

М. М. Климаш, Нажм Ахмад Байдун, Р. В. Капустяк, І. В. Демидов, М. І. Бешлей
Національний університет “Львівська політехніка”

СТВОРЕННЯ ЕФЕКТИВНИХ ІКТ-ПЛАТФОРМ ЕЛЕКТРОННОГО УРЯДУВАННЯ ІНТЕРАКТИВНОГО ТИПУ: АНАЛІЗ АРХІТЕКТУРИ СИСТЕМ ЗВОРОТНОГО ЗВ'ЯЗКУ

© Климаш М. М., Нажм Ахмад Байдун, Капустяк Р. В., Демидов І. В., Бешлей М. І., 2020

Розглянуто архітектурні особливості побудови систем цільового поширення електронного контенту для побудови інтерактивних платформ електронного урядування у частині систем зворотного зв'язку на основі найхарактерніших та найефективніших архітектур вебкраулерів, а також практичного досвіду компанії Google. Проаналізовано основні особливості реалізування вебкраулерів як систем “глибокого” пошуку й основи для наскрізного моніторингу в державному інтернет-просторі. Подано архітектуру мережевої платформи для фіксації та оброблення подій у реальному часі на основі технологій корпорації Google, що може бути основою для швидкого та ресурсоефективного розгортання систем цільової доставки заданого електронного контенту під час реалізації вибраної стратегії електронного урядування, із наданням відповідній платформі інтерактивних якостей за рахунок реалізації каналів зворотного зв'язку із суспільством.

Ключові слова: ІКТ; електронне урядування; системи контролю інформаційного простору; архітектура систем зворотного зв'язку.

M. M. Klymash, Najm Ahmad Baydoun, R. V. Kapustiak, I. V. Demydov, M. I. Beshley
Lviv Polytechnic National University

DEVELOPMENT AN EFFECTIVE INTERACTIVE E-GOVERNMENT ICT PLATFORMS: ANALYZING THE ARCHITECTURE OF FEEDBACK SYSTEMS

© Klymash M. M., Najm Ahmad Baydoun, Kapustiak R. V., Demydov I. V., Beshley M. I., 2020

This work examines the architectural features of development an e-content target distribution system for building interactive e-government platforms on feedback systems part based on the most specific and effective web crawler architectures, as well as Google's practical experience in this field and respective cloud solutions. The main features of implementation of web crawlers are analyzed, as systems of “deep” web searching and basis for cross-monitoring in the state Internet space in the context of software architecture, as well as the difficulties of their integration in the global information and communication space have been given, and outlines for the most expedient ways to overcome them have been drawn. The architecture of a network platform for real-time capturing and processing of events based on Google technologies is presented, which can be the basis for rapid and resource-efficient deployment of targeted delivery of specified electronic content while implementing the chosen e-governance strategy, providing the implemented platform with interactive qualities at the expense of implementation of feedback channels with the society, which are equipped with means of fixing the responses of representatives of the target audience to information messages and the necessary analytical tools (such as Google's BigQuery and so on). Similar effective (including cloud-based) solutions can also be implemented through social networking platforms such as Facebook, Twitter, Viber digital messengers, Telegram and more. The specific way of

implementing such solutions will obviously depend on the chosen state policy and commercial feasibility in the aspect of public-private partnership.

For the purposes described above, the so-called Deep Web Searching Engines are suitable, the architecture of which differs significantly from the traditional search engines we are used to in our daily lives. The challenges faced by deep web search systems include connecting to deep (by hierarchy of Internet positioning) web resources, processing and sampling the data they host and are most relevant to the objectives of the task, postulated by developed state informational policy. For example, targeted search for information in many profiles of a group on a social network, in a content (topical) community in a certain forum, etc. The specifics of their application require considerable speed, as it is usually intended to create or supplement large volumes of real-time data to further analyze, track changes and trends.

In the future studies based on this work, the authors believe that it is advisable to test the hypothesis of the potential for influencing the intensity of the audience response by changing the intensity of managerial influences using the distribution of some set of identified key messages, which in turn requires the study of the impact indicator to the intensity of audience response on external factors during real-time discussions, the results of which can obviously be extended to similar discussion processes in the Internet space. The relevant analysis could be performed on an open source basis.

Key words: ICT; e-government; informational space control systems;, feedback systems architecture.

Вступ

Доволі актуальним напрямом з погляду комерціалізації наукових досліджень сьогодні є розвиток платформ поширення цифрового контенту. Вони ж, своєю чергою, повинні стати одним із ключових сервісів сучасних систем електронного урядування. Проте створення інтерактивних платформ електронного урядування має на меті не лише ефективніше донесення потрібної інформації (політичної, економічної, соціальної) до визначених представників цільових аудиторій, але і реагування на їхні настрої (позитивні або негативні), обсяги (інтенсивність) обговорень у відповідних соціальних мережах, результати голосувань на форумах тощо. Особливості розгортання та архітектури систем розповсюдження контенту розглянуто у роботах [1–3, 8]. Однак сьогодні особливий інтерес викликають методи контролю ефективності доставки електронного контенту, а також відстежування настроїв певних соціальних груп, їхніх реакцій на ті чи інші привнесені інформаційні повідомлення як фактори впливу. Реалізація таких методів неможлива без застосування спеціалізованих інформаційних і комунікаційних систем із особливою архітектурою.

Зокрема, для таких цілей придатні так звані системи глибокого вебпошуку (Deep Web Searching Engines), архітектура яких істотно відрізняється від традиційних пошукових машин, які ми звикли використовувати у повсякденному житті. Завдання, з якими стикаються системи глибокого вебпошуку, передбачають встановлення підключення до глибоких (за ієрархією розміщення у мережі інтернет) вебресурсів, оброблення і здійснення вибірок даних, які на них розміщені та є найрелевантнішими щодо цілей поставленого завдання. Наприклад – цільовий пошук інформації у множині профілів групи в соціальній мережі, у тематичній спільноті на деякому форумі тощо. Специфіка їх застосування потребує значної швидкодії, оскільки зазвичай передбачається створення або доповнення великих масивів даних у реальному масштабі часу з метою їх подальшого аналізу, відстежування змін та тенденцій.

