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

**Easy:** Send \( 1 \ldots \frac{n}{m} \) to first machine, \( \frac{n}{m+1} \ldots 2\frac{n}{m+2} \) to second.

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

(\textit{situations: 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 \( \frac{n}{m} \) to the machines.

\[ \begin{array}{cccccccccc}
& 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array} \]

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

- 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 act of consecutive machines depart.
A simple randomized solution to this is called consistent hashing.

Idea: machines are darts, throw n darts at the circle

\[
\begin{align*}
\text{Items controlled by } & m_2 \\
\text{Items controlled by } & m_4
\end{align*}
\]

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

**Insert**: a new dart is thrown.

**Delete**: a dart disappears.

If the adversary chooses the sequence of insert and delete does not see our randomness then regardless of history, the partition of the circle just depends on where the current \( n \) darts fall.

So \( \mathbb{E} \text{[load on each machine]} = \frac{n}{m} \) still.

(Enough)

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

\[ \Pr \left[ \text{load on each machine} \leq \frac{n}{m} \right] \geq 1 - o(1) \]

\( \text{with high probability} \)
Attempt 1: \[ \text{Max load} \leq \frac{n}{m} \cdot \Omega(m). \]

Note that if \[ n \gg m, \]

Claim: \[ P_r[\text{load on machine } m_i \geq \frac{n}{m}] \leq \frac{1}{\sqrt{m}}. \]

(we'll be a little approximate here)

Look at dart for machine \( m_i \), say falls at position \( p \).

1. The next dart falls at \( p+1 \) (so interval for \( m_i \) has length 1) with prob. \( (1 - (1 - \frac{1}{n})^{m_i}) \)

\[ \Rightarrow \text{prob none of others fell on } p. \]

\[ \text{call this } q. \]

2. The next dart falls at \( p+2 \) with prob. \( \frac{1}{n} \)

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

\[ \leq (1-q)^2. \quad (\text{by similar reasoning}) \]

\[ \Rightarrow \text{the length of the interval between dart } m_i \]

and the next dart is like a geometric random variable with heads probability \( q = \frac{1}{m} \).

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

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

\[ \leq e^{-q(\frac{2n}{m} \log m)} = e^{-\frac{2n}{m}} = \frac{1}{\sqrt{m}}. \]
\[
\Rightarrow P_b \left[ \text{a machine with load } \geq \frac{2n}{m} \log m \right] \leq m, \quad \frac{1}{m^2} = \frac{1}{m}
\]

(Uniform bound)

\[
\Rightarrow P_b \left[ \text{max load } \leq \frac{2n}{m} \log m \right] \geq 1 - \frac{1}{m}.
\]

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

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 not geometric r.v. with mean \(\frac{n}{m}\)

but sum of \(K\) geometric random variables with mean \(\frac{n}{mk}\).

\[
P_b \left[ \text{sum of } K \text{ geometrics } \geq c \cdot \frac{n}{m} \right] \leq \text{small.}
\]

\[
\frac{n}{mk}
\]

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.

**Fact:** if sum of \(K\) geometrics \(\geq c'n/m\) then will see \(< K\) heads here.

\[
P_b \left[ \text{see } < K \text{ heads when mean in CK} \right] \leq \exp \left( -\frac{c'c-1)^2K^2}{2cK} \right)
\]

\[
= P_b \left[ X \leq cK - (c-1)K \right] = e^{-o(K)} = \frac{1}{m^2} \text{ if c is Large.}
\]
Hence the load on each machine = \( O\left(\frac{\log n}{m}\right) \) 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 heap!

- 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.