DOI 10.31891/2307-5732-2023-327-5-42-42
УДК 637.5.02

ЗАВАЛІЙ ТАРАС
Національний університет "Львівська політехніка"
https://orcid.org/0009-0002-7544-782X
e-mail: taras.i.zavalii@lpnu.ua

ЗАДАЧІ ТА АЛГОРИТМИ ОПРАЦЮВАННЯ ПОТОКОВИХ ДАНИХ

Розглянуто основні поняття галузі аналізу даних в контексті роботи з потоками, а не масивами даних. Базові принципи і алгоритми в обох випадках ті самі, але потокові дані накладають суттєві обмеження по пам'яті і часу, вимагають застосування додаткових методів накопичення, фільтрування і попереднього опрацювання. Переважно, ці методи орієнтовані на роботу з сирими даними. У статті наведено порівняльний аналіз основних типів алгоритмів, розглянуто актуальні задачі аналізу потоків даних. Подана коротка характеристика брокера повідомлень Kafka та фреймворку Spark Streaming.

Ключові слова: потокові дані, опрацювання потоків даних, online data analysis, черги повідомлень.

ZAVALIY TARAS
Lviv Polytechnic National University

ALGORITHMS AND CHALLANGES IN STREAMING DATA PROCESSING

The basic methods and tools of data analysis in the context of data streams, rather than batches, are considered. The fundamental principles and algorithms are the same in both cases, but streaming data imposes significant constraints on memory and time, requiring additional methods for accumulation, filtering, and preprocessing. Mostly, these methods are applied to raw data, and raw data is everywhere now. We have constant streams of data in many areas, such as sports analytics, medical analytics, patient monitoring, real-time stock market analysis, website visitors' data analysis, infrastructure monitoring, predictive maintenance, not to mention various scientific research projects that gather vast amounts of data.

This paper provides a comparative analysis of the main types of algorithms and discusses current applied problems in stream processing and online data analysis. Specifically, algorithms such as Stream DBScan, DGIM, HyperLogLog, Bloom filter, and Count-Min Sketch are described and compared in the context of their application and computational complexity. A brief description of the Kafka message broker and the Spark Streaming framework is presented, though the number of tools and frameworks available now is constantly expanding. They support concepts such as windowing, event time processing, and state management, machine learning libraries, and enable advanced analytics on streaming data. They also address issues of scalability and provide the throughput for handling large volumes of data.

From a technical standpoint, two factors are equally important for streaming data analysis: the choice of the technological stack and the choice of the algorithm. It is stated that the most important task is obtaining raw streaming data, selecting the optimal analysis algorithm, and considering the specifics of the data. Another challenge to tackle in future research is combining different stream processing algorithms in the multi-stage distributed architecture to achieve a higher quality of the resulting model.

Key words: streaming data, stream processing, online data analysis, message queues.

Вступ

Потокові дані (streaming data) стали частиною нашого повсякденного життя. Інформаційні системи постійно генерують велику кількість різноманітних даних, таких як логування дій клієнтів у мобільних та веб-додатках, купівля/продаж в додатках електронної комерції, активність гравців в онлайн-іграх, повідомлення і оновлення в соціальних мережах, оновлення з фінансових бірж, а також дані з медичних та IoT пристроїв. Опрацювання потокових даних (stream processing) розглядають в контексті аналізу великих даних і протиставляють пакетному опрацюванню даних (batch processing) [1]. Пакетної обробки даних недостатньо, коли мова йде про аналіз даних у реальному часі. Більшість даних, що надходять потоком у реальному часі, потребують аналізу теж у реальному часі, або ж з низькою затримкою. Це вимагає застосування спеціальних методів і засобів аналізу потокових даних [2]. оскільки обсяг даних занадто великий для зберігання в оперативній пам'яті, а на зміни потрібно реагувати з мінімальною затримкою.

Потокові дані генеруються безперервно в джерелах даних та надходять на опрацювання поступово, окремими повідомленнями (рис. 1). Причому порядок елементів в потоці не обов'язково зберігається, для гарантованого впорядкування необхідні часові мітки в повідомленнях.

