UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Surf

Parcurgere și procesare distribuită a paginilor web în cloud pentru extragerea informațiilor

propusă de

Ovidiu Pricop

Sesiunea: Iulie, 2017

Coordonator științific

Conf. dr. Sabin Corneliu Buraga

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Surf

Ovidiu Pricop

Sesiunea: Iulie, 2017

Coordonator științific

Conf. dr. Sabin Corneliu Buraga

Declarație privind originalitate și respectarea drepturilor de autor

Prin prezenta declar că lucrarea de licenţă cu titlul "Surf" este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, 3 Iulie 2017

Ovidiu Pricop

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul "Surf", codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică. De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, 3 Iulie 2017

Ovidiu Pricop

Rezumat

La nivelul Internetului, traficul global a crescut de aproximativ 850 de ori în perioada 2000 - 2015 [1]. World wide web-ul reprezintă o parte semnificativă a volumului de informaţii interschimbate pe Internet. În anul 2015 existau peste jumătate de miliard de situri web accesibile [2]. Fiecare pagină web răspunde anumitor nevoi (sociale, financiare, educaţionale etc.). O singură sursă de informaţii este uneori suficientă pentru a răspunde nevoilor unui utilizator. Alteori, este necesar un ansamblu de surse informative (e.g. newsletter-e zilnice din 5 surse diferite), pentru a urmări un subiect din mai multe perspective sau a-i întregi conţinutul.

Prezenta lucrare urmăreşte elaborarea unui serviciu web distribuit în cloud pentru parcurgerea, selectarea, colectarea şi indexarea informaţiilor la nivelul world wide web-ului. Serviciul se adresează acelor persoane care vor să automatizeze sarcinile de extragere a informaţiilor web şi garantează uşurinţă în utlizare, flexibilitate, extensibilitate şi control precis asupra costurilor.

Cuprins

Introducere

Un serviciu web reprezintă o componentă funcțională ce îndeplinește anumite sarcini. Comunicarea cu un serviciu web se realizează independent de platformă, limbajul de programare sau sistemul de operare pe care este dezvoltat. Schimbul de informații se realizează prin mesaje text ce respectă un format standardizat, precum XML [1] sau JSON[2].

Aplicatia "Surf" reprezintă un serviciu web specializat în web crawling[3], dezvoltat folosind tehnologii cloud din cadrul Amazon Web Services. Se urmărește crearea unui serviciu web cu disponibilitate permanentă, scalabil și de înaltă putere computațională care să orchestreze colectarea distribuită de informații din aria definită de utilizator.

Comunicarea cu aplicația "Surf" se realizează prin intermediul unui API RESTful[4] construit pe platforma AWS[5] API Gatway. Autentificarea utilizatorilor se va realiza printr-un serviciu OpenID Connect (e.g. Google, Facebook etc.). Autorizarea va avea ca principală componentă AWS IAM. Utilizatorilor le vor fi repartizate, în funcție de privilegiile asociate cu cheia de autentificare, o serie de roluri (i.e. drepturi de access asupra resurselor din cadrul serviciului "Surf"). Execuția codului propriu-zis, găzduit de funcții AWS Lambda, va interacționa cu serviciul pentru baze de date NoSQL AWS DynamoDB pentru a permite accesul la informații cheie pentru funcționalitatea aplicației (metadate crawling, date acces utilizatori etc.). Mediul de procesare distribuită va fi susținut de AWS Step Functions și configurat după preferințele utilizatorului. Datele extrase din procesul de web-crawling vor fi salvate în mediul persistent de stocare AWS S3.

[1]Extensible Markup Language - https://www.w3.org/XML/
[2]JavaScript Object Notation - http://www.json.org/
[3]Web crawler - https://www.techopedia.com/definition/10008/web-crawler
[4]REST - http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm
[5]Amazon Web Services - https://aws.amazon.com

1 Descrierea unui crawler web

Natura dinamică World Wide Web-ului, precum şi scara sa, justifică necesitatea existenţei unor mecanisme eficiente de localizare a informaţiilor relevante. Instrumentele specializate în căutarea datelor se bazează pe web crawler-e pentru colectarea informaţiilor din paginile web, ce sunt, ulterior, indexate şi analizate.

Crawlerul ajută utilizatorul în ceea ce priveşte navigarea Internetului, automatizând parcurgerea link-urilor. Un crawler exploatează structura web-ului şi extrage resurse, într-o manieră ordonată, ghidat de criteriile specificate de către utilizator. Rezultatul obţinut de pe urma utilizării unui web crawler este o serie de situri web.

În documentele de referinţă, web crawler-ii mai sunt cunoscuţi şi drept "wanderers", "robots", "spiders", "fish" sau "worms". Aceste denumiri nu sunt indicatori ai performanţei, întrucât aceste programe pot fi foarte rapide şi precise. Se pot parcurge zeci de mii de pagini într-un interval de câteva minute, consumând doar o fracţiune din lungimea de bandă pusă la dispoziţie [5], atât timp cât crawler-ul dispune de resursele hardware necesare.

Crawler-ii web deservesc unui număr mare de scopuri. Unul dintre acestea este mentenanţa bazei de date în care sunt stocaţi indecşii unui motor de căutare. Pentru această sarcină sunt utilizaţi crawleri de tip exhaustiv (se parcurg toate siturile web întâlnite). O altă categorie este cea a crawler-ilor bazaţi pe conţinut, folosiţi pentru adunarea datelor ce se înscriu sub o anumită tematică, minimizând efortul de a aloca resurse pentru pagini irelevante. În construirea unui astfel crawler este necesară definirea unei modalităţi de căutare şi selecţie a datelor. În cazul crawlerilor bazaţi pe conţinut, pot apărea probleme legate de explorare versus exploatare, deoarece spaţiul de căutare ar putea conţine un optim local, împiedicând algoritmul să localizeze optimul global.[6]

2 Design-ul unui crawler web

Ca și design, un crawler web se rezumă la trei elemente: o *frontieră*, în care sunt stocate URL-urile ce urmează a fi vizitate, o componentă a cărei scop este de a *descărca paginile* de pe World Wide Web și un spațiu de stocare, unde să fie salvate datele preluate de la crawler (e.g. o baza de date și/sau un sistem de fișiere) [4, 5].

2.1 Frontiera

În terminologia grafurilor, frontiera reprezintă lista nodurilor nevizitate [6]. Pentru un crawler web, frontiera reprezintă o listă de adrese web neparcurse. Procesul de căutare pornește de la o adresă numită *rădăcină*, care este preluată din web și procesată, urmând ca link-urile găsite să fie adăugate în frontieră. Această listă poate căpăta dimensiuni foarte mari încă de la începutul procesului de parcurgere a paginilor, întrucât media numărului de legături de pe o pagină web este de 7 link-uri/pagină [7]. Procesul se repetă, alegându-se, folosind un anumit algoritm, link-uri din frontieră, până când frontiera se golește, sau altă condiție de oprire permite finalizarea căutării.

2.2 Descărcarea paginilor web

Este necesar un client HTTP care trimite o cerere HTTP și primește un raspuns. În această etapă, nu trebuie trecut cu vederea "The Robots Exclusion Standard", cunoscut și drept "Robots Exclusion Standard" sau robots.txt [8]. Protocolul oferă administratorilor de servere web posibilitatea de a-și comunica politicile de acces și de a menționa paginile care nu sunt destinate crawler-ilor.

2.3 Stocarea datelor

În urma parcurgerii paginilor web, datele obținute sunt stocate într-un mediu potrivit nivelului lor de abstractizare. Spre exemplu, metadatele vor fi stocate într-un mediu în care interogarea să se poată realiza în mod eficient, iar paginile în format HTML vor fi păstrate într-un mediu de stocare mai încăpător.

[6] "[...] unexpanded (unvisited) nodes." [6]

[7] "The web may be viewed as a directed graph in which each vertex is a static HTML web page, and each edge is a hyperlink from one web page to another. Current estimates suggest that this graph has roughly a billion vertices, and an average degree of about 7."[7]

3 Crawler-ul Surf

Ca sens general, un crawler web reprezintă un program care parcurge, prin cereri succesive, situri web. Programatic, vom considera următoarele aspecte cheie în proiectarea unui crawler distribuit:

1. Punctele de pornire în parcurgerea siturilor web;
2. Adâncimea maximă a parcurgerii recursive a siturilor (i.e. cea mai îndepărtată pagină la care se poate ajunge, de la punctul de pornire, prin accesarea succesivă a legaturilor de tipul hyperlink;
3. Politica de selecție a informațiilor din paginile parcurse[9];
4. Viteza de crawling, fișierul *robots.txt*[8] și respectarea drepturilor de autor;
5. Politica de paralelizare a procesului de web crawling;
6. Politicile de retenție temporară a rezultatelor parcurgerii siturilor web și jurnalizare a acțiunilor serviciului de crawling;

În cele ce urmează, se va descrie particularizarea aspectelor generale enumerate mai sus în cadrul serviciului web "Surf". Se va crea, astfel, contextul dezvoltării aplicației și se vor puncta principalele componente funcționale implicate.

3.1 Puncte de pornire

Pentru a putea parcurge siturile web, crawler-ul "Surf" necesită unul sau mai multe puncte de pornire. Un punct de pornire este definit printr-un URL și este furnizat, ca argument din partea utilizatorului, la inițializarea crawler-ului.

3.2 Parcurgere

Crawler-ul web "Surf" are ca scop extragerea informațiilor cerute de către utilizator. Întrucat utilizatorul are posibilitatea de a furniza, ca punct de pornire, un sit web relevant pentru informația căutată, parcurgerea recursivă se va executa în manieră breadth-first. Așadar, crawler-ul va vizita toate legăturile de tip hyperlink din pagina curentă a parcurgerii înainte de a accesa legăturile din pagina următoare din punct de vedere ierarhic. Adâncimea maximă a arborelui de legături realizat prin parcurgerea URL-urilor va fi definită de către utilizator, la inițializarea sesiunii de crawling.

