УДК 004.75:004.052.3:004.042 DOI: 10.31673/2412-4338.2025.048913

**Бученко Ігор Анатолійович**
*Державний університет інформаційно-комунікаційних технологій, Київ*
ORCID 0009-0003-9012-9632

**Лемешко Андрій Вікторович**
*Державний торговельно-економічний університет, Київ*
ORCID 0000-0001-8003-3168

**Лащевська Наталія Олександрівна**
*Державний університет інформаційно-комунікаційних технологій, Київ*
ORCID 0000-0003-2148-115X

# МЕТОДИ АНАЛІЗУ ПОТОКОВИХ ДАНИХ ДЛЯ ЗАБЕЗПЕЧЕННЯ ВІДМОВОСТІЙКОСТІ РОЗПОДІЛЕНИХ СИСТЕМ

**Анотація.** *Актуальність теми зумовлена тим, що сучасні розподілені інформаційні системи (РС) є базисом функціонування цифрових платформ у хмарних обчисленнях, фінансових сервісах та Інтернеті речей (IoT). Ефективність їхньої роботи та загальна надійність безпосередньо залежать від здатності обробляти величезні обсяги даних у реальному часі. Збої в роботі окремих компонентів або некоректна обробка даних можуть призводити до масштабних відмов, втрати даних та значних фінансових збитків. У цьому контексті особливої актуальності набувають методи потокового аналізу, які дозволяють у режимі реального часу відстежувати стан інфраструктури, виявляти аномалії та своєчасно реагувати на загрози. Застосування машинного навчання (ML) у потоковому аналізі суттєво підвищує точність прогнозування збоїв та оптимізації ресурсів. Метою дослідження є аналіз сучасних методів обробки та аналізу потокових даних для забезпечення безперебійної роботи та відмовостійкості інфраструктур розподілених систем, а також визначення практичних підходів до впровадження таких методів у реальних умовах. Методологія дослідження базується на аналізі наукових публікацій та технічної документації, порівняльному аналізі сучасних платформ потокової обробки даних, методах математичного моделювання для створення моделей моніторингу та прогнозування відмов, а також емпіричних методах для дослідження продуктивності РС. Детально проаналізовано такі платформи, як Apache Kafka, Apache Flink та Apache Spark Streaming. Оцінено роль методів машинного навчання, зокрема, у контексті виявлення аномалій та адаптації до «дрейфу концепцій» (concept drift). У результаті дослідження систематизовано методи потокового аналізу для підвищення відмовостійкості РС. Визначено ключові архітектурні відмінності провідних інструментів: Apache Kafka як розподілена платформа збереження та передавання подій (принцип «журналу змін»); Apache Flink як інструмент для обробки потоків з мінімальною затримкою та підтримкою станоутримуючих обчислень (stateful computations) і контрольних точок (checkpointing); Apache Spark Streaming, що реалізує підхід міні-пакетної обробки (micro-batching). Розглянуто роль інструментів оркестрації, зокрема Kubernetes, у автоматизації розгортання, масштабування та самовідновлення потокових конвеєрів. Обґрунтовано доцільність застосування ML для прогнозування відмов у режимі реального часу. Деталізовано практичні методи забезпечення безперебійної роботи, включаючи резервне копіювання (знімки Velero у Kubernetes), кластеризацію , моніторинг (Prometheus, Grafana) та тестування відмовостійкості, зокрема, із застосуванням підходів "chaos engineering" (наприклад, Chaos Mesh). Практична значущість дослідження полягає у підготовці рекомендацій для впровадження методів потокового аналізу та систем забезпечення відмовостійкості у реальних інформаційних системах, зокрема у хмарних платформах, IoT-системах та критичних IT-інфраструктурах.*

**Ключові слова:** *комп'ютерна мережа, потокові дані, обробка в реальному часі, Apache Kafka, Apache Flink, машинне навчання, виявлення аномалій, Kubernetes, chaos engineering.*

**Buchenko Ihor**
*State University of Information and Communication Technologies, Kyiv, Ukraine*
ORCID 0009-0003-9012-9632

