36.1

Compute the seek, rotation, and transfer times for the following sets of requests: -a 0, -a 6, -a 30, -a 7,30,8, and finally -a 10,11,12,13.

编写批处理脚本如下

```
./disk.py -a 0 -c
./disk.py -a 6 -c
./disk.py -a 30 -c
./disk.py -a 7,30,8 -c
./disk.py -a 10,11,12,13 -c
```

计算结果如下:

```
-> % ./test.sh
OPTIONS seed 0
OPTIONS addr 0
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['0']

Block:   0  Seek:  0  Rotate:165  Transfer: 30  Total: 195

TOTALS      Seek:  0  Rotate:165  Transfer: 30  Total: 195

OPTIONS seed 0
OPTIONS addr 6
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['6']
```

```
Block:   6  Seek:  0  Rotate:345  Transfer: 30  Total: 375

TOTALS      Seek:  0  Rotate:345  Transfer: 30  Total: 375

OPTIONS seed 0
OPTIONS addr 30
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['30']

Block:  30  Seek: 80  Rotate:265  Transfer: 30  Total: 375

TOTALS      Seek: 80  Rotate:265  Transfer: 30  Total: 375

OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 80  Rotate:220  Transfer: 30  Total: 330
Block:   8  Seek: 80  Rotate:310  Transfer: 30  Total: 420

TOTALS      Seek:160  Rotate:545  Transfer: 90  Total: 795

OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
```

```
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate:320  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 40  Rotate:425  Transfer:120  Total: 585
```

36.2

Do the same requests above, but change the seek rate to different values: -S 2, -S 4, -S 8, -S 10, -S 40, -S 0.1. How do the times change?

编写批处理脚本如下

```
./disk.py -a 10,11,12,13 -S 2 -c
./disk.py -a 10,11,12,13 -S 4 -c
./disk.py -a 10,11,12,13 -S 8 -c
./disk.py -a 10,11,12,13 -S 10 -c
./disk.py -a 10,11,12,13 -S 40 -c
./disk.py -a 10,11,12,13 -S 0.1 -c
```

计算结果如下：

```
-> % ./2test.sh
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 2
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 20  Rotate:340  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 20  Rotate:445  Transfer:120  Total: 585

OPTIONS seed 0
```

```
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 4
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 10  Rotate:350  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 10  Rotate:455  Transfer:120  Total: 585

OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 8
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek:  5  Rotate:355  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek:  5  Rotate:460  Transfer:120  Total: 585

OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 10
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0
```

```
REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek:  4  Rotate:356  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek:  4  Rotate:461  Transfer:120  Total: 585

OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 40
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek:  1  Rotate:359  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek:  1  Rotate:464  Transfer:120  Total: 585

OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 0.1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek:401  Rotate:319  Transfer: 30  Total: 750
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek:401  Rotate:424  Transfer:120  Total: 945
```

36.3

Do the same requests above, but change the rotation rate: -R 0.1, -R 0.5, -R 0.01. How do the times change?

编写批处理脚本如下

```
./disk.py -a 10,11,12,13 -R 0.1 -S 40 -c
./disk.py -a 10,11,12,13 -R 0.5 -S 40 -c
./disk.py -a 10,11,12,13 -R 0.01 -S 40 -c
```

```
-> % ./3test.sh
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 40
OPTIONS rotateSpeed 0.1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:1050  Transfer:300  Total:1350
Block:  11  Seek:  0  Rotate:  0  Transfer:300  Total: 300
Block:  12  Seek:  1  Rotate:  0  Transfer:299  Total: 300
Block:  13  Seek:  0  Rotate:  0  Transfer:300  Total: 300

TOTALS      Seek:  1  Rotate:1050  Transfer:1199  Total:2250

OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 40
OPTIONS rotateSpeed 0.5
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:210  Transfer: 60  Total: 270
Block:  11  Seek:  0  Rotate:  0  Transfer: 60  Total:  60
```

```
Block:  12  Seek:  1  Rotate:719  Transfer: 60  Total: 780
Block:  13  Seek:  0  Rotate:  0  Transfer: 60  Total:  60

TOTALS      Seek:  1  Rotate:929  Transfer:240  Total:1170

OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 40
OPTIONS rotateSpeed 0.01
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:10499  Transfer:3000  Total:13499
Block:  11  Seek:  0  Rotate:  0  Transfer:3001  Total:3001
Block:  12  Seek:  1  Rotate:  0  Transfer:2999  Total:3000
Block:  13  Seek:  0  Rotate:  0  Transfer:3000  Total:3000

TOTALS      Seek:  1  Rotate:10499  Transfer:12000  Total:22500
```