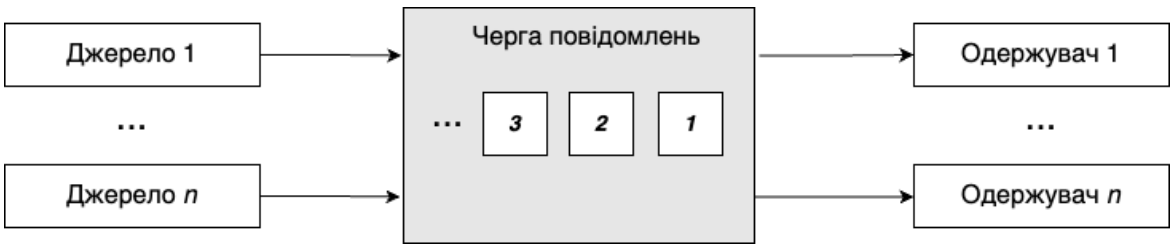

Рис. 1. Загальна схема потоку даних

Потреба в аналізі таких даних тільки зростає [3]. Наведемо приклади областей, де використання і опрацювання потокових даних є необхідним:

- спортивна аналітика, у якій рухи спортсменів (метрики) фіксуються з високою частотою [4], [5];
- медична аналітика і відстеження стану пацієнта [6], [7];
- відстеження змін на фондовому ринку в режимі реального часу [8];
- аналіз даних про відвідувачів веб-ресурсів [2], [9];
- моніторинг хмарної інфраструктури, коли потрібно реагувати на певні події в реальному часі, або ж прогнозувати значення метрик на основі попередніх значень [10];
- моніторинг транспортних засобів та промислового обладнання, для аналізу потенційних поломок та проактивної заміни запчастин (predictive maintenance) [11].

Однією з головних цілей аналізу потоку даних є розробка ефективних моделей, які можуть точно прогнозувати майбутні спостереження. Проте властиві характеристики потокових даних, такі як обсяг, швидкість, різноманітність (variety), мінливість (variability), достовірність (veracity), волатильність (volatility) і цінність, сильно обмежують доступний спектр алгоритмів опрацювання, а також ставлять певні вимоги до потоку даних, щоб забезпечити високу точність моделі. Аналіз сучасного стану досліджень у [12] показав, що питання масштабованості, відмовостійкості, послідовності, а також неоднорідності і неповноти все ще потребують подальших досліджень. Багато уваги і зусиль було спрямовано на аналіз потоків даних у реальному часі, але мало уваги приділяють стадії попередньої обробки. Наприклад, дані, отримані з IoT сенсорів, часто можуть бути “брудними” і містити значення “поза діапазоном”, нульові значення, дублікати, синтаксичні помилки, які вимагають калібрування і очищення.

До актуальних задач належить також інтеграція – коли кожен вузол розподіленої системи виконує аналіз потоків з невеликої кількості джерел, з подальшою агрегацією результатів у глобальну модель. У [13] наведено приклад такої інтеграції для алгоритму Stream-DBScan в розподіленій архітектурі Apache Flink.

Метою статті є проаналізувати сучасний стан досліджень в галузі аналізу потокових даних, зробити огляд задач та методів аналізу таких даних, що дозволить вибирати алгоритми аналізу та побудови моделей прийняття рішень.

Алгоритми аналізу потоків даних

Розглянемо особливості алгоритмів, які ітеративно застосовують до потоку вхідних даних у різних задачах та сценаріях опрацювання у реальному часі. Вони дозволяють проводити попереднє опрацювання, фільтрування, прості статистичні обчислення та обчислення у вікні, виявляти залежності між значеннями, відсутні події та помилкові дані, виявляти закономірності (trend analysis), кластери та аномалії у послідовності подій.

Алгоритми кластеризації. Враховуючи середовище реального часу, швидкість та обсяг потоків даних, алгоритми кластеризації, які застосовуються до потокових даних, повинні бути високо масштабованими. Крім того, динамічний характер даних ускладнює задання необхідної або бажаної кількості кластерів заздалегідь. Це робить методи кластеризації за розбиттям (такі як k-медіана, k-середні та k-medoid) або підходи на основі алгоритмів максимізації сподівання (expectation maximization) непридатними для аналізу даних в режимі реального часу [12] .

Алгоритми кластеризації на основі щільності (такі як DBScan, DenStream, OpticStream, FlockStream, Exclusive і Complete Clustering) на відміну від алгоритмів розбиття не вимагають апріорної кількості кластерів заздалегідь і можуть виявляти аномалії (outliers). Однак проблема з алгоритмами кластеризації на основі щільності полягає в тому, що більшість з них менш ефективні у випадку даних високої розмірності і, наприклад, не підходять для аналізу потоків соціальних мереж. А такі алгоритми як HDDStream, PreDeCon-Stream і PKS-Stream вимагають багато пам'яті [14].

Методи на основі порогових значень, ієрархічна кластеризація та інкрементальна кластеризація (онлайн-кластеризація) є більш актуальними у випадку аналізу великих даних. Було розроблено кілька онлайн-підходів до кластеризації потоків на основі порогу або інкрементальних підходів до кластеризації, таких як випадкове поле Маркова [15], сферичні K-середні онлайн [16] та конденсовані кластери. Інкрементальні підходи використовують для безперервного групування даних шляхом встановлення максимального порогу подібності (similarity threshold) між вхідним потоком і існуючими кластерами. Багато зусиль було спрямовано для підвищення ефективності алгоритмів онлайн-кластеризації, однак мало досліджень вирішують проблему встановлення порогів. Інкрементальний алгоритм встановлення порогу повинен використовувати динамічний підхід, а не покладатися на статичні значення [17], [18].