**Lemeshko Andriy**
*State University of Trade and Economics, Kyiv, Ukraine*
ORCID 0000-0001-8003-3168

**Lashchevska Nataliia**

*State University of Information and Communication Technologies, Kyiv, Ukraine*
ORCID 0000-0003-2148-115X

## METHODS OF STREAM DATA ANALYSIS FOR ENSURING FAULT TOLERANCE OF DISTRIBUTED SYSTEMS

**Abstract.** *The relevance of the topic stems from the fact that modern distributed information systems (DS) form the basis for the functioning of digital platforms in cloud computing, financial services, and the Internet of Things (IoT). Their operational efficiency and overall reliability directly depend on the ability to process vast amounts of data in real-time. Component failures or incorrect data processing can lead to large-scale outages, data loss, and significant financial damages. In this context, stream analysis methods that allow real-time monitoring of infrastructure status, anomaly detection, and timely response to threats are particularly relevant. The application of machine learning (ML) in stream analysis significantly increases the accuracy of failure prediction and resource optimization. The purpose of the study is to analyze modern methods of processing and analyzing stream data to ensure the uninterrupted operation and fault tolerance of distributed system infrastructures, as well as to identify practical approaches for implementing such methods in real-world conditions. The research methodology is based on the analysis of scientific publications and technical documentation, comparative analysis of modern stream processing platforms, mathematical modeling methods for creating monitoring and failure prediction models, and empirical methods for studying DS performance. Platforms such as Apache Kafka, Apache Flink, and Apache Spark Streaming are analyzed in detail. The role of machine learning methods is assessed, particularly in the context of anomaly detection and adaptation to "concept drift". As a result of the study, stream analysis methods for enhancing DS fault tolerance have been systematized. Key architectural differences of leading tools were identified: Apache Kafka as a distributed event storage and streaming platform (commit log principle); Apache Flink as a tool for low-latency stream processing with support for stateful computations and checkpointing ; and Apache Spark Streaming, which implements a micro-batching approach. The role of orchestration tools, particularly Kubernetes, in automating the deployment, scaling, and self-healing of streaming pipelines is examined. The feasibility of using ML for real-time failure prediction is substantiated. Practical methods for ensuring uninterrupted operation are detailed, including backup (Velero snapshots in Kubernetes ), clustering , monitoring (Prometheus, Grafana ), and fault tolerance testing, including the use of "chaos engineering" approaches (e.g., Chaos Mesh ). The practical significance of the research lies in preparing recommendations for implementing stream analysis methods and fault tolerance systems in real-world information systems, particularly in cloud platforms, IoT systems, and critical IT infrastructures.*

**Keywords:** *computer network, streaming data, real-time processing, Apache Kafka, Apache Flink, machine learning, anomaly detection, Kubernetes, chaos engineering.*

**Постановка проблеми**

Сучасні розподілені інформаційні системи є основою функціонування цифрових платформ у хмарних обчисленнях, фінансових сервісах, Інтернеті речей та інших галузях. Ці системи обробляють величезні обсяги даних у реальному часі, тому їхня надійність та безперебійна робота безпосередньо залежать від ефективності аналізу потокових даних. Збої у роботі окремих компонентів або неправильна обробка даних можуть призводити до масштабних відмов, втрати даних та фінансових збитків.

Особливої актуальності набувають методи потокового аналізу, які дозволяють у режимі реального часу відстежувати стан інфраструктури, виявляти аномалії та своєчасно реагувати на загрози. Застосування машинного навчання у потоковому аналізі підвищує точність прогнозування збоїв та оптимізації ресурсів. Дослідження таких підходів є важливим для підвищення відмовостійкості розподілених систем. Об’єктом дослідження є процес аналізу потокових даних у розподілених системах. Предметом дослідження – методи, технології та підходи до забезпечення безперебійної роботи та відмовостійкості інфраструктур розподілених систем на основі потокового аналізу.

