CS 45: Operating Systems

CLab 10: Disks

DUE by next class

Write your answers in `README.md`. Create a `COLLAB.md` file to keep track of any outside resources you might use. Be sure to push to the repo after class (even if you are not done).

1 Disks

Use `disk.py` to familiarize you with how a modern hard drive works. It has a lot of different options , and unlike most of the other simulations, has a graphical (`-G`) animator to show you exactly what happens when the disk is in action. See the `disk-README.md` for details.

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`.

```
$ python3 disk.py -G -a 0
```

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?
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?
4. FIFO is not always best, e.g., with the request stream `-a 7,30,8`, what or der should the requests be processed in? Run the shortest seek-time first (SSTF) scheduler (`-p SSTF`) on this workload; how long should it take (seek, rotation, transfer) for each request to be served?
5. Now use the shortest access-time first (SATF) scheduler (`-p SATF`). Does it make any difference for `-a 7,30,8` workload? Find a set of requests where SATF outperforms SSTF; more generally, when is SATF better than SSTF?
6. Specify a disk with different density per zone, e.g., `-z 10,20,30`, which specifies the angular difference between blocks on the outer, middle, and inner tracks. Run some random requests (e.g., `-a -1 -A 5,-1,0`, which specifies that random requests should be used via the `-a -1` flag and that five requests ranging from 0 to the max be generated), and compute the seek, rotation, and transfer times. Use different random seeds. What is the bandwidth (in sectors per unit time) on the outer, middle, and inner tracks?

2 RAID

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.

```
$ python3 raid.py -R 20 -n 5 -L 0 -c
```

2. Do the same as the first problem, but this time vary the chunk size with `-C`. How does chunk size change the mappings?
3. Do the same as above, but use the `-r` flag to reverse the nature of each problem.
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?

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