36.4

You might have noticed that some request streams would be better served with a policy better than FIFO. For example, with the request stream -a 7,30,8, what order should the requests be processed in? Now run the shortest seek-time first (SSTF) scheduler (-p SSTF) on the same workload; how long should it take (seek, rotation, transfer) for each request to be served?

运行结果如下：

```
-> % ./disk.py -a 7,30,8  -c
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
```

```
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 80  Rotate:220  Transfer: 30  Total: 330
Block:   8  Seek: 80  Rotate:310  Transfer: 30  Total: 420

TOTALS      Seek:160  Rotate:545  Transfer: 90  Total: 795
-> % ./disk.py -a 7,30,8 -p SSTF -c
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SSTF
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  30  Seek: 80  Rotate:190  Transfer: 30  Total: 300

TOTALS      Seek: 80  Rotate:205  Transfer: 90  Total: 375
```

可以看到使用SSTF的策略后总开销大幅下降 由795下降到375

36.5

Now do the same thing, but using the shortest access-time first (SATF) scheduler (-p SATF). Does it make any difference for the set of requests as specified by -a 7,30,8? Find a set of requests where SATF does noticeably better than SSTF; what are the conditions for a noticeable difference to arise?

```
-> % ./disk.py -a 7,30,8 -p SATF -c
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
```

```
OPTIONS policy SATF
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  30  Seek: 80  Rotate:190  Transfer: 30  Total: 300

TOTALS      Seek: 80  Rotate:205  Transfer: 90  Total: 375
```

可以看到没有变化

尝试后发现对于大部分序列两种调度算法的总用时都是一样的

36.6

You might have noticed that the request stream -a 10,11,12,13 wasn't particularly well handled by the disk. Why is that? Can you introduce a track skew to address this problem (-o skew,where skew is a non-negative integer)? Given the default seek rate, what should the skew be to minimize the total time for this set of requests? What about for different seek rates (e.g.,-S 2, -S 4)? In general, could you write a formula to figure out the skew, given the seek rate and sector layout information?

运行结果如下

```
-> % ./disk.py -a 10,11,12,13 -S 2 -p SATF -c
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 2
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SATF
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
```

```
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  13  Seek: 20  Rotate: 10  Transfer: 30  Total:  60
Block:  12  Seek:  0  Rotate:300  Transfer: 30  Total: 330

TOTALS      Seek: 20  Rotate:415  Transfer:120  Total: 555
```

38.1

Use the simulator to perform some basic RAID mapping tests.Run with different levels (0, 1, 4, 5) and see if you can figure out the mappings of a set of requests. For RAID-5, see if you can figure out the difference between left-symmetric and left-asymmetric layouts. Use some different random seeds to generate different problems than above.

测试结果如下：

```
-> % ./raid.py -n 5 -L 5 -R 20 -5 LS  -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 5
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 20
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:16 size:4096
  read  [disk 0, offset 5]
LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 2]
LOGICAL READ from addr:10 size:4096
  read  [disk 2, offset 3]
LOGICAL READ from addr:15 size:4096
  read  [disk 3, offset 5]
LOGICAL READ from addr:9 size:4096
  read  [disk 1, offset 3]

-> % ./raid.py -n 5 -L 5 -R 20 -5 LA -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 5
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
```

```
ARG randRange 20
ARG level 5
ARG raid5 LA
ARG reverse False
ARG timing False

LOGICAL READ from addr:16 size:4096
  read  [disk 1, offset 5]
LOGICAL READ from addr:8 size:4096
  read  [disk 3, offset 2]
LOGICAL READ from addr:10 size:4096
  read  [disk 2, offset 3]
LOGICAL READ from addr:15 size:4096
  read  [disk 0, offset 5]
LOGICAL READ from addr:9 size:4096
  read  [disk 1, offset 3]
```