Метою цієї роботи є аналіз архітектури та досвіду побудови систем зворотного зв'язку для інтерактивних платформ поширення комерційного електронного контенту. Відповідно, автори побудували цю публікацію з двох частин. Перша частина висвітлює архітектурні аспекти побудови так званих “вебпавуків” як представників спеціалізованих метапошукових систем “глибокого” пошуку, які, очевидно, зможуть максимально ефективно функціонувати в сучасному інтернет-

просторі відповідно до тематики цього дослідження. Друга частина статті стисло висвітлює результати аналізу архітектури системи фіксації та оброблення подій реального часу в мережах поширення цифрового контенту, яку напроцювала корпорація Google протягом останніх років.

Архітектурні підходи до створення метапошукових систем реального часу

У роботі [4] виділено два найперспективніші підходи до створення метапошукових систем. Першим із них є *Turbo10* – архітектура комерційної пошукової системи, призначена для доступу до “глибоких” вебресурсів з метою надійного отримання результатів пошуку [5]. Ідея такої архітектури полягає у тому, що в інтернеті є величезна кількість варіацій (у програмному розумінні) інтерфейсів для здійснення запитів, що стосуються конкретних тем або груп ресурсів. Кожен із цих інтерфейсів допомагає створити власні результати пошуку за запитом. Метапошукове ядро (в основі якого розташований так званий траулер-сервер) по суті являє собою систему реалізації запиту із отриманням інтерфейсу для здійснення власне пошукових процесів, за допомогою серії звертань до інтернет-ресурсів через відповідний менеджер (рис. 1). Архітектура типу Turbo10 передбачає застосування такого ядра, яке також виконує ранжирування релевантності результатів пошуку, кластеризацію тематики та об’єднання результатів пошуку, причому безпосередньо у веббраузері клієнта (або його еквіваленті), а не на серверній частині. Оскільки можуть виникнути ускладнення, коли деякі повільніші, “глибші” за ієрархією вебресурси не надаватимуть відповідь у межах потрібного часового інтервалу, ядро Turbo10 (компонент траулер-сервер) починає збирати результати пошуку до браузера (клієнтського вебкомпонента) асинхронно, після того, як перший “глибокий” вебресурс відповів та від нього одержано відповідні пошукові результати [6]. Такий підхід знижує ймовірність того, що клієнт (або й уся відповідна підсистема) чекатиме моменту, коли який-небудь із багатьох запитаних “глибоких” вебресурсів відповість, тоді як від усіх інших вже отримано результати (рис. 1).

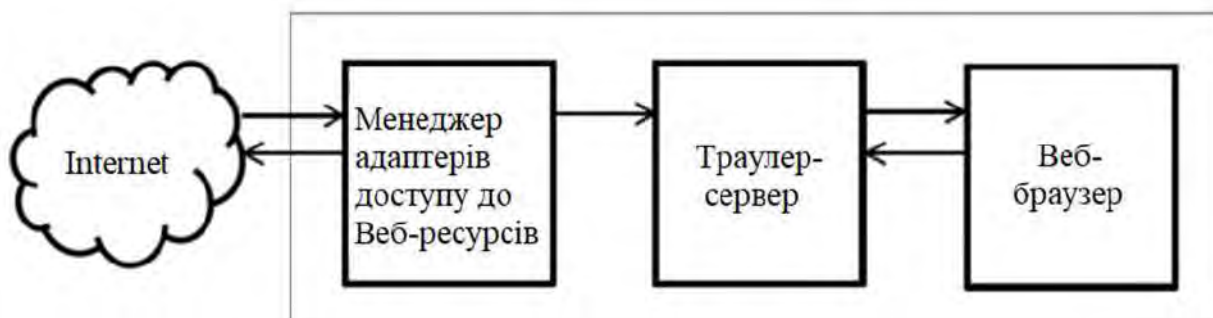


Рис. 1. Структура метапошукової машини архітектури Turbo10

Структурно ядро Turbo10 можна розділити на три частини: менеджер адаптерів доступу до вебресурсів, траулер-сервер та веббраузер. Менеджер адаптерів (рис. 2) відповідає за збереження доступу до “глибоких” вебресурсів. Застосовуючи таку архітектуру пошукової машини, розробникам потрібно надати змогу попередньо вручну задати URL-адреси ресурсів, які містять інтерфейси запитів до своїх баз даних, та зберегти можливість використати ці адреси надалі [6]. Тому описувана система передбачає використання техніки ручного введення для пошуку (і підготовчого формування) інтерфейсів запитів під час її попереднього налаштування. Отже, пошуковий механізм Turbo10 не намагатиметься самостійно сканувати більше ресурсів, покладаючись на те, що цільові ресурси (їх групи) для сканування йому задають вручну. По суті, це є деяким евристичним рішенням для уникнення перевантажень шлюзової мережевої інфраструктури, для якої “глибокий” пошук є аналогом DoS-атаки, що насправді є небажаним явищем, з яким борються численні системи виявлення мережових утручань. У менеджері адаптерів передбачено компонент, який оброблятиме надану URL-адресу для інтерфейсів запитів (за це відповідає компонент пошуку форм на вебресурсах). Після того, як інтерфейси запитів визначені, менеджер адаптерів переходить до ідентифікації параметрів, необхідних для використання пошукової системи за кожним інтерфейсом

запиту. Це приводить до реалізації архітектури метапошукової системи, яка забезпечує процес навігації через набір проміжних сторінок із потенційно корисними для користувача результатами (між інтерфейсом запитів і сторінкою результатів) (див. рис. 2).

