# Consistent Hashing:

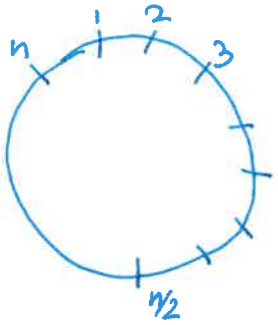
A load balancing strategy when you have $n$ items that are numbered $1..n$, and you want to spread them over $m$ machines, for easy access. (Say $n$ is fixed for now.)

Easy: Send $1..n/m$ to first machine, $n/m+1 \ldots 2n/m$ to second.

Good. But now suppose machines can be added or deleted.

(situation: these are commodity devices that are unreliable. So may come up/down unpredictably.)

Still OK. View the key space as a circle, and give consecutive intervals of size $n/m$ to the machines.

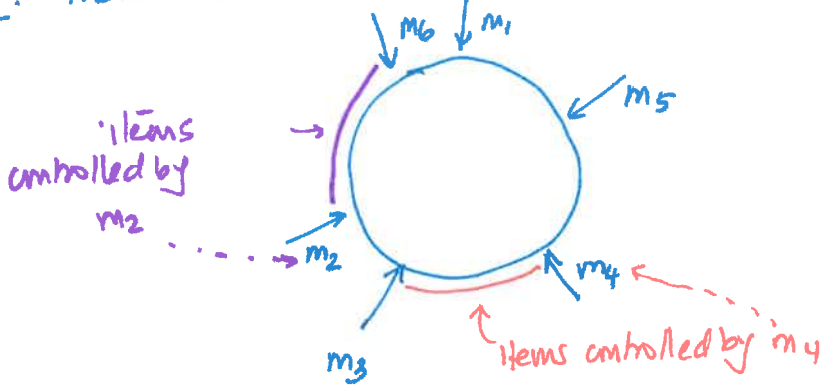



- So when new machine arrives, can do one of several things
  - (a) Re-partition circle into $m+1$ pieces, each machine getting $\frac{n}{m+1}$ of the circle. But this causes lots of churn! ☹
  - (b) take one of the $\frac{n}{m}$ parts, split into two, and give the new half with $\frac{n}{2m}$ items to new machine. (keep $\frac{n}{2m}$ on old machine).
- When machine goes down, take its part and give to previous machine on the circle.

But over time things may become imbalanced. — In case a set of consecutive machines depart.

A simple randomized solution to this is called consistent hashing.

Idea: machines are darts. Throw $m$ darts at the circle




In expectation, each machine gets $\frac{n}{m}$ items, by symmetry.

Insert: a new dart is thrown.

Delete: a dart disappears.

If the adversary choosing the sequence of inserts and deletes does not see our randomness then regardless of history, the partition of the circle just depends on where the current $m$ darts fall.

So $\mathbb{E}[\text{load on each machine}] = n/m$ still.

---

(enough)

But that's not good. Want a stronger guarantee —

$\Pr[\text{load on every machine} \leq n/m]$ $\geq 1 - o(1)$.

low load. with high probability

**Attempt 1:** Max load $\leq \frac{n}{m} \times O(\log m)$.

~~Note that if [illegible] $\geq 2$, then~~

Claim: $\Pr[\text{load on machine } m_i \geq \frac{2n}{m}\log m] \leq 1/m^2$ (we'll be a little approx here)

Look at dart for machine $m_i$, say falls at position $p$.

- the next dart falls at $p+1$ (so interval for $m_i$ has length 1)

  with prob. $\boxed{(1-(1-1/n)^{m-1})}$

  ↑ prob none of others fell on $p-1$

  ↑ call this $q$.

- the next dart falls at $p+2$

  with prob [illegible]

$\Pr[\text{at least one of the other darts fell on } p+2 \mid \text{none of them fell on } p+1]$

$\cong (1-q)\,q$. (by similar reasoning)

note: if $m \ll n$

then $(1-1/n)^{m-1} \cong 1 - \frac{m-1}{n} \pm O(\frac{m^2}{n^2})$ ← tiny

$\cong 1 - \frac{m}{n}$

so $1-(1-1/n)^{m-1} \cong m/n$.

$\Rightarrow$ the length of the interval between dart $m_i$ and the next dart is like a geometric random variable with heads probability $q = m/n$.

$\Rightarrow$ $E[\text{length of interval}] \cong 1/q = n/m$ as we thought

But also: $\Pr[\text{length of interval} \geq 2\frac{n}{m}\log m] \leq (1-q)^{\frac{2n}{m}\log m}$

$\leq e^{-q\cdot(\frac{2n}{m}\log m)} = e^{-2\log m} = \frac{1}{m^2}$

$\Rightarrow \Pr[\exists \text{ a machine with load} \geq \frac{2n}{m}\log m] \leq m \cdot 1/m^2 = 1/m.$

↑ union bound

$\Rightarrow \Pr[\max \text{load} \leq \frac{2n}{m}\log m] \geq 1 - 1/m.$

---

But can we do better? Still a lot of variability in the load, between $[\frac{n}{m}, \frac{n}{m}\log m]$

Yes. **Attempt #2**

For each machine, make $K$ darts.

Machine gets all items assigned to each of darts.

So: Now each dart is assigned $\frac{n}{mK}$ items on average.

And load on machine is <u>not</u> geometric r.v. with mean $\frac{n}{m}$

but sum of $K$ geom. random variables with mean $\frac{n}{mK}$.

$\Pr[\text{sum of } K \text{ geometrics with mean } \frac{n}{mK} > c\cdot\frac{n}{m}] \leq \text{small}.$

↑ $\frac{1}{m^{c'}}$ for some $c'$ related to $c$.

How to see this?

Flip $c\cdot\frac{n}{m}$ coins, each bias $\frac{mK}{n}$

Expect to see $cK$ heads.

<u>Fact</u>: if sum of $K$ geometrics $> \frac{cn}{m}$ then will see $< K$ heads here.

$\Pr[\text{see} < K \text{ heads when mean is } cK] \leq \exp\left(-\frac{(c-1)^2K^2}{3cK}\right)$

$= \Pr[X \leq cK - (c-1)K]$ $\quad = e^{-\Omega(K)} = 1/m^2$ if $c$ is large.

Hence the load on each machine $= O(\frac{m}{n})$ with high probability.

---

Moral of the story:

- Simple algorithm gets good load balancing.

  the naive approach gives high variance, but by breaking into smaller pieces and then summing things up, get concentration and hence lower variance.

- algorithm is also naturally distributed and robust to faults.

- how do we keep track of which interval is controlled by which machine? use a binary search tree data structure, e.g. a treap!

- useful in practice, used by Akamai to get the first of their distributed cache systems, see articles on webpage for interesting story about the start.