AID5 几种布局方式中的“左”、“右”表示校验和（也就是图中的红色的P）如何分布，如图中，“左”的校验和都是从最后一个磁盘开始，依次前推，“右”的校验和从第一个磁盘开始，依次后推。

性能影响：

在实际使用测试中，校验和的左右布局方式对 RAID5 性能没有太大影响。由于实际读写 RAID5 时命令时均匀发到每个磁盘上的，从图中也可以推断出“对称”方式在处理大块值的连续 I/O 读时有更好的性能，实际测试中对随机 I/O 性能也有提升。

需要注意的是上面所说的性能影响相对于 RAID5 中的磁盘个数以及 stripe 条带大小参数来说是比较细微的。

其它说明：

现在很多的RAID卡中都使用了 左不对称 这种 RAID5 布局方式，另外 Linux MD RAID、Windows动态磁盘（LDM）、Veritas Volume Manager（VxVM）等实现中都默认使用了 左不对称 方式。了解 RAID5 的布局方式对 RAID 的性能调试之类有一定帮助哦，HOHO。

除了 Linux MD RAID，其它 RAID5 实现中的布局方式似乎没见到可以让用户选择，Linux MD RAID 可以在使用 mdadm 命令创建时指定 --layout 参数指定布局方式，

38.2

Do the same as the first problem, but this time vary the chunk size with -C. How does chunk size change the mappings?

使用不同的chunk size进行测试 变化结果如下：

```
-> % ./raid.py -n 5 -L 0 -C 8192 -R 20 -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 8192
ARG numRequests 5
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 20
```

```
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:16 size:4096
  read  [disk 0, offset 4]

LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:10 size:4096
  read  [disk 1, offset 2]

LOGICAL READ from addr:15 size:4096
  read  [disk 3, offset 3]

LOGICAL READ from addr:9 size:4096
  read  [disk 0, offset 3]
-> % ./raid.py -n 5 -L 0 -C 16384 -R 20 -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 16384
ARG numRequests 5
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 20
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:16 size:4096
  read  [disk 0, offset 4]

LOGICAL READ from addr:8 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:10 size:4096
  read  [disk 2, offset 2]

LOGICAL READ from addr:15 size:4096
  read  [disk 3, offset 3]

LOGICAL READ from addr:9 size:4096
  read  [disk 2, offset 1]
```

38.3

Do the same as above, but use the -r flag to reverse the nature of each problem.

使用 -r 参数运行结果如下：

```
-> % ./raid.py -n 5 -L 0 -R 20 -r  -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 5
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 20
ARG level 0
ARG raid5 LS
ARG reverse True
ARG timing False

LOGICAL READ from addr:16 size:4096
  read  [disk 0, offset 4]

LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:10 size:4096
  read  [disk 2, offset 2]

LOGICAL READ from addr:15 size:4096
  read  [disk 3, offset 3]

LOGICAL READ from addr:9 size:4096
  read  [disk 1, offset 2]
```

38.4

Now use the reverse flag but increase the size of each request with the -S flag. Try specifying sizes of 8k, 12k, and 16k, while varying the RAID level. What happens to the underlying I/O pattern when the size of the request increases? Make sure to try this with the sequential workload too (-W sequential); for what request sizes are RAID-4 and RAID-5 much more I/O efficient?

