Seekourney Weaver

Projektförslag

Ripley

Som Google, fast för både ditt lokala filsystem och webben.

1 Inledning

1.1 Introduktion

I takt med att mängden digital information ökar, ökar även mängden filer både privatpersoner och organisationer behöver hantera. Genom detta uppstår en utmaning, hur man snabbt kan hitta den information man behöver i havet av filer och hemsidor. För webben finns tjänster som Google och lokalt på datorn kan man ofta söka efter filnamn, med det saknas en smidig lösning för att hitta information över båda dessa källor för flera filtyper och genom sökning på externa datorer. Det är för att lösa detta problem som Seekourny Weaver utvecklas. Systemet är en sökmotor som i bakgrunden går igenom och indexerar filer och vitlistade hemsidor så att användare enkelt kan söka efter relevant information via en hemsida. Genom att använda concurrency kan denna indexering och sökning ske snabbt och flera klienter kan samtidigt göra sökningar.

1.2 Syfte och frågeställning

Syftet med projektet är att utveckla ett effektivt system för att söka på filnamn, information i filer av olika typer, och information på vitlistade hemsidor. Genom att arkitektera systemet att kunna hantera ospecificerade filtyper möjliggör att i framtiden erbjuda informationssökning i dokument på en organisationsbred skala, men även genom vidareutvecklingar (se nedan kapitel) görs systemet relevant för större delar av samhället. Systemets agnostiska förhållningssätt för filtyper ska göra det lättare för både organisationer och privatpersoner att hitta den information de söker, som att exempelvis söka genom alla Uppsala universitets vetenskapliga rapporter (givet att det finns en indexerare för PDF dokument). Systemet ska kunna köras själv av organisationer som låter externa användare söka bland deras filer.

Det ligger även ett stort fokus på att gruppens medlemmar får utveckla sin kunskap om concurrency samt fördjupa förståelsen om hur program ska byggas för att vara användbara, vilket är delar av kursens mål. Ett stort intresse finns i gruppen för att

lära sig mer om front-end utveckling och nätverk. Med dessa fokusområden i åtanke, lyder frågeställningarna därför som följande:

- Hur kan man på ett enkelt sätt söka information i både lokala filer och på hemsidor med ett och samma system?
- Hur kan concurrency användas för att en server ska kunna hantera flera klienter samtidigt?
- Hur kan man balansera mängden RAM-minne och CPU som används av en process som körs i bakgrunden?
- Hur ska man strukturera användargränssnitt för att användaren ska förstå vad som händer och ta in relevant information?

De första två frågorna har med concurrency att göra, den sista frågan med användbarhet att göra, vilket är en del av specifikationen av uppgiften.

1.3 Relaterade arbeten

Det finns ett antal liknande projekt. Norconex har diverse crawlers, bland annat en filsystemscrawler[1] och riktar sig mot företag som vill utvinna information ur en stor mängd data som företagen själv samlar in. Dessa program är dock bundna till ett textbaserat användargränssnitt, det vill säga en TUI, medan Seekourny Weaver kommer att ha ett gränssnitt byggt som en hemsida. Norconex bygger gränssnitt för internt användande av professionella dataingenjörer, medan vi kommer att bygga gränssnitt som fungerar även för andra som är minimalt IT-kunniga. Ett ytterligare relaterat arbete är Spyglass[2] som är ett lokalt program som låter användaren söka i lokala filer och på webben. Dess utvecklare menar att SEO[3]-trick och reklamresultat filtreras bort, och därför ger det mer relevanta sökresultat. Spyglass är mer likt systemet som ska byggas, men har ett lokalt grafiskt gränssnitt, jämfört med en hemsida som kan hostas på en server.

