CS5112: Algorithms and Data Structures for Applications

Surprising hashing applications

Ramin Zabih

Some content from: Wikipedia

Cornell University

CORNELL TECH

Administrivia

- A duck problem will be on the final exam
  - But lower bounds will not be, since it’s not core
- Extra office hours on Wednesday if needed?

CORNELL TECH

Lecture outline

- String search with hashing
- Distributed hash tables
- Chord and skiplists
- Locality sensitive hashing

String search

- Find one string (“pattern”) in another
  - Naively we repeatedly shift the pattern
  - Example: To find “greg” in “richardandgreg” we compare greg against “rich”, “icha”, “char”, etc. (‘shingles’ at the word level)
- Finding pattern P in text T
  - Use the same P on different T
    - Ex: an important genetic sequence. Which patients have it?
  - Use different P on the same T
    - Ex: the Mueller report. Find various strings in it.

Rabin-Karp string search

- Instead let's use a hash function h
- We first compare h("greg") with h("rich"), then h("icha"), etc.
- Only if the hash values are equal do we look at the string
- Why? $x = y \Rightarrow h(x) = h(y)$
  – but not $\Leftarrow$ of course!
- We can pre-process the text and compute hash values starting at every character
  – For a given length of string (4 in the example above)

CORNELL TECH

Rolling hash functions

- To make this computationally efficient we need a special kind of hash function h
- As we go through "richardandgreg" looking for "greg" we will be computing h on consecutive strings of the same length
- There are clever ways to do this, but to get the flavor of them here is a naïve way that mostly works
  - Take the ASCII values of all the characters and multiply them
  - Reduce this modulo something reasonable
  - Moving window minimizes recomputation

Applying Rabin-Karp

- With a rolling hash function we can pre-process T
  - At each character, hash the k letters starting there
  - Probably need to do this for a few values of k
    - Store this in a table
- Suppose we'd like to quickly eliminate a pattern P?
  - Maybe from a large part of T?
  - Hint: can we tell that the $h(P)$ is definitely nowhere in T?
- Answer: Bloom filters!

CORNELL TECH

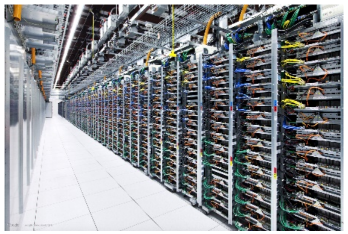
Distributed hash tables (DHT)

- BitTorrent, etc.
- Given a file name and its data, store/retrieve it in a network
- Compute the hash of the file name
- This maps to a particular server, which holds the file
- Sounds good! Until the file you want is on a machine that is not responding…
  - But is this a real issue? Aren't computers pretty reliable?

CORNELL TECH

Google datacenter numbers (2008)

- *In each cluster's first year, it's typical that:*
  - *1,000 individual machine failures will occur;*
  - *thousands of hard drive failures will occur;*
  - *one power distribution unit will fail, bringing down 500 to 1,000 machines for about 6 hours;*
  - *20 racks will fail, each time causing 40 to 80 machines to vanish from the network;*
  - *5 racks will "go wonky," with half their network packets missing in action;*
  - *The cluster will have to be rewired once, affecting 5 percent of the machines at any given moment over a 2-day span.*
  - *About a 50 percent chance that the cluster will overheat, taking down most of the servers in less than 5 minutes and taking 1 to 2 days to recover.*
- Jeff Dean, "[Google spotlights data center inner workings]", CNET May 2008



CORNELL TECH

From filename to processor

- Typically the result of a hash function is a large number
  - SHA-1 produces 160 bits (not secure!)
- Map into servers with modular arithmetic
  - Recall: 4 + 7 = 1 (mod 10)
  - mod with powers of 2 is just the low-order bits
- How do we handle a server crashing or rejoining??
- Simplest example: we mapped our files to 2 servers
  - Half hashed to bucket #0 and half to bucket #1
  - What do we do when one server crashes?

Consistent hashing

- Effectively the hash table itself is resized
- With naïve hash functions, resizing is a disaster
  - Everything needs to be shuffled between buckets/servers
  - The mapping from filenames to processors must have state to account for various machines being down
- Ideally, if we have 9 servers and add a new one, the new server should get $\frac{1}{10}$ of the files
  - Move files only from old to new (not old to old)
  - Randomly chosen from among the old files

CORNELL TECH

Hashing into the circle

- Let's convert the output of our hash function into a circle
  - For example, using the low-order 8 bits of SHA-1
- We map both servers and data onto the circle
  - For a server, hash of IP address or something similar
- Data is stored in the "next" server on the circle
  - By convention we will move clockwise

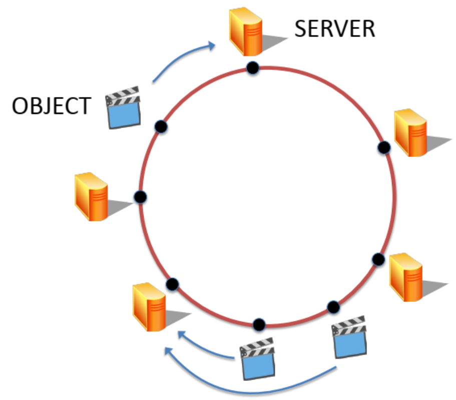


Figure from [Maggs, Bruce M.](); [Sitaraman, Ramesh K.](). (July 2015), ["Algorithmic nuggets in content delivery"]() (PDF), *SIGCOMM Computer Communication Review*, New York, NY, USA,**45** (3): 52–66

CORNELL TECH

Example of consistent hashing

- Data 1,2,3,4 stored on computers A,B,C
- Servers->data (good quiz/exam question):
  A->1,4
  B->2
  C->3
- If C crashes, we just move 3 to A
  - Shift data clockwise
- Key point: nothing else moves!

