УДК 004.75:004.65

ХМАРНА АРХІТЕКТУРА РОЗПОДІЛЕНОЇ ОБРОБКИ ТРАНСПОРТНИХ ДАНИХ НА БАЗІ АДАПТИВНОГО АЛГОРИТМУ

Терзі Д. Д.[1]; Волощук Л. А., к.т.н., доц.[2]
[1,2] Одеський національний університет імені І. І. Мечникова
[1,2] Україна, Одеса
[1] terzi.diana@stud.onu.edu.ua; [2] lavstumbre@gmail.com

Анотація. *У роботі розроблено інформаційну технологію та адаптивний метод організації обробки гетерогенних потоків даних (Adaptive Data Processing Algorithm, ADPA) у розподілених інтелектуальних транспортних системах (ITS). Запропоновано формальну модель класифікації подій, формулу адаптивного інтервалу мікропакетування з гістерезисом і мультикритеріальний навантажувальний коефіцієнт із контуром зворотного зв'язку на базі 95-го перцентиля затримки. Спроектовано багаторівневу cloud-edge архітектуру на сервісах AWS із поділом на мікросервісні шари.*

Ключові слова: *розподілені інформаційні системи; хмарна архітектура; інтелектуальні транспортні системи; адаптивна обробка даних; мікропакетування; сервіси AWS.*

Вступна частина. Сучасний розвиток концепції розумного міста (Smart City) зумовлює стрімке зростання обсягів даних, що генеруються в інфраструктурі інтелектуальних транспортних систем (Intelligent Transport Systems, ITS). Безперервний потік просторово-часової інформації від GPS-трекерів, відеокамер та дорожніх контролерів має гетерогенну структуру, високу інтенсивність і нерівномірну щільність у часі. Традиційні централізовані cloud-centric архітектури виявляються малоефективними, оскільки передача всього первинного масиву інформації до хмари перевантажує канали зв'язку та створює ефект «вузького місця» (bottleneck), критично збільшуючи час затримки (latency) [1].

З іншого боку, конвеєри обробки даних у більшості сучасних рішень спираються на статичні схеми: суто пакетна обробка (batch processing) унеможливлює своєчасне виявлення дорожньо-транспортних пригод, тоді як повноцінна потокова обробка (true stream processing) для всіх без винятку подій створює надмірне навантаження на обчислювальні вузли й призводить до невиправданих експлуатаційних витрат [2]. Стандартні механізми авто-масштабування зі статичними пороговими значеннями схильні до ефекту частого перемикання станів (flapping) і не здатні проактивно реагувати на раптові сплески навантаження [3]. Виникає науково-практичне завдання синтезу адаптивного алгоритмічного апарату та масштабованої хмарно-периферійної інфраструктури, що забезпечують компроміс між latency, throughput та раціональним споживанням ресурсів.

Метою дослідження є розробка хмарної архітектури розподіленої ITS-системи на основі інформаційної технології з використанням адаптивного методу організації обробки даних, що забезпечує вибір режиму обробки відповідно до характеристик вхідних подій і поточного навантаження системи, а також сприяє раціональному використанню обчислювальних ресурсів, масштабованості та надійності системи.

Основна частина. Для вирішення поставленої проблеми запропоновано формальну постановку задачі оптимізації обробки потоку даних. Нехай вхідний потік транспортних даних подано множиною подій $D = \{d_1, d_2, ..., d_n\}$, де кожна подія d_i описується вектором ознак: $d_i = \langle type_i, priority_i, \lambda_i, size_i, t_i \rangle$, де $type_i \in T$ – тип події (множина T: координати, дані

сенсорів, відеометадані, сигнали ДТП); $priority_i \in \{HIGH, MEDIUM, LOW\}$ – рівень критичності; λ_i – поточна інтенсивність надходження подій типу $type_i$ (подій/с); $size_i$ – розмір корисного навантаження (байт); t_i – часова мітка надходження події.

Поточний стан обчислювальної системи в момент часу t описується вектором навантаження $S(t) = \langle CPU_t, MEM_t, Q_t, R_t \rangle$, де $CPU_t \in [0, 100]$ та $MEM_t \in [0, 100]$ – завантаження процесора та пам'яті (%); Q_t – довжина черги необроблених повідомлень (backlog); R_t – поточна сумарна інтенсивність надходження даних. Задача розробленого методу Adaptive Data Processing Algorithm (ADPA) – побудувати відображення $F: (d_i, S(t)) \rightarrow (M_i, R_i)$, де $M_i \in \{STREAM, MICRO_BATCH, BATCH\}$ – режим обробки, а R_i – обсяг виділених ресурсів. Цільова функція мінімізує Latency(d) для HIGH-пріоритетних подій, мінімізує витрати $Cost(R) = \sum R_i \cdot c_i$ та максимізує Throughput(t) за умови $Latency(d) \leq L_max\text{^}critical$ та $CPU_t \leq CPU_max$.

Для оперативного прийняття рішень у реальному часі запропоновано трирівневе евристичне правило класифікації: $class(d_i) = CRITICAL$, якщо $priority_i = HIGH$ або $type_i \in T_critical$ (обробляється негайно в режимі STREAM); $class(d_i) = CONTINUOUS_MONITORING$, якщо $type_i \in T_continuous$ і $\lambda(type_i) > \theta_stream$ (режим MICRO_BATCH як компроміс між затримкою і пропускною здатністю); $class(d_i) = HISTORICAL_ANALYTICAL$ в інших випадках (режим BATCH).