Метод ковзаючого вікна (sliding window) розбиває неперервний потік даних на вікна фіксованого розміру та виконує обчислення або аналіз для кожного вікна незалежно. Він використовує буфер найновіших значень та зсуває вікно вперед по мірі надходження нових даних. Цей алгоритм широко використовується для аналізу часових рядів – при обчисленні ковзаючих середніх, виявленні тенденцій або статистичному аналізі даних з кожного вікна. До недоліків віднесемо той факт, що алгоритм ефективний в специфічному випадку, коли розмір вікна для обчислень фіксований протягом усього циклу. Тільки тоді складність алгоритму можна зменшити до $O(n)$, де n – кількість ітерацій [19].

Алгоритм DGIM дає розв’язок задачі підрахунку у вікні, яку коротко сформулюємо так: маючи потік бітів і розмір вікна n, потрібно порахувати кількість значень "1" в останніх k бітах. Зрозуміло, що відповідь дуже проста, якщо ми можемо зберігати в пам'яті всі останні n біт. Якщо це не так, ми повинні використовувати розумніший спосіб зберігання та опрацювання даних. Алгоритм DGIM дозволяє відповісти на питання з логарифмічним обсягом пам'яті, і з контрольованою точністю. А саме, для заданої точності $1/m$

необхідний обсяг пам'яті дорівнює $O(m \cdot log(n)^2)$ [20].

Фільтрація Блума. Фільтр Блума – це ймовірнісна структура даних з 70-х, яку використовують для перевірки належності елемента множині [21], [22]. Він особливо корисний для таких завдань як виявлення дублікатів, виключення елементів і перевірки на належність множині в потокових даних великих обсягів. Фільтр Блума складається з масиву бітів та кількох хеш-функцій. Для кожного елементу в потоці обчислюється хеш-значення кілька разів, і встановлюються відповідні біти в масиві. Щоб перевірити, чи новий елемент належить множині, обчислюються його хеш-значення, і якщо всі відповідні біти співпадають, ймовірність приналежності висока. Складність такої перевірки – $O(k)$, де k – кількість хеш-функцій. Однак недоліком є ймовірність помилкової позитивної відповіді [23]. В алгоритмі countBF [24] досягнуто зменшення цієї ймовірності шляхом використання простих чисел при хешуванні.

Алгоритм Флайоле-Мартина та HyperLogLog. Однією з задач аналізу є задача підрахунку унікальних значень (cardinality estimation). Припустимо, елементи потоку вибрані з деякого універсального набору. Потрібно порахувати, скільки різних елементів з'явилося в потоці, рахуючи або від початку потоку, або від якогось часу T в минулому. Оцінити кількість унікальних значень можна шляхом хешування елементів універсальної множини в досить довгий бітовий рядок. Довжина бітового рядка повинна бути достатньою, щоб можливих результатів хеш-функції було більше, ніж елементів універсальної множини. Наприклад, 64 біт достатньо для хешування URL-адреси.

Ідея алгоритму Флайоле-Мартина (Flajolet-Martin) полягає в тому, що чим більше різних елементів присутні в потоці даних, тим більше різних хеш-значень ми отримаємо. Оскільки з кожним новим елементом з потоку ми отримуємо все більше різних хеш-значень, стає більш імовірним, що одне з цих значень буде унікальним, "особливим". Унікальною властивістю може бути те, що двійкове значення закінчується послідовністю нулів, хоча існує багато інших можливих характеристик. Кожного разу, коли ми застосовуємо хеш-функцію h до елемента потоку a, бітовий рядок $h(a)$ закінчується деякою кількістю нулів, а можливо, жодним. Це число називають довжиною хвоста для a і h. Нехай R – максимальна довжина хвоста будь-якого a побаченого досі в потоці. Тоді можна використати оцінку $2R$ для кількості унікальних елементів у потоці [25]. Недоліки алгоритму в контексті підвищення точності і швидкості, вирішено в новішому алгоритмі HyperLogLog і його варіаціях [26], [27]. Складність цього алгоритму при опрацюванні потоку з n унікальних елементів є лінійною і становить $O(n)$. Це робить його придатним для аналізу потоків великих даних.