2 Systemet

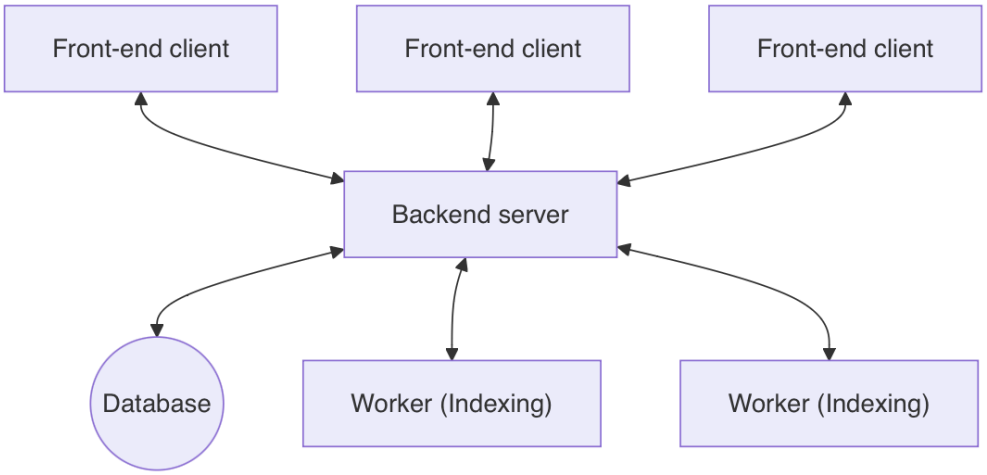
2.1 Systemarkitektur

Seekourney Weaver är en indexerad sökmotor som låter användare lätt hitta bland sina filer genom ett interaktivt sökfält som listar sökresultat efter relevans. Programmet är utformat med en frontend-klient som i “minimum viable product” (MVP) är en textprompt i terminalen och i senare versioner tar form av en hemsida.

---

[1] *Norconex*, https://opensource.norconex.com/crawlers/filesystem/
[2] *Spyglass*, https://docs.spyglass.fyi
[3] Initialism för “Search engine optimization”.

Användargränssnittsprocessen interagerar genom en API med en så kallad långlivad bakgrundsprocess (service) som i sin tur kommunicerar med en databas samt flera indexeringsprocesser genom en separat API. Dessa söker igenom filsystemet med mer och kan vara specialiserade på olika saker, och kör med samtidighet.



Figur 1; programmets kommunikationsflöde mellan processer.

De olika delarna av programmet kommer att byggas på ett sätt så att det är lätt att vidareutveckla och skala upp dessa. Huvudservern ska kunna hantera flera frontend-klienter samtidigt, samt flera indexeringsprocesser. Interagering med dess “filecrawlers” ska generaliseras för att möjliggöra vidareutvecklingar som exempelvis en separat crawler för indexering av innehållet i textfiler, och en separat “webcrawler” som indexerar vitlistade webbsidor genom hyperlänkar. Om systemet ska tillåta flera indexerare av samma typ byggs dessa (och API med huvudservern) så att de beräknar skillnaden mellan datan som redan finns i databasen mot den nya datan som hittats, och skickar den “diffen” till huvudservern. Det vill säga logiken att beräkna skillnaden i datan bryts ut till indexeringsprocesserna.

Kommunikationen mellan klienten och servern kommer att gå igenom en RESTful API. Indexerare använder en separat RESTful API och systemet kan vidareutvecklas att kräva registrering av indexerare så att användare av systemet kan lägga till egna crawlers. Det skulle även vidga målgruppen till bland annat programmerare som då skriver sina egna indexerare.

Lejonparten av programmet kommer att skrivas med språket Go. När MVP har uppnåtts kommer en enkel hemsida byggas med TypeScript, HTML och CSS samt en API mellan denna och huvudservern. I detta steg läggs även en “out-of-the-box” databas (PostgreSQL) till som ska förhindra att huvudservern tar upp för mycket RAM-minne samt låter oss hantera en obegränsad storlek på den totala indexeringsdatan.

2.2 Utvecklingar

Beroende på tid, kan ett flertal utvecklingar läggas till i projektet. Mer avancerade utvecklingar av programmet skulle kunna tillåta att huvudservern och klientprocesserna körs på olika datorer. Detta gör systemet mer användbart för organisationer, vilket är en del av projektets målgrupp.

Det är också möjligt att tillåta användaren att göra stavfel i sökningar genom “fuzzy searching”. Detta är en sökteknik som använder algoritmer som mäter likhet mellan ord. På detta sätt kan en användare få upp fler relevanta resultat utöver de som innehåller det exakta sökordet. En annan utveckling som förbättrar användbarheten är att lägga till stöd för sökning av andra filtyper som källkodsfiler och parsa dokumentation från dessa.

Förmodade prestandaförbättringar kan göras för sökfunktionen genom att byta datastrukturer. Exempelvis kan en trie användas, vilket är en trädliknande datastruktur där varje nod representerar en bokstav i ett ord. Detta används i samband med att trunkera ord i “sökträdet”, det vill säga att förenkla ord till dess rot, med hjälp av en stemmer. Det här är en lingvistisk algoritm som tar bort ändelser och böjningar från ord. Till exempel blir ordet “springer” reducerat till dess stam, “spring”.