增加每次request的大小 测试结果如下：

```
-> % ./raid.py -n 5 -L 4 -R 20 -r -S 8k -W seq -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
```

```
ARG numRequests 5
ARG reqSize 8k
ARG workload seq
ARG writeFrac 0
ARG randRange 20
ARG level 4
ARG raid5 LS
ARG reverse True
ARG timing False

LOGICAL READ from addr:0 size:8192
  read  [disk 0, offset 0]     read  [disk 1, offset 0]
LOGICAL READ from addr:2 size:8192
  read  [disk 2, offset 0]     read  [disk 0, offset 1]
LOGICAL READ from addr:4 size:8192
  read  [disk 1, offset 1]     read  [disk 2, offset 1]
LOGICAL READ from addr:6 size:8192
  read  [disk 0, offset 2]     read  [disk 1, offset 2]
LOGICAL READ from addr:8 size:8192
  read  [disk 2, offset 2]     read  [disk 0, offset 3]
-> % ./raid.py -n 5 -L 4 -R 20 -r -S 12k -W seq -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 5
ARG reqSize 12k
ARG workload seq
ARG writeFrac 0
ARG randRange 20
ARG level 4
ARG raid5 LS
ARG reverse True
ARG timing False

LOGICAL READ from addr:0 size:12288
  read  [disk 0, offset 0]     read  [disk 1, offset 0]     read  [disk 2, offset
0]
LOGICAL READ from addr:3 size:12288
  read  [disk 0, offset 1]     read  [disk 1, offset 1]     read  [disk 2, offset
1]
LOGICAL READ from addr:6 size:12288
  read  [disk 0, offset 2]     read  [disk 1, offset 2]     read  [disk 2, offset
2]
LOGICAL READ from addr:9 size:12288
  read  [disk 0, offset 3]     read  [disk 1, offset 3]     read  [disk 2, offset
3]
LOGICAL READ from addr:12 size:12288
  read  [disk 0, offset 4]     read  [disk 1, offset 4]     read  [disk 2, offset
4]
-> % ./raid.py -n 5 -L 4 -R 20 -r -S 16k -W seq -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 5
ARG reqSize 16k
ARG workload seq
```

```
ARG writeFrac 0
ARG randRange 20
ARG level 4
ARG raid5 LS
ARG reverse True
ARG timing False

LOGICAL READ from addr:0 size:16384
  read  [disk 0, offset 0]     read  [disk 1, offset 0]     read  [disk 2, offset
0]     read  [disk 0, offset 1]
LOGICAL READ from addr:4 size:16384
  read  [disk 1, offset 1]     read  [disk 2, offset 1]     read  [disk 0, offset
2]     read  [disk 1, offset 2]
LOGICAL READ from addr:8 size:16384
  read  [disk 2, offset 2]     read  [disk 0, offset 3]     read  [disk 1, offset
3]     read  [disk 2, offset 3]
LOGICAL READ from addr:12 size:16384
  read  [disk 0, offset 4]     read  [disk 1, offset 4]     read  [disk 2, offset
4]     read  [disk 0, offset 5]
LOGICAL READ from addr:16 size:16384
  read  [disk 1, offset 5]     read  [disk 2, offset 5]     read  [disk 0, offset
6]     read  [disk 1, offset 6]
```

通过观察上面的几个测试结果可以发现：
相比于RAID1，RAID4/5在更大的请求中工作效率更高，一般超过N/2个磁盘。

38.5

Use the timing mode of the simulator (-t) to estimate the performance of 100 random reads to the RAID, while varying the RAID levels, using 4 disks.

使用timing mode运行模拟结果如下：

```
-> % ./raid.py -n 100 -L 0 -R 1000 -W rand -t -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 1000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  93.02  I/Os:    24 (sequential:0 nearly:13 random:11)
disk:1  busy: 100.00  I/Os:    25 (sequential:1 nearly:11 random:13)
disk:2  busy:  98.54  I/Os:    26 (sequential:0 nearly:16 random:10)
disk:3  busy:  96.45  I/Os:    25 (sequential:0 nearly:17 random:8)
```

```
STAT totalTime 177.6
> % ./raid.py -n 100 -L 1 -R 1000 -W rand -t -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 1000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  86.52  I/Os:    24 (sequential:0 nearly:8 random:16)
disk:1  busy: 100.00  I/Os:    26 (sequential:0 nearly:11 random:15)
disk:2  busy:  91.91  I/Os:    25 (sequential:0 nearly:7 random:18)
disk:3  busy:  93.28  I/Os:    25 (sequential:0 nearly:8 random:17)

STAT totalTime 226.3
-> % ./raid.py -n 100 -L 4 -R 1000 -W rand -t -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 1000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  96.95  I/Os:    32 (sequential:0 nearly:17 random:15)
disk:1  busy:  99.41  I/Os:    34 (sequential:0 nearly:20 random:14)
disk:2  busy: 100.00  I/Os:    34 (sequential:0 nearly:22 random:12)
disk:3  busy:   0.00  I/Os:     0 (sequential:0 nearly:0 random:0)

STAT totalTime 235.9
-> % ./raid.py -n 100 -L 5 -R 1000 -W rand -t -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 1000
ARG level 5
ARG raid5 LS
```

```
ARG reverse False
ARG timing True


disk:0  busy:  91.33  I/Os:    24 (sequential:0 nearly:11 random:13)
disk:1  busy:  97.52  I/Os:    25 (sequential:1 nearly:10 random:14)
disk:2  busy: 100.00  I/Os:    26 (sequential:0 nearly:14 random:12)
disk:3  busy:  97.97  I/Os:    25 (sequential:0 nearly:12 random:13)

STAT totalTime 197.2
```

40.1

Run the simulator with some different random seeds (say 17, 18, 19, 20), and see if you can figure out which operations must have taken place between each state change.

