Project zoekmachine

Inleiding

Dit project is een ontwerp en implementatie van een zoekmachine. Het doel is om meer te leren over databases (NoSQL en relationeel), hun eigenschappen voor "big data" opslag, Python, en zoekmachines.

Architectuur

De zoekmachine bestaat uit 3 modules:

- Spider
- Indexer
- Zoek front-end

Al deze modules lezen of schrijven naar een centrale database, waarschijnlijk PostgreSQL.

Spider

De spider haalt continu webpagina's op van het internet. Dit gebeurt met meerdere *threads* of processen.

De database bevat een *queue* (wachtrij) met URL's die gecrawled moeten worden, en **wanneer** dit gedaan moet worden. Zo kan een pagina ingeroosterd worden om in de toekomst opnieuw gecrawled te worden, wat de zoekmachine actueel houdt.

De spider gaat als volgt te werk:

- Haal de eerstvolgende URL uit de queue.
- Verkrijg een *lock* op de URL, zodat geen andere *thread* deze zal downloaden.
  - Dit kan geïmplementeerd worden met een SQL-transactie.
- Download de inhoud van de URL.
- Schrijf de inhoud weg naar een andere tabel uit de database voor de indexer.
- Rooster de URL in om later opnieuw gecrawled te worden.
  - Initieel kan deze interval vrij hoog zijn, bijvoorbeeld een week.

Indexer

De indexer verwerkt de webpagina's die door de spider zijn opgehaald. Het converteert ze naar een beter doorzoekbaar format.

De indexer parset de HTML, en haalt hier zoveel mogelijk nuttige informatie uit. Individuele paragrafen en koppen worden geïdentificeerd. Deze stukken tekst krijgen een gradatie van hoe belangrijk ze zijn binnen de pagina. Zo is een `<h1>`-tag belangrijker dan een `<p>`-tag. Uit al deze paragrafen

Ook worden HTML image tags geïdentificeerd en opgeslagen, zodat er met de zoekmachine naar afbeeldingen gezocht kan worden.

Daarnaast identificeert de indexer nieuwe URL's die in de webpagina's voorkomen, en voegt deze toe aan de *queue* van de spider.

Zoek front-end

Het zoek front-end krijgt een zoekopdracht (*query*) van de gebruiker binnen, en geeft hier netjes geformatteerde zoekresultaten voor terug.

- De zoekopdracht wordt opgebroken in woorden.
  - Hierbij wordt rekening gehouden met streepjes en slashes, die belangrijk kunnen zijn voor een term.

Het zoek-frontend maakt gebruik van het format dat de indexer heeft aangemaakt, en rangschikt de resultaten volgens een algoritme.

Zoek algoritme

Resultaten sorteren

De gebruikers van een zoekmachine willen vanzelfsprekend de meest relevante resultaten als eerste zien.

Het PageRank algoritme, uitgevonden door Google, is een manier om te bepalen hoe belangrijk webpagina's zijn. Het werkt door te kijken welke pagina's verwijzen naar andere pagina's.

Het zou mooi zijn om een naïeve versie van PageRank te ontwerpen en implementeren, maar dit is niet verplicht voor dit project.

Stemming

Het Engelse woord *stemming* wil zeggen dat we woorden transformeren naar een stam-woord. Zo is de stam van de woorden “kattig” en “katten” allebei “kat”.

De zoekmachine kan dit principe toepassen om synoniemen voor de zoektermen te vinden, en zodoende de resultaten te verbeteren.

Database

In de eerste plaats zal PostgreSQL worden gebruikt als DBMS voor de zoekmachine. Postgres is een solide, feature-rijk DBMS, met goede eigenschappen qua schaalbaarheid. Aangezien dit project een onderzoek is, verwacht ik dat het ruim zal voldoen.

Andere systemen

Dit project is vergelijkbaar met de volgende systemen:

- Lucene
- Solr
- Elasticsearch

Voor de database bestaan de volgende NoSQL alternatieven (die eventueel onderzocht kunnen worden):

- Hadoop
- MongoDB

Doelstellingen

Voor dit project zijn de volgende doelstellingen opgesteld. Indien hier aan voldaan is, is het project voltooid.

Must-haves

- Het indexeren van minstens twee domeinen, waarvan elk domein minstens 250 webpagina’s telt.
- Het bieden van een web-interface waarmee gebruikers kunnen zoeken.

- Het effectief doorzoeken van de geïndexeerde inhoud op basis van zoektermen. Hierbij moeten de meest relevante resultaten als eerste verschijnen.
- Een *spider* welke met vaste interval de websites opnieuw indexeert.
- *Rate-limiting*, zodat de *spider* geen overmatig verkeer genereert voor website eigenaars.

Should-haves

- Het kunnen zoeken naar afbeeldingen op basis van de alt-tekst hiervan.
- De spider houdt rekening met `robots.txt` informatie die de site verstrekt.

Could-haves

- Een naïeve versie van het PageRank-algoritme.

Won't-haves

- Ondersteuning voor- en onderzoek naar alternatieve databases.

Hosting

De volgende opties zijn er m.b.t. hosting van de zoekmachine.

VPS met big storage

Kosten: maandelijks €20 (€10 VPS, €10 big storage)

Een VPS van TransIP beschikt over een snelle internetverbinding. Bovendien kan er opslag worden toegevoegd in de vorm van *big storage*. Dit kost €20 per 2Tb. Daarnaast kan voor een extra €10 per maand de hoeveelheid CPU cores + RAM van de VPS verdubbeld worden.

Ondanks de maandelijkse kosten is dit de aantrekkelijkste optie voor dit project.

Raspberry Pi's

Kosten: geen, hardware al in bezit. Eventueel eenmalige kosten aan een paar extra Raspberry Pi's.

Een cluster van Raspberry Pi's zou gebruikt kunnen worden voor de zoekmachine. Omdat één Pi te zwak is voor de hele zoekmachine, zouden we de componenten

verdelen over meerdere Pi's. Een netwerkswitch of router is nodig om de Pi's te netwerken.

- Database master (Pi2)
  - De database bevindt zich op een externe schijf van minstens 2Tb.
- Spider(s)
- Indexer(s)
- Webserver voor zoek front-end

Een aantal losse Pi's zijn 'werkers', elk met maar één functie. Spider of indexer. Allen communiceren via TCP met de master database.

Xserve

Kosten: geen, hardware al in bezit

Er is een tweedehands Apple Xserve (rack-mount server) met een RAID opslagapparaat die gebruikt kan worden voor dit project. In totaal beschikt het over 8Tb opslag, 2 Xeon CPU's en 2Gb werkgeheugen.

De nadelen zijn dat de server niet makkelijk ingezet kan worden ivm geluidsproductie en energieverbruik.

Bronnen

- https://en.wikipedia.org/wiki/Web_search_engine