Алгоритми узагальнення (sketching). Алгоритми узагальнення "стискають" та апроксимують потокові дані, надаючи компактні представлення даних зі збереженням важливих властивостей. Ці алгоритми, як правило, використовують обмежену кількість пам'яті для зберігання статистики, яка використовується для оцінки різних характеристик даних. Вони використовуються для різних завдань, таких як оцінювання частоти появи елементів, оцінювання квантилей, підрахунок унікальних елементів та аналіз подібності множин. Узагальнення будуються шляхом опрацювання випадкової підмножини даних та узагальнення відібраних значень. У роботі [28] досліджено статистичні властивості таких алгоритмів і стверджується, що стиснені дані можуть бути змодельовані як випадкова вибірка, таким чином це сімейство методів стиснення даних є придатне для застосування в системах логічного виведення. Зокрема, розглянуто структури (скетчі) Гауса, Адамара і Кларксона-Вудраффа, а також їх використання в однопрохідних алгоритмах узагальнення для лінійної регресії з величезними значеннями n. До прикладів алгоритмів узагальнення також належать Count-Min Sketch та t-digest [20].

Count-Min Sketch – це ймовірнісна структура даних, яка використовується у алгоритмі обчислення частоти елементів у потоці. Вона зберігає приблизну кількість кожного елемента у фіксованому об’ємі пам'яті. Count-Min Sketch складається з масиву лічильників та кількох хеш-функцій. Коли надходить елемент, обчислюють його хеш-значення, і збільшують відповідні лічильники. Для оцінки частоти появи елемента в потоці його хеш-значення обчислюються знову, і повертається мінімальне значення серед відповідних лічильників. Count-Min Sketch використовуються для таких задач як приблизний підрахунок, виявлення елементів з високою частотою (heavy hitters) та аналіз мережевого трафіку.

Розглянуті алгоритми узагальнено в таблиці 1, яка буде орієнтиром для пошуку методу аналізу залежно від поставленої задачі.

Таблиця 1

Алгоритми опрацювання потокових даних

№	Алгоритм	Складність	Задача	Джерело
1	Stream-DBScan	$O(n\log(n))$	кластеризація	[13]
2	Ковзаючого вікна	$O(n)$	статистичний аналіз	[19]
3	DGIM	$O(m\log(n)^2)$	статистичний аналіз	[20]
4	Фільтр Блума	$O(k)$	фільтрування, виявлення аномалій	[22],
5	HyperLogLog	$O(n)$	підрахунок унікальних елементів	[26], [27]
6	Count-Min Sketch	$O(\log(n))$	узагальнення, підрахунок частоти, виявлення аномалій	[20]
7	Gaussian Sketch	$O(ndk)$	узагальнення, регресійний аналіз	[28]
8	Clarkson-Woodruf Sketch	$O(nd)$		

Засоби аналізу потокових даних

Технічно, переважна більшість систем опрацювання потоків даних використовує чергу (message queue). Черги повідомлень широко застосовують в розподілених системах для зменшення залежностей між компонентами чи мікросервісами. Вони дозволяють усунути тісний зв'язок між одержувачем та відправником, динамічно додавати нових одержувачів, забезпечують гарантовану та максимально швидку відправку, отримання, та зберігання повідомлень. Системи опрацювання потоків даних мають складну архітектуру, використовують різні математичні алгоритми для опрацювання та передачі даних, тому їх розробляють спираючись на готові фреймворки і відомі патерни.

Серед всього різноманіття черг та брокерів повідомлень, таких як RabbitMQ, StormMQ, Amazon SQS, Google Pub/Sub, Azure Service Bus, RedPanda, Apache Kafka, саме остання стала лідером серед високопродуктивних платформ для потокової обробки повідомлень (подій) і увійшла в SMACK стек (Spark, Mesos, Akka, Cassandra, Kafka), який фактично є стандартом для опрацювання великих даних.

Наведемо основні переваги Apache Kafka [29]:

1. Це розподілена система з підтримкою реплікації даних.
2. Масштабоване опрацювання даних завдяки тому, що черги (topics) поділені на розділи (partitions) і підтримують паралельну обробку.
3. Всі повідомлення можуть зберігатися в постійній пам'яті після опрацювання.
4. Гарантована доставка та отримання повідомлень.
5. Наявність клієнтів для всіх мов програмування.
6. Можливість використання в системах реального часу.
7. Наявність розширення ksqlDB для зручного зберігання і вибірки даних.

На рис. 2 зображено оптимізований процес передачі даних в Kafka-брокері на системному рівні, без використання надлишкових і дороговартісних операцій копіювання між буферами прикладного додатку, операційної системи, сокетом. З оперативної пам'яті елемент потоку даних одразу копіюється в буфер мережевого інтерфейсу для передачі одержувачам.

