ВІСНИК ПРИАЗОВСЬКОГО ДЕРЖАВНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
2023р. Серія: Технічні науки Вип. 47
p-ISSN: 2225-6733; e-ISSN: 2519-271X

УДК 004.942 (075.8) **doi: 10.31498/2225-6733.47.2023.299979**

© Краснопьоров О.В.[1], Кривенко О.В.[2], Левицька Т.О.[3]

ДОСЛІДЖЕННЯ МЕТОДОЛОГІЙ ПОШУКУ ТЕКСТОВОЇ ІНФОРМАЦІЇ ІЗ ВИКОРИСТАННЯМ МОЖЛИВОСТЕЙ ПЛАТФОРМИ ELASTIC

Ця робота присвячена аналізу та оптимізації процесу пошуку додатків з використанням програмного засобу ElasticSearch. Предметом дослідження є платформа Elastic у контексті пошуку та аналізу даних. Об'єктом – оптимізація процесу пошуку та аналізу даних на базі цієї платформи. Метою даної роботи є дослідження можливостей та особливостей використання ElasticSearch для створення ефективного пошукового механізму додатків у великих аплікаційних магазинах. Визначено основні завдання для досягнення мети роботи. Проведений аналіз наукової літератури щодо методів та технологій пошуку та аналізу даних на основі платформи Elastic. Розглянуті основні складові та можливості платформи Elastic, включаючи ElasticSearch, Kibana та LogStash. Здійснений порівняльний аналіз ефективності пошуку на платформі Elastic та альтернативних рішень. Розроблений та реалізований тестовий сценарій для оцінки швидкодії та точності пошуку за допомогою платформи. Реалізований сервіс, який забезпечує функціонал пошуку для додатків. Розглянуті можливості інтеграції розробленого сервісу з існуючими додатками та системами для оптимального використання його функціоналу. Проведено тестування розробленого сервісу з використанням різних наборів даних для підтвердження його ефективності та точності. Визначені можливості масштабування та оптимізації розробленого сервісу для оптимальної продуктивності при використанні великих обсягів даних. Порівняно результати використання розробленого сервісу з аналогічними рішеннями та надані висновки щодо його конкурентоспроможності. Розроблена інструкція з використання розробленого сервісу та надані рекомендації щодо його ефективного впровадження в практичних сценаріях. Проведений аналіз отриманих результатів та зроблені висновки щодо ефективності та практичної цінності розробленого сервісу для вирішення конкретних завдань обробки та аналізу великих обсягів даних за допомогою платформи Elastic. Проведено емпіричне дослідження, використовуючи реальні набори даних, для оцінки ефективності пошуку на платформі. Вивчені можливості оптимізації та підвищення продуктивності пошуку на платформі шляхом конфігурації та налаштування. Зроблені висновки щодо ефективності пошуку за допомогою платформи Elastic та надані рекомендації щодо її використання в конкретних сценаріях.
Ключові слова: *пошук, дані, платформа, бази даних, сервіс, додаток, C#, Elastic, програмне забезпечення, Logstash, Index.*

A.V. Krasnoperov, O.V. Kryvenko, T.O. Levytska. Research of text information search methods using the capabilities of the Elastic platform. *This work is devoted to the analysis and optimization of the application search process using the ElasticSearch software tool. The subject of research is the Elastic platform in the context of data retrieval and analysis. The object is to optimize the search and data analysis process based on this platform. The purpose of this work is to study the possibilities and features of using ElasticSearch to create an effective application search mechanism in large application stores. The main tasks for achieving the goal of the work are defined. Conducted analysis of scientific literature on methods and technologies of data search and analysis based on the Elastic*

[1] *магістрант, ДВНЗ «Приазовський державний технічний університет», м. Дніпро*
[2] *канд. техн. наук, доцент, ДВНЗ «Приазовський державний технічний університет», м. Дніпро, ORCID: 0009-0006-2860-6575, krivenko_o_v@pstu.edu*
[3] *канд. техн. наук, доцент, ДВНЗ «Приазовський державний технічний університет», м. Дніпро, ORCID: 0000-0003-3359-1313, levitskaya_t_a@pstu.edu*

