Proiect 2 RIW

Implmentarea unui crawler Web

Student: Blejusca Oana

Grupa: 1409A

Prof. laborator: Alexandru Archip

An universitar: 2019-2020

1. Arhitectura aplicației

Aplicația propusă este compusă dintr-un modul principal. Acesta este reprezentat de modulul de crawling, implementare Java, care cuprinde majoritatea logicii necesare pentru un proces eficient de explorare a unei colecții de documente web, aici putând fi amintite modulele de HTTPFetcher, URLFrontier, precum și tehnicile de control ale zonelor de cache HTTP și DNS și tehnicile de marcare a URL-urilor vizitate.

Informația de tip TTL este foarte importantă, întrucât aceasta este principala metodă de validare a zonei de cache DNS, astfel, pentru fiecare intrare fiind configurat un obiect javax.Swing.Timer care să realizeze actualizarea resursei DNS la expirarea TTL-ului.

S-a urmărit o paralelizare eficientă a modulului de tip crawler, care este format dintr-un nod Master, care guvernează procesul de explorare, și mai multe noduri de tip Worker, care vor identifica și interoga resursele Web ce le-au fost trimise de nodul Master. S-a urmărit implementarea, la nivelul master-ului, a unei reguli de asignare dinamică a URL-urilor, pentru rezolvarea problemei politeții robotului Web, astfel se va păstra informația referitoare la ultimul domeniu accesat de către un worker, urmând ca pentru următoarele cereri să se considere alternarea domeniului, atunci când acest lucru este posibil.

Astfel, s-a implementat structura URLFrontier, un dicționar de cozi cu URL-uri caracteristice fiecărui domeniu vizitat, ciclarea între domenii folosind mulțimea de chei a dicționarului. Sincronizarea acestei resurse, pentru operațiile de scriere și citire s-a făcut prin excludere mutuală la nivelul fiecărui domeniu, prin lock-uri setate corespunzător. La nivelul clasei Master sunt păstrate mai multe informații referitoare la starea explorării, majoritatea dintre ele sincronizate prin excludere mutuală, astfel încât să se asigure consistența datelor stocate.

La nivelul structurii cache HTTP, s-a folosit, pentru estimarea euristică a domeniului de valabilitate a unei resurse, informația oferită de header-ul Last-Modified. Astfel, funcția de estimare va avea două componente, una direct proporțională cu intervalul de timp scurs de la ultima actualizare, iar cealaltă invers proporțională cu acest interval, considerând faptul că probabilitatea ca acea resursă să fie modificată va crește. Actualizarea resurselor se va face folosind un timer, setat în momentul marcării URL-ului resursei țintă ca și URL activ.

Pentru respectarea pseudoprotocolului REP a fost implementat un automat de parsare și extragere a regulilor specifice din cadrul fișierelor robots.txt, prezente la nivelul locației root corespunzătoare server-ului care deservește domeniul. Mai mult decât atât, la nivelul clasei Parser a fost inclusă o metodă care să extragă conținutu REP de la nivelul paginii, prezent prin intermediul elementelor meta, cu

atributul name setat pe valoarea robots. Au tratate doar clauzele standard, respectiv all/none, index/noindex, follow/nofollow.

2. Configurarea parametrilor principali

Aplicația nu necesită un număr mare de parametri de intrare, pentru startare.

Explorarea necesită o listă de seeds (URL-uri de la care să pornească explorarea). Acestea sunt oferite sub forma unei liste la nivelul metodei main, din clasa Master.

Pentru robot, s-a ales numele RIWEB_CRAWLER, în conformitate cu informațiile oferite de fișierul robots.txt prezent la adresa http://riweb.tibeica.com/robots.txt, întrucât acesta este singurul seed pe care l-am folosit, pentru testare. În cadrul dezvoltării aplicației s-a folosit ca și seed URL-ul http://dmoz-odp.org/, site de tip director, însă eterogenitatea mare a resurselor duce la necesitatea parsării unui număr mare de fișiere robots.txt, ceea ce împiedică atingerea vitezei de explorare prezentă la nivelul fișierului specificațiilor de proiect.

Atingerea performanțelor cerute a avut loc pe domeniul riweb.tibeica.com, unde aplicație, configurată cu 4 workeri a reușit să descarce 40 de pagini în 15 secunde cu stocare pe o zonă liberă de hard disk, respectiv 16 secunde după creșterea numărului de resurse din directorul de lucru. Folosirea unui singur worker a rezultat în salvarea locală a 90 de pagini în limita de un minut, atunci când numărul resurselor din directorul de lucru era mare, respectiv 94 atunci când directorul de lucru era gol. Timpul variază puternic în funcție de disponibilitatea severelor vizitate (unele resurse sunt invalide, alte nume de domenii nu sunt vizibile în DNS), viteza, chiar și la nivelul procesării secvențiale fiind destul de mare atunci când nu este parcurs nici un fișier robots.txt (aproximativ 2-3 pagini salvate local pe secundă, aici fiind excluse redirectările, sau eventualele resurse neidentificate la nivel de protocol DNS-nume de domenii invalide sau HTTP-resurse ce nu pot fi identificate la nivel de domeniu).

Testele au fost realizate pe o platformă de tip mobil (laptop), cu un procesor Intel i7-6700HQ, cu 4 nuclee fizice, 8 GB RAM DDR4, respectiv o conexiune Internet Wi-Fi, cu aproximativ 20-25 Mbps viteză de download.

La nivelul actualizării resurselor stocate local, s-au folosit cereri HTTP condiționale, prin folosirea header-ului If-Modified-Since, setat corespunzător, cu ultimul moment de timp la care resursa a fost accesată. Obținerea codului de răspuns 304 este folosită ca dovadă a gradului de actualitate a resursei. Nici aceste

interogări nu au făcut parte din numărul resurselor considerate pentru scenariile de test descrise mai sus.