Introduction to disk scheduling

Eitan Bachmat
Department of computer science
Ben-Gurion U.

What is disk scheduling?

Disk scheduling is the ability of a disk drive to reorder requests which it has recieved and schedule their service so that time spent seeking and waiting (rotational latency) is minimized thus increasing throughput.

This comes at the cost of increased response time. One solution is to batch requests and service the requests in the batch prior to servicing new requests (which are themselves batched).

We shall consider only this batched version, which is difficult enough.

Two natural questions arise

Q1 - Given a set of requests find an optimal reordering that will service all of them in minimal time

Q2 - Given an optimal or other reordering how many disk rotations are needed to service all requests ?

Before attempting to answer these questions we have to characterize the mechanical motions of the disk drive. the most convenient way is to specify the seek function $f(t)$ which gives the radial distance r which the disk head can pass in a given time t. You can find the inverse of f drawn in your hard drive manual.

It turns out that the nature of f has a huge impact on the nature of questions 1 and 2.

One thing is certain f is always a convex function.

Question 1

Andrews bender and Zhang made good progress on question 1. they showed that for general f (or even $f(t) = t^2$) finding the optimal reordering is an NP-Complete problem.

On the positive side they produced an algorithm which produces a total service time which is at most 3/2 times the optimal service time.

When f is linear $f(t) = ct$ they produced an optimal algorithm.

the results of Andrews bender and Zhang say nothing about question 2.

Question 2

Over the past 3 years I have been working (among other things) with various collaborators, Danny berend, Vladimir Braverman and Luba Sapir, all from BGU, on question 2 in a probabilistic setting.

We assume that different areas of the disk may have different popularity, thus each area in the disk, specified by coordinates (r, θ) denoting the radial and angular position respectively, is equiped with a popularity index $p(r, \theta)$. Requests are then generated by sampling the density function $p(r, \theta)$.

The following is a summary of the results thus far. L_n is the random variable counting disk rotations needed to service n requests.

The answers and our state of knowledge vary wildly with different assumptions on f.

Theorem 0.1 *For special seek functions f and distributions p, the variables L_n have an asymptotic behavior as follows:*

(i) *If $f(t) = ct$ is linear and p is uniform, then w.h.p*

$$\frac{L_n - \frac{2}{c}\sqrt{n} - (1/4)(2c)^{1/6} n^{1/6} \log^{2/3} n}{n^{1/6}(\log n)^{2/3}} \longrightarrow 0.$$

(ii) *If $f(t) = ct^a$ with $a > 1$ and p is uniform, then there exist constants C_1, C_2 such that w.h.p such that*

$$C_1 c^{-1/(a+1)} n^{a/(a+1)} \leq L_{n,a} \leq C_2 c^{-1/(a+1)} n^{a/(a+1)}.$$

(iii) *If $f(t) = \max(c(t-b), 0)$ and p is uniform then w.h.p*

$$L_n \leq bn + n^{2/3} \log n.$$

(iv) *If $c = f'(0) > 0$ and $p(r,t)$ is arbitrary, then w.h.p.*

$$\frac{L_n - \sqrt{2} C_p \sqrt{n}}{\sqrt{n}} \longrightarrow 0,$$

where $C_p = \max_\phi \int_0^1 \sqrt{p(r, \phi(r))(1 - c\phi'^2(r))} dr$, where ϕ ranges over all functions with $|\phi'| \leq 1/c$. If $p(r, \theta)$ is a function of r only, then $C_p = \int_0^1 \sqrt{p(r)} dr$.

In the rest of the talk I will concentrate on the case of f linear and $p(r, \theta)$ arbitrary. In reality p will only depend on r but it is instructive to consider the general case.

Curved space-time

Recall that Andrews, bender and Zhang presented an optimal reordering algorithm when f is linear. It turns out that their algorithm is a special version of a well known algorithm, patience sort, which originally was designed to manually sort cards.

Among other things patience sort decomposes a set of points in the plane into a minimal number of decreasing subsequences and at the same time finds a maximal increasing subsequence.

The algorithm is nearly trivial, peel off all points which have nothing left and bottom from them. Repeat until nothing is left. Each time a point is not peeled of have a pointer point from the point to a witness which did not

allow it to be peeled off.

The peeled layers are the minimal decomposition into decreasing subsequences while the trail of pointers from any element of the last layer will provide a maximal increasing subsequence. the relation with the algorithm of Andrews, bender and Zhang is obtained by rotating the (r, θ) plane by 45 degrees

The length of the longest increasing subsequence is the number of required disk rotations in the algorithm of Andrews, bender and Zhang.

What is the length of the longest increasing subsequence given a general distribution $p(r, t)$?

A theorem of Deuschel and Zeitouni states that we should calculate the longest increasing subsequence along a given path and optimize over all paths. For any given path the longest increasing subsequence is obtained by concatenating many local increasing subsequences along division

points of the curve.

What are the local contributions?

In the disk picture they are given by $\sqrt{2p(r,t)(r^2-t^2)N}$

The important thing is $\sqrt{p(r,t)(r^2-t^2)}$, this is the length element of a Lorenzian geometry.

We can now reinterpret the theorem of Deuschel Zeitouni as saying that the longest increasing subsequence (=number of rotations) is the length of the longest geodesic in the Lorentzian metric (= curved space time) given by $ds^2 = p(r,t)(r^2-t^2)$.

thus the pointers generated in the peel off process point comprise a geodesic flow in the curved space time between the inner radius of the disk and the outer radius.

The requests which are serviced during the same disk rotation lie on curves which integrate what is known as the Jacobi field of the geodesic flow and there are several other nice features but we will leave it at that.

Finaly in the more realistic case where $p(r,\theta)$ depends

only on r, things are much simpler, all geodesics in the flow are straight lines of the form $\theta = constant$, the requests which are serviced in the same rotation lie along a straight line of the form $r = constant$ and the number of rotations required is $\int_0^1 \sqrt{2p(r)}dr\sqrt{n}$

Thats all folks