**Аналіз останніх досліджень і публікацій**

Основи великомасштабної обробки даних у реальному часі, що є фундаментом для даного дослідження, викладені у праці Т. Акідау, Б. Слаткіна та С. О’Ніла [1]. Автори визначають потокові дані як безперервну послідовність подій, що потребують негайної обробки. Ця динамічна природа відрізняє потокову обробку від пакетної, де дані накопичуються статично. Дж. Крепс [6] у своїй роботі ставить під сумнів доцільність архітектури Lambda, вказуючи на надмірну складність поєднання пакетної та потокової обробки, та обґрунтовує перехід до систем, орієнтованих виключно на потоки, де логіка обробки є єдиною.

Джерела даних для таких систем є надзвичайно різноманітними: від логів серверів та даних датчиків IoT до фінансових транзакцій. Українські дослідники, зокрема С. Журавель, С. Думич та О. Шпур [10], досліджували методи збору та обробки даних у РС, акцентуючи на викликах, пов'язаних з

одночасною підтримкою різних форматів та забезпеченням послідовності обробки при високому навантаженні.

Забезпечення відмовостійкості у таких системах вимагає спеціалізованих абстракцій. М. Захарія та співавтори [9] запропонували концепцію відмовостійких розподілених наборів даних (Resilient Distributed Datasets), яка стала основою для багатьох сучасних платформ, забезпечуючи надійне зберігання результатів навіть у разі відмови вузлів.

Проте, незважаючи на значну кількість праць, присвячених окремим інструментам (наприклад, Kafka [7] або Flink [3]) чи методам (наприклад, балансування навантаження [12] або протоколи консенсусу [13]), залишається невирішеною частина загальної проблеми: відсутня комплексна систематизація методів потокового аналізу саме у контексті їхнього прямого впливу на підвищення відмовостійкості. Недостатньо вивченими залишаються питання інтеграції інструментів потокової обробки, систем оркестрації (як Kubernetes [2]) та методів машинного навчання [4, 11] у єдиний механізм проактивного забезпечення безперебійної роботи РС.

**Мета і задачі дослідження**

Метою дослідження є аналіз сучасних методів обробки та аналізу потокових даних для забезпечення безперебійної роботи та відмовостійкості інфраструктур розподілених систем, а також визначення практичних підходів до впровадження таких методів у реальних умовах.

**Результати дослідження**

*Інструменти для аналізу потокових даних*

Для побудови розподілених конвеєрів обробки даних у режимі реального часу стандартом стали платформи Apache Kafka, Apache Flink та Apache Spark Streaming.

Apache Kafka є розподіленою платформою збереження та передавання подій, що гарантує їх доставку при відмовах вузлів. Її архітектура заснована на принципі «журналу змін» (commit log), де події записуються у впорядковану стрічку, що забезпечує надійність та можливість повторного відтворення даних. Kafka дозволяє масштабувати потоки шляхом розподілу тем (topics) на партиції (partitions), кожна з яких може оброблятися незалежно, досягаючи високої пропускної здатності.

Apache Flink (рис. 1) пропонує обробку потоків з мінімальною затримкою та підтримує складні сценарії, такі як обробка у часових вікнах та станоутримуючі обчислення (stateful computations). Ключовою особливістю є вбудована система контрольних точок (checkpointing), що зберігає проміжний стан обчислень. Це забезпечує автоматичне відновлення обробки з останнього коректного стану у разі збою, що є критичним для забезпечення цілісності результатів.