编写批处理脚本如下

```
./vsfs.py -s 5
./vsfs.py -s 6
./vsfs.py -s 7
```

运行结果如下

```
-> % ./1test.sh
ARG seed 5
ARG numInodes 8
ARG numData 8
ARG numRequests 10
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (t,1)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
```

```
data bitmap   11000000
data          [(.,0) (..,0) (t,1) (y,2)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:2] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (t,1) (y,2)] [(.,1) (..,0) (c,2)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:2 r:2] [] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (t,1) (y,2)] [(.,1) (..,0) (c,2)] [o] [] [] [] [] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:4] [d a:1 r:2] [f a:2 r:2] [d a:3 r:2] [] [] [] []
data bitmap   11110000
data          [(.,0) (..,0) (t,1) (y,2) (v,3)] [(.,1) (..,0) (c,2)] [o] [(.,3)
(..,0)] [] [] [] []

Which operation took place?

inode bitmap  11111000
inodes        [d a:0 r:5] [d a:1 r:2] [f a:2 r:2] [d a:3 r:2] [d a:4 r:2] [] []
[]
data bitmap   11111000
data          [(.,0) (..,0) (t,1) (y,2) (v,3) (j,4)] [(.,1) (..,0) (c,2)] [o]
[(.,3) (..,0)] [(.,4) (..,0)] [] [] []

Which operation took place?

inode bitmap  11111000
inodes        [d a:0 r:5] [d a:1 r:2] [f a:2 r:1] [d a:3 r:2] [d a:4 r:2] [] []
[]
data bitmap   11111000
data          [(.,0) (..,0) (t,1) (y,2) (v,3) (j,4)] [(.,1) (..,0)] [o] [(.,3)
(..,0)] [(.,4) (..,0)] [] [] []

Which operation took place?

inode bitmap  11011000
inodes        [d a:0 r:5] [d a:1 r:2] [] [d a:3 r:2] [d a:4 r:2] [] [] []
data bitmap   11011000
data          [(.,0) (..,0) (t,1) (v,3) (j,4)] [(.,1) (..,0)] [] [(.,3) (..,0)]
[(.,4) (..,0)] [] [] []

Which operation took place?

inode bitmap  11111000
inodes        [d a:0 r:5] [d a:1 r:2] [f a:-1 r:1] [d a:3 r:2] [d a:4 r:2] [] []
[]
data bitmap   11011000
```

```
data          [(.,0) (..,0) (t,1) (v,3) (j,4)] [(.,1) (..,0)] [] [(.,3) (..,0)]
[(.,4) (..,0) (a,2)] [] [] []

Which operation took place?

inode bitmap  11111100
inodes        [d a:0 r:6] [d a:1 r:2] [f a:-1 r:1] [d a:3 r:2] [d a:4 r:2] [d a:2
r:2] [] []
data bitmap   11111000
data          [(.,0) (..,0) (t,1) (v,3) (j,4) (u,5)] [(.,1) (..,0)] [(.,5)
(..,0)] [(.,3) (..,0)] [(.,4) (..,0) (a,2)] [] [] []

ARG seed 6
ARG numInodes 8
ARG numData 8
ARG numRequests 10
ARG reverse False
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:3] [d a:1 r:2] [] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1)] [(.,1) (..,0)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1)] [(.,1) (..,0) (g,2)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1) (o,3)] [(.,1) (..,0) (g,2)] [] [] [] [] [] []

Which operation took place?

inode bitmap  11110000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:-1 r:2] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (g,1) (o,3) (p,3)] [(.,1) (..,0) (g,2)] [] [] [] []
[] []

Which operation took place?

inode bitmap  11110000
```

```
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:2] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (g,1) (o,3) (p,3)] [(.,1) (..,0) (g,2)] [r] [] [] []
[] []

Which operation took place?

inode bitmap  11111000
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:2] [f a:-1 r:1] [] []
[]
data bitmap   11100000
data          [(.,0) (..,0) (g,1) (o,3) (p,3) (v,4)] [(.,1) (..,0) (g,2)] [r] []