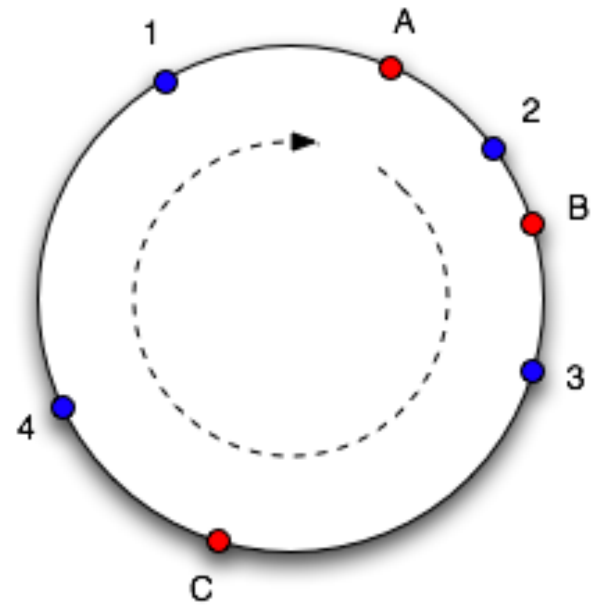


Diagram taken from Tom White based on original article

CORNELL TECH

Gracefully adding a server

- Add server D after C crashes
  – Takes 3,4 from A
- Servers->data:

  A->1

  B->2

  D->3,4
- This is a lot faster!
  – Naively, going from 3 to 4 servers moves 75% of data
  – With consistent hashing we move 25% of data
  – Advantage gets even larger for more servers

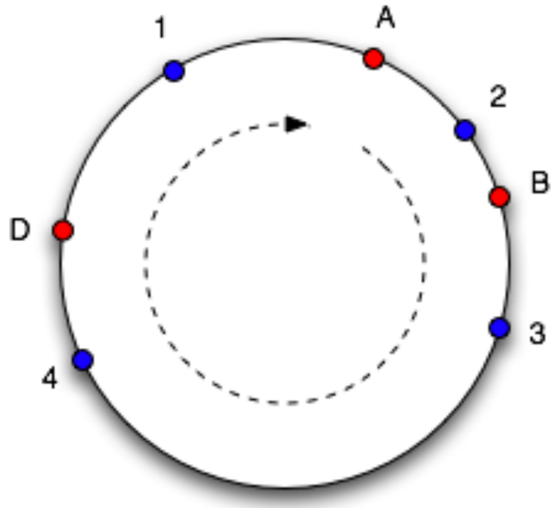


Diagram taken from [Tom White] based on [original article]

CORNELL TECH

Improving consistent hashing

- Need a uniform hash function, lots of them aren't
- Note that the data still needs to move after a crash
- Store the servers in a BST to efficiently find successor
  - This requires global knowledge about the servers
- Global knowledge is a key weakness
- Popular objects are a challenge

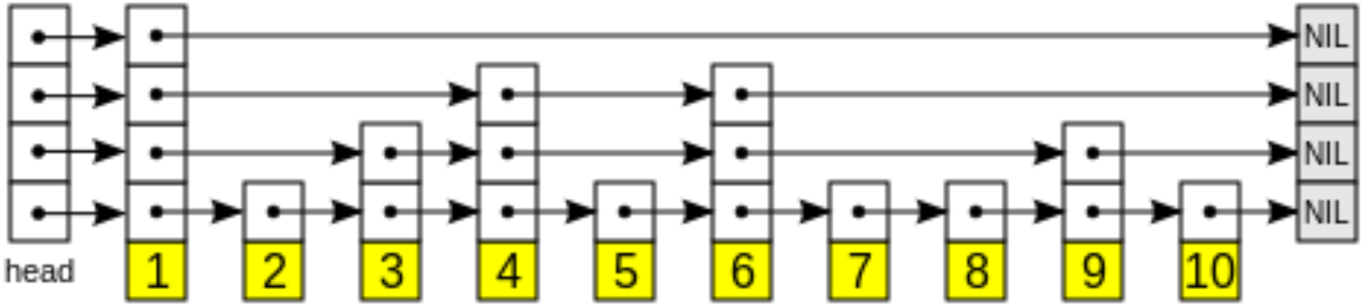
CORNELL TECH

Handling popular objects

- Each popular object can have its own hash function
- Basically, its view of the unit circle
- Ensures that you are very unlikely to have 2 popular objects share the same server

Skip lists

- Can we find an element in a sorted list quickly?
  - Hierarchy of ‘express lanes’, randomly generated



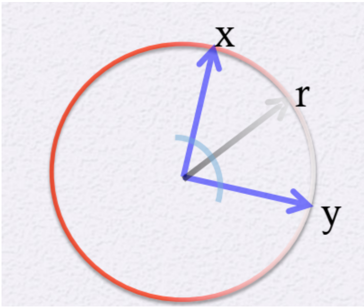
Source

  - Lookup goes in row-major order
    - If we find something greater, backtrack and drop down a level

CORNELL TECH

Other important LSH families

- We looked at Hamming distance via random subset projection
- Jacard similarity via min hash
  - Hash functions are random permutations
  - Compute the minimum value of a subset under a permutation
- Angle similarity via projection onto random vector



CORNELL TECH