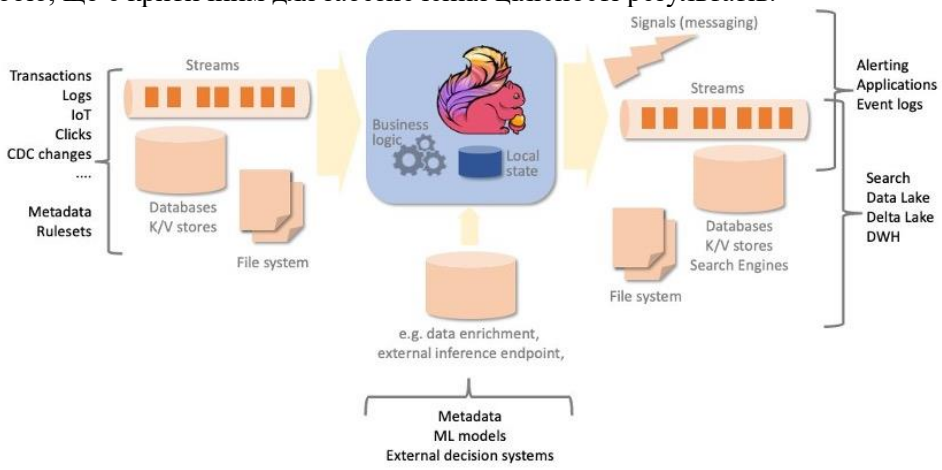


Рис. 1. Принцип роботи Apache Flink

Apache Spark Streaming (рис. 2) використовує підхід міні-пакетної обробки (micro-batching). Дані агрегуються у невеликі часові інтервали (пакети) і обробляються за допомогою механізмів Spark. Це дозволяє поєднувати потокову аналітику з пакетним аналізом історичних даних на одній платформі, що є корисним для побудови прогнозних моделей.


Рис. 2. Принцип роботи Apache Spark Streaming

Інфраструктурною основою для цих інструментів часто виступає Kubernetes (рис. 3). Ця платформа автоматизує розгортання та керування контейнеризованими компонентами. Механізми автоматичного масштабування Kubernetes дозволяють динамічно адаптувати кількість обчислювальних вузлів до навантаження. Функції моніторингу стану та автоматичного перезапуску контейнерів забезпечують самовідновлення потокових конвеєрів без втручання адміністратора. Важливу роль відіграють і системи балансування навантаження, що запобігають перевантаженню окремих компонентів.

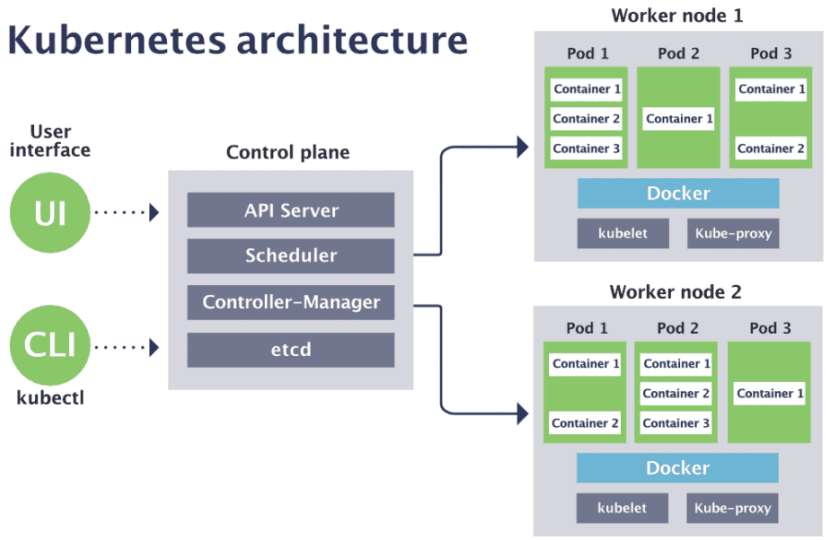


Рис. 3. Архітектура інструменту Kubernetes

*Машинне навчання у реальному часі для підвищення надійності*

Машинне навчання (ML) у потоковому аналізі дозволяє не лише відстежувати поточний стан, а й прогнозувати можливі відмови. Аналіз потоків подій у реальному часі забезпечує виявлення закономірностей, що свідчать про наближення збоїв.

Одним із головних викликів є "concept drift" (дрейф концепцій) – зміна статистичних характеристик даних, яка знижує точність моделей, навчених на історичних даних. Це актуально для динамічних систем, де трафік змінюється залежно від сезону чи конфігурації. Для вирішення цієї проблеми застосовуються методи онлайн-навчання (online learning), які постійно оновлюють моделі на основі нових даних, підтримуючи їх актуальність.

Важливе місце займає виявлення аномалій для ідентифікації загроз: апаратних збоїв, мережевих атак або несанкціонованого доступу. Для цього використовуються як прості порогові значення, так і складні методи, наприклад, ізоляційні ліси (Isolation Forest), що ефективні навіть за відсутності еталонів нормальної поведінки.

