Paralelný web crawler*

Peter Kapusta
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xkapustap@stuba.sk

5. novembra 2023

Abstrakt

V nasledujúcom článku sa budeme zaoberať využitím paralelných web crawlerov, ako aj problémom optimalizácie (spolu s metódou, ako odľahčiť dopad web crawlerov na jednotlivé webstránky), postupne uvedieme niekoľko riešení, ktoré sa momentálne využívajú a ich spôsoby komunikácie medzi jednotlivými inštanciami.

1 Úvod

Vo všeobecnosti, web crawler je program, ktorý systematicky skenuje web, objavuje webstránky a následne z nich extrahuje informácie a odkazy, ktoré vedú k novým webstránkam. Cieľom je teda čo najefektívnejšie a najrýchlejšie objaviť čo najväčšie množstvo informácií, ktoré môže neskôr využiť napríklad prehliadač. Hlavnými ťažkosťami sa pri jednoduchých web crawleroch stávajú prehľadávanie HTML kódu a získavanie využiteľných odkazov, čo však na prehľadanie a indexovanie celého internetu nestačí - vzniká potreba pre sťahovanie a spracovanie masívneho množstva informácií. Pre vyriešenie tohto problému je treba použiť viac ako jeden crawler. Tu prichádzajú do hry paralelizované web crawlery, ktoré sú výnimočné ich schopnosťou súčasne získavať odkazy z rôznych zdrojov, pomocou osobitných inštancií, bez toho, aby sa prekrývali – tým sa čo najoptimálnejšie využije maximálna rýchlosť sťahovania na danej sieti.

Pri práci s paralelnými web crawlermi je vhodné mať jasný obraz o tom, čo sú vlastne inštancie, ako vyzerajú vzťahy medzi nimi a ako sa medzi sebou prepájajú webstránky, ktorými musia inštancie prechádzať. Pre uľahčenie tohto cieľa si definujeme nasledovné pojmy: Internet môžeme vnímať ako orientovaný graf, ktorého vrcholy sú reprezentované jednotlivými webstránkami a hrany tvoria odkazy medzi nimi [1]. Inštancia je proces, ktorý asynchrónne prehľadáva internet a snaží sa nájsť prepojenia, ktoré ho dovedú k novým, ešte neobjaveným informáciam. Úlohou paralelného web crawleru je teda optimálne prechádzať komplexným internetovým grafom, a to pomocou viacerých inštancií, bez toho, aby sa ich cesty krížili.

*Semestrálny projekt v predmete Metódy inžinierskej práce ak. rok 2023/24, vedenie: Ing. Richard Marko, PhD.

Obr. 1: Orientovaný graf reprezentujúci vzťah medzi webstránkami

Obr. 2: Princíp získavania informácií web crawlerom [2]

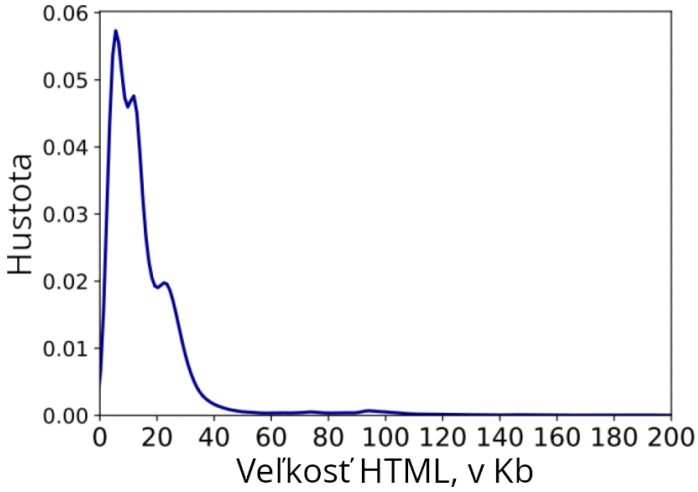
2 Porovnanie rýchlosti

2.1 Podmienky pri testovaní

Princíp fungovania paralelných web crawlerov vyžaduje implementáciu, ktorá je rýchla a nenáročná na zdroje. Táto požiadavka sa stáva nevyhnutnosťou, vzhľadom na to, že indexovaný web obsahuje minimálne 5,27 miliárd webstránok [3]. Exituje mnoho spôsobov, ako implementovať samotné procesy jednolivých web crawlerov. V tejto časti porovnáme rozdiely vo výkone web crawlerov, ktoré boli vytvorené pomocou procesov, threadov (vlákien) a systémového volania `epoll` [1], napísané v programovacom jazyku C++ (vybraný pre rýchlosť a knižnicu `pthreads`, ktorá je v ňom dostupná) [3].