p-ISSN: 2225-6733; e-ISSN: 2519-271X

platform. Covers the core components and capabilities of the Elastic platform, including ElasticSearch, Kibana, and LogStash. A comparative analysis of search performance on the Elastic platform and alternative solutions was performed. A test scenario was developed and implemented to evaluate the speed and accuracy of the search using the platform. Implemented service that provides search functionality for applications. The possibilities of integrating the developed service with existing applications and systems for optimal use of its functionality are considered. The developed service was tested using various data sets to confirm its effectiveness and accuracy. The possibilities of scaling and optimization of the developed service for optimal performance when using large volumes of data are determined. The results of using the developed service are compared with similar solutions and conclusions are given regarding its competitiveness. Instructions for using the developed service have been developed and recommendations for its effective implementation in practical scenarios have been provided. The analysis of the obtained results was carried out and conclusions were drawn regarding the effectiveness and practical value of the developed service for solving specific tasks of processing and analyzing large volumes of data using the Elastic platform. Empirical research was conducted using real data sets to evaluate the performance of search on the platform. Explored opportunities to optimize and improve search performance on the platform through configuration and customization. Conclusions are made regarding the effectiveness of search using the Elastic platform and recommendations are provided for its use in specific scenarios.
***Key words**: search, data, platform, databases, service, application, C#, Elastic, software, Logstash, Index.*

Постановка проблеми. Сучасна інформаційна епоха супроводжується неабияким зростанням обсягів даних та інформації, доступ до яких стає все більш необхідним і складним завдяки швидкому розвитку цифрових технологій та інтернету.

Однією з найважливіших задач у сфері обробки та аналізу цих даних є оптимізація пошуку та навігації користувачів у великих обсягах інформації. І саме тут істотну роль відіграє пошуковий двигун, який дозволяє ефективно та швидко знаходити релевантну інформацію серед тисяч і мільйонів записів.

Одним із сучасних та потужних інструментів для реалізації пошуку є система ElasticSearch, яка є частиною платформи Elastic. ElasticSearch використовується в багатьох сферах, включаючи пошук на веб-сайтах, аналіз логів, обробку даних та інші завдання. Ця система здатна працювати з великими обсягами даних та забезпечує високу швидкість пошуку завдяки використанню розподіленої архітектури та індексації даних.

Ця робота присвячена вивченню та оптимізації процесу пошуку додатків з використанням програмного засобу ElasticSearch. Розглядаємо різні аспекти оптимізації, включаючи налаштування індексів, оптимізацію запитів та використання аналітики для покращення результатів пошуку.

Актуальність даного дослідження визначила проблему відсутності релевантного та швидкого функціоналу пошуку в більшості корпоративних програмних продуктів.

Аналіз останніх досліджень і публікацій. Традиційні реляційні бази даних (наприклад, MySQL, PostgreSQL, Oracle, MS SQL Server) зазвичай не спеціалізуються на повнотекстовому пошуку. Основна їхні задача – забезпечення збереження і ефективного витягування даних відповідно до схеми таблиць.

Хоча деякі традиційні СУБД (наприклад, PostgreSQL) надають базові можливості повнотекстового пошуку через спеціалізовані розширення або модулі, це все ж не є основною сферою їхньої ефективності. У таких випадках рекомендується використовувати спеціалізовані повнотекстові пошукові двигуни, такі як Elasticsearch, Solr, або спеціалізовані бази даних, як ClickHouse, для аналізу та обробки великих обсягів даних [1].

Повнотекстовий пошук SQL Server є дуже потужним. Принаймні у нього є низка корисних можливостей: пошук префіксів, слів поруч одне з одним, різні часи дієслів і навіть пошук за тезаурусом. Однак, більшість розробників використовує його не так ефективно: багато магазинів використовували його для зіставлення певних рядків, думаючи, що це буде швидше, ніж LIKE

p-ISSN: 2225-6733; e-ISSN: 2519-271X

«%searchword%». Це працює в невеликому масштабі, але в міру того, як ваші дані зростають, ви зіткнетеся з проблемою продуктивності плану запитів [2].

Розглянемо, які можуть бути альтернативи традиційним пошуковим інструментам на базі традиційних баз даних [3-5]. На основі даних ранжирування популярних движків зберігання та пошуку даних для розгляду програмних рішень проаналізуємо такі продукти повнотекстового пошуку: Solr, Elastic Stack, Sphinx, ClickHouse (Табл. 1).