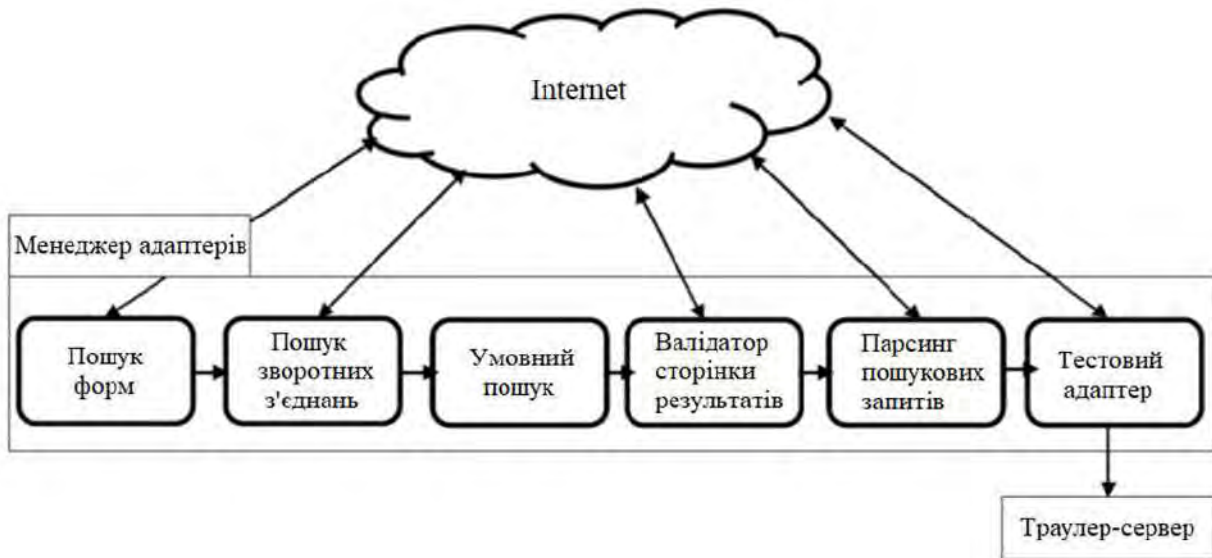


Рис. 2. Архітектура менеджера адаптерів у метапошуковій машині типу Turbo10

Компонент, що здійснює пошук зворотних посилань, отримує список із 50 вебсторінок, знайдених за контекстом, який збігається із зазначеним у формі цільового пошуку пошукової машини. Після цього для кожного зворотного посилання окремо одержують контекст за допомогою структурних міток, знайдених у тексті відповідних вебсторінок. Потім усі ці контексти передають до блока умовного пошуку, який містить умови, що визначаються конфігурацією пошукової системи. Відмінність між знайденими результатами вимірюється за допомогою попередньо обчислених значень зворотної частоти документа (IDF) щодо інвертованих значень коефіцієнтів частотності елементів його вмісту (таку структуру даних для індексування вмісту застосовують, щоб забезпечити можливість швидкого повнотекстового пошуку), яку одержують, наприклад, відповідно до бази даних Open Directory Project [7] – найбільшого, найповнішого редакторського тематичного каталогу в інтернеті. Мета цього проєкту – створення організованої тематичної класифікації для інтернету, оскільки його масштаби продовжують невпинно зростати.

Далі процес пошуку переходить до стадії надсилання тестового запиту в “глибокий” веб-ресурс. У разі успішного тестування інформація передається до валідатора сторінки результатів пошуку для того, щоб перевірити, чи ресурс справді повернув коректну сторінку результатів пошуку. Цей модуль містить програмний код операцій, які дають змогу із високим ступенем імовірності проаналізувати посилання на одержаній сторінці результатів пошуку. Модуль парсингу пошукових запитів, своєю чергою, відшукує на сторінці результатів пошуку структурну інформацію, яка відповідає списку результатів пошуку та номінує потрібні посилання. Цей модуль визначає потрібний шаблон інтерфейсу запиту на основі аналізу деякої кількості номінованих посилань, що також дає змогу перевірити на схожість різні сторінки результатів пошуку, одержані з використанням тієї самої пошукової машини [6].

Ядро траулер-сервера поповнює колекцію інтерфейсних адаптерів запитів, які надходять у розпорядження менеджера адаптерів, тобто успішно перевірені адаптери переміщуються у пул адаптерів траулер-сервера. Коли через веббраузер вводять пошуковий запит, він передається від диспетчера браузера до траулер-сервера (рис. 3). Диспетчер браузера комунікує із траулер-сервером, який залучає необхідні адаптери з пулу адаптерів інтерфейсів запитів. Після цього диспетчер браузера повертає результати у веббраузер. Взаємодію траулер-сервера і диспетчера веббраузера проілюстровано на рис. 3.