Pri testovaní implementácií, parametre ako kompilátor (GCC 9.3.0), webserver (operačný systém Ubuntu 20.04 LTS a serverový softvér NGINX), programovací jazyk (C++) a optimizácie pri kompilácii (`O3` [2]) zostávajú nezmenené. Testy budú spočívať v rýchlosti stiahnutia 132094 HTML súborov, ktoré sú uložené na serveri, s rozložením veľkostí podľa nasledujúceho grafu [3]:

[1] Rozhranie poskytované linuxovým jadrom, určené pre oznamovanie I/O udalostí [4]
[2] Stupeň optimalizácie kompilátora GCC určený pre zvýšenie rýchlosti programu [5]



Obr. 3: Graf distribúcie veľkosti HTML súborov. [3]

2.2 Podrobnosti implementácií

Implementácia, ktorá využíva procesy, ich vytvára pomocou systémového volania `fork` [3] - zároveň využíva aj funkcie `mmap` a `pthreads` pre zjednodušenie synchronizácie procesov. Počet inštancií pre jednotlivé implementácie bol vybraný tak, aby každá implementácia dovŕšila jej najväčší možný výkon [3].

2.3 Výsledky

Z experimentov napokon vyplývajú nasledujúce výsledky (v poradí od najrýchlšej implementácie po najpomalšiu) [3]:

1. Systémové volanie `epoll`
2. Thready (vlákna)
3. Procesy

Môžeme konštatovať, že systémové volanie `epoll` je najrýchlejšie v porovnaní s ostatnými testovanými implementáciami. Má aj nevýhody - `epoll` je dostupný iba na linuxovom jadre (existujú aj alternatívy dostupné v iných jadrách, ako napríklad `kqueue` na BSD systémoch), rovnako je ho ťažké implementovať, čo môže mať za následok chyby a bezpečnostné problémy. Thready majú síce nižší výkon, ale je ich ľahšie naprogramovať. Nakoniec implementácia pomocou procesov je síce najpomalšia, ale pre využitia, kde najvyššia možná rýchlosť nie je na prvom mieste, je postačujúca. [3]

[3] Funkcia, ktorá vytvorí z rodičovského procesu nový proces (program sa rozdvojí) [4]

3 Zníženie záťaže na servery

3.1 Princíp a implementácia

40% prenosu dát internetom je spôsobených web crawlermi, preto by bolo vhodné nájsť riešenie nielen pre sťahovanie čo najväčšieho množstva dát, ale aj pre minimalizáciu opätovného sťahovania webstránok, ktoré už máme v databáze, za účelom aktualizovania ich obsahu. Riešenie spočíva v pridaní súboru UPDATE do každej webstránky na internete. Tento súbor by obsahoval odkazy na adresáry webstránky, kde nastala zmena [6].

Algoritmus crawlovania [6]:

1. Crawler po prvýkrát navštívi webstránku a stiahne celý adresár (podľa súboru `robots.txt` [4])
2. Pre aktualizáciu databázy crawler najskôr navštívi súbor UPDATE
3. Porovná ho s UPDATE súborom z poslednej návštevy webstránky
4. Ak sú súbory rôzne, crawler stiahne aktualizované stránky a uloží ich do databázy

Súbor môže samozrejme pridať len sám vlastník stránky, čo by vyžadovalo veľkú úroveň koordinácie. Zo strany vlastníkov to má však výhodu - ich server nie je zaplavený žiadosťami o stiahnutie celého adresára, čo znižuje nároky na údržbu. Namiesto toho, aby crawlery zakaždým sťahovali celý adresár, stačí im overiť si UPDATE súbor. Zo strany crawlerov vzniká výhoda, že nedochádza k časovo náročnému prehľadávaniu webstránky, ktorej najnovšiu verziu už majú v databáze [6].