Таблиця 1

Порівняння розповсюджених на ринку засобів для пошуку даних поза традиційних реляційних БД

	Solr	**Elastic**	**Sphinx**	**ClickHouse**
Швидкість індексації	2,8 Мб/с	3,9 Мб/с	4,5 Мб/с	Залежить від сценарію
Швидкість пошуку	24 мс	10 мс	7 мс	100K+ RPS/c
Розмір індексу	20%	20%	30%	30%
Підтримка функцій	Full Text Search Autocomplete Faceted Multifield Synonyms Fuzzy Highlighting Geospatial Spellchecking	Full Text Search Autocomplete Faceted Multifield Synonyms Geospatial	Full Text Search Autocomplete Faceted Multifield Synonyms (word-forms) Highlighting (snippents) Geospatial Spellchecking (qsuggest)	Full Text Search Autocomplete Geospatial SQL
Індексація в реальному часі	Так	Так	Так	Ні
Підтримка ОС	Any OS with JVM	Any OS with JVM	BSD, Linux, OsX, Solaris, Windows	Linux
Схема даних	Не обов’язкова	Не обов’язково	Так	Рекомендовано
Візуалізація	Немає	Так	Немає	Немає
Підтримка API	Java API, RESTful API	Java API, RESTful API	Patented protocol	OJAI API
Підтримка скриптів та розширень	Java Plugins	Yes	No	Python Plugins
Мова розробки	Java	Java	C++	C++

Мета дослідження. Метою роботи є визначення та оцінка ефективності пошуку за допомогою платформи Elastic у контексті пошуку релевантної текстової інформації на великих обсягах структурованої та неструктурованої інформації. Для її досягнення були поставлені наступні завдання:

– провести аналіз наукової літератури щодо методів та технологій пошуку та аналізу даних на основі платформи Elastic;

– розглянути основні складові та можливості платформи Elastic, включаючи ElasticSearch, Kibana та LogStash;

– здійснити порівняльний аналіз ефективності пошуку на платформі Elastic та альтернативних рішень;

– розробити та реалізувати тестовий сценарій для оцінки швидкодії та точності пошуку за допомогою платформи Elastic;

– реалізувати сервіс, який забезпечує функціонал пошуку для додатків;

p-ISSN: 2225-6733; e-ISSN: 2519-271X

– розглянути можливості інтеграції розробленого сервісу з існуючими додатками та системами для оптимального використання його функціоналу;

– провести тестування розробленого сервісу з використанням різних наборів даних для підтвердження його ефективності та точності;

– визначити можливості масштабування та оптимізації розробленого сервісу для оптимальної продуктивності при використанні великих обсягів даних;

– порівняти результати використання розробленого сервісу з аналогічними рішеннями та надати висновки щодо його конкурентоспроможності;

– розробити інструкцію з використання розробленого сервісу та надати рекомендації щодо його ефективного впровадження в практичних сценаріях;

– провести аналіз отриманих результатів та зробити висновки щодо ефективності та практичної цінності розробленого сервісу для вирішення конкретних завдань обробки та аналізу великих обсягів даних за допомогою платформи Elastic;

– провести емпіричне дослідження, використовуючи реальні набори даних, для оцінки ефективності пошуку на платформі Elastic;

– вивчити можливості оптимізації та підвищення продуктивності пошуку на платформі Elastic шляхом конфігурації та налаштування;

– зробити висновки щодо ефективності пошуку за допомогою платформи Elastic та надати рекомендації щодо її використання в конкретних сценаріях.

Виклад основного матеріалу. Для розробки сервісу, який буде виконувати роль middleware для бізнес додатків, обрана мова програмування C# (C Sharp), яка є хорошим вибором для розробки клієнта для платформи Elastic.

Звичайно, C# не є єдиною мовою, яка може використовуватися для розробки клієнта для платформи Elastic. Інші популярні варіанти включають Java, Python, і Go. Однак C# є хорошим вибором для розробників, які шукають потужну і гнучку мову, яка підтримує широкий спектр завдань.

В роботі орієнтували наше рішення на базу даних MS SQL. MS SQL є однією з найпопулярніших баз даних у світі, і вона використовується в мільйонах підприємств по всьому світу (займає третє місце у рейтингу баз даних (БД) по версії TOPDB [6]). Це означає, що MS SQL має довгу історію стабільної роботи і має репутацію надійної технології.

