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? $O\left(\frac{\log n}{\log \log n}\right)$
what if each chooses least loaded of two bins? $O(\log \log n)$

Today: Consistent hashing, random trees and distinct elements

Reminder: notation to be consistent with first lecture

Setup: $m$ items to store on $n$ distributed web caches

Recall our favorite 2-universal hash family

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

But what if $n$ changes?

(1) $\text{mod } n \rightarrow \text{mod } n+1$

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: $t_1$ $t_2$ $t_3$ (colored chalk)

```
start  | end  | machine
--------|------|-------
0       | 1/3  | m_1   
1/3     | 2/3  | m_2   
2/3     | 1    | m_3   
```

Implementation:

1. maintain binary search tree whose keys are machine values

   To insert item, find predecessor, add item to machine

   only jobs that need to
   move, many

   To insert machine, find predecessor and steal items

   To delete machine, find predecessor and push items

   **Lemma:** w.h.p. no machine owns more than $O(\frac{\log n}{n})$ of items

   **Proof:** consider intervals of length $2 \frac{\log n}{n}$

   ```
   0 2\frac{\log n}{n} 4\frac{\log n}{n} 1
   ```
Probability no machine in a fixed interval:

\[
(1 - \frac{2\ln n}{n})^n = \frac{1}{n^2}
\]

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

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 \(\frac{m}{n+1}\)

Proof: The only items that move are those assigned to
machine \(n+1\). By symmetry, expected number of items
is \(\frac{m}{n+1}\) \(\Box\)

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

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 EP2

Setup: single 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?

Random Trees

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

hash: \( h: V \rightarrow C \)

(2) Each request (if view C)

(a) Choose random leaf, consider leaf-to-root path p

(b) Apply \( h \) to leaf, ask machine

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

Lemmas: Each cache not mapped to a leaf is asked for same page at most \( O(dq\log n) \) times
Pf: Each cache is responsible for at most \( O\left(\log n / \log \log m\right) \) virtual nodes. (balls in bins) (non-leaf)

Each virtual node can be asked at most \( d \) times before its descendants all store the page.

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

\[ C_1, C_2, \ldots, C_e \subseteq C \text{ with } \frac{|C_j|}{|C|} \geq \frac{1}{e} \forall j \]

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

\[ h_j: V \rightarrow C_j \]

Lemma: If there are at most \( n \) total requests, each cache is asked for a page at most

\[ O\left(d e^2 \log^2 n e \right) \] times

Pf: Fix cache \( c' \), and let it be mapped to \( t_i \).

Then consider the interval

\[
\frac{10 + \log n e}{n} \left[ t_i, t_i + \frac{10 + \log n e}{n} \right]
\]

of length \( 10 + \log n e \). Fix a view \( C_j \).

The probability no cache in \( C_j \) is mapped to \( I \)
is at most \((1 - \frac{1}{\log n})^{\frac{1}{2}} \leq (1 - \frac{1}{\log n})^n \approx \frac{1}{(en)^{10}}\).

Now union bound over all \(L\) views, and with probability at least \(1 - \frac{1}{10^{10}}\) we have that each view \(C_s\) maps a cache to \(I\).

Thus cache \(c_i\) is not responsible for any virtual node mapped outside of \(I\). Now we can use Chernoff to conclude \(c_i\) is responsible for at most \(O(\frac{1}{\log n})\) virtual nodes, w.h.p.

Then union bound over \(c_i\) and each virtual node gets at most \(d_q\) requests as before. \(\Box\)

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.

Now fix a virtual node \(v_j\), and let it be mapped to \(u_j\).

\[
\frac{1}{\log n} \leq \frac{1}{(en)^{10}}.
\]

and with high probability every view maps...
a cache to $I_i$. But by Chernoff, at most $O(t \log n)$ caches are mapped to $I_i$ in all of $C$.

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

Now proceed as before.