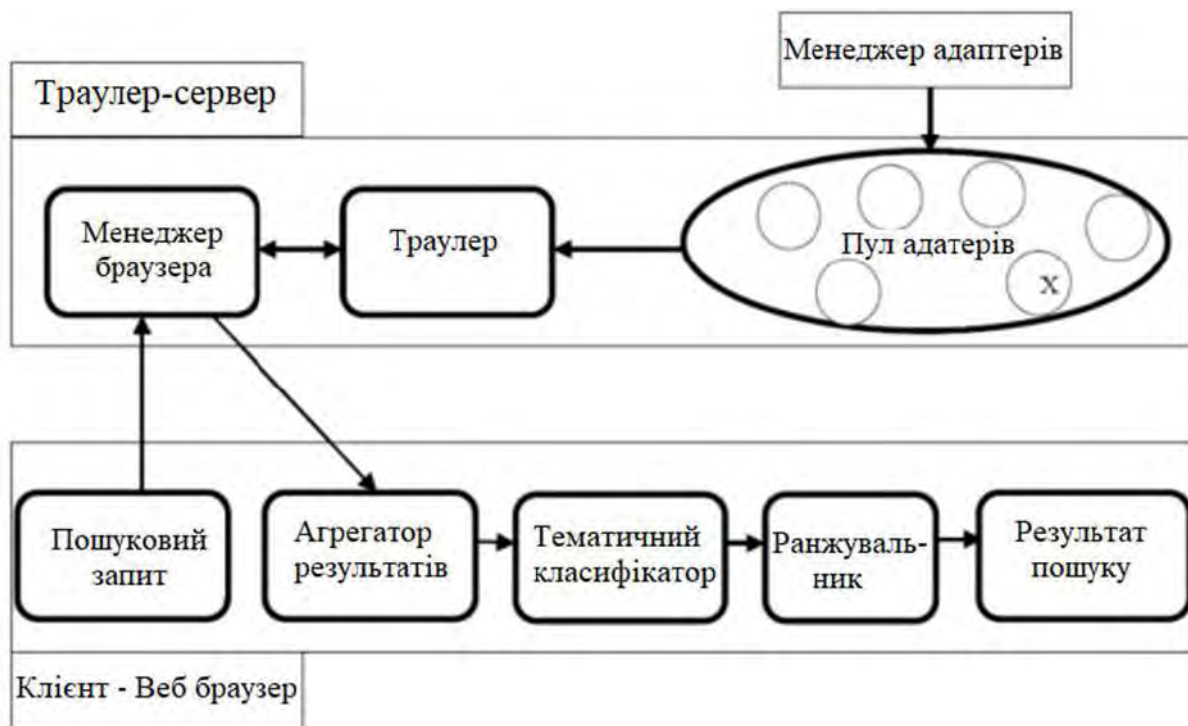


Рис. 3. Взаємодія клієнта (веббраузера) та траулер-сервера у архітектурі Turbo10

Отже, процес функціонування метапошукової системи, коли користувач надсилає запит, не надто складний – у такому разі не задіяні всі її ресурси. Адже пошукова система функціонує у два етапи: збирання ресурсів для подальшого виконання процедур пошуку відповідно до попередніх (підготовчих) налаштувань та власне надання зібраних ресурсів для одержання результатів такого пошуку. Процес підготовки інтерфейсів адаптерів запитів описано вище, і для активації пошукових ресурсів використовується заздалегідь сформований на траулер-сервері пул цих адаптерів (пошуковий запит передається серверу, сервер повертає необхідні адаптери, після цього перебіг пошуку інформації відбувається асинхронно, а у клієнтському браузері формуються і відображаються результати пошукового запиту).

Архітектура *Deep Bot*. У попередній архітектурі багато уваги звернено на реалізацію парсингу та перевірку структури “глибоких” вебресурсів. Архітектура типу *DeepBot* – прототип системи, яка, на відміну від *Turbo10*, шукає власне ресурси. Основні особливості системи – автоматизовані та налаштовані браузери (клієнти) і процес виявлення форм на відповідних вебресурсах.

Архітектура вебкраулера типу *Deep Bot* містить такі компоненти: менеджер конфігурацій, менеджер завантажень та менеджер контенту. Менеджер маршрутів, база даних для сканованих документів, індексатор, браузер-клієнти для управління вебкраулером та інтерфейс для користувачів, які здійснюють пошук за допомогою такої пошукової системи, також є елементами архітектури, але, по суті, не є компонентами ядра вебкраулера. Цю архітектуру в загальному вигляді подано на рис. 4.

Перший із компонентів вебкраулера – це менеджер конфігурацій. Цей компонент задає початкові умови функціонування пошукової системи. Він містить початкові адреси/маршрути, з яких потрібно починати вебпошук, налаштування глибини сканування інформації на кожному вебсайті, правила, які регламентують процеси оброблення різних типів документів, що завантажуватимуться під час роботи вебкраулера, та список винятків (URL-адрес). Другий компонент – менеджер завантажень. Цей компонент відповідає за вибір правильного обробника для документа, який вебкраулер намагається завантажити. І останній компонент вебкраулера – це контент-менеджер, який керує роботою системних парсерів, які визначають, чи цей документ придатний для зберігання та індексації, вони ж розпочинають процеси, що запускаються після вдалого збереження контенту, які аналізують інформацію (його вміст) [9].

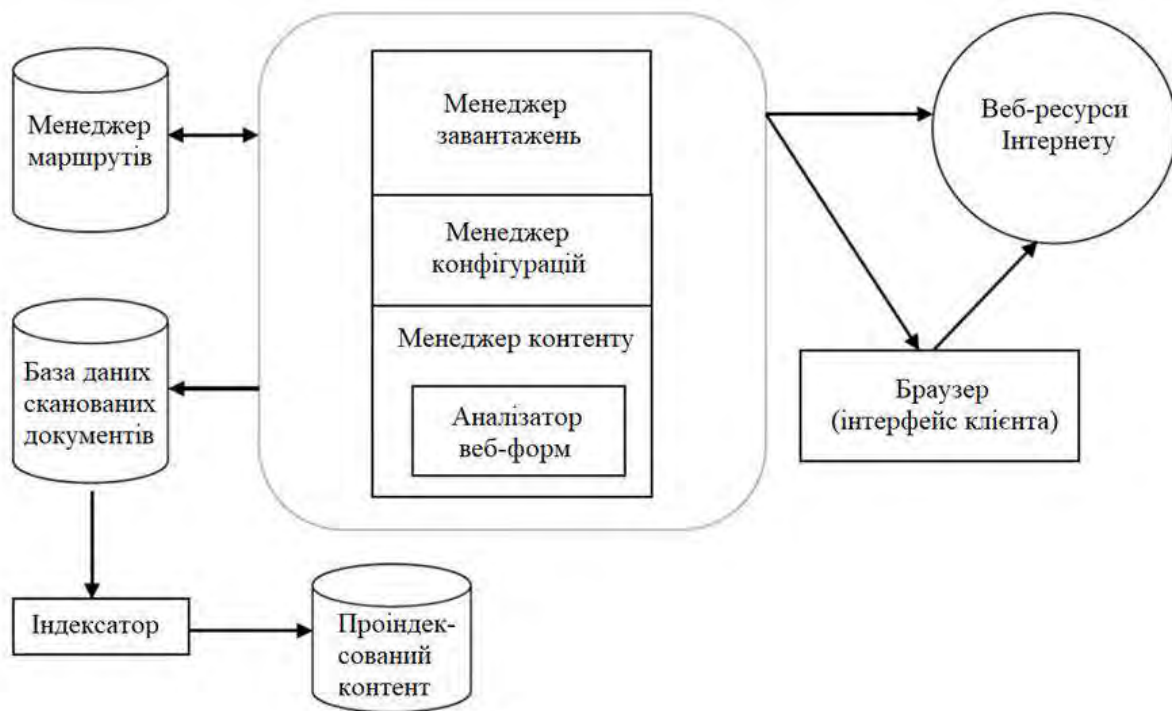


Рис. 4. Структура метапошукової машини архітектури Deep Bot

Компоненти поза ядром вебкраулера підтримують його. Менеджер маршрутів зберігає основний список маршрутів/вебадрес, до яких вебкраулери можуть отримати доступ. Кожен із вебкраулера (які, загалом, можуть функціонувати паралельно) зможе вибрати маршрут із цього списку після того, як розпочнеться сканування/пошук. Спеціалізований браузер вебклієнта створюється за допомогою стандартних API веббраузерів. Реалізуючи такі API, браузер вебклієнта одержує, наприклад, можливості працювати зі сценарієм на стороні клієнта (виконання певних скриптів), підтримувати механізми сеансів та керувати переспрямуванням за посиланнями на завантажених вебресурсах, які сканують [9].