Зауважимо, що Elastic Logstash підтримує широкий спектр баз даних, включаючи:

- Relational databases: MySQL, PostgreSQL, Oracle, Microsoft SQL Server, IBM DB2
- NoSQL databases: MongoDB, Elasticsearch, Cassandra, HBase, Neo4j
- Cloud databases: Amazon Redshift, Google Cloud BigQuery, Microsoft Azure SQL Database

Logstash може імпортувати дані з цих баз даних і експортувати дані в них. Ось деякі приклади того, як Logstash можна використовувати з базами даних: а) Імпорт даних: Logstash можна використовувати для імпорту даних з баз даних в інші системи, такі як Elasticsearch або Hadoop, б) експорт даних: Logstash можна використовувати для експорту даних з баз даних в інші системи, такі як електронні таблиці або системи відстеження помилок, в) аналіз даних: Logstash можна використовувати для аналізу даних з баз даних для виявлення тенденцій і аномалій.

Моделювання об’єкту дослідження:

1) Діаграма послідовності

Діаграма послідовності допомагає візуалізувати взаємодію між різними компонентами системи або процесу. Опишемо наші процеси (Рис. 1):

Logstash сервіс: Logstash сервіс загружає дані в індекс згідно заданого розкладу з БД.

Клієнтський додаток: Клієнтський додаток надсилає запити до сервісу пошуку.

Пошуковий запит: Клієнтський додаток формує запит на пошук конкретних даних та передає його за допомогою нашего кастомного сервісу пошуку на порт ElasticSearch ноди.

ElasticSearch сервіс: ElasticSearch обробляє запити та виконує пошук даних в індексах.

Обробка запиту ElasticSearch: ElasticSearch приймає пошуковий запит та аналізує його.

Пошук у індексах: ElasticSearch виконує пошук даних в Індексах, що включають в себе потрібну інформацію.

Результати пошуку: ElasticSearch повертає результати пошуку клієнтському додатку.

Відображення результатів: Бізнес додаток відображає результати пошуку користувачеві.

p-ISSN: 2225-6733; e-ISSN: 2519-271X

Залежно від потреби, можуть відбуватися додаткові ітерації пошуку або інші дії.

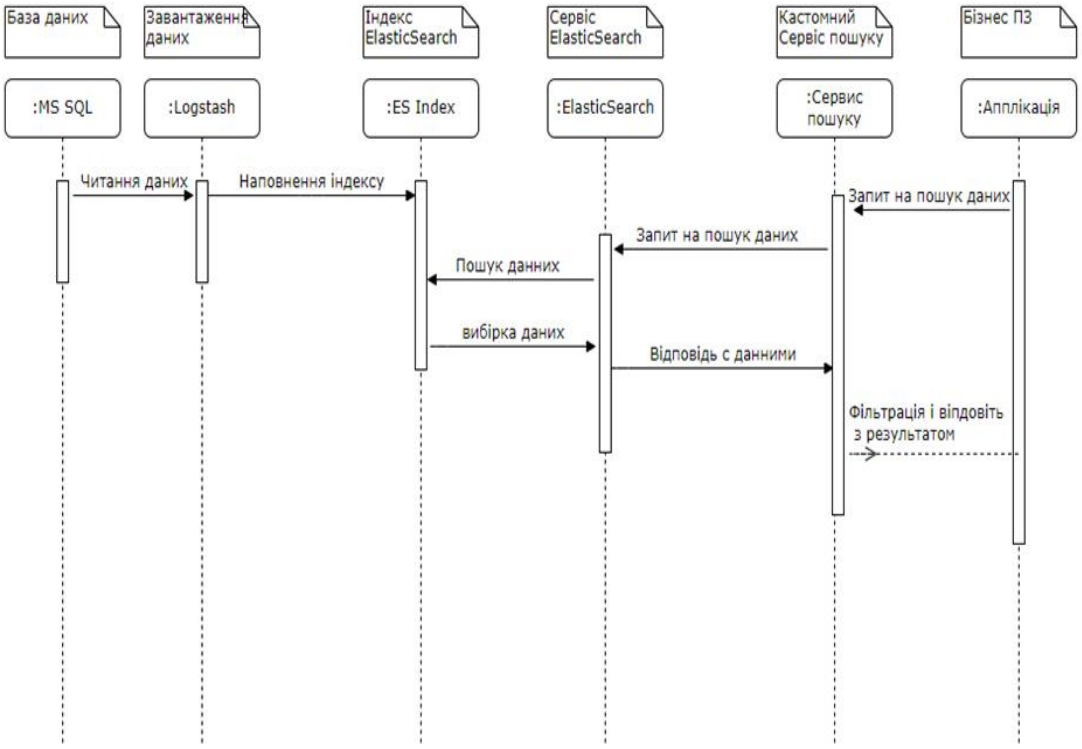

Рис. 1 – Діаграма послідовності запитів пошуку

2) Діаграма варіантів використання

Варіанти використання (Use Case Diagram) для сервісу пошуку в ElasticSearch індексі (Рис. 2).