Розглянемо приклад засобів вищого рівня, які спрощують повний цикл побудови системи опрацювання потокових даних. Фреймворк Spark Streaming є частиною популярного Apache Spark, і дозволяє організувати високопродуктивну обробку потокових даних. Дані можуть надходити із багатьох джерел, таких як Kafka, Flume, Kinesis або TCP сокетів, і можуть бути оброблені за допомогою складних алгоритмів, виражених функціями високого рівня, такими як Map, Reduce, Join та Window. Засоби Apache Spark окрім Spark Streaming включають в себе кілька інших бібліотек (Spark SQL, MLlib, GraphX) і утворюють зручну уніфіковану модель програмування [31]. Зокрема, в поєднанні з потоковими алгоритмами, такими як потоковий метод k-середніх (streaming k-means), Spark можна використовувати для прийняття рішень в реальному часі [32]. Через велику кількість технічних рішень, фреймворків та архітектур, в цьому розділі розглянуто лише такі ключові засоби як Apache Kafka та Apache Spark. Вибір конкретного технологічного стеку залежить як від задач дослідження, так і від характеристик потоку даних в предметній області, вимог до часу затримки, обсягу необхідної пам'яті тощо.

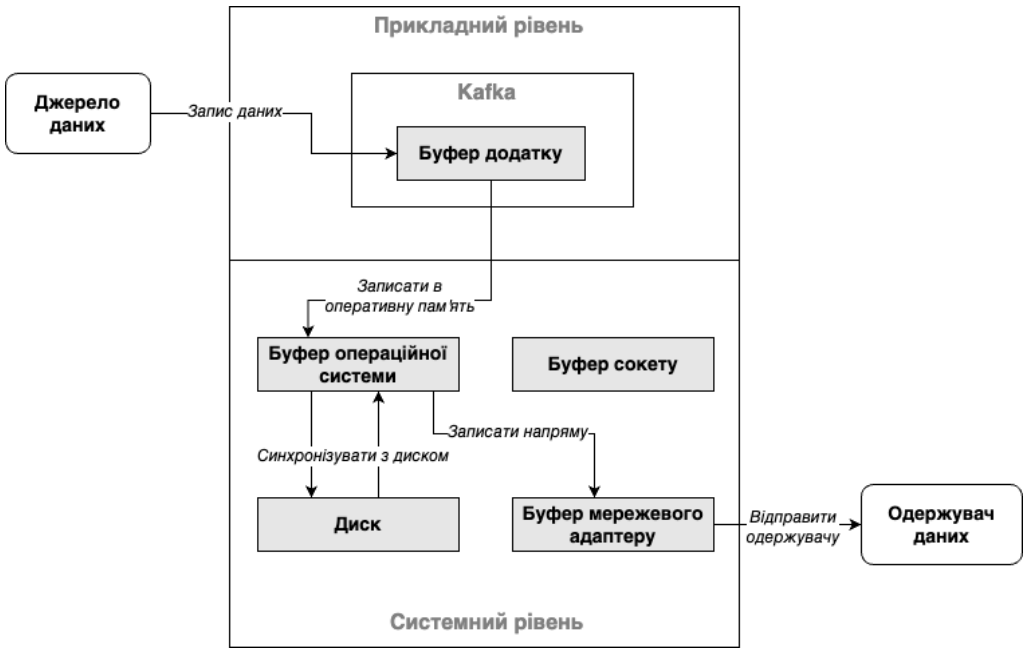

Рис. 2. Оптимальний процес передачі повідомлення в Kafka [30].

Висновки

З технічної точки зору, для аналізу потокових даних однаково важливими є два фактори – вибір технологічного стеку і вибір алгоритму аналізу. На сьогодні, доступна велика кількість інструментів, технологій, хмарних платформ для потокового опрацювання даних. Вони вирішують питання масштабованості та забезпечують пропускну здатність для великих обсягів даних. Тому цікавішою задачею є, власне, отримання сирих потокових даних, підбір оптимального алгоритму аналізу, врахування особливостей даних. Ми розглянули особливості алгоритмів потокової кластеризації, метод ковзаючого вікна, фільтри Блума, алгоритми узагальнення (sketching), DGIM, HyperLogLog. На основі здійсненого огляду, можна здійснювати постановку задачі моделювання і обирати оптимальний підхід до аналізу потокових даних великих обсягів.

Література

1. Sun D., Key Technologies for Big Data Stream Computing. / Sun D., Zhang G., Zheng W., Li K. // Big Data: Algorithms, Analytics, and Applications (1st ed.). // Chapman and Hall/CRC., – 2015, – P. 230–251.

2. Z. Qian, TimeStream: reliable stream computation in the cloud. / Qian Z., He Y., Su C. *et al.* // EuroSys '13: Eighth Eurosys Conference 2013, Prague, Czech Republic, – 2013. – P. 1–14.

