УДК **004.41**

М. О. Рейда[1]
О. О. Сергієнко[1]
О. О. Коваленко[1]

НОВІТНІ СТРУКТУРИ ДАНИХ

[1] Вінницький національний технічний університет;

Анотація

Проведено аналіз новітніх структур даних. Визначені структури, які працюють з багатопотоковими даними. Представлені приклади їх використання.

Ключові слова: структури даних, Bloom filter, HyperLogLog, Kafka, паралельна хеш-таблиця.

Abstract

The analysis of the modern data structures has been conducted. Structures, that work with multithreaded data are defined. Examples of their use are presented.

Keywords: data structures, Bloom filter, HyperLogLog, Kafka, parallel hash table.

Вступ

Структури даних – це спосіб організації даних в інформаційному просторі, а саме – на жорсткому диску, в оперативній пам’яті, інформаційних пакетах при передачі по лініям зв’язку тощо. Потреба у використанні структур даних викликана необхідністю структурованості та спрощеності даних і програмного коду. Процес обробки великих об’ємів даних потребує багато часу, тому актуальним постає питання паралельних обчислень при роботі з потоками даних. Для підвищення ефективності паралельних обчислень використовуються новітні, більш складні структури даних. Їх дослідження дозволяє виявити властивості та подальші сфери використання.

Мета – дослідити структури даних для роботи з великими обсягами даних, паралельними потоками та обчисленнями

Результати дослідження

Під час проведення дослідження було вирішено розділити структури даних на 3 групи:

- для обробки великих обсягів даних;
- для роботи з потоками даних;
- для паралельних обчислень;
- для обробки великих обсягів даних.

Фільтр Блума (англ. Bloom filter) – заощадлива до пам'яті ймовірнісна структура даних, призначена для перевірки приналежності елементів до множини. Запропонована Бартоном Говардом Блумом (англ. Burton Howard Bloom) в 1970 році. Для перевірки присутності елемента в множині, його слід обробити кожною з k хеш-функцій та отримати k позицій в масиві. Якщо будь-який з біт на цих позиціях дорівнює 0, то елемент гарантовано не перебуває в множині. Якби елемент належав множині, то всі біти отримали б значення 1 при додаванні елемента до множини. Якщо всі біти мають значення 1, тоді або елемент належить множині, або ж біти отримали значення 1 при додаванні інших елементів, що призведе до помилкового спрацювання. Тобто фільтр Блума не зберігає самі елементи, а створює набір хешів, які допомагають швидко перевіряти, чи є елемент у множині. Використовується така структура, як приклад, у фільтрації спаму або кешуванні даних. Приклад фільтру Блума для множини { x, y, z } зображено на рисунку 1.1, де кольорові стрілки вказують на місця в масиві біт, на які відображається кожен елемент. Елемент w не належить множині{ x, y, z }, оскільки його хеш відображається на позицію з 0. В цій ілюстрації m = 18 та k = 3 (рисунок 1.1) [1].

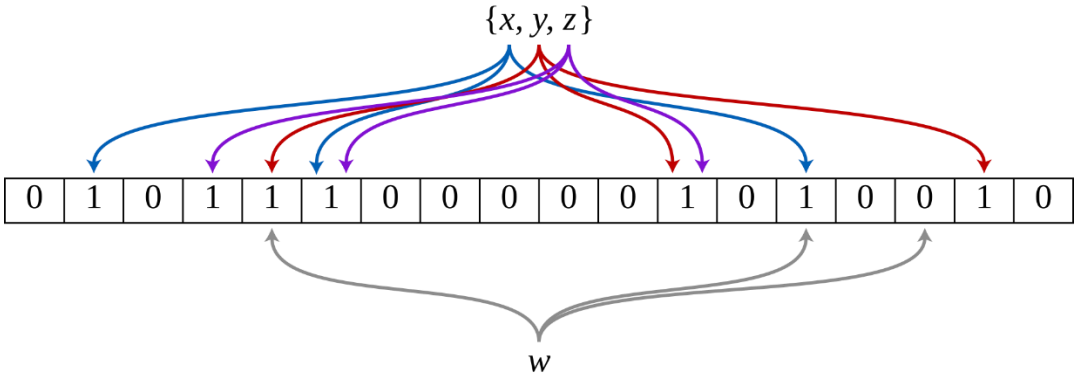

Рисунок 1 – Приклад фільтру Блума

Гіперлоглог (HyperLogLog) – це вдосконалений алгоритм для обчислення кількості унікальних елементів у великому наборі даних, який значно зменшує обсяг пам'яті, необхідний для цього завдання. Основний принцип алгоритму ґрунтується на ймовірнісному підході та хешуванні [2].

Кроки роботи HyperLogLog можуть бути представлені таким чином.

1. Хешування елементів. Кожен елемент набору даних проходить через хеш-функцію, яка створює випадкове число. Цей процес дозволяє перетворити довільні елементи у більш контрольовані числові значення, які рівномірно розподілені.

