Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

УДК 004.75 DOI: https://doi.org/10.30837/0135-1710.2026.189.165

Кириченко І. В., Демчук В. Г., Луценко В. В.

РОЗРОБЛЕННЯ Й ДОСЛІДЖЕННЯ ПІДХОДУ ДО ВИЯВЛЕННЯ АНОМАЛІЙ У ПОТОКОВИХ КОНВЕЄРАХ ДАНИХ НА ОСНОВІ ОПЕРАЦІЙНОЇ ТЕЛЕМЕТРІЇ

Об’єктом вивчення є процес операційного моніторингу, діагностики й забезпечення функціональної надійності розподілених потокових конвеєрів даних у режимі реального часу; **предметом** – методи й алгоритми автоматизованого виявлення аномалій у потокових системах на основі аналізу багатовимірних часових рядів операційної телеметрії (метрик продуктивності та використання ресурсів) інфраструктурних компонентів Apache Spark Structured Streaming і Apache Kafka. **Мета дослідження** полягає в розробленні та експериментальній валідації легковагового підходу проактивного виявлення аномалій у швидкісних конвеєрах даних, яка функціонує виключно на основі метапоказників інфраструктури без ресурсомісткої інспекції корисного навантаження, для мінімізації часу реакції на інциденти й усунення додаткових затримок в обробленні даних. **Досягнуті результати.** У процесі дослідження розроблено архітектуру системи моніторингу й сформовано 14-вимірний вектор простору ознак, який передбачає нормалізовані значення системних метрик, швидкості їх зміни (градієнти) й синтетичні безрозмірні коефіцієнти (ефективність оброблення, нормований лаг). Для класифікації станів системи застосовано ансамблевий алгоритм машинного навчання. Експериментальне моделювання типових збоїв (сплески затримки, падіння пропускної здатності, аномальний лаг) на хмарному кластері AWS підтвердило високу ефективність підходу. Запропонований багатовимірний підхід підвищив точність виявлення інцидентів (F1-score) з 0.62 (показник класичного Rule-based-методу на основі статичних порогів) до 0.92 за рівня хибних спрацювань (FPR) лише 0.8%. Середній час виявлення аномалії (MTTD) було скорочено зі 115 с до 25 с. Обчислювальні накладні витрати мікросервісу моніторингу становили менше ніж 1.5% процесорного часу кластера. **Висновки.** Експериментально доведено, що аналіз багатовимірної операційної телеметрії за допомогою методів машинного навчання є високоефективним проксі-індикатором "здоров’я" конвеєрів даних. Запропонований підхід успішно розв’язує питання важкої валідації, притаманної традиційним інструментам перевірки якості даних, і повністю відповідає парадигмі сучасної Shift-Left-архітектури. Рішення забезпечує глибоку спостережуваність із нульовим впливом на продуктивність, є надійною першою лінією проактивного захисту й створює технологічне підґрунтя для реалізації механізмів автоматичного самовідновлення інфраструктури.

Ключові слова: потокове оброблення даних; конвеєри даних; виявлення аномалій; операційна телеметрія; машинне навчання; Isolation Forest; Apache Kafka; Apache Spark; Shift-Lef-архітектура; спостережуваність систем.

Вступ

У сучасному ландшафті цифрової трансформації значення аналітики в реальному часі перейшла з категорії конкурентної переваги в статус фундаментальної вимоги для виживання бізнесу. Здатність організацій обробляти, аналізувати й приймати рішення на основі даних у момент їх генерації (протягом мілісекунд або секунд) визначає ефективність критичних операційних процесів. Відповідно до показників IDC за 2025 р., понад 63% корпоративних сценаріїв використання потребують оброблення інформації протягом лічених хвилин, щоб вона залишалася корисною для бізнесу [1]. У таких доменах, як фінансовий моніторинг, затримка в аналізі транзакції може призвести до незворотних втрат: шахрайські операції мають блокуватися до завершення авторизації, що вимагає наскрізної затримки менше ніж 100 мс. Аналогічно у сферах динамічного ціноутворення для ритейлу або сервісів замовлення поїздок, а також у персоналізації клієнтського досвіду в реальному часі будь-яке відставання аналітичного конвеєра безпосередньо корелює із втраченим доходом або деградацією лояльності користувачів [2].

Однак забезпечення надійності та якості даних у таких швидкісних потоках стикається з серйозним технологічним бар’єром. Традиційні методи перевірки якості даних, подані такими інструментами, як Great Expectations, AWS Deequ, Soda або Apache Griffin, виявляються надто "важкими" для сучасних стрімів. Основна проблема полягає в тому, що ці фреймворки базуються на парадигмі сканування самих даних – вони розраховують статистичні профілі, перевіряють розподіли й валідують кожне поле безпосередньо в потоці або в мікробатчах. Такий підхід неминуче збільшує затримку оброблення, оскільки вимагає додаткових обчислювальних ресурсів і доступу

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

до вмісту повідомлень, що часто стає bottleneck для всього конвеєра. У системах, де критичним є кожен мілісекундний інтервал, упровадження важких валідаторів вмісту нівелює переваги потокового оброблення [1, 3].

Аналіз останніх досліджень і публікацій

Розвиток сучасних цифрових екосистем призвів до безпрецедентного зростання обсягів даних, що генеруються в режимі реального часу. Для оброблення таких масивів інформації галузевим стандартом стало використання розподілених систем, де Apache Kafka є магістраллю для транспортування повідомлень, а Apache Spark – основним рушієм для їх оброблення й аналізу. Проте складність і динамічність таких архітектур роблять їх вразливими до аномалій на різних рівнях конвеєра: від інфраструктурних збоїв до логічних помилок у потоковому обробленні. У цьому розділі систематизовано наявні підходи, інструменти й виявлені обмеження, визначено дослідницьку прогалину, яку покликаний заповнити запропонований підхід.

Класичні статистичні методи – Z-score та ковзне середнє – залишаються базовими інструментами виявлення відхилень у метричних потоках. *Zhong et al.* (2023) [5] у систематичному огляді методів виявлення аномалій у часових рядах у контексті AIOps розглядають ковзне вікно з метою обчислення локальних параметрів розподілу як стандартний підхід для виявлення аномалій у KPI-метриках хмарних сервісів. Автори зазначають, що Z-score й похідні від нього методи добре адаптуються до поступового дрейфу даних завдяки ковзному обчисленню середнього й стандартного відхилення, проте фундаментально обмежені одновимірним аналізом і не беруть до уваги кореляції між метриками різних компонентів розподіленої системи. Огляд демонструє, що базові статистичні детектори оперують статичними порогами й не здатні моделювати складні поведінкові патерни, властиві гетерогенним стрімінговим конвеєрам.

Аналогічний підхід із використанням алгоритму CUSUM (*Cumulative Sum Control Chart*) для виявлення змін у метриках інфраструктури хмарних і граничних середовищ досліджено в роботі *Skaperas et al.* (2024) [6]. Автори експериментально оцінюють CUSUM і баєсівські детектори точок змін на реальних тестових стендах граничних обчислень і виявляють принципові компроміси: CUSUM демонструє обчислювальну ефективність і придатність для онлайн-виявлення, проте вимагає індивідуального налаштування параметрів для кожного типу метрики. Це унеможливлює автоматичне масштабування на гетерогенні конвеєри даних із динамічними навантаженнями, властивими для Apache Kafka і Spark.

У масштабному порівняльному дослідженні *Schmidl et al.* (2022) [7] оцінено алгоритми виявлення аномалій у часових рядах на понад 1 тис. наборах даних, зокрема з методами на основі прогнозування за алгоритмом Хольта–Вінтерса. Попри задовільну точність на регулярних і сезонних робочих навантаженнях, дослідження демонструє, що прогнозні методи суттєво деградують за умови різких нерегулярних змін, властивих для подійно-орієнтованих архітектур стрімінгових систем. Крім того, аналіз проводиться переважно в офлайн-режимі, що принципово обмежує застосовність методів у сценаріях виявлення аномалій у реальному часі.

Значного поширення набули методи машинного навчання без учителя. Алгоритм Isolation Forest застосовувався в низці сучасних досліджень завдяки своїй обчислювальній ефективності та стійкості до шуму. *Islam et al.* (2024) [8] запропонували промислове дослідження аномалій на реальних хмарних системах IBM Cloud, упроваджуючи методи машинного навчання, зокрема з Isolation Forest, до масиву телеметричних даних із багатьма ознаками. Ключовим виявленим обмеженням є необхідність повторного перенавчання або перелаштування моделі за умови зміни операційних патернів – типової ситуації для промислових Spark-кластерів під час планових і позапланових змін конфігурації.

Метод LOF *(Local Outlier Factor)* та його потокові варіанти розглянули *Zamanzadeh Darban et al.* (2024) [9] у ґрунтовному огляді методів глибокого навчання для виявлення аномалій у часових рядах. Автори систематизують як класичні методи (разом із LOF та його інкрементальними модифікаціями), так і підходи глибокого навчання, аналізуючи їх застосовність до потокових сценаріїв. Огляд демонструє, що методи на основі густини, зокрема LOF, стикаються з проблемами масштабованості в багатовимірних просторах ознак і не моделюють кореляційних залежностей між метриками різних рівнів системи – критично важливий аспект для архітектур типу Kafka + Spark.

