Precept 6

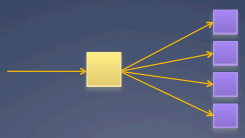
Hashing & Partitioning

Peng Sun

1

Server Load Balancing

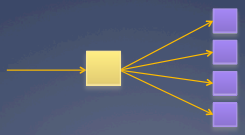
- Balance load across servers
- Normal techniques:
  - Round-robin?



2

Limitations of Round Robin

- Packets of a single connection spread over several servers



3

Multipath Load Balancing

- Balance load over multiple paths
- Round-robin?

4

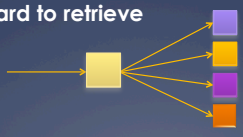
Limitations of Round Robin

- Different RTT on paths
- Packet reordering

5

Data Partitioning

- Spread a large amount of data on multiple servers
- Random?
- Very hard to retrieve



6

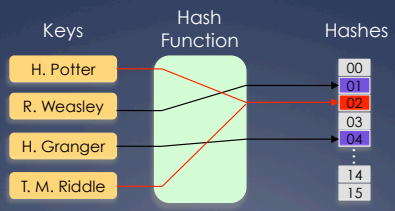
Goals in Distributing Traffic

- **Deterministic**
  - **Flow-level consistency**
  - **Easy to retrieve content from servers**
- **Low cost**
  - **Very fast to compute/look up**
- **Uniform load distribution**

7

Hashing to the Rescue

- **Map items in one space into another space in deterministic way**



8

Basic Hash Function

- **Modulo**
  - **Simple for uniform data**
  - **Data uniformly distributed over N. N >> n**
  - **Hash fn = <data> mod n**
- **What if non-uniform?**
  - **Typically split data into several blocks**
  - **e.g., SHA-1 for cryptography**

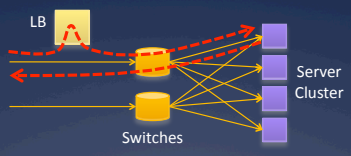
9

Hashing for Server Load Balancing

- **Load Balancing**
- **Virtual IP / Dedicated IP Approach**
  - **One global-facing virtual IP for all servers**
  - **Hash clients' network info (srcIP/port)**
  - **Direct Server Return (DSR)**

10

Load Balancing with DSR



- **Reverse traffic doesn't pass LB**
- **Greater scalability**

11

Equal-Cost Multipath Routing

- **Balancing flows over multiple paths**
- **Path selection via hashing**
  - **# of buckets = # of outgoing links**
  - **Hash network Info (src/dst IP) to links**

12

Data Partitioning

- Hashing approach
  - Hash data ID to buckets
  - Data stored on machine for the bucket
  - Cost: *O(# of buckets)*
- *Non-hashing, e.g., "Directory"*
  - Data can be stored anywhere
  - Maintenance cost: *O(# of entries)*

13

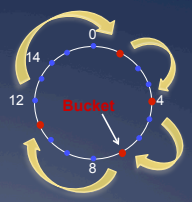
But…

- Basic hashing is not enough
- Map data onto k=10 servers
  - with (dataID) mod k
- What if one server is down?
  - Change to *mod (k-1)*?
  - Need to shuffle the data!

14

Consistent Hashing

- Servers are also in the Key Space (uniformly)
- Red Nodes: Servers' positions in the key space
- Blue Nodes: Data's position in the key space
- Which Red Node to use:
  - Clockwise closest



15

Features of Consistent Hashing

- Smoothness: Addition/removal of bucket does not cause movement among existing buckets (only immediate buckets)
- Spread and load: Small set of buckets that lie near object
- Balanced: No bucket has disproportionate number of objects

16

Another Important Problem

- How to quickly answer YES or NO?
- Is the website malicious?
- Is the data in the cache?

17

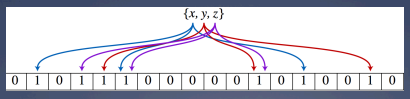
Properties We Desire

- Really really quick for YES or NO
- Okay for False Positive
  - Say Yes, but actually No
- Never False Negative
  - Say No, but actually Yes

18

Bloom Filter

- Membership Test: In or Not In
- *k* independent hash functions for each data
- If all *k* spots are 1, the item is in.



19

Bloom Filter

- Only use a few bits
  - Fast and memory-efficient
- Never gives a false negative
- Possible to have false positives

20

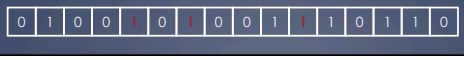
Demo of Bloom Filter

Start with an *m* bit array, filled with 0s.

To insert, hash each item *k* times. If $H_i(x) = a$, set $Array[a] = 1$.

To check if *y* is in set, check array at $H_i(y)$. All *k* values must be 1.

Possible to have a false positive: all *k* values are 1, but *y* is not in set.



21

Application of Bloom Filter

- Google Chrome uses BF:
  - First look whether website is malicious
- Storage services (i.e., Apache Cassandra)
  - Use BF to check cache hit/miss
- Lots of other applications…

22

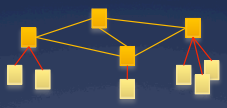
Thanks!

23

Backup

24

Hashing in P2P File Sharing



- Two Layers: Ultrapeer and Leaf
- Leaf sends hash table of content to Ultrapeer
- Search request floods Ultrapeer network
- Ultrapeer checks hash table to find leaf

25

Applying Basic Strategy

- Consider problem of data partition:
  - Given document X, choose one of k servers to store it
- Modulo hashing
  - Place X on server i = (X mod k)
  - Problem? Data may not be uniformly distributed
  - Place X on server i = (hash(X) mod k)
  - Problem? What happens if a server fails or joins (k → k ±1)?

26

Use of Hashing

- Equal-Cost Multipath Routing
- Network Load Balancing
- P2P File Sharing
- Data Partitioning in Storage Services

27