2.3 Planering

Då Go är ett nytt programmeringsspråk för gruppens medlemmar behöver tidsplanen även reflektera detta, och tillåta en inlärningsperiod. Det finns dessutom flera delar av projektet som kräver kunskap utanför vad kursen har täckt. Detta är exempelvis sorteringsalgoritmer. Därför kommer den första veckan att läggas på att lära sig Go, samt TypeScript för de medlemmar som inte arbetat med det tidigare, och läsa på om andra nödvändiga aspekter. Under denna vecka kommer också ett första utkast på projektrapporten att skrivas och lämnas in, med deadline 4/4.

Efter den första veckan ska programmeringen börja. Mindre grupper ska delas upp för att jobba parallellt på en enkel front-end och back-end, men medlemmar kan flytta mellan dessa av behov. Ett fokus kommer ligga på att utveckla en “minimum viable product”, det vill säga ett system som funkar, men saknar mycket av de planerade funktionerna. Den preliminära planen är att arbetet med detta ska ske under 1-2 veckor. Under denna period kommer arbetet med rapporten ske parallellt och ett andra utkast kommer att lämnas in senast 23/4.

Då MVP har uppnåtts kan gruppen istället gå över till att utveckla denna med mer och bättre funktionalitet. Här inleds arbetet med de funktioner vi planerat för, och

sedan går det vidare till eventuella vidareutvecklingar om tid finns. Mycket arbete under denna period kommer också att läggas på rapporten då de två sista inlämningarna ska vara fullständiga rapporter, till skillnad från de tidigare inlämningarna, som bara ska innehålla delar av den. Det tredje utkastet ska vara inlämnat 16/5 och sedan revideras texten till det fjärde utkastet med deadline 26/5. Under hela arbetet kommer också ett antal presentationer att hållas angående projektet.

Programmet byggs upp i flera hela iterationer, från MVP till slutversion, där varje iteration motsvarar en eller flera sprinter. Implementationsdelarna för varje sprint delas upp i små “issues” och struktureras i Kanban-liknande listor i en GitHub “project board’.

2.4 Utmaningar

Eftersom Seekourney Weaver ska köras som en bakgrundsprocess är det viktigt att den inte tar upp allt för mycket av RAM-minnet. Programmet ska kunna hantera indexering av en stor mängd filer, och kommer därför att hantera mycket data. Detta måste gå snabbt för att programmet ska vara användbart, men får inte ta för mycket plats från andra processer som körs på datorn. En utmaning kommer vara att hitta en balans i detta. Eftersom detta är ett projekt som ska genomföras under en relativt kort tidsperiod, görs dock en begränsning på att detta inte måste vara så optimalt som möjligt, utan endast att mängden RAM som används upplevs som rimlig.

Ett vanligt problem är också att användare stavar fel vid sökning, och då inte får upp resultaten de förväntar sig. För att förbättra användarupplevelsen är det därför bra att implementera “fuzzy searching”, där systemet tillåter mindre stavfel och eventuellt även synonymer samt relaterade termer. Detta är dock en utveckling av programmet, och skulle kräva en tydlig tidsplanering. Även med detta kan “fuzzy searching” dock komma att bortprioriteras om utvecklingen av systemet visar sig vara mer utmanande än förväntat.

En annan viktigt aspekt av programmets användarvänlighet är hur användargränssnitt ser ut. Beroende på mängden indexerad data, samt vad som söks efter, kan resultaten bli många, och informationen bör då presenteras på ett sätt som fortfarande är tydligt. Detta inkluderar frågor som om en förhandsvisning i filer ska visas, eller hur anledningen ska presenteras till att just den filen eller hemsidan inkluderas i sökresultaten. Med en stor mängd information kommer utmaningen att hemsidan som resultaten presenteras på lätt ser rörig ut, vilket måste lösas.

3. Avslutning

Sammanfattningsvis är Seekourney Weaver ett projekt som ska underlätta för både organisationer och privatpersoner att söka bland filer och hemsidor. Detta genom en indexerande sökmotor som har flera klienter i form av hemsidor. Till en början kommer detta vara en “minimal viable product”, som sedan utvecklas vidare med flera funktioner för att göra systemet användbart. Programmet ska vara utformat som flera front-end klienter som kommunicerar med en back-end server. Denna hanterar i sin tur kommunikation med en databas samt flera indexerande “workers”. Flera vidareutvecklingar kan även göras, dels med fokus på förbättring av prestanda och dels för att göra systemet mer användarvänligt och användbart. Seekourny Weaver kommer att ge värdefulla insikter i hur ett skalbart system, med användning av concurrency, byggs upp och utvecklas.