[8] http://www.robotstxt.org/robotstxt.html

3.3 Selecția informațiilor

Selecția informațiilor necesită parcurgerea siturilor web aflate la adresele URL pe care crawler-ul le are în vedere, procesarea, analizarea și prelucrarea informațiilor în funcție de tipul lor (e.g. html, xml, json, text) și extragerea blocurilor de conținut aferente.

Un "bloc de conținut" asociat unui URL reprezintă o parte a întregului conținut aflat la URL-ul respectiv, filtrată după anumite caracteristici stabilite de utilizatorul serviciului web. Aceste caracteristici pot fi:

- Pentru fișiere HTML/XML:
  - selectori CSS/jQuery [18]
- Pentru fișiere text:
  - cuvinte cheie

Pentru tipurile de conținut enumerate mai sus, sau pentru alte tipuri de conținut aflate la adresele URL vizitate de crawler, utilizatorul poate defini filtre bazate pe conținutul textual al URL-ului. Spre exemplu, se pot evita toate URL-urile care nu satisfac o anumita expresie regulată.

Evitarea selectării informațiilor duplicate se ameliorează prin normalizarea[9] URL-urilor parcurse de către crawler.

3.4 Viteză, politici de respingere și drepturi de autor

Siturile web pot implementa variate modalităţi de contracarare a încercărilor de crawling. Aplicaţia "Surf" încearcă să minimizeze riscul de respingere a cererilor de accesare a anumitor resurse web printr-o implementare neintruzivă a procesului de crawling. Un aspect esenţial care este luat în considerare în ceea ce priveşte o astfel de implementare este minimizarea volumului de date preluat de pe un anumit domeniu prin diferite metode de filtrare a linkurilor urmărite (e.g. o anumită structură a URL-ului, un anumit tip de date care se găseşte la URL-ul respectiv).
Crawler-ul web "Surf" poate satisface numeroase cerinţe ale utilizatorilor. O parte dintre aceste cerinţe poate veni din partea sistemelor anti-malware. În acest caz, nu se doreşte respectarea fişierului *robots.txt* (utilizat drept referinţă, pentru crawleri, asupra URL-urilor accesibile ale domeniului vizitat), deoarece

[9] https://tools.ietf.org/html/rfc3986#section-6

există pericolul ca un sit maliţios să blocheze o eventuală scanare. De aceea, aplicaţia "Surf" va putea fi configurată în ceea ce priveşte ignorarea fişierului *robots.txt* în procesul de parcurgere a unui domeniu.

Crawler-ul "Surf" implementează un mecanism de *blacklisting*[10]. Siturile web sau domeniile care interzic procesul de crawling (e.g. prin Terms of Service[11]) vor fi adăugate unei liste de excluziune din procesul de parcurgere executat de crawler.

3.5 Paralelizare

"Surf" implementează un mecanism de crawling distribuit. Mai exact, există posibilitatea de a împărți sarcinile de parcurgere a paginilor web prin rularea concurentă a mai multor instanțe de funcții Lambda[12].

Există doua moduri bine cunoscute de alocare a sarcinilor de web crawling: statică și dinamică[10]. Aplicația "Surf" implementează un mecanism de distribuire statică a sarcinilor:

> Fie n, *gradul de paralelizare*[13] configurat pentru execuția curentă a web crawler-ului. Alocarea statică va asocia fiecărui URL candidat pentru crawling un număr întreg în intervalul $[1,n]$ reprezentând identificatorul uneia dintre cele n instanțe paralele ale web crawler-ului. Sarcina va fi distribuită pe acea instanță și va fi executată.

3.6 Logarea acțiunilor și retenția datelor

În timpul execuției, crawler-ul generează evenimente ce sunt adaugate la istoricul execuției curente. Aceste evenimente sunt utile, în primul rând, în cazul în care execuția instanței curente a crawler-ului eșuează. Sarcinile de crawling web pot dura o perioadă considerabilă și nu este dezirabilă refacerea întregului proces de parcurgere a siturilor, atât timp cât există rezultate parțiale disponibile; execuția se poate relua pornind de la ultimul eveniment valid înregistrat în istoric. În al doilea rand, istoricul este util pentru utilizator, deoarece îl poate avertiza în legătură cu starea execuției curente a web crawler-ului.

[10] http://dictionary.cambridge.org/dictionary/english/blacklist

[11] http://www.pcmag.com/encyclopedia/term/62682/terms-of-service

[12] *Amazon Web Services Lambda Functions*:Pe baza principiilor programării funcționale, se garantează că funcțiile *Lambda* nu au efecte colaterale, deci nu pot schimba starea globală a sistemului. Din acest motiv, funcțiile *Lambda* se pretează la paralelizare

[13] Numărul maxim de execuții simultane de sarcini de crawling

Datele rezultate în urma procesului de crawling al paginilor web sunt salvate în serviciul persistent de stocare S3, din cadrul Amazon Web Services. Deoarece aceste date pot deveni volumnioase (în funcție de politica de selecție configurată), aplicația "Surf" salvează metadate asociate acestor rezultate într-o tabela DynamoDB. Pe baza acestor informații (metadate), utilizatorul are opțiunea de a accesa rezultatele complete corespunzatoare rulării crawler-ului web, aflate în S3.

Crawling în cloud

Aplicația "Surf" este construită și rulează în cloud. Dezvoltarea unei aplicații pe suportul unei infrastructuri cloud este substanțial diferită față de abordarea clasică, pe o singură mașină de calcul. Cloud computing-ul pune la dispoziție, la cerere, un set de resurse (e.g. putere de procesare, spațiu de stocare) pe care un utilizator (e.g. aplicația "Surf") le poate folosi. Faptul că toate aceste resurse sunt administrate de fiecare serviciu cloud în parte simplifică sarcinile consumatorului: se pot solicita mai multe sau mai puține resurse, după gradul de încărcare al aplicației dezvoltate în cloud. Acest lucru se realizează într-o manieră transparentă pentru utilizator, fără a necesita efort suplimentar din partea acestuia.

Dezvoltarea aplicației "Surf" folosind servicii cloud mai prezintă un avantaj considerabil, deoarece comunicarea prin Internet[14] între componentele aplicației și, implicit, utilizarea unui format agnostic de arhitectură sau limbaj de programare, conduce la un grad mare de decuplare. Fiind decuplate, componentele aplicației pot fi construite, testate și depanate[15] în izolare și pot funcționa ca module (plugin-uri) în alcătuirea arhitecturii sistemului.

A dezvolta un serviciu web folosind exclusiv tehnologii cloud permite realizarea unui sistem de subscripții granular. Un utilizator al aplicației "Surf" poate solicita una sau mai multe instanțe ale crawler-ului care să ruleze în paralel, în funcție de necesitățile proprii. Serviciile cloud permit aplicației să scaleze orizontal ca timp, permițând utilizatorului realizarea mai multor sarcini de parcurugere automată web (deci, în consecință, acoperirea unui volum mai mare de informații) în același interval de timp.

Un alt aspect relevant în realizarea serviciului web "Surf" este caracterul său *serverless*. Infrastructura *serverless* implică execuția codului sursă al progra-

[14]Protocolul de comunicare folosit este *Transmission Control Protocol*[19], întrucat acesta garantează transmiterea nealterată a datelor între două noduri ale rețelei

[15]*engl.* debugged

mului în cadrul serviciilor cloud folosite pentru dezvoltarea aplicației. Cererile de execuție a codului sunt monetizate conform unei măsuri abstracte corespunzătoare resurselor utilizate pentru satisfacerea solicitării, ceea ce diferă față de modelul clasic, în care ar fi trebuit achiziționat hardware în acest scop[11].

1 Infrastructura

Esența aplicației "Surf" se află în codul executat de serviciul web AWS Lambda, care preia componentele dezvoltate pentru crawling web și le execută, drept programe autonome, folosind infrastructura AWS. AWS Lambda execută acest cod doar când este nevoie (e.g. la cererea utilizatorului care dorește să demareze operațiunea de crawling). Se minimizează, astfel, atât costurile utilizatorului în ceea ce privește rularea crawler-ului web (nu necesită hardware dedicat pentru procesarea paginilor web), cât și costurile dezvoltatorilor crawler-ului, care nu trebuie să rezerve hardware în cloud (e.g. mașini virtuale Amazon EC2, închiriere de mașini gazdă), ca, mai apoi, când nu există cereri suficiente, aceste mașini de calcul să ramână nefolosite. De asemenea, se renunță la necesitatea de a avea un load-balancer care să distribuie sarcinile de crawling pe capacitatea hardware disponibilă, deoarece acest lucru este administrat, în fundal, de catre AWS Lambda.

Funcțiile programatice disponibile prin serviciul AWS Lambda pot fi accesate de către utilizatori printr-un API Restful, găzduit de AWS API Gateway. Această platformă permite integrarea cererilor HTTP externe cu mecanismul de autentificare și autorizare a utilizatorilor și codul executat de instantele funcțiilor Lambda. De asemenea, aplicația utilizează funcționalități de management, monitorizare și analiză a traficului din cadrul API-ului "Surf" care permit, printre altele, înregistrarea și monetizarea funcționalitatilor oferite de serviciul web pentru fiecare utilizator în parte.