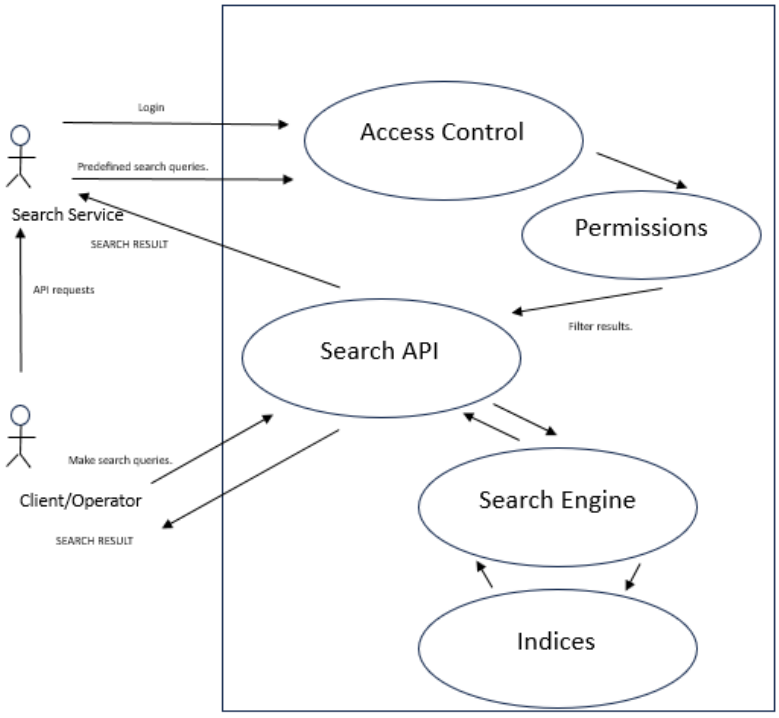

Рис. 2 – Діаграма варіантів використання

p-ISSN: 2225-6733; e-ISSN: 2519-271X

3) Інфологічна модель Індексів пошуку

Основні складові інфологічної моделі Elasticsearch включають:

Ноди: це екземпляри серверу ElasticSearch. Потрібен найменш один екземпляр для роботи індексу та не менш трьох нод для комерційної експлуатації.

Шарди: базовий блок побудови та розподілу даних. Це фундаментальна одиниця, на яку розбивається індекс для забезпечення паралельного пошуку та розподіленого зберігання даних.

Документи: Документ – це основний об'єкт, який містить дані. У Elasticsearch документи представлені у форматі JSON. Кожен документ має унікальний ідентифікатор.

Типи: Раніше у Elasticsearch існували типи, але починаючи з версії 7, вони були прибрані. Тепер кожен індекс може містити тільки один тип документу.

Індекси: Індекс – це колекція документів, яка має однакову структуру. Вони використовуються для організації даних та прискорення пошуку.

Поля: Кожен документ має поля, які містять конкретні дані. Поля можуть мати різний тип, такий як текстовий, числовий, дата, географічний тощо.

Аналізатори: Elasticsearch використовує аналізатори для обробки тексту перед зберіганням його у зворотному індексі. Аналізатори включають в себе токенізатори та фільтри, які перетворюють текст у набір токенів для полегшення пошуку.

Маппінг: Маппінг визначає, як поля в документах індексу інтерпретуються та індексуються. Маппінги можуть бути автоматично визначені Elasticsearch або встановлені користувачем.

Аналітика: Elasticsearch надає можливість використовувати агрегації для аналізу та обробки даних у великих масштабах.

Загалом, ці компоненти дозволяють Elasticsearch надати потужні можливості повнотекстового пошуку, агрегацій та аналітики, що робить його популярним інструментом для роботи з великими обсягами даних.

Створимо Індекс для пошуку постів з текстовою інформацією і приведемо інфологічну модель цього індексу (Рис. 3).