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

Автоенкодери та їх варіанти (VAE) активно застосовуються для виявлення аномалій у багатовимірних метричних потоках. *Jacob* і *Diao* (2025) [10] пропонують підхід DIVAD (*Domain-Invariant VAE for Anomaly Detection*) – архітектуру варіаційного автоенкодера з механізмом інваріантності до доменного зсуву – й оцінюють його на бенчмарку Exathlon, побудованому на реальних метриках Spark-кластерів, DIVAD досягає покращення пікового F1-показника на 20% порівняно з найкращими конкурентами, проте вимагає значних обчислювальних ресурсів для онлайн-інференсу й не тестувався на метриках, притаманних для Kafka-брокерів. *Wang et al*. (2024) [11] розробляють FCVAE – частотно-розширений варіаційний автоенкодер для моніторингу вебсистем – і демонструють переваги частотного підходу для метрик із гетерогенними часовими патернами, але визнають обмеження в одночасному моделюванні довгоперіодичних і короткоперіодичних тенденцій.

Apache Kafka є де-факто стандартом побудови потокових конвеєрів даних, і дослідження, присвячені його операційним властивостям, активно розвиваються. *Mayer et al.* (2020) [12] пропонують спеціалізовану архітектуру моніторингу для Apache Kafka на основі JMX-метрик і проводять комплексний бенчмарк продуктивності брокера, вимірюючи пропускну здатність і затримку під різними конфігураціями. У дослідженні автори документують набір ключових метрик брокера, необхідних для вчасного виявлення деградації, та наголошують на ризику перетворення Kafka на вузьке місце архітектури в разі недостатнього моніторингу. Утім запропонована система обмежується порогами, налаштованими вручну, і не бере до уваги контекст downstream-споживачів (зокрема Spark Structured Streaming), що є критичним для розуміння першопричини відхилення.

Питання латентності та пропускної здатності конвеєра Kafka – Spark як джерела операційних метрик досліджено в роботі *Fedorovych et al.* (2024) [13]. Автори фіксують ключові метрики конвеєра: пропускну здатність, затримку на рівні брокера й воркерів Spark, вплив кількості Kafka-партицій на продуктивність Spark-задач. Результати демонструють, що мікробатч-режим забезпечує більш стабільну затримку (≈ 197 мс із Kafka-джерелом) порівняно з безперервним режимом у сценаріях високого навантаження, а перевищення пропускної здатності Kafka над можливостями Spark-воркерів є індикатором накопичення затримок. Однак підхід є описовим і не пропонує алгоритму автоматичного виявлення аномалій.

Дослідження операційних метрик Apache Spark концентруються переважно на характеристиці та бенчмаркуванні конвеєрів. *Jacob et al.* (2021) [14] систематично документують набір метрик Spark-кластера в межах бенчмарку Exathlon: драйвер надає 243 метрики Spark UI (затримку планування, статистику потокових даних), кожен виконавець – 140 метрик (JVM heap memory, GC overhead, task duration, shuffle read / write). Бенчмарк охоплює шість типів аномальних подій (resource contention, process failures, misbehaving inputs) і забезпечує еталонний набір даних для оцінювання алгоритмів. Проте детекція проводиться в офлайн-режимі на завершених трасах виконання, а не в реальному часі в процесі роботи конвеєра.

Метрика тривалості мікробатчів і scheduling delay в Spark Structured Streaming як індикатори навантаженості конвеєра детально схарактеризовані в роботі *Fedorovych et al.* (2024) [13]. Результати бенчмарку демонструють, що стійке перевищення затримки оброблення над тривалістю мікробатч-інтервалу є надійним провісником деградації конвеєра, а кількість партицій Kafka безпосередньо визначає здатність Spark рівномірно розподіляти навантаження між виконавцями. Підхід є реактивним і не зважає на upstream-стан Kafka, що не дає змоги відрізнити перевантаження споживача від затримок на рівні продюсера.

Аномально високий час GC є передвісником події OOM (*OutOfMemory*). Бенчмарк Exathlon фіксує GC overhead серед ключових метрик виконавців, що корелюють з аномальними подіями в Spark-кластері, зокрема з вичерпанням пам’яті та resource contention [14]. Результати демонструють, що метрики пам’яті JVM у поєднанні з shuffle-метриками формують інформативний сигнатурний набір для класифікації типів аномалій у потоковому обробленні, проте методологія залишається ретроспективною без компоненти автоматичного виявлення аномалій у реальному часі.

Для реалізації методик виявлення аномалій дослідники використовують екосистему інструментів, що забезпечує збір, зберігання та візуалізацію даних у режимі реального часу.

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

Prometheus є найбільш поширеною платформою збору метрик у середовищах cloud-native. *Pragathi et al.* (2024) [15] досліджують розгортання моніторингового стека Prometheus + Grafana + Node Exporter у Kubernetes-кластері й порівнюють можливості Prometheus і NetData. Prometheus демонструє найкращу інтеграцію з Kubernetes і підтримує Pull-модель збору метрик з інтервалом 15 с, проте базова система сповіщень (Alertmanager) оперує лише порогами й не підтримує складні поведінкові патерни або ML-основані правила виявлення аномалій.

Grafana як платформа візуалізації та побудови дашбордів розглядається в тій самій роботі [15] в контексті операційного моніторингу розподіленої інфраструктури. Автори систематизують найкращі практики побудови дашбордів і наголошують на принциповому обмеженні: Grafana призначена для відображення метрик, а не для автономного виявлення аномалій. Наявні механізми алертингу Grafana оперують порогами або простими арифметичними виразами PromQL і не моделюють кореляцій між метриками різних підсистем.

Інтеграція Prometheus і Kafka у межах єдиного конвеєра моніторингу описана в роботі *Fedorovych et al.* (2024) [13]. Автори будують пайплайн, де Kafka водночас транспортує оброблені дані та приймає результати вимірювань (Kafka sink). Це дає змогу збирати наскрізні метрики затримки від брокера до Spark-виконавця. Незважаючи на практичну цінність підходу, система залишається реактивною: метрики збираються й використовуються для діагностики продуктивності, але не для автономного виявлення аномалій через ML-основані або адаптивні алгоритми.

Apache Spark має вбудовану систему метрик, а бенчмарк Exathlon документує повний набір метрик виконавців. *Jacob et al.* (2021) [14] описують 140 метрик на рівні executor – heap memory, GC overhead, task duration, shuffle metrics – та оцінюють їх придатність для виявлення аномалій на еталонному наборі даних. Дослідження зупиняється на рівні збору й характеристики метрик; компонента автоматичного виявлення аномалій у режимі реального часу відсутня.

Інтегровані рішення, що одночасно охоплюють Kafka і Spark в єдиному контексті виявлення аномалій, у літературі висвітлено недостатньо. Найближчою до цієї теми є робота *Jacob* і *Diao* (2025) [10], де запропоновано метод DIVAD, оцінений на бенчмарку Exathlon (дані Spark-кластера). Автори зосереджують увагу виключно на метриках Spark-виконавців, тоді як Kafka-брокер поза контекстом розгляду, а аналіз метрик брокера й кореляція зі станом Spark-задач відсутні. *Fedorovych et al.* (2024) аналізують конвеєр Kafka – Spark комплексно з погляду продуктивності, але не пропонують компоненти виявлення аномалій.

Підхід до кореляційного аналізу метрик у мікросервісних архітектурах, релевантний для розглядуваної задачі, описали *Pham et al.* (2024) [16]. Автори проводять масштабну оцінку 21 методу визначення першопричини на основі каузальних графів між метриками різних сервісів, тестуючи їх на двох загальноприйнятих бенчмарках мікросервісів. Результати демонструють, що жоден метод не домінує в усіх сценаріях, а ефективність на синтетичних датасетах не відтворює реальну поведінку у виробничих системах. Метод не адаптований до особливостей stateful-обчислень Spark і брокерської семантики Kafka.

Унаслідок проведеного огляду літературних джерел визначено суттєву прогалину: незважаючи на значну кількість робіт з виявлення аномалій у часових рядах і операційного моніторингу стрімінгових систем, жодне з розглянутих досліджень не пропонує комплексного підходу, що водночас:

– об’єднує метрики двох рівнів потокового конвеєра – брокера повідомлень (Kafka: consumer lag, throughput, partition metrics) і обчислювального рушія (Spark: executor metrics, job duration, shuffle metrics) – в єдиній моделі виявлення аномалій;

– забезпечує виявлення аномалій у режимі реального часу (low-latency inference) без необхідності накопичення великої передісторії або ресурсоємного перенавчання;

– зважає на крос-компонентні кореляції між метриками Kafka і Spark для розрізнення причинно-наслідкових ланцюжків деградацій (upstream bottleneck vs. downstream processing overload);

– адаптується до динамічної зміни патернів навантаження без ручного перелаштування порогів.

Статистичні підходи [5–7] обмежені статичними порогами, переважно офлайн-обчисленнями й одновимірним аналізом. ML-методи [8, 9] не тестувалися на специфічних комбінованих метриках Kafka / Spark і вимагають стабільних патернів навантаження. VAE-основані методи [10, 11]

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

демонструють значні обчислювальні вимоги для онлайн-інференсу й не охоплюють крос-системних метрик. У роботах з моніторингу Kafka [12, 13] і Spark [14] кожну платформу розглянуто переважно ізольовано. Інструментальні рішення [15] обмежуються порогами й візуалізацією без автономного виявлення аномалій і без кореляційного аналізу між рівнями конвеєра. Підходи до аналізу першопричин не адаптовані до особливостей об'єднаного середовища Kafka + Spark [16].