Pentru a putea accesa funcționalitățile oferite de API-ul "Surf", utilizatorii trebuie să se autentifice cu un furnizor terț de identități OpenID[16]. Credențialele obținute sunt împachetate într-o cerere catre AWS Security Token Service[17], unde sunt validate. În cazul în care validarea este îndeplinită cu succes, Security Token Service returnează utilizatorului credențiale temporare pentru a accesa servicii din cadrul AWS. Utilizatorul trebuie să includă aceste credențiale în fiecare cerere catre API-ul "Surf" (autentificare). Odată autentificat, utilizatorul

[16]http://openid.net/what-is-openid/

[17]Serviciu web ce permite unui utilizator să solicite credențiale temporare pentru autentificarea în cadrul platformei Amazon Web Services

poate interacționa doar cu acele resurse ale API-ului pentru care are drepturi de acces.

Figura 1: Procesul de autentificare [12]

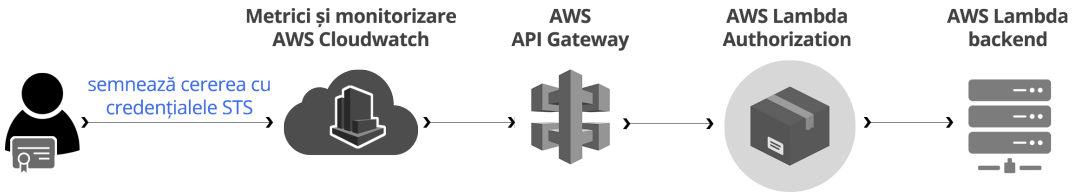


Figura 2: Procesul de autorizare [12]

Prin AWS Identity Access Management (IAM) se vor stabili restricțiile de acces asupra API-ului. Mecanismul de asignare a permisiunilor va fi susținut prin definirea unor roluri. Utilizatorii vor fi grupați, din punct de vedere al drepturilor de acces, în mai multe categorii. Spre exemplu, un utilizator poate avea rolul "administrator" (prescurtat *Ad*), iar un alt utilizator poate avea atât rolul "administrator" (*Ad*), cât și "utilizator de bază" (prescurtat *Ub*). Atât *Ad*, cât și *Ub* vor avea asignate politici de acces asupra datelor AWS. O astfel de politică este următoarea:
Politica de acces de mai sus îi permite utilizatorului ce îi este asignată accesul la toate resursele API-ului "Surf". Această politică este potrivită pentru un administrator, dar periculoasă pentru un utilizator ce accesează pentru prima data serviciul "Surf". De aceea, se vor defini drepturi de acces granulare care să

```
{
  "Version": "2012-10-17", // Stabilită de către furnizor
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Figura 3: Politică de acces IAM

aibă în vedere restricționarea la nivel "need-to-know"[18] pentru fiecare entitate ce trimite cereri către API (autorizare).

Procesul de asignare a rolurilor pentru utilizatori este coordonat atât manual (pe bază de ierarhie: administrator - utilizator de baza), cât și automat (i.e. printre altele, toți utilizatorii abia înregistrați sunt utilizatori standard). Asignarea manuală a permisiunilor are prioritate asupra asignării automate. Datele despre rolul fiecărui utilizator vor fi păstrate în baza de date.

Crawler-ul "Surf" utilizează AWS DynamoDB ca suport pentru baza de date. DynamoDB reprezintă un serviciu cloud scalabil pentru baze de date NoSQL. Datele dintr-o bază de date non-relațională NoSQL) pot fi modelate fără constrângerile unei baze de date relaționale (e.g. tabularitate). Acest lucru atrage, în cadrul aplicației "Surf", elemente operaționale cheie, pentru beneficiul cărora s-a renunțat la o abordare relațională, de tip SQL (e.g. Oracle), printre care:

- *data sharding*[19] - este necesar un sistem distribuit de baze de date care să poată scala orizontal odată cu dimensiunea datelor din baza de date și odată cu creșterea numărului de utilizatori;
- *scheme de date dinamice*[20] - se dorește posibilitatea schimbării structurii

[18] http://dictionary.cambridge.org/dictionary/english/on-a-need-to-know-basis
[19] https://docs.microsoft.com/en-us/azure/architecture/patterns/sharding
[20] https://www.mongodb.com/scale/dynamic-schema-design

datelor din baza de date fără a recrea tabelele corespunzătoare; acest lucru trebuie avut în vedere deoarece crawler-ul stochează în baza de date metadate obținute prin parcurgerea paginilor web; poate apărea oricând necesitatea introducerii unor noi metadate sau schimbării structurii celor existente, cu scopul satisfacerii nevoilor utilizatorilor.

Câteva dintre cele mai importante roluri ale bazei de date sunt găzduirea metadatelor asociate rezultatelor procesului de crawling, păstrarea asignărilor între identitățile utilizatorilor și rolurile lor, menținerea istoricului evenimentelor de crawling cu scopul reconstruirii stării procesului de parcurgere a paginilor web, în cazul unei erori, și stocarea datelor asociate volumului de trafic generat de fiecare utilizator în cadrul infrastructurii AWS.

Pentru a obține un serviciu web distribuit pentru crawling este necesară coordonarea activităților independente din cadrul sistemului, sincronizarea pașilor necesari procesului de crawling și, în final, integrarea rezultatelor. Pentru acest lucru, aplicația "Surf" folosește serviciul web AWS Step Functions (SFN). SFN are capacități de coordonare a activităților ce se doresc îndeplinite (i.e. execuția funcțiilor Lambda, sau *Lambda Tasks*) și management al stării aplicației (i.e. se pornește un proces de agregare a datelor provenite de la crawlerii care au rulat în paralel doar după ce aceștia și-au terminat execuția).

Datele obținute în urma procesului de crawling sunt stocate utilizând serviciul web Amazon S3. Fiecare rulare a unei instanțe a crawler-ului distribuit generează un *bucket*[21] S3. Datele din bucket-uri sunt disponibile, pentru utilizatori, prin plasarea de metadate, precum numele bucket-ului, într-o tabelă DynamoDB, pe care utilizatorii o pot accesa prin intermediul API-ului.

[21]Unitate, în cloud (AWS S3), ce stochează date și căreia îi pot fi atribuite permisiuni de acces și metadate corespunzatoare

2 Generarea infrastructurii

Procesul de generare a infrastructurii (engl. 'deployment') constă în crearea resurselor și pornirea sistemelor necesare pentru ca web-crawler-ul 'Surf' sa funcționeze. Mecanismul de generare a resurselor execută următorii pași:

- Crearea entităților necesare în AWS (e.g. funcții Lambda, roluri IAM, API-ul APIGateway etc.);
- Stabilirea relațiilor între resurse și injectarea dependențelor (e.g. unei rute a API-ului trebuie să îi fie asociat un rol IAM care să îi permită invocarea unei funcții Lambda);
- Generarea SDK-ului Javascript necesar pentru a putea invoca API-ul 'Surf' din cadrul clientului web;
- Generarea unui fișier de configurare injectabil în clientul web 'Surf' pentru a stabili parametrii interacțiunii între client și cloud-ul AWS (e.g. regiunea AWS în care s-a generat infrastructura, metadate legate de rolurile utilizatorilor, cheia generată pentru a restricționa accesul asupra API-ului etc.).
- Generarea unui fișier de configurare injectabil în pachetul ce conține codul funcțiilor Lambda, în vederea efectuării setărilor legate de mecanismul asincron de notificare, numele bucket-urilor S3 create etc.

Efortul necesar pentru generarea infrastructurii este mare, datorită complexității inerente a proiectului. Efectuarea manuală a pașilor în interfața web AWS (engl. "AWS Web Console") necesită foarte mult timp și este predispusă la erori. De asemenea, asigurarea disponibilității crawler-ului în mai multe regiuni globale AWS, sau pe mai multe conturi AWS, ar necesita repetarea identică a pașilor enumerați mai sus, pentru fiecare astfel de regiune sau cont. De aceea, crawler-ul web 'Surf' pune la dispoziție o modalitate automatizată de deployment, configurabilă, testabilă și reutilizabilă, asigurând idempotență[22] la nivelul efectuării operațiilor în cadrul AWS.

Mecanismul automatizat de generare a infrastructurii AWS reprezintă un program care primește, ca date de intrare, un fisier de configurare în format JSON,

[22]Termenul *"idempotență"* este folosit pentru a evidenția faptul că o parte dintre operațiile efectuate de către mecanismul automatizat de generare a infrastructurii vor avea același rezultat dacă vor fi apelate de mai multe ori. Din motive de securitate, unele operații vor fi definite explicit ca nefiind idempotente (e.g. generarea de chei de acces pentru API-ul 'Surf', generarea adresei de acces a API-ului (nu va fi suprascrisă în API-ul existent))

generează resursele AWS și oferă clientului web, ca date de ieșire, prin injectarea dependențelor, informații și mecanisme pentru utilizarea infrastructurii create. Cerințele pentru execuția cu success a deployment-ului, precum și validitatea resurselor generate, reprezintă existența credențialelor AWS necesare pentru pașii de deployment (e.g. IAM 'administrator-access') și generarea pachetului în format *.jar*, care să conțină codul funcțiilor Lambda care se doresc a fi încărcate în AWS. Mai jos, se poate observa un exemplu de fișier de configurare pentru generatorul de resurse AWS 'Surf':

