Lecture 6: Max Load, Power of 2 choices, Bloom Filters. ①

[When we last left our heroes...]

Analyzing typical worst-case insert/lookup time (under SUHA) when $m = n$ items hashed to n slots.

typical max load $\approx$ when $m = n$ balls thrown into n bins.

Recall: [we ended with] $\Pr[T[0] \text{ gets exactly } k \text{ balls}] \le \frac{1}{k!}$

$\therefore \Pr[T[0] \text{ gets} \ge 10 \text{ balls}] \le \Pr[10] + \Pr[11] + \ldots$ "union bound"

$$\le \frac{1}{10!} + \frac{1}{11!} + \frac{1}{12!} + \ldots$$

$$= \frac{1}{10!}\left(1 + \frac{1}{11} + \frac{1}{12 \cdot 11} + \frac{1}{13 \cdot 12 \cdot 11} + \ldots\right)$$

$$\le \frac{1}{10!}\left(1 + \frac{1}{11} + \frac{1}{11^2} + \frac{1}{11^3} + \ldots\right)$$

$$= \frac{1}{10!}\left(\frac{1}{1 - \frac{1}{11}}\right) \leftarrow \frac{11}{10}$$

$$= \frac{11}{10} \cdot \frac{1}{10!}$$ [[just slightly bigger than $\frac{1}{10!}$]]

"$\therefore$" $\Pr[T[0] \text{ gets} \ge k \text{ balls}] \le \frac{k+1}{k} \cdot \frac{1}{k!} \le 2 \cdot \frac{1}{k!}$ [[more like $\approx 1 \cdot \frac{1}{k!}$, but oh well]]

$\therefore \Pr[T[1] \text{ gets} \ge k \text{ balls}] \le 2 \cdot \frac{1}{k!}$

$\sim T[2] \sim \le \ldots$

$\therefore \Pr[|T[0]| \ge k \text{ OR } |T[1]| \ge k \text{ OR} \ldots \text{OR } |T[n-1]| \ge k] \le$ ← union bound again

$$\Pr[T[0] \ge k] + \Pr[T[1] \ge k] + \ldots + \Pr[T[n-1] \ge k]$$

$$\le \frac{2}{k!} + \frac{2}{k!} + \ldots \frac{2}{k!} = \frac{2}{k!} \cdot n.$$

$\Pr[\text{max load} \ge k]$

Say $n = 1000$. $\Pr[\text{max load} \ge k] \le \frac{2000}{k!}$

$\ge 6] \le \frac{2000}{720}$

$\ge 7] \le \frac{2000}{5040} \approx .4$

$\ge 8] \le \frac{2000}{40320} \approx .05$

Say $n = 10^6$, want to say "$\Pr[\text{max load} \ge k] \le \frac{1}{500}$" (say)

What k? Need $\frac{2 \cdot 10^6}{k!} \le \frac{1}{500} \iff k! \ge 10^9 \iff k \ge 12$.

"For 1M balls $\to$ 1M bins, max load ≤ 11 except w.prob. $\le \frac{1}{500}$"

For general n, need $k! \ge 1000n$. [Inverse factorial??]

How big is $k!$?? {Skip}

\# binary digits $\sim$ $\log_2(k!) = \lg(k \cdot (k-1) \cdot (k-2) \cdots 3 \cdot 2 \cdot 1)$

$$= \lg k + \lg(k-1) + \lg(k-2) + \cdots + \lg(\tfrac{k}{2}) + \cdots + \lg 2 + \lg 1$$

$$\ge \lg k/2 + \lg(k/2) + \lg(k/2) + \cdots + \lg(k/2)$$

$$= \frac{k}{2} \cdot \lg\left(\frac{k}{2}\right) = \frac{k}{2}(\lg k - 1) \ge \frac{k}{2} \cdot \frac{\lg k}{2}$$ [if $\lg k \ge 2 \iff k \ge 4$]

$$= \frac{1}{4} k \lg k$$

$$= \Omega(k \lg k).$$

"$k!$ is a $\Theta(k \log k)$-digit #"

Fact: $22!$ has 22 digits. $\therefore$ $23! \ge 10 \cdot 22!$ has ≥ 23 digits
$24! \ge 10 \cdot 23!$ has ≥ 24 digits.