Традиційний спосіб створення вебкраулера передбачає процедуру збереження посилань, які збігаються з елементами прив'язки HTML-коду. З елемента прив'язки одержують атрибут для її виконання, оскільки він містить URL-посилання. Однак Deep Bot архітектура передбачає використання ресурсів браузера для сканування. Це надає перевагу, яка полягає у тому, що можливо зберігати ідентифікатори сеансу, а також система може уникнути інтерпретації складних скриптів у HTML-коді, оскільки замість отримання та оброблення інформації з атрибута елемента прив'язки фактично відбувається “натискання” на посилання. Також системі вебкраулера не потрібно брати до уваги особливості переспрямування, оскільки клієнтський веббраузер виконує його так, як це роблять звичайні веббраузери користувачів. Навігаційна послідовність у браузерах вебклієнтів реалізується на основі мови програмування NSEQL, створеної спеціально для використання з браузерами.

Компонент “менеджер контенту” містить аналізатор форм. Цей підкомпонент обробляє поля, знайдені на вебресурсах. Це здійснюється у три етапи. По-перше, для кожної області вебресурсу система намагається зіставити задані атрибути пошуку із полями вебформи, використовуючи евристичні механізми для визначення подібності за візуальним розміщенням її елементів та текстом, питання щодо яких у цій роботі не розглянуто з огляду на їх обсяг. По-друге, за результатами виконання першого етапу визначається відповідність форми запитів атрибутам пошуку, формі полів введення деякої області вебресурсу, що аналізується. Нарешті, якщо остання форма визначена як релевантна, то вебкраулер використовуватиме її для здійснення запитів.

Отже, звертаючись до тематики цієї роботи, очевидно, що потрібно реалізувати вебкраулер для здійснення “глибоких” вебзапитів. Завдання для такого проєкту складаються з трьох пунктів.

Основне завдання – реалізація запиту для створення бази даних на основі “глибокого” пошуку без урахування специфіки коду сайтів, які скануються, або використання призначеного API для сканованої системи вебресурсів. Зрозуміло, що система програмних вебкраулерів не призначена для роботи з усіма можливими інтерфейсами запитів для здійснення пошуку, тому вона, по суті, завжди перебуватиме у стані прототипу. Тобто основного завдання досягають, коли хоча б деяка кількість “глибоких” вебресурсів відповідає на запити, що уможлиблює практичне застосування одержаних результатів. Вторинне завдання полягає у створенні функціонального вебкраулера, який здійснюватиме сканування інтернет-простору і пошук HTML-документів, що підлягають індексуванню. Вторинне завдання на практиці охопить сканування вебресурсів. Третинним завданням є збирання статистики та формування бази даних сканованих вебресурсів. Деякі із системних функцій вебкраулера призначені лише для збирання статистичних даних та фіксації результатів дій під час його застосування.

Отже, реалізація таких вебкраулерів має подвійну мету: нам потрібна система сканування вебресурсів для того, щоб збирати колекції вебсайтів, а також програма для сканування та опрацювання результатів запитів “глибоких” вебресурсів. Інші обов’язкові компоненти системи повинні відповідати за її конфігурування та за формування баз даних результатів пошуку та аналізу. Пропоновану результуючу архітектуру прототипу вебкраулера для застосування у інтерактивних системах поширення електронного контенту та підтримки процедур зворотного зв’язку подано на рис. 5.

Системний клас вебкраулера для здійснення сканування повинен містити функції підключення до вебресурсів та завантаження HTML-документа, пошуку всіх прив’язок у HTML-документі та збирання метаданих документа HTML. Він повинен аналізувати документи HTML, віднаходити вебформи, виконувати класифікацію для кожного знайденого поля форми, зокрема з’ясовувати, чи якийсь метод використовує у вебформі знайдене поле – POST або GET під час подавання та ініціювання запиту. Вебкраулер повинен бути керованим згідно зі стратегією сканування вебресурсів та містити базу даних індексованих результатів пошуку.

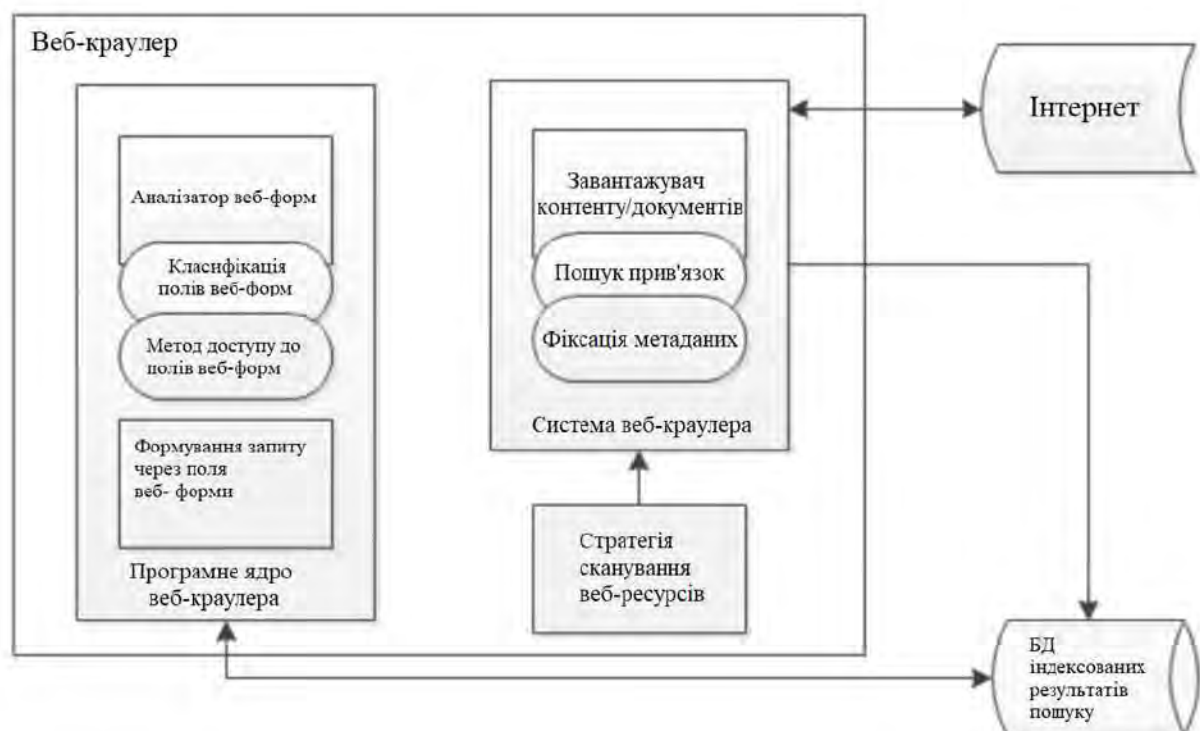


Рис. 5. Результуюча архітектура прототипу вебкраулера для застосування як системи зворотного зв’язку (на основі моніторингу із “глибоким” вебпошуком) в інтерактивних платформах поширення електронного контенту