```
{
  "awsAccountId": "011759591962",
  "awsAccessKey": "#######", // Ascuns intenționat
  "awsClientRegion": "eu-west-1",
  "lambdaCodePath": "../lambda/target/surf-lambda-backend-1.0-SNAPSHOT.jar",
  "lambdaRuntime": "java8",
  "apiGatewayEndpoint": "apigateway.amazonaws.com",
  "apiStageName": "v1",
  "apiStageMetricsEnabled": true,
  "apiStageThrottlingRateLimit": "5",
  "apiStageThrottlingBurstLimit": "20",
  "apiLoggingLevel": "INFO",
  "apiGeneratedSdkType": "javascript",
  "apiGeneratedSdkOutputPath": "../../frontend/generated/sdks/",
  "apiGeneratedSdkFolderName": "api-gateway-js-sdk",
  "clientConfigFilePath": "../../frontend/generated/config/aws-config.json",
  "lambdaConfigFilePath": "../lambda/generated/config/lambda-config.json",
  "dynamoDBWorkflowsTableReadCapacityUnits": "2",
  "dynamoDBWorkflowsTableWriteCapacityUnits": "2"
}
```

Figura 4: Fișier parțial de configurare (intrare) pentru generatorul de resurse

Pentru a asigura conexiunea clientului web cu infrastructura creată, generatorul de resurse AWS injectează un fișier de configurare în clientul web, într-un director special destinat acestui sens. Politicile de securitate implementate în generatorul de resurse nu vor permite suprascrierea fișierului/directorului din clientul web în cazul în care acesta există deja, pentru a evita pierderea datelor. Un astfel de fișier de configurare este cel prezentat în *Figura 5*:

Diagrama din *Figura 6* prezintă procesul de construire a infrastructurii, ordonând temporal pașii necesari pentru crearea și configurarea crawler-ului. Înainte de începerea deployment-ului resurselor AWS, se generează o arhivă ce conține codul sursă al funcțiilor Lambda. Procesul începe cu citirea fișierului de configurare și se încheie cu generarea API-ului prin care utilizatorii vor putea