Крім того, ML використовується для оптимізації розподілу навантаження у хмарних середовищах. Прогнозування навантаження дозволяє завчасно виділяти додаткові ресурси для завантажених компонентів, мінімізуючи ризик перевантаження та оптимізуючи витрати на інфраструктуру.

*Роль потокового аналізу в архітектурі відмовостійкості*

Потоковий аналіз є ключовим для забезпечення безперебійної роботи, оскільки дозволяє виявляти не лише інциденти, а й передумови до них, що уможливлює проактивні заходи.

Інтеграція потокового аналізу з механізмами відмовостійкості (автоматичне перезавантаження, балансування навантаження, міграція даних) забезпечує адаптивність системи. Наприклад, при виявленні деградації вузла, система моніторингу може автоматично ініціювати переміщення навантажень на інші вузли.

Потоковий аналіз також інтегрується із системами резервного копіювання, фіксуючи стан критичних компонентів у реальному часі. Це дозволяє відтворити точний стан системи на момент збою, прискорюючи відновлення.

Для забезпечення узгодженості даних у складних РС, де події надходять асинхронно, використовуються технології консенсусу, як-от RAFT. RAFT гарантує послідовне збереження та синхронізацію критичних змін стану між репліками, навіть у випадку втрати серверів чи порушень зв’язку.

Для систематизації процесу забезпечення відмовостійкості було формалізовано метод потокового аналізу у вигляді структурної схеми (рис. 4). Метод передбачає послідовну трансформацію даних через етапи буферизації, попередньої обробки та інтелектуального аналізу.

Ключовою особливістю запропонованого підходу є наявність контуру зворотного зв'язку через модуль автоматичного реагування (оркестратор), що дозволяє системі адаптуватися до змін навантаження або виникнення аномалій без втручання адміністратора. Використання проміжного шару буферизації (на базі Apache Kafka) гарантує збереження подій навіть у випадку тимчасової недоступності обчислювальних вузлів, а модуль машинного навчання забезпечує проактивне виявлення відхилень, нівелюючи вплив «дрейфу концепцій» (concept drift).

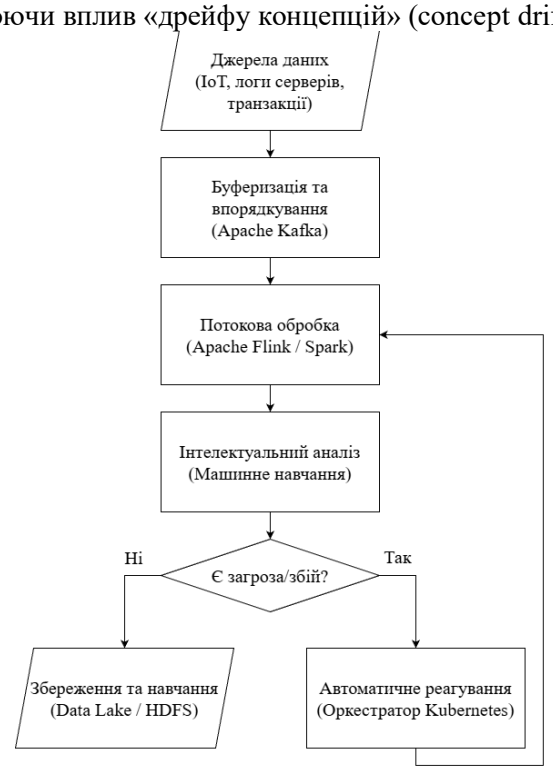


Рис. 4. Узагальнена блок-схема методу потокового аналізу для забезпечення відмовостійкості

Математичне моделювання процесів потокової обробки та відмовостійкостіДля формалізації процесів забезпечення безперебійної роботи розподіленої системи доцільно розглянути її як систему масового обслуговування (СМО). Нехай вхідний потік даних (подій) $S$ характеризується інтенсивністю надходження заявок $\lambda$ (подій/сек). Система обробки складається з $n$ вузлів, кожен з яких має середню інтенсивність обробки $\mu$.Умовою стабільної роботи системи, відсутності необмеженого зростання черги в Apache Kafka, є виконання нерівності (1).

$$\lambda < \sum_{i=1}^{n} \mu_i \,, \tag{1}$$

де $\mu$ – продуктивність $i$-го вузла обробки (worker node).