$k!$ has $\ge k$ digits (ex: more like $\Theta(k \log k)$ digits)

n has $\sim \log_{10} n$ digits. So if $k \ge \log_{10} n + 3$, $k! \ge 1000n$.

$\therefore$ "For n balls $\to$ n bins, max load $\le O(\log n)$ except w.p. $\le \frac{1}{500}$." (ex: $\le O\left(\frac{\log n}{\log \log n}\right)$)

"The power of 2 choices" [[still with "SUHA"]] 3

Idea: • Get hold of two ("independent") hash functions, h_1, h_2: $\{\text{strings}\} \to \{0, 1, \ldots, n-1\}$

• Insert(s): Look at $T[h_1(s)]$, $T[h_2(s)]$ linked lists. Append s to whichever is shorter. [[break ties arbit.]]

• Lookup(s): Look for s in both $T[h_1(s)]$, $T[h_2(s)]$.

Can't really make things more than 2x worse.

Surprise: worst-case time goes way down, typically

Balls & bins ver: To "throw" a ball, pick 2 bins at random, put ball in less-loaded bin.

Theorem: For $m = n$ balls, with high probability, max load is $O(\log \log n)$. [[! Way better than $O(\log n)$ "exponentially"!! For every # n in universe, $\log\log n \le 8$:)]]

[[Proof is a little elaborate, so I'll just give...]]

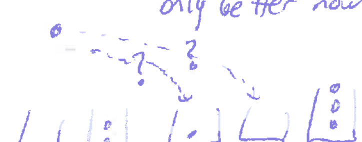
Idea of why: After throwing n balls, let α_2 = fraction of bins with ≥ 2 balls

Claim: $\alpha_2 \leq \frac{1}{2}$. Because if $> \frac{1}{2}$ bins have ≥ 2 balls, $> \frac{n}{2}$ bins thats $> n$ balls, $\Rightarrow\Leftarrow$.

Say we've thrown some of the balls. $\alpha_2 \leq \frac{1}{2}$ now. [[It's even $\leq \frac{1}{2}$ at end, so things only better now.]]

New ball thrown: What is ...

prob. new ball ends up at "height" ≥ 3?



Both bins it looks at must have ≥ 2 balls.

prob. $\leq \alpha_2^2 \leq \frac{1}{4}$.

$\therefore$ "intuitively", α_3 := fraction of bins with ≥ 3 balls $\leq \frac{1}{4}$

[[each ball thrown has $\leq \frac{1}{4}$ chance of being in bin of height ≥ 3]]

New ball thrown: what is prob. it ends up at height ≥ 4? (4)

Both bins it looks at must have ≥ 3 balls.

$\rightsquigarrow$ prob. $\leq \alpha_3^2 \leq \left(\frac{1}{4}\right)^2 < \frac{1}{16}$.

$\therefore$ (intuitively). $\alpha_4 :=$ frac. bins with ≥ 4 balls $\leq \frac{1}{16}$

$$\alpha_5 \leq \left(\frac{1}{16}\right)^2 = \frac{1}{256} = 2^{-8}$$

$$\alpha_6 \leq \alpha_5^2 \leq 2^{-16}$$

$$\alpha_7 \leq \alpha_6^2 \leq 2^{-32}$$

$$\dots$$

$$\alpha_{k+2} \leq 2^{-2^k}.$$

Intuitively if $\alpha_{k+2} < \frac{1}{n}$, then probably no bins have $\geq k+2$ balls

$$\Leftrightarrow 2^{-2^k} < \frac{1}{n} \Leftrightarrow 2^{2^k} > \frac{1}{n} \Leftrightarrow k > \log\log n.$$

So "probably" max load $\leq \log\log n + 2$.

Ex: 3 choices still "only" gives $\Theta(\log\log n)$. [So just stick with 2!]