3. M. Dayarathna, Recent Advancements in Event Processing. / M. Dayarathna, S. Perera, // ACM Computing Surveys, – Vol. 51, – Issue 2, – 2018, – P. 1–36.

4. How Apache Spark Feeds Real-Time Sports Analytics [Електронний ресурс] / I. Nuage // Medium. – Режим доступу: https://medium.com/@Talend/how-apache-spark-feeds-real-time-sports-analytics-4673ec871295 (дата звернення: 12.10.2023). – Назва з екрана.

5. L. Probst, Real-Time Football Analysis with StreamTeam: Demo. // *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. / New York, NY, USA: Association for Computing Machinery, – 2017, – P. 319–322.

6. A. Kharbouch, IoT and Big Data Technologies for Monitoring and Processing Real-Time Healthcare Data / A. Kharbouch *et al.*// International Journal of Distributed Systems and Technologies, – Vol. 10, – 2019, – P. 17–30.

7. M. Singh, A Novel Intelligent Hybrid Optimized Analytics and Streaming Engine for Medical Big Data. / M. Singh *et al.* // Computational and Mathematical Methods in Medicine, – Vol. 2022, – 2022. – P. 1–11.

8. L. Maguluri, An Efficient Stock Market Trend Prediction Using the Real-Time Stock Technical Data and Stock Social Media Data. / L. Maguluri, R. Rengaswamy // International Journal of Intelligent Engineering and Systems, – Vol. 13, – No. 4, – 2020, – P. 316–332.

9. B. Lohrmann, Elastic Stream Processing with Latency Guarantees. / B. Lohrmann, P. Janacik, O. Kao // IEEE 35th International Conference on Distributed Computing Systems, – 2015, – P. 399–410.

10. Y. Qin, Faster Multidimensional Data Queries on Infrastructure Monitoring Systems. / Y. Qin and G. Guzun // Big Data Research, – Vol. 27, – 2022, – P. 1–20.

11. Machine Learning Techniques for Predictive Maintenance [Електронний ресурс] / R. Alwis, S. Perera, S. Penchikala // InfoQ. – Режим доступу: https://www.infoq.com/articles/machine-learning-techniques-predictive-maintenance (дата звернення: 13.10.2023). – Назва з екрана.

12. T. Kolajo, Big data stream analysis: a systematic literature review. / T. Kolajo, O. Daramola, and A. Adebiyi // Journal of Big Data, – Vol. 6, – No. 47, – 2019, – P. 1–30.

13. C. Mu, Stream-DBSCAN: A Streaming Distributed Clustering Model for Water Quality Monitoring. / C. Mu, Y. Hou, J. Zhao *et al*. // Applied Sciences, – Vol. 13(9), – 2023, – P. 5408–5424.

14. K. S. S. Reddy, A review on density-based clustering algorithms for big data analysis. / K. S. S. Reddy and C. S. Bindu // International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), – 2017, – P. 123–130.

15. L. Pelkowitz, A continuous relaxation labeling algorithm for Markov random fields. // IEEE Transactions on Systems, Man, and Cybernetics, – Vol. 20, – No. 3, – 1990, – P. 709-715.

16. S. Zhong, Efficient streaming text clustering. // Neural Networks, – Vol. 18, – No. 5, – 2005, – P. 790–798.

17. J. D. Deng, Online Outlier Detection of Energy Data Streams Using Incremental and Kernel PCA Algorithms. // IEEE 16th International Conference on Data Mining Workshops (ICDMW), – 2016, – P. 390–397.

18. R. H. Rakib, Fast Clustering of Short Text Streams Using Efficient Cluster Indexing and Dynamic Similarity Thresholds. // AA'20, September 29-October 2, Virtual Event, CA, USA, – 2020. – P. 1–7.

19. CS49: Data Stream Algorithms Lecture Notes. [Електронний ресурс] / A. Chakrabarti // Dartmouth College – 2011, – Режим доступу: https://www.cs.dartmouth.edu/~ac/Teach/CS49-Fall11/Notes/lecnotes.pdf (дата звернення: 23.10.2023). – Назва з екрана.

20. J. Leskovec, Mining of Massive Datasets (3rd ed.). / J. Leskovec, A. Rajaraman, J. D. Ullman // Cambridge University Press. – 2020, – 565 P.

21. B. H. Bloom, Space/time trade-offs in hash coding with allowable errors. // Communications of the ACM, – Vol. 13, – No. 7, – 1970, – P. 422–426.

22. M. Mitzenmacher, Network Applications of Bloom Filters: A Survey. / M. Mitzenmacher, A. Broder // *Internet Mathematics Journal*, – Vol. 1, – No. 4, – 2004, – P. 485–509.