2. Розподіл на регістри. Алгоритм розбиває область значень хеш-функції на кілька підмножин, які зберігаються в масиві (набір регістрів). Наприклад, якщо використовується 16 регістрів, кожне хешоване значення потрапляє в один з них відповідно до своїх старших бітів.

3. Оцінка довжини максимального нульового префікса. Для кожного хешованого значення алгоритм підраховує кількість нулів зліва в бінарному записі (довжину префікса нулів) у певному регістрі. Це значення дає уявлення про те, наскільки «рідкісним» є конкретне значення у поточному регістрі. Чим довший нульовий префікс, тим більшою ймовірністю елемент є унікальним.

Середнє значення по регістрах. HyperLogLog обчислює середнє значення максимальних префіксів для кожного регістру та на основі цієї інформації робить оцінку кількості унікальних елементів. У цьому процесі використовується гармонічне середнє для зниження похибки [3].

Kafka – структура даних для роботи з потоками даних – зберігає повідомлення у форматі ключ-значення, які приходять від необмеженої кількості клієнтів, що називаються «продюсерами» (producers). Дані можуть бути розділені на окремі «розділи» (partitions) залежно від теми «теми» (topic). Всередині розділу, повідомлення впорядковуються за відступом (offsets) – позицією повідомлення в розділі, індексуються і зберігаються разом з позначкою часу (timestamp). Інші клієнти, які називаються «споживачами» (consumers) можуть читати повідомлення з розділів. Для потокової обробки Kafka надає Streams API, що дозволяє створювати додатки, які зчитують інформацію з Kafka і записують результати назад в Kafka. Apache Kafka також працює з зовнішіми системами потокової обробки даних, такими як Apache Apex, Apache Beam, Apache Flink, Apache Spark, Apache Storm і Apache NiFi. Kafka працює в кластері з одного або більше серверів (які ще називаються брокерами (brokers)), і розділи усіх тем (топіків) розподіляються за вузлами кластера. Додатково, розділи реплікуються декількома брокерами. Apache Kafka широко використовується в різних галузях, таких як фінансові послуги, аналітика даних, моніторинг систем і в багатьох інших сценаріях, де потрібна обробка даних у режимі реального часу [4].

Паралельна хеш-таблиця як структура для паралельних обчислень дозволяє сформувати одночасний доступ кільком потокам за допомогою хеш-функції. Паралельні хеш-таблиці – це структура паралельних індексних даних для використання в паралельних обчисленнях , які дозволяють декільком потокам ефективніше співпрацювати для обчислень серед спільних даних. Через проблеми, пов’язані з одночасним доступом, а саме спосіб і обсяг, у якому можна одночасно отримати доступ до таблиці, такі структури можуть бути реалізовані за допомогою декількох потоків та нелінійних функцій. Саме для них використовують складені типи даних для ключів і значень. Такі структури повинні формуватись відповідно до визначених вимог до вхідних та вихідних даних [5].

Висновки

Проведено аналіз структур даних для обробки великих обсягів даних, для роботи з потоками даних, для паралельних обчислень. Приведено класифікацію структур даних, описано фільтр Блума, гіперлоглог і розподілений реплікований лог Kafka. Розглянуто сфери застосування структур даних, а також визначено їх переваги та недоліки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Фільтр Блума – URL: https://uk.wikipedia.org/wiki/Фільтр_Блума (дата звернення 31.10.2024)
2. HyperLogLog – URL: https://en.wikipedia.org/wiki/HyperLogLog (дата звернення 31.10.2024)
3. HyperLogLog – URL: https://redis.io/docs/latest/develop/data-types/probabilistic/hyperloglogs/ (дата звернення 31.10.2024)
4. Apache Kafka – URL: https://uk.wikipedia.org/wiki/Apache_Kafka (дата звернення 31.10.2024)
5. Concurrent hash table – URL: https://en.wikipedia.org/wiki/Concurrent_hash_table (дата звернення 31.10.2024)

Рейда Микола Олександрович – студент 2 курсу Вінницький національний технічний університет, Вінниця, e-mail: okashnik48@gmail.com.

Сергієнко Олександр Олександрович – студент 2 курсу Вінницький національний технічний університет, Вінниця, email: shunechkiss@gmail.com.

Коваленко Олена Олексіївна – кандидат технічних наук, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: ok@vntu.edu.ua.

Reyda Mykola Oleksandrovich – 2st-year student, Vinnytsia National Technical University, Vinnytsia, e-mail: okashnik48@gmail.com.

Serhienko Oleksandr Oleksandrovich – 2st-year student Vinnytsia National Technical University, Vinnytsia, e-mail shunechkiss@gmail.com.

Olena Kovalenko – Ph.D., Associate Professor of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: ok@vntu.edu.ua.