CS 6968: Scalable Algorithms Spring 2023

Lecture 19: Distributed Hash Tables cont. - April 13th, 2023

Prof. Prashant Pandey *Scribe: Todd Thornley*

1 Overview

In the last lecture we discussed various issues and design concerns with distributed hash tables, ending with a discussion of Chord.

In this lecture we look into some questions from the previous lecture and discussed concepts from the Distributed Hashing paper.

2 Skew/Load Factor

For N items on M machines using hash function H, the expected number of items on a machine M is:

$$N/M$$

w.h.p. the number will be:

$$O(lg(N)/lglg(N))$$

if $N = M$ per balls in bins.

Load factor is defined as $\frac{\text{total items}}{\text{possible storage}}$

3 Consistent Hashing

- There are **M items** such that each of them needs to be stored in one of the **N distributed machines**
- Recal. hash functions
  2-wise independent hash function

  $H = \{h_{a,b} | a \in \{1 \ldots p{-}1\} \text{ and } b \in \{0, 1, 2 \ldots\ p{-}1\}| \text{ where } h_{a,b}(x) = (ax{+}b \mod P) \mod n$
- Using a 2-wise independent family of hash functions we can create a perfect hash
- Perfect hashing only works well if the number of machines does not change
- If the number of machines changes:

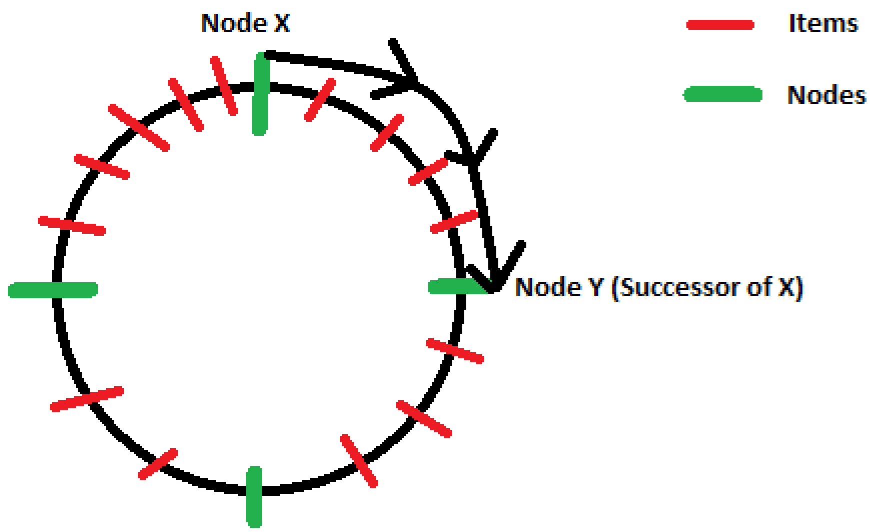

Figure 1: Consistent Hashing Diagram. Data is placed clockwise or to the right.

1. **Change the n in $h_{a,b}$ to n' to get $h'_{a,b}$:**
   By doing so almost all items will need to be moved to a new machine.
2. **Keep n unchanged:**
   Thus no moving, but the new machine will go unused creating imbalanced load.

- A strategy is needed that does not incur a lot of rehashing while also keeping the load balanced across all machines.

3.1 Basic Idea

Each machine and item is mapped to a random real number in the interval $[0, 1]$

- Store the item in the **successor** of the item's hash position
- The successor is the first machine "on the right"
- If there is no machine "on the right", the successor is the machine with the smallest number. (Wrap around to the beginning, see figure 1)

3.2 Implementation

- To dynamically maintain machines and we need binary search trees whose keys are the values assigned to the machines.
- Let $h_i$andh_m be respective hash functions that we use to hash items and machines in the interval $[0, 1]$

3.2.1 Insert

- Find the successor of $h_i(x)$ in the BST
- Store x in returned machine

3.2.2 Delete

- Find the successor of $h_i(x)$ in the BST
- Delete x in returned machine

3.2.3 Node Up

- There may be items in the successor of $h_m(y)$ that belong in y
- Find the successor of $h_m(y)$ in the BST
- Move all items whose h_i value is greater than $h_m(y)$ to y.

3.2.4 Node Down

- Find the successor of $h_m(y)$ in the BST
- Move all items in value is greater than $h_m(y)$ to returned node.

3.3 Bounds

Lemma 1:

w.h.p. no one machine has more than $O(lg(n)/n)$ as a fraction of items

Proof:

Find some interval I of length $\frac{2lg(n)}{n}$

- Pr[no machine lands in I] $= (\frac{1-2lg(n)}{n}) \approx \frac{1}{n^2}$ by union bound
- Equally split $[0, 1]$ into $\frac{n}{2lg(n)}$ such intervals
- Pr[every interval has at least 1 machine] $= 1 - \frac{n}{2lg(n)} * \frac{1}{n^2} > 1 - \frac{1}{n}$
- w.h.p. each machine owns an interval of length at most $\frac{c*lg(n)}{n}$

Bibliography.

References

[1] David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, Rina Panigrahy Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. *Symposium on Theory of Computing*, 29(1):654–663, 1997.