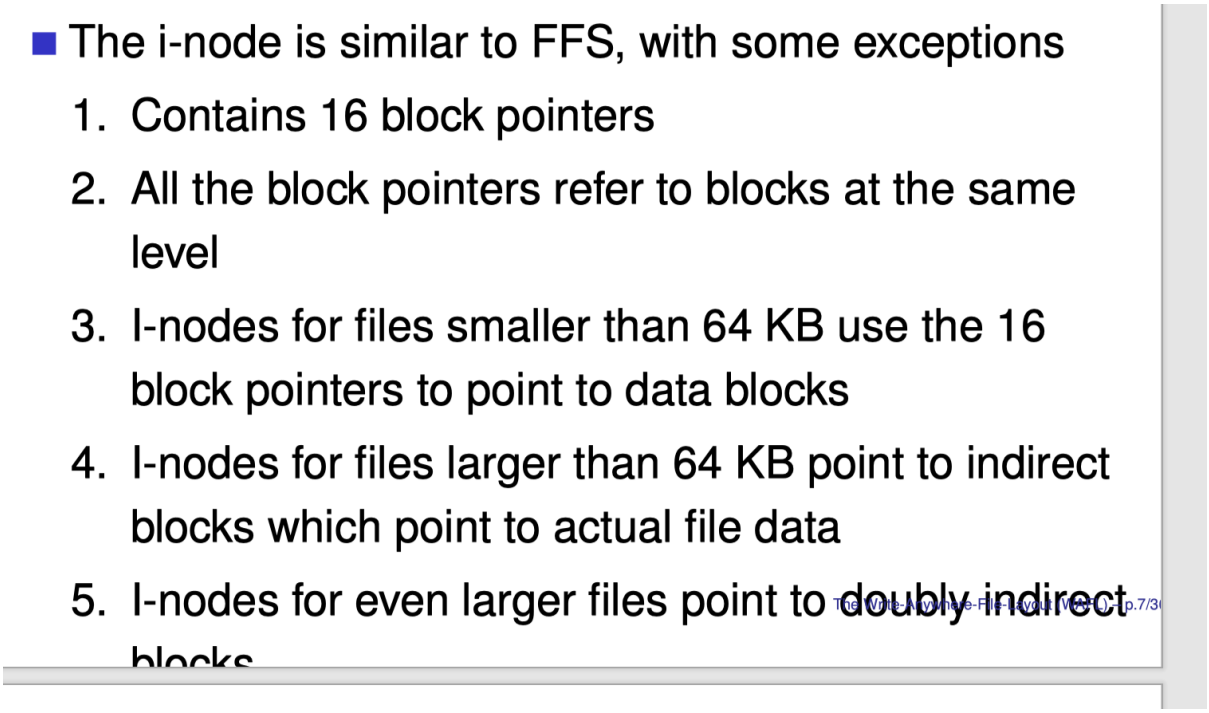
2) 如果采用两级间址的话，最大能索引多大的文件?

3) 对于一个 10GB 的文件， WAFL 如何定位偏移(offset)为 5G 所在的文件块?

WAFL 最大支持文件

前两问不太清晰，大概可以一并回答。

WAFL 最多支持二级间址：



因此最大能索引文件大小为：

$16 \times 1024^2 \times 4K = 64G$

定位到 5G

一级间址无法支持 10G 文件，需都使用二级间址。

10G 文件共需$10 \times 1024 \times 1024K \div 4K = 2560K$块

5G 位于 1280K 块后

第 0 个指针索引 `0~1024K-1` 块

第 1 个指针索引到的 `1024K~1024K+256K-1` 块为 5G 区间

5G offset 为第 1 个指针 → 一级间址块的第 256 项 → 二级间址块的第 0 项极为指向 5G offset 处的指针。

5GB / 4KB = 1310720

1310720 / 1048576 = 1 余 262144

262144 mod 1024 = 256 余 0