Для класу CONTINUOUS_MONITORING тривалість вікна мікропакета розраховується адаптивно: $\Delta t = clamp(T_base \cdot k(Q_t, CPU_t), T_min, T_max)$, де T_base – базовий SLA-інтервал, а $k(Q_t, CPU_t)$ – кусково-лінійний коефіцієнт корекції. При $Q_t > Q_high$ або $CPU_t > CPU_high$ коефіцієнт приймає значення $k_dec < 1$ (швидке розвантаження меншими порціями); при спаді $\lambda_t < \lambda_low$ та $CPU_t < CPU_low$ – $k_inc > 1$; в інших випадках $k = 1$. Наявність нейтральної зони ($k = 1$) реалізує петлю гістерезису, усуваючи коливання рішень (flapping).

Масштабування ресурсів ґрунтується на мультикритеріальному навантажувальному коефіцієнті: $LF(t) = w_1 \cdot (CPU_t / 100) + w_2 \cdot \min(Q_t / Q_max, 1) + w_3 \cdot \min(\lambda_t / \lambda_max, 1)$. Обґрунтовано ваги: $w_1 = 0{,}5$ (CPU – пряме фізичне обмеження продуктивності), $w_2 = 0{,}3$ (черга – інтегральний запізнюючий показник), $w_3 = 0{,}2$ (інтенсивність надходження – проактивний індикатор, feedforward-компонента). Якщо $LF(t) > \theta_up$, виділяється $R_{t+1} = R_t + \Delta R^+$; якщо $LF(t) < \theta_down$, виділяється $R_{t+1} = \max(R_min, R_t - \Delta R^-)$. Для адаптації до змін навантаження впроваджено контур зворотного зв'язку за правилом: $\theta_up(нове) = \theta_up(старе) + \alpha \cdot (SLA_target - P_{95}(latency \in W_\tau))$, де P_{95} – 95-й перцентиль затримки за вікно W_τ. Амортизована складність алгоритму на одну подію становить O(1).

Архітектурна реалізація розробленої технології з використанням адаптивного методу організації обробки гетерогенних потоків даних у розподілених інтелектуальних транспортних системах побудована за багаторівневою моделлю cloud-edge з використанням екосистеми AWS (рис.1): 1) Edge-рівень (AWS IoT Greengrass) – виконує локальну фільтрацію, просторове проріджування координат та аналіз критичних інцидентів безпосередньо біля джерел; 2) Ingestion-рівень (AWS IoT Core та Amazon Kinesis Data Streams) – забезпечує надійне приймання та буферизацію високоінтенсивних потоків; 3) Рівень розподіленої обробки – містить функції AWS Lambda для STREAM-шляху та кластер Apache Spark у середовищі Amazon EMR для паралельної MICRO_BATCH та BATCH аналітики; 4) Рівень зберігання та аналітики – включає Amazon S3 як Data Lake для сирих подій та Amazon Redshift для складних аналітичних SQL-запитів; 5) Мікросервісний контур –

розгорнутий в Amazon EKS з контейнеризацією Docker. Сервіси збору GPS, обробки, аналітики, детекції аномалій і сповіщень взаємодіють за схемою Event-Driven Architecture через шину Kinesis, що забезпечує низьку зв'язність (low coupling) та незалежне масштабування.



Рисунок 1 – Багаторівнева модель cloud-edge архітектури з використанням екосистеми AWS

Висновки. У роботі розроблено інформаційну технологію та метод ADPA для розподілених ITS: 1) Запропоновано евристичний алгоритм класифікації та динамічного мікропакетування з гістерезисом та складністю O(1); 2) Розроблено математичну модель оцінювання навантаження на основі мультикритеріального індексу LF(t) та контуру зворотного зв'язку за P_{95}-затримкою; 3) Спроектовано комплексну архітектуру cloud-edge на сервісах AWS (IoT Greengrass, Kinesis, EMR/Spark, Lambda, EKS, Redshift), яка забезпечує відмовостійку подієво-орієнтовану обробку даних великих обсягів.

Використання інструментів ШІ. При підготовці тексту тез доповідей використовувалася модель штучного інтелекту OpenAI GPT-4o. ШІ застосовувався для мовностилістичного редагування тексту та перекладу англомовного блоку. Усі математичні залежності, алгоритми та архітектурні рішення розроблені безпосередньо авторами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Ferdowsi, A., Challita, U., & Saad, W. (2017). Deep learning for reliable mobile edge analytics in intelligent transportation systems. arXiv preprint, arXiv:1712.04135. https://doi.org/10.48550/arXiv.1712.04135

[2] Mnyone, L. N., & Amin, N. (2026). Distributed big data architecture for smart transportation using Hadoop, Kafka, and Apache Flink. Preprints, 202603.1269.v1. https://doi.org/10.20944/preprints202603.1269.v1

[3] Fé, I., Matos, R., Dantas, J., Melo, C., Nguyen, T. A., Min, D., Choi, E., Silva, F. A., & Maciel, P. R. M. (2022). Performance-cost trade-off in auto-scaling mechanisms for cloud computing. Sensors, 22(3), 1221. https://doi.org/10.3390/s22031221

Terzi D. D.[1], Voloshchuk L. A.[2]

Cloud Architecture for Distributed Transport Data Processing Based on the Adaptive ADPA Algorithm

Abstract. *An information technology and an adaptive data processing algorithm (Adaptive Data Processing Algorithm, ADPA) for heterogeneous data streams in intelligent transport systems (ITS) are developed. A formal event classification rule, an adaptive micro-batching window formula with hysteresis, and a multi-criteria load factor with P95-latency feedback control are presented. A multi-tier cloud-edge architecture based on AWS services is designed, enabling flexible combination of stream, micro-batch, and batch processing.*

Keywords: *distributed information systems; cloud architecture; intelligent transport systems; adaptive data processing; micro-batching; AWS services.*