23. R. Patgiri, Is Bloom Filter a Bad Choice for Security and Privacy? / R. Patgiri, S. Nayak, N. B. Muppalaneni // International Conference on Information Networking (ICOIN), – 2021, – P. 648–653.

24. S. Nayak, CountBF: A General-purpose High Accuracy and Space Efficient Counting Bloom Filter. [Електронний ресурс] / S. Nayak, R. Patgiri // arXiv, – 2021, – Режим доступу: http://arxiv.org/abs/2106.04364 (дата звернення: 23.10.2023). – Назва з екрана.

25. A. Psaltis, Streaming Data // Manning Publications, – 2017, – 216 P.

26. P. Flajolet, HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. / P. Flajolet, É. Fusy, O. Gandouet, F. Meunier // Discrete Mathematics & Theoretical Computer Science (DMTCS), – 2007, – P. 127–146.

27. S. Heule, HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm. / S. Heule, M. Nunkesser, A. Hall // Proceedings of the EDBT Conference, – 2013. – P. 683–692.

28. D. Ahfock, Statistical properties of sketching algorithms. / D. Ahfock, W. J. Astle, S. Richardson // arXiv, – 2019, – Режим доступу: https://arxiv.org/pdf/1706.03665.pdf (дата звернення: 23.10.2023). – Назва з екрана.

29. Починаємо роботу з Apache Kafka. Частина I. [Електронний ресурс] / С. Моренець // DOU. – Режим доступу: https://dou.ua/forums/topic/39363/ (дата звернення: 12.10.2023). – Назва з екрана.

30. Why is Kafka fast? [Електронний ресурс] / A. Xu // – Режим доступу: https://blog.bytebytego.com/p/why-is-kafka-fast (дата звернення: 12.10.2023). – Назва з екрана.

31. Грищук Т. В. Сучасні методи обробки потокових даних / Т. В. Грищук, В. В. Стецюк // Матеріали XLVIII науково-технічної конференції підрозділів ВНТУ, Вінниця, 13-15 березня 2019 р. – 2019. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2019/paper/view/7443.

32. S. Saxena, Practical Real-time Data Processing and Analytics. / S. Saxena, S. Gupta // Packt Publishing, – 2017, – 360 P.

References

1. Sun D., Key Technologies for Big Data Stream Computing. / Sun D., Zhang G., Zheng W., Li K. // Big Data: Algorithms, Analytics, and Applications (1st ed.). // Chapman and Hall/CRC., – 2015, – P. 230–251.

2. Z. Qian, TimeStream: reliable stream computation in the cloud. / Qian Z., He Y., Su C. et al. // EuroSys '13: Eighth Eurosys Conference 2013, Prague, Czech Republic, – 2013. – P. 1–14.

3. M. Dayarathna, Recent Advancements in Event Processing. / M. Dayarathna, S. Perera, // ACM Computing Surveys, – Vol. 51, – Issue 2, – 2018, – P. 1–36.

4. How Apache Spark Feeds Real-Time Sports Analytics [Electronic resource] / I. Nuage // Medium. – Mode of access: https://medium.com/@Talend/how-apache-spark-feeds-real-time-sports-analytics-4673ec871295 (date of access: 12.10.2023). – Title from screen.

5. L. Probst, Real-Time Football Analysis with StreamTeam: Demo. // Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems. / New York, NY, USA: Association for Computing Machinery, – 2017, – P. 319–322.

6. Kharbouch, IoT and Big Data Technologies for Monitoring and Processing Real-Time Healthcare Data / A. Kharbouch et al.// International Journal of Distributed Systems and Technologies, – Vol. 10, – 2019, – P. 17–30.

7. M. Singh, A Novel Intelligent Hybrid Optimized Analytics and Streaming Engine for Medical Big Data. / M. Singh et al. // Computational and Mathematical Methods in Medicine, – Vol. 2022, – 2022. – P. 1–11.

8. L. Maguluri, An Efficient Stock Market Trend Prediction Using the Real-Time Stock Technical Data and Stock Social Media Data. / L. Maguluri, R. Rengaswamy // International Journal of Intelligent Engineering and Systems, – Vol. 13, – No. 4, – 2020, – P. 316–332.

9. Lohrmann, Elastic Stream Processing with Latency Guarantees. / B. Lohrmann, P. Janacik, O. Kao // IEEE 35th International Conference on Distributed Computing Systems, – 2015, – P. 399–410.

10. Y. Qin, Faster Multidimensional Data Queries on Infrastructure Monitoring Systems. / Y. Qin and G. Guzun // Big Data Research, – Vol. 27, – 2022, – P. 1–20.

11. Machine Learning Techniques for Predictive Maintenance [Electronic resource] / R. Alwis, S. Perera, S. Penchikala // InfoQ. – Mode of access: https://www.infoq.com/articles/machine-learning-techniques-predictive-maintenance (date of access: 13.10.2023). – Title from screen.