Отже, запропонований у цій роботі підхід, що ґрунтується на спільному аналізі операційних метрик Spark і Kafka за допомогою адаптивних алгоритмів виявлення аномалій у режимі реального часу, є оригінальним рішенням, що безпосередньо закриває виявлену прогалину.

Мета й постановка завдань

У межах цього дослідження під якістю даних у потоковому конвеєрі розуміється здатність системи забезпечувати вчасну, повну й безперервну доставку й оброблення потокових повідомлень без втрати, деградації або критичного зростання затримки.

Об'єктом вивчення є моніторинг "здоров'я" та функціональної цілісності конвеєрів передачі даних за допомогою аналізу метапоказників або операційної телеметрії самої системи. Замість того, щоб заглядати всередину кожного пакета даних, запропоновано оцінювати стан системи за непрямими ознаками – поведінкою інфраструктурних компонентів Apache Spark і Apache Kafka. Телеметрія, що передбачає метрики пропускної здатності, затримки, інтенсивності запитів і використання ресурсів, формує унікальний цифровий відбиток нормальної роботи пайплайну. Будь-яка аномалія в даних – чи то різка зміна схеми, чи то сплеск обсягу, чи то деградація якості на джерелі – неминуче викликає відлуння в операційних метриках системи оброблення [4].

Метою цього дослідження є розроблення легкого підходу виявлення аномалій, який здатний функціонувати в режимі реального часу, аналізуючи виключно телеметрію Spark і Kafka без утручання в самі дані. Такий підхід дає змогу ідентифікувати проблеми в конвеєрах із мінімальними накладними витратами, забезпечуючи високу спостережуваність і швидку реакцію на інциденти без збільшення затримки оброблення.

Матеріали й методи дослідження

Apache Kafka – це розподілена платформа потокової передачі подій із відкритим вихідним кодом, розроблена інженерами LinkedIn і передана під управління Apache Software Foundation (2011 р.). Нині Kafka є стандартом у побудові платформ потокової передачі даних масштабу підприємства, охоплюючи понад 80% компаній зі списку Fortune 100 [4].

Ключова концепція Kafka основана на моделі журналу подій: записи зберігаються у впорядкованих незмінних послідовностях – топіках, які розбиваються на партиції. Такий підхід забезпечує паралельне оброблення завдяки розподілу навантаження між партиціями, масштабованість унаслідок горизонтального нарощування кількості брокерів, відмовостійкість через реплікацію партицій між брокерами, а також гарантію черговості повідомлень у межах однієї партиції [4].

З погляду операційного моніторингу Kafka надає набір ключових JMX-метрик: UnderReplicatedPartitions (кількість партицій із реплікацією нижчою від заданого рівня), consumer_group_lag (відставання групи споживачів від позиції producer), RequestHandlerAvgIdlePercent (відсоток незайнятості обробників запитів), NetworkProcessorAvgIdlePercent і OfflinePartitionsCount. Аномальні значення перелічених метрик є першими сигналами про деградацію продуктивності конвеєра [17].

Загальну архітектуру потокового конвеєра на основі Apache Kafka й Apache Spark Structured Streaming зображено на рис. 1 [18].

Перший стовпець (ліворуч) – джерела даних:

1. комірка 1: IoT / Sensors (~100M events/day) – Дані телеметрії та показники з фізичних пристроїв і датчиків, що генеруються у великих обсягах;
2. комірка 2: Фінансові події (транзакції, FIX) – Інформація про банківські перекази, платежі або торгові операції на біржах;
3. комірка 3: Веблоги (разом) – Записи про дії користувачів на вебсайтах (історія кліків, перегляди сторінок, помилки);
4. комірка 4: Мобільні додатки – застосунки – Аналітика поведінки користувачів, краш-репорти та події з мобільних програм;
5. комірка 5: Системні метрики (сервери, мережа) – Технічні показники навантаження та стану інфраструктури (CPU, пам'ять, мережевий трафік).

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

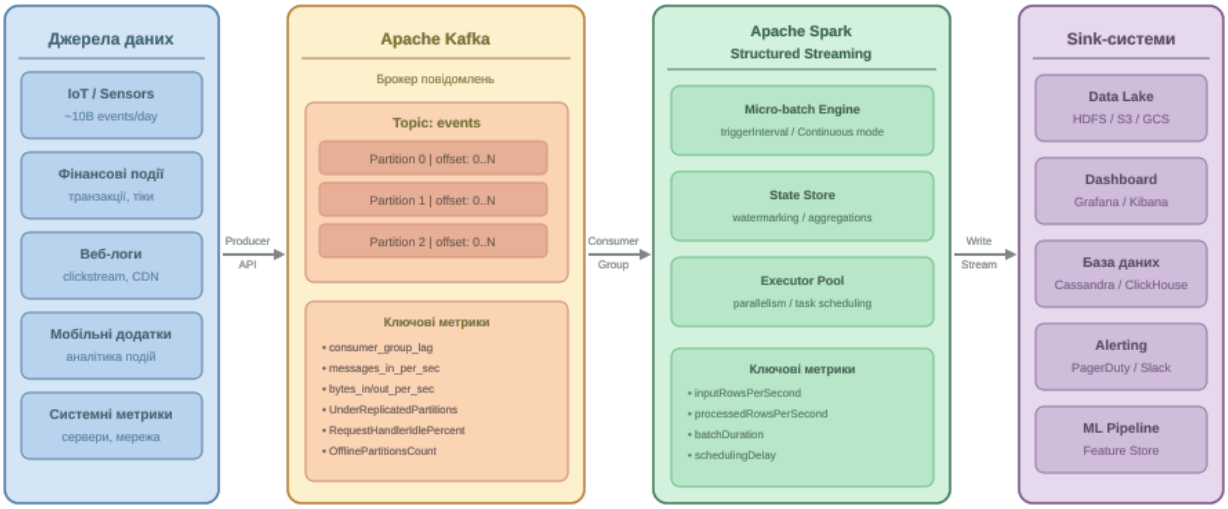

Рис. 1. Загальна архітектура потокового конвеєра на основі Apache Kafka й Apache Spark Structured Streaming

Другий стовпець – Apache Kafka:

1. комірка 1: Topic: events (Partition 0, 1, 2) – Логічний канал (топік), у який записуються всі події, розділений на партиції для забезпечення паралельного читання та масштабованості;

2. комірка 2: Ключові метрики – Набір найважливіших показників для моніторингу стану брокера та швидкості обміну повідомленнями (відставання консьюмерів, пропускна здатність, статус реплікації партицій).

Третій стовпець – Apache Spark Structured Streaming:

1. комірка 1: Micro-Batch Engine (trigger interval / continuous mode) – Рушій, що обробляє безперервний потік даних невеликими порціями через задані інтервали або в режимі реального часу;

2. комірка 2: State Store (watermarking / aggregations) – Механізм збереження проміжних результатів, який дозволяє робити агрегації за часовими вікнами та коректно обробляти події, що запізнилися (watermarking);

3. комірка 3: Executor Pool (parallelism / task scheduling) – Пул робочих процесів, які розподіляють між собою завдання для паралельної та швидкої обробки даних;

4. комірка 4: Основні метрики – Показники продуктивності самої обробки: швидкість читання та запису рядків за секунду, тривалість батчу та затримка планування.

Четвертий стовпець (праворуч) – Sink-система:

1. комірка 1: Data Lake (HDFS / S3 / GCS) – Сховище для довгострокового збереження великих обсягів оброблених (або сирих) даних у вигляді файлів;

2. комірка 2: Dashboard (Grafana / Kibana) – Інструменти для побудови графіків та візуалізації аналітики в режимі реального часу;

3. комірка 3: База даних (Cassandra / ClickHouse) – Системи керування базами даних (часто NoSQL або колоночні), оптимізовані для швидкого доступу та аналітичних запитів;

4. комірка 4: Alerting (PagerDuty / Slack) – Системи автоматичного сповіщення команд про критичні відхилення, інциденти або досягнення певних тригерів;

5. комірка 5: ML Pipeline (Feature Store) – Інфраструктура машинного навчання, куди дані надходять як розраховані ознаки для тренування або роботи моделей.

Сучасний бізнес дедалі більше покладається на безперервні потоки даних для підтримки прийняття рішень. Перехід від традиційного пакетного оброблення до потокового зумовлений потребою в миттєвій реакції на події. У фінансовому секторі це полягає в упровадженні систем реального часу, як-от FedNow у США або SEPA Instant у Європі, де розрахунки й кліринг відбуваються миттєво, що вимагає відповідної інфраструктури моніторингу з нульовою толерантністю до затримок [17].

Реальна цінність аналітики в реальному часі полягає в переході від реактивного стилю управління ("що сталося вчора?") до проактивного ("що відбувається зараз і як реагувати?"). Проте надійність цих систем прямо залежить від стабільності конвеєрів даних. Якщо пайплайн починає працювати некоректно або затримувати дані, бізнес отримує хибну інформацію або застарілі індикатори, що може

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

спричинити катастрофічні наслідки в автоматизованих системах прийняття рішень [4, 17, 18].

Проблема традиційних DQ-інструментів, як-от Great Expectations або Soda, полягає в їх архітектурній орієнтації на статику. Вони розглядають дані як набір записів у таблиці, до яких можна застосувати SQL-запити для перевірки повноти, унікальності або валідності.

Для виконання перевірки за допомогою Great Expectations необхідно [19, 20]:

1) отримати вибірку або повний набір даних із потоку;

2) завантажити ці дані в пам’ять процесу валідації (Pandas, Spark DataFrame);

3) виконати набір правил (Expectations), що часто передбачає обчислювально складні операції, зокрема розрахунок квантилів або перевірку на відповідність регулярним виразам.

