Hard Disk Drives

Annalise Tarhan

March 10, 2021

1 Compute the seek, rotation, and transfer times for the following sets of requests.

-a 0 6.5*30=195

-a 6 12.5*30=375

-a 30 12.5*30=375

-a 7,30,8 1.5*30+11*30+14*30=795

-a 10,11,12,13 4.5*30+30+13*30+30=585

2 Do the same requests as above, but change the seek rate to the following values. How do the times change?

-a 0 No change, 195 in all cases (no seek necessary)

-a 6 No change, 375 in all cases (no seek necessary)

-a 30 Only S=0.1 affects total time, increasing it to 720+12.5*30=1095.

-a 7,30,8 S=4,8,10,40 reduce the time by one full rotation to 435. S=0.1 adds four rotations, taking 2235 total.

-a 10,11,12,13 S=0.1 adds a rotation, increasing time to 945

3 Do the same requests above, but change the rotation rate to the following values. How do the times change?

-a 0 Each changes time by a factor of 1/R. Specifically, R=0.1 takes ten times longer, R=0.5 takes twice as long, and R=.01 takes a hundred times as long.

-a 6 Same as before (ten times longer, twice as long, a hundred times longer)

-a 30 Same as before (ten times longer, twice as long, a hundred times longer)

-a 7,30,8 R=0.1 takes less than ten times as long, increasing only to 4350 from 795 because the seek time doesn't change. R=0.5 takes twice as long as the original, 1590. R=0.01 takes ten times as long as R=0.1, for 43500 total.

-a 10,11,12,13 Same as the first few (ten times longer, twice as long, a hundred times longer)

4 FIFO is not always best. What order should the requests in -a 7,30,8 be processed in? How long should it take for each request to be served?

7,30,8 should be reordered to 7,8,30.

7: 0 seek, 15 rotate, 30 transfer, 45 total
8: 0 seek, 0 rotate, 30 transfer, 30 total
30: 80 seek, 190 rotate, 30 transfer, 300 total

5 Now use the shortest access-time first scheduler. Does it make any difference for the -a 7,30,8 workload? Find a set of requests where SATF outperforms SSTF. More generally, when is SATF better than SSTF?

For the given workload, there is no difference between SATF and SSTF. It would make a difference for a workload where the access time is dominated by the rotation time and focusing only on seek time gives sub-optimal results, such as -a 7,14,35.

6 What goes poorly when the following request stream runs? Try adding track skew to address the problem. Given the default seek rate, what should be the skew to maximize performance? What about for different seek rates?

Even though 12 logically follows 11 and is right next to it, the disk must perform a full rotation in order to reach it after 11. Ideally, it would be accessible immediately after one seek from the end of sector 11. Based on the possible values of S, I would estimate the formula for the optimal skew value to be something like $\lceil 1.35/S \rceil$, since the simulator requires an integer value.

7 Specify a disk with a different density per zone, or the angular difference between blocks on the outer, middle, and inner tracks. Run some random requests and compute the seek, rotation, and transfer times. What is the bandwidth (in sectors per unit time) on the outer, middle, and inner tracks?

On each track, each sector has a rotational space equal to 360 divided by the number of sectors. The given disk, -z 10,20,30, has an outer track with 36 sectors, each allotted 10 degrees of rotational space, a middle track with 18 sectors of 20 degrees each, and an inner track with 12 sectors, each 30 degrees. For the default rotational rate, one time unit per degree, the bandwidth of the outer track is 36/360=.1 sectors per time unit. The middle track's bandwidth is 18/360=.05 sectors per time unit, and the inner track's is 10/360=.023 sectors per time unit.

8 A scheduling window determines how many requests the disk can examine at once. Generate random workloads and see how long the SATF scheduler takes when the scheduling window is changed from one up to the number of requests. How big of a window is needed to maximize performance? When the scheduling window is set to one, does it matter which policy you are using?

When the scheduling window is one, the requests are served in the order they are given, so scheduling policy is irrelevant. Increasing the size of the window reduces the time dramatically at first, but returns diminish quickly. For a workload of 1000 requests, performance is maximized with a window in the range of 7%-31% of the size of the workload.

9 Create a series of requests to starve a particular request, assuming a SATF policy. Given that sequence how does it perform if you use a bounded SATF scheduling approach? In this approach, you specify the scheduling window; the scheduler only moves onto the next window of requests when all requests in the current window have been serviced. Does this solve starvation? How does it perform, as compared to SATF? In general, how should a disk make this trade-off between performance and starvation avoidance?

-a 1,4,7,10,33(starved),1,4,7,10,1,4,7,10,...
Compared to the same request with SATF, BSATF solves starvation but performs significantly worse, since it not only has to eventually reach the distant sector, it has to return from it as well. The disk shouldn't allow starvation and so must bound the window, but it should make the window as large as possible, based on how long the OS can reasonably be expected to wait for any given request.

10 All scheduling policies we have looked at thus far are greedy; they pick the next best option instead of looking for an optimal schedule. Can you find a set of requests in which greedy is not optimal?

-a 34,13,4,8 When accessed in the order given, it takes 435 time units. With SATF, it takes 495.