```
{
  "awsClientRegion": "eu-west-1",
  "awsAccessKey": "#######", // Ascuns intenționat
  "facebookWebIdentityBasicRoleArn": "arn:aws:iam::011759591962:role/fb-role",
  "apiKey": "#######" // Ascuns intenționat
}
```

Figura 5: Fișier folosit pentru configurarea clientului

accesa serviciul web. Acolo unde este necesar, se specifică interdependențele între sisteme. De exemplu, este necesar ca mai întâi să fie create funcțiile Lambda, înainte ca acestea să fie înregistrate în cadrul mecanismului de notificare asincronă oferit de SNS (de aici și numarul 4 asociat generării funcțiilor Lambda, reprezentând o prioritate mai mare decât numarul 5 asociat serviciului SNS). După finalizarea procesului de generare a infrastructurii, pașii 1 si 4 se vor rula încă odată, în contextul execuției precedente (i.e. având la dispoziție toate referințele către resursele create), pentru a actualiza codul funcțiilor Lambda în vederea accesării resurselor menționate.

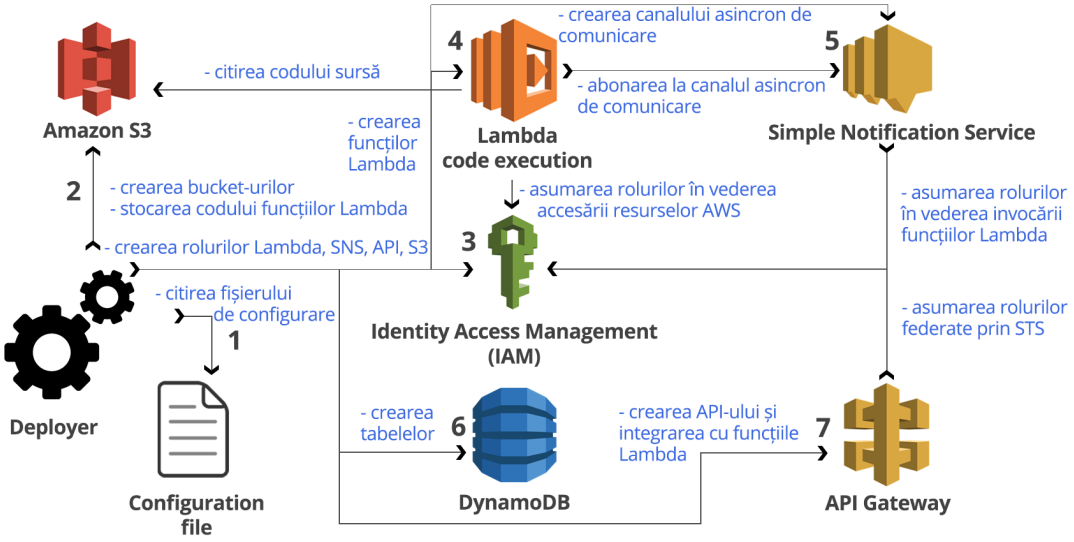


Figura 6: Generarea infrastructurii [12, 13]

3 Execuția procesului de crawling

Această secțiune prezintă, în detaliu, modul în care aplicația "Surf" îndeplinește procesul de parcurgere a siturilor web. Mai exact, se descriu mecanismele prin care, având la dispoziție infrastructura creată cu ajutorul generatorului de resurse, utilizatorul poate beneficia de rezultatele extrase din paginile web parcurse de catre crawler. Pentru simplitate, ne vom referi la ansamblul acestor mecanisme drept un *workflow*[23].

Un *workflow* reprezintă unitatea de bază pe care crawler-ul web "Surf" o poate interpreta. Dintr-un *workflow* se pot genera oricâte execuții. Începerea execuției unui *workflow* trebuie să fie precedată de crearea acestuia. O execuție a unui *workflow* întreprinde următoarele acțiuni, cu scopul parcurgerii paginilor web pentru extragerea informațiilor:

1. Preluarea, validarea și interpretarea datelor de configurare a procesului de crawling;

2. Stocarea persistentă a definiției unei execuții a unui *workflow* în baza de date, cu scopul accesării datelor necesare în contextul rulării crawler-ului;

3. Inițializarea unui automat finit și determinist, în vederea modelării parcurgerii recursive a siturilor web în manieră *breadth-first* [24], prin "AWS Step Functions";

4. Crearea sarcinilor pentru parcurgerea paginilor web (*taskuri*[25]), înregistrarea datelor și metadatelor rezultate în urma parcurgerii, precum și a datelor ce modelează execuția procesului de crawling (e.g. timpi maximi de răspus, adrese URL vizitate deja etc.);

5. Distribuirea *taskurilor* catre funcțiile Lambda (*workers*[26]) ce le vor prelua și executa;

[23]Din engl. "workflow", care se traduce, în acest context, drept un flux de lucru, sau un proces în cadrul căruia se desfășoară o serie bine definită de activități (i.e. execuții de programe)

[24]Parcurgea în manieră *breadth-first* asigură vizitarea fiecărui nod dintr-un arbore de la adâncimea curentă, înainte de a trece la parcurgerea nodurilor de la următorul nivel de adâncime

[25]În contextul execuției unui *workflow*, un *task* reprezintă o unitate de lucru, sub forma unei înregistrări în baza de date, ce poate fi preluată și executată de către un program aflat în execuție (e.g. o funcție Lambda)

[26]Un *worker* reprezintă un program executabil. În aplicatia "Surf", funcțiile Lambda reprezintă *workerii*

6. Executarea funcțiilor Lambda și stocarea/intoarcerea rezultatelor;

7. Sincronizarea execuțiilor paralele ale funcțiilor Lambda și centralizarea rezultatelor în urma execuției *task-urilor*;

8. Pornirea recursivă a unui nou *workflow*, cu scopul de a parcurge un nou nivel de adâncime în procesul de crawling și de a gestiona gradul de încărcare a sistemului distribuit (e.g. respectarea limitărilor impuse de serviciile AWS, precum numărul maxim de execuții concurente ale funcțiilor Lambda în cadrul execuției unui automat "AWS Step Functions").

3.1 Configurare

Pentru începerea execuției unui workflow, este necesară, în prealabil, crearea acestuia. *"Tabelul 1"* descrie datele necesare creării unui workflow, date dintre care o parte sunt configurabile de către utilizator (i.e. câmpul "Config."[27] are valoarea *DA*), iar o parte sunt memorate în mod implicit odată ce workflow-ul este creat. Suita de configurări descrisă în *"Tabelul 1"* coexistă cu seria de configurări generate de creatorul de resurse. Diferența între cele două constă în faptul că generatorul de resurse definește configurări la nivel de cont AWS. Împărțirea configurărilor în două nivele de abstractizare (i.e. scăzut pentru cele de la nivelul resurselor din contul AWS și ridicat pentru cele oferite în etapa de creare a unui workflow) conferă flexibilitate în utilizarea crawler-ului "Surf". Pentru a evidenția contrastul între cele două configurări, vom enumera, în continuare, câteva elemente esențiale ale configurării la nivel de resurse din contul AWS:

- timpul maxim de așteptare pentru ca o cerere cloud să fie satisfacută;
- regiunea în care sunt create resursele AWS;
- numărul maxim de cereri pe secundă admis de catre API;
- numărul maxim de cereri pe secundă admise pentru tabelele din baza de date (configurare importantă pentru controlul costului);
- stabilirea nivelului de jurnalizare (i.e. urmărire a acțiunilor utilizatorilor în cadrul interacțiunii cu API-ul "Surf");
- stabilirea limbajului în care va fi generat clientul pentru API-ul creat prin API Gateway.

[27] prescurtare de la substantivul *Configurabil*

Nume câmp	Tip	Config.	Descriere
Id	Text	NU	Id atribuit workflow-ului pentru a putea fi identificat în mod unic printre celelalte workflow-uri din tabela (din baza de date) care le găzduiește
Dată Creare	Numeric	NU	Data creării, reprezentând numărul de milisecunde de la 1970 (engl. "epoch milliseconds")
Proprietar	Text	NU	Identitatea atribuită utilizatorului ce a creat workflow-ul. Necesar pentru a stabili permisiuni de acces.
Nume	Text	DA	Nume familiar atribuit workflow-ului pentru identificare facilă
Adresă Început	Text	DA	URL-ul de la care va porni procesul de crawling
Selector URL	Text	DA	Expresie regulată ce validează dacă un URL de pe o pagină vizitată de către crawler este adăugat recursiv la lista sarcinilor pentru crawling
Adâncime maximă	Numeric	DA	Adâncimea maximă la care vor fi parcurse paginile web de către crawler (i.e. în arborele de parcurgere recursivă)
Pagini pe nivel	Numeric	DA	Numărul maxim de pagini ce vor fi parcurse de către crawler pe un anumit nivel de adâncime (i.e. în arborele de parcurgere recursivă)
Dimensiune maximă pagină	Numeric	DA	Dimensiunea maximă admisă (în octeți) pentru ca un sit web să fie parcurs
Grad de paralelizare	Numeric	DA	Numărul maxim de crawleri concurenți dintr-un workflow
Politica de selecție	JSON	DA	Definiție folosită pentru a extrage date din paginile web parcurse, în funcție de anumiți parametri
Politica de reîncercare	JSON	DA	Definiție folosită pentru a stabili cazurile în care încercările eșuate de a parcurge paginile web trebuie reîncercate

Tabelul 1: Definiția unui workflow

3.2 Inițializare

Odată ce o cerere de începere a unei execuții a unui workflow este primită de către crawler, funcția Lambda corespunzatoare începerii execuției (i.e. *"Pornire workflow"*) este invocată. Aceasta creează și salvează metadatele necesare în baza de date (e.g. timpul la care a inceput workflow-ul, cu scopul de a putea contoriza cât a durat execuția), apoi invocă funcția Lambda responsabilă pentru inițializarea mecanismului de distribuție și sincronizare a execuției în paralel a funcțiilor ce parcurg paginile web. *"Figura 7"* descrie modul în care funcția de inițializare a unui workflow este utilizată, precum și rolul pe care aceasta îl îndeplinește în procesul recursiv de crawling.