У контексті Spark Structured Streaming, де мікробатчі можуть запускатися кожні кілька сотень мілісекунд, додавання такого циклу валідації до кожного батчу призводить до експоненціального зростання часу оброблення. Навіть якщо інструмент використовує механізми push-down (передачу запитів на рівень бази даних або Spark-двигуна), саме сканування рядків створює навантаження на введення / виведення та процесор, що конкурує з основною бізнес-логікою за ресурси кластера.

Іншою критичною вадою традиційних DQ-інструментів є їх "сліпота" до стану інфраструктури. Вони можуть повідомити, що в потоці з’явилися порожні значення, але вони не знають, чи є це наслідком помилки в джерелі даних, чи результатом деградації продуктивності одного з брокерів Kafka, що призвело до часткової втрати повідомлень. Такий розрив між якістю даних та "здоров’ям" системи змушує команди інженерів витрачати години на ручну кореляцію метрик із різних систем моніторингу [19].

У великих розподілених системах, як-от Spark і Kafka, телеметрія забезпечує спостережуваність, що дає змогу розуміти внутрішній стан системи на основі її зовнішніх виходів.

Перевага використання телеметрії для виявлення аномалій полягає в її легкості. Метрики – це зазвичай числові значення (лічильники, гістограми, таймери), збір яких практично не впливає на продуктивність основної системи. Наприклад, відстеження швидкості вхідних повідомлень у Spark або лагу споживача в Kafka не потребує читання вмісту повідомлень – ці дані вже доступні в пам’яті керівних процесів.

У цій роботі звернено увагу на чотири категорії телеметрії, які в сукупності дають змогу ідентифікувати аномалії даних без їх безпосереднього аналізу.

- Пропускна здатність: кількість записів і обсяг даних у байтах за одиницю часу.
- Затримка: час оброблення батчу, час перебування повідомлення в черзі, наскрізна затримка.
- Споживання ресурсів: завантаження CPU, використання пам’яті, мережева активність під час оброблення конкретних обсягів даних.
- Стан черг і відставання: кількість необроблених повідомлень у Kafka.

Другий порядок інсайтів полягає в тому, що аномалії даних (наприклад, раптова зміна формату з JSON на XML або поява аномально довгих рядків) завжди проявляються як зміни в часі оброблення (addBatch duration у Spark) або зміні коефіцієнта стиснення в Kafka. Отже, системна телеметрія слугує проксі-індикатором якості даних.

В умовах промислової експлуатації потокові конвеєри неминуче стикаються з різноманітними відхиленнями від нормального режиму роботи. Під аномалією в контексті потокового конвеєра розуміємо статистично значуще відхилення значень однієї або декількох операційних метрик системи від їх базової лінії, яке свідчить про деградацію продуктивності або порушення цілісності даних.

Аномалії в потокових системах мають складний, багатопараметричний характер: жодна окрема метрика зазвичай не дає повної картини про стан системи. Наприклад, зростання batchDuration у Spark може бути спричинене як збільшенням вхідного навантаження (що позначиться на зростанні inputRowsPerSecond), так і внутрішніми проблемами виконавця (Executor) – GC-паузами або вичерпанням пам’яті, що не відображено в метриках Kafka. Класифікацію основних типів аномалій подано в табл. 1 [21].

Характеристичну картину аномалій в операційних метриках потокового конвеєра проілюстровано на рис. 2, де показано два типових відхилення: сплеск затримки й падіння пропускної здатності з одночасним зростанням споживчого лагу.

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

Таблиця 1. *Класифікація аномалій у потокових конвеєрах даних*

Тип аномалії	Прояв / симптоми	Причини	Вплив на систему
Сплески затримки (Latency Spikes)	Різке зростання batchDuration, schedulingDelay > норми	Перевантаження Executor, GC-паузи JVM, мережеві збої	Порушення SLA, деградація якості сервісу, накопичення лагу
Падіння пропускної здатності (Throughput Drop)	processedRowsPerSecond < inputRowsPerSecond, зростання черги	Back-pressure, вичерпання ресурсів Executor, back-pressure	Затримка даних, переповнення буферів Kafka, consumer lag
Аномальний лаг споживача (Consumer Lag)	consumer_group_lag перевищує допустимий поріг	Повільний Spark executor, різке зростання вхідного навантаження	Критична затримка реакції системи, переповнення Kafka-топіків
Втрата повідомлень (Message Loss)	Розрив між producer_offset та consumer_offset	Некоректна конфігурація acks, збій брокера Kafka	Неповні або некоректні дані в downstream-системах

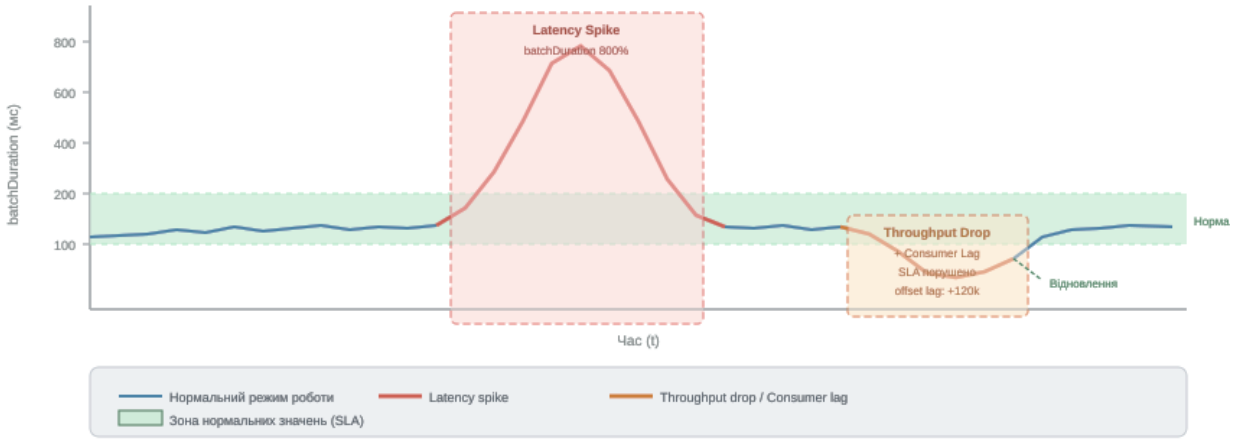

Рис. 2. Характеристичні аномалії в операційних метриках потокового конвеєра даних

Аномалії в потокових конвеєрах безпосередньо впливають на операційні показники організацій, що їх використовують. За інформацією *Gartner* [2], простій критичної інформаційної системи обходиться підприємствам у середньому $5 600 USD за хвилину або понад $300 000 за годину. У контексті фінансових платіжних систем навіть короткочасне збільшення латентності оброблення транзакцій з 100 мс до 1 с призводить до конверсійних втрат у розмірі 7% [3].

Порушення угод про рівень обслуговування (SLA) у потокових системах має двоякий характер. По-перше, технічний SLA містить гарантії щодо затримки (наприклад, p99-латентність < 200 мс), пропускної здатності (наприклад, мінімум 500000 подій/с) та доступності системи (наприклад, 99.9% час роботи). По-друге, бізнес SLA – гарантії щодо вчасності та повноти аналітичних звітів, сповіщень і бізнес-рішень, що генеруються на основі потокових даних.

Раннє виявлення аномалій (проактивний моніторинг) дає змогу скоротити MTTD (*Mean Time To Detect*) і MTTR (*Mean Time To Recover*), що безпосередньо корелює зі зниженням сукупних витрат на ліквідацію наслідків інцидентів і підтримання рівня клієнтської задоволеності.

Підхід виявлення аномалій у потокових конвеєрах даних

Запропонований підхід реалізує конвеєрну архітектуру з п’яти послідовних етапів: потокова передача даних через Apache Kafka, оброблення й агрегація метрик у Apache Spark Streaming, збір і попереднє оброблення телеметрії модулем Metric Collector, формування вектора ознак у блоці Feature Engineering і виявлення аномалій алгоритмом Isolation Forest із подальшою генерацією діагностичного сповіщення (рис. 3).

Принципова відмінність від наявних рішень полягає у використанні виключно внутрішніх метрик самої обчислювальної інфраструктури – без залучення зовнішніх агентів якості даних, розмічених навчальних наборів або апріорних знань про структуру аномалій [19, 21].

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

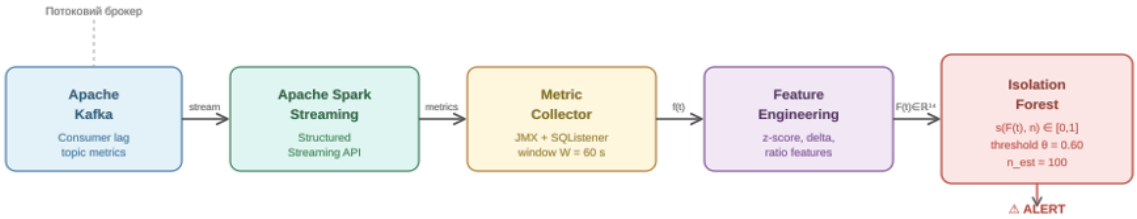

Рис. 3. Виявлення аномалій за допомогою конвеєрної архітектури

Метрики збираються через два незалежних канали. По-перше, метрики Apache Kafka отримуються через JMX Exporter (порт 9309) у форматі Prometheus з інтервалом 15 с; ключовою метрикою є avgOffsetsBehindLatest – затримка споживача, який є інтегральним показником відставання оброблення від надходження даних [12]. По-друге, метрики Apache Spark Structured Streaming надходять через StreamingQueryListener API: метод onQueryProgress() викликається після завершення кожного мікропакета й передає об'єкт StreamingQueryProgress із повним набором статистик поточного циклу [13]. Системні метрики CPU та Memory збираються агентом node_exporter (Prometheus) з того самого інтервалу [15].