Програмне забезпечення вебкраулера, що здійснює власне сканування, не є надто складним. Спочатку, на першому рівні сканування, відбувається аналіз та виділення вебформ тих вебсторінок,

які вдалося завантажити. Після успішної індексації та збереження метаданих і посилань із цих сторінок можливе ітераційне повторення вказаної процедури на глибших рівнях сканування, тобто здійснення “глибокого” вебпошуку. В міру наповнення бази даних індексованих результатів пошуку уможлиблюється їх аналізування. Стратегія повторного сканування – це, по суті, проста команда SQL, яка отримує список URL-адрес, що вже були проіндексовані, й передає їх до системи сканування вебкраулера, щоб знову індексувати та відстежити можливі зміни у контенті.

Можна констатувати, що створення ефективної метапошукової системи – це довготривалий проєкт, навіть для звичайного, “неглибокого” або ж поверхневого інтернету. Обсяг вимог до обладнання та величин пропускної спроможності каналів зв’язку для широкомасштабного використання вебкраулера є проблемою для реалізації таких пошукових систем. Основний обсяг пропускної здатності займає не перехід за посиланнями аналізованих вебсторінок, а процеси одержання результатів запитів, що здійснюються до виділених структурних елементів на цих сторінках, зокрема до відповідних вебформ з метою долучення відповідних результатів до бази даних.

З погляду архітектури програмного забезпечення система вебкраулера повинна вміти боротися з багатьма перешкодами, оскільки вона налаштована на роботу з не відомими наперед інтернет-джерелами. Такими перешкодами можуть бути тайм-аути сервера та різноманітні програмні пастки. Вебкраулери повинні бути створені так, щоб вони автоматично відновлювали своє функціонування, якщо система поверне помилку, яка б припиняла процес сканування. Щоб уникнути потрапляння до програмних пасток (наприклад, зациклювання), необхідно встановити жорсткіші обмеження щодо процедур сканування у поточній множині пов’язаних вебресурсів. Отже, виникає також обґрунтована необхідність обмежувати глибину сканування та частоту пошуку оновлень сканованого контенту, а також області сканування за деякими наперед визначеними умовами.

Окремим ускладненням під час реалізації вебкраулера, наприклад, за архітектурою, поданою на рис. 5, є масштаб завдань, які перед ними ставлять. Побудова таких систем ускладнюється, коли потрібно створити високопродуктивну платформу для пошуку оновлень у деякому (очевидно, значному) діапазоні вебресурсів. У такому разі для того, щоб наприклад, відстежувати оновлення 107 вебресурсів, вебкраулеру необхідно щосекунди завантажувати понад 4000 сторінок, а це ускладнить його взаємодію із базами даних, порушить питання створення високопродуктивних систем зберігання та архівації індексованого контенту, результатів його аналізу тощо. Звісно, вебкраулери таких масштабів можливо реалізувати лише у розподіленій версії для того, щоб кожна з фізичних платформ, на яких він функціонує, мала би змогу здійснювати кілька завантажень паралельно [10]. До того ж усі разом взяті компоненти розподіленого вебкраулера не повинні перевантажувати вебсервери, до яких вони звертаються, адже це спричинятиме наслідки, аналогічні до мережових атак типу “розподілена відмова в обслуговуванні” або просто “відмова в обслуговуванні” (DoS), якщо більшість запитів стосуються конкретного вебсервера і він не має достатніх засобів для їх опрацювання. Технічні обмеження автоматичного доступу до веббаз даних – основна причина, через яку важко отримати доступ до “глибокої” вебсторінки. Під час пошуку “глибоких” інтерфейсів для здійснення вебзапитів система повинна відкинути за налаштованою відповідно автоматизованою логікою невідповідні поля форм та не допустити їх подальшого опрацювання. Це можна здійснити, скориставшись такими методами, як мова адресації шляху (XPath, XPath) та скринінг екрана.

Після того як необхідні інтерфейси для здійснення “глибокого” пошуку одержано, необхідно здійснювати пошукові запити через них. Цього можна досягти, використовуючи специфічні для мови програмування бібліотеки функцій (cURL) або особливості конкретної мови програмування (вбудовані в неї методи). Якщо потрібно запитувати декілька пошукових інтерфейсів одночасно, можна передбачити функціональність, яка здатна перетворити запит на запити одночасно через усі визначені інтерфейси, відповідно до заданого пошукового контексту. Цього можна досягти за допомогою транслятора запитів, використовуючи, наприклад, відповідну семантичну структуру синтаксису [11]. Зрозуміло, що після надсилання запиту наступний крок веде або до одержання коду проміжної сторінки, або ж безпосередньо до сторінки із необхідними результатами. Результат зазвичай залежатиме від типу “глибокого” вебресурсу. Інтерфейси запитів можна розділити на дві групи. По-перше, інтерфейси для здійснення глобальних пошукових запитів (наприклад, Google)

пропонують користувачам можливість здійснювати пошук за ключовими словами. Це, врешті, приводить до проміжної сторінки, яка містить список можливих результатів, потенційно корисних для користувача. Проміжна сторінка – це завжди список результатів, які необхідно обробляти далі, щоб дійти до сторінки результатів, які можна використати для аналізування. По-друге, інтерфейси, які пропонують менш динамічні можливості для пошуку, можуть привести безпосередньо до сторінки результатів. Такі види інтерфейсів рідко дають змогу вести пошук за ключовими словами. Натомість вони працюють через високоієрархічні меню фільтрації, що, як правило, доволі легко автоматизувати. Після проходження всіх фільтрів одержують кінцевий результат, придатний для зберігання у базі даних та подальшого аналізу.