Bloom Filters: Say: • m truly enormous (billion, trillion....)

• strings you're storing also large $\rightsquigarrow L$ bits

(e.g. tweets: $\underset{L}{\approx} 280 \times 8 = 2000$ bits).

Can't possibly use less than Lm bits of space, right? [Right?? And hashing with $n=m$ uses $\approx Ln$ + n]

Use $\approx 8.66m$ bits total! 230x savings [230x fewer servers!!!!]

What's the catch??

Lookup errors: Lookup(s) wrongly says "yes" [even tho s was never stored] w. prob. $\approx 1.6\%$

$8.66 \approx 1.44k$, $1.6\% = 2^{-k}$, for $k = 6$. Can choose other k...

[trade off: 1.44m extra bits, to halve lookup error prob.]

[How Bloom Filters work]

[Used in Real World: Google's "BigTable", Hadoop's "HBase", etc]

Pick small k, eg. $k=6$.

Set $N = \frac{k}{\ln 2} \cdot m$ [rounded off to integer], # of bits used.

$\frac{1}{\ln 2} \approx 1.44$ [this is the 1.44k bits per item]

Alloc. array $T[0 \dots N-1]$ of bits, init. all 0's.

Choose k "independent" hash functions $h_1, \dots, h_k : \{strings\} \to \{0, 1, \dots, N-1\}$

Insert(s): Set $T[h_1(s)] = 1, \dots, T[h_k(s)] = 1$. [maybe some were already 1]

Lookup(s): Return AND of $T[h_1(s)], \dots, T[h_k(s)]$.

Space: $\approx 1.44k$ bits per item

Time: $O(1)$ [$O(k)$] operations

Delete(s): not possible, even slowly

"False positive problem": Lookup(s) may return True even if s was never inserted.

[No "false negs"]

Analysis: Prob. [false positive lookup] $\le$?

Ideas

[again, too fiddly to do rigorously here, so we'll cheat a little]

Q1: After inserting m items, what frac. of bits in $T[\cdot]$ do we expect are still 0?

A: Under SUHA, it's like throwing km balls into $N = \frac{k}{\ln 2} m$ bins, asking about frac. of empty bins ($\lambda = \frac{km}{k/\ln 2 \, m} = \ln 2 \approx .69$)

$$\Pr[\text{bin 1 empty}] = \left(1 - \frac{1}{N}\right)^{km}$$

Most useful approx. ever: $\boxed{1+x \approx e^x \text{ if } x \text{ is tiny}}$ ⑥

$\hookrightarrow 1+x+\frac{x^2}{2}+\frac{x^3}{6}+\cdots$

If $|x| = 10^{-6}$ $\qquad \frac{10^{-12}}{2}$ = negligible.

$\therefore$ $x = -\frac{1}{N}$ is tiny,

$$1+(-\tfrac{1}{N}) \approx e^{-\frac{1}{N}} \quad \therefore \left(1-\tfrac{1}{N}\right)^{km} \approx \left(e^{-\frac{1}{N}}\right)^{km} = e^{-\frac{km}{N}}$$

$$(N = \tfrac{k}{\ln 2} m) \qquad = e^{-\ln 2} = \frac{1}{e^{\ln 2}} = \frac{1}{2}.$$

$\therefore$ $\Pr[\text{bin 1 empty}] \approx 1/2$. [Same for every partic bin.]

$\therefore$ We expect after m inserts, $T[\cdot]$ is about 50-50 0's and 1's

Now say we do Lookup(s), where s has never been inserted.

By SUHA, $h_1(s), \ldots, h_k(s)$ act like k indep. and #'s in $0 \ldots N-1$,

$\therefore$ $\Pr[\text{AND of } T[h_1(s)], \ldots, T[h_k(s)] = \text{True}]$

$= \Pr[\text{false pos.}] \qquad = \left(\frac{1}{2}\right)^k.$ ☺

Summary: can store m items of <u>any</u> size

using $\approx 1.44k$ bits per item,

with $O(1)$-time Lookups/Inserts,

and Insert false-positive probs $\leq 2^{-k}$.