Критерії відбору метрик:

– доступність через стандартні інтерфейси без додаткової інструментації;

– чутливість до характерних режимів відмови;

– взаємна інформативність за умови кореляційного аналізу.

Детальна характеристика кожної метрики подана в табл. 2 [14, 22].

Таблиця 2. *Набір операційних метрик системи виявлення аномалій*

Позначення	Джерело	Фізичний зміст і значення у виявленні аномалій	Тип аномалії, що виявляється
inputRowsPerSecond	Spark SQListener	Швидкість надходження рядків з Kafka у Spark Streaming. Раптовий спад сигналізує про зупинку Producer або розрив топіка.	Message loss, Producer failure
processedRowsPerSecond	Spark SQListener	Фактична швидкість оброблення. Якщо $x_2 < x_1$ тривало, система накопичує чергу – ознака back-pressure.	Back-pressure, Executor failure
numInputRows	Spark SQListener	Кількість рядків у поточному мікропакеті. Різкий стрибок вказує на burst-навантаження.	Load burst, skew
batchDuration [мс]	Spark SQListener	Тривалість оброблення мікропакета. Перевищення тривалості тригерного інтервалу призводить до накопичення черги.	Latency spike, GC pause
triggerExecution [мс]	Spark durationMs	Час виконання тригера планувальника. Зростання відбиває затримку планування або перевантаження Driver.	Driver overload
avgOffsetsBehindLatest	Kafka consumer JMX	Середній Kafka consumer lag (відставання споживача від останнього offset). Зростання понад 5000 – критичний сигнал.	Consumer lag, throughput drop
CPU usage [%]	System metrics	Завантаженість CPU вузлів кластера. Стале значення > 85% вказує на ресурсне голодування Executor.	CPU overload
Memory usage [%]	System metrics	Рівень використання пам'яті. Перевищення > 90% призводить до GC-паузи й неминучого зростання batchDuration.	JVM GC spike, OOM

Система оперує вектором із восьми базових операційних метрик $n = 8$, що охоплюють три виміри поведінки конвеєра: продуктивність оброблення (метрики Spark Streaming), стан споживача (метрика Kafka) й ресурсне споживання (системні метрики).

Ключовою мотивацією для комбінування метрик Spark і Kafka є те, що жодна окрема метрика не несе достатньої інформації для однозначної ідентифікації аномалії. Наприклад, зростання batchDuration (x_4) може бути зумовлене як GC-паузою JVM (супроводжується зростанням Memory usage), так і back-pressure через перевантаження Kafka Consumer (супроводжується зростанням x_6). Розрізнення цих сценаріїв вимагає одночасного аналізу декількох метрик, що обґрунтовує необхідність багатовимірного підходу [10, 16].

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

Формування вектора ознак і підготовка даних має подання у вигляді багатовимірного часового ряду. Потік операційних метрик моделюється як дискретний багатовимірний часовий ряд із фіксованим кроком $\Delta t = 15$ с. У кожний момент часу t стан конвеєра описується вектором спостережень:

$$X_{(t)} = \left[x_1(t), x_2(t), \ldots, x_n(t) \right]^T \in \mathbb{R}^n, \; t \in \{1,2,\ldots,T\}, \quad (1)$$

де $n = 8$ – кількість метрик. Відповідно, за T кроків формується матриця спостережень $X \in \mathbb{R}^{n \times T}$, яка слугує вхідними даними для подальшого оброблення [10].

Вектор ознак. Для кожного моменту t формується вектор ознак:

$$F_{(t)} = \left[\bar{x}_1, \ldots, \bar{x}_n, \Delta x_1, \ldots, \Delta x_n, \eta(t), \lambda(t) \right]^T \in \mathbb{R}^d, \quad d = 2n + 2 = 18, \quad (2)$$

де $\bar{x}_i$ – нормалізовані значення; Δx_i – першопохідні (швидкість зміни); η – коефіцієнт ефективності оброблення; λ – нормований лаг споживача. Розмірність вектора $d = 2n + 2 = 18$ [9].

Z-score нормалізація. Оскільки метрики мають суттєво різні масштаби й одиниці вимірювання, застосовується z-score-нормалізація за рухомим baseline-вікном тривалістю 24 год:

$$\bar{x}_i(t) = \frac{x_i(t) - \mu_i}{\sigma_i}, \quad x_i \in (-\infty, +\infty), \quad (3)$$

де μ_i та σ_i – вибіркові середнє та стандартне відхилення метрики x_i, обчислені за baseline-вікном. Нормалізація усуває ефект різномасштабності й дає змогу Isolation Forest рівноважно розглядати всі виміри простору ознак. Параметри μ_i, σ_i оновлюються кожні 6 год за умови відсутності зафіксованих аномалій у відповідний проміжок часу [10].

Ознаки першої похідної (delta features). Поряд зі значеннями метрик система обчислює швидкість їх зміни:

$$\Delta x_i(t) = x_i(t) - x_i(t-1), \; t \geq 2. \quad (4)$$

Ознаки $\Delta x_i(t)$ є чутливими до градієнту процесу й дають змогу виявляти поступові деградації на ранніх стадіях, зокрема повільне зростання batchDuration або затримання протягом кількох годин, яке не перевищує абсолютних порогів, але демонструє стійку позитивну тенденцію [7].

Ознаки співвідношень. Для моделювання структурних залежностей між метриками додаються дві безрозмірні ознаки. Коефіцієнт ефективності оброблення $\eta(t)$ визначає відношення фактичної продуктивності до вхідного навантаження:

$$\eta(t) = \frac{processedRowsPerSecond(t)}{inputRowsPerSecond(t)}, \quad \eta \in (0,1]. \quad (5)$$

Значення $\eta(t) \approx 1.0$ відповідає нормальному режиму; $\eta(t) < 0.7$ є операційним визначенням back-pressure – стану, за якого Spark не встигає обробляти вхідний потік і черга Kafka починає зростати [22].

Нормований Kafka lag $\lambda(t)$ впроваджується для порівняння величини відставання з поточним вхідним потоком:

$$\lambda(t) = \frac{avgOffsetsBehindLatest(t)}{inputRowsPerSecond(t) + \varepsilon}, \quad \varepsilon \to 0. \quad (6)$$

Висока величина $\lambda(t)$ за одночасно низького $\eta(t)$ є сильним сигналом системного перевантаження, тоді як високий $\lambda(t)$ у разі нормального $\eta(t)$ може вказувати на короткочасний burst або неправильне партиціонування [13].

Для виявлення аномалій застосовується алгоритм Isolation Forest (IF) – ансамблевий метод навчання без учителя, що ґрунтується на принципі ізоляції.

Обґрунтування вибору IF:

– не потребує розміченого навчального набору;

– має лінійну обчислювальну складність $O(n \cdot t)$;

– є стійким до "прокляття розмірності" порівняно з методами на основі відстані (LOF, kNN);

– здатний виявляти як точкові аномалії, так і контекстні аномалії у просторі ознак $\mathbb{R}^{18}$ [8].

Принцип роботи Isolation Forest: ансамбль із T_{est} випадкових дерев ізоляції будується способом рекурсивного розбиття простору ознак уздовж випадково обраної осі та випадково обраного порогового значення. Аномальні спостереження, розташовані в розріджених ділянках простору, потрапляють у листові вузли за значно меншу кількість розбиттів (мала глибина ізоляції $h(x)$), ніж нормальні спостереження в щільних кластерах (велика $h(x)$). Оцінка аномальності обчислюється як

$$s(x,n) = 2^{-\frac{E[h(x)]}{c(n)}}, \quad c(n) = 2H(n-1) - \frac{2(n-1)}{n}, \quad (7)$$

де $H(i)$ – i-те гармонічне число; $E[h(x)]$ – середня глибина ізоляції в ансамблі.

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

Значення $s(x,\ n)\to 1$ відповідає аномальному спостереженню (мала середня глибина ізоляції), $s(x,\ n)\to 0.5$ – невизначеності, $s(x,\ n)\to 0$ – нормальному спостереженню. На рис. 4 показано типовий розподіл anomaly score для нормальних та аномальних спостережень у задачі моніторингу потокового конвеєра [23].

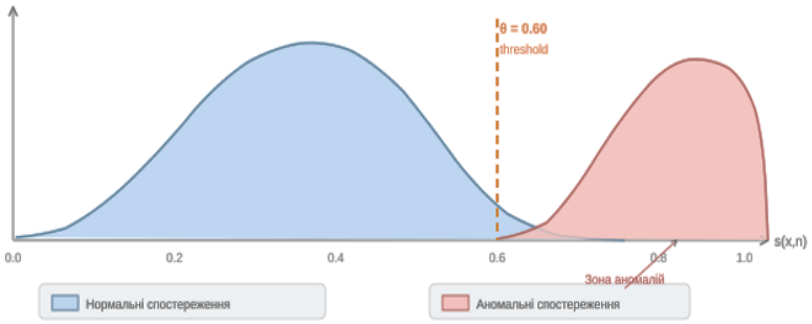

Рис. 4. Розподіл anomaly score Isolation Forest для нормальних і аномальних спостережень; вертикальна пунктирна лінія позначає поріг $\theta = 0.60$

Параметри навчання: $n\_estimators = 100$ (кількість дерев ансамблю, достатня для стабілізації оцінки дисперсії), $contamination = 0.01$ (апріорна частка аномалій у baseline, що відповідає реальним даним продуктивних систем [23]), $max\_samples = 256$ (розмір субвибірки для кожного дерева, що забезпечує різноманітність ансамблю). Модель навчається впродовж перших семи діб роботи конвеєра (warm-up period) і перенавчається щотижня в офпік-вікні (02:00–04:00) [8].