Час перебування події в системі $T_{sys}$ (затримка) складається з часу очікування в черзі $T_{queue}$ та часу безпосередньої обробки $T_{proc}$ (2).

$$T_{sys} = T_{queue} + T_{proc} \tag{2}$$

Для забезпечення відмовостійкості вводиться коефіцієнт надлишковості (replication factor) $r$. Ймовірність безвідмовної роботи (надійності) окремого вузла на інтервалі часу $t$ позначимо як $P_{node}(t)$. тоді ймовірність відмови вузла становить (3).

$$Q_{node}(t) = 1 - P_{node}(t) \tag{3}$$

У кластері, де дані реплікуються на $r$ вузлів, втрата даних відбувається лише у випадку одночасної відмови всіх $r$ реплік. Ймовірність втрати даних $Q_{data}(t)$ для партиції визначається по формулі (4).

$$Q_{data}(t) = \prod_{j=1}^{r} Q_{node_j} \approx \left(1 - P_{node}(t)\right)^r \quad (4)$$

Основним показником надійності системи є коефіцієнт готовності (Availability) $A$, який залежить від середнього часу напрацювання на відмову $MTBF$ та середнього часу відновлення $MTTR$ (5).

$$A = \frac{MTBF}{MTTR + MTTR} \quad (5)$$

Застосування автоматизованих засобів оркестрації (Kubernetes) та потокового моніторингу дозволяє мінімізувати $MTTR$ шляхом автоматичного перезапуску подів, що, відповідно до формули, підвищує загальний коефіцієнт готовності $A \to 1$.

Задача виявлення аномалій за допомогою машинного навчання формалізується наступним чином. Нехай, $x_t$ – вектор метрик стану системи в момент часу $t$ (навантаження CPU, затримка мережі, довжина черги). Функція прогнозування $F(X_{t-k}, \dots, X_{t-1})$ оцінює очікуваний стан $\hat{x_t}$. Аномалія фіксується, якщо відхилення реального значення від прогнозованого перевищує порогове значення $\theta$ (6).

$$\|x_t - \hat{x_t}\| > \theta \quad (6)$$

Адаптація до зміни концепції (concept drift) у цьому випадку реалізується через динамічне оновлення параметрів функції $F$ на основі вікна останніх спостережень $W$ (7).

$$\theta_t = \alpha \times \sigma(W_t) + \beta, \quad (7)$$

де $\sigma(W_t)$ – середньоквадратичне відхилення метрик у вікні $W_t$, а $\alpha$, $\beta$ – коефіцієнти чутливості моделі.

*Практичні методи резервного копіювання та відновлення*

Резервне копіювання (РК) є базовим інструментом відмовостійкості. Для потокових систем РК охоплює не лише підсумкові дані, а й проміжні стани обробки – контрольні точки (checkpoints). Наприклад, у Apache Flink контрольні точки можуть автоматично створюватися з заданим інтервалом і зберігатися у розподілених файлових системах (HDFS або S3-сумісних). Це забезпечує відновлення конвеєра після аварії з мінімальними втратами.

Для систем на базі Kubernetes РК охоплює конфігурації кластерів та секрети. Інструменти, як Velero, дозволяють створювати знімки (snapshots) кластерів, включаючи всі об’єкти (Pods, Services тощо). Приклад команди створення РК знімку кластера Kubernetes за допомогою Velero:

*velero backup create backup-state-registry --include-namespaces state-registry --storage-location default*

Такі підходи є критичними для захисту систем, де втрата даних неприпустима, наприклад, державних реєстрів.

*Кластеризація та горизонтальне масштабування*

Кластеризація є основою відмовостійких систем. У потокових системах, як Apache Kafka, кластеризація є невід’ємною частиною архітектури. Теми Kafka розділені на партиції, що розподіляються між брокерами (вузлами кластера). Якщо брокер виходить з ладу, його партиції автоматично передаються іншим вузлам, забезпечуючи безперервність обробки.