Existencia súboru `robots.txt` vo veľkom množstve webstránok dokazuje, že je možné takéto riešenie implementovať. Každý web crawler by mal byť schopný prehľadávať HTML kód, a každý vlastník webstránky by mal byť schopný upravovať súbor s HTML kódom, obsahujúci aktualizácie jeho webstránky. Preto by sa mal aj UPDATE súbor skladať z HTML [6].

3.2 Simulácia a výhody

V experimente bola otestovaná teória stiahnutím najprv celého adresára webstránky, potom pomocou súboru UPDATE s jednou, dvomi a tromi aktualizáciami. Z výsledkov vyplýva, že so súborom s jednou aktualizáciou sa dokopy stiahne 6,5-krát menej stránok ako bez neho [6].

Implementácia UPDATE súboru nevyžaduje sťahovanie žiadnych nových programov zo strany vlastníkov a úprava softvéru crawlerov je minimálna. Je to celkovo veľmi optimálne riešenie problému nadmerného toku crawlerov na internete [6].

[4] Súbor, ktorý oznamuje web crawlerom, ktoré časti webstránky môže alebo nemôže indexovať

4 Moderné riešenia

4.1 C-proc

Paralelné web crawlery sa skladajú z viacerých komponentov, ktoré sa dajú zrealizovať viacerými spôsobmi. Jeden z takých spôsobov je vytvorený pomocou crawlovacích inštancií C-proc. Každý C-proc je oddelený proces, ktorý nezávislo prehľadáva web. Výhodou takejto architektúry je zníženie záťaže na lokálne siete, pretože každý C-proc môže bežať na inej sieti. Jednotlivé inštancie sú ovládané centrálnym programom na správu crawlerov [7].

Paralelné web crawlery typu C-proc sa rozdeľujú na nasledovné časti [7, 8]:

1. Viacvláknový server - Koordinuje činnosť jednotlivých klientov.
   (a) URL Dispatcher
   (b) DNS Resolver
   (c) URL Mapper
2. Inštancie crawlerov - Jednotlivé programy C-proc, ktoré sťahujú webstránky

4.2 Ergate

Paralelný web crawler Ergate funguje na podobnom princípe ako C-proc, ale odstraňuje problém spoliehania sa na jeden program, ktorý riadi všetky inštancie. Tento cieľ je dosiahnutý pomocou distribuovanej architektúry, zloženej z kontrolného systému a systému na sťahovanie stránok - každý z týchto systémov sa môže skladať z viacerých inštancií ovládačov a crawlerov. Jednotlivé časti komunikujú medzi sebou cez MPI [5] a celok sa nazýva Cluster [9].

[5] Message Passing Interface - rozhranie na komunikáciu medzi procesmi

Literatúra

[1] Shruti Sharma and Parul Gupta. The anatomy of web crawlers. In *International Conference on Computing, Communication & Automation*, pages 849–853, 2015.

[2] Md Abu Kausar, VS Dhaka, and Sanjeev Kumar Singh. Web crawler: a review. *International Journal of Computer Applications*, 63(2):31–36, 2013.

[3] Andriy Sultanov, Maksym Protsyk, Maksym Kuzyshyn, Daria Omelkina, Vyacheslav Shevchuk, and Oleg Farenyuk. Comparison of performance of the popular approaches to implementing parallel crawlers. In *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, volume 1, pages 349–352, 2021.

[4] M. Kerrisk. *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. No Starch Press, 2010.

[5] Richard M Stallman. *Using the gnu compiler collection: a gnu manual for gcc version 4.3. 3*. CreateSpace, 2009.

[6] Shekhar Mishra, Anurag Jain, and AK Sachan. Smart approach to reduce the web crawling traffic of existing system using html based update file at web server. *International Journal of Computer Applications*, 11(7):34–38, 2010.

[7] Shruti Sharma, AK Sharma, and JP Gupta. A novel architecture of a parallel web crawler. *International Journal of Computer Applications*, 14(4):38–42, 2011.

[8] Tithi Dhar, Sayan Mazumder, Susnigdha Dhar, Susovan Karak, and Debraj Chatterjee. An approach to design and implement parallel web crawler. In *2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, pages 1–4, 2021.

[9] Min Wu and Junliang Lai. The research and implementation of parallel web crawler in cluster. In *2010 International Conference on Computational and Information Sciences*, pages 704–708, 2010.