Бінарне рішення щодо аномальності знімку метрик у момент t сформулюється через порогову функцію:

$$A_t = \begin{cases} 1, & \text{аномалія, якщо } s\left(F(t), n\right) > \theta, \\ 0, & \text{норма, інакше,} \end{cases} \qquad (8)$$

де θ – поріг аномальності.

Значення $\theta = 0.60$ обрано як компроміс між хибнопозитивним рівнем і рівнем пропуску аномалій (false negative rate), верифікований на анотованому наборі тестів із 42 інцидентами трьох класів (latency spike, throughput drop, consumer lag burst). Налаштування θ для конкретного середовища розгортання виконано за методом ROC-аналізу на основі оцінки з кутом нахилу кривої: $\theta^* = \text{argmax}\left(Recall - \alpha \cdot FPR\right)$, де $\alpha = 2.0$ позначає операційну вартість хибного спрацювання [6].

Рівень серйозності аномалії визначається лінійно залежно від перевищення порогу: $severity = \left(s\left(F(t),\ n\right) - \theta\right) / \left(1 - \theta\right) \in [0,\ 1]$ за умови $A_t = 1$. Значення $severity < 0.3$ визначається як WARNING, $severity \geq 0.3$ як CRITICAL, що відповідає різному регламенту реагування операційної команди [15].

Для зниження частоти хибнопозитивних спрацювань у моменти штатних перезапусків конвеєра або планових пікових навантажень упроваджується механізм мінімального часу аномалії: сигнал генерується лише за умови $A_t = 1$ протягом двох або більше послідовних знімків (тобто ≥ 30 с). Такий підхід дає змогу фільтрувати короткочасні артефакти в значеннях метрик без втрати чутливості до стійких аномальних режимів [6].

У табл. 3 подано покроковий опис алгоритму системи виявлення аномалій. Алгоритм виконується у вигляді потокового мікросервісу з фіксованим циклом опитування $\Delta t = 15$ с і затримка від виникнення аномалії до генерації алерту не перевищує 30 с (один або два цикли залежно від тривалості аномалії) [21].

1. Latency spike – це аномалія типу "сплеск затримки", яка визначається різким зростанням x_4 (batchDuration) за відносно стабільного вхідного потоку x_1. У просторі ознак цей патерн виражено у великих значеннях $\bar{x}_4$ та Δx_4 за умови помірних значень $\bar{x}_6$. Зростання batchDuration понад trigger_interval призводить до того, що наступний мікропакет починається із затримкою, внаслідок чого Kafka consumer lag $\bar{x}_6$ починає накопичуватися – формується ланцюгова реакція деградації. Isolation

ISSN 3083-7715 (online) *Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)*
ISSN 3083-7650 (0135-1710) (print) *Management Information System and Devices. 2026. No. 2 (189)*

Forest ідентифікує такий патерн через нетипову комбінацію $(\bar{x}_4, \Delta x_4, \bar{x}_6)$, яка відсутня в baseline [14].

2. Back-pressure (throughput drop). Цей патерн визначається зниженням $\eta(t)$ нижче ніж 0.7 за умови одночасного зростання $\lambda(t)$. Spark Structured Streaming за замовчуванням обмежує швидкість споживання з Kafka через параметр maxOffsetsPerTrigger: якщо обчислювальна потужність Executor вичерпана, processedRowsPerSecond (x_2) падає нижче, ніж inputRowsPerSecond (x_1), і черга Kafka починає накопичуватися. У просторі ознак цей режим відповідає одночасно малому $\eta(t)$, великому $\lambda(t)$ і помірному зростанню $\bar{x}_4$ [22].

3. CPU overload. Тривале перевантаження CPU $(\bar{x}_7 > 85\%)$ безпосередньо впливає на продуктивність JVM і може спричинити GC-паузи, виражені в стрибкоподібному зростанні batchDuration. Особлива властивість: одночасне зростання $\bar{x}_7$ і $\bar{x}_4$ за відносно стабільного $\lambda(t)$ [14].

4. Consumer lag burst – аномальний сплеск лагу споживача. Цей тип аномалії виявляється у швидкому зростанні avgOffsetsBehindLatest / consumer_group_lag за умов, коли швидкість надходження повідомлень перевищує здатність Spark-конвеєра їх обробляти [14].

Без системних метрик цей сценарій важко відрізнити від аномалії типу skewed partitioning, де зростання batchDuration пов'язане з нерівномірним розподілом задач між Executor, а не з ресурсним голодуванням.

Таблиця 3. *Покроковий алгоритм системи виявлення аномалій*

Крок	Назва	Операція	Деталі реалізації
1	Збір метрик	Отримати $X(t)$ з Kafka JMX та Spark StreamingQueryListener	Інтервал збору 15 с; Prometheus scrape + onQueryProgress callback; зберігати в pandas DataFrame із timestamp-індексом
2	Sliding window	Агрегувати $X(t)$ у вікні $W = 60$ с	rolling(window=4, min_periods=4).agg(['mean','std','max']); крок = 15 с → 4 знімки на вікно
3	Нормалізація	Обчислити $\bar{x}_i(t) = (x_i - \mu_i)/\sigma_i$ по baseline 24 год	sklearn.preprocessing.StandardScaler, fit на перших 24 год; часткове дооновлення кожні 6 год
4	Похідні ознаки	Обчислити $\Delta x_i(t) = x_i(t) - x_i(t-1)$	df.diff(1); для першого кроку $\Delta x_i = 0$
5	Ratio-ознаки	$\eta(t) = x_2/x_1$; $\lambda(t) = x_6/(x_1 + \varepsilon)$	np.divide з epsilon=1e-6; кліпувати $\eta \in [0, 1]$
6	Вектор ознак	$F(t) = [\bar{x}_1 \dots \bar{x}_6, \Delta x_1 \dots \Delta x_6, \eta, \lambda] \in \mathbb{R}^{14}$	pd.concat по осі стовпців; dtype float32
7	Навчання IF	Навчити IsolationForest на baseline F_0 (7 діб)	$n\_estimators = 100$, $contamination = 0.01$, $max\_samples = 256$; joblib.dump для збереження моделі
8	Scoring	$s(t) = model.score\_samples(F(t))$	sklearn повертає від'ємне; перетворення: $s\_pos = -s + 0.5$; $s\_pos \in [0,1]$
9	Рішення	$A_t = 1$, якщо $s(t) > \theta = 0.60$, інакше 0	Емпіричний поріг θ; налаштування через precision-recall на annotated subset
10	Алерт	Генерувати сповіщення з діагностикою	Логувати аномалію; Grafana annotation API; $severity = f(s(t) - \theta)$

Експеримент

Для емпіричної валідації запропонованого підходу розгорнуто ізольований тестовий конвеєр даних у хмарному середовищі Amazon Web Services (AWS), що імітує типову промислову архітектуру потокового оброблення в парадигмі Shift-Left. Інфраструктура передбачала:

- Apache Kafka (v.3.4.0): керований кластер із 3 брокерів (тип інстансів m5.xlarge), 1 топік transactions_stream із 12 партиціями та фактором реплікації 3;

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

• Apache Spark (v.3.4.1): кластер під управлінням YARN, що містив 1 Driver-вузол і 3 Worker-вузли (тип r5.xlarge, 4 vCPU, 32 GB RAM кожен). Мікробатч тригер (trigger interval) застосунку Structured Streaming було налаштовано на 500 мс;

• система моніторингу та інференсу: Prometheus для збору JMX і Node Exporter метрик (інтервал $\Delta t = 15$ с). Модуль розрахунку ознак і алгоритм Isolation Forest було розгорнуто як окремий мікросервіс на базі Python (бібліотека scikit-learn).

Джерелом даних був синтетичний генератор подій, який емулював потік фінансових транзакцій (формат JSON, середній розмір повідомлення становив 1.2 КБ). Базове навантаження було налаштовано на рівні 25000 повідомлень за секунду з нормально розподіленою дисперсією ±15%, що імітувало природні коливання трафіку.

Оскільки в промислових системах аномалії є відносно рідкісними подіями, для об'єктивного оцінювання алгоритму експеримент було поділено на дві фази. Перша фаза (Warm-up period) тривала сім діб: конвеєр працював у нормальному режимі без збоїв для накопичення "чистих" baseline-даних і навчання ансамблю Isolation Forest відповідно до методології.

Друга фаза (Testing phase) тривала 72 год (три доби), протягом яких у систему було штучно інжектовано 45 аномальних інцидентів (по 15 кожного класу за інформацією з табл. 1).

1. Latency Spikes (сплески затримки): імітувалися способом короткочасного утилізування процесорного часу (до 100% CPU) на Executor-вузлах за допомогою утиліти stress-ng та ініціації примусових GC-пауз.