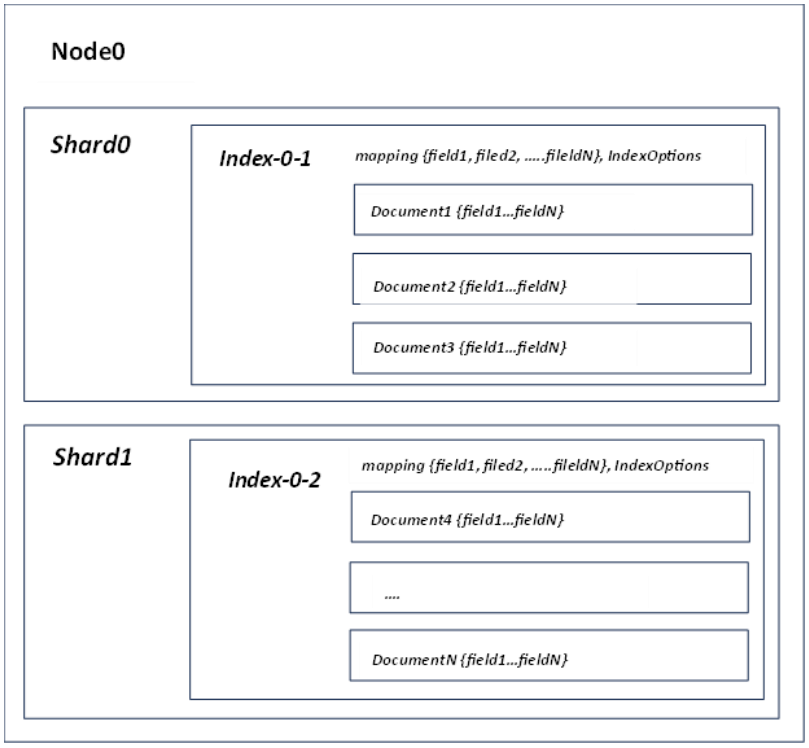

Рис. 3 – Інфологічна модель кластеру ElasticSearch

Експериментальні дослідження:

1) Програмне забезпечення експерименту

p-ISSN: 2225-6733; e-ISSN: 2519-271X

Стек Elastic платформи

Для розгортання рішення з пошуку даних за допомогою платформи Elastic необхідно розгорнути кластер ElasticSearch, створити індекс та налагодити імпорт даних, які використовували.

Використовували одну з останніх свіжих версій стека Elastic - 8.x. Кластер і стек інструментів розгортали на трьох додатках Elastic стеку:

1) ElasticSearch 8.10.2 – це наш головний двигун стеку і провайдер нашого пошукового сервісу.

2) Logstash 8.10.2 – за допомогою цього інструменту здійснювали імпорт даних із БД і створення індексу.

3) Kibana 8.10.2 – це інструмент візуалізації та інтерфейс користувача за допомогою якого, здійснюється управління адміністративними функціями стека, управління індексами, а також є можливість відладжувати запроси до індексів напряму в розділі Dev tools, а також здійснювати інші налаштування індексу та кластеру.

Слід зауважити, що версії додатків Elastic стека повинні бути однакові і бажано до мінорної версії. Деякі сервіси (наприклад, Kibana) можуть навіть не стартувати, якщо в налаштуваннях буде указано ElasticSeatrch хост, на якому буде версія движка нижче мінорної версії Kibana. Також додатково відмітимо, що мінорні версії оновлюються дуже часто – майже кожен тиждень, тому важливо слідкувати, щоб компоненти були однакових версій, але необов'язково намагатися завантажити саме останню версію стеку.

Операційні системи

Кластер може бути розгорнутий на Windows та Linux операційних системах. В рамках нашого дослідження використана операційна система Windows 10 Pro. Для нашого експерименту цього буде достатньо. Але в комерційній експлуатації рекомендовано використовувати серверні варіанти ОС Windows, такі як MS Windows 2016/2019 Server та більш свіжі версії. Це також дотичне до Linux систем – для розробки POC (Proof of Concept) дослідження можна використати Linux Ubuntu Desktop версію, але для комерційного використання краще інсталювати рішення на Linux Server редакції вашого Linux дистрибутиву.

Скриптові оболонки

Для створення індексів та імплементації деяких адміністративних операцій з індексами, використовували скрипти PowerShell (рекомендовано версії не нижче 7.x). Зауважимо, що Elastic є кросплатформним рішенням, тобто можемо використовувати за необхідністю Bash, Shell ті інші оболонки для написання скриптів, якщо в нас є таки вимоги. Але в цьому дослідженні зупинились на PowerShell.

