Typeahead

Scenario

- Google suggestion

    - Prefix -> top n hot key words
    - DAU: 500M
    - Search: 66500M = 18b (Every one search for 6 words, each word has 6 characters)
    - QPS = 18b / 86400 ~ 200k
    - Peak QPS = QPS * 2 ~ 400k
- Twitter typeahead

Initial design

- Query service

- Data collection service

Storage

Query service DB

Word count table

- How to query on the db
- Query SQL: Select * from hit_stats where keyword like ${key}% order by hitCount DESC Limit 10
    - Like operation is expensive. It is a range query.
    - where keyword like 'abc%' is equivalent to where keyword >= 'abc' AND keyword < 'abd'

keyword	hitCount
Amazon	20b
Apple	15b
Adidas	7b
Airbnb	3b

Prefix table

- Convert a keyword table to a prefix table, put into memory

prefix	keywords
a	"amazon","apple"
am	"amazon","amc"
ad	"adidas","adobe"
don	"don't have", "donald trump"

Trie

- Trie (in memory) + Serialized Trie (on disk).
    - Trie is must faster than DB because
        - All in-memory vs DB cache miss
- Store word count at node, but it's slow
    - e.g. TopK. Always need to traverse the entire trie. Exponential complexity.
- Instead, we can store the top n hot key words and their frequencies at each node, search becomes O(len).

prefix	keywords
a	"amazon","apple"
am	"amazon","amc"
ad	"adidas","adobe"
don	"don't have", "donald trump"

- How do we add a new record {abd: 3b} to the trie
    - Insert the record into all nodes along its path in the trie.
    - If a node along the path is already full, then need to loop through all records inside the node and compared with the node to be inserted.

Data collections service

- How frequently do you aggregate data
  - Real-time not impractical. Read QPS 200K + Write QPS 200K. Will slow down query service.
  - Once per week. Each week data collection service will fetch all the data within the most recent one week and aggregate them.
- How does data collection service update query service? Offline update and works online.
  - All in-memory trie must have already been serialized. Read QPS already really high. Do not write to in-memory trie directly.
  - Use another machine. Data collection service updates query service.

Scale

How to reduce response time

- Cache result
  - Front-end browser cache the results
- Pre-fetch
  - Fetch the latest 1000 results

What if the trie too large for one machine

- Use consistent hashing to decide which machine a particular string belongs to.
  - A record can exist only in one machine. Sharding according to char will not distribute the resource evenly. Instead, calculate consistent hashing code
  - a, am, ama, amax stored in different machines.

How to reduce the size of log file

- Probablistic logging.
  - Too slow to calculate and too large amount of data to store.
  - Log with 1/10,000 probability
    - Say over the past two weeks "amazon" was searched 1 billion

times, with 1/1000 probability we will only log 1 million times.

- For a term that's searched 1000 times, we might end up logging only once or even zero times.