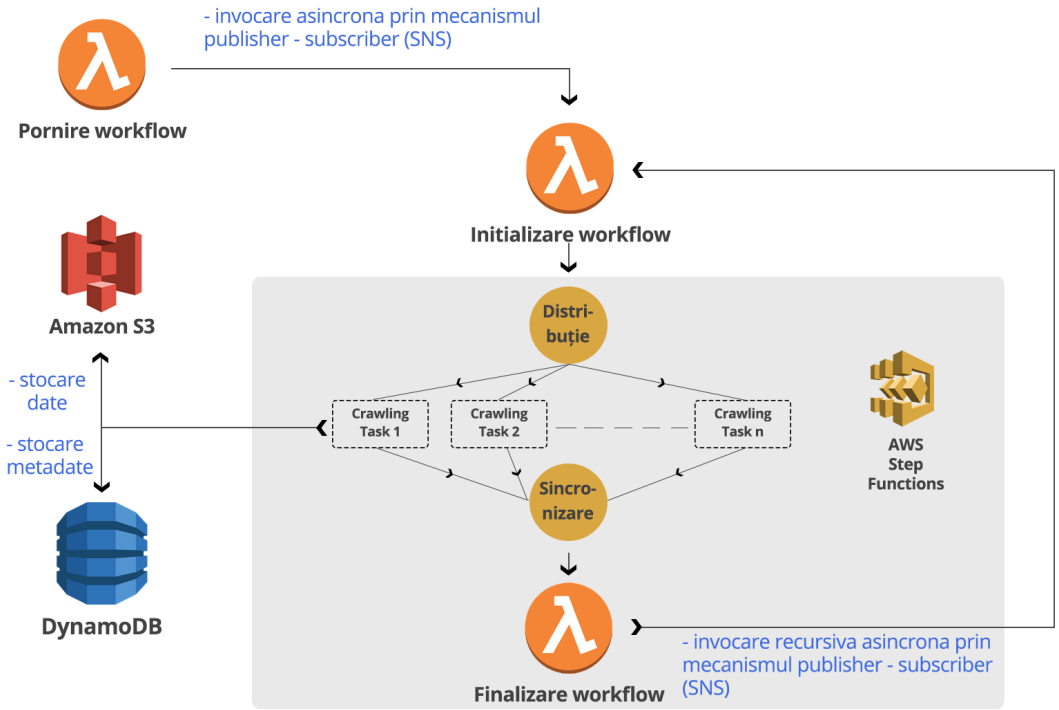


Figura 7: Arhitectura unui workflow [12, 13]

Responsabilitățile funcției de inițializare a workflow-ului sunt următoarele:

- Verificarea nivelului de adâncime la care s-a ajuns în urma procesului de crawling și finalizarea workflow-ului, în caz că nivelul de adâncime maxim a fost depășit;
- Interogarea bazei de date în vederea extragerii task-urilor pentru parcurgerea paginilor web, task-uri ce sunt executate în cadrul automatului orchestrat de către "AWS Step Functions";

- Crearea și pornirea automatului finit construit în cadrul "AWS Step Functions", pe baza task-urilor extrase din baza de date. Acest automat este definit prin două stagii secvențiale. Primul stagiu are ca scop orchestrarea distribuită a sarcinilor de crawling executate în paralel, iar cel de-al doilea are ca scop agregarea, interpretarea și salvarea datelor provenite din execuția task-urilor concurente.

3.3 Parcurgere

Funcțiile ce au ca scop vizitarea paginilor web sunt grupate, ca stagii de execuții paralele, în automatul finit configurat in "AWS Step Functions". Acestea reprezintă execuții ale definițiilor formale ale task-urilor de crawling. O parte dintre datele asociate unui task reprezintă metadatele configurate în pasul de creare a workflow-ului. *"Tabelul 2"* prezintă structura unui task, cu mențiunile necesare acolo unde definiția unui câmp este exact aceeași cu definiția câmpului respectiv din structura unui workflow.

Nume câmp	Tip	Descriere
Id	Text	Identificator unic pentru task
Id Workflow	Text	Identificatorul workflow-ului de care aparține task-ul curent
Dată Începere	Numeric	Data la care a inceput execuția task-ului (engl. epoch milliseconds)
Dată Finalizare	Numeric	Data la care s-a finalizat execuția task-ului(engl. epoch milliseconds)
Stare	Text	Starea în care se află task-ul: *Programat, Activ, Eșuat, Depășit temporal, Întrerupt*
Adresă	Text	URL-ul ce trebuie vizitat în vederea extragerii datelor și metadatelor
Adâncime	Numeric	Nivelul de adâncime, în arborele de parcurgere a paginilor web realizat de către crawler, la care se află task-ul
Erori	Listă	Eșecurile înregistrate în decursul execuției task-ului (ca structuri JSON)
Politică de selecție	JSON	*vezi Tabelul 1*
Dimensiune maximă pagină	Numeric	*vezi Tabelul 1*
Selector URL	Text	*vezi Tabelul 1*

Procesul de parcurgere a unei pagini web presupune următoarele etape:

1. Actualizarea stării task-ului în vederea reflectării condiției sale actuale (e.g. "în execuție" sau "eșuat");

2. Verificarea dreptului de acces asupra URL-ului configurat în cadrul definiției task-ului, prin consultarea fișierului robots.txt (dacă acesta există), aflat în calea de bază (engl. *root*) a sitului web de la URL-ul respectiv, în concordanță cu politica de configurare a procesului de crawling;

3. Parcurgerea sitului web:

   (a) Extragerea datelor conform politicii de selecție a datelor (configurată în pasul de creare a workflow-ului) și stocarea acestora (dacă există) in S3;

   (b) Salvarea metadatelor în legătură cu datale extrase, în vederea accesării sau căutării facile a locației în care au fost salvate datele în S3;

   (c) Selectarea tuturor legăturilor de tip URL către alte situri web din cadrul paginii vizitate, conform politicii de urmărire a legaturilor către alte pagini web, definită în pasul de creare a workflow-ului;

   (d) Crearea de noi task-uri în starea *"Programat"*, configurate cu un nivel de adâncime superior celui pe care se află URL-ul curent parcurs de către crawler și salvarea acestora în baza de date, doar dacă respectă configurările declarate ca date de intrare ale execuției workflow-ului (e.g. dimensiunea maximă a unei pagini web parcurse); acest pas mai poartă denumirea de *task scheduling*[28];

4. Actualizarea tabelei în care se ține evidența paginilor parcurse de către crawler în timpul execuției unui workflow, cu scopul de a nu se parcurge de două ori același URL;

5. Capturarea, rezolvarea și stocarea erorilor survenite în cadrul procesului de parcurgere a paginilor web.

3.4 Finalizare

Procesul de finalizare a unei sesiuni de crawling are drept obiective centralizarea și analizarea datelor provenite în urma executării, în paralel, a funcțiilor responsabile pentru parcurgerea paginilor web. Responsabilitățile funcției de

[28]engl. *task scheduling* = programare temporală a execuției unui task

finalizare se pot împărți în două categorii, în funcție de motivul ce a determinat invocarea acesteia:

1. Finalizarea unei execuții îndeplinite cu succes, caz în care nu au intervenit erori extraordinare (i.e. netratate) în cadrul cel puțin unuia dintre task-urile executate în paralel;

2. Finalizarea unei sesiuni de parcurgere în care au existat erori în cadrul execuției cel puțin unuia dintre task-urile executate în paralel, caz în care funcția de finalizare va îndeplini toate sarcinile ce au ramas neîndeplinite în urma terminării bruște a execuției sarcinilor paralele de crawling.

Indiferent de situația în care se află execuția worfklow-ului, funcția de finalizare a sesiunii de crawling are în vedere înregistrarea metadatelor cu privire la progresul workflow-ului (e.g. metrici Cloudwatch) și luarea deciziei conform căreia invocarea recursivă a următoarei sesiuni de crawling se face începand cu același nivel de adâncime sau cu unul superior (i.e. mai mare), în concordanță cu numarul maxim de pagini ce poate fi vizitat pe un anumit nivel de adâncime, de catre crawler.

4 Interfața de programare

Modalitatea prin care componentele crawler-ului web "Surf" pot fi accesate, modificate și executate este reprezentată de o interfață[14] structurată sub forma unui API RESTful[29]. API-ul adoptă stilul arhitectural REST[15] deoarece acesta ajută la decuplarea arhitecturii și simplificarea urmăririi executiei aplicației, prin faptul că nu permite memorarea stărilor intermediare în cadrul unei sesiuni de comunicare între utilizator ți crawler. De asemenea, API-ul RESTful conferă extensibilitate și flexibilitate la nivel de interfață, întrucât permite gruparea resurselor în mod ierarhic și identificarea ușoară a resurselor și a consecințelor aplicării anumitor acțiuni, prin verbe HTTP, asupra stării sistemului. *Figura 8* prezintă secvența mimimă de operații ce trebuie efectuate, la nivel de API, pentru a putea obține rezultate în urma procesului de parcurgere a paginilor web de catre crawler, modelate printr-o diagramă de secvență.

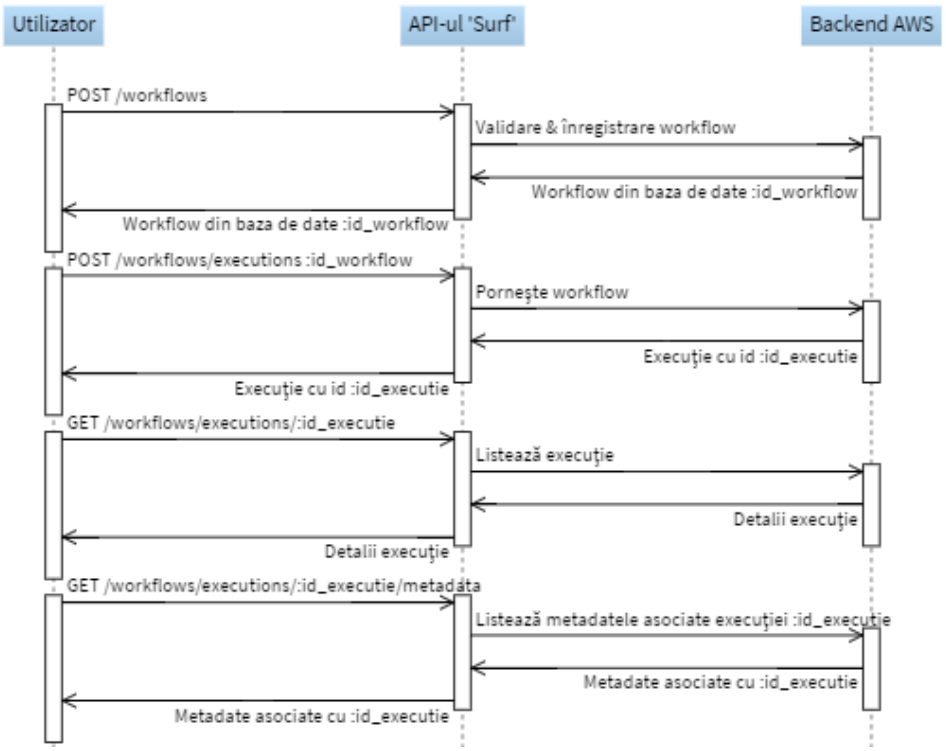


Figura 8: Interacțiunea utilizator - API [12, 13]

API-ul construit este securizat suplimentar, în afara drepturilor de acces sta-

[29]engl. *RESTful* = adjectiv de la abrevierea *REST*; un API este *RESTful* dacă adoptă stilul arhitectural *REST[15]*

bilite prin rolurile IAM, prin necesitatea deținerii unei chei de acces de către utilizatorul acestuia. Cheile de acces se creează programatic, de către generatorul de resurse, și au, fiecare, asociate, politici de restricție a utilizării API-ului. Politicile de restricție constau în limitări impuse de către sistemul cloud asupra ratei la care se accesează API-ul, folosind o anumită cheie de acces. Există doua categorii de limitări în ceea ce privește controlul traficului autentificat asupra API-ului "Surf", conform algoritmului *token bucket*:

- *Limitarea ratei de acces regulat:* se referă la controlarea numărului maxim de cereri pe secundă pe care un utilizator autentificat le poate face în mod normal;
- *Limitarea ratei de acces în cascadă*[30]*:* se referă la controlarea cantității extraordinare de cereri care pot surveni ca răspuns la un anumit eveniment extern (e.g. mai multe workflow-uri sunt pornite la o anumită oră din zi).

5 Performanță și scalabilitate

Prin natura distribuită a crawler-ului "Surf", acesta asigură viteză în parcurgerea paginilor web şi un grad crescut de redundanţă în vederea extragerii informaţiilor din sursele selectate. Scalabilitatea orizontală[31] şi verticală[32] este garantată de mecanismele de scalare ale serviciilor web din cadrul Amazon Web Services.

Fie că este vorba despre API-ul "Surf", mecanismul asincron de notificări (SNS), asigurarea spaţiului de stocare (S3) sau funcţiile Lambda (centrul computaţional al crawler-ului "Surf"), efortul utilizatorului pentru a ajusta performanţa sistemelor menţionate constă doar în a ajusta configurările corespunzătoare serviciului care se doreşte a fi scalat[33]. În consecinţă, costurile vor varia în funcţie de gradul de performanţă care se doreşte a fi atins. Graficele de mai jos (*"Figura 9"* şi *"Figura 10"*) prezintă comparaţii relative între o serie variată de configurări ale sistemului cloud ce găzduieşte aplicaţia "Surf". *"Figura 9"* are în vedere configuraţia *"adâncime maximă: 5, număr maxim de pagini pe nivel: 30, timp de terminare sarcini maxim pentru workeri: 40 secunde, adresă de pornire:*

[30] engl. *burst limit*

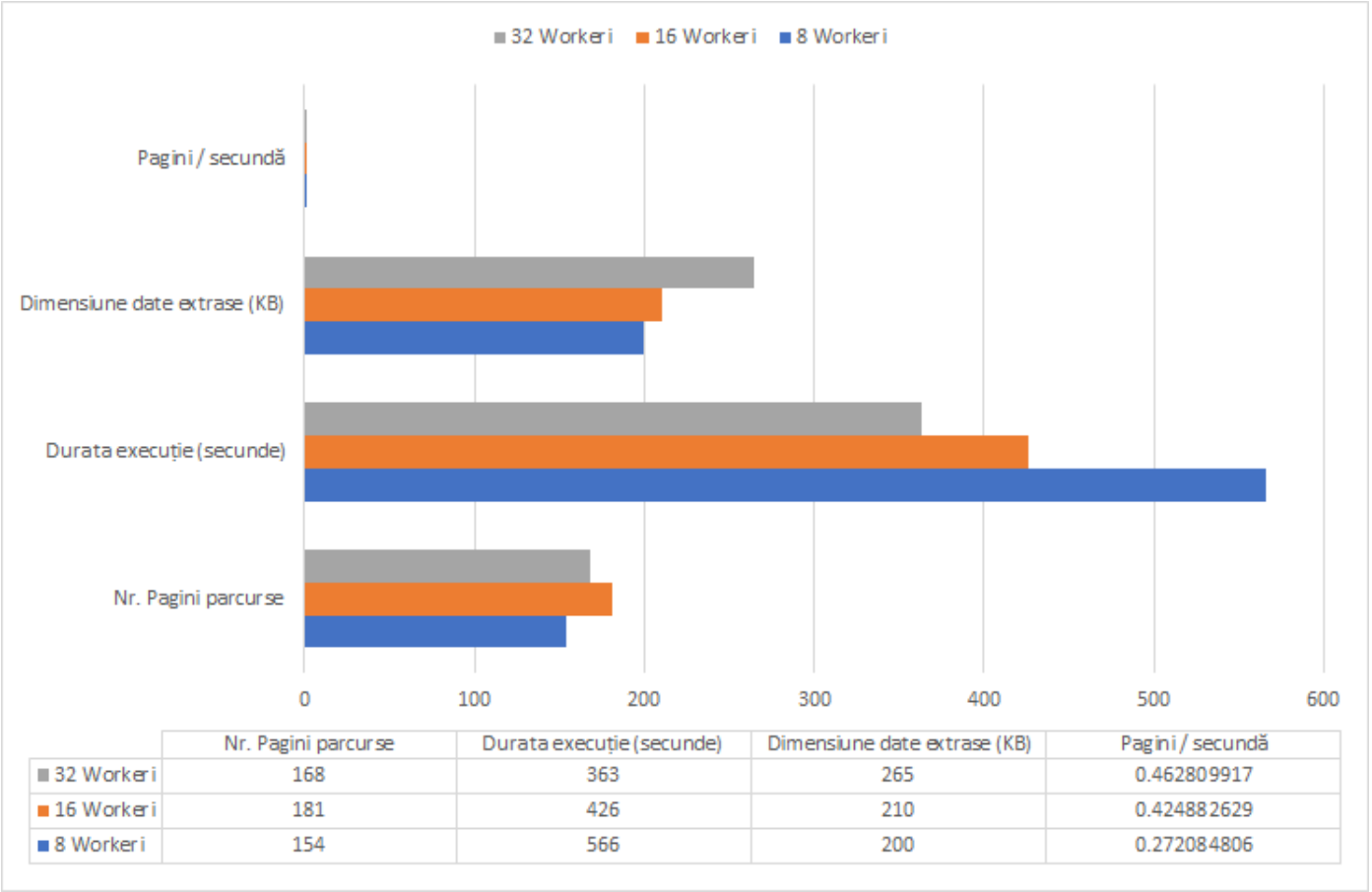
[31] *"scalabilitatea orizontală"* reprezintă posibilitatea de a adăuga mai multe maşini de lucru unui sistem informatic, pentru a-i mări performanţa

[32] *"scalabilitatea verticală"* se referă la posibilitatea de a îmbunătăţi performanţa maşinilor de calcul dintr-un sistem informatic, prin adăugarea de hardware mai performant

[33] un serviciu web poate fi scalat atât în sus, adăugând putere computaţională, cât şi în jos, renunţând la putere de calcul

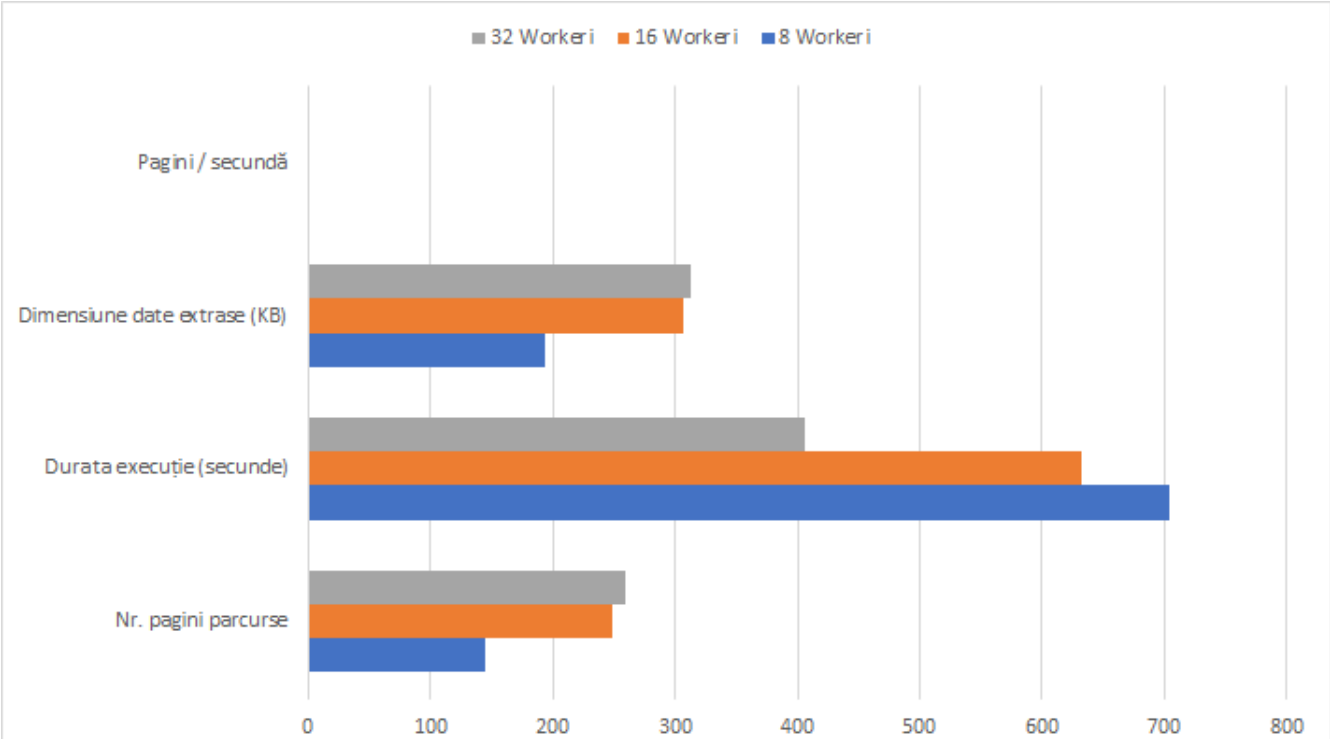
http://news.ycombinator.com/news". "Figura 10" este diferită prin faptul că prezintă configuraţia *"adâncime maximă: 3, număr maxim de pagini pe nivel: 100".*

Din diagramele din figurile 9 şi 10 se poate observa faptul că gradul de paralelizare are un impact direct asupra performanţei crawler-ului. Testele s-au realizat atunci când sistemul nu era utilizat din alte motive decât cele în scopul determinării performanţei crawler-ului. Deşi este de aşteptat ca performanţa să se degradeze odată ce mai mulţi utilizatori încep să folosească serviciul de crawling "Surf", pierderea de performanţă se poate compensa crescând cantitatea de resurse alocate pentru fiecare funcţie Lambda (i.e. procesor şi memorie) şi prin configurarea tabelelor DynamoDB în vederea acceptării unui număr mai mare de cereri pe secundă.



	Nr. Pagini parcurse	Durata execuție (secunde)	Dimensiune date extrase (KB)	Pagini / secundă
32 Workeri	168	363	265	0.462809917
16 Workeri	181	426	210	0.424882629
8 Workeri	154	566	200	0.272084806

Figura 9: Performanța crawler-ului "Surf" la parcurgerile în adâncime [12, 13, 17]



	Nr. pagini parcurse	Durata execuție (secunde)	Dimensiune date extrase (KB)	Pagini / secundă
32 Workeri	259	406	312	0.637931034
16 Workeri	249	632	306	0.393987342
8 Workeri	145	704	193	0.205965909

Figura 10: Performanța crawler-ului ”Surf” la parcurgerile pe niveluri [12, 13, 17]

6 Extensibiliate

Pe lângă facilitățile de extensibilitate oferite de utilizarea serviciilor cloud din cadrul "Amazon Web Services", arhitectura aplicației "Surf" prezintă următoarele puncte de extensie:

- Adăugare rapidă și ușoară de noi servicii cloud în cluster-ul aplicației, datorită structurii decuplate și configurabile a mecanismului de generare a infrastructurii;
- Asigurarea infrastructurii crawler-ului în mai multe regiuni globale, în conformitate cu politicile de globalizare AWS, astfel încât aplicația "Surf" să poată fi folosită la scară largă;
- Posibilitatea de a adăuga funcții Lambda de tip plug-in, alături de mecanismului de parcurgere și procesare a informațiilor din cadrul paginilor web vizitate de către crawler (*"Figura 11"*).

7 Direcții viitoare

Această secțiune prezintă, pe scurt, câteva dintre componentele proiectate pentru crawler-ul "Surf" care ar aduce avantaje reale în ceea ce privește flexibilitatea în utilizare.

7.1 Sistemul de plugin-uri

Sistemul de pluginuri (prezentat, la nivel abstract, în *"Figura 9"*), reprezintă o modalitate de a adăuga crawler-ului web "Surf" funcționalități legate de procesarea informațiilor preluate din siturile web vizitate. Plugin-urile vor fi dezvoltate ca funcții Lambda, care se vor afla în contul AWS al proprietarului crawler-ului. Fiecare plugin va adera la o interfață, cu scopul realizării schimbului de informații în cadrul automatului finit definit prin "AWS Step Functions". Totodată, fiecare plugin va specifica dacă în cadrul execuției sale vor fi salvate date și metadate în sistemele persistente de stocare. Sistemul va putea genera rezultate ale parcurgerii informațiilor din paginile web vizitate la oricare pas al execuției paralele din cadrul automatului finit, datele fiind etichetate cu prefixul funcției care le-a generat, pentru a facilita selectarea și agregarea acestora. Cu alte cuvinte, fiecare task paralel din cadrul automatului finit va trece printr-o serie de stagii configurabile în ceea ce priveste extragerea și prelucrarea informațiilor provenite de la stagiul precedent. În cazul în care nu

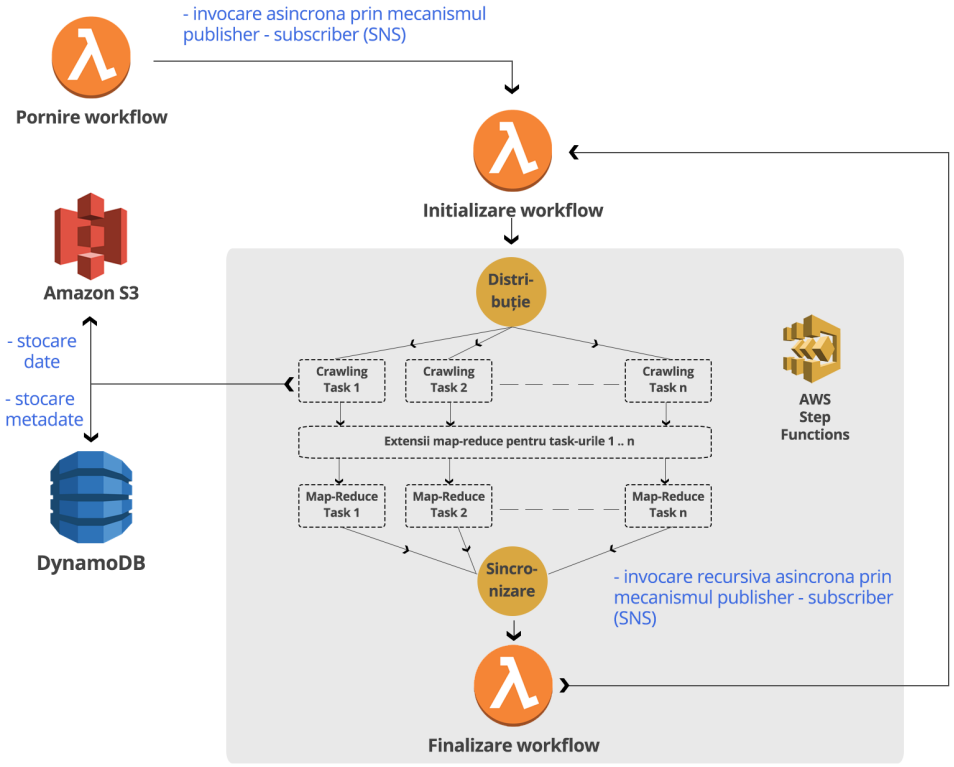


Figura 11: Plugin-uri adăugate procesului de crawling [12, 13]

se dorește specificarea de plugin-uri, atunci execuția taskurilor de crawling din automatul finit va avea comportamentul de bază descris în capitolul *"Crawling în cloud"*.

7.2 Indexare periodică

Odată ce crawler-ul web "Surf" extrage datele din paginile web vizitate, acesta creează și o serie de metadate, cu scopul indexării locației datelor în S3. Deși acest sistem funcționează, el este limitat la accesarea datelor despre localizarea informațiilor salvate. O îmbunătățire a funcționalității aplicației "Surf" ar reprezenta crearea unui sistem de indexare periodică a informațiilor. Un astfel de sistem ar asigura mecanisme complexe de localizare și extragere a datelor, atât în funcție de sintaxă, cât și de semantică, îndeplinind următoarele funcții:

- indexare periodică, prin mecanismul de generare de evenimente oferit de "AWS CloudWatch Events", a datelor generate în procesul de parcurgere a siturilor web;

- Utilizarea unui framework pentru procesarea limbajului natural
- Utilizarea funcțiilor de procesare a web-ului semantic pentru a extrage, pe baza unui algoritm de clusterizare, structuri semantice definitorii din paginile web vizitate[34].

Odată grupate și generate, datele ce alcătuiesc mecanismul de indexare pot fi clasificate utilizând algoritmi probabiliști de clasificare, pentru a restrânge spațiul de căutare atunci când se recurge la cuvinte cheie.

7.3 Sistemul de notificări