Бази даних

Дані, які використовуємо для дослідження, завантажили до MS SQL. Використовували редакцію MS SQL 2016 Enterprise Edition x64. Це обумовлено великим об'ємом даних, які знадобляться для дослідження.

MS Visual Studio

Для розробки сервісу пошуку в якості IDE використаний MS Visual Studio 2023 редакції Community Edition x64, і також встановлений .NET Framework, що потрібен для розробки сервісів ASP.NET на мові C#.

Це базовий набір програмного забезпечення, що потрібен нам для дослідження і розробки сервісу пошуку даних. Можемо використати деякі утиліти та додатки (наприклад, архіватори, файлові менеджери), але вони лише полегшують процес дослідження і розробки, та не є базово необхідними інструментами.

2) Налаштування кластеру Elastic і імпорт даних.

Створення кластеру.

Для дослідження пошуку за допомогою Elastic платформи створили одно-нодовий кластер і налаштували імпорт даних с бази даних MS SQL за допомогою Logstash.

Встановлення програмного забезпечення є доволі простим – треба лише завантажити компоненти платформи (ElasticSearch, Logstash, Kibana) і разархівувати їх в обраній директорії. В нашому випадку \Elastic. Оскільки розгортаємо кластер на операційній системі MS Windows, треба перетворити виконуваний файл на службу Windows. Наведемо мінімально необхідний файл конфігурації ElasticSearch.

p-ISSN: 2225-6733; e-ISSN: 2519-271X

```
# ====== Elasticsearch Configuration ======
# ---------------- Cluster ----------------
#cluster.name: es-app812
#
# --------------- Node --------------------
#
node.name: es-node-1
#node.attr.rack: r1
#
# ----------------- Paths -------------------
#path.data: /path/to/data
#path.logs: /path/to/logs
#
# ---------------- Memory -------------------
#bootstrap.memory_lock: true
#
# ---------------- Network -----------------
#
network.host: 192.168.1.102
#
http.port: 9200
#
# ----------_---- Discovery --------------
discovery.seed_hosts: ["192.168.1.104", "127.0.0.1", "[::1]"]
#
cluster.initial_master_nodes: ["es-node-1"]
#
# ---------------- Various ----------------
#
#action.destructive_requires_name: false
xpack.security.enabled: false
```

Висновки

Провівши дослідження пошуку даних за допомогою платформи Elastic, дійшли низки висновків. В роботі провели налаштування кластеру ElasticSearch, створили індекс із бази даних MS SQL з більш ніж 50 мільйонів документів, провели заміри продуктивності різних запитів на пошук і агрегування даних, порівняли їх швидкість. Також створили POC (proof of concept) сервіс пошуку даних, який може виступати як middleware сервіс для надання інтерфейсу або шлюзу для інтеграції пошуку в бізнес додатках і може бути масштабований окремо від головного додатку.

Зазначимо ті покращення та можливості, які здобули, залучивши платформу Elastic до функцій пошуку даних в наших додатках.

По-перше, слід зауважити наявність дуже великої швидкості пошуку даних в структурованих документах в індексах, в порівнянні з традиційними реляційними базами даних. Отримано, швидкість пошуку може сягнути разів, або навіть порядків в залежності від складності запитів і типів даних.

По-друге, необхідно відмітити велику кількість різноманітних і гнучких запитів, які направлені як на прямий пошук і вибірку даних, так і на агрегацію даних, аналітику та навіть нечіткі запити.

Проаналізували можливості стандартних аналізаторів Elasticsearch і побачили їх потужність, коли залучаємо до пошуку запити з нечіткими даними пошуку (з помилками, синонімами або з іншим порядком слів).

Також по-третє, відмітимо, що платформа має потужний API і це полегшує інтеграційні процеси, бо все може бути реалізовано на звичайних HTTP запитах, які реалізовані майже у кожній мові програмування, скриптових мовах та маємо окремі команди та інструменти для операційних систем та командних оболонок. Навіть можна використати звичайний браузер, та робити запит до платформи Elastic. Саме це дуже полегшує розробку сервісів, що можуть взаємодіяти з платформою.

