1.（1）采用Bitmap位图管理法，用每个bit表示一个盘块的使用情况（0表示空，1表示占用）。16384个磁盘块一共需要16384 / 8 = 2048个Byte = 2KB，恰好可以放在2KB的内存空间里。

（2）采用CSCAN算法，依次访问磁道120，30，50，90。

所以访问中移动的磁道总数= 20 +（120-30）+ 20 + 40 = 170，所以总的寻道时间是170\*1ms = 170ms。磁面转速6000RPM，故平均旋转延迟 = 60\*1000ms/（6000\*2）= 5ms，共访问四个磁道，所以总的旋转延迟= 4\*5ms = 20ms。每个磁道100个扇区，所以读取一个扇区需要（60\*1000ms/6000）/100 = 0.1ms，总的传输时间 = 4\*0.1ms = 0.4ms

所以总时间=170+20+0.4 = 190.4ms

（3）用FCFS调度策略，因为Flash的结构不用考虑寻道时间和旋转延迟，可以直接按IO请求的先后顺序服务。

2.（1）写放大为22个Page

（2）共发生6次擦除操作

（3）发生3次full merge

（4）Log Table：2000->0

Block Table: 1000->8, 2000->4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |  | 4 | 5 | 6 | 7 |  | 8 | 9 | 10 | 11 |  | 12 | 13 | 14 | 15 |
| e |  |  |  |  | e | f | g | h |  | a | b | c | d |  |  |  |  |  |
| Block 0 | | | |  | Block 1 | | | |  | Block 2 | | | |  | Block 3 | | | |

3.

fd2=4, child: c=b

fd2=4, c=a

fd2=4, c=r

或

fd2=4, c=b

fd2=4, child: c=a

fd2=4, c=r

4. 固定大小data block：文件对应5MB/4KB = 1280个data block。由于1024+12<1280，所以需要使用12个direct pointers和2个indirect point，共1280+2=1282个指针，对应的 存储空间=1282\*8 = 10256 bytes

extent：文件分为5个连续的部分，每个连续的部分需要一个指针和extent size，对应12 bytes，所以存储空间= 5\*12 = 60 bytes

5. （1）在bar文件内，首先superblock要读一次，inode bitmap和data bitmap分别读写一次，inode要读写，读取data block时，100KB对应25个page，需要读25个指针。每个指针追加写入1MB数据，共25MB。其中有8MB-52KB的数据由indirect point指向，写入时需要2次IO；剩余17MB+52KB数据则由double indirect pointer指向，写入时需要3次IO。

而在访问bar文件时，要读一次根目录的inode，读一次根目录的data block，读写foo目录的inode，读写foo目录的data block。

综上，共需要1+2+2+2+25+8\*2+18\*3+2+2\*2 = 108次

（2）该命令的第二个参数是文件，会实现重命名的功能，所以读写根目录的inode，读写根目录的data block，读写foo目录的inode，读写foo目录的data block（因为文件的名字存储在它的父目录中，所以需要修改的是foo的data block而不是bar文件）