În funcție de preferințele utilizatorilor, aplicația "Surf" poate fi dezvoltată în direcția configurării execuțiilor periodice pe bază de evenimente generate în cloud (e.g. expresii CRON[35] în cadrul CloudWatch Events). În acest caz, este necesar un sistem de notificări asincrone care să îi înștiințeze pe utilizatori despre progresul pe care crawler-ul îl face în decursul execuției unui workflow. Flexibilitatea sistemului cloud din cadrul "Amazon Web Services" permite utilizarea de mecanisme multiple pentru a transmite notificări precum:

- *Simple Notification Service:* permite transmiterea de notificări prin mecanismul publisher-subscriber[36]; notificările pot fi configurate pentru o gamă largă de destinații, de la adrese de e-mail, până la notificări *push* adresate sistemelor de operare mobile;
- *Simple Email Service:* permite trimiterea de e-mailuri către utilizatori; e-mailurile pot fi configurate astfel încât să accepte limbaj de markup avansat, redând o experiență mai bogată utilizatorilor.

[34]http://schema.org/
[35]https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm
[36]https://docs.oracle.com/cd/B10501_01/appdev.920/a96590/adg15pub.htm

Contribuții personale

Crawler-ul web "Surf" prezintă următoarele caracteristici cheie în ceea ce priveşte dezvoltarea, rularea şi mentenanţa unui crawler web distribuit:

- Are capacitatea de a-şi genera fiecare resursă necesară, utilizând programul definit ca *"Generator de resurse"*. Această funcţionalitate permite crawler-ului să fie construit în cadrul oricărui cont "Amazon Web Services", într-o regiune globală pe care utilizatorul o poate configura. Totodată, se permite administratorului financiar al contului AWS să aibă un control fin asupra costurilor generate de către crawler, deoarece limitele rezervate serviciilor web folosite de către crawler pot fi configurate în mod individual;

- Permite integrarea cu un sistem de autentificare sigur, bazat pe federearea identității web în raport cu un furnizor terţ de identităţi de încredere. În urma autentificării, crawler-ul web "Surf" asigură autorizarea cererilor într-o manieră flexibilă, bazată pe documente ce definesc politici de securitate. Odată autorizaţi, utilizatorilor le sunt alocate chei de acces asupra API-ului, care stabilesc limite configurabile asupra numărului de cereri pe secundă pe care aceştia le pot efectua;

- Oferă scalabilitate orizontală şi verticală, atât în sus cât şi în jos, în ceea ce priveşte mecanismul de parcurgere concurentă a paginilor web, prin stabilirea, în cadrul fiecărui workflow, a unui grad de paralelizare menit să se adapteze cerinţelor de timp şi cost ale fiecărui utilizator;

- Poate fi distribuit drept un executabil care să creeze, având definit, în prealabil, un fişier de configurare, într-un interval de aproximativ două minute, toată infrastructura AWS necesară utilizării aplicaţiei cu scopul de a parcurge paginile web pentru a extrage informaţii;

- Oferă, pentru dezvoltatori, o interfaţă de programare flexibilă şi prietenoasă, ce respectă stilul arhitectural REST; De asemenea, permite dezvoltatorilor posibilitatea de a integra alte servicii cloud din cadrul AWS pentru a extinde funcţionalităţile crawler-ului.

Concluzii

Datorită ritmului alert în care se creează sau schimbă informaţiile aparţinând world wide web-ului, există o necesitate tot mai mare de a putea prelua şi analiza datele într-un mod automat, rapid şi eficient. Crawlerii web sunt o componentă deosebit de importantă, în vederea extragerii şi indexării exhaustive a informaţiilor, pentru motoarele de căutare. Cu toate acestea, avantajele aduse de către crawleri pot fi folosite şi pentru a extrage resurse relevante dintr-un anumit context, fără a fi necesară o parcurgere a tuturor resurselor web accesibile. Odată cu apariţia şi dezvoltarea serviciilor web, infrastructura puternică şi costurile reduse oferite de către furnizorii cloud oferă modalităţi flexibile, intuitive şi extensibile de a susţine crawleri specializaţi.

Crawler-ul web "Surf" propune, în acest sens, o modalitate simplă pentru ca un utilizator interesat de resurse dintr-o anumită arie să poată să le preia automat şi să le analizeze conţinutul, fără a fi necesar să depindă de furnizori terţi de servicii web de crawling şi cu posibilitatea de ajustare minuţioasă a costurilor operaţionale. Paralelizarea sarcinilor de parcurgere a paginilor web, împreună cu scalabilitatea orizontală şi verticală asigurată de servciile cloud, garantează adaptabilitatea aplicaţiei "Surf" la cele mai diverse nevoi ale utilizatorilor săi. Flexibilitatea legată atât de costuri, cât şi de design-ul arhitectural al aplicaţiei "Surf" alături de API-ul RESTful, care expune funcţionalităţile într-un mod accesibil şi extensibil, permite dezvoltatorilor interesaţi să adauge elemente suplimentare, precum: componente de tip map/reduce, analiză semantică a conţinutului paginilor web şi tehnici euristice pentru selectarea frontierei de URL-uri. De asemenea, flexibilitatea serviciilor cloud garantează atingerea unui nivel de activitate de aproape 100%, oferind siguranţă şi încredere în utilizare.

Bibliografie

[1] *"Visual Networking Index"*, Cisco Systems, http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-net

[2] http://www.internetlivestats.com/total-number-of-websites/

[3] Vladislav Shkapenyuk and Torsten Suel. *Design and Implementation of a High-Performance Distributed Web Crawler.* http://cis.poly.edu/tr/tr-cis-2001-03.pdf

[4] Yugandhara Patil and Sonal Patil. *Review of Web Crawlers with Specification and Working.* http://www.ijarcce.com/upload/2016/january-16/IJARCCE%2052.pdf

[5] Pant Gautam, Srinivasan Padmini and Menczer Filippo. *Crawling the Web.* https://pdfs.semanticscholar.org/a4f4/5f3a3d7d53a40b7b2579392a5db88cc1b822.pd

[6] Pant Gautam, Srinivasan Padmini and Menczer Filippo. *Exploration versus exploitation in topic driven crawlers.* $https://www.researchgate.net/publication/2946755_Exploration_versus_Exploitation_in_Topic$

[7] S. Ravi Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. *Stochastic models for the Web graph.* http://cs.brown.edu/research/webagent/focs-2000.pdf

[8] Martijn Koster. *A Standard for Robot Exclusion.* http://www.robotstxt.org/orig.html

[9] http://www.cellopoint.com/media_resources/blogs/2011/03/Web_Crawlers

[10] http://cis.poly.edu/tr/tr-cis-2001-03.pdf

[11] Miller, Ron (24 Nov 2015). "AWS Lambda Makes Serverless Applications A Reality". TechCrunch. Retrieved 10 July 2016.

[12] Sursele pictogramelor includ https://www.iconfinder.com si http://simpleicon.com.

[13] Sursa pictogramelor asociate "Amazon Web Services": https://aws.amazon.com/architecture/icons/

[14] Definiția cuvântului interfață a fost adaptată pornind de la definiția de la adresa https://www.merriam-webster.com/dictionary/interface

[15] http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm

[16] Diagrama de secvență a fost generată folosind API-ul aflat la adresa http://gojs.net/latest/samples/sequenceDiagram.html

[17] Diagramele au fost realizate utilizând programul "Microsoft Excel" (https://products.office.com/ro-ro/excel)

[18] https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

[19] https://www.ietf.org/rfc/rfc793.txt

[20] https://github.com/BruceDone/awesome-crawler

[21] Brin, S. and Page, L. (1998) The Anatomy of a Large-Scale Hypertextual Web Search Engine, http://ilpubs.stanford.edu:8090/361/

[22] Notes on Theory of Distributed Systems, CPSC 465/565: Fall 2017, James Aspnes, http://www.cs.yale.edu/homes/aspnes/classes/465/notes.pdf

[23] *The Web Application Hacker's Handbook, 2nd Edition, Dafydd Stuttard and Marcus Pinto*