Problem Statement

Design a distributed web crawler.

Web crawler : A program or a software which crawls/scan the one website after another.

purpose is to index the content of websites all across the Internet so that those websites can appear in search engine results.

It is a type of bot that is typically operated by search engines like Google and Bing.

Web crawlers begin crawling a specific set of known pages, then follow hyperlinks from those pages to new pages.

Search engines like Google or Bing apply a search algorithm to the data collected by web crawlers to display relevant information and websites in response to user searches.

If an organization or website owner wants its website to rank in a search engine, it must first be indexed. If webpages aren't crawled and indexed, the search engine can't find them organically.

Main functions/utilities

1)Search Engine (Google)

2)Duplicacy check/Copywright Infringement: We can use web crawler to find if such a kind of data is available already on website to ensure and check if it is a duplicate data.

3)Keyword based search

Let us a if we want to create an application which provide all the offers present on all different e-commerse websites, we need to use crawler to find and collect all the data a display it at a single place in our application.

# Features

Scope of Functional requirements (F.R)

1)Crawl all the webpages

2)Avoid dowloading the duplicate content (sometimes the same content will be present on different webpages exactly same, so it is most of like wastage of the resource)

3)Avoid visiting the same url again (lead to the infinite loop)

Scope of non-functional requirements (N.R.F)

1)Scalable  
The no of websites increasing continuosly, today if it is 1 M, in future it will be 2 M

2)Extensible

I should be able to add the new features easily

3)Easy to understand

4)Should be polite to the websites: When we are crawling any website, we would not have taken the permission, so the websites are not aware that we are hitting the end points to collect the data.

If we bombard those websites with our crawlers, that might bring down the application itself, we have to make sure that we should not attack them with the mutiple concurrent calls.

# Estimations

Estimation of the infrastructure

How much of each of the following

i)Storage?

ii)Computation?

iii)Network?

Let us assume total no of websites 1 B, out of it 60% are genuine => 0.6 B.

Few of the websites for example wikipedia might comprise of 100’s and 1000’s of pages, other hand few websites which have 1 or 2 pages.

Let say the avg no of pages => 100 pages for website.

Total no of pages => 0.6 x 100 => 60 B pages.

We should be crawling all the pages.

Let say if we want to avoid the images/video content of the page

Avg size of webpage = 100 kb

Total is 60 B x 100 kb = 6 B x 10^3 kb = 6 B bytes x 10^6 = 6 GB x 10^6 = 6 PB.

Since we should only use the 75% of the storage, we need around 8 PB of storage.

This can not be done by monolithic system; we should go for distributed solution.

# High level design

1)start with the list of URLs (seed urls)

2)fetch the html content of the URL

3)check for duplicates, if duplicate ignore

4)If not duplicate download the content

5)Parse the content and fetch new URLs

Repeat the same steps

We just need to break the application into the 5 components based on the above steps.

1)URL provider

2)Content fetcher

3)Duplicate checker

4)Content downloader

5)URL parser

URL provider

Total no of webpages = 60 B => 60 B URLs

Avg size of 1 URL = 100 bytes

Total size = 60 B x 100 bytes

=6 B Bytes x 10^3

=6 GB x 10^3

=6 TB

So, we have to go for distributed solution.

We can represent the URL networks in terms of a graph.

For our crawling we can use BFS, since we are going layer over layer.

Content fetcher

Since there are billions of web links, one component might not suffice, we have to go for distributed solution.

There is issue with the distributed system here we bring in the concurrency or parellelism, let us say that if we have a website like facebook.com, and link of the it is present all over nodes.

All of the fetchers will try to call the facebook.com links which may lead to large number of concurrent calls to the facebook.

We had a consideration that our application should be polite.

So, we have to make sure that all the links belonging to same domian should end up at the same machine.

Let us same for microsoft domain we have

[www.microsoft.com](http://www.microsoft.com)

[www.mail.microsoft.com](http://www.mail.microsoft.com)

[www.microsoft/office/365.com](http://www.microsoft/office/365.com)

all of these belongs to same domain, and all of these should fall on the same fetcher machine.

There are basically 2 advantages of this here

i)We can avoid the bombarding of attacks and be polite with the application

ii)We do not have to make multiple connections to domain

Duplicate Checker

It has to be distributed.

It will compare the incoming data content with the stored data content on the storage layer.

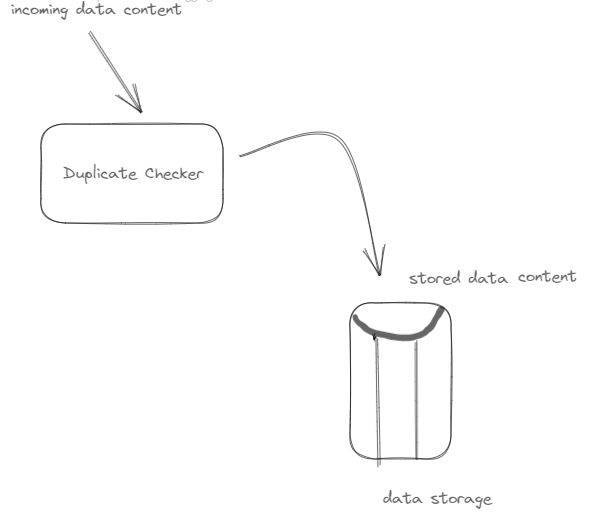
How do we compare data contents?

In the naive way we can do the character per character comparision.

But that will be very slow which will impact the performance.

For the better appraoch is using hashing, we will take the hash of the new data and compare with the hash of the existing data. If it matches it can be done comparitively very fast.

This is the common method to be used if we need to compare two information.



But the challenge is in case of the small changes like case change/small space introduced or deleted, semantic change etc, the produced has value will be different.

Normally in such case we will be using the machine learning / AI algorithms to check the similarity percentage between the content like upto 75% etc.

Content downloader

Take the content from the duplicate finder and store that in the data storage.

We saw that around 6 PB we need horizontal scaling.

Some of the examples of the distributed storage are HDFS/S3/Data lakes.

This storage is accessed by other modules also therefore we should not keep in solid storage only,we can use something External Cache like the Redis.

Url Parser

To go and scan all the newly downloaded content and to check all URLs

To check for the new URLs

Whenever a new URL comes we need to check inside the DB storage for all the URLs and decide if it is old or not.

If we do linear search it’ll O(n) , it’ll take lot of time.

If we do binary search , it’ll be O(log n)

Bloom filters : It is a datastructure which tells you if the entry is present or not in the constant time.

It is not 100% accurate all time.

If it says key is not present 100% true,but if it say key is present 80 to 100 % true.

So we may miss few URLs.

