CS 481 Problem Set 2

David Lochridge

May 9, 2014

1 Hard Disks

1.1

For -a 0:

Seek Time: 0

Rotate Time: 165

Transfer Time: 30

For -a 6:

Seek Time: 0

Rotate Time: 345

Transfer Time: 30

For -a 30:

Seek Time: 80

Rotate Time: 265

Transfer Time: 30

For -a 7,30,8:

Seek Time: 160

Rotate Time: 545

Transfer Time: 90

For -a 10,11,12,13:

Seek Time: 40

Rotate Time: 425

Transfer Time: 120

1.2

As the seek rate is adjusted, the seek time decreases, while the rotate time is increased. The Transfer time is unaffected.

1.3

As the rotation rate is adjusted, the rotation and transfer times are reduced, while the seek time is unaffected.

1.4

In the case of the 7,30,8 memory requests, the values should be accessed in sequential order (7,8,30)

By running the SSTF option, we see:

Seek Time: 80

Rotate Time: 205

Transfer Time: 90

Which reduces both the rotation time and seek time.

1.5

In the case of the 7,30,8 requests, there is not improvement of using SATF.

The conditions in which the SATF scheduler is beneficial are those in which requests are made across the disk, and additional requests are made while the disk is being read.

1.6

Introducing a skew less than 10, but greater than 1 improves the performance of the rotation time. The ideal skew for overall performance improvement on the default seek rate is 2.

For higher seek rates, the ideal skew is 1, and for lower rates, the ideal skew is higher (.1's ideal, for example, is 38)

A rough estimate of the ideal skew rate seems to be the number of consecutive memory requests divided by the seek rate, minus 2, with a minimum of 1.

1.7

The bandwidth for each sector is as follows, in sector read per unit time:

Inner track: 1/10

Middle track: 1/20

Outer track: 1/30

2 File System Implementation

2.1

The operations, in order, for seed 17 are:

mkdir()

creat()

creat()

link()

mkdir()

creat()

unlink()

mkdir()

open()/write()/close()

creat()

2.2

In general, the algorithms will attempt to allocate the first available memory for the file or folder being created.

2.3

With few blocks, the simulation failed on open/write/close and calls.

The final state of the system is likely to have many folders, files, and links, but little data.

2.4

With few inodes, the simulation failed on mkdir, creat, and link calls.

The final state of the file system is likely to have a flat structure, with few large files.

3 RAID

3.1

Requests are mapped among varied disks in RAID 0, even with adjacent memory locations.

In RAID 1, the disk returning a response is typically the same one that returned a nearby response.

In RAID 4, requests are serviced by the appropriate disk, calculated by offset divided by chuck size mod number of disks.

In RAID 5, requests are serviced in the same manner as RAID 4, accounting for the additional parity.

For large contiguous reads, RAID-5 symmetric outperforms RAID-5 asymmetric.

3.2

In increasing the chunk size:

RAID 0: The mappings are shifted among the disks, with multiple contiguous on the same disk, and the same number on the next one, and so forth.

RAID 1: The mappings are mostly unaffected.

RAID 4: The mappings are shifted among the blocks, with multiple contiguous blocks on each disk

RAID 5: Affected the same way as RAID 4

3.3

In reversing the problem:

RAID 0: The mappings are unaffected.

RAID 1: The mappings are unaffected.

RAID 4: The mappings are unaffected.

RAID 5: The mappings are unaffected.