Аналіз підходів корпорації Google до інтерактивного управління подіями у процесі поширення контенту із їх фіксацією та обробленням у реальному часі

Очевидно, що провідні інформаційно-комунікаційні провайдери упроваджують вебкраулерну архітектуру в розподіленому режимі із використанням хмарних технологій та ресурсів відповідних платформ [12, 13]. Це дає змогу виконувати високопродуктивний та доволі прецизійний моніторинг інформаційних ресурсів у глобальному масштабі та в реальному часі. Звісно, на практиці це здійснюється з комерційною метою, оскільки потребує розгортання ресурсозатратної архітектури, наприклад, такої як на рис. 6.

Архітектура системи, поданої на рис. 6, призначена для управління показами контекстної реклами в реальному часі. Але її можна адаптувати для контрольованого поширення електронного контенту на основі приватно-державного партнерства, із урахуванням необхідних параметрів (цільової аудиторії, обсягів, реакції користувачів, умов керування тощо) у межах вибраної стратегії, що особливо корисно у разі реалізації інформаційної політики із застосуванням систем електронного урядування.

Архітектура, зображена на рис. 6, приймає рішення про цільове поширення того чи іншого контенту на основі технології BigQuery та алгоритмів, покладених в основу інтелектуальної бази даних, у розпорядженні якої профілі усіх користувачів хмарної платформи чи партнерських ресурсів. Можливо фіксувати усі значущі дії цих користувачів, зіставляючи їх із пропонованим контентом та реакціями на нього. Зазначена архітектура пропонує конкретним користувачам оплачений рекламодавцем контент у межах того чи іншого бюджету на рекламну кампанію (оплати за поширення цифрового контенту). Крім того, відстежують дії щодо пропонованого контенту, реакцію, переходи за пропонованим посиланням, дії, які очікує рекламодавець, під час замовлення тих чи інших товарів або ж послуг. Всі ці події впливають на оптимізацію параметрів показів того чи іншого контенту або хоча би оформлення відповідної підписки чи реєстрації. Звичайно, не увесь контент може бути платним, оскільки це залежить від конкретної платформи, через яку його поширюють, державної інформаційної політики.

Отже, описана платформа містить вже готові механізми зворотного зв'язку для аналізу настроїв цільової аудиторії (наприклад, за запитом через BigQuery) та керування поширенням електронного контенту того чи іншого типу, наприклад, медіафайлів, ключових повідомлень або з використанням можливостей штучного інтелекту, реалізованих на основі інтелектуальної бази даних.

Описані вище прототипи вебкраулерів можна увести до складу наведеної архітектури з метою наповнення та актуалізації інтелектуальної бази даних, яка містить профілі користувачів (що становлять основу цільової аудиторії).

Під час роботи із подіями важливий пошук правильного балансу між ціною, релевантністю даних (електронного контенту) та експлуатаційними витратами, причому на кожному етапі процесу, особливо за умов:

- збирання та введення великої кількості швидкоплинних подій, які виникають унаслідок показів контенту, переходів за відповідними посиланнями, конверсії та запитів щодо контекстуальної тематики оголошень;



Рис. 6. Архітектура мережевої платформи для фіксації та оброблення подій у реальному часі на основі технологій корпорації Google

- оброблення подій у режимі реального часу або в режимі офлайн для отримання статистичних показників, таких як ступінь використання бюджету, показник переглядів, реакція користувачів;

- експорту результатів у різні бази цифрового контенту в режимі реального часу або в режимі офлайн, щоб узгодити виставлення рахунків або примусово виконувати належні операції з обслуговування та поширення електронного контенту.

Під час реалізації платформи інтерактивного поширення електронного контенту згідно із рис. 6:

- Інформація про реакцію та дії користувачів надсилається у вигляді запитів до HTTP-коду, розміщеного в Cloud Storage, або до колектора вебсерверів, розміщеного в Google Kubernetes Engine (GKE). GKE – це кероване, готове до промислового використання середовище для розгортання контейнерних програм. Воно пропонує найрелевантніші рішення щодо продуктивності від розробників програмних платформ, а також щодо ефективності використання ресурсів, автоматизації операцій та гнучкості систем із відкритим кодом для того, щоб пришвидшити вихід рішень на ринок або їх уведення в експлуатацію.

- Події, що стосуються кожного такого запиту, реєструються через журнали логування HTTP-серверів після розподілу відповідних даних із балансувальника навантаження, а події, захоплені та зафіксовані колекторами, публікуються через сервіс хмарної публікації/підписки.

- Система управління потоками даних у хмарній платформі підписується на тему/контекст сервісу хмарної публікації/підписки та обробляє відповідні (відфільтровані) події, вносячи зміни до інтелектуальної бази даних (БД), що стають доступними під час наступних аналітичних запитів на вимогу.

- Хмарний потік даних експортує необроблені та/або оброблені події у BigQuery систему для аналізування та до інтелектуальної бази даних (щодо необхідного контексту) для виконання операцій білінгу/оновлення залишків бюджетів тощо.

Висновки

У цій публікації розглянуто архітектурні особливості побудови систем цільового поширення електронного контенту для побудови інтерактивних платформ електронного урядування у частині систем зворотного зв'язку на основі найхарактерніших та найефективніших архітектур вебкраулерів, а також практичного досвіду компанії Google. Проаналізовано основні особливості реалізування вебкраулерів як систем “глибокого” пошуку й основи для наскрізного моніторингу в державному інтернет-просторі у контексті програмної архітектури, а також труднощі їх інтеграції у глобальний інформаційно-комунікаційний простір, окреслено найдоцільніші способи їх подолання. Подано архітектуру мережевої платформи для фіксації та оброблення подій у реальному часі на основі технологій корпорації Google, що може бути базисом для швидкого та ресурсоефективного розгортання систем цільової доставки заданого електронного контенту під час реалізації вибраної стратегії електронного урядування, надаючи відповідній платформі інтерактивних якостей за рахунок реалізації каналів зворотного зв'язку із суспільством, які оснащені засобами фіксації реакції представників цільової аудиторії на інформаційні повідомлення та необхідним аналітичним інструментарієм (наприклад, BigQuery). Такі ефективні (зокрема за рахунок хмарного базування) рішення можна реалізовувати також на основі платформ таких соціальних мереж, як Facebook, Twitter, цифрових месенджерів Viber, Telegram тощо. Конкретний шлях реалізації таких рішень, очевидно, залежатиме від вибраної державної політики та комерційної доцільності, зокрема в аспекті державно-приватного партнерства.

