Documentation & Design Write Up

https://github.com/shashank1992/topcrawl/blob/main/README.md

Title: Web URL Collection System

1. MVP:

The Web URL Collection System is designed to efficiently collect millions of URLs, fetch their content, and store top n topics for further analysis.
Prevent duplicate requests to sites , which could result in being blocked, But should also have a retry mechanism.
Protect the server against unusual long queries or huge data.

2. Estimation of Scale:

Considering an average of 1 Million requests, we arrive at, by approximating 1 mb of network data per request, a total of 10^6 Mb or 1 TB of network data per day.

Considering a single server takes 2 sec to process each request, 1 Million requests would take 2sec * 10^6 seconds of compute time, which has to be processed in 86400 seconds(1day) dividing both we need to crawl ~ 23 sites/ second. Accounting for any retries and additional scale, the machine requirement would be accordingly scaled up.

Storage involved, assuming we store the web url, and the relevant meta data, which could take approximately 500 bytes or 0.5 kb / site.
This will add up to 10^6 * 0.5 kb = 0.5 GB / day or 15 GB per month.
This will approximately be 200 GB / Year. (If we add a replication level of 2, it could increase)

3. Design Goals

This system has moderate write requests to the db / day and much lesser read requests. Need to ensure replication, for fault tolerance.

4.System Design:

The following components in the same hierarchy during request are required.

1. Multiple servers and distributed load balancing along with distributed

DNS, to handle parallel requests.
2. Maintain distributed queues such as RabbitMQ, to keep track of urls.
3. LRU cache of the indexed sites , a write thru cache.
4. SQL DB, (url, meta tags, isIndexed , timestamp)

Need to maintain a DB replica (replication level of atleast 2) to ensure reliability and backup data. Can also consider archiving data prior to 5 years, due to the outdated info. Newer records would consider the relevant data.

Additional key aspects to ensure scalability are

- Optimise processing algorithms for parallelism.

Retry Mechanism:
When the request, fails, the same url should enter a retry queue where retries are attempted after a fixed time delay.