[] [] [] []

Which operation took place?

inode bitmap  11111100
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:2] [f a:-1 r:1] [f
a:-1 r:1] [] []
data bitmap   11100000
data          [(.,0) (..,0) (g,1) (o,3) (p,3) (v,4) (e,5)] [(.,1) (..,0) (g,2)]
[r] [] [] [] [] []

Which operation took place?

inode bitmap  11111100
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:3] [f a:-1 r:1] [f
a:-1 r:1] [] []
data bitmap   11100000
data          [(.,0) (..,0) (g,1) (o,3) (p,3) (v,4) (e,5) (y,3)] [(.,1) (..,0)
(g,2)] [r] [] [] [] [] []

Which operation took place?

inode bitmap  11111100
inodes        [d a:0 r:3] [d a:1 r:2] [f a:-1 r:1] [f a:2 r:3] [f a:3 r:1] [f
a:-1 r:1] [] []
data bitmap   11110000
data          [(.,0) (..,0) (g,1) (o,3) (p,3) (v,4) (e,5) (y,3)] [(.,1) (..,0)
(g,2)] [r] [j] [] [] [] []

Which operation took place?

inode bitmap  11111110
inodes        [d a:0 r:3] [d a:1 r:3] [f a:-1 r:1] [f a:2 r:3] [f a:3 r:1] [f
a:-1 r:1] [d a:4 r:2] []
data bitmap   11111000
data          [(.,0) (..,0) (g,1) (o,3) (p,3) (v,4) (e,5) (y,3)] [(.,1) (..,0)
(g,2) (r,6)] [r] [j] [(.,6) (..,1)] [] [] []

ARG seed 7
ARG numInodes 8
ARG numData 8
ARG numRequests 10
ARG reverse False
ARG printFinal False

Initial state
```

```
inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:-1 r:1] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (r,1)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0) (r,1) (z,2)] [] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:-1 r:1] [f a:1 r:1] [] [] [] [] []
data bitmap   11000000
data          [(.,0) (..,0) (r,1) (z,2)] [h] [] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
inodes        [d a:0 r:2] [f a:2 r:1] [f a:1 r:1] [] [] [] [] []
data bitmap   11100000
data          [(.,0) (..,0) (r,1) (z,2)] [h] [i] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:2 r:1] [] [] [] [] [] []
data bitmap   10100000
data          [(.,0) (..,0) (r,1)] [] [i] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:2 r:2] [] [] [] [] [] []
data bitmap   10100000
data          [(.,0) (..,0) (r,1) (h,1)] [] [i] [] [] [] [] []

Which operation took place?

inode bitmap  11000000
inodes        [d a:0 r:2] [f a:2 r:1] [] [] [] [] [] []
data bitmap   10100000
data          [(.,0) (..,0) (h,1)] [] [i] [] [] [] [] []

Which operation took place?

inode bitmap  11100000
```

```
inodes          [d a:0 r:2] [f a:2 r:1] [f a:-1 r:1] [] [] [] [] []
data bitmap     10100000
data            [(.,0) (..,0) (h,1) (p,2)] [] [i] [] [] [] [] []

Which operation took place?

inode bitmap    11100000
inodes          [d a:0 r:2] [f a:2 r:2] [f a:-1 r:1] [] [] [] [] []
data bitmap     10100000
data            [(.,0) (..,0) (h,1) (p,2) (t,1)] [] [i] [] [] [] [] []

Which operation took place?

inode bitmap    11110000
inodes          [d a:0 r:2] [f a:2 r:2] [f a:-1 r:1] [f a:-1 r:1] [] [] [] []
data bitmap     10100000
data            [(.,0) (..,0) (h,1) (p,2) (t,1) (u,3)] [] [i] [] [] [] [] []
```

Now do the same, using different random seeds (say 21, 22, 23, 24), except run with the -r flag, thus making you guess the state change while being shown the operation. What can you conclude about the inode and data-block allocation algorithms, in terms of which blocks they prefer to allocate?