Під час майбутніх досліджень, на думку авторів, доцільно перевірити гіпотезу, яку автори висловили у роботі [8], про можливість впливати на інтенсивність реакції аудиторії, змінюючи інтенсивність керівних впливів із використанням поширення деякого набору визначених ключових повідомлень. Своєю чергою, це потребуватиме дослідження показника ефективності впливу зовнішніх факторів на інтенсивність реакції аудиторії під час проведення дискусії у реальному часі, результати яких, очевидно, можливо поширити на аналогічні процеси обговорення інформації в інтернет-просторі. Відповідний аналіз буде виконаний на основі відкритих джерел (доступного в інтернеті медіаконтенту).

Список використаних джерел

1. Kaddouri, A., Guezouri, M. and Mbarek, N. (2017). A new inter-cloud service-level guarantee protocol applied to space missions. *International Journal of Grid and Utility Computing*, 8(2), p. 152. doi: 10.1504/ijguc.2017.085909.
2. Jiang, L., Feng, G. and Qin, S. (2015). Content Distribution for 5G Systems Based on Distributed Cloud Service Network Architecture. *KSII Transactions on Internet and Information Systems*, 9(11), pp. 4268–4290. doi: 10.3837/tiis.2015.11.001.
3. Wen, Y., Shi, G., & Wang, G. (2011). Designing an inter-cloud messaging protocol for content distribution as a service (CoDaaS) over future internet. In: *6Th International Conference On Future Internet Technologies – CFI '11*, pp. 91–93. doi: 10.1145/2002396.2002420.
4. Lehtonen, J. (2011). Characterizing the deep Web. Master's thesis. University of Turku.
5. Zhao, H. (2015). Research on Detection Algorithm of WEB Crawler. *International Journal of Security and Its Applications*, 9(10), pp. 137–146.
6. Hamilton, N. (2003): *The Mechanics of a Deep Net Metasearch Engine*. In: *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary.
7. Graham, A. (2004). DMOZ — Directory Mozilla The Open Directory Project, <http://dmoz.org>. *The Physics Teacher*, 42(4), pp. 255–255.
8. Demydov, I., Klymash, M., Najm Ahmad Baydoun, Branytskyy, A. (2019). To the Strategy Of Informational Impacts In Cyberspace By Omori's Law. In: *Proceedings of 3rd IEEE International Conference on Advanced Information and Communication Technologies-2019 (AICT-2019)*, Lviv, Ukraine.
9. Alvarez, M., Raposo, J., Cacheda, F., Pan, A. (2006). A Task-specific Approach for Crawling the Deep Web. *Engineering Letters*, 13.
10. Olston, C. and Najork, M. (2010). Web Crawling. *Foundations and Trends® in Information Retrieval*, 4(3), pp. 175–246.
11. Salahshour, F. and Tavoli, R. (2016). Web interface query integration in holistic matching approach using clustering algorithms. *International Journal of Computer Science Issues*, 13(5), pp. 163–170.
12. Singh, V. and Rai, D. (2019). Implementation and Performance Analysis of Multi Hop Ad Hoc Cloud Network and Servers using Ad Hoc Network Protocols. *International Journal of Innovative Technology and Exploring Engineering*, 8(10), pp. 4671–4679.
13. Google Cloud. (2019). Cloud Computing Services / Google Cloud. [online] Available at: <https://cloud.google.com/>.

References

1. Kaddouri, A., Guezouri, M. and Mbarek, N. (2017). A new inter-cloud service-level guarantee protocol applied to space missions. *International Journal of Grid and Utility Computing*, 8(2), p. 152. doi: 10.1504/ijguc.2017.085909.
2. Jiang, L., Feng, G. and Qin, S. (2015). Content Distribution for 5G Systems Based on Distributed Cloud Service Network Architecture. *KSII Transactions on Internet and Information Systems*, 9(11), pp. 4268–4290. doi: 10.3837/tiis.2015.11.001.
3. Wen, Y., Shi, G., & Wang, G. (2011). Designing an inter-cloud messaging protocol for content distribution as a service (CoDaaS) over future internet. In: *6Th International Conference On Future Internet Technologies - CFI '11*, pp. 91–93. doi: 10.1145/2002396.2002420.
4. Lehtonen, J. (2011). Characterizing the deep Web. Master's thesis. University of Turku.
5. Zhao, H. (2015). Research on Detection Algorithm of WEB Crawler. *International Journal of Security and Its Applications*, 9(10), pp. 137–146.
6. Hamilton, N. (2003): *The Mechanics of a Deep Net Metasearch Engine*. In: *Proceedings of the 12th International World Wide Web Conference*, Budapest, Hungary.
7. Graham, A. (2004). DMOZ — Directory Mozilla The Open Directory Project, <http://dmoz.org>. *The Physics Teacher*, 42(4), pp. 255–255.

8. Demydov, I., Klymash, M., Najm Ahmad Baydoun, Branytskyy, A. (2019). *To the Strategy Of Informational Impacts In Cyberspace By Omori's Law*. In: *Proceedings of 3rd IEEE International Conference on Advanced Information and Communication Technologies-2019 (AICT-2019)*, Lviv, Ukraine.
9. Alvarez, M., Raposo, J., Cacheda, F., Pan, A. (2006). *A Task-specific Approach for Crawling the Deep Web*. *Engineering Letters*, 13.
10. Olston, C. and Najork, M. (2010). *Web Crawling. Foundations and Trends® in Information Retrieval*, 4(3), pp. 175–246.
11. Salahshour, F. and Tavoli, R. (2016). *Web interface query integration in holistic matching approach using clustering algorithms*. *International Journal of Computer Science Issues*, 13(5), pp. 163-170.
12. Singh, V. and Rai, D. (2019). *Implementation and Performance Analysis of Multi Hop Ad Hoc Cloud Network and Servers using Ad Hoc Network Protocols*. *International Journal of Innovative Technology and Exploring Engineering*, 8(10), pp. 4671–4679.
13. Google Cloud. (2019). *Cloud Computing Services / Google Cloud*. [online] Available at: <https://cloud.google.com/>.