6.854 / 18.415: Advanced Algorithms

Last time: Load balancing and power of two choices

what is max load of n balls in n bins? $O(\frac{\log n}{\log\log n})$

what if each chooses least loaded of two bins? $O(\log\log n)$

Today: Consistent hashing, random trees and distinct elements

remind notation to be consistent with first lecture

Setup: m items to store on n distributed webcaches

Recall our favorite 2-universal hash family

$$h_{a,b}(x) = (ax + b \mod p) \mod n$$

But what if n changes?

(1) mod n $\to$ mod n+1

Almost all items need to move so we are searching in right place

(2) keep n

New cache gets none of load

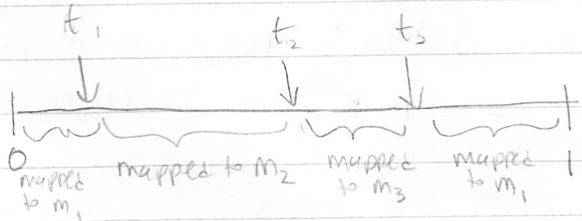
Consistent Hashing: [Karger, Lehman, Leighton, Levine, Lewin, Panigrahy]

(1) map each machine randomly to interval [0, 1]

(2) map each item ″

(3) assign each item to first machine on right (with wrap around)

Example: (colored chalk)



Implementation:

(1) maintain binary search tree whose keys are machine values

To insert item, find ~~predecessor~~ successor add item to machine

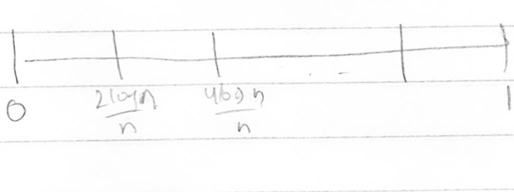
only jobs that need to move, move

To insert machine, find ~~predecessor~~ successor and steal items

To delete machine, find ~~predecessor~~ successor and push items

lemma: w.h.p. no machine owns more than $O\left(\frac{\log n}{n}\right)$ of interval

Pf: Consider intervals of length $\frac{2\log n}{n}$



Probability no machine in a fixed interval

$$\left(1 - \frac{2\ln n}{n}\right)^n = \frac{1}{n^2}$$

gives $\frac{4\log n}{n}$ why?

$\frac{n}{2\ln n}$ intervals

By union bound, probability each interval has a machine is at least $1 - \frac{1}{n}$ ☒

Exercise: Use Chernoff to prove max load (if $m \geq n$) is at most $O\left(\frac{m}{n} \log n\right)$ (don't lose extra log from balls in bins)

lemma: when a machine is added, expected number of items that move is $\frac{m}{n+1}$

Pf: The only items that move are those assigned to machine $n+1$. By symmetry, expected number of such items is $\frac{m}{n+1}$ ☒

How small will the smallest load be? By birthday paradox, expect some machine to own $\frac{1}{n^2}$ of interval

spread out load

Refinement: Each machine is mapped to $\log n$ points in $[0,1]$ interval, owns union of left segments

Relieving Hot Spots on Web

examples from paper: IBM website during Deep Blue - Kasparov

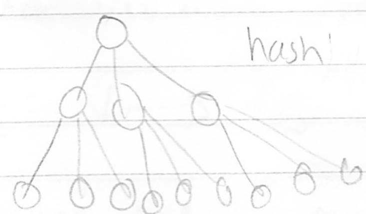
examples from real life: Victoria Secret, Star Wars EP1

<u>setup:</u> single root server, n proxy caches C

If all requests go to root it can't handle them, and web could crash due to traffic. Can we share load?

<u>Random Trees</u>

(1) choose random d-ary tree with n virtual nodes V



hash: $h_i: V \longrightarrow C_i$

why hash? many pages = many roots

(2) each request (if view C_i)

(a) choose random leaf, consider leaf-to-root path P

(b) apply h_i to ~~leaf~~ path, ask machine

(c) If storing page, return it. Otherwise increment page's counter, ask next machine on path. If counter reaches q, store page.

<u>lemma:</u> Each cache not mapped to a leaf is asked for same page at most

$$O\left(\frac{dq \log n}{\log\log n}\right) \text{ times}$$

Pf: Each cache is responsible for at most $O\left(\frac{\log n}{\log\log n}\right)$ virtual nodes. (balls in bins)

Each virtual node (non leaf) can be asked at most dq times before its descendents all store the page $\boxed{\checkmark}$

Inconsistent world: what if everyone has a different view of set of live caches?

$$C_1, C_2 \ldots C_\ell \subseteq C \quad \text{with} \quad \frac{|C_j|}{|C|} \geq \frac{1}{t} \quad \forall j$$

Now where a virtual node is mapped to depends on the particular view

$$h_j : V \to C_j$$

lemma: If there are at most n total requests, each cache is asked for a page at most

$$O(dq t^2 \log^2 n e) \text{ times}$$

part 1: how many virtual nodes can a cache be ~~mapped to by~~ the views? responsible for overall

Pf: Fix cache i, and let it be mapped to t_i. Then consider the interval

$$\underbrace{\overset{I}{\vdash\!\!\longrightarrow\!\!\dashv}}_{\frac{10 t \log n e}{n}} \; t_i$$

of length $\frac{10 t \log n e}{n}$. Fix a view C_j.

The probability no cache in C_j is mapped to I

is at most $\left(1 - \frac{10\, t \log ne}{n}\right)^{|C_j|} \leq \left(1 - \frac{10\, t \log ne}{n}\right)^{\frac{n}{t}} \leq \frac{1}{(en)^{10}}$.

Now union bound over all ℓ views, and with probability at least $1 - \frac{1}{n^{10}}$ we have that each view C_j maps a cache to I

on circle

Thus cache v is not responsible for any virtual node mapped outside of I. Now we can use Chernoff to conclude v is responsible for at most

$$O(t \log ne) \text{ virtual nodes, whp}$$

hash virtual nodes to caches

Then union bound over v, and each virtual node gets at most dq requests as before. $\blacksquare$

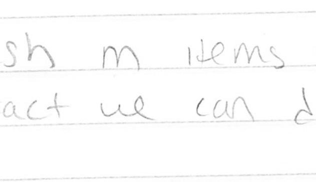
Next time: Distinct elements and Count Min

setup: router wants to estimate number of distinct IP addresses

We could hash m items into m^2 bins, and count. In fact we can do it with $\log m$ bits of storage.

part 2: How many caches can be mapped to a single virtual node over all views?

Now fix a virtual node, and let it be mapped to v_j



and with high probability every view maps a cache to I'. But by Chernoff, at most $O(t \log n \ell)$ caches are mapped to I' in all of C.

In any view, virtual node j can only be mapped to caches that land in I. Thus each virtual node can be responsible for at most $O(q t \log n \ell)$ requests. $\boxtimes$

Now proceed as before