使用seed 21,22,23,24 和 -r flag

运行结果如下

```
> % ./2test.sh
ARG seed 21
ARG numInodes 8
ARG numData 8
ARG numRequests 10
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/o");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/b");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/o/q");
```

```
  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/b", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/o/q", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/o/j");

  State of file system (inode bitmap, inodes, data bitmap, data)?

unlink("/b");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/o/j", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/o/x");

  State of file system (inode bitmap, inodes, data bitmap, data)?

mkdir("/o/t");

  State of file system (inode bitmap, inodes, data bitmap, data)?

ARG seed 22
ARG numInodes 8
ARG numData 8
ARG numRequests 10
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

creat("/z");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

unlink("/z");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/y");
```

```
  State of file system (inode bitmap, inodes, data bitmap, data)?

link("/y", "/s");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/e");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/u");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/q");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/q", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

mkdir("/d");

  State of file system (inode bitmap, inodes, data bitmap, data)?

ARG seed 23
ARG numInodes 8
ARG numData 8
ARG numRequests 10
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/c");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/c/t");

  State of file system (inode bitmap, inodes, data bitmap, data)?

unlink("/c/t");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/c/q");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/c/j");
```

```
  State of file system (inode bitmap, inodes, data bitmap, data)?

link("/c/q", "/c/h");

  State of file system (inode bitmap, inodes, data bitmap, data)?

unlink("/c/h");

  State of file system (inode bitmap, inodes, data bitmap, data)?

link("/c/q", "/r");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/c/q", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/g");

  State of file system (inode bitmap, inodes, data bitmap, data)?

ARG seed 24
ARG numInodes 8
ARG numData 8
ARG numRequests 10
ARG reverse True
ARG printFinal False

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/z");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/z/t");

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/z/z");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/z/z", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

creat("/y");

  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/y", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);
```

```
  State of file system (inode bitmap, inodes, data bitmap, data)?

fd=open("/z/t", O_WRONLY|O_APPEND); write(fd, buf, BLOCKSIZE); close(fd);

  State of file system (inode bitmap, inodes, data bitmap, data)?

link("/y", "/x");

  State of file system (inode bitmap, inodes, data bitmap, data)?

unlink("/x");

  State of file system (inode bitmap, inodes, data bitmap, data)?

mkdir("/z/w");

  State of file system (inode bitmap, inodes, data bitmap, data)?
```

通过上面的几个测试结果可以分析得出结论

喜欢从序列的前面往后分配。

Now reduce the number of data blocks in the file system, to very low numbers (say two), and run the simulator for a hundred or so requests. What types of files end up in the file system in this highly-constrained layout? What types of operations would fail?

运行命令 ./vsfs.py -d 2 -n 100 -p -c -s 21

```
-> % ./vsfs.py -d 2 -n 100 -p -c -s 21
ARG seed 21
ARG numInodes 8
ARG numData 2
ARG numRequests 100
ARG reverse False
ARG printFinal True

Initial state

inode bitmap  10000000
inodes        [d a:0 r:2] [] [] [] [] [] [] []
data bitmap   10
data          [(.,0) (..,0)] []

mkdir("/o");
File system out of data blocks; rerun with more via command-line flag?
```

Some empty files and links in the root directory.

makedir() and write() fail, but shouldn't. The last data block seems can't be used.

可以看出最后一个数据块模拟器要求不能使用

Now do the same, but with inodes. With very few inodes, what types of operations can succeed? Which will usually fail? What is the final state of the file system likely to be?

执行 ./vsfs.py -i 2 -n 100 -p -c -s 21

```
-> % ./vsfs.py -i 2 -n 100 -p -c -s 21
ARG seed 21
ARG numInodes 2
ARG numData 8
ARG numRequests 100
ARG reverse False
ARG printFinal True

Initial state

inode bitmap  10
inodes        [d a:0 r:2] []
data bitmap   10000000
data          [(.,0) (..,0)] [] [] [] [] [] [] []

mkdir("/o");
File system out of inodes; rerun with more via command-line flag?
```

All operations except unlink() will fail. Only the first inode is available.

Change to three inodes ends up with an empty directory or a small file.

创建文件，创建目录都不行。

模拟器似乎也要求必须剩下一个inode块不能被使用。