12. T. Kolajo, Big data stream analysis: a systematic literature review. / T. Kolajo, O. Daramola, and A. Adebiyi // Journal of Big Data, – Vol. 6, – No. 47, – 2019, – P. 1–30.

13. Mu, Stream-DBSCAN: A Streaming Distributed Clustering Model for Water Quality Monitoring. / C. Mu, Y. Hou, J. Zhao et al. // Applied Sciences, – Vol. 13(9), – 2023, – P. 5408–5424.

14. K. S. S. Reddy, A review on density-based clustering algorithms for big data analysis. / K. S. S. Reddy and C. S. Bindu // International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), – 2017, – P. 123–130.

15. L. Pelkowitz, A continuous relaxation labeling algorithm for Markov random fields. // IEEE Transactions on Systems, Man, and Cybernetics, – Vol. 20, – No. 3, – 1990, – P. 709-715.

16. S. Zhong, Efficient streaming text clustering. // Neural Networks, – Vol. 18, – No. 5, – 2005, – P. 790–798.

17. J. D. Deng, Online Outlier Detection of Energy Data Streams Using Incremental and Kernel PCA Algorithms. // IEEE 16th International Conference on Data Mining Workshops (ICDMW), – 2016, – P. 390–397.

18. R. H. Rakib, Fast Clustering of Short Text Streams Using Efficient Cluster Indexing and Dynamic Similarity Thresholds. // AA'20, September 29-October 2, Virtual Event, CA, USA, – 2020. – P. 1–7.

19. CS49: Data Stream Algorithms Lecture Notes. [Electronic resource] / A. Chakrabarti // Dartmouth College – 2011, – Mode of access: https://www.cs.dartmouth.edu/~ac/Teach/CS49-Fall11/Notes/lecnotes.pdf (date of access: 23.10.2023). – Title from screen.

20. J. Leskovec, Mining of Massive Datasets (3rd ed.). / J. Leskovec, A. Rajaraman, J. D. Ullman // Cambridge University Press. – 2020, – 565 P.

21. B. H. Bloom, Space/time trade-offs in hash coding with allowable errors. // Communications of the ACM, – Vol. 13, – No. 7, – 1970, – P. 422–426.

22. M. Mitzenmacher, Network Applications of Bloom Filters: A Survey. / M. Mitzenmacher, A. Broder // Internet Mathematics Journal, – Vol. 1, – No. 4, – 2004, – P. 485–509.

23. R. Patgiri, Is Bloom Filter a Bad Choice for Security and Privacy? / R. Patgiri, S. Nayak, N. B. Muppalaneni // International Conference on Information Networking (ICOIN), – 2021, – P. 648–653.

24. S. Nayak, CountBF: A General-purpose High Accuracy and Space Efficient Counting Bloom Filter. [Electronic resource] / S. Nayak, R. Patgiri // arXiv, – 2021, – Mode of access: http://arxiv.org/abs/2106.04364 (date of access: 23.10.2023). – Title from screen.

25. Psaltis, Streaming Data // Manning Publications, – 2017, – 216 P.

26. P. Flajolet, HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. / P. Flajolet, É. Fusy, O. Gandouet, F. Meunier // Discrete Mathematics & Theoretical Computer Science (DMTCS), – 2007, – P. 127–146.

27. S. Heule, HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm. / S. Heule, M. Nunkesser, A. Hall // Proceedings of the EDBT Conference, – 2013. – P. 683–692.

28. Ahfock, Statistical properties of sketching algorithms. / D. Ahfock, W. J. Astle, S. Richardson // arXiv, – 2019, – Режим доступу: https://arxiv.org/pdf/1706.03665.pdf (date of access: 23.10.2023). – Title from screen.

29. Pochynayemo robotu z Apache Kafka. Chastyna I. [Electronic resource] / S. Morenets // DOU. – Mode of access: https://dou.ua/forums/topic/39363/ (date of access: 12.10.2023). – Title from screen.

30. Why is Kafka fast? [Electronic resource] / A. Xu // – Mode of access: https://blog.bytebytego.com/p/why-is-kafka-fast (date of access: 12.10.2023). – Title from screen.

31. Hryshchuk T. Suchasni metody obrobky potokovykh danykh / Hryshchuk T., Stecyuk V. // Materialy XLVIII naukovo-tehnichnoi konferentsii pidrozdiliv VNTU, Vinnytsia, – 2019. – Mode of access: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2019/paper/view/7443.

32. S. Saxena, Practical Real-time Data Processing and Analytics. / S. Saxena, S. Gupta // Packt Publishing, – 2017, – 360 P.