Ще одна важлива річ, з якою стикнулися автори і з якою стикнеться кожний розробник, який буде оперувати з великими обсягами даних в вибірках на сотні тисяч і мільйони документів, це те, що платформа Elastic і її система пошуку ElasticSearch побудовані за замовченням дуже надійно і має в першу чергу вимогу не порушити роботу кластера та тримати час відклику

p-ISSN: 2225-6733; e-ISSN: 2519-271X

максимально коротким, а це в свою чергу не дозволить вам запросити, наприклад, один мільйон документів у вашому запиті, що може на достатньо складних запитах вплинути на роботу всього кластера. Отже, вам знадобиться налаштовувати ваш кластер під великі запити, використовувати пагінацію, встановлювати розмір відповіді та налаштовувати тайм-аути.

Іншими словами, для великих обсягів вам доведеться не тільки створювати ваші запити, але і оптимізувати їх і налаштовувати ваш кластер під ваші потреби пошуку.

Отже, слід зазначити і особливості цього рішення – як бачимо, що ціна швидкості і додаткового функціоналу є підвищення вимог до користування цим інструментом, високий рівень кваліфікації фахівців, що адмініструють кластер, розробників і навіть системних інженерів, що налаштують оптимальну інфраструктуру під ці задачі.

Також маємо відмітити, що при потребі швидкого пошуку повинні нести витрати на дисковий простір для хостингу індексів. Що подвоює, а інколи і потроює потреби в дискових сховищах, тому що в більшості випадків ви не зможете відмовитись від реляційних БД в ваших додатках і повністю перейти на документи індексу. Вони мають різни призначення і різні задачі.

І на останнє, потрібно привести кроки, що можна зробити для покращення нашого сервісу. Треба розробити можливість зберігання конфігурації в зовнішньому сервісі конфігурації. Також слід додати можливість перевірок і переключення на пошук в базі даних при відсутності підключення до індексу. В експлуатації можна перевести сервіс в контейнер і використовувати його в системах оркестрації контейнерів, таких як Kubernetes, OpenShift та інших.

Перелік використаних джерел:

1. Обзор решений для полнотекстового поиска в веб-проектах: Sphinx, Apache Lucene, Xapian. URL: https://dou.ua/lenta/articles/full-text-search-engines-overview-sphinx-apache-lucene-xapian/ (дата звернення 28.07.2023).
2. Why Full Text's CONTAINS Queries Are So Slow. URL: https://www.brentozar.com/archive/2020/11/why-full-texts-contains-queries-are-so-slow (дата звернення 28.07.2023).
3. Apache Solr. URL: https://solr.apache.org (дата звернення 28.07.2023).
4. Elastic Stack. URL: https://www.elastic.co/elastic-stack (дата звернення 13.08.2023).
5. Croft W.B., Lafferty J. Language modeling for information retrieval. Springer Science & Business Media. 2003. 246 p. DOI: https://doi.org/10.1007/978-94-017-0171-6.
6. Different ways to model your data in ElasticSearch. URL: https://medium.com/@zhaoyi0113/different-ways-to-model-your-data-in-elasticsearch-bbc719f3d4fc (дата звернення 10.08.2023).

References:

1. Obzor reshenyi dlia polnotekstovoho poyska v veb-proektakh: Sphinx, Apache Lucene, Xapian (Review of solutions for full-text search in web projects: Sphinx, Apache Lucene, Xapian) [Online]. Available: https://dou.ua/lenta/articles/full-text-search-engines-overview-sphinx-apache-lucene-xapian/. Accessed on: July 28, 2023.
2. Why Full Text's CONTAINS Queries Are So Slow [Online]. Available: https://www.brentozar.com/archive/2020/11/why-full-texts-contains-queries-are-so-slow. Accessed on: July 28, 2023.
3. Apache Solr [Online]. Available: https://solr.apache.org. Accessed on: July 28, 2023.
4. Elastic Stack [Online]. Available: https://www.elastic.co/elastic-stack. Accessed on: August 13, 2023.
5. W.B. Croft, and J. Lafferty, *Language modeling for information retrieval*. Springer Science & Business Media Publ., 2003. **doi: 10.1007/978-94-017-0171-6**.
6. Different ways to model your data in ElasticSearch [Online]. Available: https://medium.com/@zhaoyi0113/different-ways-to-model-your-data-in-elasticsearch-bbc719f3d4fc. Accessed on: August 10, 2023.

Рецензент: О.Є. П'ятикоп
канд. техн. наук, доц., ДВНЗ «ПДТУ»

Стаття надійшла 10.10.2023
Стаття прийнята 07.11.2023