2. Throughput Drop (Back-pressure): імітувалися впорскуванням "важких" повідомлень (вкладені масиви великого розміру), що збільшували час десеріалізації в Spark, не змінюючи в цьому разі вхідну швидкість потоку (x_1) .

3. Consumer Lag Burst (аномальний лаг): імітувалися мікросплесками генерації даних до 70000 подій/с на 3–5 хв, що значно перевищувало обчислювальну ємність кластера.

Для оцінювання якості застосовувалися класичні метрики: Precision (точність), Recall (повнота), F1-Score та FPR (*False Positive Rate*). Для визначення операційної ефективності вимірювався MTTD (*Mean Time To Detect*) – середній час від фактичної ін'єкції аномалії до генерації алерту $(A_t = 1)$.

Запропонована багатовимірна модель (Multivariate Isolation Forest) порівнювалася з класичним підходом (Rule-based Baseline), що використовував статичні пороги моніторингу (алерт генерувався, якщо CPU > 90%, batchDuration > 2000 мс або consumer_lag > 5000 протягом 30 с).

Результати

Експериментальне тестування підтвердило гіпотезу про те, що операційна телеметрія дає змогу ефективно ідентифікувати деградацію потокового конвеєра даних. За умови заданого порогу аномальності $\theta = 0.60$ запропонована система успішно виявила 42 з 45 штучно згенерованих інцидентів. Зведені результати порівняння підходів виявлення аномалій подано в табл. 4.

Таблиця 4. *Ефективність виявлення аномалій: порівняння запропонованого підходу з базовим (Rule-based)*

Підхід виявлення аномалій	Precision	Recall	F1-Score	FPR (хибні спрацювання)	Середній час виявлення (MTTD)
Rule-based (статичні пороги)	0.76	0.53	0.62	4.2%	115 с
Multivariate IF (запропонований)	0.91	0.93	0.92	0.8%	25 с

Запропонований підхід на основі Isolation Forest продемонстрував значну перевагу над базовим підходом (F1-Score 0.92 проти 0.62).

• Найбільший розрив у продуктивності спостерігався в сценаріях Throughput Drop (Recall = 0.93 для IF). Оскільки вхідний потік залишався стабільним, а швидкість оброблення падала, ознака ефективності оброблення $\eta(t)$ миттєво опускалася нижче ніж 0.7. Статичні правила пропускали ці інциденти майже 2 хв, очікуючи, поки лаг досягне жорсткого порогу.

• У сценаріях Latency Spikes завдяки ознакам першої похідної Δx_4 система фіксувала стійкий градієнт зростання batchDuration до того, як він перетинав абсолютні критичні межі.

ISSN 3083-7715 (online) Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189)
ISSN 3083-7650 (0135-1710) (print) Management Information System and Devices. 2026. No. 2 (189)

Середній час виявлення (MTTD) для запропонованого підходу становив 25 с. Зважаючи на вікно агрегації $(\Delta t = 15\ c)$ і механізм фільтрації (підтвердження протягом двох тактів), це означає, що алгоритм розпізнає структурні зміни в метриках майже миттєво після їх появи. Статичні правила вимагали в середньому 115 с для фіксації проблеми. Обчислювальні накладні витрати процесу вилучення телеметрії становили < 1.5% процесорного часу кластера, що повністю підтверджує легкість розробленого підходу порівняно з перевіркою корисного навантаження.

Обговорення

Високий показник F1-Score (0.92) і низький рівень хибних спрацювань (FPR = 0.8%) стали можливими виключно завдяки 14-вимірному вектору ознак $F(t)$. Як продемонстрував експеримент, ізольований аналіз окремих метрик не дає змоги відрізнити легітимний сплеск трафіку від системної деградації. Упровадження синтетичних безрозмірних ознак – коефіцієнта ефективності $\eta(t)$ і нормованого лагу $\lambda(t)$ – надало алгоритму Isolation Forest здатність до контекстного розуміння. Система навчилася розрізняти, що зростання черги Kafka є нормою під час пропорційного сплеску вхідного трафіку (Load Burst), але є критичною аномалією, якщо трафік незмінний, а пропускна здатність впала.

Традиційні системи перевірки якості даних, орієнтовані на пакетне оброблення, вимагають десеріалізації кожного повідомлення. За швидкості 25000 подій/с це створює експоненційне навантаження на I/O і процесор. Запропонований підхід, аналізуючи телеметрію системи (JMX та Listener метрики), виконується асинхронно (out-of-band). Її обчислювальна складність є константою $O(1)$ залежно від обсягу корисного навантаження. Це повністю усуває вплив моніторингу на наскрізну затримку оброблення (end-to-end latency).

Незважаючи на високу швидкість реакції, цей підхід має концептуальні обмеження.

По-перше, підхід є сліпим до прихованих семантичних аномалій у даних. Якщо генератор почне надсилати некоректні Kafka-записи, де певні поля мають неправильні значення (зберігаючи правильну схему та правильні типи), обчислювальне навантаження на Spark залишиться стабільним, і телеметрія не зафіксує відхилень. Отже, розроблений підхід не замінює інструменти Data Quality, а діє як Tier-1 для захисту інфраструктури.

По-друге, алгоритм вразливий до проблеми холодного старту: після кожного архітектурного оновлення або масштабування кластера (scale-out) модель генеруватиме хибні сповіщення й потребуватиме періоду перенавчання (warm-up) для адаптації до нової норми.

Висновки й перспективи подальшого розвитку

У зв’язку з масовим переходом бізнесу на архітектури реального часу (Real-Time Analytics) забезпечення безперервності роботи потокових конвеєрів даних стає фундаментальною вимогою. У запропонованій роботі розв’язано науково-прикладне завдання автоматизованого виявлення аномалій у розподілених потокових системах на базі Apache Spark і Apache Kafka. Зокрема, виконано певні завдання.

1. Розроблено легковаговий (lightweight) підхід до моніторингу, оснований на вилученні телеметрії. Сформовано 14-вимірний простір ознак, що містить нормалізовані значення метрик продуктивності, швидкості їх зміни (градієнти) та структурні співвідношення між входом і обробленням η, λ.

2. Доведено високу точність виявлення. Експериментальна апробація за допомогою алгоритму Isolation Forest на моделі промислового стенда довела його ефективність. Якщо порівнювати з традиційними статичними правилами, запропонований підхід підвищив точність класифікації (F1-score) з 0.62 до 0.92.

3. Мінімізовано час реакції. Застосування машинного навчання до операційних метрик дало змогу скоротити середній час виявлення інциденту (MTTD) з 115 с до 25 с за умови збереження нульового впливу на швидкість оброблення даних у конвеєрі. Це повністю відповідає сучасній парадигмі Shift-Left архітектури.

Запропонований підхід органічно інтегрується в сучасні парадигми DataOps. Перспективи подальшого дослідження лежать у площині переходу від "виявлення" до "самовідновлення" (auto-remediation). Інтеграція виходів моделі з планувальниками

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

ресурсів (наприклад, Kubernetes HPA) дасть змогу конвеєрам автоматично масштабувати Executor-вузли для запобігання перевантаженням. Крім того, перспективним є дослідження моделей глибокого навчання (Deep Learning), зокрема LSTM-Autoencoders, для кращого врахування довгострокових сезонних залежностей складних систем.

Конфлікт інтересів

Автори декларують, що не мають конфлікту інтересів, зокрема фінансового, особистого, авторського чи будь-якого іншого характеру, який міг би вплинути на дослідження, а також на результати, опубліковані в цій статті.

Фінансування

Дослідження проводилося без фінансової підтримки.

Доступність даних

Рукопис не має пов’язаних матеріалів у сховищі даних.

Використання засобів штучного інтелекту

Автор підтверджує, що не застосовував технології штучного інтелекту для написання роботи.