Приклад конфігурації для створення кластеру Kafka з трьома репліками у Kubernetes (використовуючи оператор Strimzi):

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: kafka-cluster
spec:
  kafka:
    replicas: 3
```

```
listeners:
 - name: plain
   port: 9092
   type: internal
```

Горизонтальне масштабування дозволяє оперативно реагувати на зміну навантаження. Автоматичне масштабування в Kubernetes (наприклад, Horizontal Pod Autoscaler) дозволяє динамічно додавати або прибирати обчислювальні одиниці (поди) залежно від поточного рівня використання ресурсів (CPU, пам'ять).

*Моніторинг та попереджувальне обслуговування*

Моніторинг є головним інструментом контролю стану РС. Поширеними інструментами є Prometheus (для збору метрик) та Grafana (для їх візуалізації).

Важливою є інтеграція моніторингу з потоковими платформами. Наприклад, брокери Kafka надають велику кількість метрик (кількість повідомлень, об'єм черг, затримка) через JMX-експортер, які збираються Prometheus. Ці дані дозволяють аналітично виявляти тенденції до деградації системи. Приклад конфігурації Prometheus для збору метрик з брокерів Kafka:

```
- job_name: 'kafka'
  static_configs:
    - targets: ['kafka-broker-1:9100', 'kafka-broker-2:9100']
```

Попереджувальне обслуговування передбачає аналіз зібраних метрик з використанням алгоритмів ML для прогнозування відмов на основі виявлених аномалій у потоках даних. Наприклад, у системах кібербезпеки це дозволяє виявляти початкові етапи атак.

*Оптимізація та тестування відмовостійкості (Chaos Engineering)*

Забезпечення відмовостійкості вимагає не лише моніторингу, але й оптимізації та тестування. Оптимізація потокових систем включає архітектурні рішення (наприклад, дедуплікація повідомлень у Kafka, використання компактних форматів даних як Avro) та точне налаштування параметрів (наприклад, частота контрольних точок у Flink, таймаути обробки).

Ключову роль відіграє тестування відмовостійкості. Все частіше використовується концепція chaos engineering – свідоме внесення збоїв у тестове або контрольоване продуктивне середовище для перевірки здатності системи до самовідновлення. У Kubernetes це може бути навмисне завершення роботи подів, симуляція відмови мережі або збільшення затримок. Для цього використовуються спеціалізовані інструменти, як-от Chaos Mesh. Приклад сценарію Chaos Mesh для тестування мережевої затримки (500ms) у кластері Kafka:

```
apiVersion: chaos-mesh.org/v1alpha1
kind: NetworkChaos
metadata:
  name: network-delay
spec:
  action: delay
  mode: one
  duration: "10m"
  selector:
    namespaces:
      - kafka-cluster
  delay:
    latency: "500ms"
```

Такі перевірки дозволяють виявити слабкі місця в архітектурі та оцінити ефективність механізмів автоматичного відновлення.

**Висновки і перспективи подальших досліджень**

Проведене дослідження дозволило систематизувати сучасні методи аналізу потокових даних, спрямовані на забезпечення безперебійної роботи та відмовостійкості розподілених систем. Актуальність цієї задачі підтверджується тим, що більшість сучасних ІТ-систем працюють у реальному часі та обробляють великі обсяги інформації, де стабільність є критичним фактором.

Встановлено, що потоковий аналіз дає можливість здійснювати моніторинг стану системи в реальному часі, оперативно виявляти збої та передумови до них, і негайно реагувати на інциденти. Платформи, такі як Apache Kafka, Apache Flink та Apache Spark Streaming, ідентифіковані як основні інструменти для побудови відмовостійких конвеєрів обробки даних.

Доведено, що застосування методів машинного навчання дозволяє перейти від реактивного до проактивного управління надійністю, забезпечуючи не лише виявлення аномалій, а й прогнозування потенційних збоїв.

Розглянуто ключові практичні методи забезпечення безперебійної роботи, що використовуються у сучасних РС, зокрема: резервне копіювання (включно з механізмами контрольних точок та знімків стану), кластеризація (як невід'ємна частина архітектури потокових платформ), автоматичне горизонтальне масштабування (зокрема, засобами Kubernetes) та комплексний моніторинг (Prometheus/Grafana). Окрему увагу приділено методам тестування, зокрема "chaos engineering", як ефективному способу верифікації стійкості системи до збоїв.

Дослідження показало, що тільки комплексне застосування потокового аналізу, автоматизованого моніторингу, машинного навчання та сучасних технологій управління кластерами (оркестрації) дозволяє створювати надійні, гнучкі та відмовостійкі розподілені системи, здатні ефективно функціонувати в умовах постійних змін та високих навантажень.

Перспективи подальших досліджень полягають у розробці гібридних моделей машинного навчання для більш точного прогнозування збоїв у гетерогенних розподілених середовищах, а також у дослідженні методів автоматичної адаптації параметрів потокової обробки (наприклад, частоти контрольних точок Flink) на основі прогностичних моделей навантаження.

## Список використаної літератури


1. Akidau, T. Streaming Systems: The What, Where, When, and How of Large-Scale Data Processing / T. Akidau, B. Slatkin, S. O'Neill. – O'Reilly Media, 2018. – 352 с.
2. Burns, B. Kubernetes: Up and Running / B. Burns, J. Beda, K. Hightower. – O'Reilly Media, 2019. – 326 с.
3. Carbone, P. Apache Flink: Stream and Batch Processing in a Single Engine / P. Carbone, A. Katsifodimos, S. Ewen // IEEE Data Engineering Bulletin. – 2015. – Т. 38, № 4. – С. 28–38.
4. Gama, J. A Survey on Concept Drift Adaptation / J. Gama, I. Žliobaite, A. Bifet // ACM Computing Surveys. – 2014. – Т. 46, № 4. – С. 44:1–44:37.
5. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center / B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, I. Stoica // Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI). – 2011. – С. 295–308.
6. Kreps, J. Questioning the Lambda Architecture [Електронний ресурс] / J. Kreps // O'Reilly Radar. – 2014. – Режим доступу: https://www.oreilly.com/radar/questioning-the-lambda-architecture/
7. Shapira, G. Kafka: The Definitive Guide / G. Shapira, T. Palino, R. Sivaram. – O'Reilly Media, 2021. – 432 с.
8. An Architecture for Fast and General Data Processing on Large Clusters / M. Zaharia, M. Chowdhury, T. Das [та ін.] // Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing (HotCloud). – 2012. – 6 с.
9. Zaharia, M. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing / M. Zaharia, M. Chowdhury, T. Das // Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI). – 2012. – С. 15–28.
10. Журавель, С. Дослідження методів збору та обробки даних в розподілених інформаційних системах [Електронний ресурс] / С. Журавель, С. Думич, О. Шпур // Комп'ютерні науки та інформаційні технології. – 2021. – № 1(1). – Режим доступу: https://science.lpnu.ua/sites/default/files/journal-paper/2021/dec/25668/stattya3czhuravelsdumichoshpur.pdf
11. Застосування методів інтелектуального аналізу даних до розв'язання актуарних задач [Електронний ресурс] // Електронний архів КПІ ім. Ігоря Сікорського. – Режим доступу: https://ela.kpi.ua/bitstreams/aa947d4a-666e-4b70-a5ba-ae7c6d504085/download
12. Методи балансування навантаження у розподілених системах з урахуванням обмежень [Електронний ресурс] // Відкритий архів Харківського національного університету радіоелектроніки. – Режим доступу: https://openarchive.nure.ua/entities/publication/ad7eb785-6682-4bdb-86d5-67694e10de07
13. Оптимізація використання RAFT в розподілених системах з базами даних [Електронний ресурс] // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. – 2024. – Т. 35, № 3, Ч. 1. – Режим доступу: https://www.tech.vernadskyjournals.in.ua/journals/2024/3_2024/part_1/9.pdf
14. Сметаненко, А. В. Побудова розподілених систем з використанням хмарних технологій : кваліфікац. робота / А. В. Сметаненко ; Чорноморський національний університет імені Петра Могили. – Миколаїв, 2020. – 63 с. – Режим доступу: https://krs.chmnu.edu.ua/jspui/bitstream/123456789/3652/1/Сметаненко%20Артем%20Віталійович.pdf