References

1. Goedegebuure, A., Kumara, I., Driessen, S. (2024), "Data Mesh: A Systematic Gray Literature Review", *ACM Computing Surveys*, Vol. 57, pp. 1–36. DOI: https://doi.org/10.1145/3687301
2. Sulaiman, S., Rizwan, M. (2025), "Decentralizing Data for Larger Organizations: A Data Mesh Approach", *Journal of Computational Analysis & Applications*, Vol. 34, pp. 177–186. DOI: https://doi.org/10.48047/jocaaa.2025.34.11.15
3. Borodii, I., Osukhivska, H. (2026), "Research on the efficiency of data loading and storage in Data Lakehouse architectures for the formation of analytical data systems", *Information Technology: Computer Science, Software Engineering and Cyber Security*, No. 4, pp. 28–36. DOI: https://doi.org/10.48550/arXiv.2604.21449
4. Sudhanshubhai, P. J. (2026), "The Evolution of Shift-Left Testing in Modern Software Development", *Journal of Computational Analysis & Applications*, Vol. 35, pp. 1091–1100. DOI: https://doi.org/10.48047/jocaaa.2026.35.01.88
5. Zhong, Z. (2023), "A Survey of Time Series Anomaly Detection Methods in the AIOps Domain", arXiv preprint arXiv:2308.00393, pp. 1–36. DOI: https://doi.org/10.48550/arXiv.2308.00393
6. Skaperas, S., Koukis, G., Kapetanidou, I. A., Tsaoussidis, V., Mamatas, L. (2024), "A Pragmatical Approach to Anomaly Detection Evaluation in Edge Cloud Systems", *Proc. IEEE INFOCOM Workshops*, pp. 1–6. DOI: https://doi.org/10.48550/arXiv.2401.07717
7. Schmidl, S., Wenig, P., Papenbrock, T. (2022), "Anomaly Detection in Time Series: A Comprehensive Evaluation", *Proc. VLDB Endow.*, vol. 15, no. 9, pp. 1779–1797. DOI: https://doi.org/10.14778/3538598.3538602
8. Islam, M. S., Rakha, M. S., Pourmajidi, W., Sivaloganathan, J., Steinbacher, J., Miranskyy, A. (2024), "Anomaly Detection in Large-Scale Cloud Systems: An Industry Case and Dataset", *arXiv preprint arXiv:2411.09047*, pp. 1–12. DOI: https://doi.org/10.48550/arXiv.2411.09047
9. Darban, Z. Z., Webb, G. I., Pan, S., Aggarwal, C. C., Salehi, M. (2024), "Deep Learning for Time Series Anomaly Detection: A Survey", *ACM Comput. Surv.*, vol. 57, no. 2, Art. no. 39, pp. 1–42. DOI: https://doi.org/10.1145/3691338
10. Jacob, V., Diao, Y. (2025), "Unsupervised Anomaly Detection in Multivariate Time Series across Heterogeneous Domains", *arXiv preprint arXiv:2503.23060*, pp. 1–21. DOI: https://doi.org/10.48550/arXiv.2503.23060
11. Wang, Z., et al. (2024), "Revisiting VAE for Unsupervised Time Series Anomaly Detection: A Frequency Perspective", *in Proc. ACM Web Conf. (WWW)*, pp. 1–10. DOI: https://doi.org/10.48550/arXiv.2402.02820
12. Mayer, R., Mayer, L., Laich, L. (2020), "How Fast Can We Insert? An Empirical Performance Evaluation of Apache Kafka", *in Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, pp. 186–196. DOI: https://doi.org/10.48550/arXiv.2003.06452
13. Fedorovych, I., Osukhivska, H., Lutsyk, N. (2024), "Performance Benchmarking of Continuous Processing and Micro-Batch Modes in Spark Structured Streaming", *in Proc. ITTAP, CEUR Workshop Proc.*, vol. 3896, paper 5, pp. 1–11.
14. Jacob, V., Song, F., Stiegler, A., Diao, Y., Tatbul, N. (2021), "Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series", *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2613–2626. DOI: https://doi.org/10.14778/3476249.347630

15. Pragathi, B. C., Maddirala, H., Sneha, M. (2024), "Implementing an Effective Infrastructure Monitoring Solution with Prometheus and Grafana", *Int. J. Comput. Appl.*, vol. 186, no. 38, pp. 7–14. DOI: https://doi.org/10.5120/ijca2024923873
16. Pham, L., Ha, H., Zhang, H. (2024), "Root Cause Analysis for Microservice System Based on Causal Inference: How Far Are We?", *Proc. 39th IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE)*, pp. 1–13. DOI: https://doi.org/10.48550/arXiv.2408.13729
17. Vysotska, V., Kyrychenko, I., Demchuk, V. (2025), "Competency module of shift-left architecture in big data", *in Proc. PhD Workshop on Artificial Intelligence in Computer Science at CoLInS*, pp. 1–15. DOI: https://doi.org/10.31110/COLINS/2025-3/003
18. Bhardwaj, A. K. (2026), "Integrating Security Early A Shift-Left Model for DevSecOps in Modern Software Pipelines", *2026 6th International Conference on Image Processing and Capsule Networks (ICIPCN)*. DOI: https://doi.org/10.1109/ICIPCN67432.2026.11438475
19. Vysotska, V., Kyrychenko, I., Demchuk, V., Gruzdo, I. (2024), "Holistic Adaptive Optimization Techniques for Distributed Data Streaming Systems", *CEUR Workshop Proc.*, vol. 3624, pp. 1–13. DOI: https://doi.org/10.31110/COLINS/2024-2/009
20. Mandala, N. R. (2021), "ETL in Data Lakes vs. Data Warehouses", *ESP J. Eng. Technol. Adv.*, vol. 1, pp. 224–230. DOI: https://doi.org/10.56472/25832646/JETA-V1I2P123
21. Vysotska, V., Kyrychenko, I., Demchuk, V. (2025), "Adaptive Issue Detection in Holistic Optimization of Distributed Data Streaming Systems", *in Proc. MoDaST 2025: Modern Data Science Technologies Doctoral Consortium*, Lviv, Ukraine, pp. 1–17. URL: https://ceur-ws.org/Vol-4005/paper15.pdf.
22. Sharma, M. (2025), "Streaming Queries: Enabling Real-Time Elastic Scaling in Modern Applications", *Journal of Computer Science and Technology Studies*, Vol. 3, pp. 319–326. DOI: https://doi.org/10.32996/jcsts.2025.7.3.36
23. Chandola, V., Banerjee, A., Kumar, V. (2009), "Anomaly detection: A survey", *ACM Comput. Surv.*, vol. 41, no. 3, Art. no. 15, pp. 1–58. DOI: https://doi.org/10.1145/1541880.1541882

Received (Надійшла) 18.03.2026
Accepted for publication (Прийнята до друку) 12.05.2026
Publication date (Дата публікації) 29.05.2026

Відомості про авторів / About the Authors

Кириченко Ірина Віталіївна – кандидат технічних наук, Харківський національний університет радіоелектроніки, доцент кафедри програмної інженерії, Харків, Україна;
Iryna Kyrychenko – PhD (Technical Sciences), Kharkiv National University of Radio Electronics, Associate Professor of the Department of Software Engineering, Kharkiv, Ukraine;
e-mail: iryna.kyrychenko@nure.ua
ORCID ID: https://orcid.org/0000-0002-7686-6439

Демчук Вадим Геннадійович – Харківський національний університет радіоелектроніки, аспірант кафедри програмної інженерії, Харків, Україна;
Vadym Demchuk – Kharkiv National University of Radio Electronics, Postgraduate Student of the Department of Software Engineering, Kharkiv, Ukraine;
e-mail: vadym.demchuk@nure.ua
ORCID ID: https://orcid.org/0000-0003-3700-2344

Луценко Віталій Володимирович – Харківський національний університет радіоелектроніки, студент кафедри програмної інженерії, Харків, Україна;
Vitalii Lutsenko – Kharkiv National University of Radio Electronics, student of the Department of Software Engineering, Kharkiv, Ukraine;
e-mail: vitalii.lutsenko@nure.ua
ORCID ID: https://orcid.org/0009-0003-8965-5758

Автоматизовані системи управління та прилади автоматики. 2026. № 2 (189) *ISSN 3083-7715 (online)*
Management Information System and Devices. 2026. No. 2 (189) *ISSN 3083-7650 (0135-1710) (print)*

RESEARCH OF AUTOMATED DATA QUALITY TESTING AND ANOMALY DETECTION METHODS IN DATA PIPELINES

The object of study is the process of real-time operational monitoring, diagnostics, and ensuring the functional reliability of distributed streaming data pipelines. **The research** focuses on methods and algorithms for automated anomaly detection in streaming systems, based on the analysis of multivariate time series of operational telemetry (performance metrics and resource utilization) from Apache Spark Structured Streaming and Apache Kafka infrastructure components. **The aim** of the work is to develop and experimentally validate a lightweight methodology for proactive anomaly detection in high-speed data pipelines. This methodology operates exclusively on infrastructure meta-indicators, without resource-intensive payload inspection, to minimize incident response time and eliminate additional data processing latency. **Results Achieved**. During the study, a monitoring system architecture was developed, and a 14-dimensional feature space vector was formed. This vector includes normalized system metrics, their rates of change (gradients), and synthetic dimensionless coefficients (processing efficiency, normalized lag). An ensemble machine learning algorithm was applied to classify system states. Experimental modeling of typical failures (latency spikes, throughput drops, anomalous lag) on an AWS cloud cluster confirmed the approach's high efficiency. The proposed multivariate model increased incident detection accuracy (F1-score) from 0.62 (the performance of a classic rule-based method using static thresholds) to 0.92, with a false positive rate (FPR) of only 0.8%. The mean time to detect (MTTD) anomalies was reduced from 115 to 25 seconds. The computational overhead of the monitoring microservice accounted for less than 1.5% of the cluster's CPU time. **Conclusions**. It has been experimentally demonstrated that analyzing multivariate operational telemetry using machine learning is a highly effective proxy for data pipeline health. The proposed methodology successfully addresses the "heavy validation" problem inherent in traditional data quality tools and fully aligns with the modern Shift-Left architecture paradigm. The solution provides deep, zero-impact observability, serves as a reliable first line of proactive defense, and establishes a technological foundation for implementing infrastructure auto-remediation mechanisms.

Keywords: streaming data processing; data pipelines; anomaly detection; operational telemetry; machine learning; Isolation Forest; Apache Kafka; Apache Spark; Shift-Left architecture; system observability.

Бібліографічні описи / Bibliographic descriptions

Кириченко І. В., Демчук В. Г., Луценко В. В. Розроблення й дослідження підходу до виявлення аномалій у потокових конвеєрах даних на основі операційної телеметрії. *Автоматизовані системи управління та прилади автоматики*. 2026. № 2 (189). С. 165–181. DOI: https://doi.org/10.30837/0135-1710.2026.189.165

Kyrychenko, I., Demchuk, V., Lutsenko, V. (2026), "Research of automated data quality testing and anomaly detection methods in data pipelines", *Management Information System and Devices*, No. 2 (189), P. 165–181. DOI: https://doi.org/10.30837/0135-1710.2026.189.165